

Statistical Methods for the Processing of Communications Data

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

Timothy Charles Clapp

December, 2000



**UNIVERSITY OF
CAMBRIDGE**

SIGNAL PROCESSING GROUP
Department of Engineering
St. John's College



Copyright © 2000 Tim Clapp.

Submitted September 2000.

Reprinted with minor corrections, December 2000.

Format designed by the author, based upon the format of theses by Alex Stark and Paul Troughton and typeset by \LaTeX and $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{\LaTeX}$.

To my mother

Declaration

I hereby certify that, except as indicated in the text, the work described in this dissertation is entirely original. It is not the result of work done in collaboration, and has not been submitted to any other University.

This thesis contains 91 figures, 6 tables and approximately 47,280 words.

TIM CLAPP

Key Words

The following keywords may be useful for indexing purposes:

Bayesian statistics, blind deconvolution, blind equalisation, Gibbs Sampler, Markov chain Monte Carlo, sequential Bayesian methods, particle filter.

Acknowledgements

I would like to thank my supervisor, Simon Godsill, for his advice, direction and genuine interest in my work. I also wish to thank my team of proof-readers Deirdre O'Brien, Arthur Gretton, Nick Haan, Simon Hill, James Hopgood, Matthew Orton, Tanya Reeves and Maurice Ringer for their invaluable work and contrasting opinions literary style, and Alistair Cain and Paul Walmsley for looking after most of the computer systems. Finally I would like to thank the free software community for Linux and GNU tools, the combination of which saw my comfortably though my research, often much faster than under an alternative commercial operating system.

This work was funded, in the most part, by the Engineering & Physical Sciences Research Council, with additional support from St. John's College and the Department of Engineering.

Summary

This thesis describes the use of methods derived from Bayesian statistics on the problem of blind equalisation of communications channels, although much of this work is applicable to the more general problem of blind deconvolution. In order to allow general models to be incorporated, numerical methods are used; the focus is on *Markov chain Monte Carlo* (MCMC) methods for processing of blocks of data and the use of *particle filters* for sequential processing.

In order to obtain the best performance using MCMC, the choice of the Markov chain needs tailoring to the application in hand. The use of joint sampling of all the states (transmitted data sequence) and reversible jump moves to combat delay ambiguity are proposed.

The use of particle filters is still in its infancy, and much of the focus is on the development of improvement strategies to increase its applicability to real problems. It is well known that fixed-lag methods may be used to great effect on Markovian models where later observations can provide information about states in the recent past. Methods of performing fixed-lag simulation for incorporation into particle filters are described. The use of data windowing on fixed parameter systems allows regeneration of the parameters at each time-step without having excessive demands on storage requirements.

In certain cases it is difficult to perform the updating when a new data point is received in a single step. The novel concept of introducing intermediate densities in a manner akin to simulated annealing between time steps is described. This improves robustness and provides a natural method for initialisation.

All of these techniques are demonstrated in simulations based upon standard models of communications systems, along with favourable comparisons to more conventional techniques.

Contents

Notation	xiii
Abbreviations	xiv
1 Introduction	1
1.1 What is Signal Processing?	1
1.2 Bayesian Methods	2
1.3 Communications Systems	3
1.4 Outline	4
2 Time Series Models and Communications Systems	7
2.1 Time Series Models	7
2.1.1 ARMA Models and Linear Digital Filter Models	8
2.1.2 State-Space Model and Representation	11
2.1.3 Time-Varying Models	12
2.2 Communications System Model	12
2.3 Communications Channel Models	14
2.3.1 Digital Filter Channel Model	16
2.3.2 Types of Distortion	16
2.3.3 Fixed Channel Models	18
2.3.4 Time-varying Channels	19
3 Methods for Blind Deconvolution	22
3.1 Equalisation	23
3.1.1 Zero-Forcing Equalisers	23
3.1.2 Decision Feedback equalisers	24
3.1.3 Sequence Estimation	25

3.1.4	Fractionally Spaced Equalisers	26
3.1.5	GSM: An Example of a Modern Communications System	26
3.2	Adaptive Equalisation	28
3.2.1	Performance Measures	29
3.2.2	The Least Mean Square (LMS) Algorithm	30
3.2.3	Zero-forcing Algorithm	31
3.2.4	The Recursive Least Squares (RLS) Algorithm	31
3.2.5	Summary and Further Variants	32
3.3	Blind Equalisation	33
3.3.1	Parametric Approaches	33
3.3.2	Gradient Descent-Based Algorithms	34
3.3.3	High Order Moments and Cumulants	36
3.3.4	Cyclo-stationary Statistics	37
3.3.5	Neural Networks	37
3.3.6	ARMA Channel Models	37
3.3.7	Multiple Input	38
3.3.8	Direct Estimation of the Input Sequence	39
3.3.9	Statistical Methods	39
3.4	Overcoming Problems with Blind Equalisers	41
3.4.1	Poor Performance When Converged	41
3.4.2	Convergence Time	41
3.4.3	Model Selection	42
3.5	Summary	42
4	Bayesian Methods	43
4.1	Bayesian Data Analysis	43
4.1.1	Notation	44
4.1.2	Bayes Theorem	45
4.1.3	The Posterior Distribution	46
4.1.4	Prior Information	47
4.1.5	Marginalisation	49
4.1.6	Comparison to Conventional Methods	49
4.2	Numerical Methods	51
4.2.1	Monte Carlo Methods	52

4.2.2	Density Estimates	54
4.2.3	Random Numbers	55
4.3	Markov Chain Monte Carlo	55
4.3.1	Markov Chains	56
4.3.2	Markov Chain Monte Carlo algorithms	58
4.3.3	Practical Issues	61
4.4	Summary	63
5	MCMC Methods for Blind Equalisation	65
5.1	The Gibbs Sampler for Blind Equalisation	65
5.1.1	Model	66
5.1.2	Priors	67
5.1.3	The Symbol-Based Sampler	68
5.1.4	Problems Encountered	69
5.2	Joint Sampling	70
5.2.1	The Simulation Smoother	70
5.3	Overcoming Delay Ambiguity	72
5.3.1	Reversible Jumps	73
5.3.2	Reversible Jump Moves	75
5.4	Results	79
5.4.1	Implementational Issues	79
5.4.2	Convergence	83
5.4.3	Trained (Adaptive) Equalisation	88
5.4.4	Blind Equalisation	90
5.5	Discussion and Conclusions	95
5.5.1	Gibbs Sampler Moves	96
5.5.2	Comparison with Conventional Equalisers	98
6	Particle Filters	99
6.1	Particles and Updating Distributions	99
6.1.1	Filtering and Updating	100
6.1.2	Notation	101
6.1.3	Importance Sampling	101
6.1.4	Accuracy of Estimates	102

6.2	Particle Filtering Methods	102
6.2.1	Sequential Importance Sampling	102
6.2.2	Sampling – Importance Resampling (SIR) algorithm . . .	104
6.2.3	Resampling — When?, How? and Why?	105
6.2.4	Effective Sample Size	107
6.2.5	Rejection Control	109
6.3	Choice of Importance Functions	110
6.3.1	Prior Importance Function	110
6.3.2	Posterior Importance Function	111
6.3.3	Auxiliary Particle Filter	111
6.4	Incorporating Parameters	113
6.4.1	Time-Varying Parameters	113
6.4.2	Kernel Density Estimates	114
6.4.3	Marginalisation	114
6.4.4	Simulation	115
6.5	Summary	116
7	Particle Filters for Blind Equalisation	118
7.1	Particle Filters for Equalisation	119
7.1.1	Equalisation of a Time-Invariant Channel	120
7.1.2	Equalisation of a Time-Varying Channel	121
7.2	The Fixed-Lag Approach	122
7.2.1	One-Dimensional Non-Linear Example	122
7.3	Fixed-lag Methods using Particle Filters	124
7.3.1	Using the Filtering Density: The Decision Step	124
7.3.2	Using the Filtering Density with Look-Ahead	126
7.3.3	The Smoothing Density	128
7.4	Data Windowing	131
7.4.1	Motivation and Formulation	131
7.4.2	Window Shape	131
7.4.3	Window Size	132
7.5	Results from Simulations	133
7.5.1	Fixed-Lag Methods	134
7.5.2	Data Windowing	139

7.6	Conclusion	140
8	Bridging Densities in Particle Filters	143
8.1	Existing Techniques for Avoiding Degeneracy	144
8.1.1	Resample-Move Algorithm	145
8.2	Annealing in Particle Filters	145
8.2.1	Simulated Annealing	146
8.2.2	Practical Implementation	146
8.2.3	Cooling Schedules	148
8.2.4	Resampling at Intermediate Stages	149
8.2.5	Dynamic Step Size	149
8.2.6	Strategies for Choosing a Cooling Schedule	152
8.3	Results for Gaussian Distributions	154
8.3.1	Method	154
8.3.2	Simulated Annealing Without Resampling	155
8.3.3	Simulated Annealing With Resampling	158
8.3.4	Dynamic Step Size	159
8.4	Communications Channels	164
8.4.1	Generation	164
8.4.2	Analysis	164
8.4.3	Challenges Faced with Fading Channels	165
8.4.4	Model Mismatches on Fast Fading Channels	166
8.4.5	Initialisation on Slowly Fading Channels	170
8.4.6	Some Issues and Suggestions for Improvement	175
8.5	Conclusions	177
9	Conclusions and Further Work	179
9.1	Summary	179
9.2	Conclusions	181
9.2.1	Bayesian Statistics	181
9.2.2	Markov Chain Monte Carlo Moves for the Digital Communications System 182	
9.2.3	Particle Filters: Improvements and Application	184
9.3	Suggestions for Further Work	186
9.3.1	Bayesian Statistics	186

9.3.2	MCMC Methods for Blind Deconvolution	187
9.3.3	Particle Filters for Blind Deconvolution	188
9.4	Final Words	189
A	Fixed Channel Models	190
B	Results for Manipulations of Gaussian Distributions	193
B.1	Multiplication	193
B.2	Extensions for AR / MA models	194

List of Figures

2.1	General time-series model	8
2.2	Communications link model	13
2.3	Plot showing the zero and pole locations for the case of pure phase distortion	17
2.4	Plot showing the zero and pole locations for the case of pure amplitude distortion	18
2.5	Frequency response of channels	19
3.1	An FIR filter that could be used as a zero-forcing equaliser.	23
3.2	A decision-feedback equaliser	24
3.3	Block Diagram showing adaptive equaliser structure.	28
4.1	Histogram of some samples generated from a normal distribution overlaid with corre	
5.1	Model of Communications System Used.	66
5.2	Diagram showing how frames boundaries are dealt with.	81
5.3	Convergence results for channel A.1 (fixed model order).	84
5.4	Convergence results for channel A.2 (fixed model order).	86
5.5	Convergence results for channel A.3 (fixed model order).	87
5.6	Plot of bit error rate versus signal to noise ratio for trained algorithms on channel A.1.	
5.7	Plot of bit error rate versus signal to noise ratio for trained algorithms on channel A.3.	
5.8	Plot of bit error rate versus signal to noise ratio for blind algorithms with fixed model o	
5.9	Plot of bit error rate versus signal to noise ratio for blind algorithms with fixed model o	
5.10	Plot of bit error rate versus signal to noise ratio for blind Gibbs sampler algorithm with	
5.11	Plot of bit error rate versus signal to noise ratio for blind Gibbs sampler algorithm with	
7.1	Particle filter estimate of Posterior Mean using the filtering density and a fixed-lag of 3	
7.2	Graph of bit error rate against lag for two different channels: channel A.1 and channel .	
7.3	Graph of bit error rate against signal-to-noise ratio in simulations using different specif	
7.4	Graph of bit error rate against signal-to-noise ratio in simulations using the different SI	

-
- 7.5 Graph of bit error rate against signal-to-noise ratio in simulations using the different SI
 - 7.6 Graph of bit error rate against signal-to-noise ratio in simulations using the different SI
 - 7.7 Graph of bit error rate against signal-to-noise ratio in simulations using filtering density
 - 7.8 Graph of bit error rate against signal-to-noise ratio in simulations using filtering density

 - 8.1 Histograms of posterior densities obtained for an observation with a value of 3.155
 - 8.2 Histograms of posterior densities obtained for an observation with a value of 4.157
 - 8.3 Histograms of posterior densities obtained for an observation with a value of 4 using th
 - 8.4 Histograms of posterior densities obtained for an observation with a value of 8.159
 - 8.5 A Series of histogram plots showing the intermediate probability densities. Darker grey
 - 8.6 Results using a dynamic step size determined by the variance of the log weights having
 - 8.7 A Series of histogram plots showing the intermediate probability densities for 12 intern
 - 8.8 Results obtained for an observation with a value of 8. The step-size is determined dyna
 - 8.9 A Series of histogram plots showing the intermediate probability densities for 14 intern
 - 8.10 A demonstration of the particle filter tracking a time-varying channel generated from Ja
 - 8.11 Particle filter tracking with an over-diffuse prior, $\sigma_w = 0.8$ 169
 - 8.12 Particle filter tracking with an under-diffuse prior, $\sigma_w = 0.1$ 171
 - 8.13 Particle filter with annealing tracking with an under-dispersed prior, $\sigma_w = 0.1$.172
 - 8.14 Initialisation to a fading channel with no intermediate densities, $\sigma_p = 1$.174
 - 8.15 Initialisation to a fading channel using annealing, $\sigma_p = 10$ 175

 - A.1 Impulse response, frequency response and zero plot for channel A.1191
 - A.2 Impulse response, frequency response and zero plot for channel A.2191
 - A.3 Impulse response, frequency response and zero plot for channel A.3191
 - A.4 Impulse response, frequency response and zero plot for channel A.4192

List of Tables

8.1	Means and variances for an observed value of 3.	156
8.2	Means and variances for an observed value of 4.	156
8.3	Means and variances for an observed value of 4 when intermediate resampling is used.	
8.4	Means and variances for an observed value of 8 using the annealed importance sampler	
8.5	Lock rates and mean time to lock (if lock is achieved) for annealed and plain particle £	
A.1	Filter coefficients for the channels.	190

Notation

$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2 .
$IG(\alpha, \beta)$	Inverse Gamma distribution with parameters α and β . Since there is some ambiguity in the definition of what these parameters are, the functional form of the distribution is given in equation (5.5)
$E\{.\}$	Expectation of a function.
A	The matrix “A”.
A^T	The transpose of the matrix “A”.
x	The vector “x”.
$x[i]$	The i th element of vector x .
$[1, 2, 3]^T$	An enumerated vector.
x_t	Numerical subscripts denote time index.
$x^{(i)}$	Superscripts denote <i>iteration index</i> in the context of MCMC methods and <i>particle index</i> in the context of particle filters.
$x_{0:t}$	The set of states from time 0 to time t .
$p(.)$	Probability distribution.
$p(. ..)$	Conditional probability distribution.
$\pi(.)$	Stationary distribution of Markov chain (in chapters on MCMC methods) or importance function (chapters on particle filters).
$x \sim p(.)$	The vector x is drawn from the probability distribution, $p(.)$.

Abbreviations

ASIC	Application Specific Integrated Circuit
BPSK	Binary Phase Shift Keying
FPGA	Field Programmable Gate Array
FIR	Finite Impulse Response
GSM	Global System for Mobile communications
IIR	Infinite Impulse Response
MCMC	Markov Chain Monte Carlo
ML	Maximum Likelihood
MMSE	Minimum Mean Squared Error
LMS	Least Mean Squares
LS	Least Squares
p.d.f.	probability density function
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RLS	Recursive Least Squares
SIR	Sample-Importance Resampling
SIS	Sequential Importance Sampling
TDMA	Time Division Multiple Access

Introduction

1

1.1 What is Signal Processing?

At a fundamental level Signal Processing may be described as extracting the useful or interesting information from a signal or set of signals that we observe. For example if we are sending data from one place to another, we are only really interested in recovering the data that was sent from one end to the other. One could argue that everything else is supplementary and ultimately of no interest.

The next question to ask is “*How can one go about this?*” At the simplest level, one may think of some way of cleaning the data up so as to recover the signal we are interested in — for example the use of a simple filter. This may be sufficient in certain cases, but where more severe distortion is present we require a more systematic approach.

The first thing to do is to work out what we already know about the signal and how it is corrupted. This in itself may involve additional experiments or measurements. This knowledge may be regarded as *prior information*. We can now build a model of the system we are studying. Ultimately this model is the limiting factor in the performance of any algorithm for signal detection or estimation. If

the model is poor then we cannot hope to make great gains in performance by changing our method of fitting the data to the model. Essentially models come in two types: parametric and non-parametric. I shall concentrate on the former case. The model structure is such that a large amount of data can be characterised by a small set of parameters. Since it is usually easier to estimate these parameters than the whole data, the first stage in the process is parameter estimation. The data can then be recovered with the parameters now fixed.

1.2 Bayesian Methods

Bayesian methods allow us to attack the problem directly: we can recover the data in which we are interested, within the confines of our model and considering all possible parameter values. This is achieved by considering the observations, data and parameters to be random processes, each described by a probability distribution. The actual values of these variables are viewed as one particular realisation of these processes. We are not limited to estimation directly according to probability; we are free to choose any cost function we wish to optimise within this framework.

Bayesian signal processing is a very powerful and expanding research area. Unfortunately it is often not possible to perform the resulting multidimensional integrals analytically when probability distributions that are physically realistic are used (i.e. those resulting from a realistic model framework). However the exponential growth of computing power has led to an increased popularity of numerical methods. This thesis will focus on two such methods, namely *Markov chain Monte Carlo* methods and *particle filters*. The former is a general technique for simulating points from a general probability distribution by means of iteration over a dataset and has been in widespread use in the statistics community for a number of years. Particle filters are a new and expanding field of research because they have an important distinction: they allow the data to be processed *sequentially*. This provides us with a natural and practical way of processing the large amounts of data typical in many engineering applications.

1.3 Communications Systems

The very essence of human civilisation is based on our ability to communicate. Modern technology has simply expanded the scope of this communication so that it can take place at faster and faster rates between any two, possibly mobile, users. This thesis will deal with the particular problem of mobile communications, a popular area for both researchers and consumers alike.¹ More precisely the topic covered is that of *equalisation* in a digital communications system: recovering the original transmitted signal from the received signal which consists of a blur of past and present symbols caused by multiple paths of propagation as well as additional random noise. This is particularly important as the consumer places demands for higher data rates in harsh environments (for the communications engineer).

The technique of equalisation or *deconvolution* is of much general interest and is one of the fundamental problems of signal processing — where a signal is corrupted by shifted values of itself. When neither the signal nor the convolution function is known, the problem is said to be *blind*. There is a wealth of applications for the techniques developed other than just mobile communications. Some of them are listed below:

Underwater Communications Convolution is not unique to mobile communications over land. Underwater communications has its own set of problems: the channel varies due to temperature gradients in the water moving with currents. Despite the different frequency range and particulars, the underlying problem is very much the same.

Blurred Images Images are often blurred due to incorrect focus or imperfections in the lenses or other sensing equipment used. The resulting blur is a form of two-dimensional convolution and is detrimental since the true information content of the image is obscured. Deconvolution is often required as pre-processing and is supplementary to any further extraction of information within the image.

¹In 1999 the most popular Christmas present was a mobile telephone.

Astronomical Data The field of astronomy offers one of the best known examples of an error causing unwanted convolution. Imperfections in the mirror of the Hubble Space Telescope mirror caused the images to be blurred, requiring processing before the images could be used.

Astronomers are accustomed to the effects of convolution however. Although the lens of any telescope is unlikely to be perfect, the effect of the earth's atmosphere provides more of a problem. Telescopes are frequently sited to avoid the worst of atmospheric distortion but it cannot be completely avoided, so techniques are still being developed to reduce it (e.g. [96]).

Distorted Audio Many old recordings were made by transferring the sound vibrations onto a wax disk. The transducer for this – the horn – introduced much of the distortion that we associate with a “vintage recording”. This was one of the first applications for digital deconvolution [164].

Seismic Exploration The structure of the earth is often investigated by the use of shock waves (explosions) or other signals sent through the earth's crust. The reflections received at transmitters can tell us about the boundaries present, and hence one may propose a geological structure. Frequently the problem is posed slightly differently: we know the input and received signals and are interested in the channel filter. However the framework for attaining this can be very similar.

1.4 Outline

The remainder of this thesis will proceed as follows:

Chapter 2 provides an introduction to the modelling framework used for many time series models. In particular the focus will be on moving-average and autoregressive processes and the state-space form of these models. A typical communications system is described along with a discussion of various models for the channel over which the data is transmitted.

Chapter 3 describes the general problem of deconvolution and an overview of existing techniques for recovery of the signal of interest is given. This will focus on the equalisation methods used in communications systems, their strengths and weaknesses.

Chapter 4 introduces the Bayesian approach to processing data, and the advantages such an approach gives. Numerical methods are often required for models that are physically realistic — we are interested in drawing points from the posterior probability distribution, from which inference is made. Markov chain Monte Carlo methods, one of the popular numerical techniques currently used, will be described since this is used for the first part of the research.

Chapter 5 describes the use of Markov chain Monte Carlo methods for the blind equalisation problem. For effective practical use it is necessary to adapt the Markov chain in order to achieve fast convergence to the required probability distribution, and a good exploration of the space once convergence has been achieved. The familiar problems of burst errors and delay ambiguity are encountered, but these are largely overcome by the use of joint sampling and the novel technique of a “reversible jump” to directly overcome the delay ambiguity problem. It will be demonstrated that the combination of these techniques leads to a considerable speedup in the rate of convergence which would be necessary in practical systems. It will be shown that the performance of these methods is exceptionally good.

Chapter 6 introduces the idea of using *Particle Filters*. This exciting new field allows sequential processing of data. An overview of the underlying theory behind particle filters is given. This is important to understanding the operation of the particle filter and how one may best be applied to the problem.

Chapter 7 describes the use of particle filters for equalisation. Firstly the similarity of particle filters to more standard methods in the literature is discussed. Two techniques useful for many practical applications of particle filtering will be introduced: fixed-lag approaches and the use of a “window” of data to estimate fixed parameters. These methods are described and demonstrated on the commu-

nications channel.

Chapter 8 describes the use of MCMC moves within the particle filter and introduces the novel concept of intermediate bridging densities within the particle filter by a technique akin to simulated annealing. Demonstrations of this method using simple densities are presented first before their application to a more general model for a time-varying communications system.

Time Series Models and Communications Systems

2

In this chapter I shall introduce models to characterise the mapping from an input to an output over time. The system may represent a model for the signal production, a transmission or distortion medium, such as a communications channel or room acoustics. In the former case the output is the desired signal; the input may take the form of a noise process or an impulse train. In the latter, the input is the signal which we wish to recover, and the output usually the observation made of the signal. In addition we may have inverse systems, or a cascade of systems which model both signal production and distortion.

2.1 Time Series Models

We shall start with a general model with an input which shall be denoted by $u(t)$ and the observed output by $y(t)$. Vector notation will be used to indicate that there may be more than one input or output at each time. The i th element of this vector will be denoted by square brackets (e.g. $u[i]$). This discussion is restricted to *parametric* models, that is models characterised by a set of parameters θ which may vary the model without altering the framework. Since we are interested in

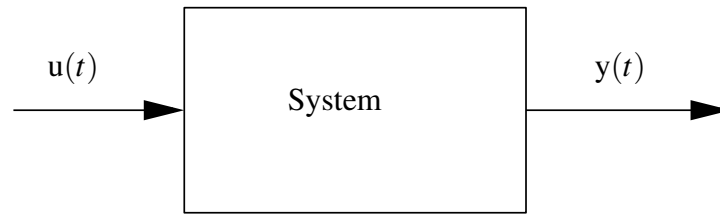


Figure 2.1. *General time-series model*

processing the data on digital machine with finite storage space (memory), we require quantisation of the values of the signal and sampling of the signal in time.

Quantisation: The quantisation or digitisation process is limited by what is commonly called the *word-length* — the number of bits allocated to store a single piece of data. We shall assume that this is sufficiently large that the effects of quantisation are negligible.

Sampling: In the sampling process the value of a signal, x , is selected as $x_k = x(kT)$, where x_k is the k th value of the discrete signal, $x(\cdot)$ is the continuous signal with T the sample period.¹ I shall denote a discrete signal from time 0 to time t with the notation: $x_{0:t}$.

2.1.1 ARMA Models and Linear Digital Filter Models

We wish to represent the system with a general-purpose model. A commonly used, general linear model for single input/single output is the auto-regressive moving average model:

$$y_t = \sum_{i=1}^{n_a} a[i] y_{t-i} + \sum_{j=0}^{n_b} b[j] u_{t-j} \quad (2.1)$$

¹Sampling in this manner introduces aliasing: each frequency above half the sampling rate is indistinguishable from a lower frequency. This effect is undesirable and can, and should, be removed by a low pass filter before sampling takes place.

By taking the Z-transform [125, chapter 4] we obtain:

$$Y(z) = \frac{B(z)}{1 - A(z)} U(z) \quad (2.2)$$

$$= \frac{\prod_{j=1}^{n_b} (z - z[j])}{\prod_{i=1}^{n_a} (z - p[i])} U(z) \quad (2.3)$$

where the sets of $\{z[j]\}$ and $\{p[i]\}$ are the zeros and poles, respectively.

We can easily see that the coefficients $a[i]$ lead directly to the location of the poles of the transfer function, and $b[i]$ are responsible for the zeros. These coefficients have real values, so poles and zeros either lie on the real axis, or as complex conjugate pairs.

An all-pole model is termed auto-regressive or a *infinite impulse response* (IIR) filter since it is recursive in nature. Strictly speaking, it still has zeros, however they all lie at the origin. This model is the most frequently used, and relatively easy to design or estimate parameters compared to the all-zeros model [167]. The AR model is suitable for modelling musical signals and speech with the excitation being either a normally distributed (Gaussian) excitation or an impulse train. This model is sometimes called *Linear Prediction*.

The all-zero model is called a moving average model or alternatively a finite impulse response filter. This model is commonly used in practice (for example in equalisers) since it is always stable (see section 2.1.1.1).

A combination or a cascade of these two models leads to a full ARMA model with both poles and (non-trivial) zeros. This model is the most powerful and would typically require fewer parameters to match the desired characteristics than either the AR or MA models alone.

2.1.1.1 Stability

A system is termed *stable* if a bounded input produces a bounded output. This is seen as necessary prerequisite for practical systems so as to obey the principle of conservation of energy. A necessary and sufficient condition for stability of a

causal system is that all the poles lie within the unit circle, i.e.

$$|p[i]| < 1, \text{ for all } i. \quad (2.4)$$

2.1.1.2 Frequency Response

The frequency response of the system may be calculated by substituting $z = e^{j\omega T}$ into the transfer function. If we use the form in equation (2.3) we get amplitude and phase of the frequency response to be:

$$|H(e^{j\omega T})| = \frac{\prod_{k=1}^{n_b} |e^{j\omega T} - z[k]|}{\prod_{i=1}^{n_a} |e^{j\omega T} - p[i]|} \quad (2.5)$$

$$\angle H(e^{j\omega T}) = \sum_{k=1}^{n_b} \angle(e^{j\omega T} - z[k]) - \sum_{i=1}^{n_a} \angle(e^{j\omega T} - p[i]) \quad (2.6)$$

where $H(z) = \frac{Y(z)}{X(z)}$.

It is clear from these equations that poles and zeros which will have greatest impact on the frequency response will lie near the unit circle.

2.1.1.3 Generalisation to Multiple Inputs and Outputs

The generalisation to multiple input/output models may take two forms:

- A set of parallel independent ARMA models. In this case for each output element, $y[i]$, there is a separate parameter set, a^i , and each output has a corresponding input, $u[i]$. This case is relatively easy to deal with, since the equations take a similar form as before.
- Correlated ARMA models. In this case each output is dependent on more than one input at the current time and previous times. It is also possible to have different numbers of inputs and outputs. Although the model remains linear, there can be a sharp increase in the number of parameters to account for the correlations.

2.1.2 State-Space Model and Representation

We have already seen that the models described can have alternative names. In addition, they can have different representations. It is often convenient to deal with different representations for different applications. The general state-space formulation, more popular in econometrics and control theory, may be written as:

$$\begin{cases} x_t = f(x_{t-1}, \theta, u_t) \\ y_t = g(x_t, \theta, v_t) \end{cases} \quad (2.7)$$

where x_t is the current state vector,

y_t is the observation,

θ is a vector of unknown parameters,

and v_t is observation noise.

The functions $f(\cdot)$ and $g(\cdot)$ represent the state evolution and observation respectively.

The key difference in this representation is that we have introduced the concept of *memory*, in the form of the *state*, directly. Since the state evolution equation is Markovian (that is it depends only on the previous state), the current state summarises all that has happened in the past.

2.1.2.1 The MA model

As an example, a state-space representation for the MA model is given below:

$$\begin{cases} x_t = \begin{bmatrix} 0^T & | & 0 \\ \hline \mathbf{I} & | & 0 \end{bmatrix} x_{t-1} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_t \\ y_t = \mathbf{b}^T x_t + v_t \end{cases} \quad (2.8)$$

where x is the state vector, of length n_b ,

\mathbf{I} is the identity matrix of size $(n_b - 1) \times (n_b - 1)$,

and \mathbf{T} denotes vector transpose.

For most models there is not a unique state-space representation: in particular there is an ambiguity in what should appear in the state equation and what should appear as part of the observation. In many cases a sensible choice of state is one which has physical significance.

2.1.3 Time-Varying Models

A time-varying model is one in which the parameters describing the model vary over time as well as the input and output signals. The general form of a time-varying model is given (in state-space form) as:

$$\begin{cases} \mathbf{x}_t = f(\mathbf{x}_{t-1}, \boldsymbol{\theta}_t, \mathbf{u}_t) \\ y_t = g(\mathbf{x}_t, \boldsymbol{\theta}_t, \mathbf{v}_t) \end{cases} \quad (2.9)$$

The only difference from equation (2.7) is that the parameters, $\boldsymbol{\theta}$, are now time-dependent. The time-domain filter representation follows in the obvious manner. For example the time-varying, single input/single output MA model may be written as:

$$\begin{aligned} y_t &= \sum_{j=0}^{n_b} b_t[j] u_{t-j} \\ &= \mathbf{b}_t^T \mathbf{u}_{t:n_b+1} \end{aligned} \quad (2.10)$$

The model structure remains the same over different times. In some cases the number of parameters may vary. For example this might occur in an MA model if one or more of the last parameters (high lags) became very small; it is then only a small step to eliminate them entirely (equivalent to fixing them to zero).

2.2 Communications System Model

A typical communications system is illustrated in Figure 2.2. This is adapted from the examples of Reader [142, chapter 2] and Proakis [136, chapter 1].

- **Source Encoding/Decoding** This encompasses any transformation of the data before coding for digital transmission takes place. The general aim is to represent the message in a succinct form and remove as much redundancy as possible. Examples include linear prediction coding of speech [167, section 1.3.4] or entropy coding. If the input is an analogue signal, sampling and digitising would be done in this step. The output from this stage will be a binary sequence.

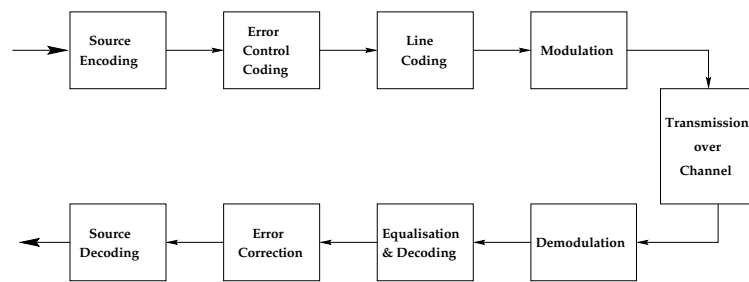


Figure 2.2. *Communications link model*

- Error Control Coding** Redundancy is introduced in the data in a deliberate and controlled fashion so as to make it possible to correct, or at least detect the existence of symbol errors at the receiver.
- Line Coding** This stage involves coding the amplitudes of the data to give desirable properties for transmission. Often a number of bits are encoded into one or more *symbols*. One technique is to introduce frequent symbol changes so that the symbol clock speed can be easily recovered in the receiver. Symbol schemes are usually symmetric and designed to keep transmitter power to a minimum, or sometimes constant. Differential coding is also sometimes used — data is coded as changes between symbols (or not). This allows the receiver to deal with the phase ambiguity that arises with symmetric schemes.
- Modulation/Demodulation** The symbols are converted into an analogue passband signal which is transmitted in accordance with some modulation scheme. The reverse occurs in the demodulator. Examples of modulation schemes and their demodulation are given in Proakis [136, chapters 4–5].

There is a blurred distinction between line coding and the modulation scheme. For example multilevel quadrature amplitude modulation (M-QAM) could be viewed as line coding such that symbols have M amplitudes and could be real or imaginary with a simple amplitude modulation scheme. For the purposes of this thesis, I will assume line coding is simple, if used at all, and the modulation scheme will represent both the amplitude/phase symbol constellation as well as the transformation from baseband to passband signal.

We are particularly interested in the equalisation and decoding section of the communications system. In particular we wish to recover the input sequence which is present before the distortion of the channel takes place.

Error correction codes go a long way to overcome the limitations encountered in practice from the technical design and the channel over which transmission takes place. Although error control coding nearly always improves the overall link performance, this does not mean that we should not design improved equalisers. Indeed, performance of error control codes is enhanced if “soft” decisions are made: i.e. a measure on how likely each symbol is rather than a hard decision of “this symbol is S_1 ”. The optimum is to use *a posteriori* probabilities [41], which supports the idea of a Bayesian equaliser.

2.3 Communications Channel Models

An ideal channel with a perfect flat response will exhibit behaviour represented in this equation:

$$r(t) = \sum_{k=0}^n s_k g(t - kT) + n(t) \quad (2.11)$$

where $r(t)$ is the observed signal,

$g(t)$ is the transmitted pulse shape,

s_k is the k th transmitted symbol,

T is the symbol period

and $n(t)$ is noise.

One of the important properties of a digital communications system is that the value of the symbol, s_k comes from a finite set,

$$s_k \in \{S_1, S_2, \dots, S_q\} \quad (2.12)$$

which is dependent upon the modulation scheme. For example for binary PAM or BPSK $q = 2$ and we could choose $S_1 = 1, S_2 = -1$. For QPSK, $q = 4$ with the possible symbol values taken from $\{1, -1, j, -j\}$.

A linear channel other than the perfect one above can be modelled with an almost identical equation. Both noise and pulse shape are modified by the channel response, but otherwise the equation remains of the same form:

$$r(t) = \sum_{k=0}^n s_k h(t - kT) + v(t) \quad (2.13)$$

where $h(t) = c(t) * g(t)$ is the modified transmitted pulse shape,

$v(t) = c(t) * n(t)$ is the modified noise,

$c(t)$ is the channel response,

and $*$ denotes convolution.

It is perhaps more meaningful if we consider the Fourier domain: the modified frequency responses are the product of the individual responses of signal and channel.

$$H(f) = C(f) G(f) \quad (2.14)$$

We may write

$$C(f) = |C(f)| e^{j\theta(f)} \quad (2.15)$$

within the bandwidth of the channel. $|C(f)|$ is the amplitude response and $\theta(f)$ the phase response. We may define the envelope delay characteristic as:

$$\tau(f) = -\frac{1}{2\pi} \frac{d\theta(f)}{df} \quad (2.16)$$

If $\tau(f)$ is not constant over the channel bandwidth we get delay distortion. This causes pulses to be spread out, resulting in *Inter-Symbol Interference* (ISI).

An optimal demodulator is in the form of a matched filter:

$$y_k = \int_0^T h(kT - t) r(t) dt \quad (2.17)$$

y_k is a discrete-time analogue signal which is a sufficient statistic for optimal decoding. Clearly if the modified transmitted pulse shape, $h(t)$, is not known, the optimal demodulator is not possible. In this case the transmitted pulse shape, $g(t)$ is usually used instead.

2.3.1 Digital Filter Channel Model

For the purposes of studying the discrete part of a communications system, a simpler model is often assumed to remove the analogue \leftrightarrow digital conversions. The most common model is that of a finite impulse response (FIR) filter with additive noise. This noise is often assumed to be Gaussian (appealing to Central Limit Theorem arguments), and it is easier to handle, though this is not always realistic. The model can be represented mathematically as:

$$y_t = \mathbf{h}_t^T \mathbf{u}_{t:t-(l-1)} + v_t \quad (2.18)$$

where y is the observed signal,

u is the transmitted signal,

h is the channel filter, of length l ,

and v is the noise.

The motivation for the FIR model is that in each received symbol there is also some contribution from a number of previous symbols, i.e. the inter-symbol interference is modelled directly.

The FIR model has a strong physical basis for a sampling receiver (i.e. the received signal is obtained from a simple sampling operation at the receiver). The filter taps represent the contribution from other symbols — the filter coefficients could therefore be complex to allow for phase differences.

Forney [39] showed that with the presence of the matched filter or other integrating demodulator operating on the continuous-time signal, the system may be reduced to this model with real coefficients. However the drawback with this model when used with a matched filter is the presence of correlation in the noise. In many models, noise is assumed to be independent and identically distributed (i.i.d.) and it is clear that this is not always the case.

2.3.2 Types of Distortion

2.3.2.1 Phase Distortion

Pure phase distortion is defined as signal distortion that includes no amplitude distortion or attenuation effects. In other words $\left| H(e^{j\omega T}) \right| = 1$ for all values of ω .

To achieve this we require a zero located at $\frac{1}{p[i]^*}$ as illustrated in Figure 2.3. This can be verified by substitution into equation (2.6).

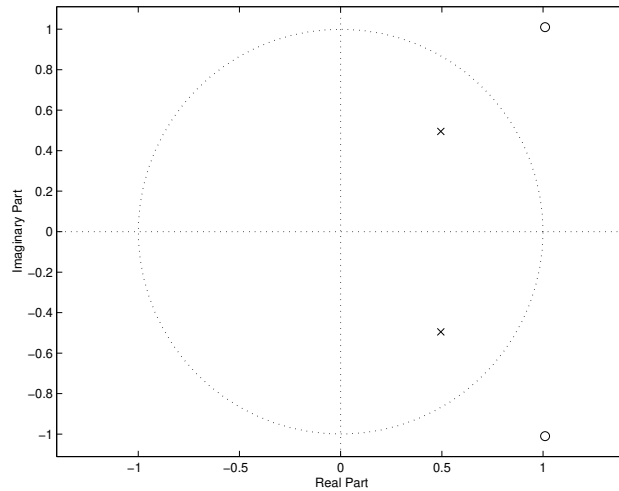


Figure 2.3. Plot showing the zero and pole locations for the case of pure phase distortion

2.3.2.2 Amplitude Distortion

Pure amplitude distortion occurs if the signal contains no phase distortion. By considering equation (2.5) we find that zeros appear in pairs: for each zero, $z[i]$ inside the unit circle there is a corresponding one located at $\frac{1}{p[i]^*}$. This can be seen in Figure 2.4.

2.3.2.3 Minimum Phase Systems

The magnitude frequency response does not uniquely characterise the system [125, section 5.4]. We have seen that for stability we require all the poles to lie within the unit circle. In certain classes of problem we also require the inverse filter to be stable. This would require all of the poles and the zeros to lie within the unit circle. Such a system is called *minimum phase*.

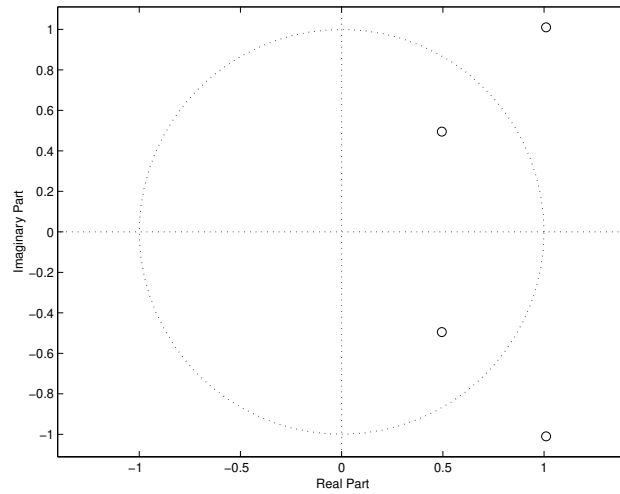


Figure 2.4. Plot showing the zero and pole locations for the case of pure amplitude distortion

2.3.3 Fixed Channel Models

In this section a few “difficult” channels, which do not vary over time, will be introduced. These channels illustrate examples of practical problems of one sort or another. The actual channels used in the simulations in later chapters are described in appendix A.

As a starting point, the reader is referred to Proakis [136, chapter 10]. He describes 3 channels (Fig. 10-2-5). Their frequency responses are plotted in Figure 2.5.

- (a) This is typical of the response of a good quality telephone channel, with $\mathbf{h} = [0.04, -0.05, 0.07, -0.21, -0.5, 0.72, 0.36, 0, 0.21, 0.03, 0.07]^T$
- (b) This is a short (length 3) channel with $\mathbf{h} = [1, 2, 1]^T / \sqrt{6}$. It has a frequency null at high frequency. In the time-domain, it means that input signals of the form $1, -1, 1, -1, 1, \dots$ give no output.
- (c) This channel, $\mathbf{h} = [1, 2, 3, 2, 1]^T / \sqrt{19}$, again has a frequency null; this time at a lower frequency.

It is known that linear equalisers do not perform at all well on channels (b) and (c) [136].

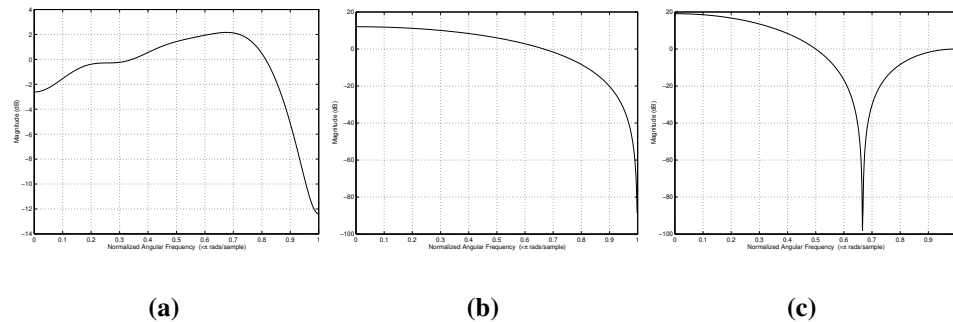


Figure 2.5. Frequency response of channels

2.3.4 Time-varying Channels

The mostly likely cause of ISI in a mobile communications channel (or any radio channel) is *multi-path propagation*. Propagation of the signal, other than the direct line-of-sight radiation takes place by the following [55]:

- **Reflection** off flat surfaces such as buildings, the ground.
- **Diffraction** — bending of the transmission path when it is obstructed by objects with sharp edges.
- **Scattering** when the medium consists of objects whose dimensions are small compared to the wavelength of the signal.

As a result, for a given communications link, there is more than one path over which transmission can occur. This leads to a variety of effects:

- **Time delay spread** caused by different path lengths and therefore different arrival times.
- **Doppler Spread** caused by different multi-path components having different Doppler shifts. This may happen even if the receiver and transmitter are stationary because of moving objects along the paths.
- **Random phase and amplitude** of multi-path components caused by different attenuation for each path.

As transmitter or receiver move, or the channel geometry changes due to other movement, the sum of these multi-path components may change markedly in phase and amplitude, a behaviour known as *fading*.

2.3.4.1 Narrowband signal models

First let us consider what happens to a narrowband signal such as a sinusoid. If we assume that there is uncorrelated scattering taking place and there are a large number of these scattered signals arriving at the receiver, we get the model proposed by Clarke [27]. The addition of these phasors with uniformly distributed phase and normally distributed amplitude (from the Central Limit Theorem) will result in a Rayleigh distribution for the magnitude of the complex signal received. Clarke showed that this model was appropriate for urban environments such as New York City. In the presence of a direct, line-of-sight path, a Rician distribution would result. The spectrum of this signal, called the *Doppler spectrum*, $D(f)$ can be shown to be [127]:

$$D(f) = \frac{1}{2\pi f_m \sqrt{1 - \left(\frac{f}{f_m}\right)^2}} \quad (2.19)$$

where f_m is the maximum Doppler shift.

Measurements of real channels justify Clarke's assumptions that result in a Rayleigh distribution, although the sophisticated models, leading to Weibull or log-normal distributions, often fit the data better [127].

The two most common methods to simulate data are:

Jakes' model The Rayleigh fading process may be simulated by summing a set of complex sinusoids [87]. This method is commonly used in simulating mobile communications channels. It relies on the fact that the Doppler spectrum given in equation (2.19) can be approximated by a small number of complex sinusoids, with accuracy increasing as the number of sinusoids increases.

Filtered Gaussian Noise An alternative popular model for simulation of the mobile communications channel is to filter Gaussian noise [3]. The filter has the same spectral shape as the Doppler spectrum. It is difficult to use in analysis

if the cut-off frequency is not known, as a different filter needs to be designed for each frequency. Implementation for simulation in DSP hardware is relatively straightforward, since the Doppler cut-off frequency may be adjusted by altering the clock speed of the discrete processing since the output is analogue [5, 67, 124].

2.3.4.2 Wideband signal models

If a sinusoid model is the ideal narrowband signal, then the equivalent for a wideband signal is an impulse. The wideband characteristics therefore cover the impulse response. The usual model is a sum of a small number of uncorrelated paths, which may be modelled as a sum of signals arriving at irregular time intervals.

Modern standards committees suggest a sum of Rayleigh fading paths spaced with irregular time intervals, and give examples for typical relative amplitudes and delays [72].

Methods for Blind Deconvolution

3

Consider a linear time-invariant system as Figure 2.1 in the discrete time domain. The output signal y_t is defined as the *convolution* of the input signal, u_t , and the impulse response, h_t , as given by:

$$y_t = \sum_{k=0}^{l-1} h_k u_{k-t} \quad (3.1)$$

The technique of *deconvolution* refers to the task of recovering the input signal (or equivalently the impulse response). It can be applied to far-ranging problems such as recovery of communications signals, restoration of blurred images, seismic exploration and general system identification.

The output is generally known. If neither of the input or convolution functions are known, the deconvolution process is much more difficult and is said to be *blind*.

3.1 Equalisation

In communications systems the term *equalisation* is used to describe the deconvolution process. The only distinction that needs to be made is that communications systems generally require on-line methods: it is not possible to spend vast amounts of computational resources some time after the data has been received.

The idea behind equalisers is to reverse automatically the effects of the channel, thereby restoring the original signal [112]. Most equalisers can be described as one of the following.

3.1.1 Zero-Forcing Equalisers

These equalisers eliminate ISI by a linear filtering operation. For a channel with an infinite impulse response (IIR), an FIR filter is required (so that the poles and zeros may cancel each other out). Conversely for an FIR channel, an IIR filter is required. In practice this is often approximated by FIR filter to ensure numerical stability, or to reduce complexity. However, since the output of the filter is a weighted sum of a part of the received data, the noise may be amplified. An example of an FIR zero-forcing equaliser is shown in Figure 3.1.

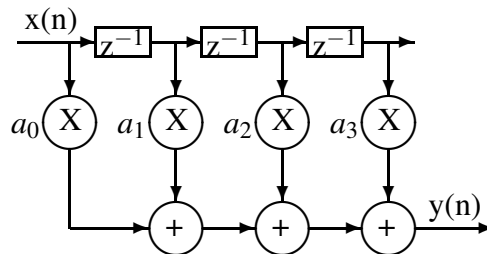


Figure 3.1. An FIR filter that could be used as a zero-forcing equaliser.

Zero-forcing equalisers have a major drawback because in the process of eliminating ISI, the system noise is amplified. This may be reduced by the use of the Wiener filter [167, §7.3–7.4] which reduces the gain of the filter according to the relative signal and noise powers across the spectrum.

3.1.2 Decision Feedback equalisers

Noise amplification can be eliminated by making a hard decision on the previous data to give the estimated symbols. The feedback uses these estimates — which are noise free. The principle can be seen in figure 3.2.

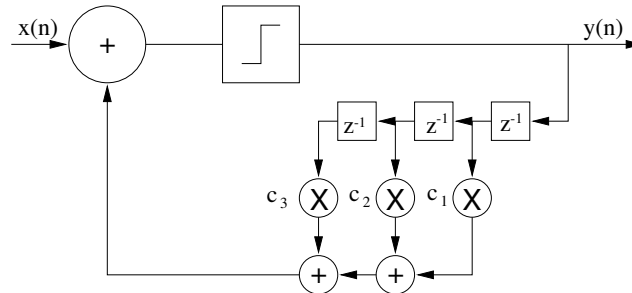


Figure 3.2. A decision-feedback equaliser

Most decision feedback equalisers used in practice have a linear transversal filter (i.e. of the same form of the zero-forcing equalisers described above) preceding the feedback part illustrated above.

Problems arise if the estimates are incorrect because the wrong feedback is given, increasing the probability of error in the next symbol. If this is in error too, incorrect feedback again results. Therefore errors usually occur in bursts.

Decision feedback equalisers are also problematic if the signal strength varies; the filter taps will need to be scaled as well. In practice many receivers have automatic gain control, so the received signal is boosted or reduced accordingly to keep an approximate constant power.

The performance of a decision feedback equaliser for a time-varying channel was discussed by Wood *et al.* [188]. Comparisons between a continuous update of the filter taps C_i and a block updating scheme are also given. It was found that the performance using block updating was not significantly degraded so long as the update rate was fast compared to the channel variation.

3.1.3 Sequence Estimation

To further improve equaliser performance, it became clear that rather than estimating the symbols individually, the estimate should be in context of the surrounding bits. In other words we are interested in selecting the best *sequence*. However the number of possible sequences increases exponentially with data length.

In 1967 Viterbi [181] described an algorithm for decoding a convolutional error correction scheme which is asymptotically optimal, given knowledge of the channel. It was soon clear to researchers such as Forney [39] that this algorithm was applicable to any kind of deconvolution problem including equalisation. In fact he and others had recognised that the introduction of known ISI increases error resilience i.e. it behaves like a convolutional code. It seems plausible therefore to suggest that if we can model the channel accurately, then performance could be better than on a channel with no ISI.

Forney not only advocated the use of the Viterbi algorithm, but also noted that the algorithm may diverge and converge again resulting in error bursts [40]. In addition, he observed that its convergence was independent of the initial conditions.

To overcome ambiguities with initial and final states, a couple of control symbols are usually added to the beginning and end of the data stream. The initial and final states are then fixed and a single path will result from the Viterbi algorithm.

It is not uncommon to have concatenated Viterbi decoders in communications systems: one decoder for equalisation and one for decoding convolutional error correction codes. Nill and Sundberg [122] go some way to feeding the reliability of the results in the first decoder to the second, while stopping short of a fully Bayesian approach. Their methods use lists of possible symbol streams with a soft decision being made on which is likely to be correct.

It is possible to join the two decoders together combining ISI reduction and error correction. However this is not usually done since the number of states the decoder can be in is much larger, resulting in greater receiver complexity. If computing power is then insufficient to implement this in real time, a reduced state Viterbi algorithm would be required (such as those in [8]). It is therefore unlikely to get a decrease in error rate overall.

Holubowicz and Morales-Moreno [84] noted that the number of states in a

combined decoder (S_d) is given by

$$S_d \leq S_c S_e \quad (3.2)$$

where S_c and S_e are the number of states in the channel and the convolutional code respectively. The inequality arises because not all $S_c S_e$ can be reached because the initial state is fixed. This can reduce the number of states in the decoder S_d by as much as a factor of two or four, with a corresponding reduction in receiver complexity.

3.1.4 Fractionally Spaced Equalisers

If the channel-distorted transmitted pulse shape is known, then a matched filter is the optimal decoder. In the case of time-varying or other unknown channels, the distortion introduced by the channel cannot be compensated for. Usually the filter is matched to the transmitted pulse shape which is notably sub-optimal. An alternative is to simply sample the channel, removing the matched filter entirely.

Considerable advantages can be obtained by sampling at a rate higher than the symbol rate. Indeed a rate only slightly faster, for example $\frac{4}{3}$ of the symbol rate can reap rewards [180]. Both Quereschi and Forney [138] and Gitlin and Weinstein [59] obtained substantial performance gains in the presence of certain types of interference.

Fractionally spaced equalisers are considerably less sensitive to timing errors, and channel tracking is improved. Not all researchers were quite so optimistic — although Reader found similar performance advantages, he concluded the gains did not outweigh the increase in complexity / computation time required [142].

3.1.5 GSM: An Example of a Modern Communications System

The Global System for Mobile Communications (GSM) provides European countries with a common digital mobile radio structure for overland communications. It is structured as follows [8]:

Time Division Multiple Access (TDMA) is employed to cope with simultaneous users. The TDMA frame of $4.615ms$ is split into eight user slots. Data

transmission is carried out with Gaussian minimum shift keying modulation, one of the continuous phase modulation schemes with a differential type precoding.

Eight-way interleaving is used to counteract burst noise, and two of the eight blocks are transmitted in each TDMA slot. The TDMA slot of $0.577ms$, 148 bits, is organised as follows:

- 3 fixed bits
- 57 bits of data (1 block)
- 1 signalling bit
- 26 bits midamble (training sequence)
- 1 signalling bit
- 57 bits of data (the other block)
- 3 fixed bits

The midamble is used to train the adaptive equaliser. The 3 fixed bits at either end allow the initial or final state of the sequence estimator (which would usually use the Viterbi algorithm, or possibly a reduced state/complexity version) to be known.

The data itself is divided into two classes (well, three really!). Class II bits (30%) have no error control coding. Class I bits (70%) have 2:1 convolutional coding. In addition, just over a quarter of these bits are Class Ia bits, which are coded with a (53,50) parity check code before the convolutional coding. If there is a parity error, the whole data burst (8 interleaved blocks / 4 TDMA frames) is discarded.

In mobile communications, the channel is constantly changing due to the movement of one or both of the transmitters/receivers. Adaptive equalisers have been developed to track the channel and keep ISI low. However the environment is such that one path may be eliminated either by destructive interference [86] or simply because of obstructions (e.g. buildings). It may not be possible to send another known sequence to train the equaliser for immediate recovery.

3.2 Adaptive Equalisation

With simple equalisers the filter tap coefficients are set beforehand according to some error criterion based upon a known or measured channel response. Adaptive equalisers use algorithms for automatically adjusting the filter tap coefficients to optimise a specified performance measure and to compensate for time variations in the channel.

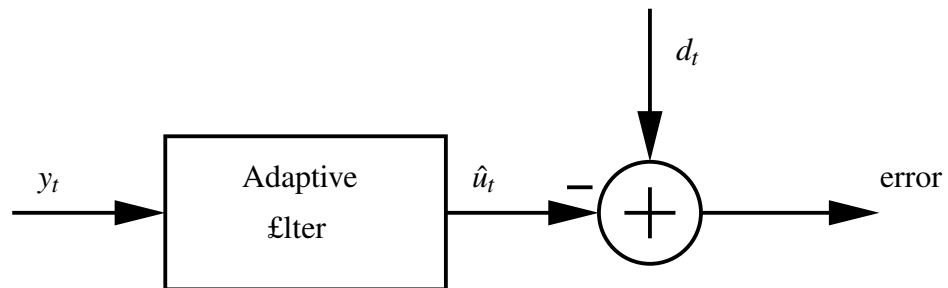


Figure 3.3. Block Diagram showing adaptive equaliser structure.

Consider the diagram in Figure 3.3. The received signal, y_t , is filtered to obtain an estimate of the transmitted signal, \hat{u}_t . The error between this and the desired signal, d_t , is then obtained. This error is used for adaptation of the equaliser.

The basic adaptive equaliser uses a training sequence to set the filter taps every so often. When the equaliser is being trained, the desired signal is set to the known transmitted signal: $d_t = u_t$. The training data may be placed at the beginning of the frame (preamble) so that the equaliser can be trained initially and then process all the successive data. For rapidly time varying channels, if a degree of latency can be tolerated, the training sequence is usually sent in the middle of the frame (midamble) so that the best approximation of the channel over the whole frame may be made.

If the equaliser has been trained and the error probability is low, the equaliser may be run in *decision-directed* mode where $d_t = \tilde{u}_t$, where \tilde{u}_t is the symbol closest to \hat{u}_t . This allows tracking to continue without the need for further training sequences. However, wrong decisions lead to the algorithm adapting in the wrong manner, and the equaliser can often lose convergence if there are too many errors.

This usually happens during a channel fade, since the SNR is severely reduced. To recover without a training sequence is said to be *blind equalisation*.

I shall assume that the adaptive filter is a linear FIR filter with filter taps $c[i]$ for $i = 0 \dots n_c$ i.e. a filter of order n_c . This has been preferred historically due to simplicity of implementation, guaranteed stability and less susceptibility to rounding errors. Extensions exist for decision feedback equalisers and for maximum likelihood sequence estimation [136, §11].

3.2.1 Performance Measures

Every adaptive equaliser requires some method of assessing performance, so this measure can be optimised. In many systems we would like to minimise Bit Error Rate (BER) but this is usually too hard, so other more tractable measures are used. Two criteria have found widespread use in optimising for the equaliser coefficients: mean squared error and peak distortion.

3.2.1.1 Mean Squared Error (MSE) Criterion

The MSE criterion, denoted by J , is given by the mean square of the error between the adaptive equaliser's output and the desired response:

$$J = E\{|d_t - \hat{u}_t|^2\} \quad (3.3)$$

The minimisation of this leads to the well-known Wiener filter, the tap weights determined by the Wiener-Hopf equations:

$$c_{opt} = R^{-1}p \quad (3.4)$$

$$\mathbf{R} = \begin{bmatrix} r_{yy}(0) & r_{yy}(1) & \dots & r_{yy}(n_c) \\ r_{yy}(1) & r_{yy}(0) & \dots & r_{yy}(n_c - 1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}(n_c) & r_{yy}(n_c - 1) & \dots & r_{yy}(0) \end{bmatrix} \quad (3.5)$$

$$= \text{autocorrelation matrix of received signal}, \quad (3.6)$$

$$\mathbf{p} = [r_{yd}(0), r_{yd}(1), \dots, r_{yd}(n_c)]^{\mathbf{T}} \quad (3.7)$$

$$= \text{cross-correlation of received and desired signals}, \quad (3.8)$$

$$r_{yy}(t) = E\{y(\tau)y(t - \tau)\}, \quad (3.9)$$

$$r_{yd}(t) = E\{y(\tau)d(t - \tau)\}, \quad (3.10)$$

$$E\{\cdot\} = \text{the expectation of a function}. \quad (3.11)$$

The solution to the equation above is given by the pseudo-inverse:

$$\mathbf{c}_t = [y_{t:t-n_c} \mathbf{y}_{t:t-n_c}^{\mathbf{T}}]^{-1} y_{t:t-n_c} d_t \quad (3.12)$$

I will label this the Least Squares (LS) solution. However this is frequently regarded as too complex to compute directly, so an iterative approach is used.

3.2.1.2 Peak Distortion Criterion

Peak Distortion is defined as the worst-case inter-symbol interference at the output of the equaliser. If the equaliser has an infinite length, this corresponds to a zero-forcing equaliser; for finite lengths there is residual ISI.

This was shown to be a convex function of the equaliser coefficients (i.e. it has a global minimum and no relative minimum) [112]. A solution may therefore be found using gradient descent.

3.2.2 The Least Mean Square (LMS) Algorithm

Minimisation of J using the steepest descent method would require knowledge of \mathbf{R} and \mathbf{p} which are not known in general, and may vary in time. The LMS algorithm [187] is an approximation to the steepest descent method substituting the expected value of the squared error with its instantaneous value. This leads to

the update equation:

$$\mathbf{c}_{t+1} = \mathbf{c}_t + \mu(d_t - \hat{u}_t)y_{t:t-n_c} \quad (3.13)$$

where μ is a step-size parameter. The algorithm converges for $0 < \mu < \frac{2}{\lambda_{max}}$ where λ_{max} is the value of the largest eigenvalue of \mathbf{R} . Convergence rate is determined by the ratio of largest to the smallest eigenvalues. This algorithm is an exact steepest descent method when the received signal is white noise, i.e. $\mathbf{R} = \sigma^2\mathbf{I}$, where the optimal step size is $\mu = \frac{1}{\sigma^2} = \frac{1}{r_{yy}(0)}$.

Normalised Least Squares We can see from above that step-size and therefore convergence is dependent on the signal power. To remove this limitation, the Normalised LMS (NLMS) algorithm has a slightly modified update:

$$\mathbf{c}_{t+1} = \mathbf{c}_t + \frac{\mu}{y_{t:t-n_c}^T y_{t:t-n_c}} (d_t - \hat{u}_t)y_{t:t-n_c} \quad (3.14)$$

3.2.3 Zero-forcing Algorithm

The zero-forcing algorithm is similar in the LMS algorithm described above using the peak distortion criterion. The aim is to eliminate cross-correlation between the error sequence and the desired signal, d_t . By approximating the ensemble average cross-correlation as instantaneous cross-correlation, we get the update equation:

$$\mathbf{c}_{t+1}[j] = \mathbf{c}_t[j] + \mu(d_t - \hat{u}_t)d_{t-j}^* \quad (3.15)$$

3.2.4 The Recursive Least Squares (RLS) Algorithm

If y_t and therefore \hat{u}_t are considered to be deterministic observed sequences instead of random processes, we may use the time-average of the squared errors in our optimisation process rather than the expectation. A weighting or forgetting factor, λ ; $0 < \lambda < 1$, is introduced to place more weight on recent observations. This allows the algorithm to track non-stationarities. We then obtain the cost function:

$$J(t) = \sum_{j=0}^t \lambda^{t-j} |e_{j|\mathbf{c}_t}|^2 \quad (3.16)$$

where $e_{j|\mathbf{c}_t} = d_j - y_{j:j-n_c}^T \mathbf{c}_t$.

The value of \mathbf{R}^{-1} is estimated recursively using the new data as it arrives. This is done in a computationally efficient manner using the matrix inversion lemma¹. The resulting updates are:

$$\mathbf{R}_t^{-1} = \frac{1}{\lambda} \left(\mathbf{R}_{t-1}^{-1} - \frac{\mathbf{R}_{t-1}^{-1} \mathbf{y}_{t:t-n_c} \mathbf{y}_{t:t-n_c}^T \mathbf{R}_{t-1}^{-1}}{\lambda + \mathbf{y}_{t:t-n_c}^T \mathbf{R}_{t-1}^{-1} \mathbf{y}_{t:t-n_c}} \right) \quad (3.17)$$

$$\mathbf{c}_{t+1} = \mathbf{c}_t + \mathbf{R}_t^{-1} (d_t - \hat{u}_t) \mathbf{y}_{t:t-n_c} \quad (3.18)$$

3.2.5 Summary and Further Variants

The algorithms described above are designed to converge to a time invariant channel. They also work for a time varying channel if the time variation is slow relative to the data rate. The LMS algorithm is simple to implement and has a very low computational cost. However even in the NLMS form it is often slow to converge. The RLS algorithm converges much faster (at a computational cost) although its structure means it may not track rapid time variations as well as LMS.

An example of how adaptive equalisation may be employed in GSM is given by Del Re *et al.* [32]. The LMS algorithm is used for equaliser training and the Viterbi algorithm is run forwards and backwards from the midamble training sequence.

More recent approaches have used per-survivor processing (PSP) [140]. This is a combination of sequence estimation and adaptation. Adaptation is performed using the data from each survivor path in the Viterbi algorithm (or any other sequence estimation algorithm). That is to say that there are separate channel estimates for each surviving path through the trellis.

Adaptive IIR filtering is a relatively new development, introducing new problems such as stability monitoring. Considerable progress has been made for use in adaptive echo cancellation and equalisation [101, 147].

Tsatsanis *et al.* [175] treat the channel as a time-varying AR process. The AR coefficients are estimated from autocorrelation coefficients, and the channel tracked by a Kalman filter. This method allows accurate tracking of fading channels.

¹if $\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T$ then $\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}[\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C}]^{-1}\mathbf{C}^T\mathbf{B}$

3.3 *Blind Equalisation*

The idea of a blind equaliser was first introduced by Sato [154] as a means of recovering an adaptive equaliser after a channel fade. Like the adaptive equalisers, the aim is to determine the channel response so as to reduce or eliminate ISI by inverse filtering.

By removing the need for training sequence, blind equalisers offer other benefits to the system.

- The most obvious benefit is that the bit rate can be increased since data can be transmitted in place of the training sequence. This can be quite significant for systems such as GSM.
- Adaptive equalisers cannot recover after a fade, so all the data is lost until the next training sequence. If a blind equaliser can converge before the next training sequence is sent, then not all the data will be lost.
- In a broadcast network, that is one where a transmitter may broadcast to many receivers, if one receiver is unable to track the channel correctly, it must wait for the next training sequence — losing all the data transmitted while it is waiting. An alternative strategy is to allow it to “ask” for a new training sequence. However if this happens too often, the data bit rate may be seriously reduced.

3.3.1 **Parametric Approaches**

Parametric approaches involve estimation of the channel before deconvolution takes place. A channel model is usually chosen based upon any knowledge of the system. The vast majority of algorithms take this form and use the FIR filter channel model described in section 2.3.1.

The idea of these algorithms are to minimise some measure of the error in fitting the model to the data. Tugnait [178] notes that the parametric approaches generally use one of three error criteria:

- **Inverse Filter Error.** The error measured between the actual signal and the

equalised signal. When the algorithm is running blind, a decision-directed approach is required. That is, since the actual signal is unknown, the signal is estimated by one or more decision boundaries.

- **Fitting Error.** The error between the statistics of the model and the estimated statistics of the signal.
- **Equation Error.** If some equation is satisfied ideally (e.g. the Yule-Walker equation in the case of ARMA modelling), then it is possible to minimise the value of the evaluated equation.

He goes on to state that only in the latter case is a global extremum guaranteed. In other cases the algorithm may get stuck in a local extremum. It comes as no surprise then that in the majority of cases, this technique is used.

Initially most algorithms filtered the received data to recover the signal (i.e. the inverse filter was sought). With the increase of computational complexity available, the channel forward filter is sought and the signal recovered by a Viterbi equaliser or similar structure. This has the advantages (for the FIR channel model) of automatic stability and causality.

Of course, using a parametric model brings other issues into play, such as the selection of model order. Gozdo and Anderson investigated the effects of over-fitting models and an incorrect noise power estimate for linear and decision feedback equalisers, both theoretically and by simulations [68]. They found that complex models often degraded performance (the wrong channel was modelled). IIR filters were worse than FIR filter in this respect, especially at high signal:noise ratios. Performance can be severely degraded in the case of noise mismatch — particularly if the channel has spectral nulls.

3.3.2 Gradient Descent-Based Algorithms

Gradient-Descent algorithms are based upon the LMS algorithm with an update based upon a memoryless non-linearity acting on the output of a linear FIR equalisation filter. These algorithms are also known as *Bussgang* techniques because the statistics of the deconvolved signal are approximately Bussgang. A review of

these methods may be found in [7] with analysis of their convergence in [33].

$$\hat{u}_t = y_{t:t-l+1} \mathbf{c}_t \quad (3.19)$$

$$\mathbf{c}_{t+1} = \mathbf{c}_t + \mu (g(\hat{u}_t) - \hat{u}_t) y_{t:t-l+1} \quad (3.20)$$

where \mathbf{c}_t are the inverse filter coefficients,

μ is the step-size parameter,

$g(\cdot)$ is the function to be minimised,

and l is the length of the channel.

The pioneering work of Sato [154] concentrated on one dimensional pulse amplitude modulation (PAM). He used the cost function:

$$g(\hat{u}_t) = \frac{E\{[\text{Re}(u_t)]^2\}}{E\{|\text{Re}(u_t)|\}} \text{sign}(\hat{u}_t) \quad (3.21)$$

This work attracted much interest and the first modified metric was proposed by Benveniste and Goursat [9]. This algorithm switches to a conventional adaptive equaliser when converged.

Goddard [60] also attracted much interest with his work on microprocessor modems. He shows the minimum dispersion corresponds with zero inter-symbol interference, and minimises

$$g(\hat{u}_t) = E\{(|\hat{u}_t|^p - R_p)^2\} \quad R_p = \frac{E\{|u_t|^{2p}\}}{E\{|u_t|^p\}} \quad (3.22)$$

The choice of $p = 2$ is also known as the *Constant Modulus Algorithm* (CMA) and is one of the most popular blind equalisation algorithms.

Foschini [42] explored some of the theoretical aspects of Goddard's work. He showed that the initialisation was not important: the algorithm would converge from any starting point. He also changed algorithms (to the minimum mean squared error adaptive equaliser) once convergence was achieved. He looked at the effect of having only a finite number of filter taps and found that this was usually sufficient. Topics such as phase acquisition, interference due to cross-polarisation and convergence time were also investigated.

A more recent comparison of various standard cost functions was done by Yamazaki *et al.* [189]. A new metric, the Channel Estimation Standard (CES),

was proposed by Lou [110, 111]. It is designed to take the statistics of the signal into account as well as just the error between the equalised signal and the decision (estimated signal).

3.3.3 High Order Moments and Cumulants

It is well known that second order statistics (autocorrelation) can only be used to identify the amplitude response of the channel and not the phase [10]. The use of higher order statistics allows identification of the phase of the channel as well, assuming the inputs are not Gaussian [100], since the high order moments are zero and therefore phase identification is not possible. In addition polyspectra-based blind equalisers guarantee convergence to the desired solution. For an overview of these methods, see [79].

Cumulants Cumulants may be considered as unbiased high-order moments with a relationship similar to that of the 2nd-order moment to variance. ARMA modelling can be achieved by use of the 2nd and 4th order cumulants [176]. Giannakis [54] advocates such a “*powerful tool*” when addressing the problem of how to estimate the impulse response of a system. Tugnait [177] disagrees with the results for IIR filters, and provides a more complete solution to the blind equalisation problem in [179]. However as with many cases of ARMA modelling, a reliable estimate of model order is required first.

Porat and Friedlander [134] used linear and non-linear least squares based algorithms with 4th order moments to identify the channel. They compare performance in decision-directed update mode with a linear equaliser. They noted that the decision directed approach may lead to divergence when there is a high number of errors.

Hatzinakos [78] concatenates an all-pass filter with a stable minimum phase filter. Estimation of these two filters are done in parallel: the all-pass filter from polyspectra and the minimum phase filter from high-order moment based linear prediction. The aim of this is to improve the convergence by using the linear prediction step.

[156, 157] present an alternative method using high order statistics. The basic idea is that it is sufficient to equalise the variance and higher-order statistics of a

sample in the received signal to that of a sample in the unobserved signal applied to the system input. This leads to several criteria, the maximisation of which leads to the desired solution. It has the advantage that it does not impose any restrictions on the input signal, except for being non-Gaussian.

Tong [171] performs sequence estimation with the Viterbi algorithm using signal decorrelation. The metric used is the square of the difference between the autocorrelation of the sequence and the estimate of the source correlation obtained from the received signal by singular value decomposition.

3.3.4 Cyclo-stationary Statistics

It was noted earlier that in general the phase of the channel cannot be identified using second-order statistics alone. However [44] noted that the modulation scheme leads to cyclostationarity properties. These can then be used to estimate the channel. However his formulation required a training sequence at a low bit rate (to minimise effects of ISI) or a simultaneous pilot tone. This training sequence did not need to be known at the receiver.

Later extensions [172] demonstrate that this method can be used for completely blind equalisation by oversampling the data. Once the channel estimate has been obtained, the signal may be recovered with a DFE or Viterbi decoder.

3.3.5 Neural Networks

Although their application is de-blurring images, the framework used by Steriti and Fiddy [163] is equivalent to an MA channel model. They attempt to obtain the blurring function (channel) and input signal jointly by minimisation. The metric used for this is the inverse filter error in both cases (assuming the blurring function of input signal known, respectively). Rather than using a gradient descent method, the minimisation is achieved by a Hopfield neural network using Euler's method for iteration.

3.3.6 ARMA Channel Models

Hsue and Yagle [85] suggest that the channel may be estimated by assuming the signal to be a white random process. The channel is modelled by a particular

ARMA process (note not simply an MA) — in fact one with a symmetric non-causal filter. Coefficients are estimated using a modified form of the Yule-Walker equations. This constrains the result to the form required.

Gurelli and Nikias [73] also assumed a linear ARMA channel model, described by its transfer function. The poles and zeros of this were estimated from the eigenvectors of the sample correlation matrix of the multiple observed data sequences.

3.3.7 Multiple Input

Many researchers have used multiple receivers to recover an identical input signal. One of the first of these was Stockham Jr. *et al.* [164] dealing with the restoration of old (horn) recordings. The reasoning went as follows: deconvolution in the time domain corresponds to multiplication in the frequency domain, so taking the Fourier transform seemed a sensible first step. If the logarithm is now taken the distortion is simply an addition. He now attempts to minimise the log(Fourier Transform). However it is clear that the musical signals (in this case that of famous singer, Caruso) were a varying offset to the minimisation term and needed to be removed or compensated for somehow. Instead of trying to fit a model from the data, he used a modern recording of the same piece of music. Being relatively undistorted, the offset for each piece of data could then be calculated. The major drawback with this method is in synchronising the two data sources. They are also obviously not exactly the same (if they were, the exercise would be pointless anyway).

Petropulu and Nikias [129] used multiple antennae to obtain two received signals. In this case the input signal is identical, and the channel for each antenna is different. The original signal was estimated by comparison of high order cepstra.

A similar idea was used by Law and Nguyen [96], in this case with astronomical images. The desired (true) image was assumed to be the same from two observation points and the blurring effect of the atmosphere different. An iterative least squares approach was used to minimise the error in this instance.

The work of Slock [159] could be applied either to multiple receivers or to oversampling / fractionally spaced data. He derived an optimal receiver in the presence of correlated noise using a Wiener filter.

3.3.8 Direct Estimation of the Input Sequence

In the case of digital communications, we have a significant amount of prior knowledge about the input signal. The most important of these is the modulation scheme.

One method of demodulation is to put each sample of the received signal into one of many classes. Chen *et al.* [23] use a blind clustering algorithm for decoding M-ary quadrature amplitude modulated (M-QAM) signals. However for high values of M, adaptation may be pulled in two directions. To overcome this, a multi-stage algorithm is used. Firstly the sample is placed in one of a few groups of classes, then a specific decision is made. For example in the case of 16-QAM, the signal is decoded only as far as saying in which quadrant each signal lies. Afterwards the decision is made as to which of the symbols within that quadrant the sample belongs. On channels with significant ISI, considerable overlap between clusters is expected, most likely resulting in poor performance. It is interesting to note that this hierarchical approach — determining which quadrant the sample lies first — is not that dissimilar to Sato's technique of choosing the sign of the signal.

The work of Li [98] takes a different approach. He deals with a bipolar input signal (i.e. each signal is ± 1). The signal is constrained to take only these values, and the inverse filter to achieve this is estimated by gradient descent. However the theory only deals with the noiseless case.

The approach of Gustafsson and Wahlberg [74] goes further than that of Li. They search over the finite set of input signals using a Bayesian maximum a posteriori (MAP) estimate and Kalman filter. An iterative algorithm similar to the list Viterbi using per-survivor processing is used so as to cull the exponential expansion of computational complexity as the data length increases. That is to say only a finite number of likely sequences are kept: with each step through the trellis, the sequences with the smallest probability are rejected.

3.3.9 Statistical Methods

The section describes methods which use probability as the metric. The optimal statistical receiver given a known channel and noise statistics is well known [1] and may easily be used to output optimum soft decisions [99]. If that channel or

its statistics are unknown the problem is made more difficult, leading to a variety of methods.

Joint Channel and Data Estimation One solution is to obtain the Maximum a Posteriori estimate of both the channel and the data sequence jointly. This may be done by the blind trellis search techniques introduced by Seshadri [155].

Channel Estimation Based on Average over Data Sequences. A channel estimate may be obtained by averaging over all the possible sequences. This is a marginal estimate for the channel. It is usually not possible to obtain this estimate directly, so an iterative solution is used. Once the channel estimate has been obtained, the sequence is solved using the Viterbi algorithm, conditional on this estimate [136, §11-5-1]. A related method using a decision feedback equaliser is described by Lee *et al.* [97].

3.3.9.1 Marginal MAP estimates

The concept of marginalisation will be explained in detail in section 4.1.5. Basically it involves removing the channel from our model by integration (analytic or numerical) to obtain an estimate of the data sequence, independent of any particular value for the channel. This method is ideologically better since no channel estimate is required, but the framework for the existence of channel still remains.

Reader and Cowley demonstrate how this can work [141]. The channel impulse response is assumed to have a Gaussian distribution *a priori*, the mean and variance being those of an estimate of its value. This is then integrated out to give the marginal distribution of the original signal. The signal is estimated as the maximum of this distribution (a marginal MAP estimate). This estimate is shown to be equivalent to maximum likelihood sequence estimation (MLSE) when the variance of the channel estimate tends to zero, and equivalent to conventional MLSE metric with per-survivor least squares channel estimates as the variance of the channel tends to infinity.

Numerical methods (Markov chain Monte Carlo) are used to obtain the same estimates in the work of both Chen and Li [22] and Doucet and Duvaut [35].

3.4 Overcoming Problems with Blind Equalisers

Despite success with blind equalisers, sometimes there are still drawbacks: high complexity, slow convergence and poor performance when converged. Various methods have been proposed to overcome some of these problems.

3.4.1 Poor Performance When Converged

In comparison with more conventional adaptive equalisers, blind equalisers track the channel less well and generally perform worse. The advantage that they offer is that convergence is possible from a wide range of starting points. This was formally noted by Picchi and Prati [130] who developed the “Stop-and-Go” decision directed algorithm.

The idea is to combine the best features of both types of algorithm. A decision is made on whether the equaliser is in a converged state or not. If not, then the channel update is via the blind equalisation algorithm. However once the channel is known, operation is switched to an adaptive equalisation algorithm. This should give good performance whilst retaining the ability to recover from fades. His decision boundary is somewhat arbitrary and not probabilistically based. A similar idea was used by Benveniste and Goursat [9] and others several years earlier.

3.4.2 Convergence Time

Slow convergence can result in the loss of much data whilst waiting for the filter taps to converge. Stochastic gradient algorithms can take around ten thousand symbols to converge. High-order statistics methods are generally faster (one–two thousand symbols) but still require enough symbols to get reliable estimates of the higher order moments.

Wesolowski [182] realised that large filter lengths mean a small step size to guarantee stability of updates. This in turn results in slow convergence. He proposes to automatically vary the filter length to speed convergence. The motivation for this is that often most of the energy is in the first few filter taps, so an approximate estimate of these taps can be made with a short filter. He uses the

“Stop-and-Go” ideas mentioned above to control this.

As mentioned in section 3.3.3, Hatzinakos improved convergence time by using concatenated filters. Fast convergence is achieved by using a linear prediction step in a minimum phase filter.

3.4.3 Model Selection

Petropulu and Nikias [129] noted that parametric deconvolution was

“sensitive to model order selection and deviations of the data from the assumed models”

His own solution was to use a non-parametric technique (section 3.3.7). However it is possible to get round this by considering all models and orders which we might consider relevant. Wesolowski’s techniques for varying model order was discussed above (section 3.4.2), but the framework for changing models is not present. For this we need *Bayesian* methodology.

3.5 Summary

In this chapter I have described methods for deconvolution methods for communications systems, more commonly known as *equalisation*. Firstly I described the fundamental designs of equalisers used when the channel impulse response is known. *Adaptive* equalisers may be used when the channel response is not known. A training sequence is sent to allow the equaliser to adapt to the characteristics of the channel. Finally *blind* equalisers were described. These do not require training, and can train themselves based upon the nature of the received sequence alone.

Ultimately in communications systems we are only interested in the symbol sequence. Any other information determined in the process of decoding (for example the impulse response of the channel in the case of many parametric decoders) is only of academic interest at best. Many methods have no way of incorporating any knowledge that we may know in advance other than in the form of initial conditions. *Bayesian statistics* provides such a framework. This will be described in the next chapter.

4

Bayesian Methods

In this chapter the theory and methods used in Bayesian statistics and how they may be applied to signal processing tasks, will be introduced. Bayesian methods have the advantage that they provide a robust framework for making inference about any quantity within the confines of our model.

However, for many problems, the necessary integration cannot be done and we must resort to numerical methods. These will be introduced, with particular attention paid to Markov chain Monte Carlo methods, one method of drawing samples from the posterior distribution in which we are interested. It will be shown how these samples may be used for inference.

4.1 Bayesian Data Analysis

The usual definition of probability is a frequentist one — the ratio of the number of times a possible outcome would occur in a large number of trials, to the total number of trials. However, the use of frequency data for *inductive reasoning* was proposed by Bayes [6]. It was not until Cox [28] showed that it was the only plausible and consistent extension of ordinary logic in which degrees of belief are

represented by real numbers, that the field of *Bayesian statistics* began to emerge in earnest.¹ At a simple level the inferential has already become a part of everyday life; for example, weather forecasts may give the chance of rain on a given day. Armed with this information one is free to choose whether to take an umbrella or postpone the walk in the countryside. It is important to note that the final decision is left to the individual concerned who is at liberty to apply their own knowledge to the problem (e.g. they do not like getting wet, or they know there is shelter) i.e. they are free to choose their own *utility*.

The key difference with classical methods is that each variable is considered to have a probability distribution. A set of observations is then perceived to be a single realisation of states and parameters. This compares to the traditional approach in which the variables are deterministic unknown constants, the value of which we are trying to infer.

4.1.1 Notation

Throughout this chapter the following notation is used:

- y represents the observations that are made.
- x represents the signal and parameters which we wish to recover.
- I represents the prior knowledge of the problem.

In some cases there may be parameters which are important in the context of the model but of no ultimate interest in terms of the quantities that we wish to estimate. In these cases the following notation will be used:

- x represents the signal in which we are interested.
- θ represents the additional parameters.

¹This is not to say that Cox was the only person to give an important contribution to this field. Indeed there was much work on this subject, and a historical overview may be found in chapter 3 of [139]

4.1.2 Bayes Theorem

Bayesian statistics gets its name because the implementation is based directly on Bayes' theorem [6]. Formally stated it may be written as:

$$p(x|y, I) = \frac{p(y|x, I) p(x|I)}{p(y|I)} \quad (4.1)$$

This follows directly from the definition of joint probability.

- $p(y|x, I)$ is called the *likelihood*. It is a measure of probability of the observation having been obtained given that particular signal. This itself is often used as a measure to determine the signal of interest. Comparisons to some of these methods will be drawn in section 4.1.6.1. In the case of additive noise, the form of the likelihood function will be the same as the distribution for the noise.
- $p(x|I)$ represents the knowledge we have of the signal before making the observations or *a priori*. The choice of prior distributions will be discussed in section 4.1.4.
- $p(y|I)$ is called the *evidence*. It appears simply as a normalising term when making inference about the signal and is frequently ignored since it is constant over all possible values of x . It is of great importance in model selection, because it is the equivalent to the likelihood for the model, independent of the actual signal. It therefore represents the evidence for a particular model, hence the name.
- $p(x|y, I)$ is the posterior distribution, and represents knowledge of the data signal, having made the observations.

One of the major strengths of Bayesian methods is the ability to update our knowledge about something with the arrival of more data. This is particularly important for the sequential methods introduced in chapter 6. In addition, the weighting between new and old knowledge is automatic because they are described by a *probability distribution*. If the prior is tightly defined (i.e. it has a low variance) and the observations alone would give a looser distribution (i.e. the likelihood has a higher variance), the posterior distribution is not very different to the prior

— we lose nothing, but gain little. In the context of performing experiments to infer something, this tells us that our experiments are of little value or there have not been enough of them. In engineering applications, such as the mobile telephone link, it may tell us that the equipment is not good enough! However if the likelihood has lower variance, this will tend to dominate the posterior distribution.

4.1.3 The Posterior Distribution

Bayes' theorem tells us how to obtain the posterior distribution from our prior knowledge and the observations we have made. It is important to understand how we may use this distribution: it is possible to infer different information, depending on our problem.

One of the most common estimators is the maximum *a posteriori* (MAP) estimator. The location represents that point in signal space which has higher probability than any other, given the data observed and the prior information we had,

$$\hat{x}_{\text{MAP}} = \underset{x}{\operatorname{argmax}} \{p(x|y, I)\}. \quad (4.2)$$

In some applications the average or mean of the signal may be used. This may also be obtained from the posterior distribution,

$$E\{x\} = \int_x x p(x|y, I) dx. \quad (4.3)$$

These are both point estimates, and as such may give no more information than many non-Bayesian methods. Bayesian methods allow us not only to make point estimates, but also give us information on the reliability of those estimates. For example we may wish to calculate the variance of the distribution or its percentiles. Indeed the expected value of any function of the signal of interest may be obtained from its posterior distribution:

$$E\{f(x)\} = \int_x f(x) p(x|y, I) dx \quad (4.4)$$

4.1.4 Prior Information

We saw earlier that if the likelihood (information from the observations) has a low variance, that is if we make informative measurements, the likelihood dominates the prior. We also saw that this will often be the case if the observations are to be useful, so what use is prior information?

Prior information can come in many more ways than just the prior distribution. In many respects, the whole system model is prior information, as it is something we believe to be true, and constrains our system. Often there is great difficulty in representing our prior knowledge in a mathematical form. It is very easy to produce results which are implausible, but harder to say exactly why.

4.1.4.1 Choice of Prior Distributions

The choice of prior distributions often causes heated debate amongst Bayesians and non-Bayesians alike. It is often (wrongly) thought that the use of a prior distribution is the only difference between Bayesian techniques and conventional methods such as maximum likelihood. In addition the “tuning” of priors can result in markedly different results. The choice of prior distribution must always be carefully made, since we do not wish to influence the posterior too much, but at the same time we would like to restrict it to a realistic shape or to a range of values.

At this stage it is useful to carefully examine the system and to work out what is really known about it. I believe you should not hesitate to let your prior represent something that you know to be true. For example if digital data is transmitted over a communications link, then we know the original must come from a known finite set of symbols. If your estimate does not illustrate this, then this prior information is lost. Examples of prior information sources are the physical properties or limitations of the system and empirical models built from experimental data.

As a corollary, the prior distribution should not represent information that is not truly known: otherwise undue bias is introduced into the results obtained.

4.1.4.2 Non-informative Priors

The idea of a non-informative prior is one which does not affect the posterior. It allows us to say that we know absolutely nothing about the variable and ensures

that the posterior is not influenced by the prior. It is easy to see that for a location variable (such as a mean) we can use a uniform prior. The case is not so clear cut for scale variables such as a variance. Jeffreys [88] proposes that the prior p.d.f. should be uniformly distributed over different scales. This is the same as assuming that the logarithm of a scale variable is uniformly distributed. It follows, therefore, that

$$p(x_s) = \frac{1}{x_s} \quad (4.5)$$

where x_s is the scale variable.

This prior is *improper*, that is it cannot be normalised ($\int p(x_s) dx_s \rightarrow \infty$). If a proper prior is required we may obtain one by placing bounds on this prior over a certain region, if we can plausibly choose, or at least estimate, such a region.

4.1.4.3 Conjugate Priors

Often one cannot pin down a variable to a particular shape of distribution. For mathematical convenience a prior is often chosen so that the posterior distribution is of the same parametric form as the distribution of the prior. This is termed a *conjugate prior*. Note that the choice of this distribution is dependent on the likelihood function. For example, let us assume that the likelihood has Gaussian form. If we choose a Gaussian prior, then the posterior will also be Gaussian (since the product of two Gaussians is a Gaussian).

4.1.4.4 Hyper-Priors

Hyper-priors are priors in which the functional form of the prior is chosen, but the selection of parameters for this prior is devolved to the estimation procedure. These parameters are called hyper-parameters. Frequently, for convenience, a conjugate prior is used for these hyper-parameters. Hyper-priors are commonly used because they appear to be ultimately uninformative, since they are data driven, but this is not necessarily so. By imposing such a prior, the one piece of information you are giving is that the variable is *drawn from the distribution with the functional form chosen*. This often does not accurately reflect the prior beliefs that you may have about the signal.

4.1.5 Marginalisation

The ability to perform marginalisation is perhaps one of the greatest strengths of Bayesian methods. Let us consider a parametric model with parameters θ . For clarity Bayes' theorem is restated including the parameters:

$$p(x, \theta | y, I) = \frac{p(y | \theta, x, I) p(x, \theta | I)}{p(y | I)} \quad (4.6)$$

We can see that the posterior distribution is the joint distribution of the signal and the parameters. Suppose we are only interested in recovering the signal: the parameters are then of no interest to us² but are an integral part of the model we are using. We may still obtain a MAP estimate as before (equation (4.2)), but we get the joint MAP estimate for both signal and parameters.

We may obtain the density $p(x | y, I)$ by integrating out the parameters:

$$p(x | y, I) = \int_{\theta} p(x, \theta | y, I) d\theta \quad (4.7)$$

This process is called *marginalisation* and the probability density obtained is the *marginal posterior density*. It is equivalent to considering all possible values of that parameter weighted according to the probability of that value occurring.

We may now obtain a marginal MAP (MMAP) estimate — a MAP estimate over the marginal density. Since we have lost the dependencies to the parameters we can now answer our original question “*What is the most probable signal, having made these observations, within the confines of the model and my prior knowledge?*”

4.1.6 Comparison to Conventional Methods

It is worth noting that many methods currently employed have their basis in optimising posterior probability. In the majority of cases, the difference is that point estimates are obtained, or the joint probability distribution is used rather than the marginal.

²In some applications, such as seismic deconvolution, the parameters are the part of interest and the signal is of no interest. The same logic may be applied with the signal, x , and parameters, θ , swapped over.

4.1.6.1 Maximum Likelihood

The method of maximum likelihood is well known and commonly used. The question posed is almost the opposite way around — “*What is the signal which is most likely to give the observations that have been made?*”

$$\hat{x}_{ML} = \underset{x}{\operatorname{argmax}} \{p(y|x, I)\} \quad (4.8)$$

For a given model, the evidence term in Bayes’ theorem is fixed: the only difference between the posterior distribution and the likelihood function is the prior. This means that point estimates obtained with maximum likelihood are the same as a MAP estimate with a non-informative prior (section 4.1.4.2).

4.1.6.2 Least-Squares Approach

A common technique is to minimise the mean squared error between the actual observations and the predicted output, given the estimated signal and parameters:

$$\hat{x}_{LS} = \underset{x}{\operatorname{argmin}} \{(y - \hat{y}(x))^2\} \quad (4.9)$$

where $\hat{y}(x)$ is the estimate of y in the model, given the value of x . Indeed the mean squared error is the definition of the variance.

For particular cases of prior and noise distribution this estimate is equivalent to a MAP estimate. In a similar fashion least squares (LS) or minimum mean squared error (MMSE) approaches may be applied to other measures of fit other than the disparity between actual and predicted observations. Examples of this include the LMS and constant modulus algorithms described in sections 3.2.2 and 3.3.2 respectively.

Although point estimates may be the same in certain cases, it is important to realise that with more complicated models, methods based upon one of these methods are not necessarily optimal, even in the presence of Gaussian observation noise and the choice of an appropriate prior. As an example, let us consider the Viterbi algorithm used for maximum likelihood sequence estimation (section 3.1.3). The algorithm can only determine the optimal trajectory through state space if the channel is known exactly. In other words it estimates the optimal se-

quence *conditional* on the channel estimate. Since we are not ultimately interested in the channel itself, and it is unknown, a better estimate of the sequence may be obtained from the marginal distribution. This also explains how performance better than Viterbi algorithm may be obtained: the Viterbi algorithm is essentially limited by the estimate of the channel used.

It is also important to note that Bayesian methods are not limited to cost functions directly related to probability measures. By obtaining the posterior probability distribution we may still use MMSE or other cost functions [91, §11] to make our chosen inference.

4.2 Numerical Methods

Thus far many of the advantages of Bayesian methods have been described. One of the biggest disadvantages is that they can only be applied analytically to a small subset of problems. For example, to get an analytic expression for the marginal distributions we require to perform the integration in equation (4.7). The upshot of this is that the use of Bayesian methods has been limited, until recently, to certain families of distributions (for priors in particular). This generally leads to linear, Gaussian models which are often unrealistic and do not represent the underlying physical system particularly well. The exponential growth and falling cost of computing power has led to increased use of numerical methods to get around these problems.

The simplest method in widespread use is that of *optimisation*. The computation of MAP estimates, for example can be made using widely available optimisation methods such as Newton-Raphson and simplex methods [135, §10]. In general a probability distribution may be multi-modal and finding a global maximum is very difficult. Methods such as simulated annealing [92], which I shall revisit in section 8.2.1, and Expectation-Maximisation (E-M) [120] go a long way towards solving these problems. If estimates of other quantities are required, such as the variance of the distribution, an additional optimisation is required.

If we are to retain all the advantages of Bayesian methods, we require a way of finding or representing the posterior distribution we are interested in. One way of achieving this is to obtain samples from the distribution. It is helpful to consider

the duality between a probability density function and a set of random draws from that distribution. Providing we have a sufficiently large number of draws, we may use the samples to make estimates of maximum, mean, variance, other moments, percentiles or other statistics of interest.

4.2.1 Monte Carlo Methods

Monte Carlo methods gain their name from the many games of chance for which the city-state is famous. They are all based upon the generation and subsequent use of random numbers drawn from some probability distribution. The numbers or points generated can then be used to perform the integration necessary for so many problems.

Suppose we are interested in the expectation of some function, $E\{f(\mathbf{x})\}$. Monte Carlo integration evaluates this by drawing samples $\{\mathbf{x}^{(i)}; i = 1, \dots, N\}$ from the probability distribution, $p(\mathbf{x})$, and then approximating:

$$E\{f(\mathbf{x})\} \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}^{(i)}) \quad (4.10)$$

When the samples $\mathbf{x}^{(i)}$ are independent, laws of large numbers ensure the approximation can be made as accurate as required by increasing N (section 4.2.1.1). The samples need not necessarily be independent — they can be generated by any process which draws samples throughout the support of $p(\cdot)$ in the correct proportions. One way of doing this is through a Markov chain with $p(\cdot)$ as its stationary distribution.

4.2.1.1 Accuracy of Estimates

The error obtained in making estimates of the mean from samples obtained directly from the distribution of interest is well known (see e.g. [149]) and can be easily extended:

Let ζ be an unbiased estimate of the quantity we are interested in:

$$\zeta \approx E\{f(\mathbf{x})\} \quad (4.11)$$

The variance of this estimate, ψ^2 , is given by:

$$\psi^2 = \frac{\sigma^2}{N} \quad (4.12)$$

where σ^2 is the variance of the underlying distribution ($\text{Var}\{f(x)\}$). An unbiased estimate of σ^2 may be obtained from:

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^N (f(x^{(i)}) - \zeta)^2 \quad (4.13)$$

We may therefore estimate the accuracy of any estimates made, or even continue generating samples until we gain the required accuracy [149, §7.1].

4.2.1.2 Implicit Marginalisation

One major advantage in using samples from the joint density for making estimates is that marginalisation is implicit. Consider the following factorisation:

$$p(x, \theta | y) = p(\theta | x, y) p(x | y) \quad (4.14)$$

We may consider the samples to have had the values for x drawn first from the marginal distribution, $p(x|y)$, and the values for θ to have been drawn from the conditional. By simply ignoring the values of θ we have samples from the marginal distribution. Of course by inter-changing x and θ in the equation it is clear that the joint samples could also be considered to be from the marginal density $p(\theta|y)$.

It is sometimes confusing to think of the samples as having arisen from both marginal distributions at the same time. An alternative way of thinking is to pose the problem the other way around. Marginalisation is the mapping of samples in $\{x, \theta\}$ space onto the subspace of $\{x\}$ alone. This is equivalent to ignoring the values for θ associated with the samples.

Implicit marginalisation is not always very efficient. Robert *et al.* [143] propose an alternative method of obtaining MMAP estimates with a technique based upon MCMC.

4.2.2 Density Estimates

We have seen that samples from the joint distribution are sufficient to make estimates of certain quantities, for example expectations. However it is occasionally useful to make an estimate of the distribution in functional form.

The simplest method for obtaining a density estimate is to construct a histogram: split the space over which we have samples into equally spaced bins; the bars have a height equal to the number of samples within that bin. This is demonstrated in Figure 4.1

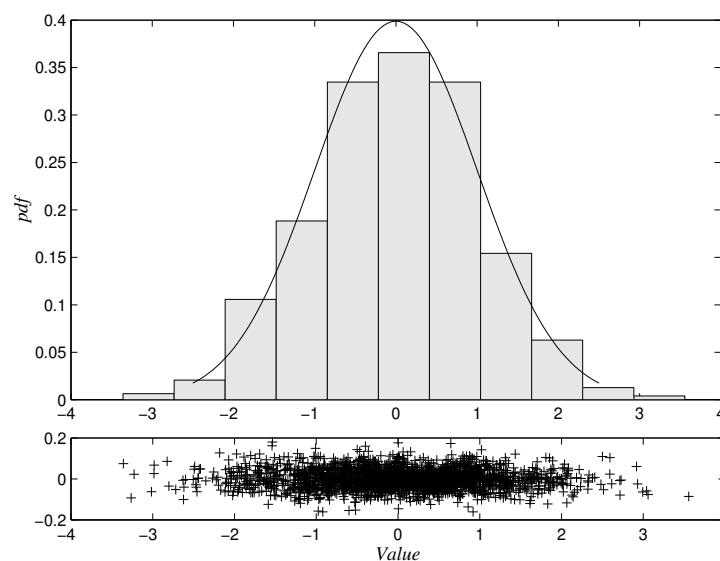


Figure 4.1. Histogram of some samples generated from a normal distribution overlaid with correct distribution. The samples are shown on the lower plot.

The histogram method is limited in the sense that it provides a piecewise constant estimate to a smooth density, however it is a special case of the more general kernel density estimation (see e.g. [158]) with a square kernel. A Gaussian density is usually chosen, although other functions are sometimes preferred [183]. Extensions exist for the estimation of marginal densities [46, 21] and the reduction of the number of points in the resulting mixture approximation [185]. With all forms of kernel density estimation, there is a trade-off between resolution and smoothness, which may be adjusted by altering the size or variance of the kernel.

4.2.3 Random Numbers

The generation of genuinely random numbers on a deterministic machine is impossible. However, there exist a large number of pseudo-random number generation techniques [149, §3] [135, §7] which are adequate for most purposes. It is important to ensure that we are generating “good” random numbers for any inference to be valid. Random number generators can be broadly classified into three types:

- **Congruential:** $x_n = ax_{n-1} + b \pmod{m}$
- **Shift-register:** $x_n = x_{n-1}T$, where T is a binary matrix and arithmetic is done modulo 2.
- **Lagged Fibonacci:** $x_n = x_{n-r} \diamond x_{n-s}$ for some integers r and s and some operation \diamond .

All of the simulations carried out in the course of my work use the random number generator in MATLAB 5. It is specifically designed to generate random floating-point numbers (as opposed to scaled integers). Technically it is an “add-with-carry” generator based upon the lagged Fibonacci method, the difference being that the carry flag is kept from the last operation. This is combined with a shift register random integer generator. Despite being very fast, to cycle through the entire period of the generator would take 10^{435} years on a Pentium computer [119]. It is based upon the work of George Marsaglia [114, 115, 116].

4.3 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods provide a relatively easy, although computationally intensive, way of generating samples from any probability distribution. By cleverly constructing a Markov chain (using one of the standard methods below) such that its stationary distribution is the distribution we wish to sample from, we may generate these samples by running the Markov chain for a sufficiently long time. The initial “burn-in” samples are discarded up to a point where we are happy that the Markov chain has converged. After this time the samples are from the correct distribution as required.

4.3.1 Markov Chains

Consider a collection of random variables $\{X_0, X_1, X_2, \dots\}$, where X_t may be regarded as the state of a system at time t and t is discrete. A Markov chain is a process with the property:

$$P(X_t | X_0, X_1, \dots, X_{t-1}) = P(X_t | X_{t-1}) \quad (4.15)$$

where $P(\cdot | \cdot)$ is the transition kernel. In other words the probability distribution of any state is dependent only on the previous state.

As an example, let us return to the idea of predicting the weather. We shall restrict ourselves to predicting only whether it will rain or not, and assume that this system has a Markov property, so that the chance of rain tomorrow is only dependent on the weather today.

Let us assume:

$$P(\text{Rain tomorrow} | \text{Dry today}) = 0.4 \quad (4.16)$$

$$P(\text{Rain tomorrow} | \text{Rain today}) = 0.8 \quad (4.17)$$

We may write our system model in state-space form as:

$$\begin{bmatrix} Pr(Rain) \\ Pr(Dry) \end{bmatrix} = \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix} \times \begin{bmatrix} Pr(Rain) \\ Pr(Dry) \end{bmatrix} \quad (4.18)$$

$$x_t = A \times x_{t-1}$$

where x_t is our state probability at time t . This represents our belief in what the weather will be like on that particular day. The matrix A may be regarded as either the state propagation or transition kernel matrix for the Markov chain. We may make estimates of the state at any future time, by repeated application of this transition kernel. By performing an eigen-decomposition this may be simplified thus:

$$\begin{aligned} x_t &= A^t x_0 \\ &= V \Lambda V^{-1} V \Lambda V^{-1} \dots V \Lambda V^{-1} x_0 \\ &= V \Lambda^t V^{-1} x_0 \end{aligned} \quad (4.19)$$

If we substitute the numerical values:

$$\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 0.4 \end{bmatrix}; \quad \mathbf{V} = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

we obtain:

$$\mathbf{x}_t \approx \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{1}{3} \end{bmatrix} \mathbf{x}_0 \quad (4.20)$$

for large enough t . Our probabilities are now *independent of initial conditions*. We have reached the *stationary* or *invariant distribution* of the Markov chain. For example if we wanted to give the chance of rain on 4th November 2094, it would be approximately the same as on 25th December 2014, namely $\frac{2}{3}$. It does not matter whether it is raining or it is dry today.

The stationary distribution of a Markov chain, $\pi(\mathbf{x})$, is defined as:

$$\pi(\mathbf{x}) = \int_y \pi(\mathbf{y})P(\mathbf{x}|\mathbf{y})d\mathbf{y} \quad (4.21)$$

where $P(\mathbf{y}|\mathbf{x})$ is the transition probability: the probability of moving from state \mathbf{x} to state \mathbf{y} . Note that for a general Markov chain, this may not necessarily exist.

To generate samples from the stationary distribution, we therefore need to pick a starting point, and simulate the chain until convergence has been achieved. Further samples generated from this chain are then drawn from the stationary distribution. Not all Markov chains are suitable for this purpose — the chains must have the following properties:

- **Irreducible:** All points may be reached from any starting point.
- **Aperiodic:** This prevents the chain from oscillating between different states or sets of states.
- **Positive recurrent:** If the initial value is sampled from the stationary distribution, subsequent values from iterations of the chain will be distributed according to the stationary distribution.

These ensure that the stationary distribution may always be reached, given sufficient time for convergence. For a discussion of these properties in the context of MCMC, the enquiring reader is referred to [169, 170, 144].

Choosing a Markov chain satisfying the invariance condition (equation 4.21) may be hard to handle, so the more restrictive condition of time *reversibility* is often used. $P(\cdot|\cdot)$ is said to be reversible with respect to $\pi(\cdot)$ if it satisfies the relation:

$$\pi(x) P(y|x) = \pi(y) P(x|y) \quad (4.22)$$

This is called *detailed balance*.

The choice of different Markov chains with a given stationary distribution affects rates of convergence and mixing properties (how quickly the whole posterior distribution is explored).

4.3.2 Markov Chain Monte Carlo algorithms

Thus far I have introduced Monte Carlo methods and idea of using Markov chains to generate samples from stationary distribution of the Markov chain. To be useful in practical problems we require a method of constructing a Markov chain with a particular stationary distribution. It is then possible to generate samples from any distribution, and in particular from the posterior probability distribution in which we are interested.

4.3.2.1 Metropolis-Hastings Algorithm

In 1953, Metropolis *et al.* [117] introduced a flexible method for generating Markov chains tailored to converge to a posterior density, $\pi(x)$. This was generalised by Hastings [77]. The only requirement of the algorithm is that we can compute $\pi(x)$ up to a normalising constant. The i th iterate of the Markov chain is denoted by a bracketed superscript, $x^{(i)}$. The algorithm proceeds as follows:

1. Draw a starting point $x^{(0)}$
2. For each i from 1 onwards:
 - (a) Draw a proposal from proposal density $q(x^* | x^{(i-1)})$.
 - (b) Draw a uniformly distributed random number, u , between 0 and 1.
 - (c) **If** $u < \alpha(x^* | x^{(i-1)})$ **then** accept the move by setting $x^{(i)} = x^*$
else set $x^{(i)} = x^{(i-1)}$.

The acceptance probability, α is given by:

$$\alpha(\mathbf{x}^* | \mathbf{x}^{(i-1)}) = \begin{cases} r(\mathbf{x}^* | \mathbf{x}^{(i-1)}) & \text{if } r(\mathbf{x}^* | \mathbf{x}^{(i-1)}) \leq 1 \\ 1 & \text{if } r(\mathbf{x}^* | \mathbf{x}^{(i-1)}) > 1 \end{cases} \quad (4.23)$$

where the ratio $r(\mathbf{x}^* | \mathbf{x}^{(i-1)})$ is given by:

$$\begin{aligned} r(\mathbf{x}^* | \mathbf{x}^{(i-1)}) &= \frac{\pi(\mathbf{x}^*)}{\pi(\mathbf{x}^{(i-1)})} \frac{q(\mathbf{x}^{(i-1)} | \mathbf{x}^*)}{q(\mathbf{x}^* | \mathbf{x}^{(i-1)})} \\ &= (\text{likelihood ratio}) \times (\text{prior ratio}) \times (\text{proposal ratio}) \end{aligned} \quad (4.24)$$

since the posterior density is proportional to the product of the prior and the likelihood. Note that when implementing this algorithm, one may use $\alpha = r$ since $u \leq 1$.

In some respect the accept/reject method is similar to optimisation techniques: points of higher posterior density are always accepted, with like steps which reduce the posterior density taken only sometimes.

To verify that this does indeed generate samples from $\pi(\mathbf{x})$, consider the transition probability:

$$P(\mathbf{x}_i | \mathbf{x}_{i-1}) = \begin{cases} q(\mathbf{x}^{(i)} | \mathbf{x}^{(i-1)}) \alpha(\mathbf{x}^{(i)} | \mathbf{x}^{(i-1)}) & \text{if } \mathbf{x}^{(i)} \neq \mathbf{x}^{(i-1)} \\ 1 - \int_{\mathbf{x}^*} q(\mathbf{x}^* | \mathbf{x}^{(i-1)}) \alpha(\mathbf{x}^* | \mathbf{x}^{(i-1)}) d\mathbf{x}^* & \text{if } \mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} \end{cases} \quad (4.25)$$

In both cases this satisfies the detailed balance equation (4.22).

This algorithm is particularly powerful because we are free to choose the proposal density whilst the stationary distribution remains the same. Typically a proposal density is chosen which is fast or easy to sample from. Generally better performance (in terms of convergence time and mixing properties) is obtained if the proposal density is a heavier-tailed distribution than the posterior. The step-size (variance of the proposal distribution) needs to be chosen with care: too small and we get poor mixing properties; too large and many moves are rejected. Roberts [145] suggests that a rejection ratio of about $\frac{1}{4}$ is close to optimal.

Single Component Metropolis-Hastings The algorithm described above assumes that all the states (all the components of \mathbf{x}) are updated in one go. This does not necessarily have to be the case. It is possible to only update a single component or a subset of components in one Metropolis-Hastings step. Indeed this framework was originally proposed by Metropolis [117], but is now usually called *single component Metropolis-Hastings*. One iteration of this algorithm involves a proposal for each of the components in turn, in some order. If a proposal is accepted, this component is updated for subsequent proposals within the same iteration. An example of this is shown in the section on the Gibbs sampler below.

4.3.2.2 The Gibbs Sampler

The Gibbs sampler [49, 45] is a special case of the single component Metropolis-Hastings algorithm, where the proposal density is chosen to be the conditional probability density with conditioning on the latest draws of the other parameters; each component is updated separately:

$$q(\mathbf{x}_j^{(i)} | \mathbf{x}^{(i-1)}) = p(x_j | x_1^{(i-1)}, x_2^{(i-1)}, \dots, x_{j-1}^{(i-1)}, x_{j+1}^{(i-1)}, \dots, x_n^{(i-1)}, y) \quad (4.26)$$

where $x_j^{(i)}$ is the i th iterate in the Markov chain of the j th component of the state vector. It is easy to check that the acceptance probability is always 1 (substitution into equation (4.23)), so the i th iteration may be summarised as:

$$\begin{aligned} x_1^{(i)} &\sim p(x_1 | x_2^{(i-1)}, x_3^{(i-1)}, \dots, x_n^{(i-1)}, y) \\ x_2^{(i)} &\sim p(x_2 | x_1^{(i)}, x_3^{(i-1)}, \dots, x_n^{(i-1)}, y) \\ &\vdots \\ x_n^{(i)} &\sim p(x_n | x_1^{(i)}, x_2^{(i)}, \dots, x_{n-1}^{(i)}, y) \end{aligned} \quad (4.27)$$

The Gibbs sampler is one of the most popular MCMC methods in use today, and is particularly useful if the full conditionals are available to sample from. However the relatively straightforward method hides the presence of the Markov chain. In addition the full conditional probability distributions are frequently difficult to sample from or simply unavailable (the marginalisation integral (4.7) is not possible analytically).

Order of Sampling The description of the Gibbs sampler above shows a *deterministic scan*, in which each of the components are sampled in order. An alternative to this is a *random scan* where the order that the components are sampled in varies at each iteration. It is very difficult to give hard and fast rules about which is superior. Stark [162, §5.4] suggests that random order is “significantly better” than a sequential one since it gives more freedom for exploration of the space. Zeger and Karim [191] suggest updating highly correlated components more frequently than other components, to improve mixing. More recent work [71] has shown that the most efficient sampling order is highly problem dependent: particularly in the area of component correlation.

4.3.3 Practical Issues

4.3.3.1 Blocking / Joint Sampling

I have described the implementation of single component samplers, but I have not said anything about how these components should be chosen. Sampling highly correlated components separately often results in poor mixing as the step size for each component is quite small. One solution is to try a different sampling strategy or reparameterisation [58]. A possible alternative is to treat a block of components together and make a joint proposal. One cannot say that, in general, the use of blocking will improve mixing or the rate of convergence [106, 146]. However, in the majority of engineering applications, we have found that sampling certain parameters jointly is indeed more efficient.

4.3.3.2 Convergence

The issue of convergence is both very complex and importance. At the most basic level we wish to ensure that the Markov chain has converged and that the samples we are generating are from the true posterior distribution. This answers the question “*How many burn-in samples are required?*” Once we have obtained convergence we then need to know how many further samples need to be generated to get an accurate description of the posterior distribution. From an engineer’s perspective, the system will have a finite amount of computational power, and good results are the thing that counts. A balance must be struck: it may be better to

have an almost converged Markov chain so that we can generate more samples before stopping the chain so as to get lower Monte Carlo variance when calculating estimates.

The starting point for monitoring convergence is the variance between different iterates of one component (within-sequence variance). This should settle when the chain has converged, however it is not always clear that this has necessarily reached its minimum value.

Perhaps the most intuitive method for assessing convergence is to monitor the variance of the simulations of a few Markov chains. If the between-sequence variance is not significantly larger than the within-sequence variance, then we may assume the chains to be converged. This is described in [47].

In practical problems, one of the components is usually the variance of observation or state noise. Monitoring either of these components, or indeed any variance term, provides an indicator as to when convergence has been achieved, since this will record a lower value, on average, when converged. It is not clear if a global minimum has been obtained, something that is often apparent if many runs of the Markov chain are made, providing at least one of the chains converges to the global minimum.

There is a school of thought that burn-in is not required [53]. This relies on that fact that if you can generate a starting point that could plausibly have been drawn from the posterior distribution, subsequent points will also be from the posterior (provided that the Markov chain is Harris recurrent [118]). This is all fine as long as you can generate a good starting point.

In their seminal paper, Propp and Wilson [137] introduce a method for ensuring convergence. Called *exact sampling* or *perfect simulation*, samples drawn from the correct distribution are guaranteed, eliminating the need for assessing convergence. Although some progress has been made [70], the algorithms are still severely limited into which state-spaces they may be applied to.

4.3.3.3 Number of chains

The use of more than one Markov chain, i.e. more than one run of the chosen method, has been suggested by a number of researchers. There are essentially three motivations for this.

- **Obtaining independent samples.** The samples produced by a Markov chain are, by their very nature, dependent. Different runs of the same chain can produce independent samples. It is now generally agreed that this requirement is misguided unless there is some need for independent samples [57, ch. 1]. It is certainly not required for ergodicity.
- **Convergence** Inter-chain and intra-chain dependencies may be considered as a guide to convergence, as described in section 4.3.3.2.
- **Exploration** It is possible that the Markov chain will appear to explore a subspace of the area of support of the probability density function of interest. It is hoped that different chains will converge to different subspaces of the area of support, so a more complete exploration of the space is possible. This is supposedly particularly true if different initial conditions are chosen. There are flaws in the above argument. If the Markov chain has converged, then draws are indeed from the required probability density function, so a more complete exploration of the space should be possible by simply taking more iterations from a single chain. If the chain has poor mixing properties, then it is difficult to see how the posterior distribution would be accurately represented by separate chains exploring different areas of the probability space. It would be better to design a “better” Markov chain with superior mixing properties.

It is clear that if the chain takes too long to converge, then burning in too many chains will be a waste of time. The issue is still open to debate with advocates of a single long chain [52] and many or few shorter ones [45]. Gilks *et al.* take a more pragmatic approach:

“If several processors are available, running one chain on each will generally be worthwhile.” [57, §1.4.4]

4.4 Summary

In this chapter the use of Bayesian methods for Signal Processing problems has been described. These methods are powerful in the sense that we are interested in

obtaining or generating samples from the posterior distribution. This allows us to make estimates of any quantity in which we are interested, including the possibility of restructuring the model so nuisance parameters can be removed through the use of marginalisation.

In practice, particularly if we use models that are physically realistic, the integration or optimisation required is often not possible analytically. Numerical methods are then required. The use of Markov chain Monte Carlo (MCMC) methods to generate samples from the posterior distribution is powerful and increasingly popular. From these samples we make estimates of the quantities required. In addition it is possible to obtain measures on the reliability of these estimates, either in the form of variance estimates, confidence limits or percentiles, something that is often not possible with conventional non-Bayesian methods.

Markov Chain Monte Carlo Methods 5 for Blind Equalisation

In the previous chapter the advantages of Bayesian methods were described, as well as some of the numerical methods for implementing them. In this chapter, we look at the application of some of those methods to the problem posed in a digital communications system: We are interested in extracting the transmitted signal from a received signal or, more technically, the transmitted symbol sequence, u_t , which maximises the probability distribution $p(u|y)$ for all t over the time period in which we are interested.

In particular several novel MCMC moves are introduced to achieve the best performance on a digital communications system. These are the use of random draws for a sequence of data, which has been previously presented in [25], and reversible jump moves for model order, delay ambiguity and phase ambiguity.

5.1 The Gibbs Sampler for Blind Equalisation

The most popular algorithm for MCMC, the Gibbs sampler, was chosen to obtain samples drawn from the posterior probability distribution. Previous work in this area includes that of [22], who describe the simplest algorithm (symbol-based

sampler) and [35] who gives the communications system as an example for their work on hidden Markov models. Neither work discusses the practical implications of the variations in the algorithms or presents a complete picture of performance at varying SNR or a comparison in performance with more conventional equalisers.

Although the Gibbs sampler is designed for processing all the data at once, due to the sequential nature of the data, it is processed in batches. As is common with many other applications, the data is split up into frames and each frame is processed separately. This occurs naturally in some systems which already have a frame structure, such as GSM. This frame shall have length N , and it will be labelled such that $t = 1, 2, \dots, N$ for notational simplicity.

5.1.1 Model

The model used for the communications system is the digital FIR filter channel with additive noise, described in section 2.3.1 and shown diagrammatically in figure 5.1. The channel filter is assumed to be time-invariant. The mathematical

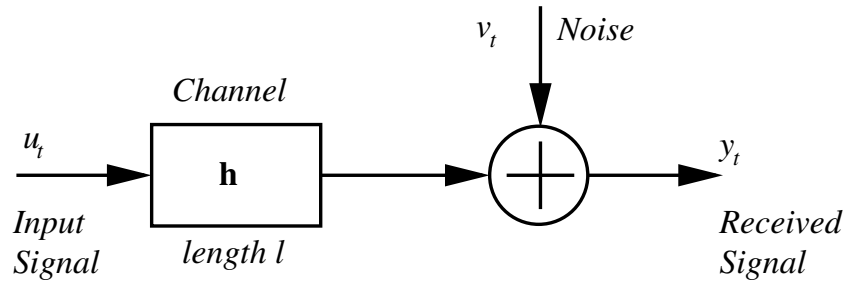


Figure 5.1. Model of Communications System Used.

formulation is given by equation (2.18), repeated here:

$$y_t = \mathbf{h}^T \mathbf{x}_t + v_t \quad (5.1)$$

where the state $\mathbf{x}_t = u_{t:t-(l-1)}$ as in equation (2.8). An uncorrelated Gaussian noise source is assumed, which leads to a likelihood of the form:

$$p(y_t | \mathbf{h}, u_{0:t}, \sigma_v^2) = N\left((y_t - \mathbf{h}^T \mathbf{x}_t), \sigma_v^2\right) \quad (5.2)$$

5.1.2 Priors

The choice of priors is a somewhat contentious issue. As a general rule, they should encompass as much physical knowledge as possible. However when there is limited information available, or none that can easily be expressed mathematically, conjugate priors (section 4.1.4.3) are chosen for mathematical convenience or simplicity of implementation (and therefore highest speed of operation).

Input Symbols. Given that there is a finite set of symbols, we may reasonably assume that the prior takes the form:

$$p(u_t = S_i) = \frac{1}{q} \quad (5.3)$$

since many communication systems have a coding scheme to keep information content high.¹ In addition there may be known symbols from a training sequence or synchronisation symbols: these should be assumed known.

Channel. A Gaussian prior was chosen:

$$p(\mathbf{h}) = N(0, \sigma_h^2 \mathbf{I}) \quad (5.4)$$

with σ_h large enough to avoid being overly informative. A more general multivariate Gaussian prior $p(\mathbf{h}) = \mathcal{N}(\mathbf{h}_0, \Sigma_h)$ could be incorporated in a straightforward manner. This is a conjugate prior for Gaussian observation noise.

Noise. The noise variance prior is an inverted gamma distribution, the conjugate prior for the variance of a Gaussian [48]. It is defined as:

$$p(\sigma_v^2) = \frac{\beta^\alpha}{\Gamma(\alpha)} (\sigma_v^2)^{-(\alpha+1)} e^{-\beta/\sigma_v^2} \quad (5.5)$$

which we write as $IG(\alpha, \beta)$. The hyper-parameters are set to a value close to zero, which the uninformative Jeffreys' prior for a scale parameter.² Chen and

¹Differential encoding is also often employed. This will result in a Markov model for x . However for equiprobable data, this reduces to the same prior.

²A non-informative distribution is obtained as α and β tend to zero.

Li [22] use an inverted chi-squared distribution, which is a special case of the inverted gamma.

5.1.3 The Symbol-Based Sampler

As described in section 4.3.2.2, the Gibbs sampler requires random draws from the full conditional distributions in some order. Sampling for noise variance and channel may be done directly. Conjugate priors were chosen, so the conditional distributions, obtained in part by the results given in appendix B, are given by:

$$p(\sigma_v^2 | x, h, y) \propto p(y | x, h, \sigma_v^2) p(\sigma_v^2) \quad (5.6)$$

$$\propto IG \left(\alpha_v + \frac{N}{2}, \beta_v + \frac{\sum_{i=1}^N |y_i - h^T x_i|^2}{2} \right) \quad (5.7)$$

and

$$p(h | x, \sigma_v^2) \propto p(y | x, h, \sigma_v^2) p(h) \quad (5.8)$$

$$\propto \mathcal{N}(\mu_H, \Sigma_H) \quad (5.9)$$

where:

$$\Sigma_H^{-1} = \frac{1}{\sigma_v^2} \sum_{i=1}^N x_i^T x_i + \Sigma_h^{-1} \quad (5.10)$$

$$\mu_H = \Sigma_H \left(\frac{1}{\sigma_v^2} \sum_{i=1}^N y_i x_i + \Sigma_h^{-1} h_0 \right).$$

To sample for the symbols let us first consider the case where each symbol is sampled independently, conditional on the surrounding symbols, as well as the current draws of the channel and noise variance. This is the approach taken in [22] and also [128], albeit in a different framework. The likelihood may be factorised as:

$$p(y_{1:N} | h, x_{1:N}, \sigma_v^2) = \prod_{i=1}^N p(y_i | h, x_i, \sigma_v^2) \quad (5.11)$$

so each symbol may be drawn from:

$$p(u_t = S_k | u_{-t}, \sigma_v^2, \mathbf{h}) = \frac{\frac{1}{q} \sum_{i=t-(l-1)}^t \mathcal{N}(\mathbf{h}^T, (x_i | u_t = S_k), \sigma_v^2)}{\sum_{j=1}^q \frac{1}{q} \sum_{i=t-(l-1)}^t \mathcal{N}(\mathbf{h}^T, (x_i | u_t = S_j), \sigma_v^2)} \quad (5.12)$$

where u_{-t} denotes the set $\{u_j\}$ for $j = 1 \dots t-1, t+1, \dots, N$ and $(x_i | u_t = S_j)$ denotes the vector x_i with the substitution to one of the elements of x_i such that $u_t = S_j$.

5.1.4 Problems Encountered

The ability of the Gibbs sampler to fully explore the posterior probability space is largely dependent on the variance of the random samples drawn. This in turn is dependent on the current draw of the noise variance. Paradoxically, a high signal-to-noise ratio therefore restricts this exploration, possibly resulting in long convergence times, or poor performance when the number of iterations is limited.

Overall performance can be quite good, although for intermediate noise levels the speed of convergence is quite slow. There are two sources of error resulting from local maxima in the posterior density. Technically, the Markov chain should explore all the space eventually. However, it illustrates both *poor mixing* and *slow convergence*. These properties are detrimental in practical systems when there is only time to run the chain for a finite number of iterations (which may be quite small).

The resulting errors occur in two forms:

- **Burst errors.** Familiar from the use of the Viterbi algorithm, bursts of errors occur when a non-optimal sequence is chosen, as a result of the correlation between neighbouring symbols. The dependency between adjacent symbols makes it hard for an alternate sequence to be selected using this sampling strategy.
- **Delay Ambiguity.** The channel and sequence are shifted by one or more time-steps. Again the strong dependencies inherent in the sampling strategy described above do not favour time-shifts.

For many applications in statistics there is plenty of computational time available to process data. In these cases, running the Markov chain for a large number of iterations is often a practical solution. However for the method to be useful in a practical telecommunications system, we need to modify the algorithm. In this case this means seeking an alternate Markov chain which mixes better and converges faster by introducing alternative moves.

5.2 Joint Sampling

Let us first consider the problem of burst errors. As mentioned above, the problem arises because we make draws for the individual symbols, dependent upon those immediately adjacent. It is hoped that performing joint sampling of all the symbols will improve matters (section 4.3.3.1).

5.2.1 The Simulation Smoother

The number of possible sequences increases exponentially with the length of the data. It is therefore not feasible to calculate and sample from the probabilities of each and every sequence for any moderate lengths of data. Instead, an alternative algorithm is proposed whose computational cost is linear in the data length.

The joint draw from $p(x|y, h, \sigma_v^2)$ is achieved by a forward and backward pass through the data. The procedure is essentially a simulation version of the Kalman filter/smoothing for discrete data [90]. Related methods, mostly derived in the context of Gaussian mixtures, can be found in [20, 43, 89, 35]. The forward pass calculates predictive and corrective probabilities for each data point in turn. The random draws themselves are performed on the backward pass. The method proceeds as follows:

Forward Pass. Step through the data in the frame in a forward direction performing these steps:

1. Calculate the probability of the states at time $t + 1$ given the observations up to the current time. This is sometimes called the *prior* or *prediction* step.

$$p(\mathbf{x}_{t+1} | y_{1:t}) = \sum_{\mathbf{x}_t} p(\mathbf{x}_{t+1} | \mathbf{x}_t, y_{1:t}) p(\mathbf{x}_t | y_{1:t}). \quad (5.13)$$

This equation agrees with intuitive reasoning, that the probability of being in a particular state at time $t + 1$ is given by the sum of the probabilities for all possible previous states, weighted by the transition probabilities. Note that this does not tell us anything about any future *symbols* in a communications system, only the states. The transition probabilities basically say whether a particular state can be reached from the current one or not.

2. Recalculate the probabilities (subject to normalisation) given the additional observation at time $t + 1$. This is called the *update* or *posterior* step.

$$p(\mathbf{x}_{t+1} | y_{1:t+1}) \propto p(y_{t+1} | \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} | y_{1:t}) \quad (5.14)$$

The values of both $p(\mathbf{x}_{t+1} | y_{1:t})$ and $p(\mathbf{x}_{t+1} | y_{1:t+1})$ are stored for use in the backward pass — there is only a finite number of states because the communications system has a discrete state space.

Backward Pass. Firstly, note the following factorisation:

$$p(\mathbf{x}_{1:N} | y_{1:N}) = p(\mathbf{x}_N | y_{1:N}) \prod_{i=1}^{N-1} p(\mathbf{x}_i | \mathbf{x}_{i+1:N}, y_{1:N}) \quad (5.15)$$

The states may be sampled by drawing from each of the probability densities listed in the product of equation (5.15) in reverse order (from \mathbf{x}_N to \mathbf{x}_1). However, for backward transitions, the state \mathbf{x}_{t+1} summarises all the information contained in the states $\mathbf{x}_{t+2:N}$ and $y_{t+1:N}$, so

$$p(\mathbf{x}_t | \mathbf{x}_{t+1:N}, y_{1:N}) = p(\mathbf{x}_t | \mathbf{x}_{t+1}, y_{1:t}) \quad (5.16)$$

$$= \frac{p(\mathbf{x}_{t+1} | \mathbf{x}_t, y_{1:t}) p(\mathbf{x}_t | y_{1:t})}{p(\mathbf{x}_{t+1} | y_{1:t})} \quad (5.17)$$

$$= \frac{p(\mathbf{x}_{t+1} | \mathbf{x}_t) p(\mathbf{x}_t | y_{1:t})}{p(\mathbf{x}_{t+1} | y_{1:t})} \quad (5.18)$$

The transition probability, $p(\mathbf{x}_{t+1} | \mathbf{x}_t)$, is known since it describes the states reachable from the current state. The other terms have been calculated in the forward pass and stored, so we may obtain a joint draw in the following manner:

1. Draw x_N from $p(x_N | y_{1:N})$.
2. For $t = N - 1, N - 2, \dots, 1$
 - (a) Draw the component x_t from:

$$p(x_t | x_{t+1:N}, y_{1:N}) = \frac{p(x_{t+1} | x_t) p(x_t | y_{1:t})}{p(x_{t+1} | y_{1:t})} \quad (5.19)$$

- (b) Now the symbol, u_t , may be uniquely determined from the state, x_t since $u_t = x_t[1]$.

5.3 Overcoming Delay Ambiguity

The problem of *delay ambiguity* is well known in the communications literature. This ambiguity arises since shifting the sequence one or more steps in time can be compensated for by shifting the channel coefficients by the same amount. For example, if the actual channel is $[0.8, 0.5, 0.2]^T$ the estimated channel may be $[0, 0.8, 0.5]^T$. Ambiguities are particularly common if either the filter length of the channel used in modelling is longer than the real (or synthetic) channel, or if there is little energy in some of the outer filter taps. The symbol-based sampler of Chen and Li [22] is particularly susceptible to this problem. The joint draw method outlined above is much better although problems still arise, commonly due to poor initialisation or convergence to local maximum.

The shift in the output data is not necessarily a major problem, however it becomes one if either the delay varies often from frame to frame, causing repeated loss of synchronisation, or if the delay ambiguity in turn causes poor equaliser performance.

To combat this problem, a number of methods have been proposed in the literature:

- If most of the energy is in the initial taps, dynamically increasing the filter length as described by Wesolowski [182] can gain benefits. By choosing a model order too low, the aim is to capture only the taps with the most energy. Once approximate convergence to these has been obtained, the filter length may be increased to capture some of the finer detail of the filter structure.

- Parallel structures for channel estimation may be used each offset by one time step. This has been used for the DFE [16, 17] and the structure producing lowest mean squared error is chosen as having converged to the correct solution. This has a major advantage in that it exploits our prior knowledge that the channel is almost correct, just out by a time-shift.
- Initialisation is often critical to correct convergence. In [148] the Gibbs sampler is run once initially and then restarted with initial conditions such that the channel estimate (or convolution wavelet in their case) is shifted. The version with the lowest estimated noise variance is chosen to give the results.

5.3.1 Reversible Jumps

The concept of reversible jumps was introduced to invoke changes between models of different order. However, their use is not limited to varying model order. Any proposal may be used to allow moves about the probability space, so as to aid convergence or exploration of the space.

5.3.1.1 Theory

The method of reversible jumps was formally introduced by Green [69]. It is essentially an extension of the Metropolis-Hastings algorithm (section 4.3.2.1) to allow for proposals of different dimensionality to the probability space in which we are interested. An auxiliary variable, z , is introduced which is used in the simulation part of the process. This is drawn from the density $p(z)$. The form of this density may vary, and is often dependent on the dimension and value of the current state. This will be denoted by using a superscript to the p.d.f., according to whether the density refers to that at a particular iteration, e.g. $p^{(i)}(\cdot)$, or the proposed move, $p^*(\cdot)$. The proposed state is then obtained by the deterministic mapping:

$$x^* = m(x^{(i-1)}, z^*) \quad (5.20)$$

Simulating the proposed new state in this way introduces a Jacobian term into the acceptance probability for the proposed move.

The auxiliary variable is used to match dimensions between the forward and

reverse moves: the sum of the dimensions of the current state and proposed auxiliary variable ($x^{(i-1)}$ and z^*) equal those for a reverse jump (x^* and $z^{(i-1)}$), where $z^{(i-1)}$ denotes the value of z that would be required to perform a reverse jump (i.e. the value of z that satisfies $x^{(i-1)} = m(x^*, z^{(i-1)})$).

The reversible jump technique allows a larger variety of moves to be proposed, which may be based upon reasonable assumptions rather than just a particular functional form. The modified algorithm proceeds as follows:

1. Draw the auxiliary random variable, z^* from $p^*(z)$.
2. Calculate new state from the mapping, $x^* = m(x^{(i-1)}, z^*)$.
3. Draw a uniformly distributed random number, u , between 0 and 1.
4. **If** $u < \alpha(x^* | x^{(i-1)})$ **then** accept the move by setting $x^{(i)} = x^*$
else set $x^{(i)} = x^{(i-1)}$.

The acceptance probability, α , is given by:

$$\alpha(x^* | x^{(i-1)}) = \begin{cases} r(x^* | x^{(i-1)}) & r(x^* | x^{(i-1)}) \leq 1 \\ 1 & r(x^* | x^{(i-1)}) > 1 \end{cases} \quad (5.21)$$

where the ratio $r(x^* | x^{(i-1)})$ is given by:

$$\begin{aligned} r(x^* | x^{(i-1)}) &= \frac{\pi^*(x^*)}{\pi^{(i-1)}(x^{(i-1)})} \frac{p^{(i-1)}(z^{(i-1)})}{p^*(z^*)} \left| \frac{\partial(x^*, z^*)}{\partial(x^{(i-1)}, z^{(i-1)})} \right| \\ &= (\text{posterior ratio}) \times (\text{proposal ratio}) \times (\text{Jacobian}) \end{aligned} \quad (5.22)$$

where $\pi^{(\cdot)}(x)$ denotes the posterior using the same superscript notation as for the proposal density. The dependency on i is due to possible changes in model order or in extreme cases, the whole model. The posterior may be decomposed into the product of prior and likelihood as usual.

It is worth noting that this algorithm is simplified if the proposals are made directly in the probability space. In other words, if $m(x^* | x^{(i-1)}, z^*) = z^*$, and dimensional changes are implicit in the proposal, the Jacobian is unity and the form of the acceptance probability is the same as for a normal Metropolis-Hastings step [61]. It is clear that the dimension matching criteria is also satisfied.

5.3.2 Reversible Jump Moves

A number of reversible jump moves may be proposed. The ones outlined below are used to attempt to overcome the problems of delay ambiguity, phase ambiguity and for model order selection i.e. determining the length of the channel.

5.3.2.1 Shift Moves

We know about, and expect delay ambiguity, therefore we may propose a delay shift as one of the moves to help the Markov chain to mix. If there is sufficient energy in the outer taps of the filter and the filter is of the correct length, we expect the chain to converge to the solution with the correct delay by sampling directly from the posterior distribution, without having to make heuristic decisions on which mode is preferred. Where there is genuine ambiguity, we expect to obtain data from each mode, in the appropriate proportions accorded by their posterior probability.

The shift move proposed takes the following form:

- Propose a direction for the shift (e.g. +1).
- Shift the data sequence by the mapping $x_{2:n}^* = x_{1:n-1}$.
- Choose a new symbol for the vacated end of the sequence: $x_1^*[1] = S_i$ for some choice of i .
- Draw a new channel vector from the full conditional density, given the new sequence: $h^* \sim p(h|x^*, \sigma_v^2)$

The acceptance probability is calculated in the normal manner, and the move accepted or rejected. In this case z would describe x_1^* , h^* and the direction of the shift.

The direction of the shift may be proposed randomly (and uniformly). In this application we have only low model orders, so it does not make sense to propose a shift of any size other than one time-step. If the length of the channel were much larger, as in the example of [148], it would probably be advantageous to propose moves of several time-steps.

There are two obvious choices for the distribution of the new symbol, namely, uniform or the full conditional distribution, given by equation (5.12). When there

is a large number of symbols, the latter is preferred, however it does not matter so much when there are only two or four symbols in the alphabet, and this simplifies the computation required.

5.3.2.2 Model Order Selection

Model order selection using reversible jumps is a familiar process. A similar framework is adopted as used in [173, 174] for AR models. Changes in model order are proposed from a discretised Laplacian distribution:

$$p(l^* | l^{(i-1)}) \propto \exp(-\lambda |l^* - l^{(i-1)}|) \quad (5.23)$$

The new channel parameters are drawn from the full conditional. This produces a good candidate channel and allows for simplification of the acceptance probability. This simplification may be done either by using the relationship,

$$\frac{p(l, \mathbf{h} | y, \mathbf{x}, \sigma_v^2)}{p(\mathbf{h} | l, y, \mathbf{x}, \sigma_v^2)} = p(l | y, \mathbf{x}, \sigma_v^2), \quad (5.24)$$

or by direct algebraic manipulation of the density ratio.

Priors It seems sensible to put an upper limit on the model order. If nothing else, this reduces computational time, since the complexity of joint sampling rises exponentially with model order. The Bayesian framework should automatically penalise complexity, so we should not have to specify a prior which does so explicitly. However, we may wish to do so if we genuinely seek a simple model and therefore, we choose the prior:

$$p(l) \propto \begin{cases} \zeta^{-l} & \text{if } l \leq l_{max} \\ 0 & \text{if } l > l_{max} \text{ or } l < 1 \end{cases} \quad (5.25)$$

for some $\zeta \geq 1$.

The prior for the channel introduces an interesting paradox, (Lindley's paradox, [102, 14]): if we choose a highly non-informative prior for the channel, that is one with large variance, this places a fairly strong prior on the model order, favouring low order models. An informative prior removes this bias. This may

be overcome to some extent by using a hyper-prior (section 4.1.4.4), where we choose a functional form for the prior, but do not specify its variance. The variance is estimated as a parameter in the course of the MCMC procedure.

5.3.2.3 Flip Move

The flip move was introduced in an attempt to nullify the requirement for differential encoding and resolve the phase ambiguity problem when no training symbols are present. The move consists of a proposal to rotate the phase of all the symbols to map onto another set of plausible symbols, followed by the phase rotation of the channel of the same magnitude in the opposite direction. The name “flip” arises since with a binary coding scheme, it involves the flipping of all the bits.

At a first glance all possible phase offsets should be equally likely, and the move is pointless. This neglects the fact that estimated symbols from the end of the previous frame are being fed into the start of the current one. These are not phase rotated. It is hoped that these few symbols will be sufficient to resolve the ambiguity and the correct phase is chosen.

The flip move has one other use when dealing with variable model order. The symbols from the previous frame may cause the incorrect model order to be selected if the current draw for the channel has a different phase to the previous frame. For example, consider a channel of length 3 and a binary PAM scheme. Two symbols are fed in at the start of the sequence, let them be +1 followed by -1. If the current draw of the channel is of opposite sign to the previous frame, these ought really to be -1 followed by +1. A better fit may be obtained if a channel of length 4 is chosen with the first tap close to zero. This effectively eliminates the contribution of the first symbol, which is perceived to be in error and halves the effect of the second symbol.

It is implemented as follows:

- Choose a phase rotation for the symbol sequence, ϕ .
- Rotate the symbol sequence by that phase, $x^* = e^{j\phi} x$.
- Draw a new value for the channel conditional on new sequence: $h^* \sim p(h|x^*, \sigma_v^2)$

5.3.2.4 Combination Moves

In certain cases it is useful to propose moves which are a combination of the moves above. Perhaps the best example is to increase the channel length by one and to add an additional symbol delay or vice-versa. For example a proposal to move from

$$\mathbf{h} = [a, b, c]^T \rightarrow \mathbf{h} = [x, a, b, c]^T \quad (5.26)$$

can take place in one jump rather than two separate ones. If these combination moves are proposed as well as moves which change the model-order only, the algorithm is effectively allowed to extend the channel length at either end, without the requirement for an intermediate state. The net effect should be improved mixing of the Markov chain.

5.3.2.5 Acceptance Ratio

It turns out that the basic acceptance ratio for every one of these moves (shift, phase and flip) is given by:

$$\alpha = \alpha_p \frac{|\Sigma_H^{-1}| \exp\{-\{\mu_H^T \Sigma_H^{-1} \mu_H\}}{|\Sigma_H^*| \exp\{-\{(\mu_H^*)^T (\Sigma_H^*)^{-1} \mu_H^*\}} \quad (5.27)$$

where Σ_H^{-1} and μ_H are given by equation 5.10, with the starred versions corresponding to substituting the values of x^* into the equation.

This simplification arises from the fact that most of the proposals for the states are deterministic, and the new values of the channel are proposed from the full conditional distribution. The factor α_p is due to the ratio of proposal densities not defined by the deterministic shifts or the new value of the channel. This includes the choice of the new symbol in shift or combination moves, the choice of direction or size of shift, and the proposal for varying model order. Having said that, the model order proposal in equation (5.23) is symmetrical, i.e. a forward and reverse proposal have the same probability, so $\alpha_p = 1$ for proposed changes in model order.

5.4 Results

5.4.1 Implementational Issues

Synthetic data is used, generated directly from the FIR filter channel model. This allows direct comparisons to be made in terms of how well the states or parameters are estimated rather than looking at the limitations of this very popular model on real data. Of course, in a real system where the underlying data does not match the model used for estimation, a degradation in performance is expected.

Differential encoding may be used where appropriate. This is done on a frame by frame basis so that the phase ambiguity may be resolved differently in each frame without affecting the overall results.

5.4.1.1 Initialisation

Initial values can be obtained by making draws from the prior distributions, however it is sometimes convenient to start with initial values generated by other methods to aid convergence or consistency of convergence in a short time. For example when training data is present, an estimate of the channel filter coefficients could be used as a starting point, which would greatly improve convergence time. The methods employed in obtaining the results displayed are outlined below:

For a single frame or the first frame in a sequence the following choice of initialisation is made:

- The symbol sequence is generated from the prior, i.e. it is completely random.
- The channel is chosen to be $[1, 0, 0, \dots]^T$. This is the ideal response if there is no ISI.
- The noise variance is generated conditional on the channel, symbol sequence and observations — a normal Gibbs sampler move.

For a second or subsequent frame in a sequence:

- The channel and noise variance are carried forward from their estimated values on the last frame. This is similar to *pre-loading* in conventional adaptive equalisers.
- The symbol sequence is drawn conditional on the channel, noise variance and observations — a normal Gibbs sampler move.

This gives improved convergence time and often keeps the phase ambiguity consistent between frames.

5.4.1.2 Priors

The variance of the channel prior was set at $\sigma_h = 40$, and the hyper-parameters for the inverted gamma distribution of the noise variance, α and β , have been both set at 10^{-3} . In both cases, these values are uninformative enough not to influence the results, but informative enough to provide initial draws in a reasonable range when required.

5.4.1.3 Order of Sampling

The sampling is done in a deterministic scan of σ_v^2 , h and x or each u_t in turn in ascending order, as appropriate (depending on whether the sequence-based or symbol-based version is used).

5.4.1.4 Frame Boundaries

Care must be taken at the frame boundaries to ensure that the results are not entirely dependent on the edge conditions. To this end, the *a posteriori* probabilities of the symbols from the previous frame are used to as prior information for the next frame. The number of symbols for which this is required is $l - 1$. At the other end, an additional $l - 1$ observations are used and the same number of extra symbols are used within the Gibbs sampler. This is illustrated in Figure 5.2. The frames are not overlapping: the centre parts of adjacent frames touch, and only the centre part is used for the estimation.

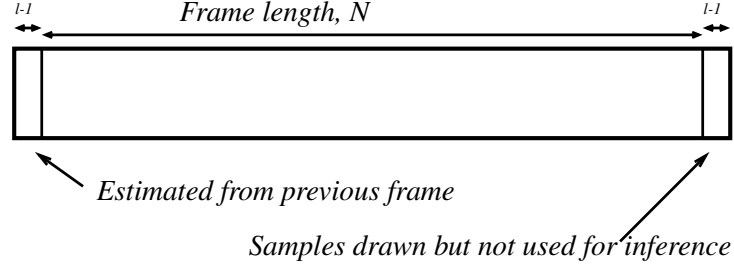


Figure 5.2. Diagram showing how frames boundaries are dealt with.

5.4.1.5 Reversible Jumps

Reversible jumps are only proposed according to the following scheme:

- A flat prior for model order is chosen, i.e. $\zeta = 1$ in equation (5.25). The acceptance ratio does not need to be adjusted because this prior selects both forward and reverse moves with equal probability.
- The model order prior is characterised by setting $\lambda = 2$ in equation (5.23), with the maximum model order given by $l_{max} = 10$.
- New symbols are chosen in the shift moves by using likelihood based proposals. For example, for a +1 shift, a new value of x_1 is chosen by drawing for $x_1[1]$ using:

$$p(x_1[1] = S_i) = \frac{p(y | (x^* | x_1^*[1] = S_i, x_{2:N}^* = x_{1:N-1}), h, \sigma_v^2)}{\sum_{j=1}^q p(y | x_{1:N} = (x^* | x_1^*[1] = S_j, x_{2:N}^* = x_{1:N-1}), h, \sigma_v^2)} \quad (5.28)$$

with $x_1[2 : l]$ uniquely determined by x_2 . This results in:

$$\alpha_p = \frac{\frac{p(y | x, h, \sigma_v^2)}{\sum_{j=1}^q p(y_{1:N} | (x | x_1[1] = S_j), h, \sigma_v^2)}}{p(y | (x^* | x_1^*[1] = S_i, x_{2:N}^* = x_{1:N-1}), h, \sigma_v^2)} \frac{\sum_{k=1}^q p(y_{1:N} | x_{1:N} = (x^* | x_1^*[1] = S_k, x_{2:N}^* = x_{1:N-1}), h, \sigma_v^2)}{p(y | (x^* | x_1^*[1] = S_i, x_{2:N}^* = x_{1:N-1}), h, \sigma_v^2)} \quad (5.29)$$

- If no change in model order is proposed, then the choice of the direction of

the shift is given by:

$$\text{Direction} = \begin{cases} +1 & \text{with probability 0.5} \\ -1 & \text{with probability 0.5} \end{cases}$$

If an increase in model order is proposed, then the direction of the shift is chosen as:

$$\text{Direction} = \begin{cases} +1 & \text{with probability 0.4} \\ \text{no shift} & \text{with probability 0.4} \\ -1 & \text{with probability 0.2} \end{cases}$$

If a decrease in model order is proposed, then the direction of the shift is chosen as:

$$\text{Direction} = \begin{cases} +1 & \text{with probability 0.2} \\ \text{no shift} & \text{with probability 0.4} \\ -1 & \text{with probability 0.4} \end{cases}$$

This favours the idea of adding a non-zero coefficient at either end of the channel filter as discussed in section 5.3.2.4.

- No reversible jumps take place in the first 10 iterations. This allows the sampler to converge to one of the local maxima more easily, since the reversible jumps are only really designed to move between local maxima.
- If reversible jumps are proposed, then five proposals for each of time-shift and combination moves (time-shift and model order) are proposed at each iteration. This is done because the reversible jump proposals are relatively cheap to calculate compared to the joint draw of the symbol sequence. In addition most proposals are unlikely to be accepted since they propose moves to a local maxima with relatively low probability.
- No flip moves are proposed at any time after 20 iterations before the end of the burn-in period. In certain cases (for example no known symbols, even from a previous frame) the sign of the channel is genuinely ambiguous. This

usually results in a flip at every time step producing symbol estimates that are difficult to interpret.

5.4.2 Convergence

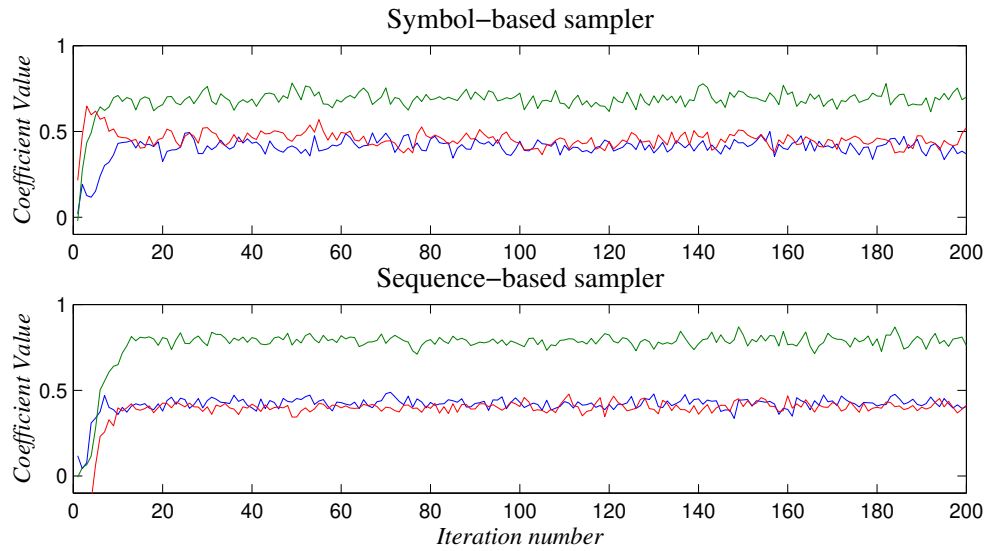
In this section the convergence of the three implementations of the Gibbs sampler will be compared: the symbol-based sampler, the joint (sequence-based) sampler and the joint sampler with reversible jumps for time-delay only. These graphs show sample results of blind equalisation on a single frame of data highlighting the advantages the improved samplers bring. No differential encoding is used, so there can be an arbitrary sign change (due to phase ambiguity). This has been corrected by hand before the results were plotted. The noise variance in each simulation is, $\sigma_v^2 = 0.1^2 = 0.3162$.

The number of errors within each frame of 200 symbols is obtained by virtue of the fact that synthetic data is used: this statistic would not be available in a real application. It has been suggested that the draws of the noise standard deviation can be used as a guide for indicating convergence. This will be investigated.

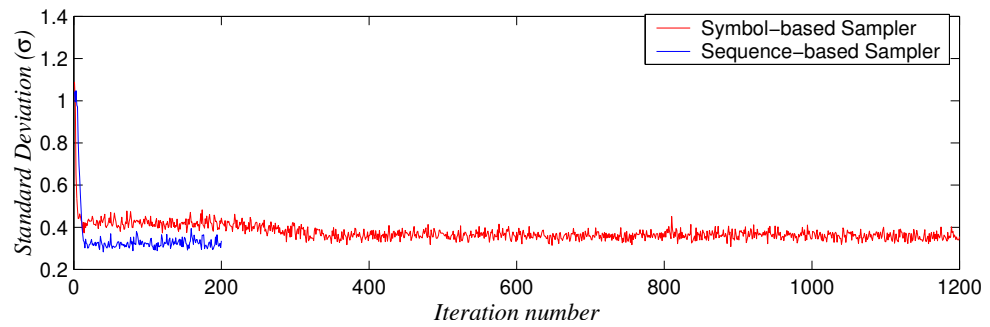
First let us look at the results of the symbol-based sampler and the sequence sampler on channel A.1 shown in figure 5.3. In this example all the channel taps have significant energy. This has the effect that there are no problems with delay ambiguity: both samplers converge to the correct channel within a few iterations as illustrated in figure 5.3(a). Only the first 200 iterations of the sampler are shown. By looking at this information alone, we can say little except that the variance on the estimated values for these coefficients is perhaps a little more for the symbol-based sampler, especially in the first 70 iterations. Convergence to approximately the correct values occurs in just 10–15 iterations in both cases.

If we look at the standard deviation of the noise in figure 5.3(b) we might infer that although the sequence-based sampler (blue) probably reaches convergence quickly, within 10–20 iterations, the symbol-based variant may not have reached it until around iteration 400. Indeed if we compare the two sets of random draws, it suggests that convergence may not have been reached at all by the symbol-based sampler as its estimated noise standard deviation is larger than that of the sequence-based sampler.

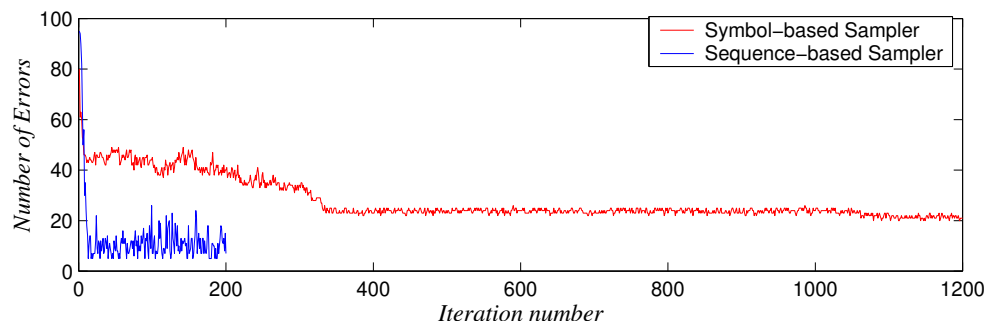
The graph of number of errors (figure 5.3(c)) reveals the true story. The



(a) Simulated Parameter Values — Blue: coefficient 1 (correct value 0.407), green: coefficient 2 (correct value 0.815), red: coefficient 3 (correct value 0.407).



(b) Simulated observation noise standard deviation (correct value 0.3162).



(c) Number of Errors in frame.

Figure 5.3. Convergence results for channel A.1 (fixed model order).

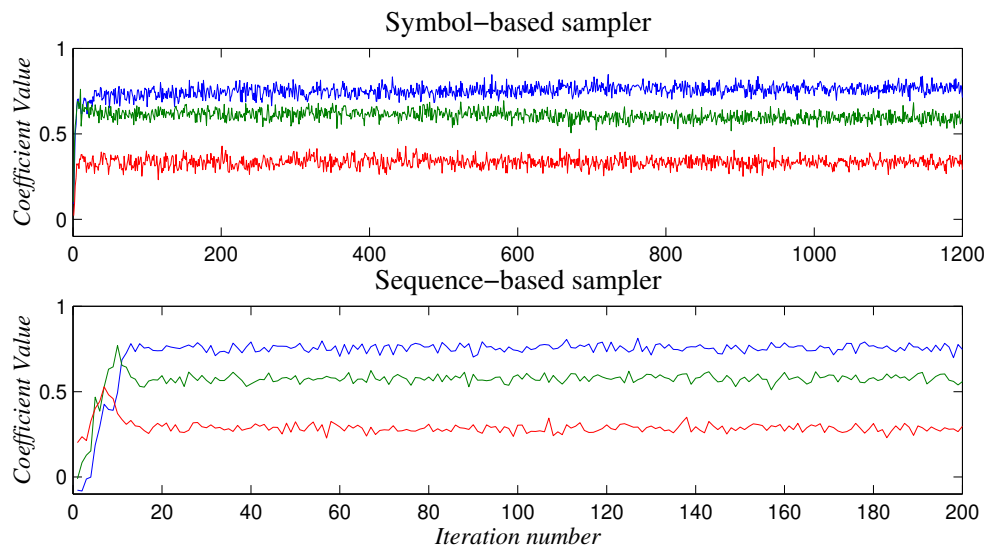
sequence-based sampler does indeed achieve convergence to the global maximum of posterior probability well within twenty iterations. The symbol-based sampler does not achieve convergence even over the 1200 iterations that it was allowed. It does reach a local maximum at around iteration 375, but by careful examination it is possible to see that a further few errors are removed at around iteration 1050. The error rate is still about double that possible if the global maximum were to be obtained. These extra errors are the cause of increased variance in the draws of the channel coefficients and an over-estimate of the noise variance. In [22], where a very similar Gibbs sampler is described, several thousand iterations are required for convergence to the global maximum.

Figure 5.4 shows similar plots for channel A.2. This channel is easier to equalise, hence the improved performance at the same signal to noise ratio. We can see a slow convergence to the correct coefficient values in 5.4(a) over the 1200 iterations for the symbol-based sampler. The sequence-based sampler appears to converge within 20 iterations (note that these graphs have a different scale on the x-axis).

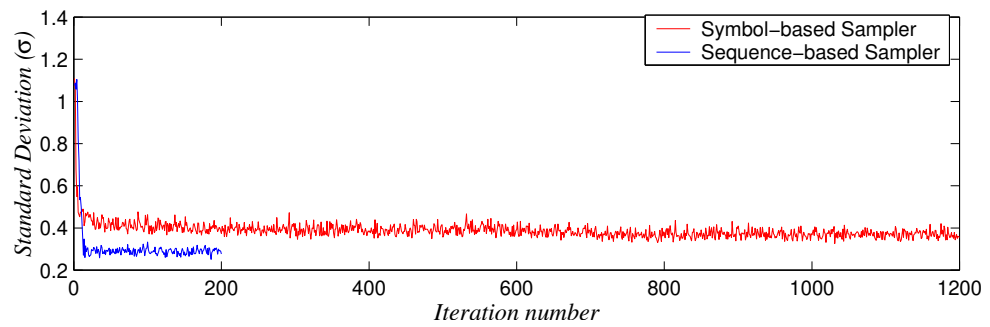
This behaviour is paralleled in the estimate of the noise standard deviation illustrated in Figure 5.4(b) and in the number of errors in Figure 5.4(c). Since the equalisation is easier, less noise is seen on the estimates and it is then clear that convergence of the symbol-based sampler is too slow to be of practical use.

We shall now turn our attention to channels which have outer taps with low energy. Recall that these channels are particularly susceptible to delay ambiguity. Such an example is Figure 5.5 which shows a similar graphs for channel A.3. Figure 5.5(a) shows the draws of the noise standard deviation. It appears from this graph that all the samplers have converged after 50 iterations, even with the benefit of seeing the results from different samplers. However in this case each of the three samplers have converged to a different delay shift, evident from Figure 5.5(c). The correct solution is that obtained when reversible jumps are employed, as shown in Figure 5.5(b).

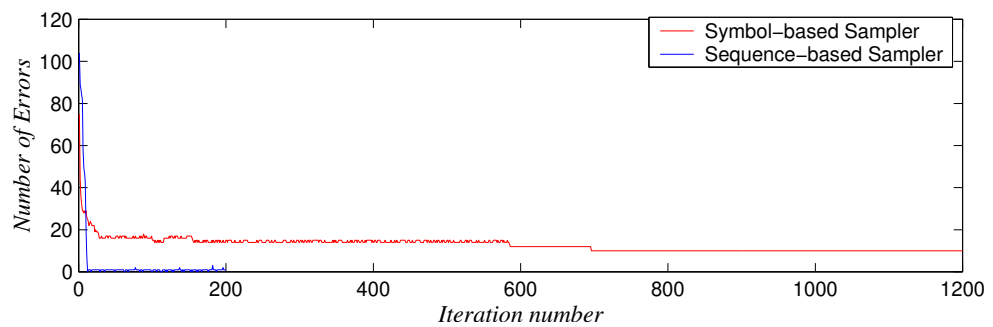
We can get an insight into the strengths and weaknesses of the different algorithms from Figure 5.5(c). The highest power coefficient of the channel should be the second. The symbol-based algorithm converges to the solution with the highest power in the first coefficient, and shows no signs of moving from this local maximum. The sequence-based sampler chooses the fourth coefficient for maxi-



(a) Simulated Parameter Values — Blue: coefficient 1 (correct value 0.766), green: coefficient 2 (correct value 0.575), red: coefficient 3 (correct value 0.287).

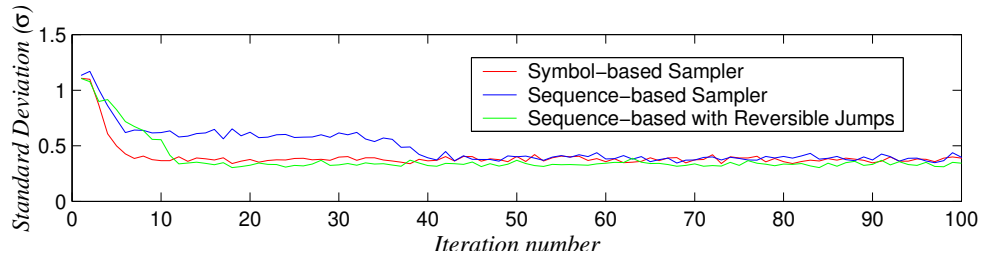


(b) Simulated observation noise standard deviation (correct value 0.3162).

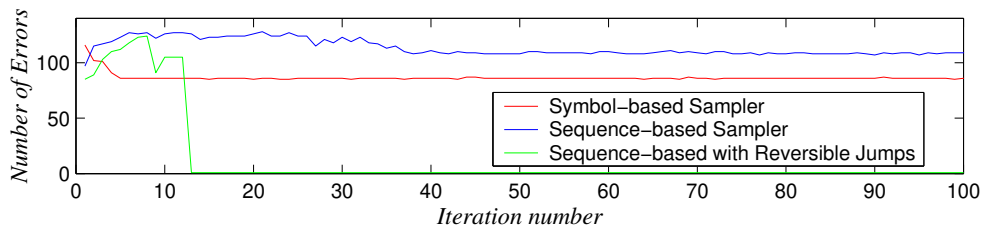


(c) Number of Errors in frame.

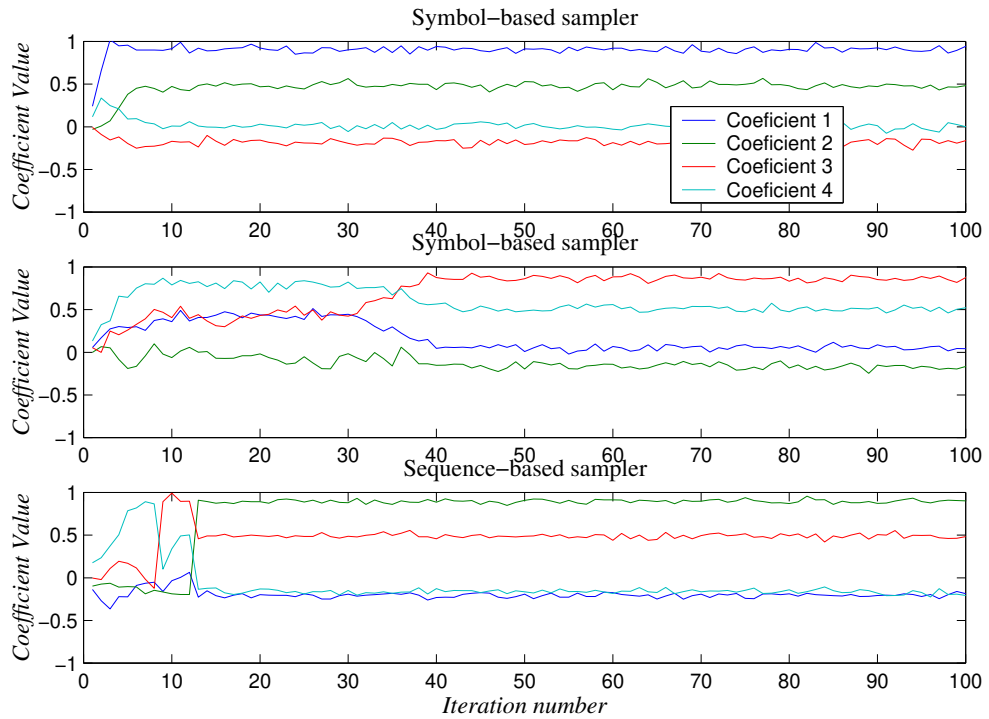
Figure 5.4. Convergence results for channel A.2 (fixed model order).



(a) Simulated observation noise standard deviation (correct value 0.3162).



(b) Number of Errors in frame.



(c) Simulated Parameter Values — Blue: coefficient 1, green: coefficient 2, red: coefficient 3, cyan: coefficient 4.

Figure 5.5. Convergence results for channel A.3 (\mathcal{L} ed model order).

imum power initially, however it moves to the third coefficient between iterations 30 and 40. Although it is not shown, in this example the sampler was run for 2000 iterations and no further time-shift occurred.

Likewise when reversible jumps are employed, the highest power is in the fourth tap initially. The movement to the correct second tap takes place by shift move reversible jumps occurring at iterations 9 and 13. Within a few more iterations, convergence to the global maximum of the probability space is achieved.

It should be noted that these plots are produced from a single frame of 200 data points — different frames may produce different results, usually dependent on which tap contains the highest power when convergence to either one of the local maxima or the global maximum is first achieved. This may depend upon initial conditions or just chance. In fact, without the reversible jumps, the sequence-based sampler converges to the correct solution most ($\approx 90\%$) of the time. It also performs time-shift moves, as illustrated when moving the highest energy tap from the fourth to third tap between iterations 30 and 40, provided there is sufficient energy in the outer taps. Indeed this ambiguity can also affect the simpler channel A.2, although convergence to the global minimum is usually achieved if a couple of hundred iterations are allowed for convergence.

However, delay ambiguity results in complete loss of the frame of data if no other synchronisation method is available. If synchronisation bits were available, they could be used to ensure correct convergence of the sampler. In addition, in the presence of higher levels of noise the sequence-based sampler does perform time-delay shifts without the need for reversible jumps. However, these are typically hundreds of iterations apart, a time-scale too large for practical systems.

5.4.3 Trained (Adaptive) Equalisation

Let us now investigate the overall performance of the Gibbs sampler as an adaptive equaliser and compare its performance to standard algorithms. The equaliser will operate on blocks of data of length 200 with 13 training symbols placed at the start of each block. These are chosen to be Barker length 13 synchronisation symbols [4]:

$$1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1 \quad (5.30)$$

The frames are overlapped slightly as described in section 5.4.1.4. This overlap allows the forward pass of the Gibbs sampler or the Viterbi algorithm to terminate in a known state, because the additional symbols form part of the next training sequence.

For the Gibbs sampler the model order is assumed unknown. 40 iterations are allowed for convergence, the next 11 being used for inference. Reversible jumps for both delay shifts and model order are employed, but no phase flip moves, since they should not be required when training data is available. The sequence chosen is that which maximises the marginal posterior probability. This is calculated simply by counting the number of times each symbol from $\{S_1, S_2, \dots, S_q\}$ occurs in those iterations used for inference, and choosing the symbol with the highest count, for each time step.

Since the Gibbs sampler is of high complexity it makes sense to compare its performance with the Viterbi algorithm. The Viterbi algorithm is a MAP estimator given the value of the channel. There are four possible choices for the channel estimates:

- **The correct channel** — since the data is synthetic, we know the actual value of the channel. This “exact” method should give the optimum solution, i.e. a best possible solution of the symbol sequence using a probabilistic method, since the value of the channel will not be known in any practical system.
- **Least Squares estimate** — this involves a matrix pseudo-inverse in order to make the best estimate (in the least squares sense) of the channel given the training sequence and the associated observations.
- **Least Mean Squares estimate over training symbols** — similar to above, except the normalised LMS algorithm (section 3.2.2) is used to approximate the LS estimate from the training sequence alone.
- **Least Mean Squares estimate over all symbols** — again the normalised LMS technique is used, except once the initial training is complete, the algorithm is used in decision-directed mode. This allows the estimated symbols to be used to help channel estimation, hopefully improving that estimate.

In all the above cases the model order is assumed to be known.

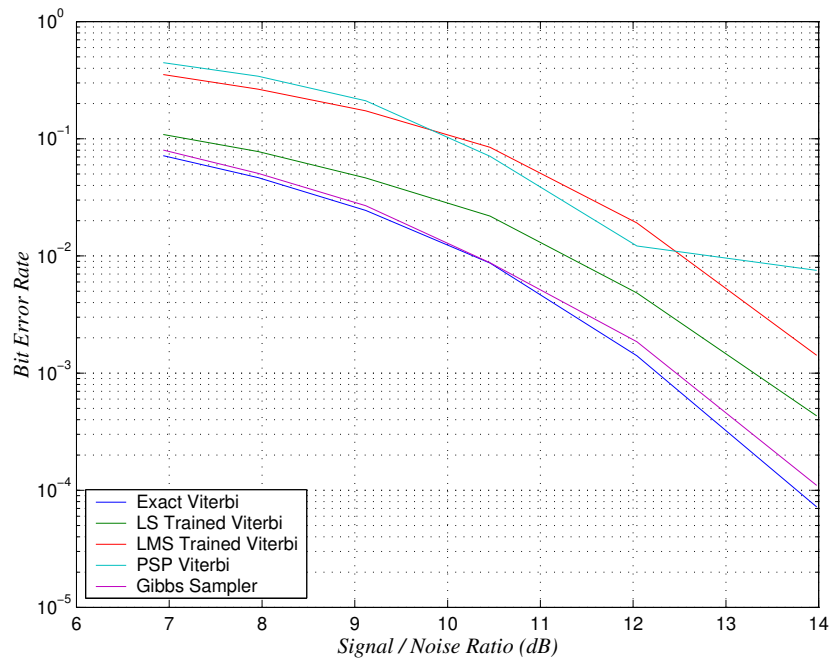


Figure 5.6. Plot of bit error rate versus signal to noise ratio for trained algorithms on channel A.1.

The overall performance of the system is given in the form of a graph of bit error rate against signal to noise ratio in Figures 5.6 and 5.7 for channels A.1 and A.3 respectively. The Gibbs sampler, despite not knowing the noise variance, the channel, or even its length, performs only marginally worse than the Viterbi algorithm with a known channel, in the presence of training data. The use of the LMS algorithm for estimating the channel or even a pseudo-inverse (exact Least-Squares) solution results in noticeably degraded performance.

The decision-directed LMS algorithm is disappointing, appearing to have a limit in the performance at an error rate of around 10^{-2} . This roll-off was present in all channels tested.

5.4.4 Blind Equalisation

The problem of blind equalisation is much harder, and it is interesting to see how much can be inferred from the received data. Differential encoding is normally employed to resolve phase ambiguity, and this is highly effective, except that a single error after equalisation results in two errors after differential decoding. We

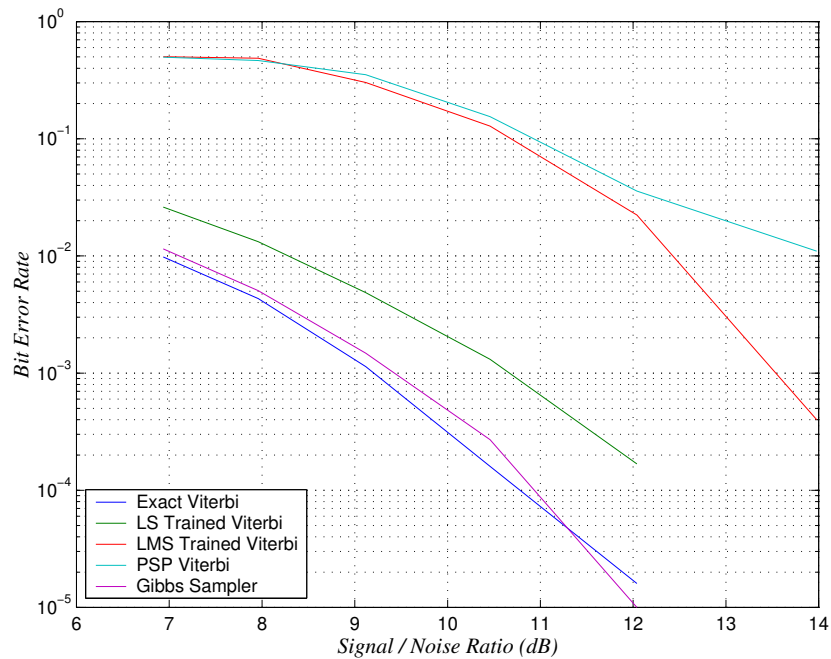


Figure 5.7. Plot of bit error rate versus signal to noise ratio for trained algorithms on channel A.3.

shall look at blind equalisation when differential encoding is employed and compare this with the use of flip moves and so-called semi-blind methods, where a small number of training symbols are transmitted.

If the model order is allowed to vary, the performance of the Gibbs sampler is very poor, as will be seen in section 5.4.4.2, on semi-blind equalisation. Therefore the model order is fixed in the first examples. The performance of the various methods are shown in figures 5.8 and 5.9 for channels A.1 and A.3 respectively. The performance of the Gibbs sampler on differentially encoded data is very good.

5.4.4.1 Flip Moves

The flip moves are an attempt to resolve the phase ambiguity problem, at least partially: they may allow us to decode adjacent frames with the same phase, but the phase ambiguity for the complete system cannot be resolved. It has already been shown that exceptional performance is achieved with the use of differential encoding, however it is worth investigating if this is required. We can see that when the order of the channel is unknown, these flip moves are only partially

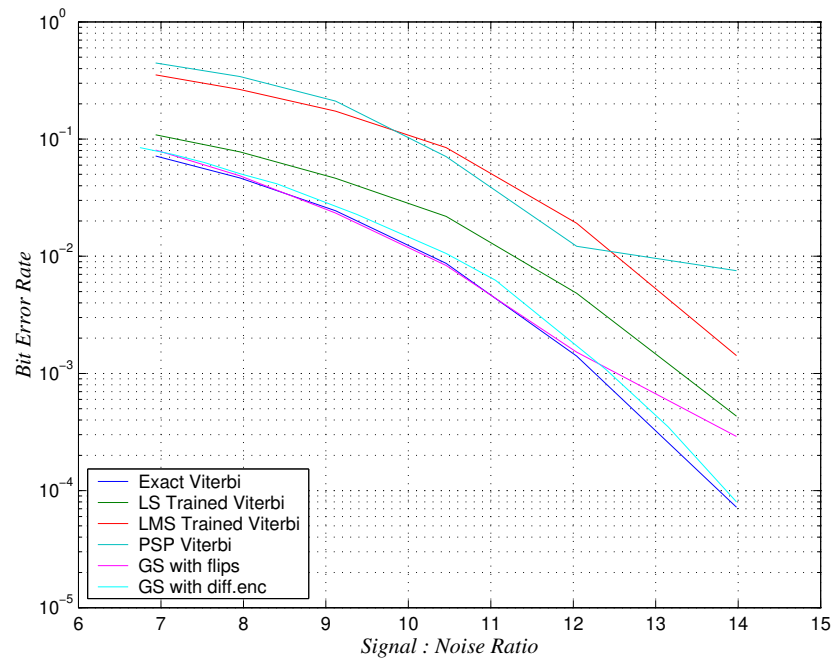


Figure 5.8. Plot of bit error rate versus signal to noise ratio for blind algorithms with fixed model order on channel A.1.

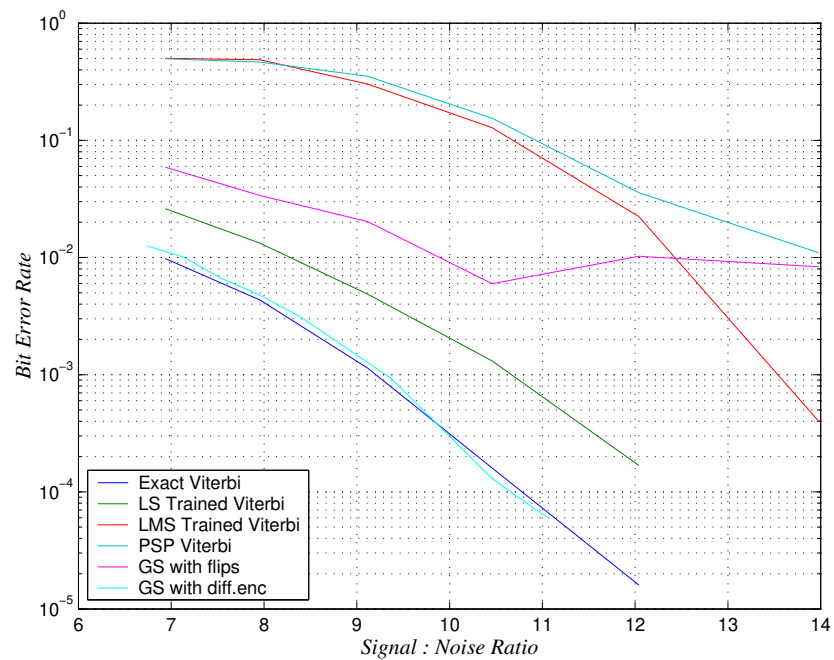


Figure 5.9. Plot of bit error rate versus signal to noise ratio for blind algorithms with fixed model order on channel A.3.

successful. Equalisation in this manner appears to be fine on a channel such as A.1 shown in Figure 5.8, however the longer channel A.4 displays disappointing performance (Figure 5.9).

The explanation for this is as follows: if the channel is susceptible to delay ambiguity (such as channels A.3 and A.4) then this allows a model order increase, time-delay shift and a slip of the sequence. For certain input sequences in the known³ bits from the overlap with the previous frame, this results in effectively removing the first of these bits.

For example:

$$\text{Bits from previous frame} = [1, -1, 1]^T \quad (5.31)$$

$$\text{Channel shift and increase in model order} = [a, b, c, d]^T \rightarrow [0, -a, -b, -c, -d]^T \quad (5.32)$$

The initial $[1, -1]^T$ of the bits from the previous frame match the last bits $[-1, 1]^T$ after the channel slip. The overall effect is that the estimated bit sequence is shifted to one time-step later, and therefore the entire frame is decoded incorrectly. This mechanism is largely responsible for the failure of the algorithm at high SNR — it is the occasional dropped frame, causing a large number of errors, rather than a equal degradation across the whole signal. The high SNR equates to a tight likelihood function (because of low noise variance). This appears to make any one mildly unlikely observation sufficient to disrupt the frame.

5.4.4.2 Semi-Blind Equalisation

It has been shown theoretically that only a small number of training symbols are required to obtain improved performance [31]. In this section comparisons are made between blind (no training symbols), semi-blind (four training symbols) and trained (thirteen training symbols) equalisation. Differential encoding is employed in all cases and the simulations were run for 100,000 data points. The training sequence of length four is chosen to be:

$$1, 1, 1, -1 \quad (5.33)$$

³Not known with complete certainty, but with a probability distribution strongly favouring one of the symbols.

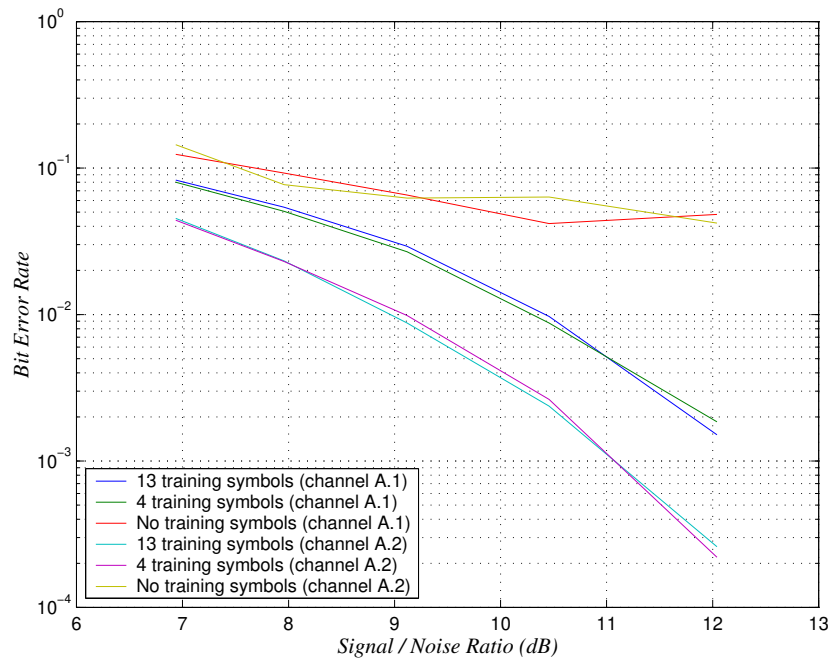


Figure 5.10. Plot of bit error rate versus signal to noise ratio for blind Gibbs sampler algorithm with unknown model order and a varying number of training symbols on channel A.1.

The results are shown in figures 5.10 and 5.11. In the case of channels A.1 and A.2 shown in figure 5.10, four training symbols are sufficient to achieve near optimal performance. This suggests that four training symbols may be sufficient for most channels of length three. The absolute performance clearly depends upon the exact nature of the channel. However by reducing the algorithm to being completely blind, we get unacceptable performance.

The length four channels A.3 and A.4 shown in figure 5.11 tell a different story: the performance is degraded with only four training symbols although still acceptable for the first result on channel A.4. However, repeating the simulation run led to a different result. Large changes in performance are due to the loss of one frame of data (delay ambiguity) — this explains the curve not being smooth. This is particularly pronounced on channel A.3 at high SNR. Repeating the run lead to repeated convergence to the wrong delay ambiguity and consequently, a catastrophic drop in performance. It is clear that 4 training symbols are insufficient to obtain optimum performance if the model order is not known on these channels. The evidence of both light performance degradation and frame drops suggests that

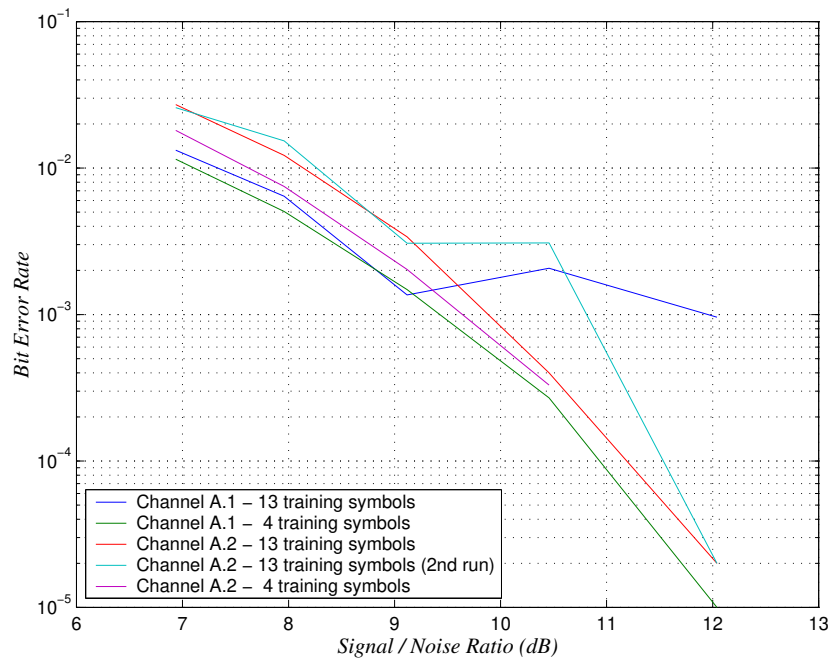


Figure 5.11. Plot of bit error rate versus signal to noise ratio for blind Gibbs sampler algorithm with fixed model order and a varying number of training symbols on channel A.4.

is a general problem for channels of length four, not just due to the susceptibility of these particular channels to delay ambiguity. It seems reasonable to suggest that you need more training symbols than the length of the channel.

5.5 Discussion and Conclusions

It has been demonstrated that the Gibbs sampler may be used to obtain near optimal performance at equalisation, both trained and blind.

- For trained equalisation there is sufficient information in a block of 200 BPSK symbols to infer model order and timing information.
- The number of training symbols required may be reduced to a figure as low as 2% of the transmitted data for shorter (length 3) channels, although more training than this is required if the channel is longer.

- For blind equalisation, the best performance is obtained if the model order is known or fixed and differential encoding is employed.
- The performance of the blind algorithm with known model order is very close to that when training data is present.
- There is no change to the algorithm whether it is in trained or blind mode: the only change is the prior knowledge on the symbol sequence.

This method can be extended for time-varying channels by incorporating a model for the evolution of h over time. However we have seen for an efficient Gibbs sampler, the values for all h_t over the frame length should be drawn jointly. This can be achieved by the method outlined in [20], however this restricts the choice of the time evolution model to be conditionally Gaussian. In addition this increases the computation required for each iteration significantly. Methods for tracking time-varying channels in a Bayesian setting will be discussed in the following chapters on particle filters.

The Gibbs Sampler can only be usefully used on a block of data. In a practical system, this imposes a latency of approximately twice⁴ the block length — typically a block will be 100–200 symbols. In some systems, such as those used for mud-pulse telemetry [166], this latency is intolerable. In these cases we would wish for an algorithm that can operate sequentially on each symbol as it arrives.

5.5.1 Gibbs Sampler Moves

- The use of joint sampling for the sequence reaps great rewards in the number of iterations required for convergence, typically reducing them by a factor of 100. It is always possible to use these techniques in the discrete state-space system inherent in digital communications systems. The complexity of the algorithm is given by

$$\text{Complexity} = O(N \times q^l) \quad (5.34)$$

⁴The first block of symbols must be buffered until a complete block is contained within the buffer. In a practical system we wish to utilise our hardware efficiently, so the processing time will be equal to the time it takes for the buffer to fill up. This results in a maximum latency of twice the block length. If this is insufficient time to process the block, a number of parallel processors may be employed, say N , each operating on one block of data for the time it takes to receive N symbols. This parallel structure will obviously increase the latency correspondingly.

where N is the length of the frame, q is the number of symbols in the constellation and l is the length of the channel. This may make the algorithm prohibitively expensive if the number of points in the constellation or the length of the channel is too large.

- Reversible jumps to overcome delay ambiguity are highly effective on the channels that are susceptible to this problem. In addition they do not incur much overhead compared to the joint sampling of the sequence.
- Reversible jumps for model order selection are relatively easy to implement and do not degrade the performance of the algorithm significantly if training data is present. This favours their usage in nearly all cases because of the increased flexibility of the model. The only proviso is that it is occasionally possible to over-estimate the channel order and get the time-delay wrong. The effect of the latter is a time-shifted sequence, which may be catastrophic.
- The use of \mathfrak{M} ip moves without differential encoding is generally a poor idea since differential encoding on its own produces better results. If for some reason differential encoding cannot be used and no training data is available, then \mathfrak{M} ip moves do provide a means of extracting moderate performance from the Gibbs sampler running in blind mode. However performance does not exceed an error rate of around 10^{-2} , even with high signal to noise ratios.
- The results shown here are using only 51 iterations in total, substantially less than the tens of thousands typically used in the statistics community. In many engineering applications, such as digital communications there is insufficient time to do that much processing on what is a very large amount of data, so it is therefore important that the performance is still very good despite the possibility of unconverged data and a small number of samples from which to make inference.

5.5.2 Comparison with Conventional Equalisers

- The Gibbs sampler performs exceptionally well, exceeding the benchmark Viterbi algorithm by a reasonable margin in many cases.
- The variation in results for the Viterbi algorithm using different methods for estimating the channel underlines the importance of this estimate being accurate and highlights the fact that the “optimality” of MLSE is always limited by this accuracy.
- The Gibbs sampler is very computationally intensive: the most expensive part of the algorithm being the forward filtering / backward sampling joint draw for the symbol sequence. Each iteration is similar in complexity to the Viterbi algorithm, making a marked increase in the computational power required.
- There is no roll-off in performance at high SNR: in all simulations performed we have not seen any evidence of an irreducible or residual bit error rate.

6

Particle Filters

Up to now, the discussion on simulation-based Bayesian techniques has been limited to the processing of batches of data. Bayes' theorem itself provides the foundation for updating knowledge with the arrival of new data (section 4.1.2). In addition, sequential methods provide a very natural way of processing large amounts of data that arrive point by point rather than in a batch, typical applications being telecommunications systems, target tracking and the processing of medical data.

6.1 Particles and Updating Distributions

Particle filters work by representing the current probability distribution by a set of points or *particles* drawn from that distribution. To update the samples from this probability distribution with the arrival of new data, weights are assigned to the particles according to their *importance* to the new distribution. Clearly the probability distribution is only accurately described with a sufficiently high number of particles. If the variance of these weights becomes large, then only a small number of particles are contributing towards the representation of the probability distributions.

6.1.1 Filtering and Updating

There are two general applications for which we may use particle filters:

- **Filtering.** We are interested in making estimates of the current states based upon all the information available from past to present time.
- **Updating.** We are interested in a particular piece of information, but we do not have a particular time-scale for when this information is required. We use the particle filter to update our current beliefs about this particular piece of information with the arrival of new data.

The topic of filtering has attracted a great deal of interest over the years. The most significant breakthrough was the formulation of the Kalman filter [90], the optimal filter in the linear Gaussian case. Since then, focus has shifted to dealing with cases for which the Kalman filter does not apply. Proposed methods include the extended Kalman filter (EKF), reviewed in [2], and approximations based upon mixtures [184] or a grid in state space [93, 160]. All of these methods have problems — questions such as: when is the linearisation of the EKF valid? How should I choose my grid / mixture model centres? Particle filters attempt to circumvent these problems while adding one of their own: How many particles are needed to approximate the infinity required for the method to be guaranteed to work?

The process of filtering involves making new estimates within the particle filter structure and to give outputs for each new data point that arrives. This allows us to design a memory efficient implementation of the particle filter, where each particle need only represent the current state. All past information is summarised by these samples, and we will not need to output updated information on each past state. Fixed-lag methods are discussed in chapter 7.

The concept of updating is akin to that of learning: we update our knowledge as new data arrives, putting everything into perspective. In this case each particle will represent a *stream* or *trajectory* through state-space. This requires more storage, but allows us to give out our most up to date information when it is required.

6.1.2 Notation

The framework for particle filters is best understood if we consider a state-space model that evolves over time. The following notation is used:

- y represents the observations that are made.
- x represents the states. These are usually also the signal which we wish to recover.
- θ represents the parameters which may be fixed for all states or time-varying.
- *superscripts* as in $x^{(i)}$ denote particle index.

6.1.3 Importance Sampling

This section describes the concept of importance sampling. Importance sampling is used when we cannot simulate directly from the distribution of interest, and forms the basis for sequential updating used in particle filters.

Suppose we are interested in the expectation of some function, $E\{f(x)\}$, but we cannot generate random draws from $p(x)$ directly. Let us choose an alternative density, $\pi(x)$, close to $p(x)$ from which we can easily draw samples. We may write:

$$E\{f(x)\} = \int \frac{f(x)p(x)}{\pi(x)} \pi(x) dx \quad (6.1)$$

which may be estimated by drawing N samples, $x^{(i)}$, from $\pi(x)$ and evaluating the expression:

$$E\{f(x)\} \approx \sum_{i=1}^N f(x^{(i)}) \frac{w^{(i)}}{\sum_{j=1}^N w^{(j)}} \quad (6.2)$$

where:

$$w^{(i)} = \frac{p(x^{(i)})}{\pi(x^{(i)})} \quad (6.3)$$

The density $\pi(\cdot)$ is called the importance function and the $w^{(i)}$ are the importance weights. The choice of a valid and suitable importance function is discussed in [51].

6.1.4 Accuracy of Estimates

There are three sources of variance in the estimates made: those inherent from the variance of the posterior distribution itself, the effect of making inference using samples from this distribution rather than using the distribution itself and the additional losses resulting from having unequal weights as a result of importance sampling.

We can do nothing about the former in our analysis: it is inherent in the problem; if this is the largest source of variation and this variation is too large we need to tackle the problem from a different perspective. As engineers this means we must design better equipment (reduce inherent noise) or change the way current equipment is used. Examples of this include increasing transmitter power or antenna gain in a communications system, or simply make more observations e.g. in seismic or astronomical data.

The accuracy of estimates obtained from sampling directly from the posterior was discussed in section 4.2.1.1. If importance sampling is used, we may get additional variation introduced, depending on the importance function used. It is even possible to choose an importance function that is more efficient for estimation than drawing from the posterior distribution itself [51].

6.2 Particle Filtering Methods

There are two general approaches to particle filtering: those based upon sequential importance sampling and those based upon the Sampling — Importance Resampling algorithm. Both approaches will be described and it will be shown how they are inter-related. The differences arise essentially between when and how often (re)sampling takes place. In this section the methods described are those in which the state distribution does not depend upon parameters, θ . The extensions to include parameters will be discussed later in the chapter (section 6.4).

6.2.1 Sequential Importance Sampling

Sequential importance sampling (SIS) is the repeated application of importance sampling for the arrival of each data point. In essence it is a method of updating

a set of points drawn from one distribution to a range of others which are closely related so that importance sampling may be used.

Suppose we have N samples, $x_{0:t}^{(i)}$, where the superscript denotes the particle (or sample) index, from the probability distribution of interest at time t , $p(x_{0:t} | y_{0:t})$. With the arrival of a new data point y_{t+1} , we would like to update this distribution to $p(x_{0:t+1} | y_{0:t+1})$ without modifying the simulated past trajectories, $x_{0:t}^{(i)}$. We therefore need a way of both incorporating values for the new state, x_{t+1} and a way to perform the smoothing on the current states given the new data point, y_{t+1} .

We may update by using importance sampling sequentially, updating at each time step using:

$$p(x_{0:t+1} | y_{0:t+1}) = \frac{p(y_{t+1}, x_{t+1} | x_t)}{p(y_{t+1} | y_{0:t})} \times p(x_{0:t} | y_{0:t}) \quad (6.4)$$

which follows directly from Bayes' theorem and the Markovian assumptions of the model. Typically one cannot calculate the normalising term, $p(y_{t+1} | y_{0:t})$; however this is not required since it is independent of the state trajectory. Since we already have (particles representing) the distribution $p(x_{0:t} | y_{0:t})$, we use importance sampling on the distribution $p(y_{t+1}, x_{t+1} | x_t)$.

If an importance function of the form:

$$\pi(x_{0:t+1} | y_{0:t+1}) = \pi(x_{t+1} | x_{0:t}, y_{0:t+1}) \pi(x_{0:t} | y_{0:t}) \quad (6.5)$$

$$= \pi(x_0 | y_0) \prod_{k=1}^t \pi(x_{k+1} | x_{0:k}, y_{0:k+1}) \quad (6.6)$$

is chosen, the importance weights may be evaluated recursively [95, 34].

The unnormalised incremental importance weights may then be calculated by:

$$w_{t+1}^{(i)} = \frac{p(y_{t+1}, x_{t+1} | x_t^{(i)})}{\pi(x_{t+1} | x_{0:t}^{(i)}, y_{0:t+1})} \times w_t^{(i)} \quad (6.7)$$

where $w_t^{(i)}$ represents the weights at time t for the i th particle.

In choosing a suitable importance function, any factorisation of the joint density, $p(y_{t+1}, x_{t+1} | x_t)$, may be exploited so as to simplify this expression. The key to the particle filter performing well lies in the choice of a good importance function. This does not necessarily result from a factorisation.

After some time, this method may introduce importance weights that vary widely: they may be equalised by resampling (section 6.2.3) or rejection control (section 6.2.5).

6.2.2 Sampling – Importance Resampling (SIR) algorithm

The SIR algorithm is introduced in [151, 152] as a general non-iterative method of generating N samples from an arbitrary (posterior) distribution using importance sampling:

1. Choose a suitable importance function, $\pi(x)$ ¹.
2. **Sample** step. Obtain M samples (where $M \gg N$) from the distribution $\pi(x)$, denoted $\pi(x^{(i)})$.
3. **Importance** step. Assign respective weights by calculating importance ratios (as given by equation (6.3)).
4. **Resample** step. Resample with probability proportional to the weights, to choose N values of i so as to obtain N samples, effectively drawn from $p(x)$.

This can be applied to form a particle filter by repeating the following steps at each time [66]:

- Sample M particles from our prior sample of N equally weighted particles and propagate through the state equation (usually requiring further simulation) to generate prior samples, $x_{t+1}^{(i)}$.
- Calculate the importance weights in the usual manner.
- Resample N particles with probability proportional to the weights to generate (unweighted) samples from the posterior.

The main difference between SIR and SIS is that SIR generates unweighted samples from the distribution of interest by repeated use of (re-)sampling.

¹This need not have the recursive property in equation (6.6) since resampling always takes place.

6.2.3 Resampling — When?, How? and Why?

It is apparent that if we repeat the SIR algorithm many times, the resample and following sample step are next door to each other. It is also clear that repeated resampling will result in progressive degradation of the particles resulting in degeneracy. Indeed for repeated applications the initial sample step does not gain anything (since the initial set of particles are equally weighted) except the increase in the number of particles (if $M > N$). It is preferable instead to simply replicate the initial set $\frac{M}{N}$ times, choosing M to be a multiple of N . This will reduce the variance of inference made since we have used stratified sampling (section 6.2.3.2) at the sample step. The resample step will also introduce additional variation [150], especially when the weights are equal.

On the other hand, by simply considering the sequential importance sampling algorithm outlined in section 6.2.1 the results are still valid even if no resampling is done. However, without any resampling, the importance weights will become steadily more diverse and in the worse case, any inference made will use only the particle with the highest weight (as this will be much larger than any of the others). This can occur alarmingly easily in practice, especially if there is only a small number of particles representing the distribution.

Clearly a balance needs to be struck between excessive resampling and no resampling at all. This places the ideal particle filter somewhere in between the extremes SIR and SIS. There is some discussion of the gains and losses of resampling in [108], although there are no robust rules for the general case.

The easiest heuristic approach is to resample after a set number of steps. This resample rate could vary between 1 and 100, depending on the application. The drawback is that outliers can often cause a degradation, and it may be some time before a resampling step takes place. Ideally we would like to resample at the earliest point where resampling would result in a gain.

One method of assessing when we have reached this point is an estimate of the effective sample size (section 6.2.4). One method therefore is to resample whenever the effective sample size is less than fixed value.

- Choose a lower limit for the effective sample size N_{lim} , typically taken somewhere between 10–90% of the total number of particles.
- Resample if $ESS < N_{lim}$, otherwise no resampling takes place.

6.2.3.1 Resampling Algorithms

The direct approach to implementing the resampling step is as follows: First calculate the boundaries (thresholds) by doing a cumulative sum of the normalised weights, in any order. Then for each index, i

1. Draw a uniform random number, u_i , between 0 and 1.
2. Use a binary (or other) search algorithm to locate the position of u_i within the thresholds.
3. Set the resampled index according to the index of the location of u_i .

This method is $O(N \log(N))$, where N is the number of particles. The complexity may be reduced to $O(N)$ if a set of N ordered uniform variates are obtained. These may be obtained by any of the methods outlined in [50]. A single loop can then be used to obtain the indices [131, 19] in a straightforward manner.

6.2.3.2 Stratified Sampling

The aim of stratified sampling, in the context of particle filters, is to resample from the weights, such that the number of particles selected is *equal* to their weights. This will reduce the variance introduced by resampling [135, 149]. In practice the weights will not be multiples of $\frac{1}{N}$. There are three methods in the literature for achieving stratification.

Residual Resampling. This is a method described in [108] which only achieves partial stratification. For each normalised weight, $w_*^{(i)}$, the particle is resampled $n^{(i)}$ times where $\frac{n^{(i)}}{N} \leq w_*^{(i)} < \frac{n^{(i)}+1}{N}$. The remaining particles are randomly selected as before from the residuals, $rw^{(i)}$, where:

$$rw^{(i)} = w_*^{(i)} - \frac{n^{(i)}}{N} \quad (6.8)$$

Branching Corrections. There exist a variety of minimum variance branching corrections, that is methods where the resampling scheme leads to a minimum variance estimator. These methods satisfy two criteria: the $n^{(i)}$ are chosen in a way such that they take the value of the two closest integers either side $w_*^{(i)}$ and

that $E\{n^{(i)}\} = w_*^{(i)}$. One such example, described in [29], begins as in the residual resampling method. However the remaining particles are chosen as follows: For each index i , choose an additional particle with this index with probability equal to $r w^{(i)} \times N$. This method produces a fully stratified sample but has the disadvantage of a fluctuating number of particles. A comparison between branching corrections and resampling is given in [30].

Systematic Sampling. Another example of a minimum variance method is that described in [19, 18, 38]. This proceeds in the same manner as the random resampling algorithm, except using:

$$\begin{aligned} u_1 &\sim U\left(0, \frac{1}{N}\right) \\ u_i &= u_1 + \frac{i}{N} \end{aligned} \tag{6.9}$$

instead of each u_i drawn independently from $U(0, 1)$. This has the advantage that the u_i are automatically ordered and only one random number needs to be generated. A clear explanation of how this achieves our goal may be found in [38].

6.2.4 Effective Sample Size

The *effective sample size* may be defined as the number of particles representing the distribution if the associated weights are equal. This provides an effective indicator to the efficiency of the particle filter when compared to the number of particles used.

There are two estimators in the literature for this [103, 19]: Liu and Chen [103] describe a “rule of thumb” based upon the coefficient of variation, $C(w)$, which is defined as

$$C^2(w) = \frac{1}{N} \sum_{j=1}^N (N w_*^{(j)} - 1)^2 \tag{6.10}$$

where $w_*^{(j)}$ are normalised weights (i.e. $w_*^{(j)} = \frac{w^{(j)}}{\sum_j w^{(j)}}$). The effective sample size

(ESS) is then given by:

$$\text{ESS} = \frac{N}{1 + C^2(w)} \quad (6.11)$$

$$= \frac{1}{\sum_{j=1}^N (w_*^{(j)})^2} \quad (6.12)$$

This gives some intuitive behaviour: If we have equal weights, the effective sample size is N . If M weights are equal and non-zero with $N - M$ weights near zero, the effective sample size is M . This estimator is in fact more than a simple heuristic measure: it may be derived by comparing the variance in the estimate of some measure with the variance obtained purely by random sampling [121]. In addition it has other useful properties:

- It is independent of the measure or expectation that we wish to obtain from the particle filter.
- It is relatively easy to calculate online to serve as a monitor for the effectiveness of the sampler (section 6.2.3).

However the error in the value for this effective sample size, may sometimes be quite large, and in the worse case it is infinitely wrong [38]. The analysis does not apply to particle filters where resampling has taken place.

Carpenter *et al.* [19] propose a Monte-Carlo method for the estimation of ESS. This is obtained by using a particle filter to analyse the same data L times (with different random seeds) and is dependent upon the function of interest, $f(x)$. For each run, an estimate for this function:

$$\hat{\mu}_i = E\{f(x)\} \quad (6.13)$$

and an estimate of the variance of this estimate:

$$\hat{\sigma}_i^2 = E\{f(x)^2\} - \hat{\mu}_i^2 \quad (6.14)$$

are calculated using equation (6.2). We expect the variance of the estimates to be equal to $\frac{\sigma^2}{N^*}$ where σ^2 is the actual variance of x from sampling theory (section 4.2.1.1). If we assume that the averages for the value and variance of the

estimates over the separate runs are equal to the actual values, the ESS may be calculated as:

$$\text{ESS}_2 = \frac{L}{\sum_{i=1}^L \left(\hat{\mu}_i - \frac{1}{L} \sum_{j=1}^L \hat{\mu}_j \right)^2} \times \frac{1}{L} \sum_{j=1}^L \hat{\sigma}_i^2 \quad (6.15)$$

This method of estimating ESS is applicable to any particle filter and is therefore suitable for analysing performance but it is not suitable for online assessment of efficiency.

6.2.5 Rejection Control

Liu *et al.* [109] describe an alternative method for resampling based upon the ideas of sequential importance sampling. Particles with a weight below a certain threshold are randomly rejected at time t according to:

$$\text{Pr}(i\text{th Particle accepted}), r_t^{(i)} = \begin{cases} 1 & \text{if } w_t^{(i)} > c, \\ \frac{w_t^{(i)}}{c} & \text{if } w_t^{(i)} < c. \end{cases} \quad (6.16)$$

To correct for this random rejection step, accepted streams are re-weighted:

$$w_t^{(i)} = \frac{w_t^{(i)}}{r_t^{(i)}} \quad (6.17)$$

Methods for deriving reasonable values for the threshold, c , are discussed in the paper.

This method will decrease the number of particles. It is suggested in the same paper [109] that an additional stream is then generated from time 0, under the same scheme. This is highly inefficient and often not practical if all the intermediate data is not stored, as is common if just filtering is required. It is not clear if this method performs any better than simply resampling; however it does reduce computation if no resampling takes place, since particles that are unlikely to influence the final posterior significantly are rejected at an early stage.

6.3 Choice of Importance Functions

6.3.1 Prior Importance Function

The most intuitive choice for an importance function in Bayesian context is the prior distribution, $p(x_{t+1} | x_t)$. It is also the most intuitive choice for anyone familiar with filtering as it is the one-step ahead prediction density, before the observation is made. Indeed, particle filters based upon this choice of importance function are described in [76, 75, 66, 65, 94] and are often the easiest to implement. By substituting $\pi(x_{t+1} | x_{0:t}, y_{0:t+1}) = p(x_{t+1} | x_t)$, into equation (6.7) we obtain:

$$w_{t+1}^{(i)} = p(y_{t+1} | x_{t+1}^{(i)}) w_t^{(i)} \quad (6.18)$$

It therefore turns out that the incremental weight is the likelihood of obtaining the observation given the newly simulated state. Hence the requirements are that the likelihood can be evaluated and samples can be drawn from the prior distribution. If we have a state-space model in the form of equation (2.7), this involves generating the state innovations and calculating the next state from the state propagation equation.

Example: Consider the example of a one-dimensional Gaussian random walk performed by a person where x_t represents the current position (the next position will have mean x_t and variance σ^2), y_t represents the current observation with additive noise independently and identically distributed over time as $L(y_t | x_t)$. Let us also suppose that we have particles, $x_0^{(i)}$, where $i = 1 \dots N$ which are drawn from the distribution representing where we think the person is initially. The algorithm proceeds iteratively as follows:

1. For each i from $1 \dots N$, sample particles $x_{t+1}^{(i)}$ from $\mathcal{N}(x_t^{(i)}, \sigma^2)$.
2. For each i , calculate associated importance weights given by the likelihood: $w_{t+1}^{(i)} = L(y_{t+1} | x_{t+1}^{(i)}) w_t^{(i)}$.
3. If required, resample particles with the probability of selecting the i th particle proportional to $w_{t+1}^{(i)}$ and reset the weights to equal values.

6.3.2 Posterior Importance Function

The biggest drawback with simulating from the prior is when we get poor overlap between the prior and posterior. This results in a high variation in the weights, and as a consequence a low effective sample size, and quite possibly poor estimates. To overcome this we would like to use some information from our latest observation.

The choice of:

$$\pi(x_{t+1} | x_{0:t}, y_{0:t+1}) = p(x_{t+1} | x_t, y_{t+1}) \quad (6.19)$$

as used in [190, 95, 103] does exactly this. The importance weights obtained from equation (6.7) are then given by:

$$\begin{aligned} w^{(i)} &= p(y_{t+1} | x_t) \\ &= \int_{x_{t+1}} p(y_{t+1} | x_{t+1}) p(x_{t+1} | x_t) dx_{t+1} \end{aligned} \quad (6.20)$$

This importance function is optimal in the sense of minimising the importance weights [34]. The only problem is evident in equation (6.20) above: we need to evaluate the integral analytically. The evaluation of the same integral is required when sampling from the importance function given in equation (6.19). In the case of discrete states, this integral is a sum which is always possible to evaluate, although it may be computationally expensive if there are many states.

6.3.3 Auxiliary Particle Filter

The auxiliary particle filter or auxiliary sample importance resampling algorithm (ASIR) was introduced in [131, 133]. It was introduced in an attempt to get round the degeneracy problems encountered when the observation is informative (low variance) or in the tails of the prior. The idea is based upon the SIR method and proceeds as follows:

- Let us introduce a auxiliary variable, k , which represents the index of the set of particles from the last iteration from which the current one is derived, i.e. $k^{(i)} = j \Rightarrow$ at the current iteration we set $x_{0:t}^{(i)} = x_{0:t}^{(j)}$.

- Rather than sampling the indices of the prior sample randomly, choose the index from an importance function which takes the observation into account.

$$\pi(k | x_{0:t+1}, y_{0:t+1}) \propto p(y_{t+1} | \mu_{t+1}^{(k^{(i)})}) \quad (6.21)$$

The idea is to pick “good” particles — those which are in regions of high posterior probability — by our choice of $\mu^{k^{(i)}}$.

- This biased sampling introduces a first-stage weight:

$$v^{(i)} = \frac{1}{p(y_{t+1} | \mu_{t+1}^{(k^{(i)})})} \quad (6.22)$$

- The new states are also chosen using an importance function in the standard manner, with associated weights $w_{\ddagger}^{(i)}$.
- The final importance weights are then given as the product of the two weights:

$$w^{(i)} = v^{(i)} w_{\ddagger}^{(i)} = \frac{w_{\ddagger}^{(i)}}{p(y_{t+1} | \mu_{t+1}^{(k^{(i)})})} \quad (6.23)$$

As a result, it is hoped that the final importance weights are less variant than if the ASIR method had not been used.

Note that the original form of the algorithm [131, 133] imposed two unnecessary limitations:

- The second-stage importance function was limited to the prior importance function: in fact any importance function can be used.
- A resampling step took place using the second-stage weights: this is not required as the weights may be carried forward to the next “sample” step.

Despite these generalisations, there remains the limitation that resampling takes place at every step of the iteration which can and often will lose information if there is a good overlap between prior and posterior. In these cases it would be better not to use the auxiliary particle filter.

6.4 Incorporating Parameters

The framework for implementing particle filters described above does not incorporate parameters directly. Methods for incorporating parameters will be described in this section. For time-varying parameters this is fairly straightforward as they may be incorporated into the state (see next section). Incorporating fixed parameters is a more complicated issue, which deserves further attention.

By extension of equation (6.4), we may write:

$$p(x_{0:t+1}, \theta_{t+1} | y_{0:t+1}) = \frac{p(y_{t+1}, x_{t+1}, \theta_{t+1} | x_t, \theta_t)}{p(y_{t+1} | y_{0:t})} \times p(x_{0:t}, \theta_t | y_{0:t}) \quad (6.24)$$

where the subscript indexing θ may index time, in the time-varying case, or the samples of θ after time t , for fixed parameters.

6.4.1 Time-Varying Parameters

Time-varying parameters are relatively easy to implement in the particle filter framework. We may increase the state vector to include parameters as well as “states”. A good example of this is a time-varying communications channel. By choosing the state to carry exactly the same information as used in previous chapters, the state represents the last L symbols, where L is the length of the channel. The channel may be added to the state vector, completely independently of the symbols. The particle filter can then be implemented in the usual manner since usually both state evolution equations are independent of each other i.e.

$$p(x_{t+1} | x_t, \theta_t) = p(x_{t+1} | x_t) \quad (6.25)$$

$$p(\theta_{t+1} | \theta_t, x_t) = p(\theta_{t+1} | \theta_t) \quad (6.26)$$

In fact as long as either one of these conditions apply, implementation is straightforward, since the component for which the marginal density is available may be drawn first, as in the data augmentation algorithm [165].

The process of adding the channel to the state vector will increase the dimension of the state. As a consequence, more particles may be required to get a good

estimate of the posterior distribution. This is not really an issue with the mode of implementation, more a reflection that the problem is harder to solve.

We may also assume that fixed parameters are in fact time-varying ones which do not move very much e.g. a random walk with a small variance. This approach is suggested in [66]. This may be implemented in the same manner as has already been discussed, but the use of an incorrect model loses the prior information that the parameters are, in fact, fixed.

6.4.2 Kernel Density Estimates

A more recent approach is to use kernel density estimation [158] to estimate the distribution of the parameters, and use this to simulate updated values for these parameters at each time step [104, 105]. The distribution $p(\theta | y_{0:t})$ is given by a kernel density estimate which takes the form of a mixture density with N mixtures, each component centred on the particles, $\theta_t^{(i)}$. On the face of it, this does not differ from assuming the parameters are time-varying, however the use of shrinkage [185, 186] ensures that an over-dispersed approximation does not occur. The extra variation, caused by adding “noise” (in the form of a kernel density), is contained by moving the centre point of each component of the mixture distribution closer to the mean of the particles, $\theta_t^{(i)}$. In this way, on selection of the particle index, i , either deterministically, randomly or by use of the auxiliary particle framework, the previous state sequence $x_{0:t}^{(i)}$ is chosen along with a value for θ_{t+1} , drawn from the i th component of the mixture distribution describing $p(\theta | y_{0:t})$ after shrinkage.

6.4.3 Marginalisation

If we are not actually interested in the parameters themselves, just the states, then we should marginalise them analytically if possible: for example a Gaussian channel in a communications system [108]. This is an example of Rao-Blackwellisation and it can be shown [34] that the variance of estimates is reduced. In this case, particles do not hold any information about the parameters and we may use the update equation:

$$p(x_{0:t+1} | y_{0:t+1}) = \frac{\int_{\theta} p(y_{t+1}, x_{t+1}, \theta | x_t) d\theta}{p(y_{t+1} | y_{0:t})} \times p(x_{0:t} | y_{0:t}) \quad (6.27)$$

This may be a good approach, even if we are interested in the value of θ . A Rao-Blackwellised estimate of the density [107] may be obtained subsequently:

$$p(\theta | x_{0:t+1}, y_{0:t+1}) = \sum_{i=1}^N p(\theta_{t+1} | x_{0:t+1}^i, y_{0:t+1}) \quad (6.28)$$

6.4.4 Simulation

It is possible to directly simulate updated values of the parameters at each time step by using the update equation:

$$p(x_{0:t+1}, \theta_{t+1} | y_{0:t+1}) = \frac{p(y_{t+1}, x_{t+1} | x_t, \theta_{t+1}) p(\theta_{t+1} | x_{0:t}, y_{0:t})}{p(y_{t+1} | y_{0:t})} \times p(x_{0:t} | y_{0:t}) \quad (6.29)$$

and obtaining draws of the parameters from:

$$\theta \sim p(\theta_{t+1} | x_{0:t}, y_{0:t}) \quad (6.30)$$

The states may be generated using any importance function, as previously described, which may be conditional on the newly drawn parameters. The importance weights are unchanged by this procedure.

Note that we can obtain estimates of any function involving either states or parameters at the end each time step. The values for the parameters are discarded at the start of the next step to implicitly obtain the marginal distribution $p(x_{0:t} | y_{0:t})$. This method is very flexible and seems to have been overlooked by many researchers. However there is one drawback — we must do one of the following:

- Update sufficient statistics in order to sample from the full conditional distribution $p(\theta_{t+1} | x_{0:t}, y_{0:t})$, either directly or using importance sampling. Direct sampling is only possible for a small class of distributions. For example this would be possible if this distribution was Gaussian. In this case, it is usually preferable to marginalise this parameter, however one might use this method if they were interested in an estimate of some function of the parameters.
- Store the whole observation history and the values of the states from time 0 to the current time. This is very expensive in memory requirements and is

likely to make the draws of the parameters more computationally expensive every time step.

- Use a “sliding window” over which to estimate parameters (this will be discussed in detail in section 7.4). This conveniently gives a fixed memory and computational requirement, and the ability to trade these off with the accuracy required.

This formulation of drawing the parameters from the p.d.f. conditional on the states and observations at the end of the last step means that θ is drawn from the same distribution as it would be in an iteration of the Gibbs sampler. In fact any MCMC step to obtain new values of the parameters could be used here instead. The use of MCMC moves within the particle filter will be discussed in detail in section 8.1.1.

6.5 Summary

The use of *particle filters* is a rapidly growing research area in many branches of science, statistics and engineering, since a sequential approach often provides a very natural way of processing large amounts of data that arrive point by point rather than in a batch. Monte Carlo methods have been shown to give dramatic improvements in performance for non-linear and non-Gaussian state-space models when compared with classical sequential techniques such as the extended Kalman filter. They allow sequential updating in the Bayesian framework using a parallel implementation, thereby making them ideal for implementation in dedicated hardware.

In this chapter an overview of the theory behind the use of these methods has been presented. They are based upon the repeated application of importance sampling so we may update a set of particles drawn from one distribution to another closely related one. Estimates of the *Effective Sample Size* may be used to monitor performance online or to study variation in behaviour of different particle filter implementations.

The choice of importance function was also discussed, as this plays a vital role in the efficiency of the algorithm. Great efficiency savings may be made by

choosing an importance function which uses the new observation as well as the current state.

Finally various methods for incorporating parameters were considered. The use of marginalisation has many theoretical advantages, although marginalisation analytically is not always possible. In this case parameters must be represented in the state, either by simulation or using kernel density estimates.

Particle Filters for Blind Equalisation **7**

In this chapter the results obtained using the particle filter for equalisation will be presented and two modifications to the particle filter will be introduced: namely methods for fixed-lag simulation and data windowing. These methods are important for use in practical applications and in particular communications systems.

Most communications systems involve a continuous stream of data which, in its purest form, is not naturally divided into blocks or frames. This makes them an ideal application for a sequential processing method such as the particle filter.

Markovian systems have an implicit dependency between adjacent states. To get good results we need to make estimates by using information both prior to and following the state which we wish to infer. The prior information is neatly encoded in the particle cloud itself. Future data is not taken into account until updates have taken place. It therefore follows that by allowing a small lag to incorporate information from later observations, our estimates will be improved. Some of the results on the use of fixed-lag particle filters for digital communications systems have been previously published in [26, 24].

The use of data windowing was introduced briefly in the section on incorporating parameters (section 6.4). The idea is to eliminate perpetually increasing storage requirements for simulating fixed parameters. This is clearly required when

processing the very large data sets common in engineering applications when sufficient (summary) statistics cannot be used.

7.1 Particle Filters for Equalisation

The use of particle filters for equalisation has a number of benefits over both conventional methods and the MCMC methods already discussed. The Bayesian framework is retained; we may still use the desirable properties of *marginal* MAP estimates of the sequence, used to great effect as described in chapter 5, however the algorithm may now be run sequentially. The basic framework has many similarities to more conventional techniques, such as per-survivor processing, the list Viterbi algorithm and genetic algorithms; the particle filter may be seen as a generalisation of these techniques.

Per-Survivor Processing. The per-survivor processing algorithm [140] proceeds in a trellis structure as for the Viterbi algorithm. The difference is that the channel estimate is not assumed correct: for each new state an adaptation step (usually along the lines of a gradient descent step converging to the MAP estimate of the channel given the sequence) is made. In the particle filter the channel adaptation is done by simulation rather than in a deterministic manner, conditional on the previous state of the channel and/or new state of the symbol sequence.

List Viterbi algorithm. There are a variety of methods including the list Viterbi algorithm in which the best N state sequences are stored [122]. These are chosen deterministically based upon the metric. Similarly in the particle filter N sequences are stored (although typically N will be much larger for the particle filter) but the selection process is based upon some form of random resampling (which may contain deterministic elements to reduce the variance of resulting estimates).

Genetic Algorithms Genetic algorithms allow a model to evolve based upon a model of genetic mutation. The structure is very similar to that of the particle filter and analogues may be found for system noise, observations, weights, etc.

The inquiring reader is referred to [82, 83] for a description of the relationship between the two.

7.1.1 Equalisation of a Time-Invariant Channel

The optimal method for filtering the data without lookahead is the method of *sequential imputations* as described by Liu and Chen [103]. This direct approach requires analytic marginalisation of the parameters \mathbf{h} , which is in fact possible for the communications model we shall consider, because the prior distribution for the channel is Gaussian, but will not be possible for many other models of practical interest.

At each step only a draw of the state from $p(\mathbf{x}_{t+1} | \mathbf{x}_t, y_{t+1})$ is required. The incremental importance weight is given by $p(y_{t+1} | \mathbf{x}_t)$, and updates of the mean and variance of the marginal distribution of the parameters, θ , are obtained by recursive relationships derived from the matrix inversion lemma in the conditionally Gaussian parameter case. The noise variance is assumed known.

If we wish to simulate the channel (if marginalisation is not possible, or not desirable if we are interested in the channel parameters) then we may proceed as described in section 6.4.4. Simulation of the noise variance is achieved from the full conditional i.e. a Gibbs Sampler move.

The new importance function is given as:

$$\pi(\mathbf{x}_{t+1}, \mathbf{h}_{t+1}, \sigma_v^2 | \mathbf{x}_{0:t}, y_{0:t}, \mathbf{h}_t) = p(\mathbf{x}_{t+1} | \mathbf{x}_t) p(\mathbf{h}_{t+1} | \mathbf{x}_{0:t}, \sigma_v^2, y_{0:t}) p(\sigma_v^2 | \mathbf{x}_{0:t}, \mathbf{h}_t, y_{0:t}) \quad (7.1)$$

Therefore at each time step we need to do the following:

1. For each particle, indexed j :
 - (a) Draw $[\sigma_v^2]_{t+1}^{(j)}$ from $p(\sigma_v^2 | \mathbf{x}_{0:t}^{(j)}, \mathbf{h}_t^{(j)}, y_{0:t})$.
 - (b) Draw $\mathbf{h}_{t+1}^{(j)}$ from $p(\mathbf{h} | \mathbf{x}_{0:t}^{(j)}, [\sigma_v^2]_{t+1}^{(j)}, y_{0:t})$.
 - (c) Draw $\mathbf{x}_{t+1}^{(j)}$ from $p(\mathbf{x}_{t+1} | \mathbf{x}_t^{(j)})$.
2. Calculate the weights which are given by the likelihood:

$$w_{t+1}^{(j)} = p(y_{t+1} | \mathbf{x}_{t+1}^{(j)}, \mathbf{h}_{t+1}^{(j)}, [\sigma_v^2]_{t+1}^{(j)}) w_t^{(j)}$$

3. If the weights have a high variance (or equivalently, the effective sample size is too low), rejuvenate by resampling and reset the weights to equal values.¹

Note that the subscript t refers to a time index: as before in the case of h and σ_v^2 , which are time-invariant parameters, it simply refers to the draw at time t , not that the parameters are time-varying, or we assume them to be.

An alternative importance function resulting in lower variance of the weights, incorporating the information from the next observation in the draw for the current state would be to replace step 1(c) with:

- Draw $x_{t+1}^{(j)}$ from $p(x_{t+1} | x_t^{(j)}, y_{t+1}, h_{t+1}^{(j)}, [\sigma_v^2]_{t+1}^{(j)})$.

This results in the weights being given by:

$$w_{t+1}^{(j)} = p(y_{t+1} | h_{t+1}^{(j)}, x_t^{(j)}, [\sigma_v^2]_{t+1}^{(j)}) w_t^{(j)} \quad (7.2)$$

which may be calculated by summing over all allowed states, given the previous state (since all non-zero transition probabilities are equal).

7.1.2 Equalisation of a Time-Varying Channel

The extension to a time-varying channel may be regarded as a simplification. Step 1(b) above is replaced by:

- Draw $h_{t+1}^{(j)}$ from $p(h_{t+1} | h_t)$

Indeed the draw for the channel and symbol sequence, 1(b) and 1(c) respectively, may now be done in either order. The remainder of the algorithm proceeds as before. In effect the state has been augmented with the channel h_t , and the particles will track the evolution of h_t over time. We would expect a larger number of particles to be required for adequate tracking of the distribution because of this increase in dimensionality of the state vector.

An example of a time-varying channel would be to allow the channel coefficients to evolve according to a Gaussian random walk. This is described in section 8.4.2.

¹The term resampling is used to refer to any of the methods described in section 6.2.3, not just the simplest version.

The use of the filtering density to estimate the transmitted symbols will not work effectively: the pulse is spread out over several observations, and the filtering density will only take the first of these into account. Our decisions must be delayed to make best use of the available information: for this we need a framework for fixed-lag estimation within the particle filter.

7.2 The Fixed-Lag Approach

The actual requirements of a system often lie somewhere in between the two extremes of updating and filtering. In many applications a small amount of delay, or latency, is allowable in the estimation of certain quantities. It will then be profitable to perform fixed-lag smoothing rather than filtering. This is particularly true if we are trying to obtain a data sequence, such as in telecommunications, as opposed to inference on underlying structures that might take place on financial time-series data. Indeed significant gains in performance have been reported in the communications literature, e.g. [1, 81, 97].

7.2.1 One-Dimensional Non-Linear Example

Let us consider a simple example. It is a univariate non-stationary growth model used in references [93] and [66].

The model is given by

$$x_t = 0.5x_{t-1} + 25 \frac{x_{t-1}}{1 + x_{t-1}^2} + 8 \cos(1.2(t-1)) + u_t \quad (7.3)$$

$$y_t = x_t^2 + v_t \quad (7.4)$$

where u_t and v_t are zero-mean Gaussian white noise with variances 10 and 1 respectively. Clearly when the value of y_t is close to zero, with a moderate amount of noise, it is ambiguous as to whether the state x_t is positive or negative by looking at the observation alone. However if our prior samples show x_{t-1} all negative, then it is clear that the true value is probably on the negative side. As x moves away from zero, the distribution is again bi-modal, however in this case prior information gives no further information as to which mode is correct. The state

evolves in such a way as to resolve this ambiguity at some later time. When this happens, perhaps half of the particle streams will be rejected: if all state trajectory is stored the ambiguity will not just be resolved for the current time, but for earlier times as well.

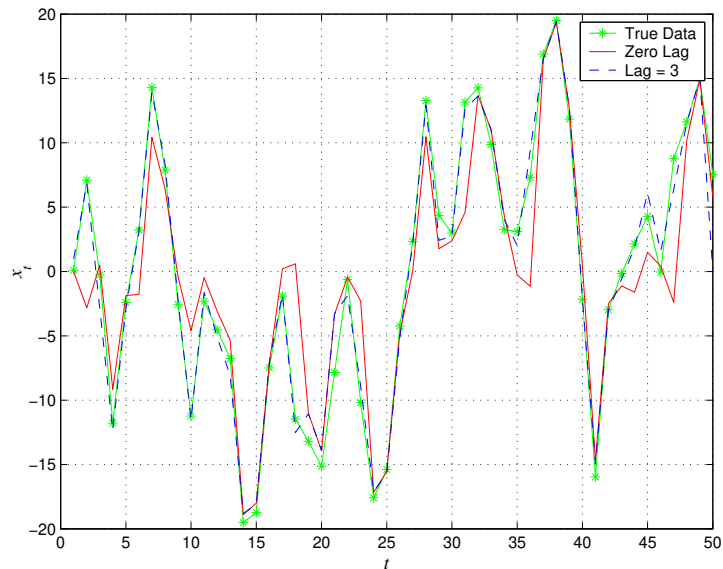


Figure 7.1. Particle filter estimate of Posterior Mean using the filtering density and a fixed-lag of 3 employing the method described in section 7.3.1.

To illustrate this figure 7.1 shows the plots of the posterior means using the filtering density and smoothed density with a lag of 3 i.e. $p(x_{t-3}|y_{0:t})$, using the method described below in section 7.3.1. The lagged estimate is far superior, particularly when the state moves away from a value close to zero, for example at $t = 17$ for the reasons described above. The modification to the algorithm is only minor with a small amount of extra storage required.

Due to the bi-modality of the likelihood and hence the posterior, the posterior mean is not a good estimator of the true state in this particular problem: a MAP estimate made from some kernel density estimate would be far superior, as evident in figure 4 of [66].

7.3 Fixed-lag Methods using Particle Filters

The need and advantages of fixed-lag methods are apparent from the example in the previous section. However there is a number of different ways that the most recent observations may be used. The following sections describe the methods for a general problem with observations, y_t , states, x_t , and time-invariant parameters, θ .

7.3.1 Using the Filtering Density: The Decision Step

The simplest method for fixed-lag simulation is to update the filtering density as each data point arrives as in standard SIS, i.e. track the evolution of $p(x_{0:t} | y_{0:t})$, but also storing the state history of each particle back to a lag of p states. Given the stored histories of each particle back to time $t - p$ it is then straightforward, using the importance weights from time t , to make an estimate of the smoothing density, from which estimates such as the MAP estimate may be extracted. Obtaining the output in this way will be termed a *decision step*.

The particle filter updates are performed in the usual manner: there is no change to the actual implementation. The true density is still being tracked and the decision step does not affect the operation of the particle filter itself. In effect each particle still represents the entire history through state-space from the starting point to the current time. Resampling will cause a gradual degeneration in the number of distinct points in the particulate mixture for any point in the distant past. Eventually this will reduce to a single particle. Clearly this is, in general, a bad idea since the posterior distribution is not properly represented. At this point it is instructive to compare this method to trace-back in the Viterbi equaliser.

7.3.1.1 Comparisons with the Viterbi Equaliser

The Viterbi algorithm is designed to output the maximum likelihood sequence through discrete state-space, given the correct channel. In order to achieve this all possible state sequences are considered, although its recursive structure means that this is done in a computationally efficient manner. At each time step, only

the best paths are stored which reach each state. The hope is that at some point in the future the paths will converge and the best sequence through state-space can be found. This degeneracy to one state sequence is essential to finding the correct (ML) sequence. In practice, only a finite delay is permitted before a decision is made in the trace-back stage.

The particle filter is more complex since it aims to represent the posterior probability distribution, rather than solving for the single maximum likelihood sequence. With the discrete state-space system inherent in digital communications, the typical behaviour is that most of the time, the vast majority of particles represent one state only. When there is more ambiguity, effectively the state traces diverge giving a correct sequence and the plausible alternatives. It is choosing the wrong sequence in one of these cases that causes a burst error (as in the case of the Viterbi algorithm [39]). With additional data we may update the relative weights of these sequences. After a moderate amount of time, the various state sequences are now weighted according to their true posterior probabilities, since further observations will not significantly influence these states.

Herein lies the problem: since the particle filter may only be implemented with a finite number of particles, the resampling and reweighting that takes place only really reflects the current state and those states in the recent past, not those in the distant past. As a consequence, the estimates of those states in the distant past *can only get worse*. In the worst case, after considerable resampling, only one value for the state will be represented, even if it is defined on a continuous state-space. This is a limitation in the practical implementation of particle filters.

The use of a decision step at a reasonable lag should overcome this problem, since we will obtain our estimates at the one time when we expect the density estimates to be most accurate. Clearly, the size of this lag should be chosen with care, however as long as there is a fairly large number of particles, the degeneration will not happen too fast.

7.3.1.2 Method

Let us assume we have available at time t , N samples $x_t^{(j)}$ from the distribution $p(x_t | y_{0:t})^2$, possibly with associated weights $w_t^{(j)}$.

²Initial samples could be created by any batch-based MCMC method, for example.

We may update to the distribution $p(x_{t+1} | y_{0:t+1})$ as follows:

1. For each particle, indexed j :
 - (a) Draw $\theta^{(j)}$ from $p(\theta | x_{0:t}^{(j)}, y_{0:t})$.
 - (b) Draw $x_{t+1}^{(j)}$ from $p(x_{t+1} | \theta^{(j)}, x_t^{(j)}, y_{t+1})$.
 - (c) Calculate importance weights given by $w_{t+1}^{(j)} = p(y_{t+1} | \theta^{(j)}, x_t^{(j)}) w_t$.
2. Output or store the estimate $\hat{f}(x_{t-p+1}) = \frac{\sum_{i=1}^N f(x_{t-p+1}^{(i)}) w_{t+1}^{(i)}}{\sum_{j=1}^N w_{t+1}^{(j)}}$.
3. If Effective sample size is too small, resample and reset importance weights to equal values.

Note that the required estimation is made before resampling since the resampling will lose information [150]. However the losses are not that great and if resampling takes place at every iteration it may be preferable to make estimates after it has taken place since it is computationally much easier because the weights are equal.

7.3.2 Using the Filtering Density with Look-Ahead

Consider tracking the filtering density, but at p steps back in time from the latest observation. The extra observations available, $y_{t+1:t+p}$, may be used in the importance function when drawing for the state. The hope is that the states drawn at this stage will be kept at later iterations, and there will be less variation in the weights.

To simplify notation, we will shift our time-base forward by p samples. Hence at time t we will draw fixed-lag samples of x_t using information from $y_{0:t+p}$. This method may also be referred to as using the lagged filtering density.

The joint prediction density may be factorised as follows:

$$p(y_{t+1}, x_{t+1}, \theta | x_{0:t}, y_{0:t}) = p(x_{t+1} | x_t, \theta, y_{t+1}) p(y_{t+1} | x_t, \theta) p(\theta | x_{0:t}, y_{0:t}) \quad (7.5)$$

It is possible to sample directly from the first term; however we have chosen instead to use the importance function:

$$\pi(x_{t+1} | x_t, \theta, y_{t+1:t+p+1}) = p(x_{t+1} | x_t, \theta, y_{t+1:t+p+1}) \quad (7.6)$$

with the hope of using the additional data to bias the draw towards states that will be useful at later iterations. This may be accomplished by using forward filtering backward sampling techniques described in section 5.2.1, drawing from $p(\mathbf{x}_{t+1:t+p+1} | \mathbf{x}_t, \boldsymbol{\theta}, \mathbf{y}_{t+1:t+p+1})$ and discarding the unwanted imputed values, $\mathbf{x}_{t+2:t+p+1}$. This introduces the importance weight:

$$w_{t+1} = \frac{p(\mathbf{x}_{t+1} | \mathbf{x}_t, \boldsymbol{\theta}, \mathbf{y}_{t+1})}{\pi(\mathbf{x}_{t+1} | \mathbf{x}_t, \boldsymbol{\theta}, \mathbf{y}_{t+1:t+p+1})} w_t \quad (7.7)$$

$$\begin{aligned} &= \frac{\frac{p(\mathbf{y}_{t+1} | \mathbf{x}_{t+1}, \boldsymbol{\theta}) p(\mathbf{x}_{t+1} | \mathbf{x}_t)}{p(\mathbf{y}_{t+1} | \mathbf{x}_t, \boldsymbol{\theta})}}{\frac{p(\mathbf{y}_{t+1:t+p+1} | \mathbf{x}_{t+1}, \boldsymbol{\theta}) p(\mathbf{x}_{t+1} | \mathbf{x}_t)}{p(\mathbf{y}_{t+1:t+p+1} | \mathbf{x}_t, \boldsymbol{\theta})}} w_t \\ &= \frac{p(\mathbf{y}_{t+1} | \mathbf{x}_{t+1}, \boldsymbol{\theta}) p(\mathbf{y}_{t+1:t+p+1} | \mathbf{x}_t, \boldsymbol{\theta})}{p(\mathbf{y}_{t+1} | \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{y}_{t+1:t+p+1} | \mathbf{x}_{t+1}, \boldsymbol{\theta})} w_t \end{aligned} \quad (7.8)$$

The first term of both numerator and denominator are easy to compute. For calculation of the second terms, note the following factorisation:

$$\begin{aligned} p(\mathbf{y}_{t:t-p+1} | \boldsymbol{\theta}, \mathbf{x}_{t-p}) &= p(\mathbf{y}_t | \boldsymbol{\theta}, \mathbf{y}_{t-1:t-p+1}, \mathbf{x}_{t-p}) \times \\ & p(\mathbf{y}_{t-1} | \boldsymbol{\theta}, \mathbf{y}_{t-2:t-p+1}, \mathbf{x}_{t-p}) \cdots p(\mathbf{y}_{t-p+1} | \boldsymbol{\theta}, \mathbf{x}_{t-p}) \end{aligned} \quad (7.9)$$

Each of these terms may be expressed as:

$$\begin{aligned} p(\mathbf{y}_{t-k} | \boldsymbol{\theta}, \mathbf{y}_{t-k-1:t-p+1}, \mathbf{x}_{t-p}) &= \\ & \sum_{\mathbf{x}_{t-k}} p(\mathbf{y}_{t-k} | \boldsymbol{\theta}, \mathbf{x}_{t-k}, \mathbf{x}_{t-p}) p(\mathbf{x}_{t-k} | \boldsymbol{\theta}, \mathbf{y}_{t-p+1:t-k-1}, \mathbf{x}_{t-p}) \end{aligned} \quad (7.10)$$

The first term is the likelihood and the second is the prediction density, both of which are calculated during the forward pass when making the draw from equation (7.6). These may be used when calculating $p(\mathbf{y}_{t+1:t+p+1} | \mathbf{x}_t, \boldsymbol{\theta})$, although a separate forward pass is needed for the calculation of the term, $p(\mathbf{y}_{t+1:t+p+1} | \mathbf{x}_{t+1}, \boldsymbol{\theta})$.

Note that in this case the draw for the parameters does not include any information from the extra received data, $\mathbf{y}_{t+1:t+p+1}$. However, in the case of fixed parameters, the prior rapidly becomes quite strong so any additional information in these states does not affect the posterior for $\boldsymbol{\theta}$ very much.

7.3.2.1 Method

Let us assume we have available at time t , N samples, $\mathbf{x}_t^{(j)}$, from the distribution $p(\mathbf{x}_t | \mathbf{y}_{0:t})$, possibly with associated weights $w_t^{(j)}$. At time $t + 1$ the additional observation y_{t+p+1} is available. These samples may be updated to the distribution $p(\mathbf{x}_{t+1} | \mathbf{y}_{0:t+1})$ as follows:

1. For each j :
 - (a) Draw $\theta^{(j)}$ from $p(\theta | \mathbf{x}_{0:t}^{(j)}, \mathbf{y}_{0:t})$.
 - (b) Draw $\mathbf{x}_{t+1}^{(j)}$ from equation (7.6).
 - (c) Calculate importance weights given by equation (7.8).
2. Output or store the estimate $\hat{f}(\mathbf{x}_{t+1}) = \sum_{i=1}^N f(\mathbf{x}_{t+1}^{(i)}) \frac{w_{t+1}^{(i)}}{\sum_{j=1}^N w_{t+1}^{(j)}}$.
3. If Effective sample size is too small, resample and reset importance weights to equal values.

7.3.3 The Smoothing Density

Rather than using the particle filter to estimate a filtering density, the smoothing density, $p(\mathbf{x}_{0:t-p} | \mathbf{y}_{0:t})$, may be represented directly. Ideologically this is a better approach, although it introduces some practical problems.

The fixed-lag smoothing distribution is given by:

$$\begin{aligned}
 p(\mathbf{x}_{0:t-p+1} | \mathbf{y}_{0:t+1}) &= \frac{p(\mathbf{y}_{t+1} | \mathbf{y}_{0:t}, \mathbf{x}_{0:t-p+1}) p(\mathbf{x}_{0:t-p+1} | \mathbf{y}_{0:t})}{p(\mathbf{y}_{t+1} | \mathbf{y}_{0:t})} \\
 &= \frac{p(\mathbf{y}_{t+1} | \mathbf{y}_{0:t}, \mathbf{x}_{t-p+1}) p(\mathbf{x}_{t-p+1} | \mathbf{y}_{t-p+1:t}, \mathbf{x}_{0:t-p}) p(\mathbf{x}_{0:t-p} | \mathbf{y}_{0:t})}{p(\mathbf{y}_{t+1} | \mathbf{y}_{0:t})}
 \end{aligned} \tag{7.11}$$

Let us define the importance sampling distribution in the usual way:

$$\begin{aligned}
 \pi(\mathbf{x}_{0:t-p+1} | \mathbf{y}_{0:t+1}) &= \pi(\mathbf{x}_0 | \mathbf{y}_{0:p}) \prod_{k=1}^{t-p+1} \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{0:k+p}) \\
 &= \pi(\mathbf{x}_{t-p+1} | \mathbf{x}_{0:t-p}, \mathbf{y}_{0:t+1}) \pi(\mathbf{x}_{0:t-p} | \mathbf{y}_{0:t})
 \end{aligned}$$

then the unnormalised importance weight satisfies:

$$\begin{aligned} w(\mathbf{x}_{0:t+1}) &= \frac{p(\mathbf{x}_{0:t-p+1} | \mathbf{y}_{0:t+1})}{\pi(\mathbf{x}_{0:t-p+1} | \mathbf{y}_{0:t+1})} \\ &= w(\mathbf{x}_{0:t}) \frac{p(\mathbf{y}_{t+1} | \mathbf{y}_{t-p:t}, \mathbf{x}_{0:t-p+1}) p(\mathbf{x}_{t-p+1} | \mathbf{y}_{t-p+1:t}, \mathbf{x}_{0:t-p})}{\pi(\mathbf{x}_{t-p+1} | \mathbf{x}_{0:t-p}, \mathbf{y}_{0:t+1}) p(\mathbf{y}_{t+1} | \mathbf{y}_{0:t})} \end{aligned} \quad (7.12)$$

For a general distribution, the term $p(\mathbf{y}_{t+1} | \mathbf{y}_{t-p:t}, \mathbf{x}_{0:t-p+1})$ cannot be evaluated. In some models we may assume that for a sufficiently large value of the lag, p , the term $p(\mathbf{y}_{t+1} | \mathbf{y}_{t-p:t}, \mathbf{x}_{0:t-p+1})$ will be approximately constant for all values of $\mathbf{x}_{0:t-p+1}$.³ This is saying simply that future observations are independent of states from the distant past. This assumption is taken to be valid in the following derivations. Therefore, the method is limited to how small a lag may be chosen, although this is often not a restriction in practice since this assumption also implies that the lag is large enough to gain the full benefit of fixed-lag methods.

We now consider generating samples from the joint distribution $p(\mathbf{x}, \boldsymbol{\theta} | \mathbf{y})$ by using the following factorisation:

$$p(\mathbf{x}_{t-p+1}, \boldsymbol{\theta} | \mathbf{y}_{0:t}, \mathbf{x}_{0:t-p}) = p(\mathbf{x}_{t-p+1} | \mathbf{x}_{0:t-p}, \mathbf{y}_{t-p+1:t}, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathbf{y}_{0:t}, \mathbf{x}_{0:t-p}) \quad (7.13)$$

Note that we can easily sample from the distribution:

$$(\mathbf{x}_{t-p+1}, \tilde{\mathbf{x}}_{t-p+2:t}) \sim p(\mathbf{x}_{t-p+1}, \mathbf{x}_{t-p+2:t} | \mathbf{x}_{0:t-p}, \mathbf{y}_{t-p+1:t}, \boldsymbol{\theta}) \quad (7.14)$$

as before. The imputed values, $\tilde{\mathbf{x}}_{t-p+2:t}$, are retained for use in sampling $\boldsymbol{\theta}$ in the next iteration. We cannot easily sample from $p(\boldsymbol{\theta} | \mathbf{y}_{0:t}, \mathbf{x}_{0:t-p})$; however we may sample from the importance distribution:

$$\pi(\boldsymbol{\theta} | \mathbf{y}_{0:t}, \mathbf{x}_{0:t-p}) = p(\boldsymbol{\theta} | \mathbf{y}_{0:t}, \mathbf{x}_{0:t-p}, \tilde{\mathbf{x}}_{t-p+1:t}) \quad (7.15)$$

³This is a similar approximation to that made in the trace-back stage of Viterbi equalisers.

This introduces an incremental importance weight:

$$\begin{aligned}
 w_{t+1} &= \frac{p(\boldsymbol{\theta} | y_{0:t}, \mathbf{x}_{0:t-p})}{p(\boldsymbol{\theta} | y_{0:t}, \mathbf{x}_{0:t-p}, \tilde{\mathbf{x}}_{t-p+1:t})} w_t \\
 &= \frac{\sum_{\tilde{\mathbf{x}}_{t-p+1:t}} p(\boldsymbol{\theta} | y_{0:t}, \mathbf{x}_{0:t-p}, \tilde{\mathbf{x}}_{t-p+1:t})}{p(\boldsymbol{\theta} | y_{0:t}, \mathbf{x}_{0:t-p}, \tilde{\mathbf{x}}_{t-p+1:t})} w_t
 \end{aligned} \tag{7.16}$$

At first glance it may appear that the numerator of equation (7.16) is expensive to calculate, however all the values of terms will be calculated when sampling from equation (7.14); only a small number of additional calculations are required.

7.3.3.1 Method

Let us assume we have available at time t , N samples, $\mathbf{x}_{t-p}^{(j)}$, from the distribution $p(\mathbf{x}_{t-p}^{(j)} | y_{0:t})$, possibly with associated weights $w_t^{(j)}$. At time $t+1$ the additional observation y_{t+1} is available. These samples may be updated to the distribution $p(\mathbf{x}_{t-p+1} | y_{0:t+1})$ as follows:

1. For each j :
 - (a) Draw $\boldsymbol{\theta}^{(j)}$ from equation (7.15).
 - (b) Draw $\mathbf{x}_{t-p+1}^{(j)}, \tilde{\mathbf{x}}_{t-p+2:t}^{(j)}$ from equation (7.14) and draw the additional $\tilde{\mathbf{x}}_{t+1}^{(j)}$ from $p(\mathbf{x}_{t+1} | y_{t+1}, \tilde{\mathbf{x}}_t^{(j)}, \boldsymbol{\theta})$.
 - (c) Calculate importance weights given by equation (7.16).
2. Output or store the estimate $\hat{f}(\mathbf{x}_{t-p+1}) = \sum_{i=1}^N f(\mathbf{x}_{t-p+1}^{(i)}) \frac{w_{t+1}^{(i)}}{\sum_{j=1}^N w_{t+1}^{(j)}}$.
3. If the weights have a high variance (or equivalently, the effective sample size is too low), rejuvenate by resampling with probability proportional to the weights and reset the weights to equal values.

7.4 Data Windowing

7.4.1 Motivation and Formulation

Many engineering applications involve very large data sets: for example CD-quality audio is represented by 16-bit numbers arriving at 44.1kHz. It is not possible to do complex processing on such large sets in one go. As a result, there exist many analysis methods that operate on a small portion of that data at a time. Perhaps the best example of this is the short-time Fourier transform.

Sequential methods do allow us to process the whole data set, but to overcome practical issues, we may borrow the idea of choosing a window of the data to make our estimates. However rather than choosing adjacent or overlapping windows, we choose a sliding window. This window is used to aid fixed-parameter estimation, removing the necessity to store the entire state and observation history.

Let us consider a window of size r . We wish to draw samples from the density $p(\theta | x_{0:t}, y_{0:t})$. Instead only the latest r samples will be used:

$$\theta \sim p(\theta | x_{t-r+1:t}, y_{t-r+1:t}) \quad (7.17)$$

This technique does not use all the available information, but it will result in reduced computational complexity and storage requirements.

7.4.2 Window Shape

Equation 7.17 describes a rectangular window, where all data points considered have equal weighting on the resulting density. It is possible to choose other window shapes. Consider the factorisation:

$$p(\theta | x_{0:t}, y_{0:t}) = p(\theta | x_0) \prod_{i=1}^t \frac{p(x_i | x_{i-1}, \theta)}{p(x_i | x_{i-1})} \quad (7.18)$$

which relies on the Markovian dependency of the states. Introducing a data window of length r is the equivalent of saying

$$\theta \sim \hat{p}(\theta | x_{0:t}, y_{0:t}) = p(\theta | x_0)^{\alpha_0} \prod_{i=1}^t \left[\frac{p(x_i | x_{i-1}, \theta)}{p(x_i | x_{i-1})} \right]^{\alpha_i} \quad (7.19)$$

where $0 < \alpha_i < 1$. A rectangular window is therefore defined by

$$\alpha_i = \begin{cases} 1 & \text{if } t - r + 1 < i < t \\ 0 & \text{if } i \leq t - r \end{cases} \quad (7.20)$$

The choice of window shape may affect the estimation properties. Returning to the example of the short-time Fourier transform, a rectangular window produces sharp peaks in the estimated spectrum, but there is a lot of spectral leakage, in the form of side-lobes surrounding this peak. The spectral leakage is reduced by choosing a window with smoother edges (e.g. Barlett, Hamming, Hanning), at the expense of smearing the central peak (see, for example, [63, §2.7], [167, §10.1]).

For parameter estimation on a time-series, the behaviour is not so well-known. It certainly seems sensible to have a flat front ($\alpha_i = 1$ for i near t) to the window (so new data is instantly incorporated), but it may be wiser to have a smoother back to the window, so distant previous observations are lost slowly. This is similar to the forgetting factor of the RLS algorithm and has strong connections with the idea of disengagement described in [15].

The possible use of a non-rectangular window may need extra storage or computation over that required for a rectangular window, which is highly dependent on the nature of the problem. The use of non-rectangular windows is beyond the scope of this thesis.

7.4.3 Window Size

The maximum size of the window is limited by the amount of storage and computational power available to perform updates. If we are not directly limited by these constraints, it would seem sensible to choose a size based upon the variance of the estimates obtained. Complexity will be proportional to the product of the number of particles and the size of the window.

For certain classes of distribution we can say that:

$$\text{Var}\{p(\boldsymbol{\theta} | x_{t-q+1:t}, y_{t-q+1:t})\} \propto \frac{1}{q} \quad (7.21)$$

where q is the size of the window.

In this case it seems sensible to choose a window of a size similar to the number of particles used. In the examples given later this leads to a figure of a few hundred. Indeed, with a window of size 200, the variance on the channel parameters due to the length of the window is similar to that obtained when using the Gibbs Sampler described in chapter 5 with a frame size of 200. The excellent results obtained suggest that a window any larger than this would be wasted effort, at least for the case of the communications channel models to which these methods have been applied in this thesis.

7.5 Results from Simulations

The communications system is simulated directly from equation (2.18) using a binary pulse amplitude modulation scheme where bits are encoded as ± 1 . All methods require an estimate of the initial density: this was obtained using a Gibbs sampler as described in chapter 5 which was run over a frame of length 200 samples.

In all cases 250 particle streams were chosen. Resampling took place using the systematic sampling algorithm described in section 6.2.3.2 whenever the effective sample size, as defined by equation 6.11 in section 6.2.4, fell below 200. This is an order of magnitude lower than many implementations of the particle filter, however it is sufficient for our purposes since we can produce a large number of points close to the posterior modes due to the efficient algorithms employed and the comparative simplicity of the discrete state-space system. The input symbol at time t was determined by the marginal MAP estimate, given by the symbol S_k that maximises the expected value of the function:

$$f(\mathbf{u}_t) = \begin{cases} 1 & \text{if } u_t = S_k \\ 0 & \text{if } u_t \neq S_k \end{cases} \quad (7.22)$$

over all values of k . This is calculated simply by counting the number of times each symbol from $\{S_1, S_2, \dots, S_q\}$ occurs across all particles and choosing the symbol with the highest count.

7.5.1 Fixed-Lag Methods

7.5.1.1 Length of Lag

The performance of the fixed-lag MAP approach using various lags is compared in our simulations with the standard sequential imputations methods in which the joint MAP state sequence is estimated after the arrival of all the data.

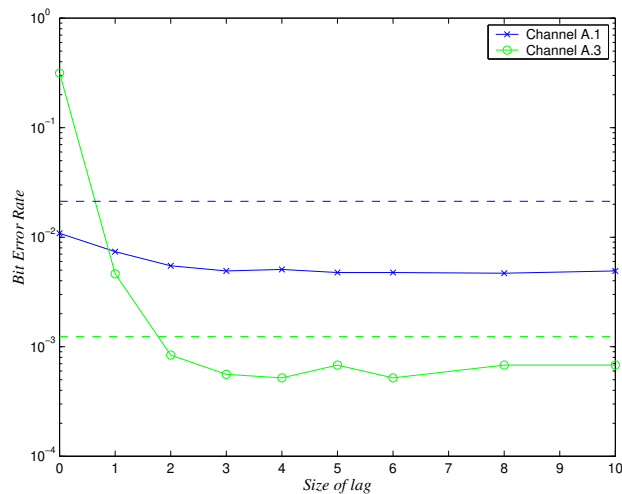


Figure 7.2. Graph of bit error rate against lag for two different channels: channel A.1 and channel A.3. The dashed lines show the results when data is output at the end of the run.

The graphs in Figure 7.2 show the results on bit error rate for blind equalisation on two different channels in the presence of noise with variance 0.1^2 . There is a wide variation between the two channels. This is because channel A.3 has very little power in the first filter tap. Even a lag of one gives a huge increase in performance. More energy is present in the first tap of channel A.1, so there is not such a huge difference between the first and second lags. In both cases it appears that only a small lag of four or more (equal or just longer than the length of the channel) is sufficient to obtain good performance. Any further lag does little to influence the results. The dashed lines indicate the effect of storing the complete

particle histories and outputting the data at the end of the run (as described in the sequential imputations algorithm). There is a significant performance degradation in doing this, arising from the degeneracy of the earlier states. It is clear that outputting the required inference at a fixed lag is therefore highly desirable because:

- Superior performance is obtained.
- Less storage space is required.
- The resampling step can be performed much faster since a small and fixed amount of data needs to be copied.⁴

Having seen the effect of various lags on the performance it is interesting to apply this knowledge to the Gibbs Sampler. It was noted in section 5.4.1.4 that special attention was needed at the frame boundaries. In fact the system of slightly overlapping frames that was used meant that the points at the end of the frame had a small lag, equal to one less than the channel length, and the lag effectively increases linearly as you move towards the beginning of the frame. This overcame most of the performance degradation that one might otherwise see.

7.5.1.2 Performance of the Various Fixed-Lag Algorithms

The lag was chosen to be $p = 15$, five times the length of channels A.1 and A.2. This should be sufficient that optimal performance is obtained, even on the longer channels A.3 and A.4. Differential encoding was not employed, but the bits were inverted manually when the phase ambiguity was incorrectly resolved. In a practical system, either differential encoding or a small number of known bits are required to resolve this ambiguity. Nominally 10,000 points were used for the calculation of bit error rates. This is insufficient to produce accurate results at high SNR, so additional points were processed to ensure the stability of the bit error rates. Occasionally, the curves are not as smooth as would be desired, but there were limits on the computer power available.

⁴This statement is only true for software implementations — if the algorithm were to be implemented in hardware, this copy operation would take place in parallel and there would therefore have no influence on the speed. It would have a huge effect on chip area though!

Computationally the lagged draw of a state (or equivalently a joint draw of the states) is the most expensive part of the algorithms. As a result, tracking the filtering density and using a decision step (section 7.3.1) is by far the fastest method, similar in complexity to the method of sequential imputations [103]. Using the filtering density with look-ahead (section 7.3.2) was the slowest by almost a factor of 2 over tracking the smoothing density (section 7.3.3), due to the extra pass through the lagged data required for calculation of the importance weights. In addition, the computational complexity varies exponentially with the length of the channel when a forward pass is required, but only with the square of the channel length for the simpler update of the filtering density with fixed-lag decision output.

The comparatively poor performance of the filtering density with look-ahead algorithm is perhaps due to this algorithm essentially behaving as a directed filter rather than a smoothing algorithm. All of our other algorithms give a genuinely smoothed output; however it is trivial to add a delayed decision step in the same way as for SIS. By having a lag of seven, with a decision delayed by eight further samples allows a fairer comparison — this version has a similar complexity to that of tracking the smoothing density.

The effect of this is shown in figure 7.3 for channel A.1. Similar results are seen on the other channels. The filtering with look-ahead on its own gives the poorest performance, with a mild improvement when the lag is increased to 15. The use of the decision step after eight samples with a basic lag of seven gives a further improvement, and it is this version of the algorithm that will be presented in the graphs to compare all the fixed-lag filtering methods. However, in contrast to what we have seen when tracking the filtering density, a further performance gain is seen if the results are stored for output at the end of the processed data. This is an important result in itself as it suggests that the degeneracy problem is largely overcome using this directed filtering method, at least for a small, discrete state-space set.

The results for all the algorithms are shown in figures 7.4, 7.5, and 7.6. The reference plot of the bit error rate achieved using the Viterbi algorithm with the known channel is also shown. The final point for the method using the filtering density with look-ahead in figure 7.5 is not shown as a consistent result could not be obtained and memory limitations prevented the longer run required.

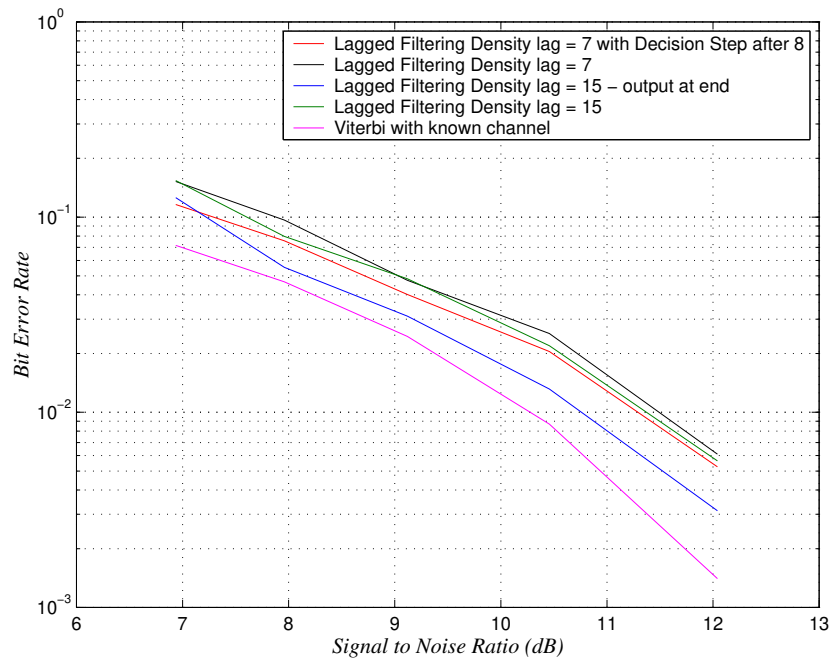


Figure 7.3. Graph of bit error rate against signal-to-noise ratio in simulations using different specifications of the filtering density with look-ahead algorithm.

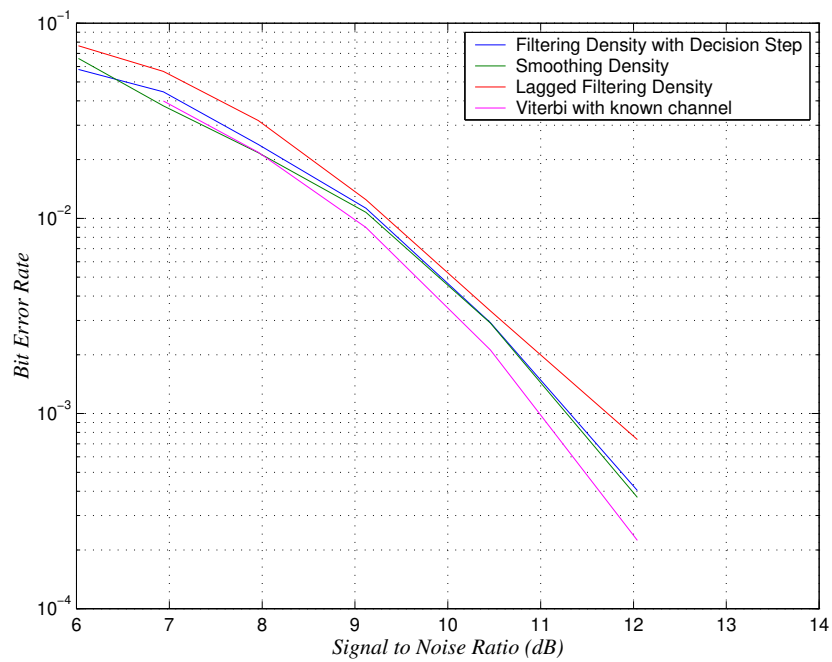


Figure 7.4. Graph of bit error rate against signal-to-noise ratio in simulations using the different SIS algorithms for channel A.2

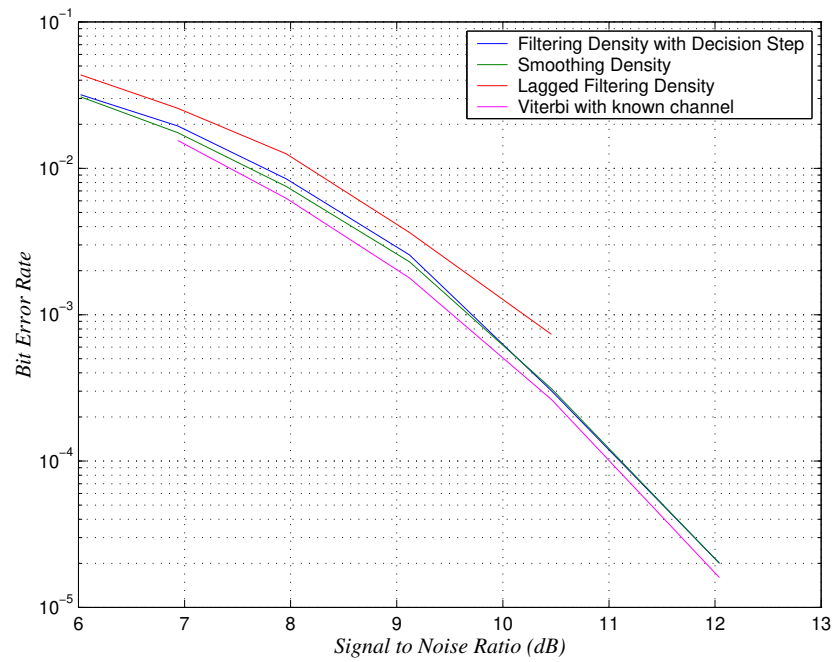


Figure 7.5. Graph of bit error rate against signal-to-noise ratio in simulations using the different SIS algorithms for channel A.4.

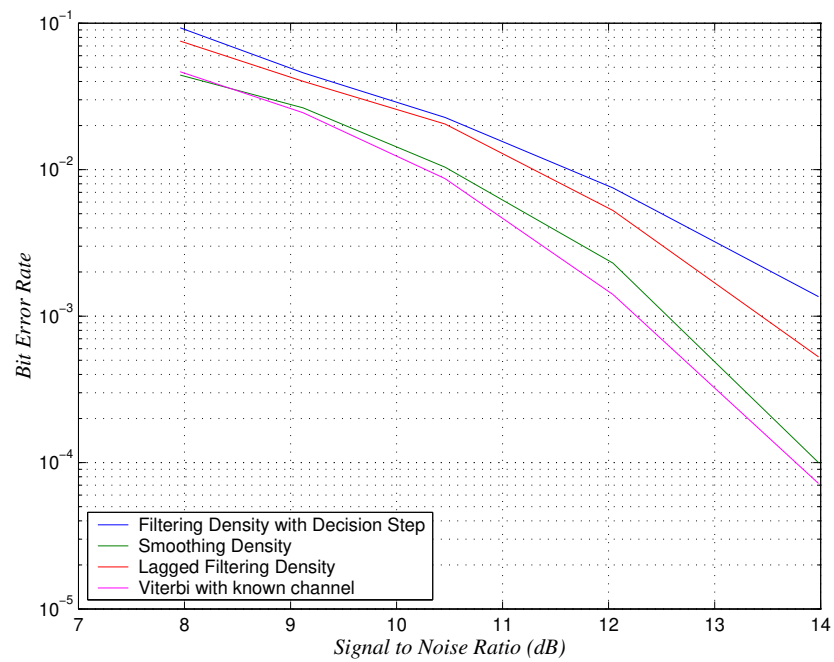


Figure 7.6. Graph of bit error rate against signal-to-noise ratio in simulations using the different SIS algorithms for channel A.1.

The general rule — illustrated on channels A.2 and A.4 in Figures 7.4, and 7.5 respectively — is that the results are very close between all algorithms: in particular the smoothing density tracking and the filtering density with decision step give the best performance, close to the reference Viterbi performance. The smoothing density algorithm possibly has a slight edge. The filtering density with look-ahead has a mild performance penalty, and a high computational burden, and is therefore the least preferred.

Thus far, the filtering density with decision step is the preferred method, being considerably more flexible and much faster than using the smoothing density. However for certain channels, such as that of A.1 shown in Figure 7.6, a very disappointing performance is obtained. The lagged filtering density method finally gets out of last place, with the smoothing density tracking retaining its excellent performance relative to the Viterbi benchmark.

It is not entirely clear why this channel results in a different relative performance of the algorithms, although possible reasons include the null in the frequency response at the Nyquist frequency for channel A.1 or the fact that it is non-minimum phase (see appendix A).

7.5.2 Data Windowing

In order to test the validity of using a window over past data to estimate fixed parameters, it is instructive to compare the performance using a data window with the normal method of using all the past accumulated data. The results shown in Figure 7.7 use a rectangular window of length 200 on channels A.2 and A.4. The performance reference Viterbi algorithm with the channel known is also shown. In this Figure it is clear that there is very little difference in the performance obtained whether a window is used for the parameter estimation or not.

The same result was found for channel A.1 shown in Figure 7.8. In this case the use of a window of data for the estimation of the channel parameters gives an improvement in performance, albeit a very mild one. However the overall performance is not as close to the reference Viterbi algorithm, as was noted in the previous section.

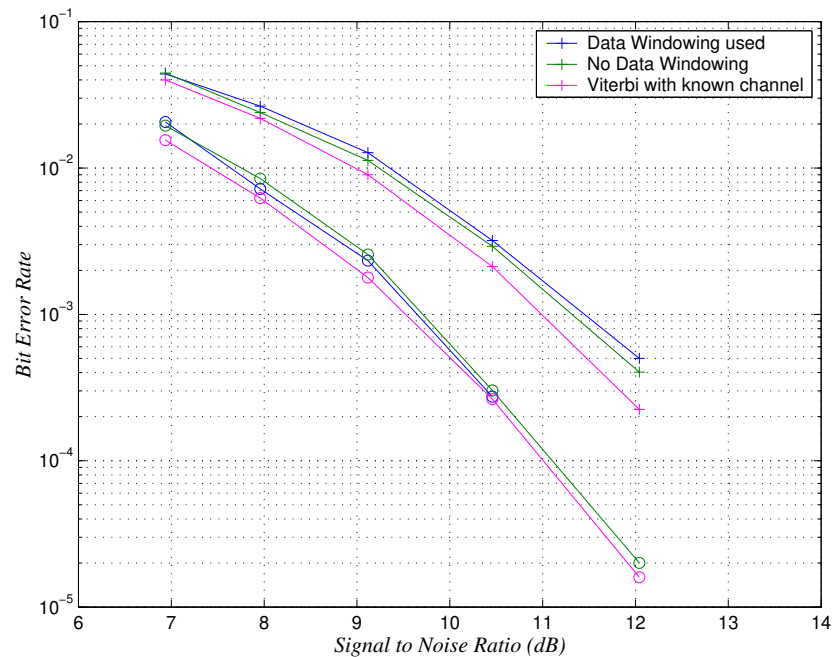


Figure 7.7. Graph of bit error rate against signal-to-noise ratio in simulations using filtering density with decision step algorithms with and without data windowing for channels A.2 (x) and A.4 (o). The reference exact Viterbi is also shown.

7.6 Conclusion

In this chapter two important modifications to the particle filter have been introduced:

Fixed-lag Smoothing is of profound use in a wide variety of practical problems and three separate methods of achieving this were described. Of these, two are preferred, depending on the exact nature of the problem tackled and computational power available:

- **Filtering density with decision step.** This method has a similar level of complexity to tracking the filtering density, and at the same time adds the power intrinsic to fixed-lag methods. It does not appear to perform optimally for every single case, however.

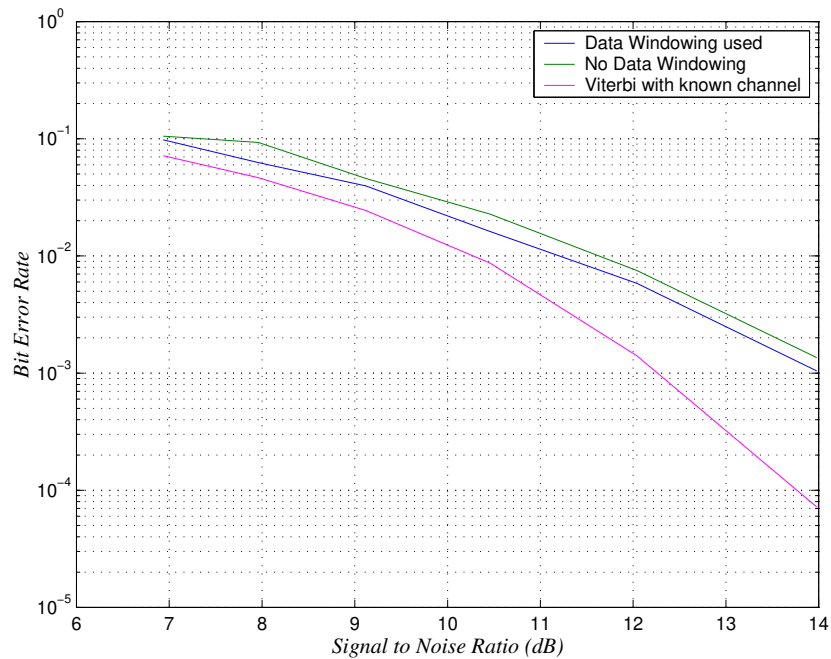


Figure 7.8. Graph of bit error rate against signal-to-noise ratio in simulations using filtering density with decision step algorithms with and without data windowing for channel A.1.

- **Smoothing density.** This method appears to be more robust, although it has two drawbacks: the MCMC-like moves increase the computational complexity considerably and assumptions made mean that the method may only be used in cases where the lag is large enough that any further increase in the lag is unlikely to influence the overall performance.

Data Windowing allows fixed parameters to be estimated within the particle filter framework. The use of a window overcomes the problem of degeneracy, allowing new values of the fixed parameters to be obtained, without requiring the storage of the entire state and observation history.

Examples of these methods in action were given for the digital communications system with time-invariant parameters. Very promising results were obtained.

The next chapter will describe further the synergy of particle filter and MCMC methods and will go on to describe how this may be used to introduce intermediate

bridging densities between the probability distributions at one time step and the next. The applications will include the time-varying Rayleigh fading channel.

Bridging Densities in Particle Filters

8

It is common to encounter problems with the particle filter if the prior and posterior distributions are substantially different. This often occurs when we get an informative observation (the variance of the likelihood is small compared to that of the prior), or if the observation lies in the tails of the distribution. The net effect of this is that there are relatively few particles generated from the prior that lie in regions of high likelihood. The variance of the importance weights becomes large and after resampling, the updated posterior distribution is represented by a very small number of particles or even a single one. This degeneracy is highly undesirable and frequently does not result in a good approximation to the distribution and estimates obtained from it are poor. In addition, if the number of particles is restricted for practical implementational reasons, this degeneracy may be quite common.

It follows from the above discussion that the particle filter lacks robustness: observations inconsistent with the model can cause this degeneracy. With this in mind, it may be wise to choose models which have heavier tailed prior distributions than we believe to be the case. This allows some scope to account for the discrepancy between results on real data, and data simulated from the same model used for analysis. This choice is consistent with the guidelines for choosing a

good importance function.

In this chapter the concept of intermediate bridging densities will be introduced which smooth the transition from the distribution at one time step to the one at the next. These intermediate densities formed are akin to *simulated annealing* between the two time steps. The framework presented will be published in [62]. This technique has important consequences on methods of initialisation of the particle filter: it provides a more robust method for generating initial samples.

This chapter also describes the amalgamation of particle filter and MCMC methods: how MCMC moves may be used within the particle filter framework. This is particularly important as MCMC moves are used to generate samples from the intermediate densities, although a burn-in of the Markov chain is not required.

8.1 Existing Techniques for Avoiding Degeneracy

There exist a number of techniques for alleviating the degeneracy problem without using intermediate distributions. They involve either selecting better prior samples by some means, or moving the resampled points:

Prior Editing. This was introduced by Gordon *et al.* [66] for application in the SIR form of the particle filter. In the sample step, if the new weight of the particle is too small, then this is unlikely to be accepted at the resampling stage. The particle is therefore rejected immediately and an alternative drawn in its place. This methods effectively edits the prior in an ad hoc manner based upon the weighting, but it is effective at reducing degeneracy.

Auxiliary Particle Filter. This method (introduced by Pitt and Shephard [132] and described in section 6.3.3) formalises the above approach so that particles in areas of high likelihood are selected in the sample step using some estimate of their likelihood. The bias by selection in this manner is removed by the introduction of an additional weight.

Improved Importance Function. The choice of an importance function which takes the observation into account can greatly improve the new states that are

estimated resulting in better estimates and lower variance in the resulting weights. However this is often not straightforward with more complicated models.

Resample-Move Algorithm. After resampling, each particle is moved by one or more MCMC iterations. The “moved” particles are then kept and are propagated as usual to form the prior sample at the next iteration.

8.1.1 Resample-Move Algorithm

The resample-move algorithm, introduced in [56, 113], is an important advance which combines the sequential particle filter methods with the more conventional MCMC techniques.

It was noted that after resampling, a set of points from the posterior distribution was obtained, although not all of these points would necessarily be distinct. By using one or more iterations of an appropriate MCMC move, these points should be dispersed better over the posterior distribution. Usually only local MCMC moves (i.e. only moving the current state) would be applied, although there is nothing to stop past states from being moved also. This means that the particles can always be made distinct, even if there is poor overlap between prior and posterior. In addition, if the move is too great for importance sampling to work effectively, the MCMC iterations will move the points closer to the required distribution. The only drawback is that the MCMC iterations can add a significant additional computational burden.

It is worth pointing out that the movement of particles using an MCMC move turns out to be identical to the moves proposed for fixed parameters described in section 6.4.4 and used in section 7.1.1. The concept of re-simulating parameters each time the posterior distribution is updated is equivalent to a move from one place to another within each of these distributions.

8.2 Annealing in Particle Filters

In some cases the difference between prior and posterior distributions is just too large to bridge in a single step, given a finite number of particles. To overcome

this, intermediate distributions are introduced so that each step between these intermediate distributions is small enough to avoid the degeneracy problem.

8.2.1 Simulated Annealing

The term simulated annealing is borrowed from the material science world: to attain a particular crystal structure, the temperature of a substance is raised and the material is cooled at a particular rate so that the correct phase transitions may occur. The optimisation analogue [92] is based upon the principle that increasing temperature allows greater movement about the space and between modes. Simulated annealing is often used to generate points from a highly multi-modal distribution which may be difficult by other methods. For example, when using MCMC methods, the mixing within a Markov chain may not easily allow moves between modes. By raising the “temperature”, the posterior distribution is flattened allowing easier exploration of the space. This concept of temperature is similar to altering the variance of the distribution, where a high temperature corresponds to a large variance. When cooling takes place, an accurate description of the distribution may be obtained, as each mode will have been visited.

8.2.2 Practical Implementation

We wish to move from prior density to the posterior in n steps, containing $n - 1$ intermediate densities. This method is a development of that proposed by Neal [121] for non-sequential annealed importance sampling. The prior distribution will be denoted $p_0(x)$ and the posterior density $p_n(x)$.¹ The intermediate densities, $p_j(x)$, which would be chosen in conventional simulated annealing, are given by:

$$p_j(x) = p_n(x)^{\beta_j} \tag{8.1}$$

where $j = 1 \dots n$, $0 < \beta_j \leq 1$, $\beta_{j-1} < \beta_j < \beta_{j+1}$.

¹Note that this is the opposite way round compared to [121], which reflects the convention in the simulated annealing literature. However, with this notation, $p_j(x)$ conveniently denotes the probability density at the j th intermediate step.

The choice of densities

$$\begin{aligned} p_j(\mathbf{x}) &= p_0(\mathbf{x})^{1-\beta_j} p_n(\mathbf{x})^{\beta_j} \\ &= p_0(\mathbf{x}) L(\mathbf{x})^{\beta_j} \end{aligned} \quad (8.2)$$

where $L(\mathbf{x})$ is the likelihood, provides a more logical transition from prior to posterior in our setting. At the j th step, the i th particle has an update to its weight given by:

$$w^{(i,j)} = w^{(i,j-1)} \frac{p_j(\mathbf{x}^{(i,j)})}{p_{j-1}(\mathbf{x}^{(i,j)})} \quad (8.3)$$

The algorithm then proceeds as follows:

1. Set $w^{(i,0)} = w_t^{(i)}$.
2. Prior step ($j = 1$):
 - (a) Draw from the prior, for each particle, in the usual fashion.
 - (b) Calculate the new importance weight, $w^{(i,1)}$, using equation (8.3).
3. For each step j from $2 \dots n$:
 - (a) For each particle i from $1 \dots N$:
 - i. Draw the new values for states, $\mathbf{x}^{(i,j)}$, by performing one or more iterations of a Markov chain with stationary distribution $p_{j-1}(\mathbf{x})$ and starting point $\mathbf{x}^{(i,j-1)}$. This is the equivalent of drawing from the effective prior at the j th step.
 - ii. Calculate the new importance weight, $w^{(i,j)}$, using equation (8.3).
 - iii. [Optional] Resample in the usual manner (as discussed in section 8.2.4), if required.
4. Set $w_{t+1}^{(i)} = w^{(i,n)}$.
5. Resample if required, as usual.

Note that if there are no intermediate densities, i.e. $n = 1$, the algorithm proceeds as in the usual operation of the particle filter.

8.2.3 Cooling Schedules

The previous section describes the algorithm in detail, given a choice of the sequence $\{\beta_j\}$. This section will propose how to choose such a sequence. Popular cooling schedules are:

- **Logarithmic.** This is a very slow cooling schedule with guaranteed convergence under the conventional simulated annealing scheme, given by: $\beta_j = \log_n(j)$.
- **Arithmetic.** The β_j are spaced uniformly from 0 to 1: $\beta_j = \frac{j}{n}$.
- **Geometric.** Each successive value for β is calculated by multiplying the previous value by a constant factor, determined by the chosen value β_1 and the number of steps: $\beta_j = \left(n \sqrt[n]{\frac{1}{\beta_1}} \right)^{j-1}$.
- **Hybrid.** This is a mix of arithmetic and geometric schedules: a few small arithmetic steps, followed by geometric cooling for the remaining period. This has a number of extra controllable parameters: the ratio determining the number of steps in each region and the change-over point. This method has the advantage that small steps may be taken initially without resulting in very large steps at the end.
- **Exponential.** $\beta_j = \exp(-c(n-j)^{1/D})$, where D is the dimensionality of the parameter space.

The aim of cooling in simulated annealing is to allow a smooth transition from a high temperature (where exploration is easy and larger steps to move between modes are possible) to a correct temperature in which we are producing points drawn from the correct distribution. In our application, a set of intermediate densities is chosen to move from prior to posterior. The basic idea is to try to minimise the final variance in the weights. To achieve this, the losses at each step should be approximately equal. In practice, the geometric and hybrid schedules appear to provide a reasonable balance between taking small enough steps and not taking too many.

8.2.4 Resampling at Intermediate Stages

The method described by Neal [121] works well provided that adjacent distributions in the cooling schedule are not too far apart. As each step introduces more variation in the importance weights, too many steps, introducing a small amount of variation, or a few steps, each introducing a large amount of variation, results in the familiar degeneration. As such, Neal's formulation has limited application: while it improves performance when observations lie well within the tails of the prior, for complete outliers (such as those inconsistent with the analysis model) we gain very little over simpler methods such as resample-move. However, the performance on multi-modal distributions may be better, in accordance with the relative advantages of simulated annealing versus MCMC.

The additional computation required for such a high number of intermediate steps (typically hundreds) makes the algorithm impractical for online applications: in many cases we could do better by running an stand-alone MCMC algorithm at each step, although convergence would need to be monitored.

If we look at a run just through the intermediate distributions, the operation is identical to the resample-move algorithm, but without the resampling. By adding in a resampling step before the MCMC moves, we regain the complete sample size, since all particles are equally weighted after resampling (although we still have resampling losses as described in section 6.2.3). This should allow us to space the intermediate distributions much further apart, without the losses at each stage adding up.

8.2.5 Dynamic Step Size

The sequential annealed importance sampler described thus far can, in principle, allow us to update from the prior to posterior given any observation, no matter how unlikely that observation may be. However in practice, a cooling schedule must still be chosen. Neal's arguments for a geometric cooling schedule in [121] assume that the log weights have a Gaussian distribution (by appealing to Central Limit Theorem arguments). The distribution of the log weights observed in experiments on both AR and MA models is far from Gaussian, particularly if the number of particles is not that large. In fact, Neal uses a hybrid scheme so the number of steps, the number of arithmetically spaced steps and the changeover point remain as parameters that require adjusting to the problem tackled.

The computational complexity is proportional to the number of steps, so to minimise the computational cost it is desirable to choose the minimum number of steps, adjusting the distance between the intermediate distributions so that each step is of a similar size. To achieve this, it seems reasonable to choose steps such that each step reduces the effective sample size by a certain factor. If we then resample at each step, this is similar to running the particle filter without resampling until the effective sample size becomes too low (section 6.2.3). At the same time we are choosing the minimum number of steps required to move from prior to posterior without degeneracy occurring. In practice, we may choose the required effective sample size to be some fraction of the total number of particles (somewhere in the range 20%–80% is a good start).

Note that the weights introduced at step j are equal to

$$w^{(i,j)} = L(y, x^{(i,j)})^{\beta_j - \beta_{j-1}} \quad (8.4)$$

so, in principle, it is possible to choose $\delta\beta_j = \beta_j - \beta_{j-1}$ to get the required effective sample size. In the general case, this is not straightforward to calculate. In addition, the effective sample size is not a particularly reliable estimator of the degeneracy of the distribution (particularly if it is very small or close to the number of particles). It makes sense therefore to find a $\delta\beta_j$ such that the required effective sample size is only approximately correct.

The number of steps is automatically determined in this process, by simply limiting the value of β_j such that it does not exceed unity. The only downside is that the last step is often quite small relative to the previous steps. It is therefore prudent to check whether the next value of β_j is close enough to unity that a slightly larger step could be taken.

8.2.5.1 Numerical Optimisation Methods

The first approach tried is to use a numerical optimisation algorithm to choose the value of $\delta\beta$ to give the (approximate) effective sample size required. A suitable cost-function is the absolute value difference between the calculated effective sample size (equation 6.2.4) and the required effective sample size. We may also place a bound on the space such that $0 \leq \delta\beta_j \leq 1 - \beta_{j-1}$. We have seen from the discussion of cooling schedules, that the effective sample size is highly nonlinear

in $\delta\beta_j$, so the numerical optimisation will perform better operating on $\log(\delta\beta_j)$.

8.2.5.2 Heuristic Methods

There exist a few adaptive schedules in the literature for dynamic step size in simulated annealing, based upon physical ideas such as constant rate of entropy reduction [126, 153]. It is not clear how effective they would be in the context of a particle filter. The methods presented below are ones discovered by experimentation and are known to work within the particle filter framework on certain classes of problems, although no claims for optimality, or even efficiency, are made.

Log Weights. It is often better to store the log weights rather than the actual weights for numerical reasons. In certain cases (e.g. the observation noise is from the exponential family) we can calculate log weights directly without first calculating the real weights. Therefore, in these cases it makes sense to choose some measure on the log weights so as to determine the value of $\delta\beta_j$ for a reasonable step-size.

The use of the *variance of the log weights* to determine the step size is proposed. This has the advantage that $\delta\beta_j$ may be calculated as

$$\delta\beta_j = \sqrt{\frac{V}{\text{Var}\{L(\mathbf{y}, \mathbf{x}^{(1:N,j)})\}}} \quad (8.5)$$

where V is parameter controlling the step size. Its value will vary depending on the application.

Overlap of Normal Distributions. An alternative method may be derived based upon the estimated overlap of the distributions under Gaussian approximations. To develop the method, the following assumptions are made:

- The propagated prior sample from the j th intermediate density is Gaussian, given by the distribution $\mathcal{N}(\mu_j, \sigma_j^2)$ (including those from the last time-step, $\mathcal{N}(\mu_0, \sigma_0^2)$).
- The likelihood is also Gaussian, given by $\mathcal{N}(y, \sigma^2)$.

- We wish the mean of the next intermediate density, μ_j , to satisfy:

$$|\mu_j - \mu_{j-1}| \approx k\sigma_{j-1} \quad (8.6)$$

for some value of k of $O(1)$, where k is a step-size parameter.

It is clear that many cases, these assumptions will not hold. However, it is hoped that this method will still provide a reasonable guide as to choosing a step size, since this is not required to be particularly exact. It is unlikely that this method will be very effective, for example, when tracking multi-modal distributions.

In assuming that the densities are Gaussian, the intermediate density is also Gaussian, the mean, μ_j , and variance, σ_j^2 , of which we may calculate analytically. The mean is given by:

$$\mu_j = \sigma_j^2 \left(\frac{\mu_{j-1}}{\sigma_{j-1}^2} + \frac{\delta\beta_j y}{\sigma^2} \right) \quad (8.7)$$

$$= \left(\frac{1}{\sigma_{j-1}^2} + \frac{\delta\beta_j}{\sigma^2} \right)^{-1} \left(\frac{\mu_{j-1}}{\sigma_{j-1}^2} + \frac{\delta\beta_j y}{\sigma^2} \right) \quad (8.8)$$

Setting $|\mu_j - \mu_{j-1}| = k\sigma_{j-1}$ and solving for $\delta\beta_j$, we obtain:

$$\delta\beta_j = \frac{k\sigma^2}{\sigma_{j-1}(|y - \mu_{j-1}| - k\sigma_{j-1})} \quad (8.9)$$

The values of μ_{j-1} and σ_{j-1} are obtained from estimates using the particles available.

It is possible to obtain a value for $\delta\beta_j$ that is less than 0. The interpretation of this is that we would need to overcool the system to get the desired overlap. Our requirements are only that a minimum overlap is required, so such a value tells us that we may reach our final distribution in one further step.

8.2.6 Strategies for Choosing a Cooling Schedule

In many case we are limited in the amount of computation available at each step. Computational complexity is proportional to the product of the number of parti-

cles and the number of steps:

$$(\text{Computational complexity}) \propto n \times N \quad (8.10)$$

If we assume we want the same number of particles at the end of each step,² most practical limitations are likely to have one of the following forms:

- **Fixed number of steps, n , and particles, N .** This is likely to arise where a highly parallel architecture (enough to process all particles at the same time) is present, for example on dedicated DSP chips or a custom ASIC/FPGA. No advantage is gained by using fewer particles since some nodes will be left idle. In addition, by reducing the number of steps we also gain little (unless it is by a factor of 2 or more) since the limitation is now time to process. Eliminating one step will have all nodes idle (since the clock speed of the processor will be fixed).
- **Limited computational power.** This would be found in practice if no, or limited, parallelisation over the particles is present e.g. running in software on a PC, when output is required at a fixed rate. We need to find a method of determining the number of steps first, then reducing the number of particles to fit the limited power.

In the former case the problem is simply that of choosing a cooling schedule and the parameters for it. Geometric cooling has the advantages that it is fairly efficient, and requires only one additional parameter (as n is fixed): the first step size, β_1 . If this were estimated, perhaps by one of the methods outlined above, then the rest of the processing for the time-step can proceed unattended.

The latter case is somewhat harder to determine for an online application. It is useful to bear in mind an indicator of relative performance of the algorithms: that is to say there is little gain in using a more complex algorithm, if simply increasing the number of particles would produce the desired result.

²If we remove this constraint the behaviour of the particle filter would be much harder to predict: this opens a whole new avenue of research.

8.3 Results for Gaussian Distributions

To demonstrate the effectiveness of annealed importance sampling, let us first consider a one-dimensional case where both prior and likelihood are Gaussian. This lends to an analytic solution for the posterior using the result in appendix B. Results obtained can therefore be compared with the exact values calculated. In practical terms the prior may represent the range of values a particular variable may take. This variable is observed directly in the presence of noise (which accounts for the likelihood). We shall look at the posterior distributions obtained by the various particle filters as well as the inferred mean and variance.

8.3.1 Method

The prior density is taken to be Normally distributed with zero mean and unity variance. The likelihood is also normally distributed, with controllable mean and variance 0.1^2 . Two iterations of the Markov chain are done at each step (i.e. $n_{iter} = 2$), where each iteration consists of three Metropolis-Hastings steps with Gaussian proposals of variances 0.1^2 , 0.3^2 and 0.5^2 . The number of particles is 10,000. Until we start using dynamic step size adjustment, geometric cooling is used with $\beta_1 = 0.0005$. This value was chosen as it gave good results for most of our examples, although in practice the exact choice depends on shape and parameters of the prior and posterior distributions. A value too small leads to many small steps initially, and too few large steps later on. A value too large results in problems at the first step, or too large a degradation in effective sample size.

The annealed importance sampler is compared to the standard particle filter (with the prior as the importance function) and the resample-move method. The auxiliary particle filter was not used in this comparison as it is difficult to come up with a suitable importance function. If this model were to be used in an application, the true density would be available to draw samples from.

In order to balance out the extra computational cost required for annealing, it is useful to see what benefits may be obtained by allowing more computational time to the other algorithms. This may be done in the following fashion for an annealed importance sampler with n steps and n_{iter} MCMC iterations at each step.

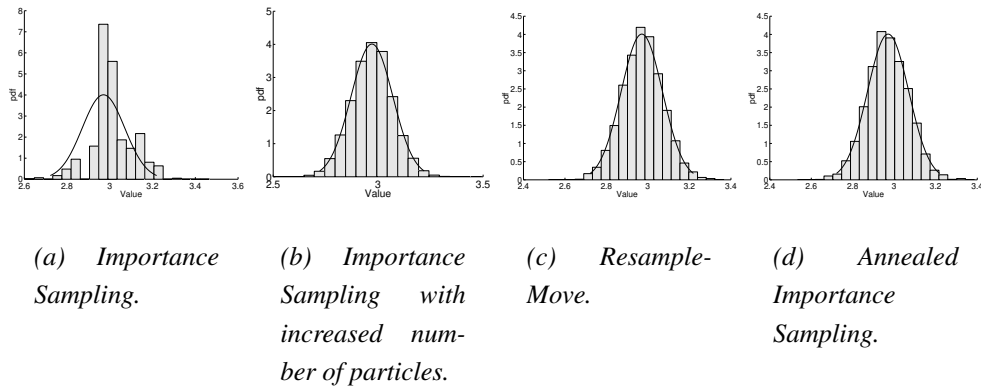


Figure 8.1. Histograms of posterior densities obtained for an observation with a value of 3. The solid curve on each plot is the exact result, obtained analytically.

- For the standard particle filter, increase the number of particles by a factor of n .
- For the resample-move algorithm, increase the number of MCMC steps to $n \times n_{iter}$, so that the total number of MCMC iterations remains the same.

8.3.2 Simulated Annealing Without Resampling

The results are presented in the form of histograms of the posterior densities obtained from the particles along with the true density which is calculated analytically. The first examples show the benefits and pitfalls of simulated annealing with no resampling at the intermediate densities (Figure 8.1), with 150 steps taken between prior and posterior. In this as in all of these results, $n_{iter} = 3$. In the first instance we consider an observation with the value of 3. This is on the limit for the plain importance sampler with 10,000 particles: only a few samples from the prior lie in areas of high likelihood (Figure 8.1(a)). Scaling the number of particles improves matters (Figure 8.1(b)), however the memory requirements for 1,500,000 particles are considerable! Both the resample-move algorithm and the annealed particle filter have no trouble in estimating the posterior distribution as demonstrated in Figure 8.1(c-d).

The means and variances obtained are given in table 8.1 It is interesting to note that the mean obtained by the importance sampler is quite good, but the higher-

order moments are less accurate.

Method	Mean	Variance
Importance sampler	3.010	0.0921
Importance sampler (increased number of particles)	2.969	0.0990
Resample-Move	2.973	0.0985
Annealed importance sampler	2.970	0.0993
Real (analytic result)	2.970	0.0995

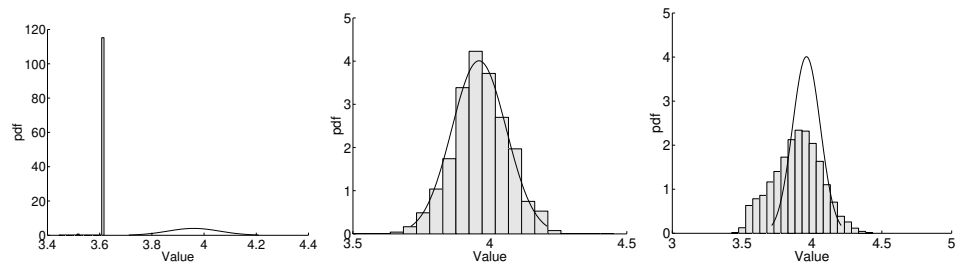
Table 8.1. Means and variances for an observed value of 3.

If the observed value happens to be 4, none of the prior samples lie in regions of high likelihood. Indeed, there is very little overlap between the prior and the posterior. This is clearly demonstrated in figure 8.2(a): nearly all the weight is placed on one point, which is not even in the right place. The resample-move algorithm starts with the same points as the plain importance sampler. A few iterations of the Markov chain are not sufficient to move to the new density (figure 8.2(c)). We observe the results from several Markov chains some of which have converged and some have not.

If we increase the number of iterations by a factor equal to the number of steps via intermediate densities (i.e. to 300 iterations), these chains converge giving the result in figure 8.2(d). The annealing run results in the effective sample size being reduced by a factor of around 4, sufficient for making a good estimate of the posterior distribution. The means and variances obtained are given in table 8.2.

Method	Mean	Variance
Importance sampler	3.615	0.1009
Importance sampler (increased number of particles)	3.966	0.0967
Resample-Move algorithm	3.894	0.1712
Resample-Move (scaled number of steps)	3.960	0.1004
Annealed importance sampler	3.960	0.0993
Real (analytic result)	3.960	0.0995

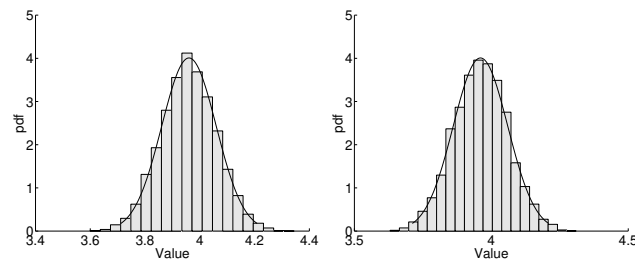
Table 8.2. Means and variances for an observed value of 4.



(a) *Importance Sampling.*

(b) *Importance Sampling (increased number of particles).*

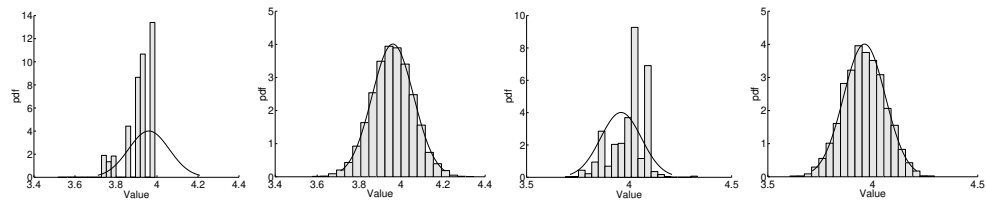
(c) *Resample-Move.*



(d) *Resample-Move (extra iterations).*

(e) *Annealed importance Sampling.*

Figure 8.2. Histograms of posterior densities obtained for an observation with a value of 4. The solid curve on each plot is the exact result, obtained analytically.



(a) Importance Sampler with increased number of particles.

(b) Resample-Move (extra iterations).

(c) No intermediate resampling.

(d) Resampling at intermediate densities.

Figure 8.3. Histograms of posterior densities obtained for an observation with a value of 4 using the annealed importance sampler with 10 intermediate densities, with and without resampling at the intermediate densities. The solid curve on each plot is the exact result, obtained analytically.

8.3.3 Simulated Annealing With Resampling

Let us now compare the results of annealing runs with and without resampling with just 10 steps through intermediate densities. The algorithms and parameters are otherwise identical. An observed value of 4 is used as in the previous example, so the previous results for the importance sampler and resample-move algorithm can be used in a direct comparison.

To make comparisons of similar computational complexity, further simulations were run to reflect the reduced number of bridging densities: Figures 8.3(a) and 8.3(b) show the new graphs. The result obtained in the annealing run without resampling is shown in Figure 8.3(c). The problems illustrated arise not because the sample size is severely impoverished in a single step of the algorithm, but rather that the sum of the losses over all the stages are too great. By resampling at each intermediate stage, we get a clear improvement, as seen in Figure 8.3(d). The means and variances obtained are given in table 8.3.

In all the following results we shall resample at each intermediate step.

Method	Mean	Variance
Importance sampler (increased number of particles)	3.917	0.0700
Resample-Move (scaled number of steps)	3.958	0.0993
Annealed importance sampler (no intermediate resampling)	3.999	0.0850
Annealed importance sampler (resampling at intermediate densities)	3.960	0.0983
Real (analytic result)	3.960	0.0995

Table 8.3. Means and variances for an observed value of 4 when intermediate resampling is used.

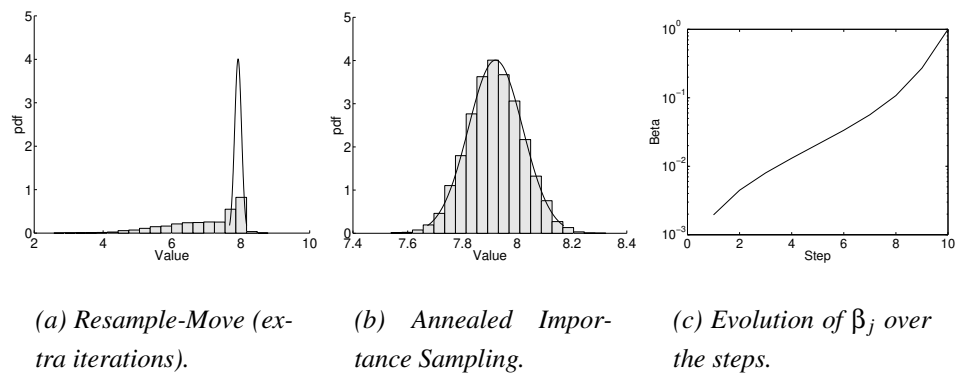


Figure 8.4. Histograms of posterior densities obtained for an observation with a value of 8. The solid curve on plots (a) and (b) is the exact result, obtained analytically.

8.3.4 Dynamic Step Size

8.3.4.1 Numerical Optimisation Methods

The annealed particle filter was run using a numerical optimisation routine to determine the step-size in the manner described in section 8.2.5.1. The desired effective sample size is 2,500 ($= \frac{10,000}{4}$). The Nelder-Mead simplex (direct search) method was used with the function value tolerance set high ($= 20$), since errors of 20–30 in the actual effective sample size can easily be tolerated.

Figure 8.4 shows the histograms obtained with an observed value of 8. This value is more consistent with an outlier due to the model not being completely representative of the physical process generating the data. The algorithm chooses 10 intermediate steps. Figure 8.4(a) shows the histogram for the resample-move

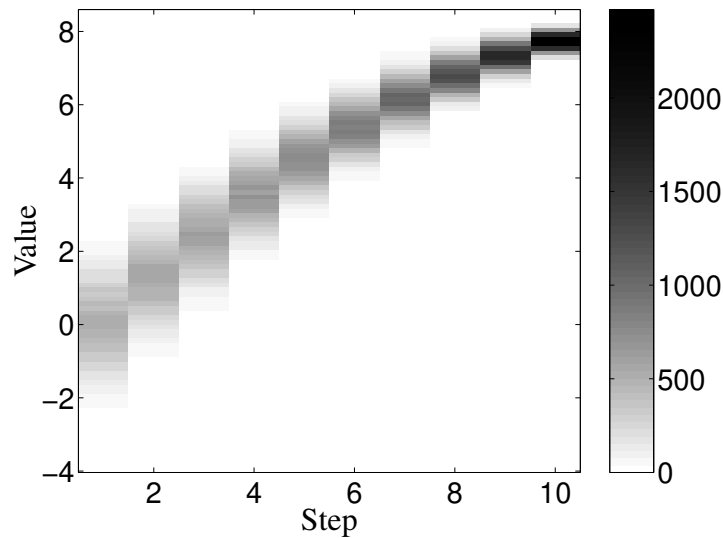


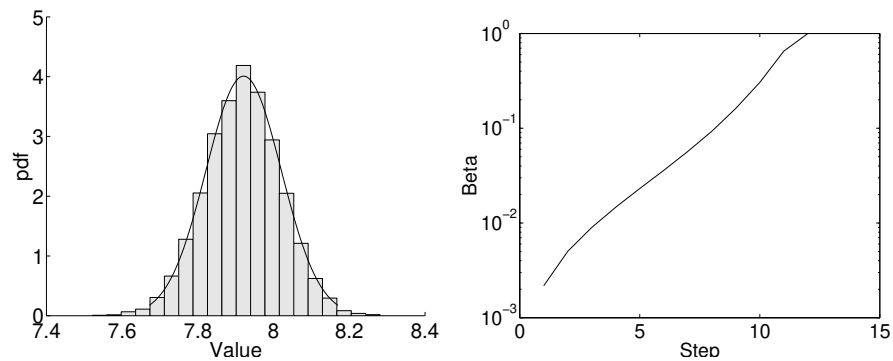
Figure 8.5. A Series of histogram plots showing the intermediate probability densities. Darker grey represents higher probability density.

algorithm (with increased number of iterations). Again we see evidence of a lack of convergence from the Markov chain moves. Figure 8.4(b) shows the annealing run obtaining an accurate description of the posterior distribution. Figure 8.4(c) shows the values of the β_j at each step on a logarithmic scale. The resulting plot is close to a straight line (geometric cooling).

Figure 8.5 shows histograms of all the intermediate densities. We can see that a reasonable overlap between adjacent densities is required, clearly showing why the single-step algorithms do not work well for such outliers.

This method works very well in the sense that it does provide a robust method for moving between any two densities, with only one control parameter which, since it describes the loss (in terms of effective number of particles), is easy to set to a reasonable value.

The main draw-back with the use of optimisation methods is that the computation involved at each step for the optimisation alone is often higher than the other operations of the particle filter. In addition, this appears as part of the sequential operations acting on all of the particles at the same time and cannot be executed in parallel. This limits the practical applications as doubling the number of steps or number of particles is likely to be better overall. It does, however, provide a method for determining what a reasonable number of steps is and the step size



(a) Histogram of the posterior density obtained overlaid with the curve of the analytic result.

(b) Evolution of β_j over the steps.

Figure 8.6. Results using a dynamic step size determined by the variance of the log weights having observed a value of 8.

might be, for comparison with less precise methods for determining step size.

8.3.4.2 Variance of Log Weights

Attention is now turned to the heuristic measures for determining step-size (described in section 8.2.5.2), starting with using the variance of the log weights as a guidance to the effective sample size. One example is shown here, using an observed value of 8. The posterior density obtained, shown in Figure 8.6(a), matches the analytic result very well.

The plot in Figure 8.6(b) shows the evolution of the values of β_j over the intermediate steps. We can see that this lies somewhere in between geometric cooling (a straight line) and the result obtained from numerical optimisation.

A histogram of the intermediate and final densities is shown in Figure 8.7. As before, there is a reasonable overlap of all the intermediate distributions, ensuring reliable operation.

8.3.4.3 Overlap of Normal Distributions

We would expect this method to work very well as the assumptions fit the underlying model for this simple example. One example is given here: an observation

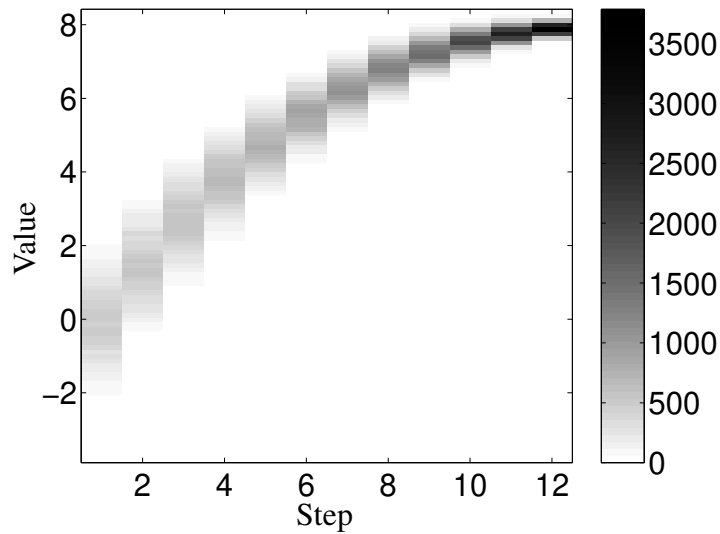


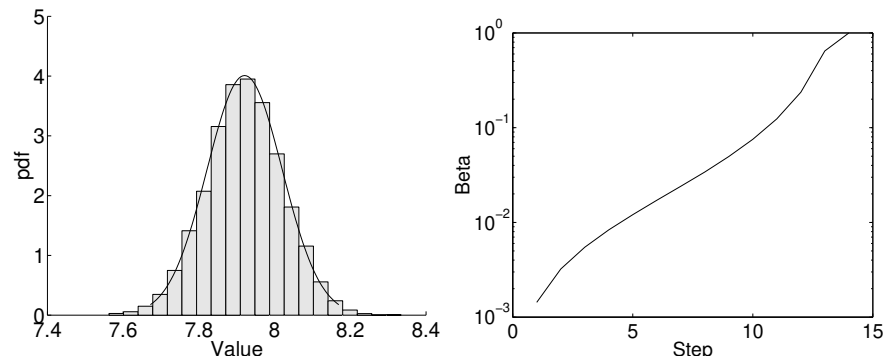
Figure 8.7. A Series of histogram plots showing the intermediate probability densities for 12 intermediate densities selected by the variance of the log weights. Darker grey represents higher probability density.

with the value 8. The value of $k = 1$ was chosen. This gives a very consistent set of values for effective sample size, around $0.4N$. A histogram of the final density obtained and the values of β_j obtained are shown in Figure 8.8. A set of histograms of the densities obtained is shown in Figure 8.9. The mean and variances obtained are shown in table 8.4.

Method	Mean	Variance
Annealed importance sampler	7.922	0.0996
Real (analytic result)	7.920	0.0995

Table 8.4. Means and variances for an observed value of 8 using the annealed importance sampler with intermediate densities selected by the overlap of normal distributions.

This demonstrates that the method works very effectively and converges to the desired solution in 12 intermediate steps. However, its use is as yet unproven in the case where the underlying Gaussian assumptions do not hold.



(a) Histogram of the posterior distribution obtained overlaid with the curve of the analytic result.

(b) Evolution of β_j over the steps.

Figure 8.8. Results obtained for an observation with a value of 8. The step-size is determined dynamically using the method predicting the overlap of normal distributions.

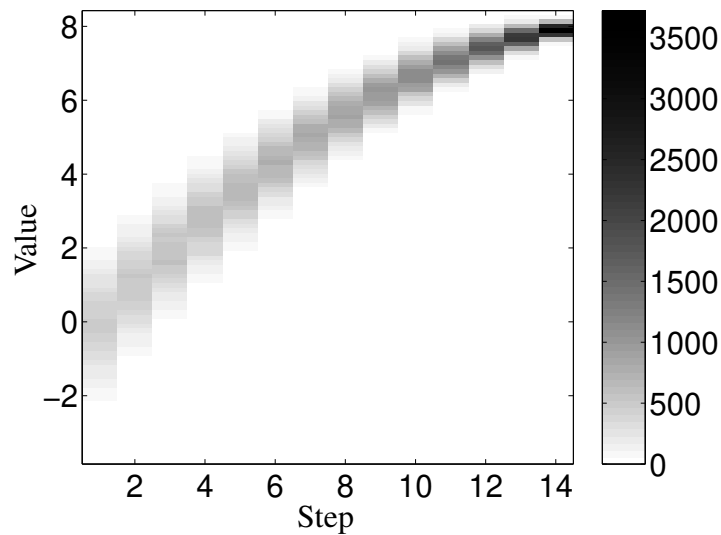


Figure 8.9. A Series of histogram plots showing the intermediate probability densities for 14 intermediate densities selected by the overlap of normal distributions. Darker grey represents higher probability density.

8.4 Communications Channels

The results of the Normal distributions in the previous section provide a good way of demonstrating the annealing method under particular conditions, but clearly they do not represent the general results we might expect to find in practical applications. In this section, the annealed particle filter is tested on realistic applications. Returning to the communications channel model, the behaviour on time-varying channels is investigated.

8.4.1 Generation

The advanced models of fading channels were described briefly in section 2.3.4.1. For the purpose of simulation, Jakes' model is used, with a random start on the time base. The coefficients may then be weighted separately, but only the case of equal weights is used here. Only real coefficients are generated: this allows the results to be displayed and interpreted in an easier manner, although extension to complex models is straightforward. Three taps are generated independently, according to the model. These taps are convolved with the bit sequence, giving an FIR channel as in previous chapters, except that the convolution function is now time-varying. As before, the observation noise is Gaussian.

8.4.2 Analysis

The analysis model used is that of the FIR filter channel with the filter coefficients undergoing a random walk, described by the following equation:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \mathcal{N}(0, \sigma_w^2 \mathbf{I}) \quad (8.11)$$

The variance is assumed fixed and possibly known. It is interesting to see the behaviour of the model when this is given a wrong value (in particular if the value is too small), as this allows us to observe the behaviour and performance when the model is a poor fit to the data. This can often occur in practice, particularly when a simple model such as this one is chosen to simplify implementation and maximise execution speed.

For the implementation of the particle filter without intermediate densities, the symbols were chosen as described in section 7.1.1, i.e. in the same manner as used in the majority of the previous chapter. This chooses the symbol taking the latest observation into account.

8.4.2.1 Intermediate Bridging Densities

For simplicity of implementation, the importance function using no information from the latest observation (i.e. all allowable states are equally probably), was used. The new value for the channel is chosen in the same manner as in the conventional particle filter, i.e. a random walk from the value at the previous time-step.

At each of the intermediate densities, four MCMC moves were performed using Metropolis-Hastings steps: a random walk with variance $\sigma_{mh}^2 = 0.5^2$ for each of the channel coefficients, and a move proposing changing the current symbol to any other, uniformly. This formulation is simplistic, but it allows the particle filter to operate very fast — several times faster than with time-invariant coefficients.

Dynamic Step Size The variance of the log weights was found to be an ineffective method for determining step size on the communications channel model. There should be a correlation (ideally a very strong one) between the variance of the log weights and the effective sample size (ESS) obtained. This is true for the normal distribution case, but not for the communications channel — there was no value of V in equation 8.5 which ensured all obtained values of ESS would be sufficiently large. The overlap of normal distributions method was mostly effective, however. As an additional safeguard, if the ESS resulting from the particular value of $\delta\beta_j$ was still too low, $\delta\beta_j$ was halved successively until the ESS was above the desired limit.

8.4.3 Challenges Faced with Fading Channels

The difficulties in equalisation of fading channels are very different from a time-invariant channel. By their very nature, fading channels produce a wide range of signal-to-noise ratios within a very short time scale. It is almost inevitable that there will be a point where the signal is swamped by the noise, and tracking will be lost. Therefore, the issue of recovery now becomes an issue. It should not be

the case that the particle filter is totally lost in these situations, but the estimates of the channel will be very vague around zero, with a BER of close to 0.5. In order to maintain a reasonable BER performance in the presence of such characteristics, error correction coding would need to be used in a practical system. Somewhat counter-intuitively, faster fading channels would be better as these produce very short error bursts, although the use of interleaving can split the bursts so that standard error correcting codes could be used.

The overall effect has an impact on methods for measurement. The bit error rate is less dependent different levels of signal to noise, and much more dependent on whether a fade occurs, and how fast one may recover from it. The focus of the results will be on the ability of the particle filter to recover and its robustness to inaccurate models rather than its absolute performance.

Phase ambiguity. Since the phase of the channel is lost during a fade, we have an identifiability problem. The use of differential encoding and decoding on the output of the particle filter is no longer ideal as the particle filter can capture more than one phase simultaneously. This is a particular problem when using intermediate densities with the overlap of Gaussians method for determining step size, since the variance will be overestimated in a multi-modal distribution, leading to more intermediate steps than is actually necessary.

To overcome this ambiguity, we can perform the differential encoding directly within the particle filter. The internal states stored are the decoded sequence, and thus unambiguous. These states need to be re-encoded for use when calculating the likelihood.

8.4.4 Model Mismatches on Fast Fading Channels

The characteristics of fast fading is that channel fades tend only to last a short time, and recovery from this position is much easier. Attention is therefore focussed on model mismatches, in particular the behaviour when we expect the random walk to have a higher or lower variance than that which would give good results.

The channel is generated using Jakes' model with 9 cisoids, with a Doppler frequency of 100 Hz and sampling frequency of 20 kHz . We shall only look at the real part of the channel, for simplicity of display, although modification

to include imaginary parts is straightforward. The particle filter is usually run with 200 particles. When intermediate densities are used, this is reduced to 100 particles to attempt to compensate for the extra steps required. Random Gaussian noise is added to the data, with variance $\sigma_v^2 = 0.1^2$.

The first example will be of a correctly operating particle filter with no intermediate densities with $\sigma_w = 0.3$. It is modified to make use of the differential encoding as described above. A plot of the parameter tracks is shown in figure 8.10. The top part of the graph shows the estimated and actual values of the channel parameters using the filtering density and no lag: the blue, green and red regions are the areas in which the middle 90% of the resampled particles lie for the first, second and third taps respectively. Within these regions is a solid magenta line, the mean of the distribution, and a dashed black line which shows the actual value. The smaller plot below indicates the location of symbol errors — a red bar denotes an error.

A moderate fade occurs at around $t = 400$, with a smaller one at around $t = 740$. In this example we do not get any major problems due to fading on all coefficients simultaneously, just the occasional error. This is because the analysis model represents the true behaviour fairly well.

The results shown in figure 8.11 are where an over-dispersed prior is used, $\sigma_w = 0.8$. We would expect all areas of high likelihood to lie within the propagated prior sample, therefore we would still expect tracking to occur. Indeed this is the case (with a phase change half-way through the data) although we can see that there are much broader bands between 5% and 95% percentiles. As a result of the more diffuse posterior, an error burst is present at the fade around $t = 400$. In this case we see recovery at a different phase (i.e. the opposite sign for this BPSK example) — in fact there are phase changes at around $t = 100, 250, 400, 730$.

The use of intermediate densities would not be expected to give us much benefit in this case: the problem is not that the transition from prior to posterior does not take place correctly, rather that the vague prior makes the posterior less tightly defined, resulting in reduced performance. This was borne out in simulation (not shown).

A completely different effect is observed if σ_w is set too low. We are effectively observing a dataset with many of its points being outliers. An example is shown in figure 8.12 with $\sigma_w = 0.1$. The first thing to notice is that the number of

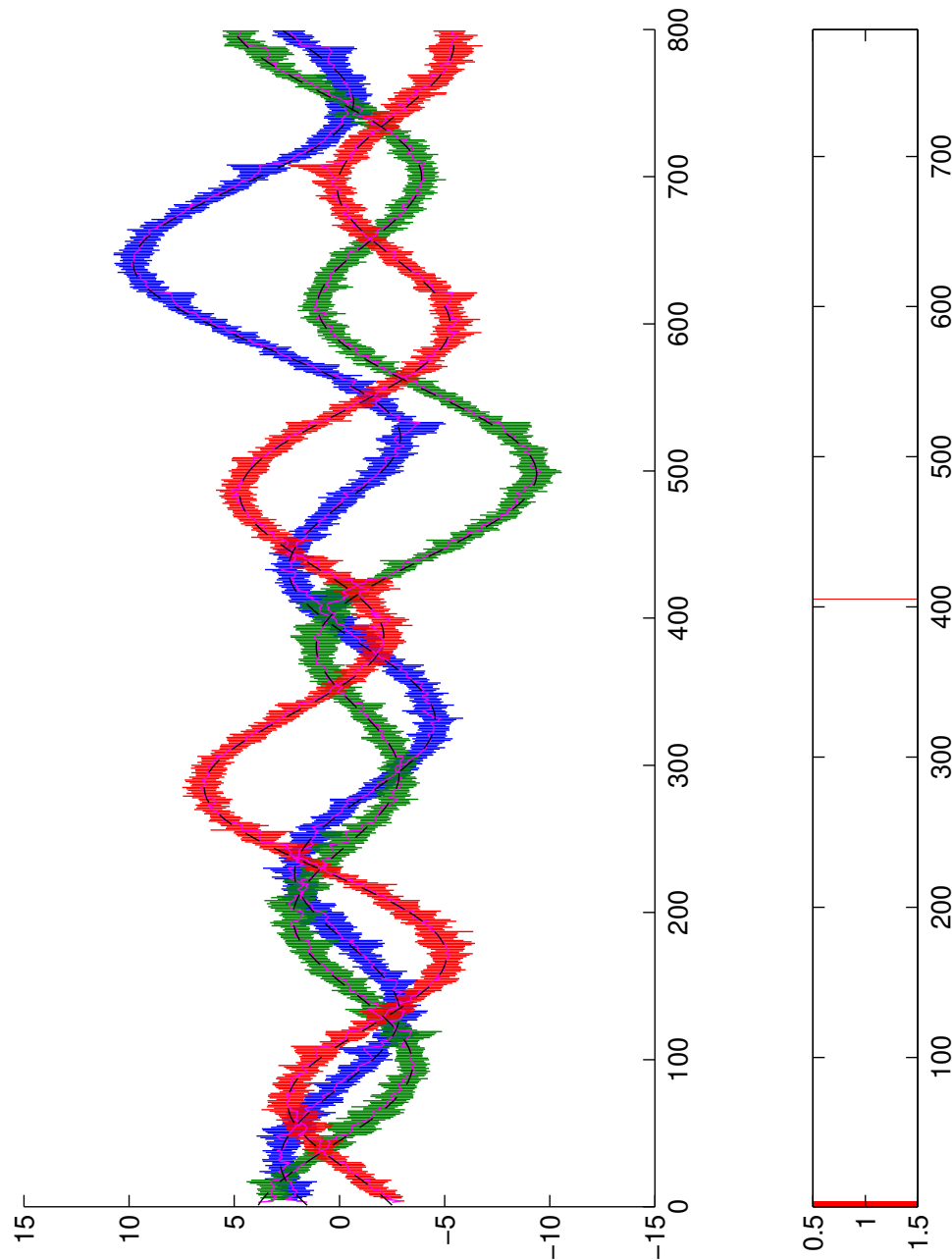


Figure 8.10. A demonstration of the particle filter working with $\sigma_w = 0.3$. In the top graph, the blue, green and red regions indicate the region between 5 and 95% percentiles for the 1st, 2nd and 3rd channel taps respectively. The magenta line in the middle of this region is the estimated mean from the filtering density and the dashed black line is the actual value. The plot underneath has a red bar whenever there is a symbol error, obtained from the fixed-lag filtering density.

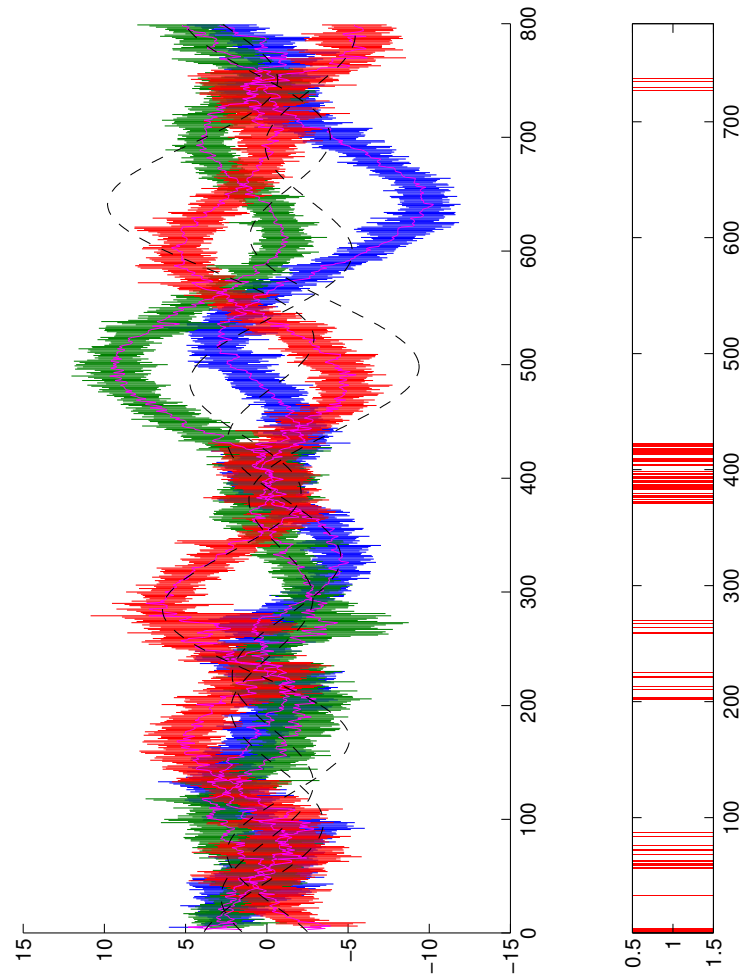


Figure 8.11. The results with an over-diffuse prior, $\sigma_w = 0.8$. In the top graph, the blue, green and red regions indicate the region between 5 and 95% percentiles for the 1st, 2nd and 3rd channel taps respectively. The magenta line in the middle of this region is the estimated mean from the filtering density and the dashed black line is the actual value. The plot underneath has a red bar whenever there is a symbol error, obtained from the fixed-lag filtering density.

errors is very large, far too many to be corrected by any error correction capability that may have been applied. The second is that the particle filter cannot keep up with the true tracks whenever the gradient exceeds a certain limit: this is not too surprising since this is directly dependent on the variance of the propagated prior. Perhaps it is more surprising that the particle filter locks on to the signal at all!

This situation is exactly the case in which intermediate densities would be useful. This is indeed the case, as shown in figure 8.13, where the same prior is used and annealing takes place between each step. The step size is determined dynamically using the overlap of normals method, with the overlap set to $k = 0.8$. Typically the transition from prior to posterior took 1–3 steps, with 2 being the modal number of steps (hence the reason why it is run with half the number of particles). Occasionally the number of steps was much higher, around 10.

In practice, the correct solution is to find the correct model or estimate σ_w online. However, this does demonstrate the ability of the annealed particle filter and the robustness that such a technique adds.

8.4.5 Initialisation on Slowly Fading Channels

One application that seems ideal for using intermediate bridging densities is that of initialisation: rather than using an MCMC method, or a known starting point, the initial cloud of particles is generated from the overall prior, which may be fairly vague. We would typically expect the first few points to convey a lot of information, therefore the processing might benefit greatly by using intermediate densities. Convergence to the correct values may be easier to see on a slowly fading channel. Jakes' model was used again for simulation, the parameters used in this case is the same Doppler frequency of 100 Hz, with a sampling frequency of 100 kHz.

Initialisation properties were investigated by looking at the ability of the particle filter to “lock” onto the correct channel. The particle filter was run for the first 100 observations only. It was deemed to have locked if there was a reasonable period at the end of the run in which there were only one or two errors. Usually lock occurred within the first 20 iterations, if the particle filter was to lock at all.

The intermediate densities were determined dynamically as before, using the method of overlap of normal distributions. The number of particles was increased

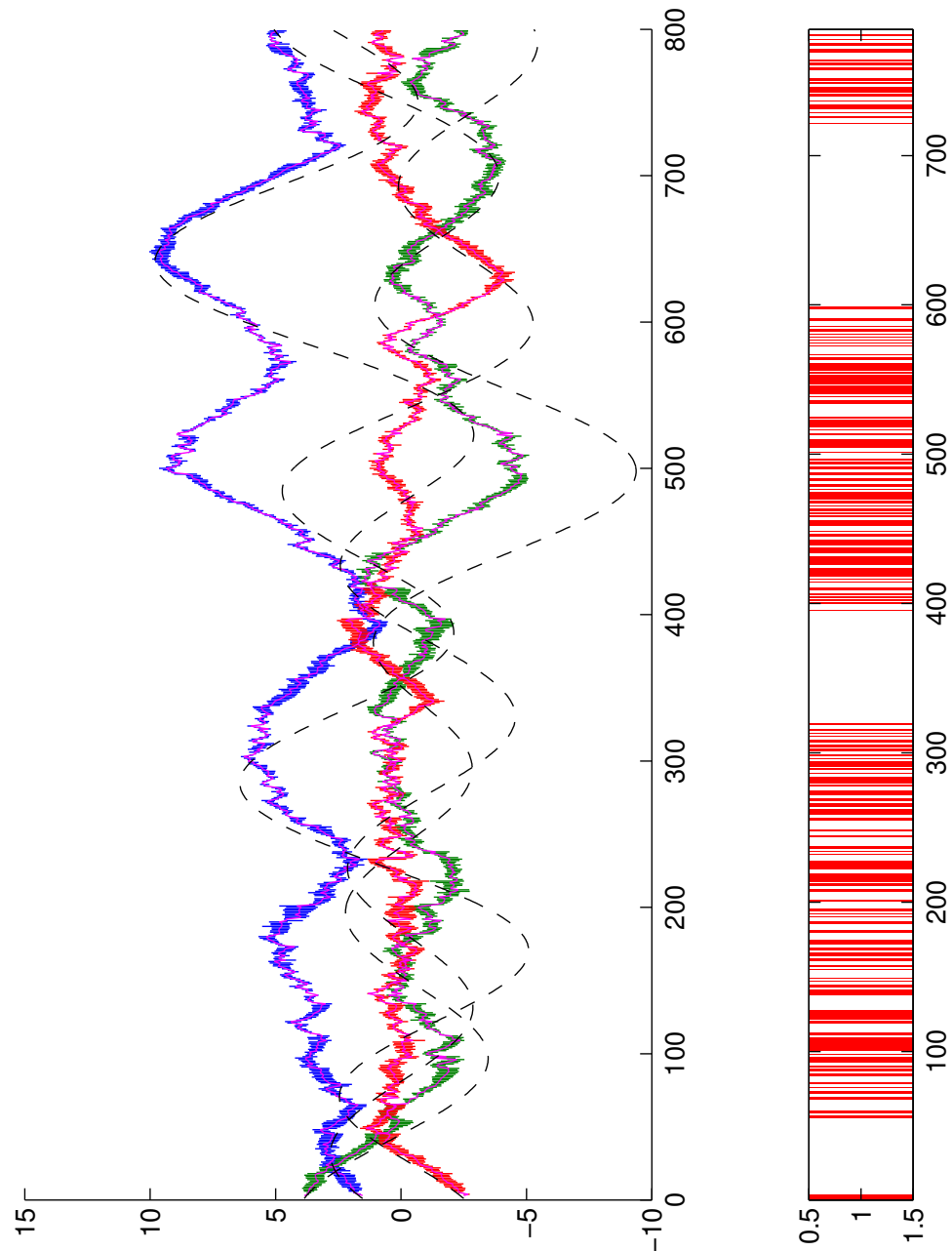


Figure 8.12. The results with an under-diffuse prior, $\sigma_w = 0.1$. In the top graph, the blue, green and red regions indicate the region between 5 and 95% percentiles for the 1st, 2nd and 3rd channel taps respectively. The magenta line in the middle of this region is the estimated mean from the filtering density and the dashed black line is the actual value. The plot underneath has a red bar whenever there is a symbol error, obtained from the fixed-lag filtering density.

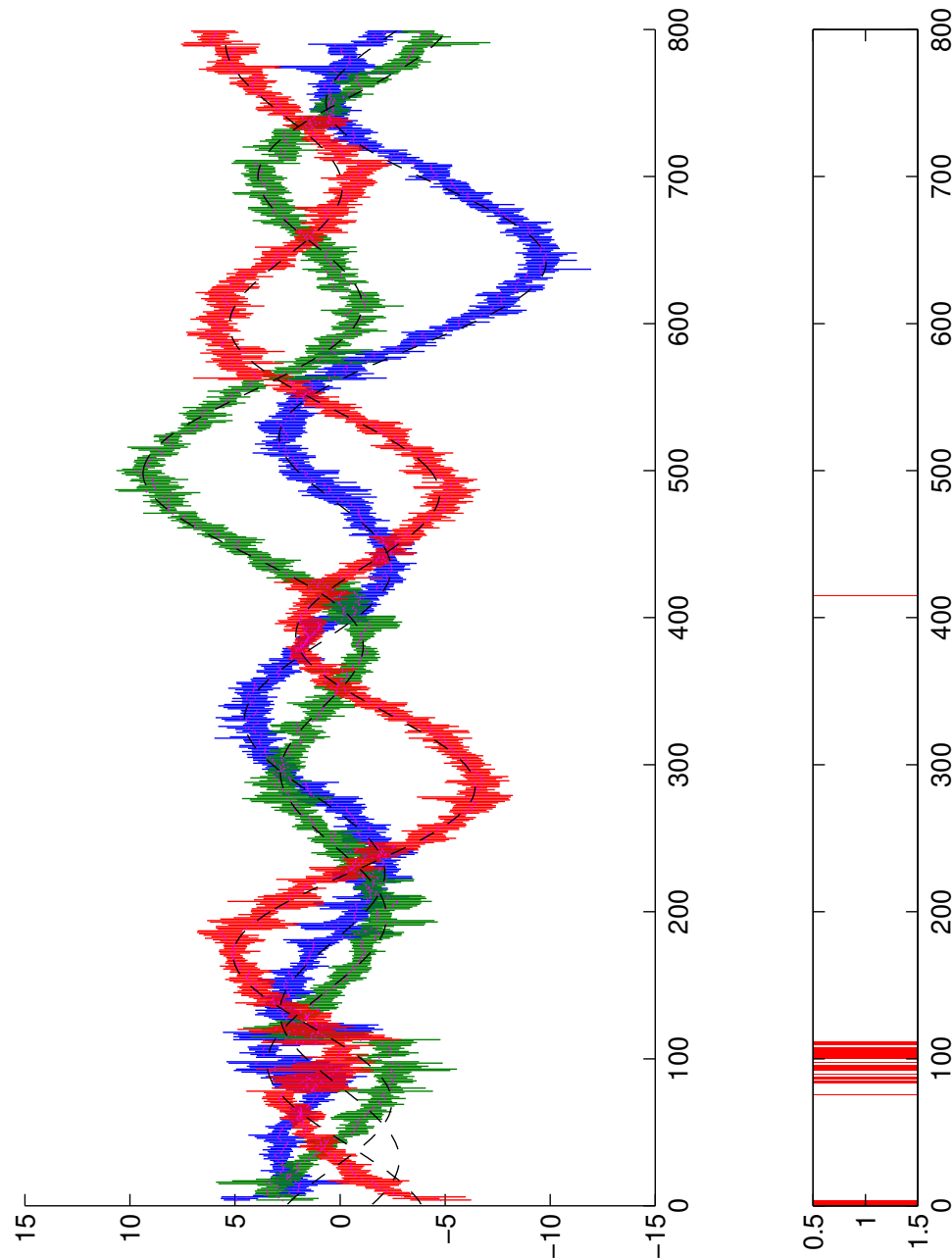


Figure 8.13. The results when adding the intermediate densities with $\sigma_w = 0.1$. In the top graph, the blue, green and red regions indicate the region between 5 and 95% percentiles for the 1st, 2nd and 3rd channel taps respectively. The magenta line in the middle of this region is the estimated mean from the filtering density and the dashed black line is the actual value. The plot underneath has a red bar whenever there is a symbol error, obtained from the fixed-lag filtering density.

to 500 for the plain particle filter and 250 for the annealed particle filter to help reduce the effects of very small cloud size. These represent approximately similar complexity, the annealed particle filter again choosing a mean of around two intermediate steps. For comparison, the annealed particle filter was run separately with 500 particles. The analysis random walk model was set with $\sigma_w = 0.1$. This gives little over- or under-dispersion of the prior at each time-step. Values for each of the elements of h for the initial particles were generated from $\mathcal{N}(0, \sigma_p^2)$. A value of $\sigma_p = 1$ is too small to be a good prior, and $\sigma_p = 10$ too large, but safe. The compromise of $\sigma_p = 3$ is a reasonable reflection of the data. All of the previous results on fast fading are generated using the same model, albeit with a different sampling frequency. Therefore any of the figures 8.10–8.13 may be viewed to extract a idea of the range of values the channel might take.

The experiment was repeated with 500 different initialisations to Jakes' model to obtain the lock rate, the percentage of times lock was achieved in the total number of runs, and the mean time taken to lock, if lock was achieved. These are given in table 8.5.

Algorithm	Number of Particles	σ_p	Lock Rate	Mean Time to lock
Particle filter	500	1	48.5%	7.9
Annealed particle filter	250	1	73.8%	9.6
Annealed particle filter	500	1	76.6%	9.4
Particle filter	500	3	47.6%	8.0
Annealed particle filter	250	3	78.0%	7.5
Annealed particle filter	500	3	78.6%	6.6
Particle filter	500	10	22.4%	12.2
Annealed particle filter	250	10	69.4%	9.8
Annealed particle filter	500	10	75.4%	8.2

Table 8.5. Lock rates and mean time to lock (if lock is achieved) for annealed and plain particle filters on a slowly fading channel.

Examples of the initialisation are shown in figure 8.14, where no intermediate densities are used, and figure 8.15, using intermediate densities. In this case, both particle filters have locked. It is clear from the graphs that one of the big differences is that when using intermediate densities, we retain a fairly vague estimate until sufficient data is obtained — without bridging densities, degeneracy

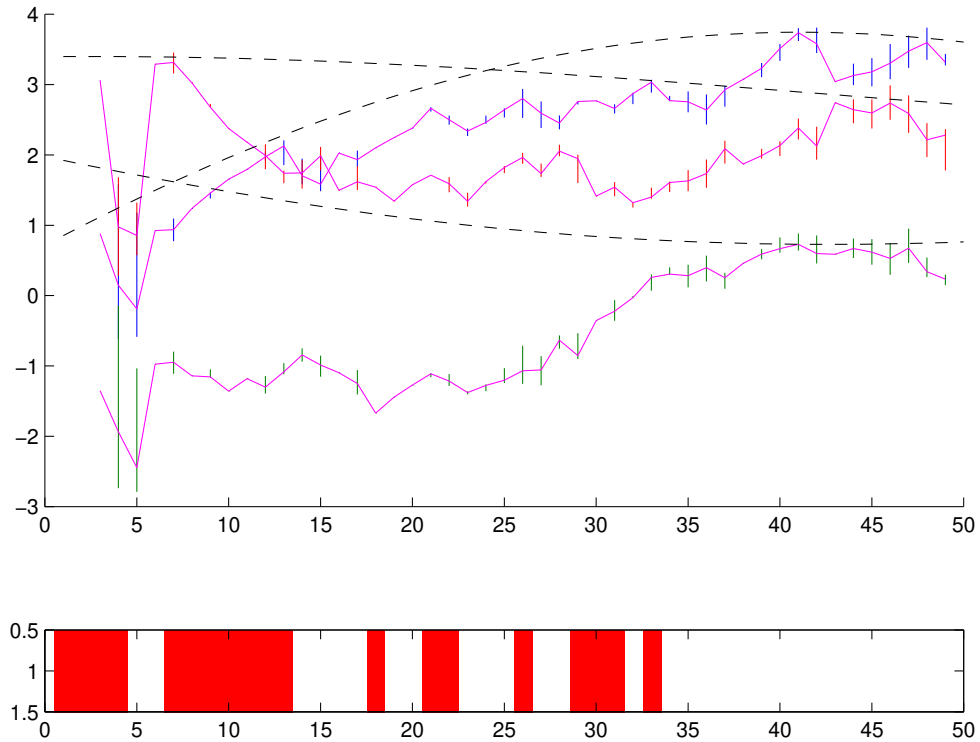


Figure 8.14. *Initialisation to a fading channel with no intermediate densities, $\sigma_p = 1$.*

occurs and this useful property is lost. This occurs with all values of σ_p tried.

There is a clear improvement in the number of instances where lock is achieved when intermediate densities are used. However, some anomalies were noted in the results. If the same dataset was processed several times, the achievement, or not, of lock was not consistent. This suggests that it is not the data alone that accounts for the cases when lock is not achieved.

For the cases of $\sigma_p = 3$ and $\sigma_p = 10$, the use of intermediate densities brings an improvement in mean time taken to lock, another useful effect. A possible explanation exists for the observation of the plain particle filter locking faster when $\sigma_p = 1$. Due to the tight prior, lock would only be expected to occur when the values of the channel lie within the prior. The achievement of lock on values further out by the annealed particle filter may take considerably longer to achieve, resulting in a larger mean time to lock.

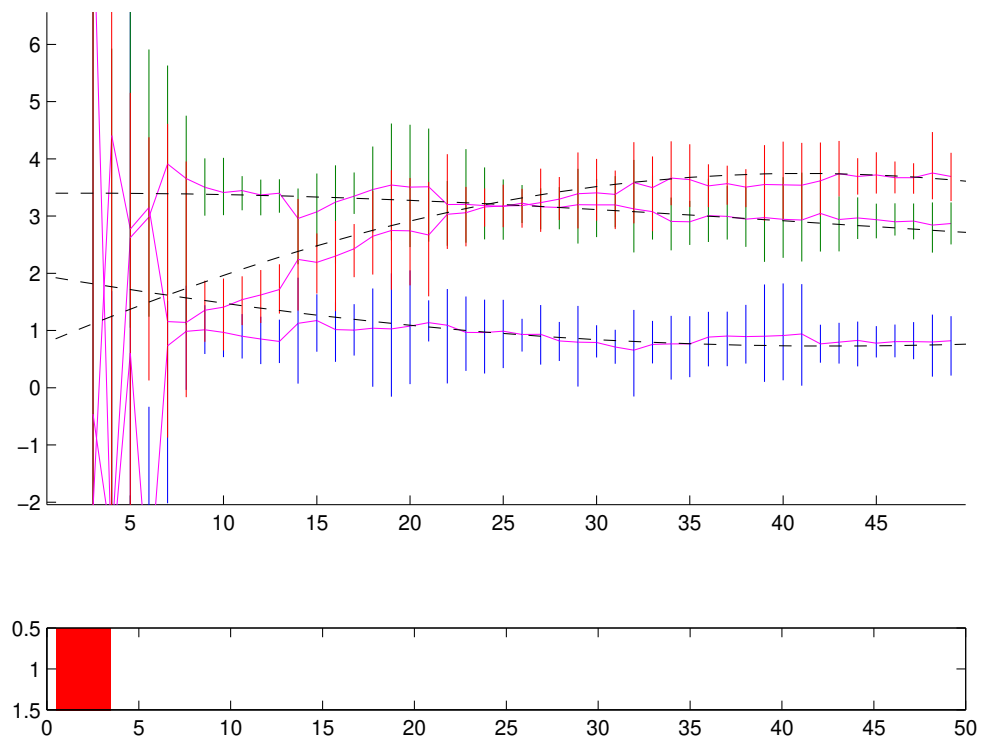


Figure 8.15. *Initialisation to a fading channel using annealing, $\sigma_p = 10$.*

8.4.5.1 Recovery

When the fading rate is decreased, the fades last longer, which can make it harder to regain track as the symbol states have a random initialisation due to a prolonged error burst. One would expect this ability to initialise to be helpful when recovering from channel fades. However, it is often harder when attempting to recover from a fade, since successive observations do not carry a great deal of information (the channel filter coefficients are still small and have similar values to each other), so intermediate densities do not necessarily add a great deal of benefit. Indeed little benefit was observed in simulations.

8.4.6 Some Issues and Suggestions for Improvement

Delay Ambiguity. The problem of delay ambiguity returns whenever blind initialisation is required. No measures were taken to overcome this in the examples

shown. Possible solutions include:

- The introduction of known training or synchronisation symbols into the transmitted data sequence.
- The use of MCMC or similar moves to move between different delay ambiguities.

Model Choice. The use of a random walk channel in analysis ignores two important pieces of prior information that we have available:

1. The overall statistics of the channel are Rayleigh distributed.
2. The movement of the channel from one point in time to the next is not well modelled by white noise: there is a strong low-pass element to this, arising from the Doppler spectrum.

The former issue may be addressed directly: instead of simply considering the propagation model to be dependent only on the position of the last point, we may multiply by an additional regularisation prior. For a complex process with Rayleigh-distributed amplitude, this would take the form of a normal distribution on both real and imaginary parts.

The latter issue can only be addressed by using a better model. Possible other choices for channel models include:

- Auto-regressive model for coefficients
- Sum of cisoid model

However the introduction of a more complex model not only increases the computational cost,³ but more importantly it affects the identifiability of the parameters of such a model. This may well be an issue if a symbol stream and several, possibly time-varying, parameters are to be estimated from a single observation.

³This may easily be overcome by waiting a couple of years — if the exponential growth in speed of integrated circuit technology continues.

8.5 *Conclusions*

This chapter has been mainly concerned with the use of Markov chain Monte Carlo moves within the framework of the particle filter. Such steps allow the particles to be moved to better describe the posterior distribution within the sequential framework that the particle filter provides.

The novel method of using a form of simulated annealing between distributions at successive time steps allows the particle filter to remain fully operational when it would otherwise have degenerated to a single particle, which may not even be from the true posterior distribution. Annealing allows the information contained within the likelihood of the observation to be introduced gradually to the prior and it provides a natural and easy way to split a single move in to smaller steps via intermediate distributions.

It has also been shown that we may rejuvenate (resample) at each intermediate density to ensure degeneracy does not occur. Practical experience suggests that this resampling is of utmost importance: it can overcome degeneracy once the final density is reached without an excessive number of intermediate steps. As a corollary, it appears that if degeneracy would not occur without resampling, then there are too many intermediate densities, or even that the annealing is not required at all.

For online implementations, at least in software, a dynamic choice of step-size i.e. number and location of intermediate distributions is very useful. A method based upon the overlap of distributions, assuming them to have the shape of a Gaussian, was derived and shown to work on practical problems, although it may not be appropriate for use on highly multi-modal distributions. Of the other methods proposed, the use of numerical optimisation for step-size determination has too strong a computational overhead for most applications, and the variance of the log weights appears not to be generally applicable. The whole area of dynamic determination of step-size would be of great interest if this technique of generating intermediate densities by simulated annealing were to become widely used.

Examples of the performance of this method were given for blind equalisation of the fading communications channel. With an accurate albeit simple model, in-

intermediate distributions were not required, and this leads to a method which could operate much faster than the other algorithms described in this thesis (including those on time-invariant channels), although this is mostly due to the simplicity of the model used, allowing for very fast simulation. The additional use of annealing methods improves robustness, particularly when behaviour not expected by the analysis model is encountered.

Conclusions and Further Work

9

9.1 Summary

This thesis is divided into three main sections:

- Chapters 2 and 3 form an overview of the problem which has been tackled and the previous work in this subject area. This started with an overview of time-series modelling, and in particular the auto-regressive and moving average models (digital filter models) which are commonly used for time-series analysis. The discussion moved on to the area of particular interest within that scope of this thesis — that of equalisation of a digital communications channel. The most popular model for this, using a finite impulse response filter, forms the basis for the processing which is subsequently applied.

The third chapter discusses the previous successful attempts at tackling the more general problem of deconvolution, including the so-called *blind* deconvolution, where the convolution function is not known. The main focus is on applications in the literature pertaining to equalisation of the commu-

nications channel.

- Chapters 4 and 5 are concerned with the theory and applications of Markov chain Monte Carlo methods, respectively. Firstly a grounding in Bayesian statistics is described, as this represents the underlying theory from which we may build to obtain high-performance algorithms. Unfortunately many practical problems cannot be solved by analytic means, resulting in a movement to the use of numerical methods. The exponential rise in computer processing power over the last couple of decades mean that these methods are now real possibilities for practical application. MCMC methods are just one of the many numerical methods available having desirable properties in terms of convergence to a global optimum and flexibility in choice of the underlying model.

The application of these methods to a particular problem requires some care to obtain the best performance, in terms of reliable convergence to the required global optimum and the number of iterations required to reach this region, which has a direct effect on the speed of operation or computational complexity of the algorithm.

- The remaining chapters move to the exciting and relatively new field of *particle filters* which have a considerable advantage over many other numerical methods in that they may be operated sequentially. Since this field is still developing rapidly, the focus is on more general techniques that may be applied to a wealth of problems, rather than the tailored application to the problem in hand. The description is split into a theory chapter (chapter 6), and two chapters on the application.

Chapter 7 discusses the use of two new techniques. It is well known that fixed-lag methods may be used to great effect on Markovian models where later observations can provide information about states in the recent past. Methods of performing fixed-lag simulation for incorporation into particle filter are described. The use of data windowing on fixed parameter systems allows regeneration of the parameters at each time-step without having excessive demands on storage requirements.

Chapter 8 describes the use of MCMC moves within the particle filter

framework to improve the performance. The particular innovation described is the use of intermediate bridging between one time step and the next to allow a smooth transition which would not otherwise be possible. This provides a degree of robustness against inexact models and outlying observations. This application discussed is on a time-varying digital communications system.

9.2 Conclusions

9.2.1 Bayesian Statistics

The use of Bayesian statistics is not a universal panacea for all problems: the strength lies in the ability to incorporate any information we have available about the system. If we can build good system models, it provides a framework in which to use those models for inference of any information in which we are interested. The other main advantage of Bayesian statistics lies in the ability to perform marginalisation — this allows us to ignore the details of the model or models used and provide a solution to the most direct question that the problem addresses, such as “*What is the most likely transmitted data sequence given these observations?*”

In many case where a model is representing real knowledge about a system, a solution in closed form is not possible. Numerical methods are now used and this thesis only considers two classes of the available methods: Markov chain Monte Carlo techniques and the use of particle filters.

In many real-life applications the exact solution, or accurate extraction of the posterior probability density, which is often the focus in the statistics literature, is not the ultimate aim. Frequently mostly correct results, most of the time, are good enough as sources errors will exist elsewhere in the system. For this reason it is important to consider the use of algorithms with a small number of iterations or particles, as appropriate, as these are more likely to be used for on-line processing.

9.2.2 Markov Chain Monte Carlo Moves for the Digital Communications System

The use of Markov chain Monte Carlo methods provide a fairly robust solution to obtaining estimates of quantities in complex models within a Bayesian statistical framework. They offer several advantages:

- Flexibility in choice of underlying model.
- Flexibility in implementation.
- Good performance due to underlying statistical framework.

They also suffer from a number of drawbacks:

- High computational complexity.
- It is sometimes difficult to assess whether convergence has been obtained, possibly resulting in reduced performance.
- Poor implementation can lead to slow convergence and/or poor mixing, which may result in a solution being found in a local maximum — which may be sub-optimal or completely wrong.
- It is not sequential — new data necessitates restarting the algorithm and running until convergence has been achieved again.

For use in the general problem of blind deconvolution, two major improvements have been proposed.

9.2.2.1 *Joint Draw of the Data Signal*

The data signal often consists of components that are not independent. The simpler random draws of each element independently results in poor mixing and poor convergence of the Markov chain. The use of a joint draw over all the data, described in section 5.2.1 overcomes this problem.

The joint draw can impose a significant extra computational burden due to the more complicated calculations required. This is more than compensated for by the reduction in the number of iterations required (from $O(10,000)$ to $O(10)$).

It is important to note that a joint draw of the data sequence in the manner described is *always* possible in a digital communications system because the data sent is discrete, so a discrete state-space system is formed. The integration required then reduces to a sum. However, if there is a very large number of possible symbols this may be too complex to perform quickly enough at each time step. In problems where a continuous state-space system is formed, this method can only be used when the integration is possible analytically, for example if the transition probabilities are Gaussian.

9.2.2.2 Summary

The techniques described in this thesis provide a practical solution to blind equalisation using MCMC methods. Due to good mixing properties and fast convergence properties of the Markov chain resulting from these techniques, only a very small number of iterations are required (most of the results presented use only 50). This relatively low complexity for an MCMC algorithm allows possible use on real applications with large datasets, although the computational requirements remain fairly high compared to traditional techniques.

9.2.2.3 Reversible Jump Moves for Delay Ambiguity and Determination of Model Order

The problem of delay ambiguity is well-known in blind equalisation and occurs in an analogous form in many other deconvolution applications. The MCMC algorithm can easily get stuck in a local maximum corresponding to a different delay ambiguity. This may be overcome by proposing moves directly between likely local maxima as described in section 5.3.2.1. These moves are implemented by a method which uses the technique of *reversible jumps* which ensures that the stationary distribution of the Markov chain is unaffected. The use of this method was shown to be highly effective, particularly in cases with poor initialisation or when the convolution function or channel coefficients have low power at the edges.

The reversible jump framework can also be used in its more customary role of model order selection of the convolution function. It was shown that this does not have a significant effect on performance when training data is present, but there

is frequently insufficient information for model order selection in blind mode. However the amount of training data required is much lower than for systems in current use (e.g. GSM). These results bear out the theory [31] that “semi-blind” techniques, using a very small number of training symbols, are sufficient for many applications and provide one such method for implementing this.

9.2.3 Particle Filters: Improvements and Application

Particle filters allow sequential processing of data in a Bayesian framework. A number of modifications to the particle filter framework were proposed in this thesis in order to obtain better results on the communications system. These improvements are fairly general and can be used in a variety of applications in far-ranging fields and go some way in overcoming the limitations of particle filters.

9.2.3.1 Fixed-lag Simulation

Three methods were proposed to achieve simulation of the states with a small number of further observations taken into account in order to improve estimates made. Of these three only two are worth seriously considering for serious use:

- **Filtering Density with Decision Step, x7.3.1.** This involves generating particles representing the posterior density in the usual manner. Instead of making estimates of the states immediately after updating the density with the corresponding observation, the estimates are delayed until the additional observations have been made. The decision step is delayed by a fixed amount rather than taking place instantly or at the end of the data. This has the advantages that almost no additional processing power is required (although it does require a small amount of additional memory) over that required for filtering, it can be applied to any system and generally produces very good results.
- **Smoothing Density, x7.3.3.** The use of a random draw from the smoothing density and an approximation to allow calculation of the weights produced good results in every example tested. The use of the conditional density as described however limits the applicability to instances where simulation from this density is possible and adds a significant computational burden.

The results suggest that this method is more robust than the previous one.

9.2.3.2 *Data Windowing*

There are many practical models which contain time-invariant parameters, a situation which the particle filter is not automatically equipped to deal with, although possible solutions exist [66, 105]. The use of a window containing the most recent past observations and states to allow simulation of the parameters during run-time was investigated. This was found in simulation to give excellent results compared to the more accurate simulation using all the past data back to the start of the run. Indeed a window size of a similar size to the particle cloud produced almost indistinguishable results.

9.2.3.3 *Intermediate Bridging Densities*

In occasional cases there is a poor overlap between the prior and the likelihood, which results in degeneracy in the particle filter. This happens because all the weight is placed on one or two points if the algorithm is allowed to proceed in the normal manner. This situation can arise from an outlier, a highly informative observation or a poor underlying model, such as that the prior does not encompass some of the more extreme but still reasonably likely observations. Since the current density is only stored as a set of points drawn from that density it is possible to get no effective overlap at all. The point (usually only one would be selected in this case) is now not even from the prior and the whole premise for propagation of the posterior density through time, so there is no guarantee of correct operation ever again.¹

The use of intermediate bridging densities by using simulated annealing between time steps has been introduced. These overcome the problems described as long as each intermediate step does not move too far. Heuristic measures to determine whether the use of intermediate densities is required, and how many and where to place them if they are, are also described. One of these, which calculates the step size required for a reasonable overlap if the prior and intermediate

¹Some researchers report “convergence” of the particle filter onto the posterior density, although this has not been observed on the communications channel model. This would therefore suggest that this behaviour is application specific.

distributions are Gaussian, appears to provide good performance, even when the distributions are not Gaussian, such as that encountered on the communications channel.

These methods may also be used for initialisation of the particle filter.

9.2.3.4 Communications Channels

The particle filter was applied to the equalisation problem. It was found that very good performance was obtained using a reasonable lag on the time-invariant channel. The parameters were simulated at each stage using either sufficient statistics or data windowing — there was little difference between the performance of either of these methods.

The application to time-varying fading channels was also considered. It was found that even with a simple random walk model on the channel parameters, reasonable tracking performance was obtained on coefficients generated by Jakes' model. The use of intermediate bridging densities by annealing provides a method both for initialisation and also if the underlying model is underdispersed relative to reality.

9.3 *Suggestions for Further Work*

9.3.1 Bayesian Statistics

If the field of Bayesian statistics continues to grow at the rate it has been doing, and the use of numerical methods becomes widespread, it is clear that researchers cannot be experts both in the field where challenging problems lie and in the development of techniques to provide a solution.

For the researcher with a problem to solve, it is vital to build the best possible model, whether through physical considerations, experimental observations or by reviewing the literature. Once a model is obtained, the problem needs to be posed in a manner that the power of the Bayesian framework can be exploited to its full potential.

The development and refinement of methods for obtaining solutions continues to grow at a high rate, however there is a great need for easier accessibility to

these techniques. The best way of serving this need would be to develop software to assist in the use of this technique. This may take the form of the relatively high-level interface such as BUGS (Bayesian inference Using Gibbs Sampling)² [168, 161], medium-level functions such as a MATLAB toolbox or low-level libraries in languages such as C/C++/Java.

9.3.2 MCMC Methods for Blind Deconvolution

It is hoped that this thesis has given a fairly complete exploration of blind equalisation of time-invariant channels, and in particular the techniques required to achieve this. The models used are fairly general and sometimes simplistic. There remains a volume of application specific work that could be carried out. Examples include:

- **Time-varying channels.** The modifications required to use the Gibbs sampler on a time-varying channel were described briefly, although no performance measurements have been made.
- **Higher order modulation schemes.** The results presented have been restricted to BPSK, which is seldom used in practice. Extensions to QPSK and upwards or multi-level QAM are described in the text, but not implemented.
- **Other Noise Distributions.** The noise has always been assumed to be Gaussian. This may be largely correct for the land–satellite channel [37], but not for land–land links. This should be relatively easy to add in the MCMC framework, especially if Metropolis-Hastings steps are used.
- **Correlated noise.** There may be a correlation between successive samples of noise, or as is frequently the case due to the design of demodulators, correlations between signal and noise.
- **Impulsive noise.** The noise may not be at a constant level, indeed it is possible to have abrupt changes in the level of noise due to interference.

²It should be noted that BUGS is ill-suited to time-series models as it does not incorporate the joint sampling techniques required to ensure fast convergence, or the ability to process very large amounts of data.

This results in impulsive noise. One method for approaching this problem that may be applied with little modification is described in [64].

- **Symbol Timing and Fractional Spacing.** Throughout, accurate symbol timing and sampling has been assumed to take place prior to the equaliser. This would not be the case in a practical system, and a fractionally spaced equaliser may be preferred.
- **Concatenated coding.** The output of a Bayesian equaliser is the symbol probabilities, which should be ideal to feed into the decoder of an error correction code, but it is not known how the distribution of errors, etc. will affect the end to end performance of a system with error correction coding.

9.3.3 Particle Filters for Blind Deconvolution

The use of particle filters is a new and quickly evolving field. There is much scope for the development of both the development of the techniques and their application. Much of the suggested further work above applies equally to this section, although a brief look at fading channels was undertaken. It is this area in particular that there is a wealth of further work that could be undertaken.

One of the most important areas is that of modelling. The random walk model used has a great benefit in the speed of operation (and relative simplicity of implementation), but performs poorly in channel fades and recovery from channel fades. More sophisticated models might allow at least reasonable tracking or prediction of the channel through a fade, even if the data symbols are lost. This would allow quick recovery from the fade, with a resulting increase in system performance. Possible alternative models include:

- Auto-regressive (AR) models.
- Sum of cisoids.

There is still much work to be done on formulation of a robust, generalised particle filter. Avenues of research, particularly for practical applications, include the effect of number of particles of robustness as well as degradation in performance, and the possibility of a dynamic scheme to determine the number of particles required at each step.

The particle filter using intermediate bridging densities still has a number of unknowns. The trade-off between number of intermediate steps against number of particles is perhaps one of the most important of these, as is a reliable method for determining step-size dynamically.

9.4 Final Words

It is in the very nature of human beings to communicate, indeed it is often said that communication is the key to the domination of the planet by our race. It is hoped that some of the work presented in this thesis might just make that communication a little more accessible.

Fixed Channel Models



In this appendix the impulse response, frequency response and Z-plane plot of the location of zeros are given for the static channels used in the simulations.

Note that channels A.1, A.2 and A.4 are all minimum phase. Channel A.3 is non-minimum phase.

Channel	Filter coefficients
A.1	.407, .815, .407
A.2	.766, .575, .287
A.3	-0.1833, 0.9162, 0.4812, -0.1987
A.4	0.9162, -0.1833, 0.4812, -0.1987

Table A.1. *Filter coefficients for the channels.*

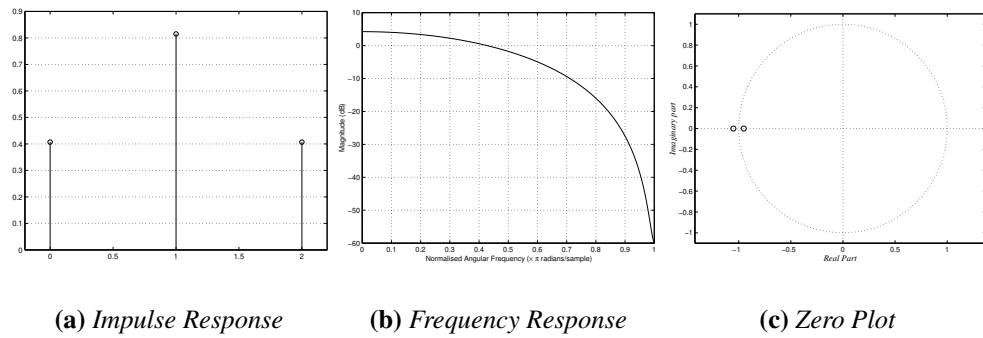


Figure A.1. Impulse response, frequency response and zero plot for channel A.1

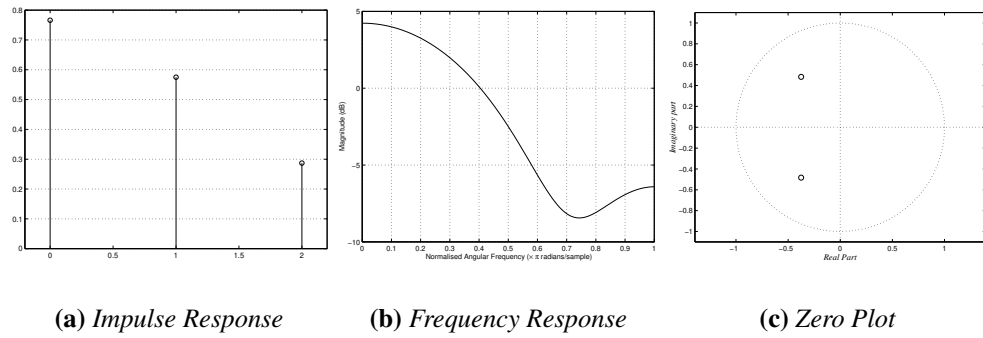


Figure A.2. Impulse response, frequency response and zero plot for channel A.2

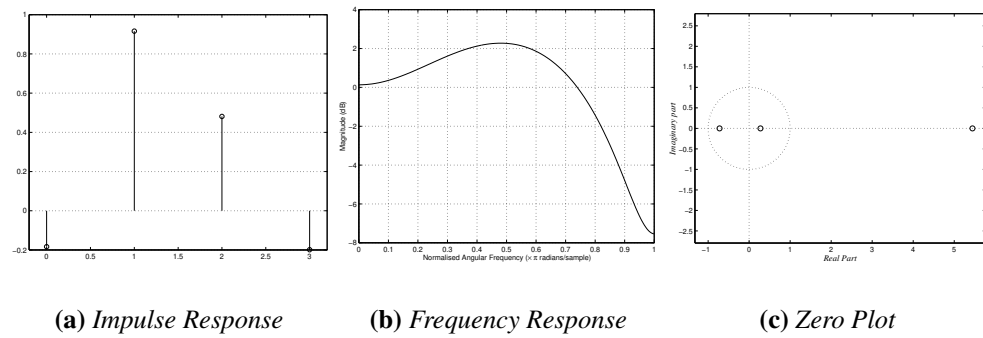


Figure A.3. Impulse response, frequency response and zero plot for channel A.3

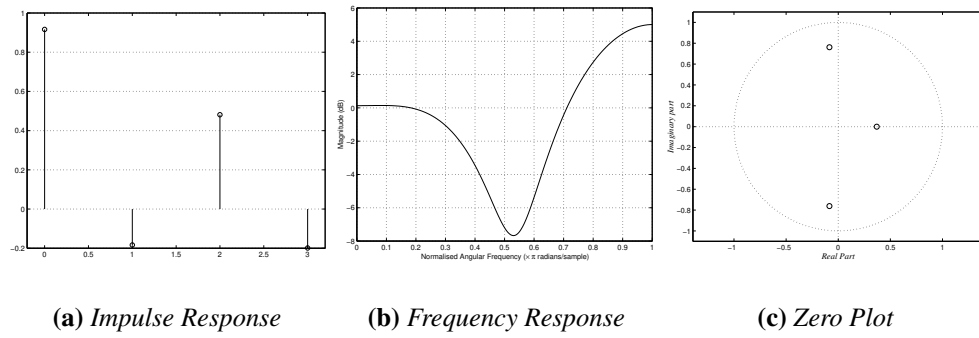


Figure A.4. Impulse response, frequency response and zero plot for channel A.4

Results for Manipulations of Gaussian Distributions

B

In many cases Gaussian distributions are chosen because they are analytically tractable or by appealing to Central Limit Theorem arguments. This appendix provides a few mathematical results for the manipulation of these multivariate distributions, used in the course of this work.

B.1 Multiplication

Multiplication usually arises since the posterior is the product of the prior and likelihood. If both prior and likelihood are Gaussian, the posterior is also Gaussian, hence a Gaussian prior for location is the conjugate prior for a Gaussian likelihood.

Let us consider the multiplication of two distributions, $\mathcal{N}(\mu_1, \Sigma_1)$ and

$\mathcal{N}(\mu_2, \Sigma_2)$:

$$\begin{aligned} \mathcal{N}(\mu_1, \Sigma_1) \mathcal{N}(\mu_2, \Sigma_2) &= \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma_1|} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_1)^T \Sigma_1^{-1} (\mathbf{x} - \mu_1) \right\} \\ &\quad \times \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma_2|} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_2)^T \Sigma_2^{-1} (\mathbf{x} - \mu_2) \right\} \quad (\text{B.1}) \\ &= \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma_0|} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_0)^T \Sigma_0^{-1} (\mathbf{x} - \mu_0) \right\} \end{aligned}$$

where $\Sigma_0^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1}$
and $\mu_0 = \Sigma_0 \left(\Sigma_1^{-1} \mu_1 + \Sigma_2^{-1} \mu_2 \right)$.

One can easily see that this follows for exponent terms by multiplying out and completing the square: the constant of proportionality then follows directly since the distribution must be normalised correctly.

B.2 Extensions for AR / MA models

In the case of AR or MA models the state vector, \mathbf{x} , is frequently multiplied by a parameter vector or parameter matrix. One may extend the above result, by consideration of the process of completing the square, to give simple steps for deriving the resulting posterior distribution:

1. Multiply out the exponent terms.
2. Collect terms in $\mathbf{x}^T \cdot \mathbf{x}$ (e.g. $\mathbf{x}^T \Sigma_1^{-1} \mathbf{x}$). The sum of these terms will give the required Σ_0^{-1} .
3. Collect terms in \mathbf{x}^T . The mean of the resulting distribution, μ_0 , is given by the sum of these terms multiplied by Σ_0 .

If we are only interested in simulation from the resulting posterior, the constant of proportionality is not important. If required, this is most easily obtained by determining the constant required for a normalised distribution, rather than calculating the normalising constant algebraically.

Bibliography

- [1] K. ABEND and B. D. FRITCHMAN. “Statistical Detection for Communication Channels with Intersymbol Interference”. In *Proceedings of the IEEE*, volume 58, pages 779–785. 1970.
- [2] B. D. O. ANDERSON and J. B. MOORE. *Optimal Filtering*. Prentice-Hall, 1979.
- [3] G. A. ARREDONDO, W. H. CHRIS and E. H. WALKER. “A Multipath Fading Simulator for Mobile Radio”. *IEEE Transactions on Communications*, 22(4): 241–4, 1973.
- [4] ARRL. *Spread Spectrum Sourcebook*. The American Radio Relay League, USA, 1991.
- [5] J. R. BALL. “A Real-Time Fading Simulator for Mobile Radio”. *The Radio and Electronic Engineer*, 52(10): 475–478, 1982.
- [6] T. BAYES. “An essay towards solving a Problem in the Doctrine of Chances”. *Philosophical Transactions of the Royal Society of London*, 53: 370–418, 1763. Published posthumously by his friend, Richard Price.
- [7] S. BELLINI. “Bussgang Techniques for Blind Deconvolution”. In [80], chapter 2.
- [8] G. BENELI, A. FIORAVANTI, A. GARZELLI and P. MATTEINI. “Some Digital Receivers for the GSM Pan-European Cellular Communication System”. In *IEE Proceedings on Communications*, volume 141, pages 168–176. 1994.

- [9] A. BENVENISTE and M. GOURSAT. “Blind equalizers”. *IEEE Transactions on Communications*, 32(8): 871–883, 1984.
- [10] A. BENVENISTE, M. GOURSAT and G. RUGET. “Robust Identification of a Non-minimum phase system: Blind adjustment of a linear equalizer in data communication”. *IEEE Transactions on Automatic Control*, AC-25(3): 385–399, 1980.
- [11] J. M. BERNARDO, J. O. BERGER, A. P. DAWID and A. F. M. SMITH (eds.). *Bayesian Statistics 4*. Clarendon Press, London, 1992.
- [12] J. M. BERNARDO, J. O. BERGER, A. P. DAWID and A. F. M. SMITH (eds.). *Bayesian Statistics 6*. Oxford University Press, 1999.
- [13] J. M. BERNARDO, M. H. DEGROOT, D. V. LINDLEY and A. F. M. SMITH (eds.). *Bayesian Statistics 3*. Oxford University Press, 1988.
- [14] J. M. BERNARDO and A. F. M. SMITH. *Bayesian Theory*. Wiley, 1994.
- [15] C. BERZUINI, N. G. BEST, W. R. GILKS and C. LARIZZA. “Dynamic conditional independence models and Markov chain Monte Carlo methods”. *J. Amer. Statist. Assoc.*, 92(440): 1403–1412, 1997.
- [16] D. J. BROOKS and J. A. CHAMBERS. “Multi-branch DFE for Rayleigh fading multipath channels”. *Electronics Letters*, 33(18): 1537–1538, 1997.
- [17] D. J. BROOKS, S. LAMBOTHARAN and J. A. CHAMBERS. “Optimised MSE and Delay for Blind Equalizers”. In *IEE Colloquium on Adaptive Signal Processing for Mobile Communication Systems*, pages 10/1–10/5. IEE, Savoy Place, London, 1997. Reference No: 1997/383.
- [18] J. CARPENTER, P. CLIFFORD and P. FEARNHEAD. “Building Robust Simulation-based Filters for Evolving Data Sets”, 1999. (unpublished) Available from <http://www.stats.ox.ac.uk/>.
- [19] J. CARPENTER, P. CLIFFORD and P. FEARNHEAD. “An Improved Particle Filter for Non-linear Problems”. *IEE Proceedings - F: Radar, Sonar and Navigation*, 146: 2–7, 1999.

- [20] C. K. CARTER and R. KOHN. “Gibbs sampling for state space models”. *Biometrika*, 81(3): 541–553, 1994.
- [21] M.-H. CHEN. “Importance-Weighted marginal Bayesian posterior density estimation”. *Journal of the American Statistical Association*, 89(427): 818–824, 1994.
- [22] R. CHEN and T. LI. “Blind Restoration of Linearly Degraded Discrete Signals by Gibbs Sampling”. *IEEE Transactions on Signal Processing*, 43(10): 2410–2423, 1995.
- [23] S. CHEN, S. MCLAUGHLIN, P. M. GRANT and B. MULGREW. “Multi-stage blind clustering equalizer”. *IEEE Transactions on Communications*, 43(2/3/4): 701–705, 1995.
- [24] T. CLAPP and S. GODSILL. “Fixed-lag Blind Equalization and Sequence Estimation in Digital Communications Systems using Sequential Importance Sampling”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 5. IEEE Signal Processing Society, Phoenix, Arizona, 1999.
- [25] T. CLAPP and S. J. GODSILL. “Bayesian Blind Deconvolution for Mobile Communications”. In *IEE Colloquium on Adaptive Signal Processing for Mobile Communication Systems*, pages 9/1–9/6. IEE, Savoy Place, London, 1997. Reference No: 1997/383.
- [26] T. C. CLAPP and S. J. GODSILL. “Fixed-Lag Smoothing using Sequential Importance Sampling”. In J. BERNARDO, J. BERGER, A. DAWID and A. SMITH (eds.), *Bayesian Statistics 6*. Oxford University Press, 1999.
- [27] R. H. CLARKE. “A Statistical Theory of Mobile-Radio Reception”. *Bell Systems Technical Journal*, 47: 957–1000, 1968.
- [28] R. T. COX. “Probability, frequency and Reasonable Expectation”. *American Journal of Physics*, 14(1): 1–13, 1946.
- [29] D. CRISAN, P. DEL MORAL and T. LYONS. “Discrete Filtering Using Branching and Interacting Particle Systems”. *Markov Processes and Related Fields*, 5(3): 293–319, 1999.

- [30] D. CRISAN and M. GRUNWALD. “Large Deviation Comparison of Branching Algorithms versus Resampling Algorithms: Application to Discrete Time Stochastic Filtering”. Research Report 1999-9, Cambridge University Statistical Laboratory, 1999.
- [31] E. DE CARVALHO and D. T. M. SLOCK. “Cramer-Rao bounds for semi-blind, blind and training sequence based channel estimation”. In *Proceedings of Signal Processing advances in Wireless Communications*, pages 129–132. IEEE Signal Processing workshop, Paris, 1997.
- [32] E. DEL RE, G. CASTELLINI, L. PIERLUCCI and F. CONTI. “A within-burst Adaptive MLSE receiver for modile TDMA cellular systems”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume IV. IEEE Signal Processing Society, 1992.
- [33] Z. DING, C. R. JOHNSON JR. and R. A. KENNEDY. “Global Convergence Issues with Linear Blind Adaptive Equalization”. In [80], chapter 3.
- [34] A. DOUCET. “On Sequential Simulation-Based Methods for Bayesian Filtering”. Technical Report CUED/F-INFENG/TR. 310, Cambridge University Engineering Department, 1998.
- [35] A. DOUCET and P. DUVAUT. “Fully Bayesian Analysis of Hidden Markov models”. In G. RAMPONI, G. L. SICURANZA, S. CARRATO and S. MARSÌ (eds.), *Proceedings of EUSIPCO*, volume 1, pages 244–247. Edizioni LINT, Trieste, Italy, 1996.
- [36] A. DOUCET, N. DE FREITAS and N. GORDON (eds.). *Sequential Monte Carlo Methods in Practice*. Chapman and Hall, 2001. (To appear).
- [37] B. G. EVANS (ed.). *Satellite Communication Systems*. Institution of Electrical Engineers, 1999.
- [38] P. FEARNHEAD. *Sequential Monte Carlo methods in filter theory*. Ph.D. thesis, University of Oxford, 1998.
- [39] G. D. FORNEY. “Maximum likelihood sequence estimation of digital sequences in the presence of intersymbol interference”. *IEEE Transactions on Information Theory*, 18(3): 363–378, 1972.

- [40] G. D. FORNEY. “The Viterbi algorithm”. In *Proceedings of the IEEE*, volume 61, pages 268–278, 1973.
- [41] G. D. FORNEY JR. *Concatenated Codes*. MIT Press, 1966.
- [42] G. J. FOSCHINI. “Equalizing without altering or detecting data”. *AT&T Technical Journal*, 64(8): 1885–1911, 1985.
- [43] S. FRÜHWIRTH-SCHNATTER. “Applied state space modelling of non-Gaussian time series using integration based Kalman filtering”. *Statistics and Computing*, 4: 259–269, 1994.
- [44] W. A. GARDNER. “A New Method of Channel Identification”. *IEEE Transactions on Communications*, 39: 813–817, 1991.
- [45] A. E. GELFAND and A. F. M. SMITH. “Sampling-Based Approaches to Calculating Marginal Densities”. *Journal of the American Statistical Association*, 85(410): 398–409, 1990.
- [46] A. E. GELFAND, A. F. M. SMITH and T.-M. LEE. “Bayesian Analysis of Constrained Parameter and Truncated Data Problems using Gibbs Sampling”. *Journal of the American Statistical Association*, 87(418): 523–532, 1992.
- [47] A. GELMAN. “Inference and Monitoring Convergence”. In [57], chapter 8.
- [48] A. GELMAN, J. B. CARLIN, H. S. STERN and D. B. RUBIN. *Bayesian Data Analysis*. Chapman & Hall, 2–6 Boundary Row, London, UK, 1995.
- [49] S. GEMAN and D. GEMAN. “Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6: 721–741, 1984.
- [50] I. GERONTIDES and R. L. SMITH. “Monte Carlo generation of order statistics from a general distribution”. *Applied Statistics*, 31(3): 236–243, 1982.
- [51] J. GEWEKE. “Bayesian Inference in Econometric Models Using Monte Carlo Integration”. *Econometrica*, 57(6): 1317–1339, 1989.

- [52] C. J. GEYER. “Practical Markov chain Monte Carlo”. *Statistical Science*, 7: 473–511, 1992.
- [53] C. J. GEYER. “Burn-In is Unnecessary”. Available from <http://www.stat.umn.edu/~charlie/mcmc/burn.html>, 1998.
- [54] G. B. GIANNAKIS. “Cumulants: A powerful tool in signal processing”. In *Proceedings of the IEEE*, volume 75, pages 1333–1334. IEEE, 1987.
- [55] J. D. GIBSON (ed.). *The Mobile Communications Handbook*. IEEE Press, 1996.
- [56] W. R. GILKS and C. BERZUINI. “Following a Moving Target - Monte Carlo Inference for Dynamic Bayesian Models”, 1998. (unpublished).
- [57] W. R. GILKS, S. RICHARDSON and D. J. SPIEGELHALTER (eds.). *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 2–6 Boundary Row, London, UK, 1996.
- [58] W. R. GILKS and G. O. ROBERTS. “Strategies for improving MCMC”. In [57], chapter 6.
- [59] R. D. GITLIN and S. B. WEINSTEIN. “Fractionally-Spaced Equalization: An Improved Digital Transversal Equalizer”. *Bell Systems Technical Journal*, 60: 275–296, 1981.
- [60] D. GODDARD. “Self-Recovering equalization and Carrier Tracking in two-dimensional data communication systems”. *IEEE Transactions on Communications*, 28(11): 1867–1875, 1980.
- [61] S. J. GODSILL. “On the relationship between MCMC model uncertainty methods”. *Journal of Computational and Graphical Statistics*, 2000. (In press).
- [62] S. J. GODSILL and T. C. CLAPP. “Improvement strategies for Monte Carlo particle filters”. In [36]. (To appear).
- [63] S. J. GODSILL and P. J. W. RAYNER. *Digital Audio Restoration*. Springer, 1998.

- [64] S. J. GODSILL and P. J. W. RAYNER. “Robust Reconstruction and Analysis of autoregressive signals in impulsive noise using the Gibbs Sampler”. *IEEE Transactions on Speech and Audio Processing*, (to appear).
- [65] N. GORDON, D. SALMOND and C. EWING. “Bayesian State Estimation for Tracking and Guidance using the Bootstrap Filter”. *Journal of Guidance Control and Dynamics*, 18(6): 1434–1443, 1995.
- [66] N. GORDON, D. SALMOND and A. F. M. SMITH. “Non-linear / Non-Gaussian Bayesian State Estimation”. *IEE Proceedings - F: Radar and Signal Processing*, 140(2): 107–113, 1993.
- [67] R. A. GOUBRAN, H. M. HAFEZ and A. V. H. SHEIKH. “Implementation of a Real-Time Mobile Channel Simulator using a DSP chip”. *IEEE Transactions on Instrumentation and Measurement*, 40(4): 709–714, 1991.
- [68] F. GOZZO and J. B. ANDERSON. “The impact of white Gaussian noise mismatch on linear and decision feedback equalizers”. *IEEE Transactions on Communications*, 43(2/3/4): 173–177, 1995.
- [69] P. J. GREEN. “Reversible jump Markov chain Monte Carlo computation and Bayesian model determination”. *Biometrika*, 82: 711–732, 1995.
- [70] P. J. GREEN and D. J. MURDOCH. “Exact Sampling for Bayesian Inference: Towards General Purpose Algorithms”. In [12].
- [71] P. E. GREENWOOD, I. W. MCKEAGUE and W. WEFELMEGER. “Information bounds for Gibbs Samplers”. *The Annals of Statistics*, pages 2128–2156, 1998.
- [72] GSM Recommendation 05.05. “Radio Transmission and Reception”. ETSI/PT 12, 1991.
- [73] M. I. GURELLI and C. L. NIKIAS. “EVAM: An Eigenvector-based algorithm for multichannel blind deconvolution of input coloured signals”. *IEEE Transactions on Signal Processing*, 43(1): 134–149, 1995.

- [74] F. GUSTAFSSON and B. WAHLBERG. “Blind equalization by direct examination of the input sequences”. *IEEE Transactions on Communications*, 43(7): 2213–2222, 1995.
- [75] J. HANDSCHIN. “Monte Carlo Techniques for Prediction and Filtering of Non-Linear Stochastic Processes”. In *Automatica*, volume 6, pages 555–563. 1970.
- [76] J. HANDSCHIN and D. MAYNE. “Monte Carlo Techniques to Estimate the Conditional Expectation in Multi-stage Non-Linear Filtering”. *International Journal of Control*, 9(5): 547–559, 1969.
- [77] W. K. HASTINGS. “Monte Carlo sampling methods using Markov chains and their applications”. *Biometrika*, 57(1): 97–109, 1970.
- [78] D. HATZINAKOS. “Blind Equalisation based on Prediction and polycepstra principles”. *IEEE Transactions on Communications*, 43(2/3/4): 178–181, 1995.
- [79] D. HATZINAKOS and C. NIKIAS. “Blind equalization Based Higher-Order Statistics (H.O.S.)”. In [80], chapter 5.
- [80] S. HAYKIN (ed.). *Blind Deconvolution*. Prentice Hall, Englewood Cliffs, NJ, 1994.
- [81] J. A. HELLER and I. M. JACOBS. “Viterbi Decoding for Satellite and Space Communication”. In *IEEE Transactions on Communication Technology*, volume COM-19, pages 835–848. 1971.
- [82] T. HIGUCHI. “Monte Carlo filter using the genetic algorithm operators”. *Journal of Statistical Computation and Simulation*, 59(1): 1–23, 1997.
- [83] T. HIGUCHI. “Self-organizing Time Series Model”. In [36]. (To appear).
- [84] W. HOLUBOWICZ and F. MORALES-MORENO. “Convolutional coding of binary CPM schemes with no increase in receiver complexity”. *IEEE Transactions on Communications*, 43(2/3/4): 1221–1224, 1995.

- [85] J.-J. HSUE and A. E. YAGLE. “Blind Deconvolution of Symmetric Non-causal Impulse Responses Using Two-sided Linear Prediction”. *IEEE Transactions on Signal Processing*, 42(6): 1509–1518, 1994.
- [86] B. T. IRONS and K. D. DONOHUE. “Probability of Erasure in non-Rayleigh Fading Channels — a Simulation study”. *IEEE Transactions on Communications*, 43(2/3/4): 1246–1247, 1995.
- [87] W. C. JAKES (ed.). *Microwave Mobile Communications*. Wiley, 1974.
- [88] H. JEFFREYS. *Theory of Probability*. Oxford University Press, 3rd edition, 1961.
- [89] P. DE JONG and N. SHEPHARD. “The simulation smoother for time series models”. *Biometrika*, 82(2): 339–350, 1995.
- [90] R. KALMAN and R. BUCY. “New Results in Linear Filtering and Prediction Theory.” *Journal of Basic Engineering, Transactions ASME Series D*, 83: 95–108, 1961.
- [91] S. KAY. *Fundamentals of Statistical signal processing: Estimation theory*. Prentice Hall Signal Processing series. Prentice Hall, Englewood cliffs, New Jersey, 1993.
- [92] S. KIRKPATRICK, C. D. GELATT and M. P. VECCHI. “Optimisation by Simulated Annealing”. *Science*, 220: 671–680, 1983.
- [93] G. KITAGAWA. “Non-Gaussian state-space modelling of non-stationary time series (with discussion)”. *Journal of the American Statistical Association*, 82(400): 1032–63, 1987.
- [94] G. KITAGAWA and W. GERSH. *Smoothness priors analysis of time series*, chapter 6.5, pages 78–85. Springer, 1996.
- [95] A. KONG, J. LIU and W. H. WONG. “Sequential Imputations and Bayesian missing data problems”. *Journal of the American Statistical Association*, 89(425): 278–288, 1994.

- [96] N. F. LAW and D. T. NGUYEN. “Multiple frame projection based blind deconvolution”. *Electronics Letters*, 31(20): 1734–1735, 1995.
- [97] G.-K. LEE, S. GELFAND and M. P. FITZ. “Bayesian Decision Feedback Techniques for Deconvolution”. *IEEE Journal on Selected Areas in Communications*, 13(1): 155–165, 1995.
- [98] T. LI. “Blind Identification and Deconvolution of Linear Systems Driven by Binary Random Sequences”. *IEEE Transactions on Information Theory*, 38(1): 26–38, 1992.
- [99] Y. LI, B. VUCETIC and Y. SATO. “Optimum Soft-Output Detection for Channels with Intersymbol Interference”. In *IEEE Transactions on Information Theory*, volume 41, pages 1937–46. 1995.
- [100] K. S. LIU and M. ROSENBLATT. “Deconvolution and estimation of transfer function phase coefficients for non-Gaussian linear processes”. *The Annals of Statistics*, 10(4): 1195–1208, 1982.
- [101] T. J. LIM and M. D. MACLEOD. “A Tutorial on Linear Adaptive Filters”. Technical Report CUED/F-INFENG/TR 227, Cambridge University Engineering Department, 1995.
- [102] D. V. LINDLEY. “A statistical paradox”. *Biometrika*, 44: 187–192, 1957.
- [103] J. LIU and R. CHEN. “Blind Deconvolution via Sequential Imputations”. *Journal of the American Statistical Association*, 90(430): 567–576, 1995.
- [104] J. LIU and M. WEST. “Sequential Analysis of Dynamic Models”. Poster at “The Sixth Valencia International Meeting on Bayesian Statistics”, 1998.
- [105] J. LIU and M. WEST. “Combined parameter and state estimation in simulation-based filtering”. In [36]. (To appear).
- [106] J. LIU, W. H. WONG and A. KONG. “Covariance structure of the Gibbs sampler with applications to the comparison of estimators and augmentation schemes”. *Biometrika*, 81(1): 27–40, 1994.

- [107] J. S. LIU. “Sequential Monte Carlo methods for dynamical systems”. In *Journal of the American Statistical Association*, volume 93, pages 1032–1044. 1998.
- [108] J. S. LIU and R. CHEN. “A note on Monte Carlo methods for dynamical systems”. *unpublished*, 1996.
- [109] J. S. LIU, R. CHEN and W. H. WONG. “Rejection Control and Sequential Importance Sampling”. *Journal of the American Statistical Association*, 93(443): 1022–1031, 1998.
- [110] Y. LOU. “Comparison of Adaptive blind equalizers”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 545–548. IEEE Signal Processing Society, San Fransisco, 1992.
- [111] Y. LOU. “Channel Estimation Standard and adaptive blind equalization”. *IEEE Transactions on Communications*, 43(2/3/4): 182–186, 1995.
- [112] R. W. LUCKY. “Automatic Equalization for Digital Communications”. *Bell Systems Technical Journal*, 44: 547–588, 1965.
- [113] S. N. MACEACHERN, M. CLYDE and J. S. LIU. “Sequential Importance Sampling for Nonparametric Bayes models: The Next Generation”, 1996. (unpublished).
- [114] G. MARSAGLIA. “The Mathematics of Random Number Generators”. In *Proceedings of Symposia in Applied Mathematics*, volume 46, pages 73–89. 1992.
- [115] G. MARSAGLIA and L. H. TSAY. “Matrices and the structure of random number sequences”. *Linear algebra and its applications*, 67: 147–156, 1985.
- [116] G. MARSAGLIA and A. ZAMAN. “A new class of random number generators”. *Annals Applied Probability*, 1(3): 462–480, 1991.

- [117] N. METROPOLIS, A. W. ROSENBLUTH, M. N. ROSENBLUTH, A. H. TELLER and E. TELLER. “Equation of state calculations by fast computing machines”. *Journal of Chemical Physics*, 21(6): 1087–1092, 1953.
- [118] S. P. MEYN and R. L. TWEEDIE. *Markov Chains and Stochastic Stability*. Springer-Verlag, London, 1993.
- [119] C. MOLER. “Cleve’s corner: Random thoughts”. In *MATLAB News and Notes*. 1995.
- [120] T. K. MOON. “The Expectation Maximisation Algorithm”. *IEEE Signal Processing Magazine*, pages 47–59, 1996.
- [121] R. M. NEAL. “Annealed Importance Sampling”. Technical Report 9805, Dept. of Statistics and Dept. of Computer Science, University of Toronto, 1998.
- [122] C. NILL and C. E. W. SUNDBERG. “List and Soft symbol output Viterbi Algorithms: extensions and comparisons”. *IEEE Transactions on Communications*, 43(2/3/4): 277–287, 1995.
- [123] J. J. K. Ó RUANAIDH and W. J. FITZGERALD. *Numerical Bayesian Methods applied to signal processing*. Springer, New York, 1996.
- [124] J. J. OLMOS, G. GELONCH, F. J. CASADEVALL and G. FEMENIES. “Design and Implementation of a Wide-Band Real-Time Mobile channel Emulator”. *IEEE Transactions on Vehicular Technology*, 48(3): 746–764, 1999.
- [125] A. V. OPPENHEIM and R. W. SCHAFER. *Discrete-Time Signal Processing*. Prentice-Hall International, 1989.
- [126] R. H. J. M. OTTEN and L. P. P. P. VAN GINNEKEN. “Floorplan design using simulated annealing”. In *IEEE International Conference on Computer Aided Design (ICCAD-84)*. 1984.
- [127] K. PAHLAVAN and A. H. LEVESQUE. *Wireless Information Networks*. Wiley, 1995.

- [128] S. PERREAU and P. DUHAMEL. “On-line blind equalization of FIR channels using a Gibbsian technique”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. 1997.
- [129] A. P. PETROPULU and C. L. NIKIAS. “Blind deconvolution of coloured signals based on high order cepstra and data fusion”. *IEE Proceedings - F: Radar and Signal Processing*, 140(6): 356–361, 1993.
- [130] G. PICCHI and C. PRATI. “Blind equalization and carrier recovery using a “Stop-and-Go” decision directed algorithm”. *IEEE Transactions on Communications*, 35(9): 877–887, 1987.
- [131] M. PITT and N. SHEPHARD. “Filtering via Simulation: Auxiliary Particle Filters”. *Journal of the American Statistical Association*, 94(446): 590–599, 1999.
- [132] M. PITT and N. SHEPHARD. “Filtering via Simulation: Auxiliary Particle Filters”. *Journal of the American Statistical Association*, 1999.
- [133] M. K. PITT and N. SHEPHARD. “Auxiliary variable based particle filters”. In [36]. (To appear).
- [134] B. PORAT and B. FRIEDLANDER. “Blind equalization of digital communications channels using high-order moments”. *IEEE Transactions on Signal Processing*, 39: 552–526, 1991.
- [135] W. H. PRESS, S. A. TEUKOLSKY, W. T. VETTERLING and B. P. FLANNERY. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1989.
- [136] J. G. PROAKIS. *Digital Communications*. Electrical engineering series. McGraw-Hill, 3rd edition, 1995.
- [137] J. G. PROPP and D. B. WILSON. “Exact sampling with coupled Markov chains and applications to statistical mechanics”. *Random Structures and Algorithms*, 9: 223–252, 1996.

- [138] S. U. H. QUERESHI and G. D. FORNEY, JR. “Performance and properties of a T/2 Equalizer”. In *National Telecommunications Conference Record*, pages 11.1.1–11.1.14. Los Angeles, California, 1977.
- [139] A. P. QUINN. *The Bayesian Paradigm in Signal Estimation*. Ph.D. thesis, Cambridge University Engineering Department, 1991.
- [140] R. RAHELI, A. POLYDOROS and C. K. TZOU. “Per-Survivor Processing: A General Approach to MLSE in Uncertain Environments”. *IEEE Transactions on Communications*, 43(2/3/4): 354–364, 1995.
- [141] D. G. READER and W. G. COWLEY. “Blind maximum likelihood sequence detection”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE Signal Processing Society, 1996.
- [142] D. J. READER. *Blind Maximum Likelihood Sequence Estimation*. Ph.D. thesis, University of South Australia, 1996.
- [143] C. P. ROBERT, A. DOUCET and S. J. GODSILL. “Maximization of Marginal Posterior Distributions using Markov Chain Monte Carlo Methods”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 1753–1756. IEEE Signal Processing Society, Phoenix, Arizona, 1999.
- [144] G. O. ROBERTS. “Markov chain concepts related to sampling algorithms”. In [57], chapter 3.
- [145] G. O. ROBERTS. “MCMC: Introductions to algorithms and theory”, 1998. Lecture series at the MCMC Summer School, Rebild, Denmark.
- [146] G. O. ROBERTS and S. K. SAHU. “Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler”. *Journal of the Royal Statistical Society*, 59(2): 291–217, 1997.
- [147] R. DA ROCHA LOPES, C. A. F. DA ROCHA and J. M. T. ROMANO. “Predictive Self-Learning Equalization for Mobile Radio Channels: and LMS x

- RLS Comparison”. In *Proceedings of Signal Processing advances in Wireless Communications*, pages 337–340. IEEE Signal Processing workshop, Paris, 1997.
- [148] O. ROSEC and J.-M. BOUCHER. “Bayesian Estimation of Non-Minimum Phase Wavelets applied to Marine Reflection Seismic Data”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 5. IEEE Signal Processing Society, Phoenix, Arizona, 1999.
- [149] A. M. ROSS. *Simulation*. Academic Press, 2nd edition, 1997.
- [150] S. M. ROSS. “Bayesians should not Resample a Prior Sample to Learn About the Posterior”. *The American Statistician*, 50(2): 116, 1996.
- [151] D. B. RUBIN. “Comments on ‘The calculation of posterior distributions by data augmentation’ — Tanner and Wong[165]”. *Journal of the American Statistical Association*, 82(398): 543–546, 1987.
- [152] D. B. RUBIN. “Using the SIR Algorithm to Simulate Posterior Distributions”. In [13], pages 395–402.
- [153] P. SALAMON, J. D. NULTON, J. PEDERSON, G. RUPPEINER and L. LIAO. “Simulated annealing with constant thermodynamic speed”. *Computer Physics Communications*, 49: 423–428, 1988.
- [154] Y. SATO. “A method of self-recovering equalization for multilevel Amplitude-Modulation systems”. *IEEE Transactions on Communications*, 23: 679–682, 1975.
- [155] N. SESHADRI. “Joint Data and Channel Estimation using Blind Trellis Search Techniques”. In [80], chapter 6.
- [156] O. SHALVI and E. WEINSTEIN. “New Criteria for Blind Deconvolution of Nonminimum Phase Systems (Shannels)”. *IEEE Transactions on Communications*, 36: 312–321, 1990.
- [157] O. SHALVI and E. WEINSTEIN. “Universal Methods for Blind Deconvolution”. In [80], chapter 4.

- [158] B. W. SILVERMAN. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
- [159] D. T. M. SLOCK. “Spatio-temporal training-sequence-based channel equalization and adaptive interference cancellation”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE Signal Processing Society, 1996.
- [160] H. W. SORENSON. “Recursive estimation for non-linear systems”. In J. C. SPALL (ed.), *Bayesian Analysis of Time Series and Dynamic Models*. Dekker, 1988.
- [161] D. J. SPIEGELHALTER, A. THOMAS and N. G. BEST. *WinBUGS Version 1.2 User Manual*. MRC Biostatistics Unit, 1999.
- [162] A. STARK. *Statistical model selection techniques for data analysis*. Ph.D. thesis, Cambridge University Engineering Department, 1995.
- [163] R. J. STERITI and M. A. FIDDY. “Blind deconvolution of images by the use of neural networks”. *Optics Letters*, 19(8): 575–577, 1994.
- [164] T. G. STOCKHAM JR., T. M. CANNON and R. B. INGEBRETSEN. “Blind Deconvolution through Digital Signal Processing”. In *Proceedings of the IEEE*, volume 63, pages 678–692. IEEE, 1975.
- [165] M. A. TANNER and W. H. WONG. “The calculation of posterior distributions by data augmentation, with discussion”. *Journal of the American Statistical Association*, 82(398): 528–550, 1987.
- [166] R. W. TENNENT. *Digital Modems for Mud Pulse Telemetry*. Ph.D. thesis, Cambridge University Engineering Department, 1997.
- [167] C. W. THERRIEN. *Discrete Random signals and statistical signal processing*. Prentice Hall Signal Processing series. Prentice Hall, Englewood cliffs, New Jersey, 1992.
- [168] A. THOMAS. “BUGS: a statistical modelling package”. *RTA/BCS Modular Languages Newsletter*, 2: 36–38, 1994.

- [169] L. TIERNEY. “Markov Chains for exploring Posterior Distributions (with discussion)”. *The Annals of Statistics*, 22(4): 1791–1762, 1994.
- [170] L. TIERNEY. “Introduction to general state-space Markov chain theory”. In [57], chapter 3.
- [171] L. TONG. “Blind Sequence Estimation”. *IEEE Transactions on Communications*, 43(12): 2986–2994, 1995.
- [172] L. TONG, G. XU and T. KAILATH. “Blind Identification and Equalization Based on Second-Order Statistics: A Time Domain Approach”. *IEEE Transactions on Information Theory*, 40(2): 340–349, 1994.
- [173] P. T. TROUGHTON and S. J. GODSILL. “A Reversible Jump Sampler for Autoregressive Time Series, Employing Full Conditionals to Achieve Efficient Model Space Moves”. Technical Report CUED/F-INFENG/TR. 304, Cambridge University Engineering Department, 1997.
- [174] P. T. TROUGHTON and S. J. GODSILL. “A Reversible Jump Sampler for Autoregressive Time Series”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume IV, pages 2257–2260. 1998.
- [175] M. K. TSATSANIS, G. GIANNAKIS and G. ZHOU. “Estimation and Equalization of Fading Channels with Random Coefficients”. *Signal Processing*, 53: 211–229, 1996.
- [176] J. K. TUGNAIT. “Identification of Linear Stochastic Systems via Second- and Fourth- Order cumulant matching”. *IEEE Transactions on Information Theory*, 33(3): 393–407, 1987.
- [177] J. K. TUGNAIT. “Comments on: Cumulants: A powerful tool in Signal Processing”. In *Proceedings of the IEEE*, volume 77, page 491. IEEE, 1989. In reply to [54].
- [178] J. K. TUGNAIT. “Approaches to FIR system identification with noisy data using higher order statistics”. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(7): 1307–1313, 1990.

- [179] J. K. TUGNAIT. “Blind Equalisation and estimation of Digital Communication FIR channels using cumulant matching”. *IEEE Transactions on Communications*, 43(2/3/4): 1240–45, 1995.
- [180] G. UNGERBOECK. “Fractional tap-spacing equalizer and consequences for clock recovery in data modems”. *IEEE Transactions on Communications*, 24: 856–864, 1976.
- [181] A. J. VITERBI. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. *IEEE Transactions on Information Theory*, 13(2), 1967.
- [182] K. WESOLOWSKI. “Adaptive Blind Equalizers with automatically controlled parameters”. *IEEE Transactions on Communications*, 43(2/3/4): 170–172, 1995.
- [183] M. WEST. “Kernel Density Estimation and Marginalization Consistency”. *Biometrika*, 78(2): 421–425, 1991.
- [184] M. WEST. “Modelling with Mixtures”. In [11].
- [185] M. WEST. “Approximating Posterior Distributions by Mixtures”. *Journal of the Royal Statistical Society*, 55(2): 409–422, 1993.
- [186] M. WEST. “Mixture models, Monte Carlo, Bayesian updating and dynamic models”. In J. H. NEWTON (ed.), *Computing Science and Statistics: Proceedings of the 24th Symposium on the Interface*, pages 325–333. Interface Foundation of North America, Fairfax Station, Virginia, 1993.
- [187] B. WIDROW. “Adaptive Filters, I: Fundamentals”. Technical Report 6764–6, Stanford Electronics Laboratory, Stanford University, 1966.
- [188] S. L. WOOD, M. G. LARIMORE and J. R. TREICHLER. “The impact of an adaptive equalizer’s update behaviour symbol error rate in a non-stationary environment”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume III, pages 1608–1611. IEEE Signal Processing Society, New York, 1988.

-
- [189] K. YAMAZAKI, R. A. KENNEDY and Z. DING. “Globally convergent blind equalization algorithms for complex data systems”. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume IV, pages 553–556. IEEE Signal Processing Society, San Francisco, 1992.
- [190] V. ZARITSKII, V. SVETNIK and L. SHIMELEVICH. “Monte Carlo Technique in Problems of Optimal Data Processing”. *Automation and Remote Control*, 12: 95–103, 1975.
- [191] S. L. ZEGER and M. R. KARIM. “Generalizes linear models with random effects: a Gibbs sampling approach”. *Journal of the American Statistical Association*, 86: 79–86, 1991.