Akademie věd České republiky
Ústav teorie informace a automatizace

Academy of Sciences of the Czech Republic
Institute of Information Theory and Automation

RESEARCH REPORT

Li He

Bayesian Modelling and Estimation of ARMAX Model and Its Finite Mixtures

No. 2080 June 2003

ÚTIA AV ČR, P. O. Box 18, 182 08 Prague, Czech Republic
Fax: (+42)(2)688 4903
E-mail: utia@utia.cas.cz

Bayesian Modelling and Estimation of ARMAX Model and Its Finite Mixtures

By
Li He

Supervisors
Ing. Miroslav Kárný, DrSc.(ÚTIA AV ČR)
Prof. Ing. Pavel Žampa , CSc. (FAV, ZČU)

A Thesis
Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
At West Bohemia University
Czech Republic

Contents

Abstract **V**

Acknowledgements **VI**

Symbols and Notations **VII**

1 Introduction **1**
1.1 State of the Art . . . 1
1.2 Aims and Ideas . . . 3
1.3 Thesis Layout . . . 4

2 Preliminaries **6**
2.1 Bayesian Paradigm . . . 6
2.1.1 Uncertainty and Probability . . . 6
2.1.2 Bayesian Modelling . . . 8
2.1.3 General Bayesian Estimation and Prediction . . . 10
2.2 Mixture Distributions . . . 12
2.2.1 Description of Static Mixtures . . . 12
2.2.2 Related Issues . . . 13
2.3 Direct Search Methods . . . 14
2.3.1 Nelder-Mead Simplex Method . . . 15
2.3.2 Multidirectional Search . . . 18

3 Stochastic Models of Dynamic Systems **22**
3.1 Dynamic Mixture Modelling . . . 22
3.2 ARX Models . . . 25
3.3 ARMAX Models . . . 26
3.4 ARX Mixtures . . . 28
3.5 MARMAX Models . . . 28
3.6 ARMMAX Models . . . 29

4 Bayesian Estimation and Prediction **30**
4.1 Common Tools . . . 30
4.1.1 Estimation with Forgetting . . . 30
4.1.2 Algorithm of Dyadic Reduction . . . 32
4.1.3 Exponential Family . . . 32
4.2 Bayesian Solution to ARX Models . . . 34
4.2.1 GiW Conjugate Prior Pdf . . . 34

4.2.2 $L'DL$ Decomposition of Extended Information Matrix 35
4.2.3 Estimation and prediction 37
4.3 Mixture Estimation and Numerical Approximation 39
4.3.1 Alternative Description of Mixtures 39
4.3.2 Dirichlet Conjugate Prior Pdf of Mixing Weights 39
4.3.3 Approximate QB Estimation 40
4.3.4 Iterative Construction of Prior Pdf with Flattening 42
4.3.5 Iterative and Batch QB Estimations of ARX Mixtures 43

5 Bayesian Solution to ARMAX Models with Known MA Part 49
5.1 Known MA Part up to Noise Covariances 50
5.1.1 LD Filter 50
5.1.2 Estimation and Prediction 51
5.2 Known MA Part up to C-parameters 52
5.2.1 Extended LD Filter 52
5.2.2 Estimation and Prediction 53
5.3 Some Properties of LD type Filters 55

6 Bayesain Solutions and Modelling Properties of ARMAX Mixtures 57
6.1 Relationships to ARMAX models 58
6.2 Estimation and Prediction of ARMAX Mixtures with Known C-parameters . . . 58
6.2.1 QB Estimation and Prediction of MARMAX Models 59
6.2.2 QB Estimation and Prediction of ARMMAX Models 62
6.3 Approximate Parallelism 66

7 Improved Bayesian Solutions to ARMAX Models 70
7.1 MMQ and MAQ Methods 70
7.1.1 Problem Formulation 70
7.1.2 Choices of MDS method and ARMAX Mixtures 71
7.1.3 Description of MMQ/MAQ Estimation 72
7.1.4 Implementation Aspects 76
7.2 PEB Method 79

8 Illustrative Examples 80
8.1 Preliminaries 80
8.1.1 Software Aspects 80
8.1.2 Measures of Performance 82
8.2 Basic Properties of MMQ and MAQ 83
8.2.1 Estimating Simulated ARMA 83
8.2.2 Estimating Simulated OE 85
8.3 Comparisons of MMQ, MAQ and PEB Methods 87
8.3.1 Simulated ARMAX Process 87
8.3.2 Estimated Model and Implementation 88
8.3.3 Result and Analysis 88
8.4 Transportation Problem 89
8.4.1 Data Description 90
8.4.2 Estimated Model and Implementation 90
8.4.3 Results and Analysis 91
8.5 Summary 91

9 Conclusions and Future Work **93**
9.1 Main Results 93
9.2 Future Work 95

A Mixtures at Factor Level **99**

B QB Estimation at Factor Level **101**

Abstract

Exploiting probabilistic finite mixtures, the dissertation examines Bayesian modelling and estimation in the presence of colored stochastic disturbances. Focused on ARMAX model and two types of its finite mixtures:

We discuss a feasible dynamic mixture modelling and reexamine classical LD pre-whitening filters that impose no constraint on the MA noise part of an ARMAX model so that the roots of the C-polynomial are allowed to be even at the stability boundary.

Two types ARMAX mixtures were then studied: One is so-called MARMAX model, a natural mixture generalization of ARMAX model. The other one is ARMMAX model, a novel system description tool introduced by the thesis. It is a special finite mixture with a common ARX part in all its ARMAX components. The ARMMAX model structure depicts well the situations when there is a fixed deterministic input-output relationship described by the common ARX part while the characteristics of the stochastic disturbance part may vary. Based on the reexamined LD filter, the proposed MARMAX-QB and ARMMAX-QB algorithms provide efficient estimations for these mixtures with given C-parameters. Some properties of these two mixtures have also been studied. Especially, we show that they could provide certain approximate "algorithmic" parallelism, since their estimations provide a quantitative measure of descriptive quality of ARMAX components in parallel.

For on-line use, the ARMAX mixtures offer the flexibility in system description. The flexibility gained is paid at a relatively low price with the required computation load close to several recursive least-squares estimations.

For off-line use, the ARMAX mixtures offer the possibility to improve Bayesian estimation and prediction of single ARMAX model by dealing with unknown C-parameters in a novel way. While preserving Bayesian estimation setting up to the ARX part, the presented MMQ and MAQ estimation algorithms are able to cope with the unknown C-parameters not only in high dimensions (≥ 2) but also in the case with the roots of C-polynomial near/on the stability boundary. The main idea behind is to apply the selected multi-directional search (MDS) in the approximate "algorithmic parallel environment" of the ARMAX mixtures to generate a sequence of points that convergence to a critical point, ideally the "true" C-parameters of the studied ARMAX model. The algorithmic parallelism provided by the mixtures makes the parallel function evaluations of the MDS accessible to a single-processor machine. With the C-parameters of the used mixtures specified by the MDS procedure, the rest of estimation relies on the proposed ARMMAX-QB or MARMAX-QB algorithms.

Promising properties of the underlying theory and algorithms are illustrated on simulated and real data sets. Besides some sensitive testings, the comparisons between the MMQ and MAQ estimations with another proposed PE-related estimation are also provided.

Acknowledgements

This thesis was prepared during my study at Faculty of Applied Sciences, University of West Bohemia, Czech Republic. The work was undertaken at the Adaptive Systems Department, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic. It is within a joint project "Central Graduate School in Control in Systems Theory" (CEGS) that involves several research institutes and universities.

It is a pleasure to acknowledge the support I have received during the years leading to this thesis:

First of all, I would like to express my thanks to my supervisors, Ing. Miroslav Kárný, DrSc. and Prof. Ing. Pavel Žampa, CSc. for their guidance with patience and enthusiasm. Especially, I am grateful to Dr. Kárný, who has always been prepared to answer questions and help over the difficulties both in my scientific study and in my life staying abroad.

I have enjoyed to work with the colleagues of the institute and the university. Special thanks go to Slávek Belda, Ivan Nagy, Petr Gebouský and Tatiana V. Guy.

This work was financially supported by the CEGS through my home institute and university. It was also partially supported by the grants of AV ČR S1075102 and GAČR 102/03/0049.

Finally, I would like to thank my parents, brother, sister and Jan for their encouragement and *love*.

Symbols and Notations

Conventions

If x denotes a quantity, generally

x^*	the range of x
n_x	the number of elements in the set x^*
x_t	the value of x at discrete time instant t
$x_{i;t}$	the i-th entry of x_t
$x(t)$	a sequence up to time t, $x(t) \equiv \{x_1, \ldots, x_t\}$
$\int f(x)dx$	integral of function $f(x)$ over the range of x, i.e., $\int f(x)dx \equiv \int_{x^*} f(x)dx$.

Notations

f	reserved letter for probability (density) function
$\|$	conditioning symbol
$f(\cdot\|\cdot)$	conditional probability (density) function
$\mathring{t}$	horizon of discrete time instants, $\mathring{t} < \infty$
e_t	white noise at discrete time instant t
u_t	input variable at discrete time instant t
y_t	output variable at discrete time instant t
d_t	observed data at discrete time t, $d_t \equiv (y_t, u_t)$ particularly, $d(0)$ is the fixed prior input-output information
$d(t)$	observed data sequence up to time t, $d(t) \equiv \{d_1, \ldots, d_t\}$
ψ_t	regression vector, consisting of the past history of data $d(t-1)$ and the current input u_t
$'$	transposition symbol
Ψ_t	observed data vector $\Psi_t = [y_t, \psi_t']'$
p_t	the value of the unmeasured pointer at discrete time instant t, the pointer labels the active component of a mixture at each time instant
$p(t)$	pointers sequence up to time t, $p(t) \equiv \{p_1, \ldots, p_t\}$
p^*	the range of p_t, $p^* \equiv \{1, \cdots, n_p\}$, where n_p is identical to the number of mixture components
α	mixing probabilities of mixture $\alpha = [\alpha_1, \ldots, \alpha_{n_p}]$
θ	regression coefficients
Θ	overall unknown parameter set of a model, its precise meaning varies with the local context

$\hat{}$	denote point estimates, such as $\hat{\Theta}, \hat{\theta}, \hat{\alpha}$. In particular, $\hat{y}(t\|t-1)$ is one step prediction of output
$^{[o]}$	denote true description of a system, such as true pdf $^{[o]}f(\cdot)$ or true parameter value of parameter $^{[o]}\Theta$
$x \sim f$	x is distributed accorded to f
$\mathcal{N}_x(\mu, r)$	Gaussian (normal) distribution of x with mean μ and variance r
$GiW_{\theta,r}(V, \nu)$	Gauss-inverse-Wishart probability density function of θ and r, which owns sufficient statistics V and ν, where V is called extended information matrix and ν is the degrees of freedom
$Di_\alpha(\kappa)$	Dirichlet probability density function of α, which is shaped by a positive vector statistic κ
L, D	matrices determining $L'DL$ or LDL' decomposition, L is a lower triangular matrix with unit diagonal elements and D is a diagonal matrix with positive diagonal elements
$\mathcal{L}(d(t), \Theta)$	likelihood function, i.e., joint probability density function of observed data sequence $d(t)$, which is taken as a function of the parameter Θ
$\mathcal{D}(\cdot\|\|\cdot)$	Kullback-Leibler distance of a pair of densities
$\Gamma(\cdot)$	gamma function
$\mathcal{B}(\cdot, \cdot)$	beta function
$\delta_{\cdot,\cdot}$	Kronecker delta
$\text{supp}[g(x)]$	the support of nonnegative function $g(x)$, i.e., the subset of x^* on which $g(x) > 0$
$\arg max_x\ g(x)$	the value of x that maximizes function $g(x)$
λ	forgetting factor in stabilized forgetting, $\lambda \in (0, 1]$
Λ	flattening rate in iterative construction of prior probability density function, $\Lambda \in (0, 1)$
$\mathcal{E}[\cdot]$	expectation operator
$tr(A)$	trace of a matrix A
A^{-1}	inverse of a matrix A
$\|A\|$	determinant of a square matrix A
$\ln(x)$	natural logarithm of x, i.e., $\ln(x) = \log_e(x)$
$\exp(x)$	exponential of x
$\propto$	proportionality, i.e., equality up to a normalizing factor
$\equiv$	define as
$\mathbf{1}_p$	unit coordinate vector

Abbreviations

pf	probability function
pdf	probability density function
a.s.	almost surely
KL	Kullback-Leibler distance
SISO	single-input single-output
MIMO	multi-input multi-output
AR(X)	AutoRegressive model with/without exogenous input
ARMA(X)	AutoRegressive moving average model with/without exogenous input

MA	moving average part of a ARMA(X) model
ARMAX (n_a, n_b, n_c)	ARMAX model with input part, output part, MA part having orders n_a, n_b and n_c, respectively
OE	output error model
MARMA(X)	a general finite mixture of ARMA(X)
ARMMA(X)	a special finite mixture of ARMA(X), with all its ARMA(X) components having a common AR(X) part
PE	prediction error method
PEB	PEB estimation of ARMAX, i.e., a coupling of the PE method with Bayesian estimation
LS	least-squares method
EM	expectation and maximization algorithm
QB	Quasi-Bayes estimation algorithm
MARMAX-QB	QB estimation of MARMAX with known MA parts
ARMMAX-QB	QB estimation of ARMMAX with known MA parts
NM	Nelder-Mead simplex method
MDS	multidirectional search method
MAQ	MAQ estimation of ARMAX, i.e., a coupling of the ARMMAX-QB estimation with the MDS search
MMQ	MAQ estimation of ARMAX, i.e., a coupling of the MARMAX-QB estimation with the MDS search.

Chapter 1

Introduction

In the study of dynamic systems, a problem frequently encountered is that, in addition to possibly being excited by known inputs, the systems may also be driven by some unmeasured stochastic disturbances. ARMA(X) model (AutoRegressive Moving-Average with/without eXternal input) has been one of general tools to describe such systems in the presence of colored disturbances.

Exploiting Bayesian inference and finite mixture analysis, the thesis studies this standard model and investigates two types of its finite mixtures with emphasis on both theoretical and algorithmic aspects.

1.1 State of the Art

After being apparently first used in a statistical framework by Cochrane and Orcutt in 1949, the ARMAX model was introduced into system identification by Åström and Bohlin in 1965. Then Clarke introduced it into control literature in 1967. Since then, the ideas and techniques related to this model have played more and more important role in such diverse areas of science and technology as signal processing, adaptive control, communication, and biomedical engineering. It is also known to be equivalent to the linear state-space model that forms the corner stone of modern estimation and control theory, see [1] or [2].

Describing stochastic disturbances as a moving average (MA), the ARMAX model gains certain flexibility. It, however, also results in a nonlinear problem in its estimation with the product of two unknown quantities, namely the MA coefficients and the noise term, presenting in the cost function. Therefore, actually in many situations, such as adaptive control, to avoid the associated convergence and computational difficulties of ARMAX, people often still have to use another simpler AR(X) model (Auto-Regression with/without the eXogenous input). The popularity of the ARX model is mainly because that its estimation is simple and can be performed exactly by *Least Squares* (LS) method. Despite its usefulness, however, ARX has limited applicability since it copes only with uncorrelated process noise.

Among the multitude of estimation variants of ARMAX models, it seems that *approximate minimization of the sample variance of the Prediction Error* (PE) method, [1] or [2], and *Extended Least Squares* (ELS), see for example, [3], have become two popular options. Especially, the PE method has become something like a golden standard. These methods are, however, oriented only to point estimation. Consequently, only asymptotic information on precision of estimates is available. It implies that other tasks, for instance structure estimation, are weakly supported. Moreover, these methods impose the restrictive constraint on MA part by requiring its strict

stability and may face a range of problems when this assumption is (almost) violated.

From Bayesian viewpoint, the difficulty in estimation of ARMA(X) stems from the lack of sufficient statistic with the dimension smaller than the number of data samples. In 1981, Peterka [4] relaxed the stability restriction on the MA part and proposed a real-time Bayesian estimation of AR(X) part when its noise covariances are known. Essentially, he showed that LD factorization of the known covariance/correlation matrix acts as an optimal, time-varying, pre-whitening filter on the observed data. The LD type filtering does not impose any constraint on the MA part and allows its roots to be even at stability boundary. What's more, the resulting estimation with a chosen canonical state representation was proved to be algorithmically simpler and numerically more robust than the standard Kalman filter. The filtered data are then used in standard Bayesian estimation of its ARX part. Consequently, the uncertainties are under the control, Bayesian structure estimation can be used [5], etc.

Since it is rarely met in practice that the noise covariances of the model are known, Peterka [6] then extended the idea in 1986 to real-time estimation of ARMA(X) model under a weaker condition that the MA part is known only up to its C-parameters. Unfortunately, further attempts, such as to estimate the unknown C-parameters recursively, are hindered by the time-varying evolution of the LD filtering. To be at least able to select the most suitable one among finite prior candidates of the C-parameters, Peterka [7] proposed one improvement in 1989 based on a Bayesian comparison of hypothesis. He considered all chosen C-parameters simultaneously and evaluated *a posteriori* probabilities on hypotheses that specific C-parameters in the given set is the most suitable alternative. This improvement gives, however, no general rule how to generate such candidates. Moreover, regardless of the quality of the candidates, *a posteriori* probabilities converge to a zero-one vector in a generic case.

Thus, the extensive Bayesian use of ARMAX model is hindered mainly by the difficult estimation of its C-parameters. One of motivations of the thesis is to prolong the line of Peterka and to address improved Bayesian estimation of this model by dealing with its unknown C-parameters.

Meanwhile, the literature review and practical applications have indicated that a single model cannot deal with some complex problems, for instance, modelling non-homogeneous data in econometrics. Here non-homogeneous means that the data of interest arises from two or more sources. As a result, some more comprehensive models are called for combining the features of different models to describe and learn the behavior of complex systems. As one of such options, probabilistic finite mixtures have become a popular modelling tool with a widespread practical use [8]. To explore possible improvements on Bayesian approach by means of probabilistic mixtures is the other motivation of this work.

Bayesian inference simply considers finite mixtures as a special case of a multi-parameter estimation problem. Consequently, it is not difficult to obtain formal Bayesian solutions for mixtures, as long as estimation of individual components can be made. The problem is that numerical implementation of the formal solution is plagued with computational difficulties. For this reason, the formal solutions have to be complemented by some feasible approximations. With the developments of numerical analysis and increasing power of computers, there are several numerical methods currently available to provide such approximation with certain efficiency. Among them, Expectation-Maximization (EM) algorithm has become the most often used method, see for example [9]. Generally, this method is used in parameter estimation problem when the data are incomplete or have missing (hidden, unmeasurable) values [10]. Estimation of mixtures is one of its well-known applications. The EM algorithm provides a local maximum with a monotonic but rather slow convergence. Another classical method worth to be inspected is Markov Chain Monte Carlo (MCMC) technique. The MCMC technique can be easily combined with Bayesian approach to fit mixtures, but its known applications mainly deal with low dimensions [11]. The

thesis is mainly interested in and shall further study a relatively new method, Quasi-Bayes (QB) estimation method [12]. As an extension of a known algorithm [8], it has good properties, a Bayesian motivation as well as predictable and feasible computational complexity. Our interest in the QB method has been driven by these reasons.

Encouraged by these progress in mixture estimation and the success in estimation of single ARX model, estimation of ARX mixtures can be used extensively. Naturally, it is desirable to extend the existing mixture scheme of ARX to the case of ARMAX.

The thesis investigates the properties and the estimations on two types of ARMAX mixtures: One is MARMAX model, a general finite mixtures with ARMAX components. It is a natural mixture generalization of the ARMAX model. The other one is so called ARMMAX model, which is a novel system description tool introduced by the thesis. The ARMMAX model is a special finite mixture of ARMAX with the common ARX part in all its ARMAX components. Obviously, such a model structure depicts well the situations when there is a fixed deterministic input-output relationship described by the common ARX part while characteristics of the noise parts vary.

1.2 Aims and Ideas

The thesis generally attempts to improve Bayesian modelling and parameter estimation in the presence of stochastic colored disturbances. More specifically, we attempt to achieve the following aims:

1. *To reexamine Bayesian modelling from the view of probabilistic dynamic mixtures.*

   Although Bayesian approach can be easily taken to fit mixture models, a consistent mixture overview of Bayesian modelling [4] so far has not been well built yet. There are several possible ways to introduce mixtures into Bayesian modelling. Since mixtures can be interpreted as universal approximators of probability density functions [8, 13], the simplest and the most natural way is to make use such a property.

   However, facing our inability in estimation of general dynamic mixtures, we introduce and interpret a restricted description of dynamic mixtures in a feasible way.

2. *To reexamine the LD type filters. The emphasis is put on the* extended LD filter where the MA part is known only up to its C-parameters [6].

   This helps to clarify a few its implementation issues for efficient practical use of this type filters. It will also make a basis for us to address the case when the MA part is unknown. The whole remaining study will rest on these two reexaminations.

3. *To investigate the two types of ARMAX mixtures, namely the MARMAX model and the ARMMAX model. Specifically, to study and compare their modelling properties, to explore their feasible estimation algorithms and their possible applications.*

   The extended LD filter and the QB estimation, especially QB estimation of normal ARX mixture, throw light on how to efficiently estimate ARMAX mixtures when the MA parts are given.

   The developed MARMAX-QB and ARMMAX-QB estimations provide a quantitative measure of descriptive quality of ARMAX components in parallel. In this sense, the two ARMAX mixtures actually provide certain approximate "algorithmic parallelism" to inspect several ARMAX's in parallel.

4. *To improve Bayesian parameter estimation of ARMAX model by dealing with its unknown C-parameters. Our improvement is to reduce the uncertainty of the MA part by a point estimation of the C-parameters and preserve Bayesian setting up to the ARX part.*

   Since only a point estimate of the C-parameters is sought for, a natural idea is to use the existing point estimation methods, such as PE and ELS, for this task and then combine it with Bayesian method for the rest estimation. Following this idea, we consider a PE-related method, so-called PEB method. The nice properties of the PE method enable the hybrid PEB method to deal efficiently with the unknown C-parameters in high dimension (≥ 2). However, the PEB method inherits the requirements on the stability of the C-polynomial as well from the PE method.

   As pointed out by Peterka, the C-polynomial with its roots lying close to/on the unit circle is often a rule rather than an exception, especially in case of fast sampling. Thus, the potential difficulties in the stability requirements on the C-polynomial motivated us to adopt the idea of the extended LD filter rather than the other often used filters, such as stable inverse. This also encourages us to follow the line of Peterka to seek for approaches with their applicability not being limited by such stability requirements and the dimension (≥ 2) of the problem.

   We shall show that estimation of the C-parameters can be formulated as an unconstraint maximization problem and how the use of the extended LD filter restricts greatly our choices to deal with such a problem. With its simplicity, robustness and guaranteed convergence properties [14], a derivative-free *multi-directional search* (MDS) method [15] has finally been selected. As this method is generally more suitable to be executed in parallel, it is of our interest to make its parallel computing easier to be accessible to a single-processor machine. The approximate "algorithmic" parallelism of the ARMAX mixtures suggests us one interesting way to tackle this issue. The trick is to combine the MDS method with the ARMMAX-QB/MARMAX-QB estimation in the approximate "algorithmic parallel environment" of ARMMAX/MARMAX model for the parallel evaluations of objective function in the MDS. It finally leads to so-called MAQ and MMQ estimation algorithms.

1.3 Thesis Layout

After the introduction of this chapter, the remaining eight chapters of the thesis are organized in the following way: The reviews of Chapter 2 and Chapter 4 form the theoretical and algorithmic basis. The reexaminations of Chapter 3 and Chapter 5 present a restricted dynamic mixture modelling and the mechanism of the LD type filters. Aided by them, the four chapters followed are devote to the core issues and the conclusions. Specifically, the individual chapters are characterized as follows.

As a preliminary, *Bayesian Paradigm* together with *finite mixtures* and *simplex-based direct search* methods are briefly recalled in Chapter 2.

In Chapter 3, we reexamine Bayesian modelling from the view of probabilistic finite mixtures. The reexamination enables us to consider the standard classes of models, such as ARX, ARMAX, together with mixture of ARX, MARMAX, ARMMAX models within a unifying modelling framework. Our rest discussion actually is devoted to the exploration of the relationships between these models and their Bayesian estimations.

The existing Bayesian solutions: estimation of ARX models, approximate QB estimation of mixtures and QB estimation of ARX mixtures are recalled in Chapter 4. To provide the main schemes and principles necessary for achieving satisfactory estimation results, the essences of

several important related techniques are included as well. Chapter 5 revises the estimation of the ARMAX with known MA part, more exactly, LD type filters. These two chapters serve as an algorithmic basis, especially the extended LD filter plays important role in the estimation algorithms developed later.

Chapter 6 focuses on the investigation of the MARMAX and the ARMMAX mixtures. We shall discuss some of their modelling properties and propose the MARMAX-QB/ARMMAX-QB estimations method to address the estimation and prediction of these mixtures with given C-parameters of the MA parts.

Three hybrid methods, the MAQ and MMQ methods together with the PEB method, are then presented to improve Bayesian estimation on an ARMAX model. This gives the content of Chapter 7.

The effectiveness of the proposed estimations are illustrated on some simulated and real data sets, which goes to Chapter 8. It demonstrates the promising properties of the MAQ and MMQ algorithms and confirms the underlying theory. Besides some sensitive tests, we present the comparisons between these two estimations and the PEB method as well.

In the concluding chapter, 9, we summarize the main results and outline some directions of future work.

Some notes on the scopes

It is also fair to point out several restrictions of our discussion:

- The discussion focuses mainly on single-output (SO) normal (Gaussian) type models.

  majority of estimations presented can be easily extended to the multi-output (MO) case.

  For the estimation of ARMAX model, when the assumption that the MA part is known can be made, the majority results of Chapter 5 are directly applicable on its MO extension, see [6]. When such an assumption cannot be made, MO extension of Chapter 6 may not so straightforward. The approach studied in [16], where a MO linear system was described by a collection of SO models and thereafter the identification based on such kind of entry-wise models, provides one possible way to deal with the problem.

  To show the main idea how to extend the various SO QB mixture estimations to the corresponding MO cases, we present the factors level description of mixtures and the corresponding QB mixture estimation in Appendices at the end of the thesis.

- The proposed estimations are applicable on the the processes with external input. However, the problem of designing control strategy is not included. A detailed treatment related to the ARMAX models can be found in [17].

- The important structure estimation is not treated at all. Bayesian structure estimation [5], [18] can directly be used to relax this restriction.

Chapter 2

Preliminaries

The Bayesian modelling and estimation are used throughout the thesis. We shall briefly review its philosophy here together with some other relevant material which we need for the later discussion, such as *finite mixture distributions* and *simplex-based direct search* methods. Since the complete discussion of these issues are beyond the scope of the thesis, only the main ideas and principles behind are included here.

2.1 Bayesian Paradigm

The *Bayesian* methodology is one of techniques developed to cope with uncertainties and to solve statistical problems [4]. Generally, its final purpose is to provide a rational basis for some kind of decision-making by providing probability distributions of unknown quantities conditioned on the available knowledge.

2.1.1 Uncertainty and Probability

In Bayesian view, random means uncertain rather than a sole description of a process of observing a repeatable event. According to it, all forms of uncertainty can be inherently quantified by means of *probability*. Not only input-output data but also some unknown quantities such as model parameters and hypotheses are taken as random. Instead of being interpreted in terms of limits of relative frequencies or other objective ways, the concept of probability in Bayesian inference is used to describe uncertainty about the unknown random quantities.

Numerical values of probability of a random variable are described by *probability density function* (*pdf*) or *probability function* (*pf*) according to the type of the argument. The manipulation with the pdfs relies on a few properties and relations, we summarize some of the most important ones as follows.

Proposition 2.1.1 (Calculus with pdfs) *For any random variables α, β, γ, the following properties and relations hold for their* pdf/pf*:*

- Nonnegativity
$$f(\alpha, \beta|\gamma), f(\alpha|\beta, \gamma), f(\beta|\alpha, \gamma), f(\alpha|\gamma) \geq 0.$$
- Normalization
$$\int f(\alpha, \beta|\gamma) d\alpha d\beta = \int f(\alpha|\beta, \gamma) d\alpha = \int f(\alpha) d\alpha = 1.$$

- Independence

  If α does not depend on β, then they are mutually (unconditionally) independent.

  $$f(\alpha|\beta) = f(\alpha) \Leftrightarrow f(\beta|\alpha) = f(\beta) \Leftrightarrow f(\alpha,\beta) = f(\alpha)f(\beta). \tag{2.1}$$

  If α, β are conditionally independent under the condition γ, then

  $$f(\alpha|\beta,\gamma) = f(\alpha|\gamma) \Leftrightarrow f(\beta|\alpha,\gamma) = f(\beta|\gamma). \tag{2.2}$$

  Note that conditional independence does not imply unconditional independence.

- Marginalization

  $$f(\alpha|\gamma) = \int f(\alpha,\beta|\gamma)d\beta, \;\; f(\beta|\gamma) = \int f(\alpha,\beta|\gamma)d\alpha. \tag{2.3}$$

  This relation determines the marginal distribution from the corresponding joint probability distribution.

- Chain rule

  $$f(\alpha,\beta|\gamma) = f(\alpha|\beta,\gamma)f(\beta|\gamma) = f(\beta|\alpha,\gamma)f(\alpha|\gamma). \tag{2.4}$$

  This relation gives the rule how a joint probability distribution can be decomposed. It together with the marginalization operation determine the structure of the Bayesian reasoning.

  If we rewrite (2.4) as

  $$f(\alpha|\beta,\gamma) = \frac{f(\alpha,\beta|\gamma)}{f(\beta|\gamma)}. \tag{2.5}$$

  then, it describes the conditioning operation.

- Bayes rule

  $$f(\beta|\alpha,\gamma) = \frac{f(\alpha|\beta,\gamma)f(\beta|\gamma)}{f(\alpha|\gamma)} = \frac{f(\alpha|\beta,\gamma)f(\beta|\gamma)}{\int f(\alpha|\beta,\gamma)f(\beta|\gamma)d\beta}. \tag{2.6}$$

  This famous rule shows how a prior pdf $f(\beta|\gamma)$ can be corrected in the light of new information brought by the quantity α. This rule plays a critical role in the updating the prior pdf into a posteriori one when dealing with Bayesian estimation and prediction. Note that the integral in the denominator is just a normalizing factor which does not depend on β.

Note that the above relations are described only for the continuous case for simplification of the formulas. However it has to be kept in mind that the integration has to be replaced by summation whenever the argument of pdfs is discrete. In addition, for simplification on notation of pdfs, we do not explicitly and repeatedly state the conditions which are fixed when solving a given problem, we also omit the conditions which shall not create confusions when not stated explicitly.

In the analysis and design, *Kullback-Leibler (KL) distance* [19] is often used as a convenient tool to measure the proximity (distance) of a pair of pdfs.

Proposition 2.1.2 (KL distance) *Let f, g be a pair of pdfs acting on a common set x^*. Then, the Kullback-Leibler distance $\mathcal{D}(f||g)$ is defined as*

$$\mathcal{D}(f||g) \equiv \int_{x^*} f(x) \ln\left(\frac{f(x)}{g(x)}\right) dx, \tag{2.7}$$

and it has the following basic properties

1. *$\mathcal{D}(f||g) \geq 0$, i.e., nonnegativity holds.*
2. *$\mathcal{D}(f||g) = 0$ if and only if $f = g$ almost everywhere on x^*.*
3. *$\mathcal{D}(f||g) = \infty$ if and only if $g = 0$ and $f > 0$ on a set of positive measure.*
4. *$\mathcal{D}(f||g) \not\equiv \mathcal{D}(g||f)$, i.e., symmetry doesn't hold.*
5. *Triangle inequality doesn't hold.*

2.1.2 Bayesian Modelling

To interact with a system, a description of the system properties, a *model* is needed. Modelling of relationships among a sequence of observations is to provide a means to help us to get better understanding about the inspected complex systems. This section is devoted to a brief review of Bayesian modelling. It shall be reexamined from the view of finite dynamic mixtures in the next chapter.

System Models in General

Consider a stochastic system on which a time-oriented discrete data sequence $d_1, d_2, \cdots, d_t, \cdots$ is observed at discrete time instant $t = 1, 2, \cdots$. The sequence of data observed on the system up to time t is denoted by

$$d(t) = (d_1, \cdots, d_t),$$

where data record d_t is composed of a pair of variables

$$d_t = (u_t, y_t),$$

with u_t defining a directly manipulated input to the system at time t and y_t defining the output, i.e., the observed response of the system at time t to the past history of data $d(t-1)$ and the current input u_t. If y_t is a scalar value, one speaks of a SO (single-output) system, otherwise, it is a MO (multi-outputs) system. The discussion of the thesis is based on SO models.

Adopting Bayesian viewpoint, a complete description of the stochastic behavior of data at time instant t is given by a joint conditional pdf

$$f(d_t|d(t-1)) = f(u_t|d(t-1))f(y_t|u_t, d(t-1)). \tag{2.8}$$

The set of the first factor in the right hand of the equation

$$f(u_t|d(t-1)) \tag{2.9}$$

gives actually the *control strategy*. They describe the transformation, in general stochastic, by which the input u_t is determined on the basis of the known past history of the process. The collection of the remaining factor

$$f(y_t|u_t, d(t-1)) \tag{2.10}$$

gives a general description of system, a *system model*, in the form of a set of probability densities conditioned on data. It models the dependence of the output y_t on the known past history of input-output data, including the current input.

In practice, we usually have just partial knowledge of the system because of the existence of the unknown parameter Θ. Therefore, only a *parametric system model* is available as a set of conditional probability densities

$$f(y_t|u_t, d(t-1), \Theta). \tag{2.11}$$

It is parameterized by a finite dimensional parameter Θ and describes the dependence of the output y_t not only on the data but also on the unknown parameter Θ.

A model is called *linear*, if the expected value of output y_t conditioned on the past history of the input-output process depends on the past data linearly and if the variance of y_t does not depend on these data.

Modelling of Continuous Output

To make the general *parametric system model* (2.11) practical, it is necessary to express it through a finite number of parameters. The parametrization can be done in many ways differing in underlying assumptions.

Usually, if the output y_t is a continuous random variable, it may be useful to introduce a related random variable e_t as a difference between the output y_t and its expected value conditioned on the past history of the input-output process

$$e_t \quad = \quad y_t - \hat{y}_t(u_t, d(t-1), \Theta), \tag{2.12}$$

where the conditional expected value of output is

$$\hat{y}_t(u_t, d(t-1), \Theta) \equiv \mathcal{E}[y_t|u_t, d(t-1), \Theta] = \int y_t f(y_t|u_t, d(t-1), \Theta) dy_t. \tag{2.13}$$

One of important properties of the sequence $e_t, t = 1, 2, \cdots$ is its *whiteness*:

- these random variables have zero unconditional expectation, $\mathcal{E}[e_t] = 0$
- they are mutually uncorrelated, $\mathcal{E}[e_t e'_{t-i}] = 0, \quad i \neq 0,\ i < t.$

If we assume that the entire form of the distribution of e_t is independent of the past input-output data

$$f(e_t|u_t, d(t-1)) = f(e_t) \tag{2.14}$$

with the one-to-one mapping between y_t and e_t, the system model then can be given in the form of a stochastic equation

$$y_t = \hat{y}_t(u_t, d(t-1), \Theta) + e_t \tag{2.15}$$

If further on, the normality (Gaussian type) of e_t is assumed and the conditional expected value $\hat{y}_t \equiv \hat{y}_t(u_t, d(t-1), , \Theta)$ is expressed as a function of the past history of the input-output data through a finite set of parameter, *parametric system model* (2.11) can be specified as

$$f(y_t|u_t, d(t-1), \Theta) = \mathcal{N}_y(\hat{y}_t, r_e) \tag{2.16}$$

Here r_e is the time-invariant noise variance, when it is unknown, it should be included into the parameter collection and estimated as well.

Thus, under the above assumptions, the modelling problem for the problems of continuous output can be greatly reduced. Supposing various approximations of the $\hat{y}_t(u_t, d(t-1))$, we can arrive at different type models classes such as classic linear ARX model and ARMAX model. The detail derivation of some selective models go to the next chapter together with the mixture models under a mixture modelling framework.

2.1.3 General Bayesian Estimation and Prediction

In this section, we shall give an overview of general Bayesian estimation and prediction. The majority discussion of the rest thesis inspects how to apply this general theorem to provide specific parameter estimation and prediction for several dynamic processes driven by stochastic disturbances.

In the *classical* approach, the parameter is assumed to be *deterministic* but *unknown* constant, therefore a point estimate is mostly used.

Unlike it, the principle of the *Bayesian* approach is to take the uncertainty of the unknown parameters into account and consider the parameter itself as a *random* variable to which a certain prior *pdf* is assigned, even though it is actually a fixed unknown constant. Thus, Bayesian paradigm is known to lead to conceptually consistent treatment, since expectation in a cost function is taken with respect to both the stochastic behavior of the system and the uncertainty of parameter. However, the price for this is that it meanwhile increases enormously the corresponding "information state", given by the conditional probability of the original state and the parameter. Consequently, only the cases where the conditional probability can be specified through a finite-dimensional sufficient statistic, recursive updating of the full "information state" is practicable. For instance, the difficulty of the parameter estimation of ARMAX is due to the lack of such kind of sufficient statistic whose dimension is smaller than the number of data used for estimation.

Given a parameterized model, based on observations of other random variables that are correlated with the parameter Θ, Bayesian inference explores the relations among three types of *pdfs*:

- *a priori* pdf $f(\Theta|d(0))$.

  It expresses the preliminary (expert) knowledge of the unknown parameter or uncertain events and has to be assigned by analysis of the certain system or by the experience before the observed data are incorporated. Specification of a proper prior distribution has not been completely solved yet, especially in the context of mixtures. We shall review the iterative construction of prior pdf with flattening, one of most advanced techniques, in Chapter 4.3.4.

- *predictive pdf* $f(y_t|u_t, d(t-1))$.

  It is used to predict the next output y_t based on the known past history of the input-output data.

- *a posteriori pdf* $f(\Theta|d(t))$.

  It characterizes the uncertainty of unknown parameters or other events in light of the data.

Meanwhile, the existence of unknown parameter Θ generally leads to the conditional pdfs describing control strategy depend on the parameter, i.e.,

$$f(u_t|d(t-1), \Theta). \tag{2.17}$$

However, to be able to deal with the possible applications in adaptive control with a closed control loop allowed, a so-called *natural conditions of control* have to be used to excluded this dependence.

Definition 2.1.1 (Natural Conditions of Control) *Under* natural conditions of control*, the input u_t may depend on the unknown parameter Θ only through the past observed data $d(t-1)$. It means*

$$f(u_t|d(t-1),\Theta) = f(u_t|d(t-1)),$$

or equivalently the sole external signal u_t brings no information on the unknown parameters Θ,

$$f(\Theta|u_t, d(t-1)) = f(\Theta|d(t-1)).$$

With the informative knowledge provided by input-output data and a given prior, the Bayes rule (2.6) immediately gives us a tool for the sequential updating of parameter Θ.

Proposition 2.1.3 (General Bayesian estimation and prediction) *Under natural conditions of control, Definition (2.1.1), consider a parameterized model (2.11) of the studied system. then*

i) the evolution of a posteriori pdf *$f(\Theta|d(t))$ of unknown parameter is given by the recursion*

$$f(\Theta|d(t)) \quad = \quad \frac{f(y_t|u_t, d(t-1),\Theta)f(\Theta|d(t-1))}{f(y_t|u_t, d(t-1))} = \frac{\mathcal{L}(d(t),\Theta)f(\Theta)}{\mathcal{I}(d(t))}, \tag{2.18}$$

where the given prior pdf is written in the form $f(\Theta|d(0)) \equiv f(\Theta)$, since we have no data available at time $t = 0$. The product $\mathcal{L}(d(t),\Theta)$ with the inserted observations $d(t)$, called likelihood function*,*

$$\mathcal{L}(d(t),\Theta) = \prod_{\tau=1}^{t} f(d(\tau)|d(\tau-1), u_\tau, \Theta), \tag{2.19}$$

contains all information about Θ which can be extracted from data $d(t)$. $\mathcal{I}(d(t))$ is a normalization factor*, which is independent of Θ,*

$$\mathcal{I}(d(t)) = \int \mathcal{L}(d(t),\Theta)f(\Theta)d\Theta.$$

ii) the predictive pdf is

$$f(y_t|u_t, d(t-1)) = \int f(y_t|u_t, d(t-1),\Theta)f(\Theta|d(t-1))d\Theta, \tag{2.20}$$

or alternatively it can be expressed in terms of $\mathcal{I}(\cdot)$ as follows

$$f(y_t|u_t, d(t-1)) \quad = \quad \frac{\mathcal{I}(d(t+1))}{\mathcal{I}(d(t))}. \tag{2.21}$$

Remarks 2.1.1

1. *Note that it is often not easy to apply the above analytical Bayesian solution to a given particular case, for example in mixture estimation. But formal solution shows clearly the essence of the estimation and helps to construct reasonable approximations. In the later chapters, we shall return to this issue for some detail discussion.*

2. *The recursive evolution of the pdf $f(\Theta|d(t))$ allows us to interpret a posteriori pdf as a prior one before processing the new observations.*

3. *Let $\mathcal{M}$ be the set of models, which are considered as candidates to represent the system under study, label its elements by a "structural" pointer s and consider it as a part of parameters of system. Then, when it is unknown, the uncertainties we have to face are not only the uncertainty of model parameters Θ but also that of the structure of model. Bayesian approach treat also the latter uncertainty. The resulting* structure estimation *is solved elegantly as a special case of estimation [5]. For simplicity, throughout the thesis, we assume the structures of the studied models are given as prior information.*

2.2 Mixture Distributions

In order to describe and learn the behavior of complex systems, probabilistic mixtures together with multiple models, neural networks and cluster analysis are widely used to combine the features of different models. In practical applications, mixtures often also serve for describing a system with some unmeasurable data. In another words, mixtures have the ability to learn complex topologies. A finite mixture model describes the observed data by a convex combination of a finite number of probability density functions [8].

2.2.1 Description of Static Mixtures

Static mixtures with time-independent components and mixing weights are described here in this section. The thesis is mainly devoted to the discussion on dynamic processes, therefore a description of dynamic mixtures is needed. This will be considered in the next chapter when we address mixture modelling.

Suppose that a random variable or vector x takes values in a sample space, x^*, its distribution can be represented by *probability density functions* in the form

$$f(x) = \sum_{p=1}^{n_p} \alpha_p f_p(x), \qquad x \in x^* \tag{2.22}$$

where

$$\alpha_p \geq 0, \;\; p = 1, \cdots, n_p, \;\; \sum_{p=1}^{n_p} \alpha_p = 1. \tag{2.23}$$

With a finite number of components, i.e., $n_p < \infty$, then we shall say x admits a *finite mixture distribution* and $f(x)$ is a *finite mixture density function*. The parameters $\alpha = [\alpha_1, \cdots, \alpha_{n_p}]$ are called *mixing weights*. Individual *pdfs*, $f_p(x), i = 1, \cdots, n_p$, describe the *components* of the mixture.

Component densities often have specified parametric forms, so that the corresponding *finite parametric mixture* has the following description

$$f(x|\Theta) = \sum_{p=1}^{n_p} \alpha_p f_p(x|\Theta_p) \tag{2.24}$$

where each component $f_p(x|\Theta_p)$ is parameterized by the associated unknown parameters Θ_p. The complete parameters set of the mixture is the collection of all component parameters and mixing weights,

$$\Theta \equiv \{\alpha_p, \Theta_p\}_{p=1}^{n_p}.$$

Here if the number of components in mixture, n_p, is unknown, then it may has to be included into Θ to take this uncertainty into accounts.

Note that in the above descriptions, components are denoted in a way consistent with [8]. Another notation

$$f(x|\Theta_p, p) \equiv f_p(x|\Theta_p).$$

is actually adopted in our later discussion

Components need not to come from the same distribution, but most often they do. Different kinds of mixtures are formed, with the corresponding components modelled in a variety of distributions, like Gaussian (normal), Poisson, Gamma, log-Normal, Multinomial/Binomial, etc.. Normal mixtures, with an unspecified number of components, provide a flexible class of models which is widely used in statistical modelling. As remarked by Ferguson (1983) [8]:

An arbitrary density on the real line can closely be approximated by a normal mixture.

2.2.2 Related Issues

Universal Approximation Property

Mixtures can be interpreted as universal approximators of probability density functions or approximators based on radial basis functions. It means [8]:

> *Provided the number of component densities is not bounded, certain forms of mixture can be used to provide arbitrarily close approximations to a given probability distribution.*

Meanwhile, the insight could be also brought to *universal approximation property of mixtures* from *universal approximation theorem of neural networks* [13]. In the neural networks context, the hidden units provide a set of "functions" that constitute an arbitrary "basis" for the input patterns when they are expanded into the hidden-unit space. These function are called *radial basis functions*. Based on them, mixture models are closely related to neural networks. For example, Gaussian mixture models can be viewed as a form of generalized radial basis function network in which each Gaussian component is a basis function or 'hidden' unit.

The above universal approximation property of mixtures is important by providing necessary theoretical support for the approximations. It plays a critical role when we build a mixture-based Bayesian modelling framework later on. Nevertheless, from practical viewpoint, both universal approximation theorem of neural networks and the universal approximation property of mixtures have their corresponding limited practical value. In the mixture Bayesian modelling, Chapter 3, we shall return to this issue for finite dynamic mixtures.

Mixture Estimation

Efficient mixture estimation is one of most general statistical problems related to mixtures. In earlier period, *Graphical* method and *Moments* method were used for parameter estimation of mixtures. However, graphical method can only provide crude estimation of the underlying parameters while moments methods suffer from that it is not practical for a large number of parameters. Compared to them, *Maximum Likelihood Estimation*, *Bayesian Estimation* have got more and more attention for their convenient statistical properties.

By Bayesian inference, a finite mixture is simply a special case of a multi-parameter estimation problem. There is little additional note needed to be said to apply Bayesian paradigm to finite mixtures and obtain their formal Bayesian solutions. However, the implementation of these

formal Bayesian solutions are plagued with computational difficulties. The difficulties essentially stem from the complex analytical expression of *a posteriori* probability density function. The calculation of likelihood function and its integral would be the main computational burden. Consequently, the number of terms that need to be stored and handled blows up exponentially and cannot be handled exactly.

Fortunately, with the developments from numerical analysis field and increasing power of computers, there are many numerical methods available to provide certain efficient approximate estimation of mixtures, such as EM (Expectation- Maximization) algorithm [9], MCMC (Markov Chain Monte Carlo) techniques [11], and QB (Quasi-Bayes) estimations [12], to name some of them. The thesis adopts the QB mixture estimations: the existing estimation of normal ARX mixture is recalled in the next chapter, then we extend the QB estimation to the estimations of ARMA(X) mixtures in the Chapter 6.

The number of components

Another problem often encountered in mixture analysis is that the number of components in mixture, n_p, is unknown so that we need to take this uncertainty into accounts as well. How to decide the number of components is an important but difficult issue in many mixture applications. It often leads to the problems closely related to *cluster analysis*, see for example [20]. Together with the structure of individual parametric components, it is consider under *structure estimation* by Bayesian inference.

For the mixtures used in the thesis, in the case of mixture of ARX, this issue is well solved as a part of structure estimation, see [18]. The idea could be well extended to ARMAX mixtures cases. In particular, when employing ARMAX mixtures to estimation single ARMAX, the number of components in the mixtures coincides with the number of vertices in the simplex of the MDS algorithm and can be determined by the order of the MA part, see Chapter 6.

2.3 Direct Search Methods

When dealing with the estimation on the C-parameters of an ARMAX model later in Chapter 7, a general unconstrained optimization problem is encountered,

$$\min_{x \in R^n} g(x) \tag{2.25}$$

to minimize the scalar-valued nonlinear function $g(x)$ of n-dimensional x, $x \in R^n$, $g : R^n \to R$. Here, the information about the gradient of the objective function $g(x)$ is unavailable.

To solve the above problem, although there are many methods potentially can be exploited, we have focused on two important examples of *direct search* methods or more exactly *simplex-based direct search* methods. A direct search method does not use numerical or analytic gradients and use only function values to minimize the nonlinear function $g(x)$. A detailed discussion of direct search methods can be found in [21] or [22]. Simplex-based methods are a large subclass of direct search methods. Note that it should not be confused with the simplex algorithm of Dantzig for linear programming.

Essentially, simplex-based methods evolve *simplex*, an pattern with the evolving $n+1$ vertices in R^n,

$$\mathcal{S} =< x_1, \cdots, x_{n+1} >$$

Thus, in two-dimensional space, a simplex is a triangle; in three-dimensional space, it is a pyramid, etc. Generally, *non-degenerate simplex* is needed. A simplex is said to be non-degenerate,

if the n edges adjacent to any given vertex in the simplex span the space R^n. Two types of non-degenerate simplex are most often used:

- A *regular* simplex. It has all edges in the same length with the specified orientation. It can be created by a simple procedure described in the book [23].
- A *right-angled* simplex. With a given initial guess x_1, the rest vertices of a *right-angled* simplex is defined to be of some fixed distance in each of n coordinate directions from the initial guess point.

$$x_p = x_1 + \beta_p \mathbf{1}_p, \;\; p = 2, \cdots, n+1, \tag{2.26}$$

  where $\mathbf{1}_p$ denotes the unit coordinate vector and the scalar β_p is non-zero.

Some other kinds of non-degenerate simplex are also possible to be used. For example, the routine of the Nelder-Mead (NM) simplex method in Matlab toolboxes adopts a different type initial simplex.

Each iteration of a simplex-based direct search begins with a simplex from previous iteration. Next, one or more trial points are generated by some operations from the current simplex. Then the function values of the generated trial points are compared with the function values of the vertices in the current simplex. After each operation, if at least one trial vertex has a better function value than that of the current best vertex, the operation is called *successful*. To *accept* one operation, we replace the vertices of current simplex by the trial points after the operation. The iteration is terminated as a new simplex with its vertices function values satisfying some termination conditions.

The following two subsections reviews two of the most popular *simplex-based direct search* methods, respectively.

2.3.1 Nelder-Mead Simplex Method

As the most famous simplex-based direct search method, Nelder-Mead (NM) simplex method, was proposed by Nelder and Mead in 1965 [24], based on the idea of Spendley, Hext, and Himsworth [25]. Despite of its popularity in practice, this method has been perceived as completely "heuristic", since it is often plagued with a weak convergence analysis and some other troubles [21]. A recent progress has been reported in [26] that: In R^1, it is robust and convergence to a stationary point is guaranteed under the standard assumptions. In higher dimension $R^n, n > 1$, some general properties can also be proven, but no property can be provided to guarantee global convergence even for R^2.

The NM method creates a sequence simplex and modifies them to adopt the local landscape by five possible operations: *reflection*, *expansion*, *inside contraction*, *outside contraction*, *shrinkage*, each associated with a scalar of the operation. A nearly universal choice of these scalars are

$$\rho = 1, \quad \chi = 2, \quad \xi = 1/2, \quad \sigma = 1/2$$

where ρ, χ and σ are used for *reflection*, *expansion* and *shrinkage*, respectively. ξ is for both *inside contraction* and *outside contraction*. In Figure 2.1 and Figure 2.2, the effects of the possible operations are shown for the case of two dimensions in one iteration of the search..

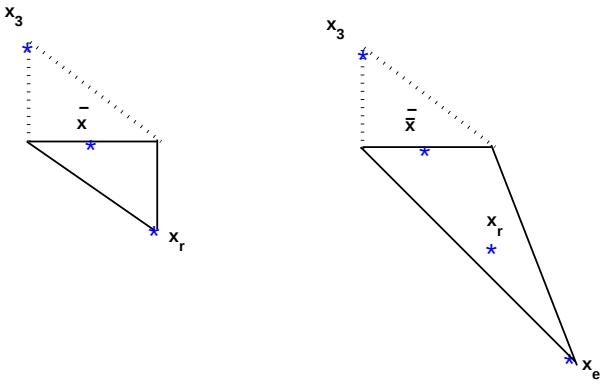


Figure 2.1: NM simplices after a reflection step and an expansion step. The original simplex is shown in dash with the best point denoted as x_1 and the worst one as x_3

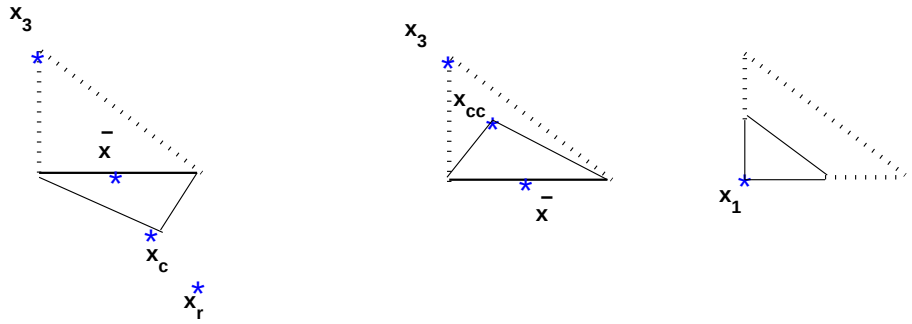


Figure 2.2: NM simplices after an outside contraction, an inside contraction and a shrink step.

One of advantages of this method is that it requires only one or two function evaluations in each iteration to construct a new simplex while the other methods may need much more function evaluations. It searches a strict improvement over the worst point. There are two possible outcomes of the NM method at each iteration: A single new test point shall be accepted and replace the worse vertex for the next iteration; or if a shrink is performed, a set of n new points, containing the best point, forms the new simplex for next iteration.

To perform the NM search, a routine is available in Matlab optimization toolbox [27] and the following basic description shows its principles.

Algorithm 2.3.1 (NM algorithm)

Initial phase

- *Select an initial guess* x_1^0 *and define a non-degenerate initial simplex formed by* $n+1$ *vertices*

$$< x_1^0, \cdots, x_{n+1}^0 >$$

- *Select the scalars,* ρ, χ, σ *and* ξ, *for reflection, expansion, contraction, shrink operations.*

- *Select stopping rules and set the counter* $j := 0$.
- *Evaluate the function values* $g(x_i^0)$, *for* $i = 1, \cdots, n+1$.
- *Order labels so that* $g(x_1^0) \leq g(x_2^0) \leq \cdots \leq g(x_{n+1}^0), i = 1, \cdots, n+1$, *using some given tie-breaking rule, see [26].*

Iterative phase

Do while stopping rule is not met, set $j := j + 1$.

1. ***Reflection***
   - *Define a reflected point* x^r
   $$x^r = \bar{x} + \rho(\bar{x} - x_{n+1}^{j-1}),$$
   where $\bar{x} = \sum_{i=1}^{n} x_i^{j-1}/n$ *is centroid of the best points (i.e., all vertices except the worst point* x_{n+1}^{j-1}*).*
   - *Evaluate the function value* $g(x^r)$.
   - *Accept the reflected point* x^r *and terminate the iteration, if* $g(x_1^{j-1}) < g(x^r) < g(x_n^{j-1})$.
2. ***Expansion***
   If $g(x_1^{j-1}) > g(x^r)$,
   - *Define expanded point* x^e
   $$x^e = \bar{x} + \chi(x^r - \bar{x}) = \bar{x} + \rho\chi(\bar{x} - x_{n+1}^{j-1}).$$
   - *Evaluate the function value* $g(x^e)$.
   - *Accept the expanded point* x^e *and terminate the iteration, if* $g(x^e) < g(x^r)$. *Otherwise, accept the reflected point* x^r *and terminate the iteration.*
   - *Go to step 5.*
3. ***Contraction*** *If* $g(x^r) > g(x_n^{j-1})$, *perform a contraction between* $\bar{x}$ *and the better one between* x_{n+1}^{j-1} *and* x^r.
   a. outside *if* $g(x_n^{j-1}) \leq g(x^r) < g(x_{n+1}^{j-1})$, *i.e.* x^r *is strictly better than* x_{n+1}^{j-1}.
   - *Define outside contracted point*
   $$x^c = \bar{x} + \xi(x^r - \bar{x}).$$
   - *Evaluate the function value* $g(x^c)$.
   - *Accept the outside contracted point* x^c *and terminate the iteration, if* $g(x^c) \leq g(x^r)$. *Otherwise, go to shrink step 4.*

   b. inside *if* $g(x^r) \geq g(x_{n+1}^{j-1})$,
   - *Define inside contracted point*
   $$x^{cc} = \bar{x} - \xi(\bar{x} - x_{n+1}^{j-1}).$$
   - *Evaluate the function value* $g(x^{cc})$.
   - *Accept the inside contracted point* x^{cc} *and terminate the iteration, if* $g(x^{cc}) \leq g(x^{n+1})$. *Otherwise, go to shrink step 4.*

4. ***Shrink***

   - *Define and accept n new shrink points, for $i = 2, \cdots, n+1$*

     $$x_i^s = x_1^{j-1} + \sigma(x_i^{j-1} - x_1^{j-1})$$

   - *Evaluate the function values $g(x_i^s)$, for $i = 2, \cdots, n+1$*

5. ***Order***
   Order the labels so that $g(x_1^j) \leq g(x_2^j) \leq \cdots \leq g(x_{n+1}^j)), i = 1, \cdots, n+1$, using some given tie-breaking rule [26].

2.3.2 Multidirectional Search

Inspired by the NM simplex method, *multidirectional search* (MDS) [15] has brought a new interest in direct search methods since 1989. It offers the following favorite features:

- It can be executed in parallel to take advantage of the computational parallelism.
- It does not require the information about the derivative of the objective function.
- It has strong proved convergence properties and robustness.
- It works well with "noisy" function values.

A great advantage of the MDS is its strong convergence analysis [14]. However the guaranteed property is paid by a rather slow convergence.

The basic idea of multidirectional search algorithm is to perform concurrent searches in multiple directions. Its goal is to construct a sequence of points, denoted as the best vertices of the simplex, that convergence to a critical point, ideally a maximizer of the objective function.

In contrast to the NM method, the MDS method searches a point strictly improving over the best vertex instead over the worst point as the NM does. Meanwhile, it requires $2n$ function evaluations in each iteration which is higher than that needed in the NM. But the acceptance criteria used in both methods is simple decrease rather than sufficient decrease.

There are three possible operations used in the MDS method: *reflection, expansion, contraction* with the associated operation scalars: ρ, χ and ξ, where unit reflection factor $\rho = 1$ is almost always used so that we shall not explicitly denote it later on. Although these operations are defined in the same way as the corresponding ones of the NM, they now involve the n edges of the simplex emanating from the best vertex so that the entire simplex is reflected, expanded, contracted. Figure 2.3 shows these three possible operations in the j-th iteration of search:

Firstly, a *reflection* step takes place. Through the best vertex x_1 of the current simplex, the remaining vertices are reflected to give the trial reflected points

$$x_i^r = 2x_1^{j-1} - x_i^{j-1}, \qquad \text{for} \quad i = 2, \cdots, n+1$$

If at least one reflected vertices has a better function value than the best vertex, the reflection is called successful and then one *expansion* step is followed. In this operation, reflected edges are further expanded to generate the new expanded vertices and check if any improvement can be made by increasing the searching step length,

$$x_i^e = x_1^{j-1} + \chi(x_1^{j-1} - x_i^{j-1}), \qquad \text{for} \quad i = 2, \cdots, n+1$$

with expansion factor χ.

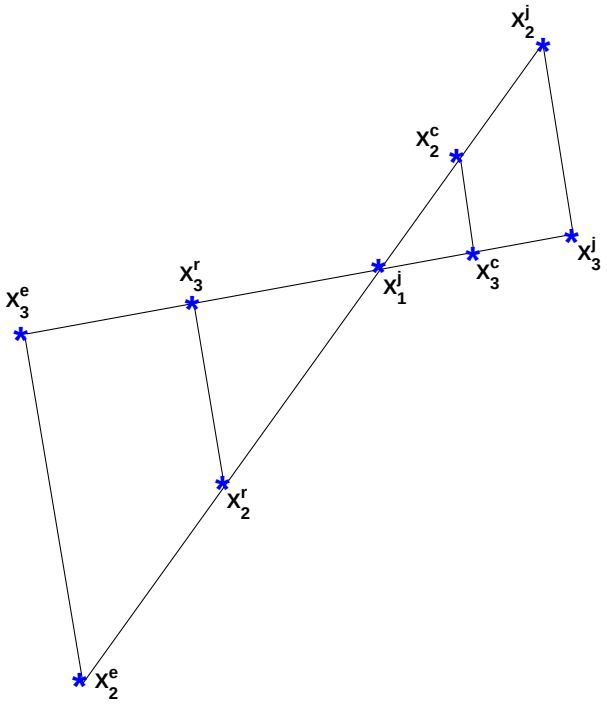


Figure 2.3: Three possible trial steps of the MDS method at the j-th iteration, where the current simplex is $< x_1^j, x_2^j, x_3^j >$ with x_1 denoting the best point while x_3 denoting the worst one. The trial points of the three operations are distinguished by the upper index.

Only when some expanded vertex is better than all the reflected vertices, we would accept the expansion simplex, otherwise we accept the reflection. Thus when reflection is successful, the final accepted simplex would be either a expansion simplex or a reflection one.

If a reflection is unsuccessful, i.e., none of reflected vertices gives a better function value than the best vertex, then a natural response is try to restart the search with a smaller simplex. It means that one *contraction* step is taken and accepted to contract the current original simplex by halving its every edge.

$$x_i^c = x_1^{j-1} + \xi(x_1^{j-1} - x_i^{j-1}), \qquad \text{for} \quad i = 2, \cdots, n+1$$

with the contraction factor ξ.

The MDS method preserves all information about the relative scaling elements across all iterations of the search, while the same cannot be said for the NM simplex algorithm. In contrast to the NM, the shape (angle) always remains the same as that of the original one in the MDS, although the size of the simplex is also modified all the time. The expansion and contraction steps automatically adjust the size of simplex by re-scaling the lengths of all the edges in the simplex. So that if the initial simplex is either too small or too big, the algorithm can rescale accordingly. But it, particularly in the case of too big initial simplex, may prove to be quite costly to spend a significant number of iterations to expand or contract simplex before any real progress can be made.

For implementation, the MDS needs to specify

- An initial simplex $\mathcal{S}_0 =< x_1^0, \cdots, x_{n+1}^0 >$ to start the search procedure.

If the problem is scale dependent so that the variable may differ signifcantly in scale, the use of right-angled simplex would be advantageous, otherwise, the use of a regular simplex is a "safer" choice.

- Scaling scalars associated with the operations.

  Formally, the MDS only requires the rational expansion factor χ is strictly greater than 1, i.e., $\chi > 1$, while the contraction factor ξ is a rational number between 0 and 1, i.e., $0 < \xi < 1$. But they are usually assigned the following values

  $$\chi = 2, \quad \xi = 1/2$$

  and unit reflection factor.

- Stopping criteria.

  The following two rules are often used:

  - To limit the number of iterations j by an upper bound J.
  - To inspect the relative size of simplex measured by the length of the longest edge adjacent to the best vertex x_1^j

    $$\frac{1}{\Delta} \max_{1 \le i \le n} ||x_i^j - x_1^j|| \le \epsilon, \;\; \epsilon \in (0,1), \;\; j \le J.$$

    Where $\Delta = \max(1, ||x_1^j||)$.

The multidirectional search algorithm described here shall help us to understand the specific features of its tailored versions MMQ and MAQ in Chapter 7.

Algorithm 2.3.2 (MDS algorithm)

Initial phase

- *Select an initial guess* x_1^0.
- *Define a non-degenerate initial simplex*

  $$\left\langle x_1^0, \cdots, x_{n+1}^0 \right\rangle$$

  formed by $n+1$ *vertices.*
- *Select expansion* $\chi \in (1, \infty)$ *and contraction* $\xi \in (0,1)$ *factors*
- *Select stopping rules.*
- *Set* j*, the counter of the total number of iterations used in the search, to zero.*
- *Evaluate the function values* $g(x_i^0)$*, for* $i = 1, \cdots, n+1$.
- *Swap the labels so that* $g(x_1^0) = \arg\min_i g(x_i^0), i = 1, \cdots, n+1$.

Iterative phase

Do while stopping rule is not met, set $j := j + 1$.

1. ***Reflection***

   - *Define* n *reflected vertices*

   $$x_i^r = 2x_1^{j-1} - x_i^{j-1}, \qquad i = 2, \cdots, n+1.$$

   - *Evaluate the function values* $g(x_i^r)$, *for* $i = 2, \cdots, n+1$.
   - *Go to the step 2 if* $\min g(x_i^r) < g(x_1^{j-1})$. *Otherwise, go to the step 3.*

2. ***Expansion***

   - *Define* n *expanded vertices*

   $$x_i^e = x_1^{j-1} + \chi(x_1^{j-1} - x_i^{j-1}), \qquad i = 2, \cdots, n+1.$$

   - *Evaluate the function values* $g(x_i^e)$, *for* $i = 2, \cdots, n+1$.
   - *Accept the expanded simplex, if* $\min g(x_i^e) < \min g(x_i^r)$, *i.e., replace* x_i^j *by the expanded trial points* x_i^e, *for* $i = 2, \cdots, n+1$. *Otherwise, accept the reflected simplex, i.e., replace* x_i^j *by the reflected trial points* x_i^r, *for* $i = 2, \cdots, n+1$.
   - *Go to step 4.*

3. ***Contraction***

   - *Define* n *contracted vertices*

   $$x_i^c = x_1^{j-1} + \xi(x_1^{j-1} - x_i^{j-1}), \qquad i = 2, \cdots, n+1.$$

   - *Evaluate the function values* $g(x_i^c)$, *for* $i = 2, \cdots, n+1$.
   - *Accept the contracted simplex, i.e., replace* x_i^j *by the contracted points* x_i^e, *for* $i = 2, \cdots, n+1$.

4. ***Swap***
   Swap the labels so that $g(x_1^j) = \arg\min_i g(x_i^j), i = 1, \cdots, n+1$.

Chapter 3

Stochastic Models of Dynamic Systems

Although finite probabilistic mixtures have become a fruitful application branch of Bayesian approach, a consistent mixture view of Bayesian modelling so far has not been well built yet.

First section of this chapter is devoted to the discussion on one possible dynamic mixture modelling. In particular, a restricted description of dynamic mixtures is introduced and interpreted.

The remaining sections present the descriptions of some specified parametric models within a unifying modelling framework.

3.1 Dynamic Mixture Modelling

Recall that to provide a *parametric system model* by Bayesian modelling is to define a family of conditional probability density functions, see Chapter 2.1.2

$$f(y_t|u_t, d(t-1), \Theta), \quad \text{for} \quad t \in t^*$$

We then face a problem of approximating such probability density functions.

The universal approximation property of mixtures discussed in Chapter 2.2.2 is applicable to dynamic cases as well. Thus, dynamic mixtures can be interpreted as universal approximators of probability density functions or approximators based on radial basis functions.

Based on these two facts, it is *conceptually* straightforward to extend dynamic mixtures to the Bayesian modelling as follows

$$f(y_t|u_t, d(t-1), \Theta) \;=\; \sum_{p=1}^{n_p} \alpha_p(u_t, d(t-1), \Theta) f(y_t|u_t, d(t-1), \Theta_p, p), \tag{3.1}$$

with its dynamic mixing weights satisfying

$$\alpha_p(u_t, d(t-1), \Theta) \geq 0, \;\; p = 1, \cdots, n_p, \;\; \sum_{p=1}^{n_p} \alpha_p(u_t, d(t-1), \Theta) = 1. \tag{3.2}$$

and its p-th *dynamic component* being described by the density $f(y_t|u_t, d(t-1), \Theta_p, p)$ with its associated component parameter set Θ_p.

To interpret the dynamic mixture description (3.1), let us consider a sequence of discrete random variables $\{p_t\}_{t\in t^*}$, each variable p_t has n_p possible values $\{1,\cdots,n_p\}\equiv p^*$. Then, the operations of marginalization (2.3) and chain rule (2.4) imply the following relations

$$\begin{aligned} f(y_t|u_t,d(t-1),\Theta) &= \sum_{p=1}^{n_p} f(y_t,p_t=p|u_t,d(t-1),\Theta) \\ &= \sum_{p=1}^{n_p} f(y_t|u_t,d(t-1),\Theta_p,p) f(p_t=p|u_t,d(t-1),\Theta). \end{aligned} \tag{3.3}$$

Note that here an assumption that the output y_t and the variable p_t at time t are conditionally independent of the past history of the variables $p(t-1)=\{p_1,\cdots,p_{t-1}\}$ has to be used.

Similarly to the natural conditions of control, Definition 2.1.1, if we assume that a sole external signal u_t brings no information on the unknown variable p_t, such that

$$f(p_t=p|u_t,d(t-1),\Theta)=f(p_t=p|d(t-1),\Theta), \tag{3.4}$$

and represent it by means of a data-dependent variable

$$\alpha_p(d(t-1),\Theta)=f(p_t=p|d(t-1),\Theta). \tag{3.5}$$

Thus, the discrete random variables $\{p_t\}_{t\in t^*}$ can actually be considered as pointers which label the active component at each discrete time instance t.

Obviously, using (3.4) – (3.5) to rewrite (3.3), we then obtain the general description of dynamic mixtures (3.1) under the condition that a sole external signal u_t brings no information on the unknown variable p_t.

Unfortunately, the complexity associated with the general description limits our ability to handle the corresponding estimation. Therefore, a restricted description of dynamic mixtures with constant mixing weights $\alpha_p(\Theta)$ and *dynamic component* $f(y_t|u_t,d(t-1),\Theta_p,p)$

$$f(y_t|u_t,d(t-1),\Theta) = \sum_{p=1}^{n_p} \alpha_p(\Theta) f(y_t|u_t,d(t-1),\Theta_p,p), \tag{3.6}$$

is actually used in the thesis. Note that in the later discussion, the notation $\alpha_p=\alpha_p(\Theta)$ is often used for the sake of simplicity.

The possibility to use such a restricted description is supported by a fact revealed by the proposition below.

Proposition 3.1.1 (Asymptotic convergence of mixing weights) *If we assume that a sole external signal u_t brings no information on the unknown variable p_t, i.e., (3.4), the stochastic process with random variables $\{\alpha_p(d(t-1),\Theta), t\in t^*\}$ is almost surely a convergent martingale.*

Proof: For a fixed Θ and p, using the definition (3.5) and the definition of expectation, we have

$$\mathcal{E}[\alpha_p(d(t),\Theta)|d_1,....,d_{t-1},\Theta]=\int f(p_t=p|d(t),\Theta) f(d_t|d(t-1),\Theta)\,dd_t. \tag{3.7}$$

Meanwhile, using chain rule (2.4), the following relation holds

$$f(p_t=p|d(t),\Theta) = \frac{f(p_t=p,d_t|d(t-1),\Theta)}{f(d_t|d(t-1),\Theta)}. \tag{3.8}$$

Substitute (3.8) into (3.7). After a cancellation, it follows

$$\mathcal{E}[\alpha_p(d(t),\Theta)|d_1,....,d_{t-1},\Theta] \;=\; \int f(p_t=p,d_t|d(t-1),\Theta)\,dd_t \tag{3.9}$$

Applying marginalization (2.3) and the definition (3.5) into (3.9), we finally obtain

$$\mathcal{E}[\alpha_p(d(t),\Theta)|d_1,....,d_{t-1},\Theta] \;=\; \alpha_p(d(t-1),\Theta). \tag{3.10}$$

Thus, according to [28], the process with random variables $\{\alpha_p(d(t-1),\Theta), t\in t^*\}$ is a martingale.

On the other hand, it is non-negative and bounded, since it corresponds to the probability (3.5).

According to martingale convergence theorem [29], it is then straightforward that $\alpha_p(d(t-1),\Theta)$ convergent almost surly to a constant probability. □

Clearly, this fact allows us to interpret our restricted dynamic mixtures (3.6) as an asymptotic limit of the general dynamic description (3.1).

Remarks 3.1.1

1. *It is easy to verify that the description of static mixtures with time-independent components and mixing weights, which is given in Chapter 2.2.1, corresponds to the general description (3.1) restricted by the assumptions,*

   Not only $\alpha(u_t,d(t-1),\Theta)$ is assumed to be constant and a sole external signal u_t brings no information on it

   $$\alpha(u_t,d(t-1),\Theta)=\alpha(\Theta), \tag{3.11}$$

   or in another words, the variable p_t at time t is conditionally independent of the past history of the data and the variables $p(t-1)$,

   $$f(p_t|u_t,d(t-1),p(t-1),\Theta)=f(p_t|\Theta),$$

   but also the output y_t at time t is conditionally independent of the past history of the data and the variables $p(t-1)$, such that

   $$f(y_t|u_t,d(t-1),p(t-1),\Theta)=f(y_t|\Theta). \tag{3.12}$$

2. *Note that, with the condition on the number of components, i.e., $n_p<\infty$, finite mixtures may be not always able to arbitrarily closely approximate a probability density functions by means of finite mixtures. In this sense, the finite mixture view of Bayesian modelling can not substitute the general Bayesian modelling as long as finite mixtures are in use.*

3. *The discussion of the thesis shall focus on the above component level description of SO mixtures. A brief discussion of the description of MO mixtures is given in the Appendix, where each component is further decomposed into a product of factors for predicting individual scalar output entries.*

4. *To design estimation method for the mixtures, it is convenient to adopt alternative descriptions of mixtures (3.6) using the introduced* discrete random pointer p_t. *We leave the detailed discussion on this issue to Chapter 4 and Chapter 6 when we discuss mixture estimations for specific models.*

5. Structure of a parameterized mixture *is determined by the number of components and* structures of parameterized components. *Structure of a parameterized normal regression component is determined by the structure of the corresponding regression vector. In the case of ARMA(X), the structure of the MA term has to be taken into account as well.*

To allow single-component mixture, $n_p = 1$, some general model classes, such as ARX, ARMAX, can be derived similarly to the traditional way [4] and [17].

With more than one components, $1 < n_p < \infty$, some more comprehensive model properties could be introduced, for example non-linearity. At present, the most research of the mixture field is focused on extending the basic models classes to the corresponding mixtures.

Selectively, some parametric models are described and introduced in the remaining sections. All of them are used in the thesis.

3.2 ARX Models

Consider the mixture description of dynamics (3.6), and assume that:
i) There is only one component;
ii) The older input-output data cannot bring any additional information about the expected value of output. Or in other words, only a finite and fixed length of the past input-output history is significant for the prediction of the output.
iii) The expected value of output depends on the past data linearly and its variance does not depend on these data.

The last two assumptions imply the expected value of output $\hat{y} = \hat{y}_t(u_t, d(t-1), \theta) = \mathcal{E}[y_t|u_t d(t-1), \theta]$ is a linear function of the finite number, say n, of foregoing input-output history

$$\hat{y}_t(u_t, d(t-1), \theta) \quad = \quad \theta' \psi_t, \tag{3.13}$$

where $'$ denotes transposition. The data $d_t = [y_t, u_t]$ are observed at time instance $t < \infty$. The n_{ψ_t}-dimensional vector $\psi_t = [u_t, y_{t-1}, u_{t-1}, \cdots, y_{t-n}, u_{t-n}]'$ is often called *regression vector*, it is a known function of the finite past data. The vector $\theta = [b_0, a_1, b_1, \cdots, a_n, b_n]'$ contains *regression coefficients*. It is a vector of unknown parameters with its dimension identical to n_ψ. Note that for the sake of simplicity, a constant term is missing here and in the model description given below, although such a constant term usually can not be eliminated if the parameters are unknown.

Applying it into the reduced system description (2.15), i.e.,

$$y_t = \hat{y}_t(u_t, d(t-1), \theta) + e_t, \quad t \in t^*$$

we come to the input-output relation of ARX model (Auto-Regression with eXogenous input), which is also often called as *linear regression*

$$y_t = \theta' \psi_t + e_t. \tag{3.14}$$

If it is assumed to be Gaussian type, then the ARX model is determined in terms of conditional pdfs as

$$f(y_t|u_t, d(t-1), \Theta) = \mathcal{N}_y(\theta' \psi_t, r_e), \quad t \in t^* \tag{3.15}$$

and parameterized by

$$\Theta \quad = \quad \{\theta, r_e\},$$

where $r_e = \mathcal{E}[e_t e_t']$ is constant noise variance.

Note that, when the regression model is identified from real data in practical applications, it is important to incorporate a sufficient long past history into the model and guarantee the white noise properties of e_t. It means that although only a finite and fixed length of the past input-output history are assumed to be used, this length n must be chosen sufficiently large.

3.3 ARMAX Models

Although ARX model is a very useful tool for system description, its applicability is limited by lacking of adequate freedom in describing the properties of the disturbance term. To add flexibility to it, the disturbance term could be described as *colored noise*, a moving-average (MA) of white noise.

Consider the mixture description of dynamics (3.6), and assume that:
i) There is only one component;
ii) The expected value of output $\hat{y}_t = \hat{y}_t(u_t, d(t-1), \Theta)$ is a function of the entire past history of input-output data.

To express $\hat{y}_t$ through a finite number of parameters, let us assume it is defined recursively through a finite number m of past data and a weighted sum of n_c past history of itself

$$\hat{y}_t + \sum_{i=1}^{n_c} c_i \hat{y}_{t-i} \quad = \quad \sum_{i=1}^{m} g_i y_{t-i} + \sum_{i=0}^{m} b_i u_{t-i}, \tag{3.16}$$

where $\hat{y} = \hat{y}_t(u_t, d(t-1))$. In a similar way as in the last section, a constant term is intentionally ignored for simplicity.

Applying it into the reduced system description (2.15), i.e.,

$$y_t = \hat{y}_t(d(t-1), u_t, \Theta) + e_t,$$

then it gives the input-output relation of ARMAX model (Auto-Regression with Moving Average noise and eXternal input)

$$y_t = \sum_{i=1}^{n} a_i y_{t-i} + \sum_{i=0}^{n} b_i u_{t-i} + \sum_{i=1}^{n_c} c_i e_{t-i} + e_t, \tag{3.17}$$

where $a_i = g_i - c_i$ with g_i, c_i being appropriated complemented by zeros. $n = m + n_c$ is a finite number.

Our discussion on the estimation of this model later on is closely related to that of the linear regressions (3.14). For this reason, we shall rewrite the above description of the ARMAX models in regression form as follows

$$y_t = \theta' \psi_t + v_t, \tag{3.18}$$

where the *regression* vector ψ_t and *regression coefficients* θ are defined in a similar way as last section. v_t is a *colored noise* with zero mean and the finite time-invariant correlation span

$$\begin{aligned} \mathcal{E}[v_t v_{t-i}] \quad &= \quad r_i, \quad for \quad i = 0, 1, ..., n_c, \\ &= \quad 0, \quad for \quad |i| > n_c, \end{aligned}$$

which can be considered as a moving average (MA) defined on the sequence of mutually uncorrelated and thus entirely unpredictable white noise $\{e_t\}$,

$$v_t = \sum_{i=0}^{n_c} c_i e_{t-i}$$

Note that $c_0 = 1$ has to be defined here. $c_1, \cdots, c_{n_c}$ are often called *C-parameters* and represented by an n_c-dimensional vector

$$C = [c_1, \cdots, c_{n_c}].$$

Obviously, the following relation then holds

$$r_i \quad = \quad r_e \sum_{k=i}^{n_c} c_i c_{k-i}, \qquad i = 0, 1, ..., n_c,$$

where r_e is constant noise variance. Moreover, if the known past history of input-output is long enough so that the influence of initial conditions $\hat{y}_i$, for $i = 1, \cdots, \max(n, n_c)$, may be negligible and if the normality of e_t can be assumed, then an ARMAX is fully parameterized through the following parameter set

$$\Theta \quad = \quad \{\theta,\ r_e,\ C\}$$

or alternatively

$$\Theta \quad = \quad \{\Theta_a,\ \ C\}.$$

Throughout the thesis, the part described by

$$\Theta_a = (\theta, r_e)$$

is often discussed as a whole and called as the AR(X) part of the corresponding ARMA(X) model. For this reason, we have introduced such a notation here.

In the above discussion, we considered the ARMA(X) models as an extension of regression models assuming that infinitely long past history of the process was available for the observation. The recursive definition (3.16) of the conditional expected value $\hat{y}_t$ makes sense only when the stability restriction is imposed on the C-polynomial (in the backward shift operator z^{-1})

$$C(z^{-1}) = 1 + c_1 z^{-1} + ... + c_{n_c} z^{-n_c}$$

by requiring all its roots to lie strictly outside the unit circle. However, when the ARMA(X) model is understood as a generator of the process driven by a colored noise v_t, then the stability of the C-polynomial $C(z^{-1})$ need not to be required.

These two interpretations of the ARMA(X) models are conceptually rather different also in the meaning of the random variable e_t:

- It is considered as the difference between the true value of output and its expected value.
- It is considered as a fictitious, unobservable and actually nonexisting white noise.

When it is defined in the former way, it could be reconstructed from the observed data, while it is not possible for the latter case if the C-polynomial is unstable.

Thus, we shall handle the ARMAX models in such a way that these two case are not distinguished. The study of LD type filters later on in Chapter 5 shall show, with a finite and growing length of observations, the time-invariant C-parameters have to be considered as time-varying,

they have to be recalculated and update in real time. It also shows they finally converge to the coefficients of a stable polynomial even when the original C-polynomial is unstable. This favorable fact makes us able to relax the stability requirement on the C-polynomial $C(z^{-1})$ to allow its roots to be even at the stability boundary.

Remarks 3.3.1

Note that an alternative form of the standard ARMAX model is known as Delta model. *Although their theoretically equivalence, Delta model appears to be numerically more robust and therefore more suitable for real-time computation in adaptive and self-tuning control.*

Both the ARMAX and Delta models *can be transformed into a canonical state-space model. State space representations make the the algorithms compact and uniform for both univariate and multivariate case for control, prediction, and system identification. They may require less the computational complexity as well.*

A study on these alternative representations of the ARMAX model can be found in [17]. The treatment on the problems of the related estimation, prediction and control can be found there as well.

3.4 ARX Mixtures

Consider the general description of mixtures (3.6), and assume that:

i) There is more than one but a finite number, $1 < n_p < \infty$, of components;

ii) Each component is described by one ARX model.

Then (3.6) is specified as an ARX mixture,

$$f(y_t|u_t, d(t-1), \Theta) \;=\; \sum_{p=1}^{n_p} \alpha_p f(y_t|u_t, d(t-1), \Theta_p), \tag{3.19}$$

with the mixing weights $\alpha = [\alpha_1, \cdots, \alpha_k]$ satisfying

$$\alpha_p \geq 0, \;\; p = 1, \cdots, n_p, \;\; \sum_{p=1}^{n_p} \alpha_p = 1,$$

Each normal ARX component is described by $f(y_t|u_t, d(t-1), \Theta_p) = \mathcal{N}_{y_t}(\theta_p' \psi_{p;t}, r_{e,p})$, and parameterized by the unknown associated parameter $\Theta_p = \{\theta_p, r_{e,p}\}$. The parameter set Θ of the mixture is the collection of all components' parameters and mixing weights

$$\Theta \equiv \{\alpha, \Theta_p\}_{p=1}^{n_p}.$$

This type of mixtures has been the most popular option in the applications of mixtures.

3.5 MARMAX Models

Consider the mixture description (3.6), and assume:

i) There is more than one but a finite number, $1 < n_p < \infty$, of components;

ii) Each component is described by one ARMAX.

Then, the general description (3.6) is specified as a natural mixture generalization of the autoregressive moving average. For this reason, we call it MARMAX model.

At each time instance t, the *probability density function* (pdf) of the MARMAX is given as

$$f\left(y_t|u_t, d(t-1), \Theta\right) = \sum_{p=1}^{n_p} \alpha_p f(y_t|u_t, d(t-1), \Theta_{a,p}, C_p) \tag{3.20}$$

where mixture weights satisfy

$$\alpha_p \geq 0, \;\; p = 1, \cdots, n_p, \;\; \sum_{p=1}^{n_p} \alpha_p = 1,$$

and $f(y_t|u_t, d(t-1), \Theta_{a,p}, C_p)$ describes its p-th ARMAX component. Here, the parameter $\Theta_{a,p}$ and $C_p = [c_{p,1}, \cdots, c_{p,n_c}]$ describe the ARX part and the C-parameters of the p-th components, respectively.

If the normality is assumed, the MARMAX model is then parameterized by the parameter set

$$\Theta \equiv \{\Theta_a, \; \Theta_c, \; \alpha\},$$

with

$$\Theta_a \equiv \{\Theta_{a,p}\}_{p=1}^{n_p}, \quad \Theta_{a,p} = (\theta_p \;\; r_{e;p}), \quad \Theta_c \equiv \{C_p\}_{p=1}^{n_p}, \quad \alpha \equiv [\alpha_1, \cdots, \alpha_{n_p}] \tag{3.21}$$

3.6 ARMMAX Models

Consider the mixture description (3.6), and assume:

i) There is more than one but a finite number, $1 < n_p < \infty$, of components;

ii) Each component is described by one ARMAX;

iii) There is a common deterministic ARX part in all ARMAX components while the characteristics of the stochastic noise parts vary.

These assumptions lead to the general description (3.6) being specified as a special mixture of ARMAX. This novel type model describes well the cases when common ARX part has a physical meaning of interest while C-parameters of the stochastic disturbances may vary. Since it mainly offers flexibility with respect to the MA noise term of the model, we call it ARMMAX model.

The *probability density function* of the ARMMAX model at each time instance t is

$$f\left(y_t|u_t, d(t-1), \Theta\right) = \sum_{p=1}^{n_p} \alpha_p f(y_t|u_t, d(t-1), \Theta_a, C_p) \tag{3.22}$$

with mixture weights

$$\alpha_p \geq 0, \;\; p = 1, \cdots, n_p, \;\; \sum_{p=1}^{n_p} \alpha_p = 1.$$

and $f(y_t|u_t, d(t-1), \Theta_a, C_p)$ describing its p-th ARMAX component.

Here the last assumption implies that there is a common parameter set Θ_a parameterizing the ARX parts of all ARMAX components. Thus, if the normality is assumed, the ARMMAX model is parameterized by

$$\Theta \equiv \{\Theta_a, \; \Theta_c, \; \alpha\},$$

with the common parameter set of the ARX parts $\Theta_a = \{\theta, r_e\}$ and different C-parameters of the MA parts $\Theta_c \equiv \{C_p\}_{p=1}^{n_p}$.

Chapter 4

Bayesian Estimation and Prediction

Some existing Bayesian estimations and predictions are reviewed in this Chapter. They provide the solid basis for us to discuss the estimations of the more difficult cases, such as MARMAX, ARMMAX and ARMAX models, in the later chapters.

To provide the main schemes and principles necessary for achieving satisfactory estimation results, the essences of the critical related techniques are included as well. The emphasis here is put on the grasping their main ideas. More detail treatments and broader views can be found in the given references.

4.1 Common Tools

There are many issues that may influence our ability to solve estimation and prediction tasks for various parameterized models. In particular, several of them are in common for all models considered in the thesis. For this reason, we first collect a few tools to deal with them, before we go to the Bayesian solutions of each specific model individually.

4.1.1 Estimation with Forgetting

Estimation with forgetting is often used to track slow changes of parameters and make the system adaptive. This is because the assumption that a certain set of parameters is strictly time-invariant is difficult to be fulfilled precisely in practice. Moreover, the parameters of the chosen model structure can be slightly different for different time intervals, since any mathematical model can be only an approximate description of reality. Thus, it is practically important to extend parameter estimation to parameter tracking.

The basic idea of tracking *slowly varying parameters* is to grasp both the significant time-varying relationships and the slow changes effects. This requires a compromise between

- admitting time variations of Θ_t,
- assuming that $\Theta_{t+1} \approx \Theta_t$.

The approach called *stabilized forgetting* [30] is often used to handle this optimization problem.

In terms of KL distance (2.7), the above requirements can be equivalently expressed by minimizing two KL distances simultaneously

- $\mathcal{D}\left(\hat{f}||f\right)$

- $\mathcal{D}\left(\hat{f}||f_A\right)$

where $\hat{f} \equiv f(\Theta_{t+1} = \Theta|d(t))$ is the pdf achieving the compromise between both of the requirements, $f \equiv f(\Theta_{t+1} = \Theta_t = \Theta|d(t))$ is the pdf assuming no parameter changes after measuring d_t and before processing d_{t+1}, while $f_A \equiv f_A(\Theta_{t+1} = \Theta|d(t))$ is an alternative one describing parameters after some changes.

To control the compromise, let us select a positive weight $\lambda \in (0, 1]$, then the desired pdf $\hat{f}$ is found as a minimizing argument of the functional

$$\lambda\mathcal{D}\left(\hat{f}||f\right) + (1-\lambda)\mathcal{D}\left(\hat{f}||f_A\right), \ \lambda \in (0, 1]. \tag{4.1}$$

It coincides with the geometric mean of the pair of pdfs

$$\hat{f} \propto f^{\lambda} f_A^{1-\lambda}, \tag{4.2}$$

or in the complete form

$$f(\Theta_{t+1} = \Theta|d(t)) \propto [f(\Theta_{t+1} = \Theta_t = \Theta|d(t))]^{\lambda}[f_A(\Theta_{t+1} = \Theta|d(t))]^{1-\lambda}. \tag{4.3}$$

Here the parameter λ is called *forgetting factor*, which can be interpreted as the probability that the parameters do not change. It can be either taken as a tuning knob or estimated.

Remarks 4.1.1

1. *Loosely speaking, the forgetting operation is a compromise between the selected alternative f_A and the posterior pdf $f(\Theta_{t+1} = \Theta_t = \Theta|d(t))$. It preserves the basic property of time updating so that the posterior pdf on parameters propagates without obtaining any new measured information.*

2. *The alternative pdf $f_A(\cdot)$ expresses the belief where the parameters might move within the time interval $[t, t+1)$ while we have no new observable information. This pdf has to be specified externally and possibly updated in the each time iteration.*

   Often, the pessimistic uniform alternative pdf ($\propto 1$) has been used. This special case of stabilized forgetting is called exponential forgetting*. It allows us to follow relatively fast parameter changes but it forgets the accumulated information with, often too high, exponential rate. For this reason, it is worth to preserve what we feel as a guaranteed information. The prior pdf $f(\Theta_{t+1} = \Theta)$ is a typical, reasonably conservative, choice of the alternative pdf $f_A(\Theta_{t+1} = \Theta|d(t))$.*

3. *The non-trivial alternative pdf prevents us to forget the "guaranteed" information as it is always incorporated after exponential forgetting. This stabilizes the whole learning and reflects positively in its numerical implementations. Without this, the posterior pdf may become too flat whenever the information brought by new data is not sufficient.*

4. *It is instructive to inspect the influence of forgetting on the data. The older the data are, the stronger "flattening" is applied to the corresponding model. Consequently, the older data influence the estimation results less than the new ones. Data are gradually "forgotten". This explains why the probability λ is called* forgetting factor*.*

5. *The forgetting factor λ is a parameter fixed throughout the time updating. The closer to unity the λ is, the slower changes are expected, or in another word, the higher weight the posterior pdf corresponding to the time-invariant case gets.*

4.1.2 Algorithm of Dyadic Reduction

To counteract numerical troubles in the decompositions and minimizations of a nonnegative definite quadratic form, an elementary *algorithm of Dyadic reduction* is often in need [17].

Proposition 4.1.1 (Algorithm dydr) *Consider a symmetric nonnegative definite or a positive semi-definite matrix matrix M of the rank 2, it can be expressed as a weighted sum of two dyads,*

$$M = [a, b] \begin{bmatrix} D_a & 0 \\ 0 & D_b \end{bmatrix} [a, b]', \tag{4.4}$$

where D_a, D_b, are positive scalar weights and a, b are column vectors of the same length $n_a \geq 2$.

The representation (4.4) is not unique and different equivalent forms can be found to define the identical kernels of the single quadratic form. If we consider a special equivalent form

$$M = [c, d] \begin{bmatrix} D_c & 0 \\ 0 & D_d \end{bmatrix} [c, d]', \tag{4.5}$$

where the j-th entries of c, d have the fixed values

$$c_j \equiv 1,\ d_j \equiv 0, \quad \text{for a chosen } j \in \{1, \ldots, n_a\}. \tag{4.6}$$

Then, the second decomposition (4.5) can be determine uniquely by means of the first one (4.4), using a simple algorithm given below.

Algorithm 4.1.1 (Dyadic reduction: dydr)

1. *Set $D_c = a_j^2 D_a + b_j^2 D_b$.*
2. *Set $x = \frac{a_j D_a}{D_c},\ y = \frac{b_j D_b}{D_c}$.*
3. *Set $D_d = \frac{D_a D_b}{D_c}$.*
4. *Set $c_j = 1,\ d_j = 0$.*
5. *Evaluate remaining entries of c, d as follows $c_i = x a_i + y b_i,\ d_i = -b_j a_i + a_j b_i,\ i \neq j$.*

4.1.3 Exponential Family

It is practically important in real time identification to require only a finite and fixed size of the memory to store finite-dimensional *sufficient statistics* without loss any useful information. It is implied by the fact that the data set is growing persistently in real time. For instance, the difficulty in the estimation of ARMA(X) model stems from the lack of such kind of sufficient statistic.

Therefore, it is of interest to study model classes with such property. They are essentially from exponentially family, enriched by uniform distribution with unknown boundaries. Many useful properties are associated with this family. In particular, they are known to have *conjugate (self-reproducing) prior* and fixed-dimensional sufficient statistics which exist both for unknown parameters and the predicted output.

To be able to deal with dynamic cases, our definition of exponential family additionally requires a non-standard recursive updating of the data vector as below.

Agreement 4.1.1 (Exponential family) *For a random variable y_t, its pdf is said to belong to the dynamic exponential family if it can be written in the form*

$$f(y_t|u_t, d(t-1), \Theta) \equiv f(y_t|\psi_t, \Theta) = A(\Theta)D(\Psi_t)\exp[\langle B(\Psi_t), C(\Theta)\rangle], \tag{4.7}$$

where $\langle\cdot,\cdot\rangle$ is a functional operator linear in the first argument. In our context, we define

$$\langle x, y\rangle = \begin{cases} x'y & \text{if } x,\ y \text{ are vectors, } ' \text{ is transposition} \\ \mathrm{tr}[x'y] & \text{if } x,\ y \text{ are matrices, tr denotes trace} \end{cases} \tag{4.8}$$

$\Psi_t' \equiv [y_t, \psi_t']$ *is a finite dimensional* data vector*, it is recursively updated by* $(d_t, \Psi_{t-1}') \to \Psi_t'$,

$A(\Theta)$ *is a non-negative scalar function of parameters* Θ,

$D(\Psi_t)$ *is a non-negative scalar function of data* Ψ_t,

$B(\Psi_t), C(\Theta)$ *are functions of* Ψ_t *and* Θ *respectively, with compatible finite and fixed dimensions.*

An inspection whether there is a wider set of parameterized models with the advantageous properties of the exponential family opens just a narrow space [31]. Essentially, the exponential family coincides with all parameterized models that are sufficiently smooth functions of parameters and with supports independent of parameters. Uniform distribution with unknown constant boundaries represents one of a few feasible examples of pdfs out of the exponential family. The class of models that lead to a finite-dimensional characterization of pdfs occurring in filtering is even more restrictive. Its discussion can be found in [32].

The general functional recursions of updating parameters and predicted outputs for the exponentially family can be reduced to an algebraic recursive updating of the finite-dimensional sufficient statistics, i.e., a array V_t and a scalar ν_t. Thus, the estimation and prediction within this family becomes very simple, especially with the conjugate prior pdf.

Proposition 4.1.2 (Estimation and prediction in exponential family) *Under natural conditions of control (Definition 2.1.1), consider a model, which belongs to the exponential family (4.7) and is parameterized by time-invariant* $\Theta_t = \Theta \in \Theta^*$.

i) If a priori pdf $f(\Theta)$ is assigned with its support being restricted by the indicator $\chi(\cdot)$ of the set Θ^*, then a posteriori pdf of unknown parameters is

$$f(\Theta|d(t)) = \frac{\mathcal{L}(d(t), \Theta)f(\Theta)}{\mathcal{I}(V_t, \nu_t)} = \frac{A^{\nu_t}(\Theta)\exp[\langle V_t, C(\Theta)\rangle]\chi(\Theta)f(\Theta)}{\mathcal{I}(V_t, \nu_t)}, \tag{4.9}$$

with statistics recursively evolving as

$$V_t = V_{t-1} + B(\Psi_t),\ V_0 = 0; \quad \nu_t = \nu_{t-1} + 1,\ \nu_0 = 0. \tag{4.10}$$

The predictive pdf is given by

$$f(y_t|u_t, d(t-1)) = \frac{D(\Psi_t)\mathcal{I}(V_{t-1} + B(\Psi_t), \nu_{t-1} + 1)}{\mathcal{I}(V_{t-1}, \nu_{t-1})}, \tag{4.11}$$

where normalization integral $\mathcal{I}(V, \nu)$ is evaluated by

$$\mathcal{I}(V, \nu) = \int A^{\nu}(\Theta)\exp[\langle V, C(\Theta)\rangle]f(\Theta)\chi(\Theta)\, d\Theta. \tag{4.12}$$

ii) If the prior is chosen in the conjugate form determined by the "prior statistics" V_0, ν_0

$$f(\Theta) \propto A^{\nu_0}(\Theta)\exp[\langle V_0, C(\Theta)\rangle]\chi(\Theta), \tag{4.13}$$

and if

- V_0, ν_0 *replace the zero initial conditions in (4.10),*
- *the indicator* $\chi(\cdot)$ *is formally used as the prior pdf.*

Then the estimation and prediction formulas (4.9), (4.11) are valid.

iii) If further on, stabilized forgetting with forgetting factor $\lambda \in [0,1]$ *is used to allow slow parameter changes and the alternative pdf given in the conjugate form determined by the pair of sufficient statistics* $V_{A;t}, \nu_{A;t}$*, then, the estimation and prediction formulas (4.9), (4.11) remain valid with the statistics evolving according to the recursion*

$$\begin{aligned} V_t &= \lambda(V_{t-1} + B(\Psi_t)) + (1-\lambda)V_{A;t}, \quad V_0 \text{ given} \\ \nu_t &= \lambda(\nu_{t-1} + 1) + (1-\lambda)\nu_{A;t}, \quad \nu_0 \text{ given.} \end{aligned} \tag{4.14}$$

Note that $D(\Psi_t)$ doesn't influence the estimation and it enters the prediction independently. Meanwhile, the need to have the complete recursion of the statistics explains why our definition of exponential family requires for the possibility to update Ψ_t recursively.

4.2 Bayesian Solution to ARX Models

After having recalled the common tools, we shall review the existing Bayesian estimation of ARX models in this section and QB estimations of (ARX) mixtures in the next section.

Consider a parameterized normal ARX model described by the pdf

$$f(y_t|u_t, d(t-1), \Theta) = \mathcal{N}_{y_t}(\theta'\psi_t, r_e), \tag{4.15}$$

with

$$\mathcal{N}_{y_t}(\theta'\psi_t, r_e) \equiv (2\pi r_e)^{-0.5} \exp\left\{-\frac{(y_t - \theta'\psi_t)^2}{2r_e}\right\} = (2\pi r_e)^{-0.5} \exp\left\{-\frac{1}{2r_e}\mathrm{tr}\left(\Psi_t\Psi_t'[-1,\theta']'[-1,\theta']\right)\right\}, \tag{4.16}$$

where ψ is the *regression vector*, while $\Psi \equiv [d, \psi']'$ is the data vector, $'$ denotes transposition. The model is parameterized by the regression coefficients and noise variance of the driving white noise

$$\Theta \equiv \{\theta, r_e\}.$$

For simplicity, the index e of r_e is often omitted whenever there is no danger of confusion.

It is straightforward to apply general Bayesian estimation and prediction to ARX models. However, in order to get numerically satisfactory identification result, several techniques specifically related to this model have to be complemented.

4.2.1 GiW Conjugate Prior Pdf

Recall the fact that normal regression models belong to the exponential family, with the following correspondence to the general form of exponential family (4.7)

$$\mathcal{N}_d(\theta'\psi, r) = A(\Theta)D(\Psi)\exp\left[\langle B(\Psi), C(\Theta)\rangle\right],$$

with

$$\begin{aligned} A(\Theta) &\equiv r^{-0.5}, \quad D(\Psi) \equiv (2\pi)^{-0.5}, \quad B(\Psi) \equiv \Psi\Psi' \\ C(\Theta) &\equiv 2r^{-1}[-1,\theta']'[-1,\theta'], \quad \langle B, C\rangle \equiv \mathrm{tr}[B'C]. \end{aligned}$$

Thus, it possess conjugate (self-reproducing) prior in the form of Gauss-inverse-Wishart (GiW) distribution

$$GiW_\Theta(V,\nu) \equiv GiW_{\theta,r}(V,\nu) \equiv \frac{r^{-0.5(\nu+n_\psi+2)}}{\mathcal{I}(V,\nu)} \exp\left\{-\frac{1}{2r}\mathrm{tr}(V[-1,\theta']'[-1,\theta'])\right\}, \quad (4.17)$$

where n_ψ is the dimension of regression vector ψ, it is identical to the dimension n_θ of the vector of regression coefficients. The positive definite matrix V_t is called *extended information matrix*. It together with the degrees of freedom ν_t form sufficient statistics for estimation of Θ.

The updating of V_t is often poorly conditioned and the use of its $L'DL$ *decomposition* is the only safe way to counteract the induced numerical troubles. This issue is reviewed in the next section, and thereafter the properties of the GiW pdf are summarized there using the $L'DL$ decomposition.

4.2.2 $L'DL$ Decomposition of Extended Information Matrix

$L'DL$ Decomposition

The recursively updating of the extended information matrix V plays decisive role in the estimations. If a stabilized forgetting with a forgetting factor λ is used, as a specification of (4.14), the V in the context of AR(X) is updated by the recursion

$$V_t = \lambda(V_{t-1} + \Psi_t\Psi_t') + (1-\lambda)V_{A;t}$$

Obviously, a recursion of the type

$$V = \lambda V + \beta\Psi\Psi'$$

is then required in essence. Here the positive scalar β actually equal to the forgetting factor λ. However, in the mixture context, normal ARX component/factor requires a weighted updating $V_{t-1} + w\Psi_t\Psi_t'$ instead. This leads β to be determined by both the forgetting factor λ and the weight w, see the next section for details. Thus, for the completeness and consistence of the notations, we introduce the quantity β.

On the other hand, as mentioned earlier, the updating is poorly conditioned and it is necessary to use an $L'DL$ *decomposition*

$$V = L'DL, \quad (4.18)$$

where L is a *lower triangular matrix with a unit diagonal and* D *is a* diagonal matrix with positive diagonal entries. Experience confirms that a safe processing of real data records is impossible without it. Moreover, the evaluation of the determinants occurring in various pdfs becomes simple with the $L'DL$ decomposition.

Hence, it follows that the recursion $V = \lambda V + \beta\Psi\Psi'$ has to be converted into a recursion $L'DL = \lambda L'DL + \beta\Psi\Psi'$ and the updating is applied on the matrices L, D instead of V itself.

Based on the algorithm of Dyadic reduction, Proposition 4.1.1, the efficient updating can be performed as follows

Algorithm 4.2.1 (Updating of $L'DL = \lambda L'DL + \beta\Psi\Psi'$)

1. *Set* $b = \Psi,\ D_y = \beta$.

2. *Call the function* dydr, *for* $j = \mathring{\Psi}, \mathring{\Psi} - 1, \ldots, 1$
   $[D_j, D_y, j\text{-th column of } L', b] = \mathsf{dydr}\,(\lambda * D_j, D_d, j\text{-th column of } L', b, j)$

The book [4] or [33] gives a broader view on the $L'DL$ decomposition.

Complete squares and minimization of quadratic forms

Next, let us examine how the $L'DL$ decomposition of the extended information matrix V works on the quadratic forms

$$[-1,\, \theta']V[-1,\, \theta]'$$

in the considered pdf (4.16).

Proposition 4.2.1 (Completion of squares) *Consider the extended information matrix V and its $L'DL$ decomposition, and partition them according to the two parts ordered in the corresponding data vector $\Psi \equiv [y, \psi']'$*

$$\begin{aligned} V &= \begin{bmatrix} V_y & V'_{y\psi} \\ V_{y\psi} & V_\psi \end{bmatrix},\ V_y \text{ is scalar}, \\ L &= \begin{bmatrix} 1 & 0 \\ L_{y\psi} & L_\psi \end{bmatrix},\quad D = \begin{bmatrix} D_y & 0 \\ 0 & D_\psi \end{bmatrix},\ D_y \text{ is scalar}. \end{aligned} \tag{4.19}$$

Then, it holds for the quadratic form that

$$\begin{aligned} [-1,\, \theta']V[-1,\, \theta']' &\equiv [-1,\, \theta']L'DL[-1\,\theta']' = (\theta - \hat{\theta})'L'_\psi D_\psi L_\psi(\theta - \hat{\theta}) + D_y \\ \hat{\theta} &\equiv L_\psi^{-1}L_{y\psi} \equiv \text{least-squares estimate of } \theta, \end{aligned} \tag{4.20}$$

where the quadratic form is minimized by $\theta = \hat{\theta}$ and the minimum reached is D_y.

Basic properties and moments of GiW pdf

Now, let us summarize some properties of the GiW pdf based on the above partitioned $L'DL$ decomposition of the extended information matrix.

Proposition 4.2.2 (Basic properties and moments of GiW pdf) *For the GiW pdf (4.17), if the partitioned $L'DL$ decomposition of the extended information matrix (4.19) is considered, then*

1. *$GiW_{\theta,r}(V, \nu)$ has, besides (4.17), the following alternative expressions*

$$\begin{aligned} GiW_{\theta,r}(V,\nu) &\equiv GiW_{\theta,r}(L, D, \nu) \\ &= \frac{r^{-0.5(\nu+n_\psi+2)}}{\mathcal{I}(L, D, \nu)} \exp\left\{-\frac{1}{2r}\left[(L_\psi\theta - L_{y\psi})'D_\psi(L_\psi\theta - L_{y\psi}) + D_y\right]\right\} \quad (4.21) \\ &\equiv \frac{r^{-0.5(\nu+n_\psi+2)}}{\mathcal{I}(L, D, \nu)} \exp\left\{-\frac{1}{2r}\left[(\theta - \hat{\theta})'C^{-1}(\theta - \hat{\theta}) + D_y\right]\right\}, \quad (4.22) \end{aligned}$$

   with the least-squares estimate of θ,

$$\hat{\theta} \equiv L_\psi^{-1}L_{y\psi},$$

the covariance factor of least-squares estimate,

$$C \equiv L_\psi^{-1} D_\psi^{-1} (L'_\psi)^{-1},$$

and the normalization integral $\mathcal{I}(L, D, \nu)$*,*

$$\mathcal{I}(L, D, \nu) = \Gamma(0.5\nu) D_y^{-0.5\nu} |D_\psi|^{-0.5} 2^{0.5(\nu+n_\psi)} (2\pi)^{0.5 n_\psi}, \tag{4.23}$$

$$\Gamma(x) = \int_0^\infty z^{x-1} \exp(-z)\, dz < \infty, \quad \textit{for } x > 0 \tag{4.24}$$

which doesn't depend on the unknown parameters θ *and* r*. Clearly, it is finite iff* $\nu > 0$ *and* V *is positive definite (or equivalently,* D *is positive definite).*

2. *The GiW pdf has the following marginal pdfs and moments:*

   - *The marginal pdf for the unknown noise variance* r*,*

$$f(r|L, D, \nu) = \frac{r^{-0.5(\nu+2)}}{\mathcal{I}(D_y, \nu)} \exp\left[-\frac{D_y}{2r}\right], \tag{4.25}$$

   with the associated normalization integral and moments

$$\begin{aligned}
\mathcal{I}(D_y, \nu) &\equiv \frac{\Gamma(0.5\nu)}{(0.5 D_y)^{0.5\nu}}, \\
\mathcal{E}[r|L, D, \nu] &= \frac{D_y}{\nu - 2} \equiv \hat{r}, \quad \mathrm{var}[r|L, D, \nu] = \frac{2\hat{r}^2}{\nu - 4}, \ \mathcal{E}[r^{-1}|L, D, \nu] = \frac{\nu}{D_y}, \\
\mathcal{E}[\ln(r)|L, D, \nu] &= \ln(D_y) - \ln(2) - \frac{\partial \ln(\Gamma(0.5\nu))}{\partial(0.5\nu)}.
\end{aligned}$$

   - *The marginal pdf for the unknown regression coefficients* θ

$$f(\theta|L, D, \nu) = \mathcal{I}^{-1}(D, \nu) \left[1 + D_y^{-1} (\theta - \hat{\theta})' L'_\psi D_\psi L_\psi (\theta - \hat{\theta})\right]^{-0.5(\nu+n_\psi)}, \tag{4.26}$$

   with the associated normalization integral and moments

$$\begin{aligned}
\mathcal{I}(D, \nu) &\equiv \frac{|D_\psi|^{0.5} \prod_{i=1}^{n_\psi} \Gamma(0.5(\nu + n_\psi - i))}{D_y^{0.5 n_\psi}}, \\
\mathcal{E}[\theta|L, D, \nu] &= L_\psi^{-1} L_{d\psi} \equiv \hat{\theta}, \ \mathrm{cov}[\theta|L, D, \nu] = \frac{D_y}{\nu - 2} L_\psi^{-1} D_\psi^{-1} (L'_\psi)^{-1} \equiv \hat{r} C.
\end{aligned}$$

4.2.3 Estimation and prediction

Normal regression models belong to the exponential family, thus the estimation and prediction reduce to algebraic recursive updating of the finite-dimensional sufficient statistics.

Proposition 4.2.3 (Estimation and prediction of ARX models) *Consider a normal regression model (4.15), if a conjugate prior* $GiW_\Theta(V_0, \nu_0)$ *(4.17) and a conjugate alternative* $GiW_\Theta(V_{A;t}, \nu_{A;t})$ *in stabilized forgetting (see Section 4.1.1), are used, then, the Bayesian posterior pdf of unknown parameters remain as*

$$f(\Theta|d(t)) = GiW_\Theta(V_t, \nu_t) \tag{4.27}$$

with the sufficient statistics evolving according to the recursions

$$V_t = \lambda(V_{t-1} + \Psi_t\Psi_t') + (1-\lambda)V_{A;t},\ V_0 \text{ given} \tag{4.28}$$

$$\nu_t = \lambda(\nu_{t-1} + 1) + (1-\lambda)\nu_{A;t}, \quad \nu_0 > 0 \text{ given.} \tag{4.29}$$

The partitioned $L'DL$ decomposition of the matrix V, (4.19) and Algorithm 4.2.1, has to be propagated for numerical stability.

The predictive pdf *is known in the form of* Student distribution. *With the data vector $\Psi_t = [y_t, \psi_t']'$, its values can be directly evaluated according to the expression of the* Student form

$$f(y_t|\psi_t) = \frac{\sqrt{2}\Gamma\left(0.5(\nu_{t-1}+1)\right)\left[D_{y;t-1}(1+\zeta)\right]^{-0.5}}{\Gamma(0.5\nu_{t-1})\left(1+\frac{\hat{e}_t^2}{D_{y;t-1}(1+\zeta)}\right)^{0.5(\nu_{t-1}+1)}}, \tag{4.30}$$

where $\Gamma(\cdot)$ is the gamma function (4.24), $\hat{e}_t \equiv y_t - \hat{\theta}_{t-1}'\psi_t$ is the prediction error, and $\zeta \equiv \psi_t' L_{\psi;t-1}^{-1} D_{\psi;t-1}^{-1} (L_{\psi;t-1}')^{-1}\psi_t$.

Alternatively, the values of predictive *pdf* can also numerically evaluated as the ratio

$$f(y_t|\psi_t) = \frac{(2\pi)^{-0.5}\mathcal{I}(V_{t-1} + \Psi_t\Psi_t', \nu_{t-1}+1)}{\mathcal{I}(V_{t-1}, \nu_{t-1})}, \tag{4.31}$$

where the same type integrals $\mathcal{I}(V,\nu)$ (4.23) is used.

Remarks 4.2.1

1. *The evolution of sufficient statistics is equivalent to well-known recursive least-squares (LS) estimates:*

$$\hat{\theta} \equiv L_\psi^{-1}L_{y\psi} \quad \textit{is LS estimate of } \theta, \tag{4.32}$$

$$\hat{r} \equiv \frac{D_y}{\nu-2} \quad \textit{is LS estimate of } r, \tag{4.33}$$

$$\hat{r}C \equiv \hat{r}L_\psi^{-1}D_\psi^{-1}(L_\psi')^{-1} \quad \textit{is covariance matrix of LS estimate of } \theta. \tag{4.34}$$

2. *The predictive pdf is the key element needed in the estimation. It is thus worthwhile to check which is computationally simpler between the presented two formulas in the evaluation of its values.*

   The former (4.30) is less suitable for the evaluation if the prediction errors $\hat{e}_t$ are not explicitly required, but it has advantage that the prediction errors $\hat{e}_t$ suit for an intuitive judgement of the predictive properties of the inspected model.

   The latter one (4.31) use an explicit expression for the integral $\mathcal{I}(V,\nu)$ (4.23) and is often more suitable for the evaluation. The derivation of this formula can be found in [4]. It is this formula (4.31) that is most often used in the thesis.

4.3 Mixture Estimation and Numerical Approximation

Consider the mixture

$$f(y_t|u_t, d(t-1), \Theta) = \sum_{p=1}^{n_p} \alpha_p f(y_t|u_t, d(t-1), \Theta_p, p), \tag{4.35}$$

which is parameterized by the collection

$$\Theta \equiv \{\alpha_p, \Theta_p\}_{p=1}^{n_p}.$$

The mixture weights $\alpha = [\alpha_1, \cdots, \alpha_{n_p}]$ satisfy

$$\alpha_p \geq 0, \;\; p = 1, \cdots, n_p, \; n_p < \infty, \; \sum_{p=1}^{n_p} \alpha_p = 1,$$

and each component is parameterized by Θ_p and described by

$$f(y_t|u_t, d(t-1), \Theta_p, p).$$

As mentioned in Chapter 2.2.2, although to obtain formal Bayesian solutions for the above mixture is easy, its implementation is plagued with computational difficulties that the resulting formal analytical expression of posterior *pdf* consists of a product of a sum of function depending on observed data and unknown parameters. Here in this section we consider a feasible approximate treatment by the Quasi-Bayes (QB) estimation.

4.3.1 Alternative Description of Mixtures

For the design and description of the QB estimation method, it is convenient to use discrete random pointers $p_t \in p^* = \{1, 2, \cdots, n_p\}$ to label the active component at each discrete time instance $t = 1, 2, \cdots, \mathring{t}$. The pointers $p_1, \cdots, p_{\mathring{t}}$ are assumed to be mutually independent with time-invariant probabilities

$$f(p_t = p|u_t, d(t-1), \Theta) = \alpha_p$$

Similarly to the EM method, the pointer p_t can be considered as the unmeasured data of the system. Thus, with them together with measured output y_t, there is an alternative joint view on the mixture at each time instant t,

$$f(y_t, p_t|u_t, d(t-1), \Theta) = \prod_{p \in p^*} [\alpha_p f(y_t|u_t, d(t-1), \Theta_p, p)]^{\delta_{p,p_t}}, \tag{4.36}$$

where δ is *Kronecker* symbol defined by $\delta_{a,b} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$. By marginalization (4.36) over p_t, the marginal pdf $f(y_t|u_t, d(t-1), \Theta, \alpha)$ gives the general mixture description (4.35).

4.3.2 Dirichlet Conjugate Prior Pdf of Mixing Weights

For the introduced pointers $p_t \in \{1, \ldots, n_p\} = p^*$, it holds

$$\alpha_{p_t} = f(p_t|u_t, d(t-1), \Theta) = \prod_{p \in p^*} \alpha_p^{\delta_{p,p_t}},$$

where, by definition,

$$\alpha_p = Pr\{p_t = p\} = f(p_t = p|u_t, d(t-1), \Theta), \quad p = 1, 2, .., n_p,$$

which satisfy the constrains $\alpha_p \geq 0, \sum_{p=1}^{n_p} \alpha_p = 1$.

It corresponds to a special case of *Markov chain*. And as we known, Markov Chains belong to the exponential family so that they possess conjugate (self-reproducing) prior in the form know as Dirichlet distribution.

Thus, the distribution on mixing weight α can be specified as in the conjugated *Dirichlet* form.

$$f(\alpha) = Di_\alpha(\kappa). \tag{4.37}$$

The Dirichlet distribution is shaped by n_p-vector statistic κ with positive entries κ_p,

$$Di_\alpha(\kappa) = \frac{\prod_{p\in p^*} \alpha_p^{\kappa_p - 1}}{\mathcal{B}(\kappa)} \propto \prod_{p\in p^*} \alpha_p^{\kappa_p - 1}, \tag{4.38}$$

where $\mathcal{B}(\kappa) \equiv \frac{\prod_{p\in p^*} \Gamma(\kappa_p)}{\Gamma(\sum_{p\in p^*} \kappa_p)}$.

4.3.3 Approximate QB Estimation

Based on the discussion of the previous two subsection, approximate QB estimation method is described here.

Let us specify prior pdfs of each individual components $f(\Theta_p|d(t))$, ideally in conjugate form. Moreover, assume the *pdf* of all parameters $f(\Theta|d(t))$ in the product form

$$f(\Theta|d(t)) \propto \prod_{p\in p^*} f(\Theta_p|u_t, d(t-1)) Di_\alpha(\kappa_{p;t}). \tag{4.39}$$

Then, the Bayes rule can be used to update it to *a posteriori* pdf

$$f(\Theta, p_{t+1}|u_{t+1}, d(t)) \propto \prod_{p\in p^*} [f(y_{t+1}|u_{t+1}, d(t), \Theta_p, p)]^{\delta_{p,p_{t+1}}} f(\Theta_p|d_t) \alpha_p^{\kappa_{p;t} + \delta_{p,p_{t+1}} - 1}.$$

Here the alternative mixture description (4.36) and the Dirichlet conjugate prior pdf (4.37) of mixing weights are used.

In order to obtain the approximation of the desired *pdf* $f(\Theta|u_{t+1}, d(t))$, a marginalization over p_{t+1} could be performed. However, it destroys the assumed form of (4.39). One possible way to preserve this form is to replace $\delta_{p,p_{t+1}}$ by its conditional expectation

$$\delta_{p,p_{t+1}} \quad \approx \quad w_{p;t+1} \equiv \mathcal{E}[\delta_{p,p_{t+1}}|u_{t+1}, d(t)] = f(p_{t+1} = p|u_{t+1}, d(t)). \tag{4.40}$$

here the scalars $w_{p;t+1}$ can be calculated by integrating (4.40) over the parameters Θ

$$w_{p;t+1} \quad = \quad \frac{\hat{\alpha}_{p;t} f(y_{t+1}|u_{t+1}, d(t), p)}{\sum_{\tilde{p}\in p^*} \hat{\alpha}_{\tilde{p};t} f(y_{t+1}|u_{t+1}, d(t), \tilde{p})} = \frac{\kappa_{p;t} f(y_{t+1}|u_{t+1}, d(t), p)}{\sum_{\tilde{p}\in p^*} \kappa_{\tilde{p};t} f(y_{t+1}|u_{t+1}, d(t), \tilde{p})}, \tag{4.41}$$

which requires the point estimate of mixing weight

$$\hat{\alpha}_{p;t} = \frac{\kappa_{p;t}}{\sum_{\tilde{p}\in p^*} \kappa_{\tilde{p};t}} \propto \kappa_{p;t}, \tag{4.42}$$

and the Bayesian prediction of each component

$$f(y_{t+1}|u_{t+1}, d(t), p) \equiv \int f(y_{t+1}|u_{t+1}, d(t), \Theta_p) f(\Theta_p|u_{t+1}, d(t))\, d\Theta_p = \frac{\mathcal{I}(d(t+1)|p)}{\mathcal{I}(d(t)|p)}. \quad (4.43)$$

Note that the approximation (4.41) can be interpreted as the Bayes rule applied to the discrete unknown random variable $p_t \in p^*$ with the prior probability $\hat{\alpha}_{p;t}$.

By inserting the approximation (4.40) and (4.41) into (4.39), the approximately updated posterior pdf preserves the same functional product form as (4.39). Here, the posterior of each individual components is updated by

$$f(\Theta_p|d(t+1)) \propto [f(y_{t+1}|u_{t+1}, d(t), \Theta_p, p)]^{w_{p;t+1}} f(\Theta_p|d(t)) \quad (4.44)$$

and the updating *a posteriori Dirrichlet pdf* of the mixing weights leads to a updating its statistic

$$\kappa_{p;t+1} \quad = \quad \kappa_{p;t} + w_{p;t+1}. \quad (4.45)$$

Algorithm 4.3.1 (QB estimation algorithm of mixtures)

Initial (off line) mode

- *Select the complete structure of the mixture, i.e. specify the number of components n_p, and the structure of each component.*
- *Select prior pdfs $f(\Theta_p)$ of the individual components, ideally, in the conjugate form with respect to the parameterized components $f(d_t|\psi_{p;t}, \Theta_p, p)$. Specify the initial values of the associated statistics $V_{p;0}, \nu_{p;0}$.*
- *Select prior pdf of component weight α in the form of Dirichlet pdf, $f(\alpha) = Di_\alpha(\kappa)$. Specify the initial values $\kappa_{p;0} > 0$. Intuitively motivated values about $0.1\mathring{t}$ were found reasonable.*
- *Select forgetting factor $\lambda \in (0, 1]$, alternatives $f_A(\Theta_p)$ and $f_A(\alpha)$ for stabilized forgetting.*

Sequential (on line) mode,

1. *Compute the point estimates of the component weights $\hat{\alpha}_{p;t}$ by means of (4.42).*
2. *Acquire the data record d_{t+1}.*
3. *Compute the values of the predictive pdfs for each individual components $p \in p^*$, with (4.43) and the measured data record d_{t+1}.*
4. *Compute the probabilistic weights $w_{p;t+1}$ by (4.41).*
5. *Update a posteriori* Dirrichlet pdfs *of mixing weights by the evolution of the scalars $\kappa_{p;t+1}$ by (4.45).*
6. *Update a posteriori* pdfs *$f(\Theta_p|d(t+1))$ of the parameters associated with individual components according to the weighted Bayes rule (4.44).*
7. *Evaluate, if need be, the characteristics of $f(\Theta_p|d(t+1))$ describing other parameters Θ_p.*
8. *Repeat the sequential mode when $t \leq \mathring{t}$.*

Remarks 4.3.1

1. *Note that the predictions of individual components without data weighting is performed for evaluation of the weights $w_{c;t+1}$. The parameter estimation algorithm performs almost the same algebraic operations but with weighted data. This simple observation is exploited in implementation of the algorithm. Moreover, it can be used in judging of estimation quality.*

2. *The algorithm is applicable whenever the Bayesian estimation and prediction for each component can be managed. Thus, the scope is restricted to the parameterized model class admitting finite-dimensional statistics. It consists essentially of the* exponential family *augmented by the uniform distribution with unknown boundaries. Normal regression models and Markov chains, are prominent examples of dynamic components. Others are more or less restricted to static ones.*

   Later in Chapter 6, we shall discuss how to extend the idea of the QB algorithm to deal with the estimation of ARMAX mixtures based on a pre-whitening.

3. *To increase the chance to gain a successful estimation, some iterations are necessary so that* iterative QB estimation *is in need. Meanwhile, a hybrid of the quasi-Bayes and EM algorithms, batch QB estimation, may be desirable sometimes.*

   To avoid redundant discussion, we shall only demonstrate the essence of using iterations and batch version in the next section on the estimation of ARX mixtures.

4.3.4 Iterative Construction of Prior Pdf with Flattening

A proper specification of a prior pdf is generally not a trivial task, especially in the context of mixtures where the prior knowledge is dealt with at the component level: a prior pdf for each component is considered as an entity on its own [34]. A study on the issue with various techniques can be found in [35]. Here we only consider one of most important techniques: how to construct iteratively prior pdf using flattening to reduce our uncertainty on the quality of a given prior.

Generally, the Bayes rule specifies the posterior estimates in a non-iterative fashion

$$f(\Theta|d(\mathring{t})) \propto f(d(\mathring{t})|\Theta)f(\Theta),$$

which relies not only on information provided by data but also on the prior knowledge. To increase the chance to obtain a successful estimation, using some iteration is proved to be helpful. It solves the estimation problem as a repeated task with the same data sample. The basic idea behind is to use *a posteriori* pdf $f(\Theta|d(\mathring{t}))$ instead of $f(\Theta)$ as the new guess of prior pdf in each iteration, hoping that $f(\Theta|d(\mathring{t}))$ could give a better clue about the unknown parameters than the original $f(\Theta)$.

It leads to an iterative use of the Bayes rule,

$$f_n(\Theta|d(\mathring{t})) \propto f(d(\mathring{t})|\Theta)f_{n-1}(\Theta|d(\mathring{t}))$$

with $f_n(\Theta|d(\mathring{t}))$ being the posterior pdf obtained through the n-th iteration. If $f_0(\Theta) \equiv f(\Theta)$, then $f_n(\Theta|d(\mathring{t})) \propto [f(d(\mathring{t})|\Theta)]^n f(\Theta)$.

Meanwhile, notice the fact that if the original prior pdf has been assumed unreliable, its posterior pdf then is possibly too concentrated on a false set of parameters. It implies a direct

using the new obtained *a posteriori* pdf as a new prior could be dangerous. To prevent this possible false over-confidence, a *flattening mapping* $\mathcal{G}$

$$\mathcal{G} : f_{n-1}\left(\Theta|d(\mathring{t})\right) \to \hat{f}_n(\Theta) \tag{4.46}$$

has to be used after each time of iteration, as a compromise between the following contradictory requirements

- $\hat{f}_n(\Theta)$ resembles $f_{n-1}(\Theta|d(\mathring{t}))$;
- $\hat{f}_n(\Theta)$ is flat enough. To serve for this purpose, a prototype of the flat distribution $\bar{f}(\Theta)$ is in use and the $\hat{f}_n(\Theta)$ is required to resemble it. Uniform pdf, even improper one, often suits to be such a flattening alternative.

In each individual iterations, if we express the above requirements in terms of the KL distance (2.7), then they lead to minimizing two KL distances simultaneously

- $\mathcal{D}\left(\hat{f}||f\right)$
- $\mathcal{D}\left(\hat{f}||\bar{f}\right)$.

Similarly to what we did in Section 4.1.1 for *stabilized forgetting*, a positive weight q can be selected as a design parameter to control the compromise, then the desired pdf $\hat{f}$ is found as a minimizing argument of the functional

$$\mathcal{D}\left(\hat{f}||\tilde{f}\right) + q\mathcal{D}\left(\hat{f}||\bar{f}\right),\ q > 0 \tag{4.47}$$

which coincides with the geometric mean of the pair of pdfs

$$\hat{f} \propto f^{\Lambda}\bar{f}^{1-\Lambda} \quad \text{with} \quad \Lambda = 1/(1+q) \in (0,1). \tag{4.48}$$

In its repetitive form, it follows

$$\hat{f}_n(\Theta) = \frac{\left[\mathcal{L}(d(\mathring{t}),\Theta)f_{n-1}(\Theta)\right]^{\Lambda_n}[\bar{f}(\Theta)]^{1-\Lambda_n}}{\int\left[\mathcal{L}(d(\mathring{t})\Theta)f(\Theta)\right]^{\Lambda}[\bar{f}(\Theta)]^{1-\Lambda}\,d\Theta} \propto \left[\mathcal{L}(d(\mathring{t}),\Theta)f_{n-1}(\Theta)\right]^{\Lambda_n}[\bar{f}(\Theta)]^{1-\Lambda_n}. \tag{4.49}$$

Here the non-negative power $\Lambda_n \in (0,1)$ is called *flattening rate*. The examination on the influence of Λ_n and the specification of their value by means of asymptotic analysis can be found in [35].

4.3.5 Iterative and Batch QB Estimations of ARX Mixtures

After the discussion on iterative construction of prior pdf with flattening in previous section, here we shall illustrate iterative and batch uses of QB Method on the estimation of normal AR(X) mixtures. The purpose of this section is threefold:

- *To illustrate iterative use of the QB method with flattening.*

  Iterations and therefore flattening are extensively used in the QB estimation of mixtures. In the design of the QB method, it is assumed that the estimate of mixture parameters Θ is of a product form

$$f(\Theta|d(t)) \propto \prod_{p\in p^*} f(\Theta_p|d(t))Di_{\alpha}(\kappa_{p;t}).$$

The KL distance of a pair of such products is simply sum of KL distances between the corresponding marginal pdfs creating the product. Consequently, the discussion of last section can be applied to each of them with its specific weight and thus its specific Λ_n.

- *To illustrate hybrid batch use of the QB Method.*

  The construction of the QB estimation implies that its results depend on the order in which data are processed. Experiments have shown that this dependence may be significant. As a hybrid of the quasi-Bayes and EM algorithms, an algorithm called *batch QB estimation* (BQB) can be used to avoids this drawback.

  The BQB estimation is used in iterative form with a very slow convergence. Compared to iterative QB estimation, the difference is that in each iteration only new estimate of δ_{p,p_t} is updated while the other parameters preserve their values from previous iteration without updating. Essentially, it estimates pointers to components within n-th iteration by using approximation of pdfs of the mixture parameters that is fixed within the n-th stage of the iterative search for the posterior pdf.

- *To specify the QB method on the estimation of normal AR(X) mixtures.*

  If each parametric component in mixture is specified as a normal regression, it gives a normal ARX finite mixture. So far, this type mixtures has become the most popular option in the applications of mixtures. Estimation normal AR(X) mixtures is a typical direct application of the QB method with the consideration on the Gauss-inverse-Wishart (GiW) conjugate pdf of each individual components and numerically stable learning algorithms based on $L'DL$ decomposition.

Algorithm 4.3.2 (Iterative QB estimation algorithm of normal ARX mixtures)

Initial (off line) mode

- *Select the stopping criterions, such as monitoring the increment of log-likelihoods with a selected tolerance and checking the number of iterations*
  - *Set a number N, the maximum number of iterations which is allowed in the iterative estimation.*
  - *Set the iteration counter $n := 0$;*
- *Select the structure of the mixture to specify the number of components n_p and the structure of each component. The structure of each component here is determined by the structure of the corresponding data vector Ψ_p.*
- *Select the prior pdfs $f(\Theta_p)$ of the individual parameterized components in the conjugate GiW form,*

$$f(\Theta_p) = GiW_{\theta_p,r_p}(V_{pn;0}, \nu_{pn;0}) \equiv GiW_{\theta_p,r_p}(L_{pn;0}, D_{pn;0}, \nu_{pn;0})$$

  and specify the initial values $L_{pn;0}, D_{pn;0}, \nu_{pn;0}$,

- *Select prior pdfs of component weights $\alpha = [\alpha_1, \cdots, \alpha_{n_p}]$ in the form of Dirichlet,*

$$f(\alpha) = Di_\alpha(\kappa)$$

  and specify the initial values $\kappa_{p;0} > 0$.

- *Select forgetting factor $\lambda \in (0,1]$, a set of alternative pdfs $f_A(\Theta_p)$ and $f_A(\alpha)$ for stabilized forgetting.*
- *Select a set of pre-prior pdfs, $\bar{f}(\Theta_p)$ and $\bar{f}(\alpha)$ to serve for flattening.*

Iterative mode, *running for $n = 1, \ldots, N$.*

1. *Use the current prior pdf $f_n(\Theta) = Di_\alpha(\kappa_n) \prod_{p \in p^*} GiW_{\theta_p, r_p}(L_{pn;0}, D_{pn;0}, \nu_{pn;0})$ in the following evaluations.*
2. *Set the likelihood value of the mixture, often natural logarithm likelihood (log-likelihood) is used, $l_{n;0} = 0$.*

   Sequential mode, *running for $t = 1, 2, \ldots, \mathring{t}$.*

   (a) *Compute the point estimates of the components weight*

   $$\hat{\alpha}_{pn;t-1} = \frac{\kappa_{pn;t-1}}{\sum_{\tilde{p} \in p^*} \kappa_{\tilde{p}n;t-1}}.$$

   (b) *Acquire the data record d_t.*

   (c) *Calculate the value of the predictive pdf for each individual component, $p \in p^*$.*

   - *Perform, for each individual component, a* trial updating *of the statistics by the data vector $\Psi_{pn;t}$.*

   $$\begin{aligned} V_{pn;t} &= V_{pn;t-1} + \Psi_{pn;t}\Psi'_{pn;t}, \\ \nu_{pn;t} &= \nu_{pn;t-1} + 1. \end{aligned}$$

   Note that here this updating uses neither stabilized forgetting nor the date weighting. The partitioned $L'DL$ decomposition of extended information matrix $V_{p;t}$ are updated and used.

   - *Determine the values the predictive pdfs by means of the above trial updates and the formula (4.31)*

   $$f(y_t|u_t, d(t-1), p) = \frac{(2\pi)^{-0.5}\mathcal{I}(L_{pn;t}, D_{pn;t}, \nu_{pn;t})}{\mathcal{I}(L_{pn;t-1}, D_{pn;t-1}, \nu_{pn;t-1})}.$$

   (d) *Update log-likelihood value of the mixture*

   $$l_{n;t} = l_{n;t-1} + \ln\left(\sum_{p \in p^*} \hat{\alpha}_{pn,t-1} f_n(y_t|u_t, d(t-1), p)\right).$$

   (e) *Compute the probabilistic weights $w_{pn;t}$,*

   $$w_{pn;t} = \frac{\hat{\alpha}_{pn;t-1} f(y_t|u_t, d(t-1), p)}{\sum_{\tilde{p} \in p^*} \hat{\alpha}_{\tilde{p}n;t-1} f(y_t|u_t, d(t-1), \tilde{p})}, \; p \in p^*.$$

   (f) *Update the scalars of the Dirichlet pdf*

   $$\kappa_{pn;t} = \kappa_{pn;t-1} + w_{pn;t}, \quad p \in p^*.$$

(g) Update Bayesian parameter estimates of different components, i.e., update the corresponding statistics

$$L_{pn;t-1}, D_{pn;t-1}, \nu_{pn;t-1} \rightarrow L_{pn;t}, D_{pn;t}, \nu_{pn;t}.$$

The updating is equivalent to the weighted version of (4.28)-(4.29)

$$V_{pn;t} = \lambda(V_{pn;t-1} + w_{pn;t}\Psi_{pn;t}\Psi'_{pn;t}) + (1-\lambda)V_{Apn;t}, \quad (4.50)$$
$$\nu_{pn;t} = \lambda(\nu_{pn;t-1} + w_{pn;t}) + (1-\lambda)\nu_{Apn;t}, \quad (4.51)$$

Here λ is the forgetting factor and the statistics with the index of "A" describe the alternative pdf f_A for stabilized forgetting.

Note that in the mixture context, slightly different from the LD updating of single regression model of Section 4.2.2, a recursion of $V = \lambda V + \beta\Psi\Psi'$ type is used with the positive scalar β being determined by both the forgetting factor λ and the weight $w_{pn;t}$.

(h) Evaluate, if need be, characteristics of the pdf $f(\Theta_{pn}|d(t))$ describing other parameters Θ_{pn}.

(i) Go to the beginning of Sequential mode *while data are available, while $t \leq \mathring{t}$.*

3. *Stop, if $n \geq \mathring{n}$ or if the increment of log-likelihood $l_{n;t}$ is smaller than the given tolerance. Otherwise,*

   - *Select a flattening rate Λ_n and apply flattening operation on $f(\Theta_{pn}|d(\mathring{t}))$ and $f(\alpha_{pn}|d(\mathring{t}))$ according to (4.49) for each individual components.*
   - *Increase the iteration counter $n := n+1$, set $\bar{l} = l_{n;\mathring{t}}$, and go to the beginning of* Iterative mode.

Remarks 4.3.2

1. *The predictive pdf could also be computed in its Student form. It makes sense only when prediction errors are explicitly needed.*
2. *Although it may be worth to consider stabilized forgetting on the statistics $\kappa_{c;t}$ with its specific forgetting factor and alternative, this possibility is not currently used in our discussion.*

The corresponding processing-order-independent *batch QB estimation* algorithm of normal AR(X) mixtures can be described as follows. This algorithm uses Bayesian predictors for estimating the Kronecker symbol δ_{p,p_t}. They, among other, respect uncertainty of the current estimates of unknown parameters. Predictions become too cautious if this uncertainty is too high. It may break down the algorithm completely. Knowing it, the remedy is simple. Essentially, predictions used in the EM algorithm that ignore these uncertainties have to be used in several initial iterative steps of the algorithm.

Algorithm 4.3.3 (Batch QB estimation algorithm of ARX mixture)

Initial mode Initial (off line) mode

- *Select the stopping criterions, such as monitoring the increment of log-likelihoods with a selected tolerance and checking the number of iterations*
  - *Set a number N, the maximum number of iterations which is allowed in the iterative estimation.*
  - *Set the iteration counter $n := 0$.*
- *Select the structure of the mixture to specify the number of components n_p. The structure of each component is determined by the structure of the corresponding data vector Ψ_p.*
- *Select the prior pdfs $f(\Theta_p)$ of the individual parameterized components in the conjugate GiW form,*

$$f(\Theta_p) = GiW_{\theta_p,r_p}(V_{pn;0}, \nu_{pn;0}) \equiv GiW_{\theta_p,r_p}(L_{pn;0}, D_{pn;0}, \nu_{pn;0})$$

  and specify the initial values $L_{pn;0}, D_{pn;0}, \nu_{pn;0}$.
- *Select prior pdfs of component weights $\alpha = [\alpha_1, \cdots, \alpha_{n_p}]$ in the form of Dirichlet,*

$$f(\alpha) = Di_\alpha(\kappa)$$

  and specify the initial values $\kappa_{p;0} > 0$.
- *Select forgetting factor $\lambda \in (0, 1]$, a set of alternative pdfs $f_A(\Theta_p)$ and $f_A(\alpha)$ for stabilized forgetting.*
- *Select another set pre-prior pdfs, $\bar{f}(\Theta_p)$ and $\bar{f}(\alpha)$ to serve for flattening.*
- *Make copies $L_{p;0} = L_{pn}$, $D_{p;0} = D_{pn}$, $\nu_{p;0} = \nu_{pn}$ and $\kappa_{p;0} = \kappa_{pn}$.*
- *Set a maximum for the log-likelihood of the mixture $\bar{l} = -\infty$.*

Iterative mode, *running for $n = 1, \ldots, N$.*

1. *Use the current prior pdf $f_n(\Theta) = Di_\alpha(\kappa_n) \prod_{p \in p^*} GiW_{\theta_p,r_p}(L_{pn}, D_{pn}, \nu_{pn})$ in the following evaluations.*
2. *Set the initial log-likelihood of the mixture $l_{n;0} = 0$.*

   Sequential mode, *running for $t = 1, 2, \ldots, \mathring{t}$.*

   (a) *Compute the point estimates of the components weight*

$$\hat{\alpha}_{pn;t-1} = \frac{\kappa_{pn;t-1}}{\sum_{\tilde{p} \in p^*} \kappa_{\tilde{p}n;t-1}}.$$

   (b) *Construct the data vectors $\Psi_{p;t}$.*

   (c) *Compute the values of the predictive pdfs for components, with the data vectors $\Psi_{p;t}$*

$$f_n(y_t|\psi_{p;t}, p) = \int f(y_t|u_t, d(t-1), \Theta_p, p) f_n(\Theta_p)\, d\Theta_p$$

   where the prior pdfs of the components $GiW_{\theta_p,r_p}(L_{pn}, D_{pn}, \nu_{pn}), p \in p^*$ are constant during the time cycle.

(d) Update the log-likelihood of the mixture

$$l_{n;t} = l_{n;t-1} + \ln\left(\sum_{p\in p^*} \hat{\alpha}_{pn;t-1} f_n(y_t|u_t, d(t-1), p)\right).$$

(e) Compute the probabilistic weights $w_{pn;t}$ *to approximate* δ_{p,p_t} *as*

$$w_{pn;t} = \frac{\hat{\alpha}_{pn;t-1} f_n(y_t|u_t, d(t-1), p)}{\sum_{\tilde{p}\in p^*} \hat{\alpha}_{\tilde{p}n;t-1} f_n(y_t|u_t, d(t-1), \tilde{p})}.$$

(f) Update the statistics determining the posterior pdfs, evolving from copies of the prior statistics $L_{p;0}$, $D_{p;0}$, $\nu_{p;0}$ *and* $\kappa_{p;0}$

$$\begin{aligned} L'_{pn;t}D_{pn;t}L_{pn;t} &= \lambda(L'_{pn;t-1}D_{pn;t-1}L_{pn;t-1} + w_{pn;t}\Psi_{p;t}\Psi'_{p;t}) + (1-\lambda)L'_{Apn;t}D_{Apn;t}L_{Apn;t} \\ \nu_{pn;t} &= \lambda(\nu_{pn;t-1} + w_{pn;t}) + (1-\lambda)\nu_{Apn;t}, \\ \kappa_{p;t} &= \kappa_{p;t-1} + w_{pn;t}. \end{aligned}$$

Here the updating is date weighted. λ *is the forgetting factor and the statistics with the index of "A" describe the alternative pdf* f_A *for stabilized forgetting.*

(g) Go to the beginning of Sequential mode *if* $t \leq \mathring{t}$. *Otherwise continue.*

3. *Stop, if* $n \geq \mathring{n}$, *or if the log-likelihood of the mixture does not increase* $l_{n;\mathring{t}} < \bar{l}$ *or the increment of likelihood is smaller than the given tolerance.*

   Otherwise,

   - *Apply flattening operation on* $f(\Theta_{pn}|d(\mathring{t}))$ *and* $f(\alpha_n|d(\mathring{t}))$.
   - *Increase the iteration counter* $n := n+1$, *set* $\bar{l} = l_{n;\mathring{t}}$, *and go to the beginning of* Iterative mode.

Chapter 5

Bayesian Solution to ARMAX Models with Known MA Part

In the last chapter, we recalled the existing Bayesian solutions of ARX model and its finite mixtures. From now on in the following chapters, we come to discuss ARMA(X) model and two types of its finite mixtures. This chapter recalls and reexamines Bayesian estimation and prediction of ARMA(X) with known MA part, or more exactly the corresponding LD filters. It makes the basis for the remaining discussion of later chapters.

Consider a SO normal ARMAX model in a regression form describing the relationships of a (possibly multi-variate) external input u_t to a scalar output y_t,

$$y_t = \theta' \psi_t + v_t, \tag{5.1}$$

where $'$ denotes transposition. The n_ψ-dimensional *regressor* ψ_t is a known function of u_t and the past data $d(t-1)$. The unknown parameter set θ describes the deterministic part of the model, it is a vector with its dimension identical to n_ψ. $v_t = e_t + \sum_{i=1}^{n_c} c_i e_{t-i}$ is a *colored* stationary noise of order n_c, which is usually interpreted as a moving average defined on a sequence of mutually un-correlated, zero-mean, Gaussian noise $\{e_t\}$. Thus its covariances

$$\begin{aligned} \mathcal{E}[v_t v_{t-i}] &= r_i, \quad for \quad i = 0, 1, ..., n_c, \\ &= 0, \quad for \quad |i| > n_c. \end{aligned}$$

fulfill the following relation with the noise variance r_e,

$$r_i = r_e \sum_{k=i}^{n_c} c_i c_{k-i}, \quad \text{with} \quad c_0 = 1.$$

The model is generally parameterized through the parameter set

$$\Theta \;=\; \{\Theta_a,\; C\},$$

with the AR(X) part of the model described by $\Theta_a = (\theta, r_e)$ and the C-parameters of the model are represented by an n_c-dimensional vector $C = [c_1, \cdots, c_{n_c}]$.

It is well-known that there are some close relationships between ARMAX and ARX under some conditions. Here we consider two of such cases:

- When we know the stochastic MA noise part of the model up to its covariances, either $r_i,\ \ i = 1, ..., n_c$ are directly known or both r_e and the C-parameters C are known.

- When we know the MA part only up to its C-parameters. It means the noise variance r_e is unknown and has to be estimated.

The discussion on Bayesian solutions of the ARMA(X) under the above two situations forms the content of this chapter. The emphasis is put on the reexamination of the case where the MA part is known only up to its C-parameters.

As mentioned in chapter 3, Delta models, an alternatively form of the ARMA models, may appear to be numerically more robust and therefore more suitable for real-time computation in adaptive and self-tuning control. The discussion of this chapter is based on ARMAX models. However, it can be easily extended to the case of Delta description when necessary.

5.1 Known MA Part up to Noise Covariances

Under the assumption that the covariance matrix is known, Peterka [4] in 1981 showed that LD factorization of the known noise correlation matrix of the model acts as an optimal, time-varying, pre-whitening filter on the observed data. These results are recalled in this section.

5.1.1 LD Filter

Assume we know the covariances $r_i, \quad i = 1, ..., n_c$ directly or we know both noise variance r_e and C-parameters, then, the ARMAX model (5.1) now is parameterized only by

$$\Theta = \theta \tag{5.2}$$

and the covariance matrix Ω of the noise v_t is Toeplitz matrix with its i-th sub- and super-diagonals satisfying

$$\Omega_{t,t\pm i} \quad = \quad r_i \quad \text{or} \quad \Omega_{t,t\pm i} = r_e \sum_{k=i}^{n_c} c_i c_{k-i}, \qquad \text{for} \quad i = 0, 1, \cdots, n_c \quad \text{and} \quad c_0 = 1,$$

Let us consider its $L'DL$ *decomposition*

$$\Omega = LDL',$$

where $D = \text{diag}[D_1, D_2, \ldots, D_t]$ is a diagonal matrix with positive diagonal elements D_τ, $\tau = 1, \ldots, t$, and L is a lower triangular matrix of the form

$$L = \begin{bmatrix} 1 & & & & & \\ L_{2,1} & 1 & & & & \\ \vdots & \ddots & \ddots & & & \\ L_{n_c-1,n_c} & \cdots & L_{n_c-1,1} & 1 & & \\ & \ddots & \cdots & \ddots & \ddots & \\ & & L_{t,n_c} & \cdots & L_{t,1} & 1 \end{bmatrix}$$

with its zero entries represented by the empty entries. The non-zero elements of L, D matrices can be evaluated recursively as follows

$$D_1 = r_0, \quad L_{2,1} = r_1 D_1^{-1}, \quad D_2 = r_0 - L_{2,1}^2 D_1 \tag{5.3}$$

and for $\tau = 3, 4, ..,$ and $i = n_c, n_c - 1, ..., 1$ with $n = \min(n_c, \tau)$

$$L_{\tau,i} = \left(r_i - \sum_{k=i+1}^{n} L_{\tau,k} D_{\tau-k} L_{\tau-i,k-i} \right) D_{\tau-i}^{-1}, \quad D_\tau = r_0 - \sum_{k=i+1}^{n} L_{\tau,k} D_{\tau-k} L_{\tau,k}. \tag{5.4}$$

these elements are the functions of either C-parameters and noise variance r_e or noise covariances r_i.

Using the above LD decomposition, Peterka showed the following relationships between ARMAX model and ARX model.

Proposition 5.1.1 (Relationships of ARMAX with known covariances to ARX) *Consider a normal ARMAX model, if its noise covariance matrix is known, then the model is parameterized only by unknown parameters $\Theta = \theta$ and the conditional* probability density function *of its output y_t can be expressed by*

$$f(y_t|u_t, d(t-1), \theta) = (2\pi D_t)^{-0.5} \exp\left[-\frac{(y_t - \mu_t)^2}{2D_t} \right] = \mathcal{N}_{y_t}(\mu_t, D_t) \tag{5.5}$$

with

$$\mu_t = \theta' \tilde{\psi}_t + \sum_{i=1}^{n} L_{t,i} \tilde{y}_{t-i}, \quad n = \min(n_c, t).$$

The filtered data vector $\tilde{\Psi}_t = [\tilde{y}_t, \tilde{\psi}_t']'$ is obtained by passing the observed data vector $\Psi_t = [y_t, \psi_t']'$ through the the pre-whitening filter

$$\tilde{y}_t + \sum_{i=1}^{n} L_{t,i} \tilde{y}_{t-i} = y_t, \qquad \tilde{y}_1 = y_1, \tag{5.6}$$

$$\tilde{\psi}_t + \sum_{i=1}^{n} L_{t,i} \tilde{\psi}_{t-i} = \psi_t, \qquad \tilde{\psi}_1 = \psi_1. \tag{5.7}$$

where the filter is determined by the L, D entries (5.3)–(5.4).

5.1.2 Estimation and Prediction

Since the covariances are known, a sufficient statistic of fixed dimension can be found both for the estimated parameters and the predicted output. Hence its exact Bayesian solution of real-time parameter estimation and prediction, possibly controlled in closed adaptive control loop, can be obtained without loss of information. To apply the general Bayesian estimation and prediction Proposition 2.1.3 on (5.5), Peterka proved the following proposition.

Proposition 5.1.2 (Bayesian estimation and prediction of ARMAX with known covariances) *Under natural conditions of control, see Definition 2.1.1, consider a normal ARMAX model. If its covariances are known so that the model is parameterized only by $\Theta = \theta$, and if the prior pdf of the unknown parameter θ is assumed in the normal form*

$$f(\theta|d(t-1)) \sim \mathcal{N}_\theta(\hat{\theta}_{t-1}, R_{\theta;t-1}),$$

then the normality is reproduced, so that

$$\begin{aligned} f(y_t|u_t, d(t-1)) &\sim \mathcal{N}_{y_t}(\hat{y}_t, R_{y;t}), \\ f(\theta|d(t)) &\sim \mathcal{N}_\theta(\hat{\theta}_t, R_{\theta;t}). \end{aligned}$$

Here the following algebraic recursion holds for the conditional expectations $\hat{y}_t$ *and the covariances* $R_{y;t}$, $R_{\theta;t}$

$$\begin{aligned}
\hat{y}_t &= \tilde{\psi}_t'\hat{\theta}_{t-1} + \sum_{i=1}^{n} L_{t,i}\tilde{y}_{t-i}, \quad n = \min(n_c, t) && (5.8)\\
R_{y;t} &= D_t + \tilde{\psi}_t' R_{\theta;t-1}\tilde{\psi}_t && (5.9)\\
\hat{\theta}_t &= \hat{\theta}_{t-1} + K_t(\tilde{y}_t - \tilde{\psi}_t'\hat{\theta}_{t-1}) && (5.10)\\
K_t &= R_{\theta;t-1}\tilde{\psi}_t R_{y;t}^{-1} = R_{\theta;t}\tilde{\psi}_t' D_t^{-1} && (5.11)\\
R_{\theta;t} &= R_{\theta;t-1} - R_{\theta;t-1}\tilde{\psi}_t R_{y;t}^{-1}\tilde{\psi}_t R_{\theta;t-1} && (5.12)
\end{aligned}$$

Thus, the overall algorithm for the real-time estimation of the parameter θ and for the one-step-ahead prediction of output y_t is obtained when the recursions (5.8)–(5.12) are supplemented with the recursive updating of the L, D entries (5.3)–(5.4) and the data filtering by (5.6)–(5.7).

5.2 Known MA Part up to C-parameters

Now let us consider a more general situation where only C-parameters of the ARMA(X) model are known. Can we still achieve the optimal prefiltering using LD factorization?

With an introduced canonical state-space representation of ARMA model (Delta model), Peterka developed the algorithm for recursive joint estimation of the rest unknown parameters and the state in [6, 7]. To address the problem of estimating/searching unknown C-parameters later, it may be more convenient for us to reexamine the results here based on regression form of ARMA model and to clarify a few related issues.

5.2.1 Extended LD Filter

If the MA part of the model is known only up to its C-parameters $C = [c_1, \cdots, c_{n_c}]$, then the noise variance r_e has to be estimated and an ARMAX model is parameterized by a pair

$$\Theta_a = (\theta, r_e). \tag{5.13}$$

Meanwhile, we find it is convenient to introduce a matrix S as

$$S_{t,t\pm i} = s_i, \qquad \text{for} \quad i = 0, 1, \cdots, n_c \tag{5.14}$$

with $s_i = \sum_{k=i}^{n_c} c_i c_{k-i}$. This leads to

$$\Omega_{t,t\pm i} = r_e s_i \qquad \text{for} \quad i = 0, 1, \cdots, n_c$$

here the relation $r_i = r_e s_i$ is used.

Consequently, the covariance matrix Ω can be re-written in term of S

$$\Omega = r_e S.$$

Clearly, when only C-parameters are known, we know the covariance matrix Ω only partially up to the matrix S. For this reason we call it *partial covariance matrix*.

Let us apply LDL' *decomposition* to this partial covariance matrix S

$$S = LDL',$$

where L is a lower triangular matrix and $D = \text{diag}[D_1, D_2, \ldots, D_t]$ is a diagonal matrix with positive diagonal elements D_τ, $\tau = 1, \ldots, t$. The non-zero elements of L, D matrices are the functions of C-parameters and can be evaluated recursively in a similar way as that of the previous section

$$D_1 = s_0, \quad L_{2,1} = s_1 D_1^{-1}, \quad D_2 = s_0 - L_{2,1}^2 D_1 \tag{5.15}$$

and for $\tau = 3, 4, ..,$ and $i = n_c, n_c - 1, ..., 1$ with $n = \min(n_c, \tau)$

$$L_{\tau,i} = \left(s_i - \sum_{k=i+1}^{n} L_{\tau,k} D_{\tau-k} L_{\tau-i,k-i} \right) D_{\tau-i}^{-1}, \quad D_\tau = s_0 - \sum_{k=i+1}^{n} L_{\tau,k} D_{\tau-k} L_{\tau,k}. \tag{5.16}$$

Note that in contrast to the last section where the elements of the filter are determined by the covariances, here they are functions of only the C-parameters.

With this decomposition, the similar derivation to that of the last section leads to the conclusion that the conditional pdf of output y_t is a normal one like (5.5) with some slight changes.

Proposition 5.2.1 (Relationship of ARMAX with known C-parameters to ARX) *Consider a normal ARMAX model. If only C-parameters of its color noise are known, then the model is parameterized by $\Theta_a = (\theta, r_e)$ and the conditional* probability density function *(pdf) of its output y_t can be expressed in a normal form as follows*

$$f(y_t | u_t, d(t-1), \Theta_a) = (2\pi r_e D_t)^{-0.5} \exp\left[-\frac{(y_t - \mu_t)^2}{2 r_e D_t} \right] = \mathcal{N}_{y_t}(\mu_t, r_e D_t), \tag{5.17}$$

where

$$\mu_t = \theta' \tilde{\psi}_t + \sum_{i=1}^{n} L_{t,i} \tilde{y}_{t-i},$$

with $n = \min(n_c, t)$ and

$$\begin{aligned} \tilde{y}_t + \sum_{i=1}^{n} L_{t,i} \tilde{y}_{t-i} &= y_t, \quad \tilde{y}_1 = y_1, \\ \tilde{\psi}_t + \sum_{i=1}^{n} L_{t,i} \tilde{\psi}_{t-i} &= \psi_t, \quad \tilde{\psi}_1 = \psi_1. \end{aligned} \tag{5.18}$$

where the filtered ARX model is parameterized by the unknown parameters $\Theta_a = (\theta, r_e)$ and acts on the filtered data vector $\tilde{\Psi}_t = [\tilde{y}_t, \tilde{\psi}_t']'$. Note that the entries of L, D now are determined by the LDL' decomposition *of the partial covariance matrix S.*

5.2.2 Estimation and Prediction

If we define the filtered data $\tilde{\Psi}$ after a scaling as follows

$$\tilde{\tilde{\Psi}}_t = \left[\frac{\tilde{y}_t}{\sqrt{D_t}} \quad \frac{\tilde{\psi}'_t}{\sqrt{D_t}} \right]', \tag{5.19}$$

then, the conditional *probability density function* of output y_t (5.17) can be equivalently reexpressed in terms of $\tilde{\tilde{\Psi}}_t$

$$f(y_t|u_t, d(t-1), \Theta_a) = (2\pi r_e)^{-0.5} D_t^{-0.5} \exp\left\{-\frac{1}{2r_e}\mathrm{tr}\left(\tilde{\tilde{\Psi}}_t \tilde{\tilde{\Psi}}_t'[-1, \theta']'[-1, \theta']\right)\right\}. \quad (5.20)$$

It has the following correspondence to the exponential family (4.7)

$$f(y_t|u_t, d(t-1), \Theta_a) = A(\Theta_a) D(\Psi_t) \exp\left[\langle B(\Psi_t), C(\Theta_a)\rangle\right] \quad (5.21)$$

with

$$\begin{aligned} A(\Theta_a) &\equiv r_e^{-0.5}, & D(\Psi_t) &\equiv (2\pi D_t)^{-0.5}, \\ B(\Psi_t) &\equiv \tilde{\tilde{\Psi}}_t \tilde{\tilde{\Psi}}_t', & C(\Theta_a) &\equiv 2r_e^{-1}[-1, \theta']'[-1, \theta'], \qquad \langle B, C\rangle \equiv \mathrm{tr}[B'C]. \end{aligned}$$

Here $B(\Psi_t)$ is a function of Ψ_t and indirectly expressed in term of $\tilde{\tilde{\Psi}}_t$. $D_t^{-0.5}$ is a function of known C-parameter and determined by the LD factorization of the known partial covariance matrix. Strictly speaking, such a time-varying variable is not a standard item of $D(\Psi_t)$ in our definition of the exponential family (4.1.1). It, however, is important to note that $D_t^{-0.5}$ will make no influence on the parameter estimation and will enter the prediction independently without involving in the rest calculation. This observation suggests that we can still select a conjugate prior for the parameters $\Theta_a = (r_e,\ \theta)$ in the form of GiW pdf (4.17).

Obviously, applying the above proposition into the Proposition 4.1.2 of estimation and prediction in exponential family, it follows

Proposition 5.2.2 (Bayesian estimation and prediction of ARMAX with known C-parameters) *Under natural conditions of control, Definition 2.1.1, consider the normal ARMAX model. If we assume that the C-parameters are known, then the model is parameterized by $\Theta_a = (\theta, r_e)$. If the extended LD filter (5.15)-(5.16) is used:*

i) The conditional probability density function *(pdf) of output y_t is equivalently expressed as*

$$f(y_t|u_t, d(t-1), \Theta_a) = (2\pi r_e)^{-0.5} D_t^{-0.5} \exp\left\{-\frac{1}{2r_e}\mathrm{tr}\left(\tilde{\tilde{\Psi}}_t \tilde{\tilde{\Psi}}_t'[-1, \theta']'[-1, \theta']\right)\right\},$$

and a conjugate prior and a conjugate alternative can be selected in the GiW form (4.17)

$$f(\Theta) = GiW_\Theta(V_0, \nu_0) \quad \textit{and} \quad f_A(\Theta) = GiW_\Theta(V_{A;t}, \nu_{A;t})$$

with a forgetting rate λ (see Section 4.1.1).

ii) The Bayesian posterior pdf of parameter then remains in the GiW form $GiW_\Theta(V_t, \nu_t)$, with its sufficient statistics evolving according to the recursions

$$V_t = \lambda\left(V_{t-1} + \tilde{\tilde{\Psi}}_t \tilde{\tilde{\Psi}}_t'\right) + (1-\lambda)V_{A;t}, \qquad V_0 \textit{ given} \quad (5.22)$$

$$\nu_t = \lambda(\nu_{t-1} + 1) + (1-\lambda)\nu_{A;t}, \qquad \nu_0 > 0 \textit{ given.} \quad (5.23)$$

iii) The values of predictive pdf *are evaluated as the ratio*

$$f(y_t|\psi_t, d(t-1)) = \frac{(2\pi)^{-0.5} D_t^{-0.5}\, \mathcal{I}\left(V_{t-1} + \tilde{\tilde{\Psi}}_t \tilde{\tilde{\Psi}}_t', \nu_{t-1} + 1\right)}{\mathcal{I}(V_{t-1}, \nu_{t-1})}, \quad (5.24)$$

where the integrals $\mathcal{I}(V,\nu) = \mathcal{I}(L,D,\nu)$ can be computed by the formula (4.23) with the filtered data $\tilde{\tilde{\Psi}}_t$. D_t is is the function of the C-parameters and determined by the LDL' decomposition of the partial covariance S.

One-step prediction of the output y_t can be evaluated directly by the formula

$$\hat{y}_t \;=\; \hat{\theta}'\tilde{\psi}_t + \sum_{i=1}^{n} L_{t,i}\tilde{y}_{t-i}, \qquad n = \min(n_c, t),$$

where $L_{t,i}$ is the the function of the C-parameters and determined by the LDL' decomposition of the partial covariance S.

Note that here two kinds of LD decomposition are involved in the estimation and prediction: One is the $L'DL$ decomposition of extended information matrix V to avoid the induced numerical troubles in the estimation, while the other one is the LDL' decomposition of the partial covariance matrix S to prefilter data.

In summary, compared to the last section, our extension here is able to cope with a more general case by estimating the unknown noise variance as well. Meanwhile, a wider set of unknown parameters including noise variance leads to a different choice of prior from that of the last section.

5.3 Some Properties of LD type Filters

In the previous sections, we described the LD type filters to provide real-time Bayesian estimation of the ARMA(X) model when the MA part is known. Here we discuss some of their properties which are vital when addressing the case with unknown C-parameters later on.

As we have showed above, LD factorization of the known (patrial) covariance matrix acts as an optimal, time-varying, pre-whitening filter on the observed data:

- The main power of the LD filters is that they impose no restriction on the stability of the C-polynomial.
  - The time-evolution of the LD type filters actually provides a sort of spectral factorization of the polynomial product

    $$C(z) r_e C'(z^{-1}),$$

    where $C(z^{-1}) = 1 + c_1 z^{-1} + \cdots + c_n z^{-n_c}$ is the C-polynomial in the backward shift operator z^{-1}. Thus the pre-whitening made by this filter does not depend on the stability of the polynomial $C(z^{-1})$ so that it is able to cope with all kinds of roots: strictly stable, close to the boundary, at the stability boundary and strictly unstable. The factorization converges relatively fast. [36] showed that if there are no roots on the unit circle, it converges geometrically.

    The often used filters, such as a stable inverse (spectral factor) imposes stability restrictions. For a given canonical state representation, Peterka showed that the estimation of main parameters and the state is algorithmically simpler and numerically more robust than the standard Kalman filter.

    It is showed by Aasnaes and Kailath [37] that the often used predictor with $L_{t,i} = L_{\infty,i}$ doesn't produce optimal predictions (not even asymptotically) if the roots of the polynomial $C(z^{-1})$ lie on the unit circle and the convergence to the optimality may be very slow if the roots are close to the unit circle.

- For a given C-polynomial $C(z^{-1})$, for growing t,
  - the diagonal elements of D converge monotonically. One of our interesting observation of the convergence of D_t is that: $D_\infty = 1$ as long as all roots strictly insides the unit circle, otherwise $D_\infty \geq 1$.
  - the time-varying entries $L_{t,i}$ of the triangular matrix L (5.1.1) converge

    $$L_{t,i} \to \tilde{c}_i, \quad t \to \infty, \quad i = 1, \cdots, n_c$$

    these limits define a polynomial $\tilde{C}$

    $$\tilde{C}(z^{-1}) = 1 + \tilde{c}_1 z^{-1} + \cdots + \tilde{c}_{n_c} z^{-n_c}.$$

    If the given C-polynomial $C(z^{-1})$ has all its roots outsides or on the unit circle, the relation between it and $\tilde{C}(z^{-1})$ holds

    $$\tilde{C}(z^{-1}) = \frac{1}{c_0} C(z^{-1})$$

    when there are some roots lying on the unit circle, this convergence goes from the outsides of the unit circle. If the given C-polynomial $C(z^{-1})$ has some roots to be insides the unite circle, then

    $\tilde{C}(z^{-1})$ *is the stable reflection of* $C(z^{-1})$ *with the guaranteed stability*

    Thus asymptotically for $t \to \infty$, the LD factorization produces the coefficients of the stable reflection of the C-polynomial and ensure the stability of the optimal filters even when this C-polynomial has some unstable roots.

- The uncertainties of unknown parameters are under the control, Bayesian structure estimation of the ARX part can be used [5], etc.

- The evaluation of the filter is computationally inexpensive.

  In SO cases, it is sufficient to only store $n_c^2 + 2n_c + 1$ scalars at each time instant in the recursive evaluations of the extended LD filter, namely $n_c^2 + n_c$ elements for the lower triangular matrix L and $n_c + 1$ diagonal elements for the matrix D.

- Real-time Bayesian estimation of the ARX part can be provided using the filtered data.

  One important fact has to be emphasized here is that:

  Consider two situations in the estimation, one case where the given C-polynomial has some unstable roots while alternatively when it is its stable reflection that is given.

  Then, as showed earlier, the obtained estimates of noise covariances r_i are finally identical in both cases. But they do produce different estimates of noise variance r_e.

  Knowing these facts helps us to avoid unnecessary difficulties in general cases. However, we shall show one case later where the difficulties created by such a simple difference may become not trivial any more. It is detailed studied later in Chapter 6 and Chapter 7.

- The C-parameters of MA term have to be considered as time varying, i.e., the filter is time-varying, even when the covariances are time-invariant. The variations are data independent and driven only by the time-invariant C-parameters. This variations hinder the attempts to estimate the unknown C-parameters recursively. Moreover, they make evaluation of derivatives of the related likelihood function at least impractical.

Chapter 6

Bayesain Solutions and Modelling Properties of ARMAX Mixtures

The optimal LD filters, as shown in the last chapter, lead to a real-time Bayesian estimation of the AR(X) part and imposes no restriction on the stability of the MA part, if an ARMA(X) is known up to its MA Part. Based on the extended LD filter, this chapter presents an investigation on the two types ARMA(X) finite mixtures, which were introduced and described in Chapter 3:

The MARMAX model (3.20), a natural mixture generalization of the linear ARMA(X) with each of its component $f(y_t|u_t, d(t-1), \Theta_a, C_p), p \in p^*$ described by one ARMA(X),

$$f\left(y_t|u_t, d(t-1), \Theta\right) \;=\; \sum_{p=1}^{n_p} \alpha_p f(y_t|u_t, d(t-1), \Theta_{a,p}, C_p, p), \tag{6.1}$$

which is parameterized by the parameter set

$$\Theta \equiv \{\Theta_a,\ \Theta_c,\ \alpha\},$$

with $\Theta_a \equiv \{\Theta_{a,p}\}_{p=1}^{n_p}$, $\Theta_c \equiv \{C_p\}_{p=1}^{n_p}$, $\alpha \equiv [\alpha_1, \cdots, \alpha_{n_p}]$. Here the mixing weights α satisfy $\alpha_p \geq 0,\ p = 1, \cdots, n_p,\ \sum_{p=1}^{n_p} \alpha_p = 1$. The parameters $\Theta_{a,p} = \{\theta_p, r_{e,p}\}$ and $C_p = [c_{p,1}, \cdots, c_{p,n_c}]$ describe the ARX part and the C-parameters of the p-th components, respectively.

The ARMMAX model (3.22), a special mixture of ARMAX having a common deterministic ARX part in all ARMAX components while the characteristics of the stochastic noise parts vary,

$$f\left(y_t|u_t, d(t-1), \Theta\right) \;=\; \sum_{p=1}^{n_p} \alpha_p f(y_t|u_t, d(t-1), \Theta_a, C_p, p), \tag{6.2}$$

which is parameterized by the parameter set

$$\Theta \equiv \{\Theta_a,\ \Theta_c,\ \alpha\}.$$

Here, in contrast to the MARMAX model, all ARX parts of the ARMMAX model are described by a single common parameter set

$$\Theta_a = \{\theta, r_e\}. \tag{6.3}$$

Our discussion focuses on a few properties of these two mixtures and develops their efficient Bayesian estimations and predictions when the C-parameters of the models are known.

6.1 Relationships to ARMAX models

It is no doubt that the MARMAX model is richer than a single ARMAX model. What about then the role of the ARMMAX model in system description?

The ARMMAX model describes well the cases when the common ARX part has a physical meaning of interest while characteristics of the stochastic disturbances may vary. Thus, it is more flexible than a single ARMAX model by providing more freedom in describing stochastic part of the input-output relationship.

It is not only intuitively obvious but also can be shown formally as follows.

Proposition 6.1.1 (Moments of ARMMAX model) *For an ARMMAX model with a given selection of C-parameters $\Theta_c = \{C_p\}_{p\in p^*}$, an equivalent single ARMAX model exists in terms of the first moment. The equivalence does not hold with respect to variance.*

Proof: Results are implied by the mixture definition, the linearity of the expectation and the identity $E\left[y^2\right] = \mathrm{var}[y] + E^2[y]$.

For simplicity, the proof is presented with $n_p = 2$. For a 2-component ARMMAX, the mixing weights are

$$\alpha = [\alpha_1, \alpha_2], \qquad \text{with } \alpha_2 = 1 - \alpha_1,$$

and a pair extended LD filters, determined by $\Theta_c = \{C_1, C_2\}$ and (5.15)-(5.16), generate the filtered data

$$\tilde{\Psi}_{p;t} = [\tilde{y}_t, \tilde{\psi}'_{p;t}]', \qquad p \in \{1, 2\}.$$

Then, the corresponding conditional expectation $E[\cdot|\cdot]$ and variance $\mathrm{var}[\cdot|\cdot]$ of the output y_t are

$$\begin{aligned} E\left[y_t|u_t, d(t-1), \Theta\right] &= \alpha_1\left(\theta'\tilde{\psi}_{1;t} + \Delta_{1;t}\right) + \alpha_2\left(\theta'\tilde{\psi}_{2;t} + \Delta_{2;t}\right) \qquad (6.4)\\ &= \theta'\left(\alpha_1\tilde{\psi}_{1;t} + \alpha_2\tilde{\psi}_{2;t}\right) + \alpha_1\Delta_{1;t} + \alpha_2\Delta_{2;t},\\ \mathrm{var}\left[y_t|u_t, d(t-1), \Theta\right] &= \alpha_1\alpha_2\left[\theta'\left(\tilde{\psi}_{1;t} - \tilde{\psi}_{2;t}\right) + \Delta_{1;t} - \Delta_{2;t}\right]^2 + \qquad (6.5)\\ &\quad + r_e\left(\alpha_1 D_{1;t} + \alpha_2 D_{2;t}\right). \end{aligned}$$

where $\Delta_{p;t} = \sum_{i=1}^{n} L_{t,i}\tilde{y}_{t-i}$, $n = \min(n_c, t)$, $p \in \{1, 2\}$, c.f. Proposition 5.2.1.

Smooth dependence of the filtered data vectors on the Θ_c implies that an equivalent single ARMAX model in terms of the first moment exists with its C-parameters being a convex combination of the C_1 and C_2. Clearly, the equivalence can not be found with respect to variance, since the dependence of the conditional variance on data cannot be neglected. □

6.2 Estimation and Prediction of ARMAX Mixtures with Known C-parameters

The possibility to use the ARMAX mixtures is supported by the LD filters and the QB mixture estimation algorithms described in previous chapters. They throw light on how to estimate the MARMAX and ARMMAX when the C-parameters are known.

Before starting the discussion, it is worthwhile to state an important fact we rely on: If assume

- the normality of the models,

- C-parameters of MA parts $\Theta_p \equiv \{C_p\}_{p\in p^*}$ are fixed.

Then, individual components of the ARMAX mixtures can be pre-whitened and represented by filtered regressions.

Specifically, each component of the MARMAX (3.20), say the p-th component, can be described by the filtered regression

$$f(y_t|u_t, d(t-1), \Theta_{a,p}, C_p, p) \quad = \quad \mathcal{N}_{\tilde{y}_{p;t}}(\theta_p'\tilde{\psi}_{p;t}, r_{e,p}D_{p;t}), \quad p \in p^*,$$

while that of the ARMMAX (3.22) can be described by

$$f(y_t|u_t, d(t-1), \Theta_a, \Theta_c, p) \quad = \quad \mathcal{N}_{\tilde{y}_{p;t}}(\theta'\tilde{\psi}_{p;t}, r_e D_{p;t}), \quad p \in p^*.$$

In both cases, with its advantages stated in the Chapter 5, the extended LD filters are used for the pre-whitening. The filtered data $\tilde{\psi}_{p;t}$ and the LD decomposition factors such as $D_{p;t}$ are functions of the known C-parameters.

With this fact, two extended Quasi-Bayes algorithms are developed here for the estimations of normal MARMAX model and normal ARMMAX model, respectively. For simplicity, we only give the basic description of the algorithms without the consideration of their iterative and batch versions, although they are necessary and important in practical use. The discussion on these two versions QB estimations for ARX mixtures in Section 4.3.5 shows clearly their essence and can be easily extended and applied on the ARMAX mixtures. For the same reason, the algorithms are described to the level of components without the consideration of factors. They could be extended to the treatment of factor level using the complement of Appendix B.

6.2.1 QB Estimation and Prediction of MARMAX Models

Based on the extended LD filters and therefore the filtered regressions, it is easy to extend the QB mixtures estimation of Section 4.3, in particular the QB estimation of normal ARX mixture Section 4.3.5, to provide the estimation of the MARMAX as follows.

Proposition 6.2.1 (QB estimation of MARMAX model with known C-parameters) *Consider the MARMAX model (3.20) and assume that its C-parameters $\Theta_c \equiv \{C_p\}_{p=1}^{n_p}$ are known.*

i) Let us introduce unobserved discrete random pointers $p_t, t \in t^*$ to label the active component at each time instant t and assume it takes the value $p \in p^* \equiv \{1, \ldots, n_p\}$ with the probability α_p.

Then, considering the pointer p_t together with the observed data d_t leads to the joint pdf

$$f(y_t, p_t|u_t, d(t-1), \Theta) = \prod_{p\in p^*} \left[\alpha_p f(y_t|u_t, d(t-1), \Theta_{a,p}, C_p, p)\right]^{\delta_{p,p_t}}, \tag{6.6}$$

with its marginal pdf $f(y_t|u_t, d(t-1), \Theta)$ describing the MARMAX mixture. $\Theta_{a,p} = \{\theta_p, r_{e,p}\}$ describes the unknown AR(X) part of the p-th component in the mixture. δ is the Kronecker *function.*

ii) If normality is assumed, then each of its components can be described by the filtered regression

$$f(y_t|u_t, d(t-1), \Theta_{a,p}, C_p, p) \quad = \quad \mathcal{N}_{\tilde{y}_{p;t}}(\theta_p'\tilde{\psi}_{p;t}, r_{e,p}D_{p;t}), \ p \in p^* \tag{6.7}$$

using the filter (5.15)-(5.16) determined by the LDL' decomposition *of the corresponding known partial covariance matrix. The filtered data vector $\tilde{\Psi}_{p;t}=[\tilde{y}_{p;t}, \tilde{\psi}_{p;t}']'$ is generated by means of*

(5.18). $D_{p;t}$ is the t-th diagonal element of the matrix D_p, here the D_p is the diagonal matrix from the $L'DL$ decomposition *of the patrial covariance matrix of the p-th component.*

Consequently, the joint (6.6) pdf can be specified as

$$f(y_t, p_t|u_t, d(t-1), \Theta) = \prod_{p\in p^*} \left[\alpha_{p_t} \mathcal{N}_{\tilde{y}_{p;t}} \left(\theta'_p \tilde{\psi}_{p;t}, r_{e,p} D_{p;t}\right)\right]^{\delta_{p,p_t}}. \tag{6.8}$$

iii) Under the natural condition of control, Definition 2.1.1, let us take the conjugate priors of the mixing weights $f(\alpha)$ and the parameters of individual components $f(\Theta_{a,p})$ in the Dirichlet *form and the* GiW *form, respectively*

$$\begin{aligned} f(\Theta_{a,p}|d(t-1), C_p) &= GiW_{\Theta_{a,p}}(V_{p;t-1}, \nu_{p;t-1}),\ p \in p^* \\ f(\alpha|d(t-1), \Theta_c) &= Di_\alpha(\kappa_{t-1}). \end{aligned}$$

Moreover, let us assume the pdf *of all unknown parameters be of the product form*

$$f(\Theta_a, \alpha|d(t-1), \Theta_c) \propto \prod_{p\in p^*} GiW_{\Theta_{a,p}}(V_{p;t-1}, \nu_{p;t-1}) Di_\alpha(\kappa_{p;t-1}) \tag{6.9}$$

and approximate the Kronecker δ_{p,p_t} *by its conditional expectation $w_{p;t}$,*

$$w_{p;t} \equiv E\left[\delta_{p,p_t}|d(t), \Theta_c\right] = \frac{\kappa_{p;t-1} f(y_t|u_t, d(t-1), p)}{\sum_{\tilde{p}\in p^*} \kappa_{\tilde{p};t-1} f(y_t|u_t, d(t-1), \tilde{p})},\ p \in p^*. \tag{6.10}$$

Then, a posteriori pdf of the whole mixture in time instant t preserves in the same product form as (6.9). Here a posteriori pdf of its p-th component is expressed by

$$f(\Theta_{a,p}|d(t), \Theta_c) \propto \left[f(y_t|\tilde{\psi}_{p;t}, \Theta_{a,p}, p)\right]^{w_{p;t}} GiW_{\Theta_{a,p}}(V_{t-1}, \nu_{t-1}),\ p \in p^* \tag{6.11}$$

with the associated statistics being updated according to the recursions

$$V_{p;t} = V_{p;t-1} + w_{p;t} \tilde{\tilde{\Psi}}_{p;t} \tilde{\tilde{\Psi}}'_{p;t}, \qquad \nu_{p;t} = \nu_{p;t-1} + w_{p;t}. \tag{6.12}$$

Meanwhile, a posteriori Dirichlet pdf *of the mixing weights α is given by its updated statistics*

$$\kappa_{p;t} = \kappa_{p;t-1} + w_{p;t}. \tag{6.13}$$

The estimation uses the filtered data after a normalization $\tilde{\tilde{\Psi}}_{p;t} = \frac{\tilde{\Psi}'_{p;t}}{\sqrt{D_{p;t}}}$ (5.19).

The predictive pdfs of each component is of Student form and its value can be calculated by

$$f(y_t|u_t, d(t-1), p) = \frac{(2\pi D_{p;t})^{-0.5}\, \mathcal{I}\left(V_{p;t-1} + \tilde{\tilde{\Psi}}_{p,t} \tilde{\tilde{\Psi}}'_{p;t}, \nu_{p;t-1} + 1\right)}{\mathcal{I}(V_{p;t-1}, \nu_{p;t-1})},\ p \in p^* \tag{6.14}$$

where the same type of integrals $\mathcal{I}(V, \nu)$ (4.23) is used. $D_{p;t-1}$ is the function of the C-parameters and determined by the LDL' decomposition of the corresponding partial covariance matrix.

Proof: The first statement is directly implied by marginalization over p_t. The second statement is based on the pre-whitening by the extended LD filters.

With them, to apply Bayes rule, c.f. (2.6) under the natural conditions of control, Definition 2.1.1, it leads to the the exact updating with the unknown value δ_{p,p_t}.

To preserve *a posteriori* pdf of the parameters in the desirable product form, we approximate δ_{p,p_t} by its expectation $w_{p;t}$ using marginalization over Θ_a and α. Here, the facts are exploited that the predictive pdf of each component is of a Student type (4.30) and the elementary property of Dirichlet pdf $Di_\alpha(\kappa)$ stating that $E[\alpha_p|\kappa] \propto \kappa_p$. □

The above estimation proposition leads to the following algorithm,

Algorithm 6.2.1 (MARMMAX-QB algorithm)

Off line (initial) mode

- *Require the C-parameters of the model $\Theta_c = \{C_p\}_{p\in p^*}$ to be given in some way.*
- *Select structure of the mixture by specifying*
  - *the number of components n_p.*
    This number also determines the number of filters needed.
  - *the structure of each individual component.*
    It means to determine the structure of the ARX part by the corresponding data vector, the structure of the MA part by its order n_c and the structure of C-polynomial.
- *Select the prior pdfs $f(\Theta_{a,p})$ of each parameterized component in the conjugate form*

  $$f(\Theta_{a,p}) = GiW_{\Theta_{a,p}}(V_{p;0}, \nu_{p;0}) \equiv GiW_{\Theta_{a,p}}(L_{y\psi p;0}, L_{\psi p;0}, D_{yp;0}, D_{\psi p;0}, \nu_{p;0}), \quad p \in p^*$$

  with the corresponding specified initial statistics values $L_{y\psi p;0}, L_{\psi p;0}, D_{yp;0}, D_{\psi p;0}, \nu_{p;0}$.
- *Select prior pdfs of component weights in the conjugate Dirichlet form*

  $$f(\alpha) = Di_\alpha(\kappa_0),$$

  with the specified initial values $\kappa_{p;0} > 0$.
- *Select forgetting factor $\lambda \in (0, 1]$, a set of alternative pdfs $f_A(\Theta_p)$ and $f_A(\alpha)$ in conjugate forms for stabilized forgetting.*
- *Specify the initial values of n_p extended LD filters defined by (5.15)-(5.16) for prewhitening.*

On line (sequential) mode

1. *Evaluate the point estimates the mixing weights α_p based on data $d(t-1)$,*

   $$\hat{\alpha}_{p;t-1} \equiv \mathcal{E}[\alpha_p|d(t-1)] = \frac{\kappa_{p;t-1}}{\sum_{\tilde{p}\in p^*} \kappa_{\tilde{p};t-1}}.$$

2. *Acquire the new data item d_t.*
3. *Run n_p LD filters (5.15)-(5.15) (in parallel) to generate the filtered data $\tilde{\tilde{\Psi}}_{p;t}$, according to (5.19).*
4. *Calculate the value of the predictive pdf for each individual components:*

- *Perform a* trial updating *of the corresponding statistics,*

$$\begin{aligned} V_{p;t} &= V_{p;t-1} + \tilde{\tilde{\Psi}}_{p,t}\tilde{\tilde{\Psi}}'_{p;t}, \quad p \in p^* \\ \nu_{p;t} &= \nu_{p;t-1} + 1. \end{aligned}$$

Here neither stabilized forgetting nor date weighting is used. The partitioned $L'DL$ decomposition of extended information matrix $V_{p;t}$ has to be used for the sake of numerical stability, see Section 4.2.2,

$$L_{p;t-1}, D_{p;t-1}, \nu_{p;t-1} \to L_{p;t}, D_{p;t}, \nu_{p;t}.$$

- *Determine the values of the predictive pdfs, using the above trial updates and the formula (6.14).*

5. *Compute the probabilistic weights $w_{p;t}$, using (6.10).*
6. *Update a posteriori pdf $f(\alpha|d(t))$ of mixing weights, i.e., update the scalars $\kappa_{p;t}$, using (6.13).*
7. *Update a posteriori pdf $f(\Theta_{a,p}|d(t))$ of each individual components, i.e., update the corresponding statistics, using the recursions (6.12).*

   Here, the filtered data vector is weighted by $w_{p;t}$ in the updating. The $L'DL$ decomposition of the extended information matrix

$$L_{p;t-1}, D_{p;t-1}, \nu_{p;t-1} \to L_{p;t}, D_{p;t}, \nu_{p;t}, \quad p \in p^*$$

   has to be used again.
8. *Evaluate, if need be, the characteristics of $f(\Theta_{a,p}|d(t))$ describing other parameters.*

   For instance, the point estimates of the $\Theta_{a,p} \equiv [\theta_{a,p}, r_{e,p}]$, according to (4.32)–(4.34).
9. *Repeat the sequential mode, when $t \leq \mathring{t}$.*

6.2.2 QB Estimation and Prediction of ARMMAX Models

With the restriction that there is a common ARX part in all components, the estimation of ARMMAX model is relatively simpler than that of MARMAX mode and requires less computation.

Proposition 6.2.2 (QB estimation of ARMMAX model with known C-parameters) *Consider the ARMMAX model (3.22) and assume that its C-parameters $\Theta_c \equiv \{C_p\}_{p=1}^{n_p}$ are known.*

i) Introduce unobserved discrete random pointers $p_t, t \in t^*$ to label the active component at each time instant t and assume it takes the value $p \in p^* \equiv \{1, \ldots, n_p\}$ with the probability α_p.

Then, there is the joint pdf of the unobserved pointer p_t and the observed data d_t

$$f(y_t, p_t|u_t, d(t-1), \Theta) = \prod_{p \in p^*} [\alpha_p f(y_t|u_t, d(t-1), \Theta_a, C_p, p)]^{\delta_{p,p_t}} \tag{6.15}$$

Its marginal pdf $f(y_t|u_t, d(t-1), \Theta)$ is described by the ARMMAX model. $\Theta_a = \{\theta, r_e\}$ is the parameters set describing the common unknown AR(X) part of all components.

ii) If normality is assumed, then each of its components can be described by the filtered regression

$$f(y_t|u_t, d(t-1), \Theta_a, \Theta_c, p) \;=\; \mathcal{N}_{\tilde{y}_t}(\theta'\tilde{\psi}_{p;t}, r_e D_{p;t}), \quad p \in p^* \tag{6.16}$$

using the filter (5.15)-(5.16) determined by the LDL' decomposition *of the corresponding known partial covariance matrix. The filtered data vector $\tilde{\Psi}_{p;t}=[\tilde{y}_{p;t}, \tilde{\psi}'_{p;t}]'$ is therefore generated by means of (5.18). $D_{p;t}$ is determined by the extended LD filters.*

Consequently, the joint pdf (6.15) can be specified as

$$f(y_t, p_t|u_t, d(t-1), \Theta) = \prod_{p\in p^*} \left[\alpha_p \mathcal{N}_{\tilde{y}_t}\left(\theta'\tilde{\psi}_{p;t}, r_e D_{p;t}\right)\right]^{\delta_{p,p_t}}. \tag{6.17}$$

iii) Under the natural condition of control, Definition 2.1.1, let us take the conjugate priors of the mixing weights $f(\alpha)$ and the common pdf $f(\Theta_a)$ of all components in the Dirichlet *form and the* GiW *form, respectively*

$$\begin{aligned} f(\Theta_a|d(t-1), \Theta_c) &= GiW_{\Theta_a}(V_{t-1}, \nu_{t-1}),\ p \in p^* \\ f(\alpha|d(t-1), \Theta_c) &= Di_\alpha(\kappa_{p;t-1}), \end{aligned}$$

Moreover, assume the pdf *of all unknown parameters be of the product form*

$$f\left(\Theta_a, \alpha|d(t-1), \Theta_c\right) = GiW_{\Theta_a}(V_{t-1}, \nu_{t-1}) Di_\alpha(\kappa_{t-1}). \tag{6.18}$$

Then, it holds

$$f(\Theta_a, \alpha, p_t|d(t), \Theta_c) = \tag{6.19}$$

$$= GiW_{\Theta_a}\left(V_{t-1} + \sum_{p\in p^*} \delta_{p,p_t} \tilde{\tilde{\Psi}}_{p;t} \tilde{\tilde{\Psi}}'_{p;t}, \nu_{t-1}+1\right) Di_\alpha\left(\kappa_{t-1} + \sum_{p\in p^*} \delta_{p,p_t} [\underbrace{0, \ldots, 0}_{p-1}, 1, 0, \ldots]'\right).$$

If we approximate δ_{p,p_t} by its expectation $w_{p;t} \equiv E\left[\delta_{p,p_t}|d(t), \Theta_c\right]$

$$\begin{aligned} w_{p;t} &= \frac{\kappa_{p;t-1} f(y_t|u_t, d(t-1), p)}{\sum_{\tilde{p}\in p^*} \kappa_{\tilde{p};t-1} f(y_t|u_t, d(t-1), \tilde{p})} \\ &\propto \mathcal{I}\left(V_{t-1} + \tilde{\tilde{\Psi}}_{p;t} \tilde{\tilde{\Psi}}'_{p;t}, \nu_{t-1}+1\right)(\kappa_{p;t-1}+1). \end{aligned} \tag{6.20}$$

Then, a posteriori pdf of the parameters at time instant t is preserved in the same product form as (6.18), with the associated statistics updated according to the recursions

$$V_t \;=\; V_{t-1} + \sum_{p\in p^*} w_{p;t} \tilde{\tilde{\Psi}}_{p;t} \tilde{\tilde{\Psi}}'_{p;t}, \qquad \nu_t = \nu_{t-1} + 1, \tag{6.21}$$

$$\kappa_{p;t} \;=\; \kappa_{p;t-1} + w_{p;t}, \qquad p \in p^* \tag{6.22}$$

with the filtered data after a normalization $\tilde{\tilde{\Psi}}_{p;t} = \frac{\tilde{\Psi}'_{p;t}}{\sqrt{D_{p;t}}}$ (5.19). They describe the Quasi-Bayes estimation of the ARMMAX model.

The predictive pdfs of each component is of Student form and its value can be calculated by

$$f(y_t|u_t, d(t-1), \Theta_c, p) = \frac{(2\pi D_{p;t})^{-0.5}\, \mathcal{I}\left(V_{t-1} + \tilde{\tilde{\Psi}}_{p,t} \tilde{\tilde{\Psi}}'_{p;t}, \nu_{t-1}+1\right)}{\mathcal{I}(V_{t-1}, \nu_{t-1})},\ p \in p^* \tag{6.23}$$

where the same type of integrals $\mathcal{I}(V, \nu)$ (4.23) is used. $D_{p;t-1}$ is the function of the C-parameters and determined by the LDL' decomposition of the corresponding partial covariance matrix.

Proof: The proof can be carried out similarly to that of Proposition 6.2.1, with the attention paid on the influence of the assumption that all components have the the common ARX part parameters.

The assumption implies that a single *extended information matrix* V_t and a scalar ν_t are updated in the estimation for all components by means of (6.21), instead of updating the statistics for each component individually. Consequently, the predictive pdf of each component (5.19) may not only determined by the corresponding C_p but also by the others C-parameters Θ_c. □

The above proposition of the estimation leads to the following algorithm,

Algorithm 6.2.2 (ARMMAX-QB algorithm)

Off line (initial) mode

- *Require the C-parameters of the model* $\Theta_c = \{C_p\}_{p\in p^*}$ *to be given in some way.*
- *Select structure of the mixture by specifying*
  - *the number of components* n_p.
    This number also determines the number of filters needed.
  - *the structure of each individual component.*
    It means to determine the structure of the ARX part by the corresponding data vector, the structure of the MA part by its order n_c *and the structure of C-parameters.*
- *Select the prior pdfs* $f(\Theta_a)$ *for all parameterized components in the conjugate form*

$$f(\Theta_a) = GiW_{\Theta_a}(V_0, \nu_0) \equiv GiW_{\Theta_a}(L_{y\psi;0}, L_{\psi;0}, D_{y;0}, D_{\psi;0}, \nu_0),$$

  with the corresponding specified initial statistics values $L_{y\psi;0}, L_{\psi;0}, D_{y;0}, D_{\psi;0}, \nu_0$.
- *Select prior pdfs of component weights in the conjugate Dirichlet form*

$$f(\alpha) = Di_\alpha(\kappa_0)$$

  with the specified initial values $\kappa_{p;0} > 0$.
- *Select forgetting factor* $\lambda \in (0, 1]$, *a set of alternative pdfs* $f_A(\Theta_p)$ *and* $f_A(\alpha)$ *in the conjugate forms for stabilized forgetting.*
- *Specify the initial values of* n_p *extended LD filters defined by (5.15)-(5.16) for prewhitening.*

On line (sequential) mode

1. *Evaluate the point estimates the mixing weights* α_p *based on data* $d(t-1)$,

$$\hat{\alpha}_{p;t-1} \equiv \mathcal{E}[\alpha_p | d(t-1)] = \frac{\kappa_{p;t-1}}{\sum_{\tilde{p}\in p^*} \kappa_{\tilde{p};t-1}}.$$

2. *Acquire the new data item* d_t.
3. *Run* n_p *LD filters (5.15)-(5.15) (in parallel) to generate the filtered and scaled data* $\tilde{\tilde{\Psi}}_{p;t}$, *according to (5.19).*

4. *Calculate the values of the predictive pdfs for each individual component:*

   - *Perform a* trial updating *of the statistics for each individual component,* $p \in p^*$,

   $$\begin{aligned} V_{p;t} &= V_{t-1} + \tilde{\tilde{\Psi}}_{p,t}\tilde{\tilde{\Psi}}'_{p;t}. \\ \nu_{p;t} &= \nu_{t-1} + 1. \end{aligned}$$

   Here neither stabilized forgetting nor date weighting is used. The partitioned $L'DL$ *decomposition of extended information matrix* $V_{p;t}$ *has to be used for the sake of numerical stability, see Section 4.2.2*

   $$L_{t-1}, D_{t-1} \to L_t, D_t,$$

   - *Determine the values the predictive pdfs by means of the above trial updates and the formula (6.23).*

5. *Compute the probabilistic weights* $w_{p;t}$ *using (6.20).*

6. *Update a posteriori pdf* $f(\alpha|d(t))$ *of mixing weights, i.e., update the scalars* $\kappa_{p;t}$ *using (6.22).*

7. *Update a posteriori pdf* $f(\Theta|d(t))$ *of the common AR(X) part, i.e., update the corresponding common statistics, using the recursions (6.21)-(6.22).*

   Here, the filtered data is weighted by $w_{p;t}$ *to update a single set* V_t, ν_t *for all components. The* $L'DL$ *decomposition of the extended information matrix*

   $$L_{t-1}, D_{t-1}, \nu_{t-1} \to L_t, D_t, \nu_t,$$

   has to be used again.

8. *Evaluate, if need be, the characteristics of* $f(\Theta_a|d(t))$ *such as the point estimates of the common parameter* $\Theta_a \equiv [\theta_a, r_e]$, *according to (4.32)–(4.34).*

9. *Repeat the sequential mode, when* $t \leq \mathring{t}$.

Remarks 6.2.1

1. *The predictive pdf is the key element needed both in the QB-MARMAX estimation and the QB-ARMMAX estimation. Being convenient for evaluation of likelihood function, the ratio (4.31) is used to compute their values. However, when prediction errors are explicitly needed, it may be also advantageous to directly use the expression of the Student form (4.30).*

2. *The MARMAX-QB and ARMMAX-QB estimations require feasible computational load which is close to several recursive least-squares estimations. The computational burden increases linearly with the number of components* n_p.

3. *Similarly to the last chapter, two kinds of LD decomposition are involved in the both estimations:*

   - $L'DL$ *decomposition of extended information matrix* V *in its updating to avoid the induced numerical troubles.*
   - LDL' *decomposition of partial covariance matrix* S *in prewhitening.*

6.3 Approximate Parallelism

Another important property of the ARMAX mixtures is that they can be interpreted as an approximate parallel realization of several ARMAX models. In another words, the ARMAX mixtures and therefore their estimations provide a quantitative measure of descriptive quality of the corresponding ARMAX components in parallel. In this sense, the ARMAX mixtures provide certain *"algorithmic" parallelism* to inspect several ARMAX's in parallel. However, it has to be stressed that they can only produce *approximate* parallelism, since an approximation is used in the numerical mixture estimation.

With their different model structures, MARMAX models and ARMMAX models provide different level parallelism. We shall firstly give two propositions below to support this claim on the MARMAX models, we then discuss the case of ARMMAX models afterwards.

Proposition 6.3.1 (Asymptotic of Bayesian estimation) *Under natural conditions of control, Definition 2.1.1, if some finite constants satisfying* $0 < \underline{K}_\Theta \leq \overline{K}_\Theta < \infty$ *exist, at the time instant* $\bar{t}_\Theta \in \{1, 2, \ldots\}$ *for almost all* $\Theta \in \Theta^*$*, such that*

$$\underline{K}_\Theta f(y_t|u_t, d(t-1), \Theta) \leq {}^{[o]}f(y_t|u_t, d(t-1)) \leq \overline{K}_\Theta f(y_t|u_t, d(t-1), \Theta), \ \forall t > \bar{t}_\Theta, \tag{6.24}$$

where ${}^{[o]}f(y_t|u_t, d(t-1))$ *denotes the "true" generator of data. The index* Θ *indicates that the constants may dependent on it.*

Then, the pdf $f(\Theta|d(t))$ *converges almost surely (a.s.) to a pdf* $f(\Theta|d(\infty))$*. Moreover, the support* $\mathrm{supp}\,[\,f(\Theta|d(\infty))] = \{\Theta : \ f(\Theta|d(\infty)) > 0\}$ *of the asymptotic pdf* $f(\Theta|d(\infty))$ *coincides with the following set of minimizing arguments*

$$\mathrm{supp}\,[\,f(\Theta|d(\infty))] = \arg \inf_{\Theta \in \mathrm{supp}[\,f(\Theta)] \cap \Theta^*} \mathcal{H}_\infty \left({}^{[o]}f||\Theta \right), \tag{6.25}$$

where $\mathcal{H}_\infty \left({}^{[o]}f||\Theta \right) = \lim_{t\to\infty} \mathcal{H}_t \left({}^{[o]}f||\Theta \right)$*, with*

$$\mathcal{H}_t \left({}^{[o]}f||\Theta \right) \ = \ \frac{1}{t} \sum_{\tau \leq t} \int {}^{[o]}f(y_\tau|u_\tau, d(\tau-1)) \ln \left[\frac{{}^{[o]}f(y_\tau|u_\tau, d(\tau-1))}{f(y_\tau|u_\tau, d(\tau-1, \Theta)} \right] dy_\tau. \tag{6.26}$$

Thus, if there is a unique consistent estimate of Θ*, then the Bayesian estimation provides it.*

Proof: Under the natural conditions of control, *a posteriori* pdf can be written in the form

$$f(\Theta|d(t)) \ \propto \ f(\Theta) \exp\left[-t\mathcal{H}(d(t)||\Theta)\right], \ \text{with} \tag{6.27}$$

$$\mathcal{H}\left(d(t)||\Theta\right) \ = \ \frac{1}{t} \sum_{\tau \leq t} \ln \left[\frac{{}^{[o]}f(y_\tau|u_\tau, d(\tau-1))}{f(y_\tau|u_\tau, d(\tau-1), \Theta)} \right]. \tag{6.28}$$

This form exploits the fact that the non-normalized *a posteriori* pdf can be multiplied by any factor independent of Θ.

Let us fix the argument $\Theta \in \Theta^*$ and define the deviation

$$e_{\Theta;\tau} \equiv \ln \left[\frac{{}^{[o]}f(y_\tau|u_\tau, d(\tau-1))}{f(y_\tau|u_\tau, d(\tau-1), \Theta)} \right] - {}^{[o]}\mathcal{E}\left[\ln \left[\frac{{}^{[o]}f(y_\tau|u_\tau, d(\tau-1))}{f(y_\tau|u_\tau, d(\tau-1), \Theta)} \right] |u_\tau, d(\tau-1) \right], \tag{6.29}$$

where ${}^{[o]}\mathcal{E}[\cdot|u_\tau, d(\tau-1)] = \int {}^{[o]}f(y_\tau|u_\tau, d(\tau-1)) \ln \left[\frac{{}^{[o]}f(y_\tau|u_\tau, d(\tau-1))}{f(y_\tau|u_\tau, d(\tau-1), \Theta)} \right] dy_\tau$ is the conditional expectation of $\ln \left[\frac{{}^{[o]}f(y_\tau|u_\tau, d(\tau-1))}{f(y_\tau|u_\tau, d(\tau-1), \Theta)} \right]$ with respected to ${}^{[o]}f(y_\tau|u_\tau, d(\tau-1))$. A direct check reveals that the introduced deviations $e_{\Theta;\tau}$ are zero-mean and mutually non-correlated.

Followed (6.26), (6.28) and (6.29), it holds

$$\mathcal{H}(d(t)||\Theta) = \mathcal{H}_t\left({}^{[o]}f||\Theta\right) + \frac{1}{t}\sum_{\tau\leq t} e_{\Theta;\tau}.$$

The assumption (6.24) implies that the variance of $e_{\Theta;\tau}$ is bounded. Consequently, the last term on the right hand side of the above expression converges to zero almost surely (*a*.s.), see [29]. The first term is non-negative, as it can be viewed as a sum of KL distances (2.7). Due to (6.24), it is also finite. Thus, (6.28) converges almost surely to the non-negative value $\mathcal{H}\left({}^{[o]}f||\Theta\right)$.

Moreover, the pdf $\mathcal{H}\left({}^{[o]}f||\Theta\right)$ remains unchanged, if we subtract $\inf_{\Theta\in\mathrm{supp}[\,f(\Theta)]\cap\Theta^*}\mathcal{H}_\infty\left({}^{[o]}f||\Theta\right)$ from the exponent of its non-normalized version (6.27). Then, the exponent contains

$$-t\times \text{ an asymptotically non-negative factor.}$$

Thus, the pdf $f(\Theta|d(\infty))$ may be asymptotically non-zero only on the minimizing arguments of (6.25).

If the unique ${}^{[o]}\Theta$ exists such that ${}^{[o]}f(y_\tau|u_\tau, d(\tau-1) = f\left(y_\tau|u_\tau, d(\tau-1), {}^{[o]}\Theta\right)$, then it is the unique minimizing argument due to the elementary properties of the KL distance. □

The following proposition exploits the results of the above proposition to justify our claim on the parallel nature of the MARMAX model.

Proposition 6.3.2 (Parallelism of MARMAX model) *Under the natural conditions of control, Definition 2.1.1, consider a MARMAX model parameterized by the collection* $\Theta_c \equiv \{C_p\}_{p\in p^*}$

i) Then, the predictor of the model has the form

$$f(y_t|u_t, d(t-1), \Theta_c) = \sum_{p=1}^{n_p}\hat{\alpha}_{p;t-1}(\Theta_c)f(y_t|u_t, d(t-1), C_p), \tag{6.30}$$

with the point estimate of mixing weights

$$\hat{\alpha}_{p;t-1}(\Theta_c) = \mathcal{E}[\alpha_p|d(t-1), \Theta_c],$$

and the component predictor

$$f(y_t|u_t, d(t-1), C_p) \propto \mathcal{I}\left(V_{p;t-1}(C_p) + \tilde{\tilde{\Psi}}_t(C_p)\tilde{\tilde{\Psi}}_t'(C_p), \nu_{p;t-1}+1\right). \tag{6.31}$$

In addition, the following inequality holds

$$0 \;\leq\; \mathcal{H}\left({}^{[o]}f||\Theta_c\right) \leq \sum_{p\in p^*}\hat{\alpha}_{p;\infty}(\Theta_c)\mathcal{H}_p\left({}^{[o]}f||\Theta_c\right), \tag{6.32}$$

with a.s. existing

$$\hat{\alpha}_{p;\infty}(\Theta_c) \;=\; \lim_{t\to\infty}\hat{\alpha}_{p;t}(\Theta_c),\; 0\leq\mathcal{H}_p\left({}^{[o]}f||\Theta_c\right) = \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=1}^{t}\ln\left(\frac{{}^{[o]}f(y_\tau|u_\tau, d(\tau-1))}{f(y_\tau|u_\tau, d(\tau-1), C_p)}\right).$$

ii) If the "true" system is described by a single ARMAX model and we use the MARMAX model in its estimation,

- *the C-parameters of the "true" ARMAX system are defined by some vector* $^{[o]}C$,
- *the C-parameters* C_p, $p \in p^*$, *of each component of the MARMAX model has the same order* n_c *as that of the* $^{[o]}C$,
- *the C-parameters* Θ_c *of the MARMAX are the vertices of a non-degenerate simplex in* n_c+1*-dimensional real space.*

Then Θ_c *may belong to the support of a posteriori pdf* $f(\Theta_c|d(\infty))$ *only if the weighted component entropy rates* $\hat{\alpha}_{p;\infty}(\Theta_c)\mathcal{H}_p\left(^{[o]}f||\Theta_c\right)$ *are simultaneously minimized. In other words,* Θ_c *may belong to the support of a posteriori pdf only if the vector* C_p *maximizes asymptotically the weighted* component log-likelihood,

$$l_p(d(t),\Theta_c) \equiv \hat{\alpha}_{p;t}\sum_{\tau=1}^{t}\ln[f(y_\tau|u_\tau,d(\tau-1),C_p)],\ p\in p^*, t\to\infty \tag{6.33}$$

with the component predictors (6.31).

Proof: The form of the mixture predictor (6.30) is simply obtained by taking the conditional expectation with respect to the α and the ARX parts. The form of the component predictors (6.31) is implied by Proposition 4.2.3.

The Jensen inequality implies the inequality between finite sums defining asymptotically the involved entropy rates.

Almost sure convergence of $\hat{\alpha}_{p;t}$ follows from the fact that, as a conditional expectation of a bounded variable, it is a bounded martingale, see Chapter 3.1.

Properties of a specific component entropy rate can be shown exactly as in the proof of Proposition 6.3.1.

The minimum can be reached if the (asymptotic) estimate of component weights $\hat{\alpha}_{t-1}(\Theta_c)$ is

- either a zero-one probabilistic vector so that only the pdf $f(\Theta|d(\infty))$ of one component have the smallest KL distance to the true one.
- or a vector with non-zero entries simultaneously so that more than one components having the same smallest KL distance to the true one.

The latter possibility is excluded by the use of C_p's defining a non-degenerate simplex. □

The above propositions justify partially the parallelism of ARMMAX models as well. However, assuming a common AR(X) part, ARMMAX models leads to a joint updating of the common parameter $\Theta_a \equiv \{\theta_a, r_e\}$ in the estimation, i.e., (6.21)

$$V_t \;=\; V_{t-1} + \sum_{p\in p^*} w_{p;t}\tilde{\tilde{\Psi}}_{p;t}\tilde{\tilde{\Psi}}'_{p;t},\quad \nu_t = \nu_{t-1}+1.$$

This joint updating may bring some undesirable influences. Currently, no formal analysis is available to measure this influence. Here we just give a preliminary discussion on one of most significant problems.

The problem stems from the use of LD filter. Recall that the estimation of an ARMA(X) based on the extended LD filter, Section 5.3, under two situations

- a C-polynomial having some unstable roots is given,
- its stable reflection is given.

may lead to the identical estimate of the colored noise covariances r_i in both cases but different estimates of noise variance r_e.

This simple fact brings no trouble for MARMA(X) models. However, it could bring some bias in the estimation for ARMMA(X) models through the joint updating of the common AR(X) part when both stable and unstable cases are involved.

It is well-known that, for any C-polynomial (in the backward shift operator z^{-1})

$$C(z^{-1}) = 1 + c_1 z^{-1} + ... + c_{n_c} z^{-n_c},$$

there always exists another polynomial $\tilde{C}(z^{-1})$ as the stable reflection of the $C(z^{-1})$. $\tilde{C}(z^{-1})$ has the guaranteed stability and shares the same spectral density as the $C(z^{-1})$.

On the other hand, although we showed that LD factorization converge to the coefficients of the stable reflection $\tilde{C}(z^{-1})$ even when the original given C-polynomial $C(z^{-1})$ is unstable, the transient period of the filter before convergence is important and directly influences the practical results.

Thus, one possible idea here is to use a stable replacement operation before we run the filters:

> *To replace a given C-polynomial by its stable reflection when it occurs to be unstable.*

It has to be stressed here that such a stable replacement operation may have some numerical precision problem to certain level in practical:

> *It is only possible to prove that the roots of a given polynomial can be found within some roundoff error, or reversely the coefficients of a polynomial could only be found with its roots equal to some given roots within roundoff error.*

Therefore, other solutions are worth of considering. One of them is discussed in the concluding chapter.

Chapter 7

Improved Bayesian Solutions to ARMAX Models

Under the assumptions that the covariances or the C-parameters of its stochastic MA part are known, we had a first inspection on Bayesian estimations of ARMA(X) model in the Chapter 5. Now let us return to this topic for some further study. We consider improved Bayesian estimation and prediction of ARMA(X) model to have Bayesian estimation setting up to its ARX part and to reduce the uncertainty of its MA part by searching for a point estimate of the C-parameters.

7.1 MMQ and MAQ Methods

Consider the SO normal ARMA(X) model (5.1), i.e.,

$$y_t = \theta' \psi_t + v_t.$$

In contrast to the case of Chapter 5, here we do not know the parameters of its stochastic MA part, neither the noise variance r_e nor the n_c-dimensional C-parameters vector $C = [c_1, \cdots, c_{n_c}]$ are known. The model is then parameterized by

$$\Theta = \{\Theta_a, \ C\},$$

where $\Theta_a = (\theta, r_e)$ describes the AR(X) part of the model.

Our goal on the estimation of the MA part here is to seek for an approach with its applicability not being limited by the stability requirement and to deal efficiently with the unknown C-parameters in high dimensions (≥ 2).

7.1.1 Problem Formulation

From Bayesian viewpoint, to estimate the unknown C-parameters of the above model is to use *a posteriori* pdfs by specifying the Proposition 2.1.3 as follows:

For any given *a priori* pdf $f(C)$, the observations $d(t)$ correct it to *a posteriori* pdf $f(C|d(t))$ through the Bayes rule (2.6). Under the natural condition of control, Definition 2.1.1, it reads

$$f(C|d(t)) \propto \mathcal{L}(d(t), C) f(C), \tag{7.1}$$

where $\propto$ means proportionality, i.e., equality up to a normalizing factor which is independent of C. The *likelihood function*

$$\mathcal{L}(d(t), C) = \prod_{\tau=1}^{t} f(y_\tau | u_\tau, d(\tau - 1), C) \tag{7.2}$$

contains all information about C-parameters which can be extracted from the observed data $d(t)$. For a fixed C, its value is simply a product of values of the predictive pdfs.

Unfortunately, such *a posteriori* pdf could only be used *formally*. The complex nature of likelihood function $\mathcal{L}(d(t), C)$ makes its general analytical treatment difficult, see the next section for more detail. Assuming that a finite set of the competing C's are available, Peterka [7] in 1989 made the formula (7.1) applicable by a Bayesian comparison of hypothesis to select the most promising candidate. Essentially, he evaluated simultaneously several *posterior* probabilities $f(C|d(t))$ on the hypotheses that a specific given C-parameters vector is the best one among the given set based on the observed data samples. However, no rule has been given how to generate such a candidate set. In addition, another problem he encountered is that *a posteriori* probabilities converge to a zero-one vector in generic cases, see Proposition 6.3.2, regardless of the quality of the original choice of candidates.

To overcome the practical restriction of the Bayesian comparison of hypotheses while preserving its ability to use full Bayesian solution with respect to the ARX part motivates the rest discussion of this section. Essentially, an efficient numerical maximization procedure is sought for generating the most interesting competitors around the maximum of the posterior pdf.

For presentation simplicity, we restrict ourselves to a uniform *a priori* pdf $f(C)$ in (7.1) and search for the candidates in a neighborhood of

$$\arg\max_{C} f(C|d(t)) = \arg\max_{C} \mathcal{L}(d(t), C), \quad C \in R^{n_c}. \tag{7.3}$$

With the given order n_c of the MA part, it defines a n_c-dimensional optimization problem of maximizing the scalar-valued nonlinear function $\mathcal{L}(d(t), C) : R^{n_c} \to R$. Note that usually this optimization would have to subjected to the constraint that $C \in C^*$, here $C^* \subset \mathbf{R}^{n_c}$ is the space determined by the stability requirement on its C-polynomial. Nevertheless, for the reasons stated in Chapter 3.3 and Chapter 5, we are allowed to consider it as an unconstrained problem.

7.1.2 Choices of MDS method and ARMAX Mixtures

It hasn't been an easy task to choose a proper method to solve the optimization problem (7.3) in our estimation setting. A variety of reasons stated below have focused our attention on the multidirectional search method (MDS), which was reviewed in Chapter 2.3.2:

First, an efficient evaluation of the gradient is inhibited by the time-variation of the LD filter and therefore the complex nature of the $\mathcal{L}(d(t), C)$ (7.2). Thus, we have to restrict ourselves to derivative-free methods.

Then, The fact that evaluations of the $\mathcal{L}(d(t), C)$ (7.2) could be expensive narrows down the options further on by excluding stochastic optimization methods.

Meanwhile, as shown in Chapter 5, if we want to use LD type filters for prewhitening, we have to consider the C-parameters of the model as time varying even when they are originally time-invariant. The variations are data independent and driven only by the time-invariant C-parameters. Therefore, although the evaluation of the type filters is computationally inexpensive, its associated C-parameters variations hinder the attempts to estimate the unknown C-parameters recursively. It seems to preclude to use Monte-Carlo-based maximization.

Moreover, the optimized likelihood function may be multi-modal with quite sharp but smooth modes. Thus, we can rely at most on the continuous differentiability of the objective function $\mathcal{L}(d(t), C)$ with respect to C.

These considerations reduce our options more or less to simplex-based direct search approaches. The review of Chapter 2.3 showed that the NM algorithm, the most popular simplex method, has only weak convergence analysis established in one dimension. This excludes it

from our considerations since we want an algorithm also efficient in higher dimensions (≥ 2). The review also revealed that the MDS method has brought a new interest in the direct search methods by the following properties:

- It has strong convergence properties and verified robustness.
- It is a derivative-free method, i.e., it does not require the information about the derivative of the objective function.
- It works well with "noisy" function values.
- It is suitable to be executed in parallel.

Obviously, the first three properties are all favorable to our problem. Especially with its strong convergence analysis, the MDS method guarantees the efficiency in high dimension.

However, its fourth property needs to be justified for its efficient use in practical. The original intention of the MDS method is to execute several evaluations of objective function in parallel on a machine of multi-processors by taking advantage of the computational parallelism. But it then faces problems such as the cost of synchronization (i.e. interprocessor communication) and some additional restrictions on the objective function. What's more, it is only single-processor machines that are most often available.

Therefore, our practical interest here is to look for an efficient use of the MDS method on a single-processor machine rather than parallel ones. The studied approximate parallelism of the ARMAX mixtures in the last chapter suggests the possibility and feasibility to use these mixtures to provide certain "algorithmic parallel environment" for the parallel evaluations involved in the multidirectional search.

Recall that the MARMAX and ARMMAX model can only provide approximate parallelism with the approximation used in the corresponding numerical mixture estimations. In another words, only approximate likelihoods are available rather than their exact values. In this point, it is attractive for us that the MDS method is able to work well with "noisy" function values.

With the limitation discussed in Chapter 6.3, the ARMMAX models, compared to the MARMAX models, may seem less suitable to provide the parallel evaluations for the MDS. However, with its certain efficiency confirmed by experiments, it may still make sense to discuss this possibility here.

7.1.3 Description of MMQ/MAQ Estimation

Now let us combine organically the essential ingredients selected in last section:

- The MDS method,
- The parallelism of the ARMAX mixtures, with the associated efficient iterative estimation algorithm MARMAX-QB/ARMMAX-QB,

This leads to so-called MMQ (MDS-MARMAX-QB) estimation and MAQ (MDS-ARMMAX-QB) estimation of an ARMAX model. These two estimation procedures share the same main scheme and can be distinguished according to the type of mixtures used: With MARMAX type mixtures, it corresponds to the MMQ estimation. With ARMMAX type mixtures, it becomes the MAQ estimation.

The main idea here is that unknown C-parameters could be searched by the MDS method in "approximated algorithmic" parallel environment of the ARMAX mixtures to generate a sequence of points that convergence to a critical point, ideally the "true" C-parameters. More exactly,

- The number of vertices in the used simplex is equal to the number of components in the corresponding ARMAX mixtures.

  For the n_c-dimensional optimization problem (7.3), the number of vertices in the used simplex is $n_c + 1$ according to Chapter 2.3. Here, n_c is the order of the MA part and it is required to be known. Thus, $n_c + 1$-vertex simplex and $n_c + 1$-component mixtures are adopted in the estimations.

- The evolution of simplex corresponds to the redefinition of MARMAX/ARMMAX mixture.

  The search rules of the MDS method drive the evolution of simplex. Then, the vertices of the evolving simplex specify the C-parameters $\Theta_c = \{C_p\}_{p\in p^*}$ of the used mixtures.

- The mixtures facilitate the objective-function evaluations for the MDS.

  The estimation of the mixtures relies on the MARMAX-QB/ARMMAX-QB algorithms developed in last chapter. In particular, the estimation of the ARX part relies on running several extended LD filters in parallel so that Bayesian setting can be preserved at least up to this part.

For a given C-parameter C_p, $C_p \in \Theta_c$, when the mixtures are not used, it corresponds to a single ARMA(X) model with the likelihood $\mathcal{L}(d(t), C_p)$. When the mixtures are used, it corresponds to a single ARMA(X) component of the mixture used with the component likelihood value $l_p(d(t), C_p)$, which can be interpreted as an approximation of the likelihood $\mathcal{L}(d(t), C_p)$.

Thus in the context of the MMQ/MAQ estimation, the optimization problem (7.3) can be approximately expressed in terms of the component likelihoods $l(d(t), C)$. Consequently, we shall use the component likelihood values to judge the quality of the individual given C-parameters.

As usual, natural logarithm likelihoods (log-likelihoods) are used in practice because it is numerically easier. In the case of using the MARMAX, the component log-likelihoods can be given as follows

$$l_p(d(t), C_p) = \sum_{\tau=1}^{t} \ln\left[f(y_\tau|u_\tau, d(\tau-1), C_p)\right], \qquad C_p \in \Theta_c, \;\; p \in p^* \tag{7.4}$$

while in the case of the ARMMAX, the log-likelihood of each component may also depended on the C-parameters of the other components,

$$l_p(d(t), \Theta_c) = \sum_{\tau=1}^{t} \ln\left[f(y_\tau|u_\tau, d(\tau-1), \Theta_c)\right], \qquad p \in p^* \tag{7.5}$$

where $\Theta_c = \{C_p\}_{p\in p^*}$, $p^* = \{1, \cdots, n_c + 1\}$.

It is worth of stressing that component log-likelihoods are reflected in the overall log-likelihood of the corresponding mixtures, which are in the following forms respectively for the the MARMAX

$$\mathcal{L}(d(t), \Theta_c) = \sum_{\tau=1}^{t} \ln\left[\sum_{p\in p^*} \hat{\alpha}_{p;\tau}(\Theta_c) f(y_\tau|u_\tau, d(\tau-1), C_p)\right], \tag{7.6}$$

and for the ARMMAX

$$\mathcal{L}(d(t), \Theta_c) = \sum_{\tau=1}^{t} \ln\left[\sum_{p\in p^*} \hat{\alpha}_{p;\tau}(\Theta_c) f(y_\tau|u_\tau, d(\tau-1), \Theta_c)\right]. \tag{7.7}$$

The values of the mixture log-likelihood can be obtained as a byproduct of the QB estimations. They may serve for monitoring of success of the search made via component log-likelihoods.

MMQ/MAQ Algorithm

Here an algorithmic description of the MMQ/MAQ estimation is given below. For the involved Bayesian estimations, we have to consider the construction of prior pdfs. They shall be discussed together with the other implementation details in next section.

The procedure searches a point strictly improving over the best vertex. Three possible operations are defined to generate the trial points, see Chapter 2.3.2: *reflection, expansion, contraction.* After each operation, if at least one corresponding trial point has a higher log-likelihood than that of the current best vertex, the operation is called *successful.* To *accept* one operation, we replace the current vertices of the simplex by the corresponding trial points after the operation.

For simplicity, we denote $l = [l_1, \ldots, l_{n_c+1}]$ as the vector of all component log-likelihood values. The upper index distinguishes the operations or iteration stage, while the lower index distinguishes the component. For instance, l_p^r means the log-likelihood of the p-th component in *reflection* operation. The same rules is applied on the notation of C-parameters. For instance, $C_p^0 = [c_{1,p}^0, \cdots, c_{n_c,p}^0]$ means the *initial* values assigned to the C-parameters of the p-th component.

Algorithm 7.1.1 (MMQ/MAQ Estimation Algorithm)

Initial phase

- *Specify the order n_c of the MA part and select the type of mixtures, MARMAX or ARMAX, to be used for function evaluations.*
- *Select a starting point of the search procedure, i.e., a suitable initial guess C_1^0 of the C-parameters.*
- *Generate an initial non-degenerated simplex based on C_1^0,*

  $$< C_1^0, \cdots, C_{n_c+1}^0 >$$

  by defining the other n_c vertices C_p^0, $p = 2, \cdots, n_c + 1$, see the next section.
- *Select the expansion and contraction scalars $\chi \in (1, \infty)$, $\xi \in (0, 1)$ with the default unit reflection factor.*
- *Select stopping rules, see the next section.*
- *Specify the options of the corresponding off-line steps for the MARMAX-QB estimation or the ARMMAX-QB estimation, see Chapter 6.2 and Chapter 4.3.5.*

  To increase the chance to gain a successful estimation, some iterations are necessary so that iterative estimation is in need. To avoid redundant discussion, we did not give the description of iterative MARMAX-QB estimation or the ARMMAX-QB estimation. These iterative versions, however, can be easily obtained by the the discussion of Chapter 4.3.4 and Chapter 4.3.5.
- *Set j, the counter of the total number of iterations used in the search, to zero.*
- *Perform an initial iterative MARMAX-QB or ARMMAX-QB estimation, see Chapter 6.2 and Chapter 4.3.5.*

  Accumulate the component log-likelihoods l^0. Here the vector l^0 has its each individual entries l_p^0, $p = 2, \cdots, n_c + 1$, defined by (7.4) or (7.5).

  Accumulate, if required, the mixture log-likelihoods defined by (7.6) or (7.7).

- *Swap the vertices so that the vertex with the highest component likelihood is labelled as* C_1^0.

Iterative phase

Do while stopping rule is not met, set $j := j + 1$.

1. ***Reflection***

   - *Define* n_c *reflected vertices*

   $$C_p^r = 2C_1^{j-1} - C_p^{j-1}, \quad p = 2, \cdots, n_c + 1.$$

   - *Specify the C-parameters of the* $n_c + 1$ *components in the used mixture by the above reflected vertices and the current best vertex* C_1^{j-1}.
     Specify the corresponding a priori pdf on the ARX part as the flattened posterior pdf from the previous iteration.
   - *Perform iterative MARMAX-QB/ARMMAX-QB estimation, see Chapters 6.2 and 4.3.5.*
     Accumulate the component log-likelihoods l^r *for the mixture determined by the reflection. Here the vector* l^r *has its each individual entries* l_p^r, $p = 2, \cdots, n_c + 1$, *defined by (7.4) or (7.5).*
     Accumulate, if required, the mixture log-likelihoods defined by (7.6) or (7.7).
   - *Determine the components having the highest log-likelihood*

   $$m_r = \arg \max_{p \in p^*} l_p^r,$$

     here $m_r \in p^*$, $p^* = 1, \cdots, n_c + 1$.
   - *Go to the step 2, if* $m_r > 1$. *Otherwise, go to the step 3.*

2. ***Expansion***

   - *Define* n_c *expanded vertices*

   $$C_p^{j,e} = C_1^{j-1} + \chi(C_1^{j-1} - C_p^{j-1}), \quad p = 2, \cdots, n_c + 1.$$

   - *Specify the C-parameters of* $n_c + 1$ *components in the used mixture by the above expanded vertices and the current best vertex* C_1^{j-1}.
     Specify the corresponding a priori pdf on the ARX part as the flattened posterior pdf from the previous iteration.
   - *Perform iterative MARMAX-QB/ARMMAX-QB estimation, see Chapters 6.2 and 4.3.5.*
     Accumulate the component log-likelihoods l^e *for the mixture determined by the expansion. Here the vector* l^e *has its each individual entries* l_p^e, $p = 2, \cdots, n_c + 1$, *defined by (7.4) or (7.5).*
     Accumulate, if required, the mixture log-likelihoods defined by (7.6) or (7.7).
   - *Determine the components having the highest log-likelihood*

   $$m_e = \arg \max_{p \in p^*} l_p^e,$$

     here $m_e \in p^*$, $p^* = 1, \cdots, n_c + 1$.

- *Accept the expansion, if* $l^e_{m_e} > l^r_{m_r}$, *by replacing* C^j_p *by the expanded vertices* C^e_p, *for* $p = 2, \cdots, n+1$. *Otherwise accept the reflection by replacing* C^j_p *by the reflected vertices* C^r_p, *for* $p = 2, \cdots, n_c + 1$.
- *Go to step 4.*

3. ***Contraction***

   - *Define* n_c *contracted vertices*

     $$C^c_p = C^{j-1}_1 + \xi(C^{j-1}_1 - C^{j-1}_p)$$

     and accept them by replacing C^j_p *by the contracted points* C^e_p, *for* $p = 2, \cdots, n+1$.
   - *Specify the C-parameters of* $n_c + 1$ *components in the used mixture by the expanded vertices and the current best vertex* C^{j-1}_1.
     Specify the corresponding a priori pdf on the ARX part as the flattened posterior pdf from the previous iteration.
   - *Perform iterative MARMAX-QB/ARMMAX-QB estimation, see Chapters 6.2 and 4.3.5.*
     Accumulate the component log-likelihoods l^c *for the mixture determined by the expansion. Here the vector* l^c *has its each individual entries* l^c_p, $p = 2, \cdots, n_c + 1$, *defined by (7.4) or (7.5).*
     Accumulate, if required, the mixture log-likelihoods defined by (7.6) or (7.7).
   - *Determine the components having the highest log-likelihood*

     $$m_c = \arg \max_{p \in p^*} l^c_p,$$

     here $m_c \in p^*$, $p^* = 1, \cdots, n_c + 1$.
   - *Go to the step 4 if* $m_c > 1$, *otherwise go to the step 1.*

4. ***Swap***

   Swap the vertices so that a (new) best point is labelled as C^j_1 *according to the components likelihoods of the accepted mixture.*

Remarks 7.1.1

It may be advantageous sometimes to use batch MARMAX-QB/ARMMAX-QB estimation in a way similarly to Chapter 4.3.5.

7.1.4 Implementation Aspects

As a coupling of the MARMAX-QB/ARMMAX-QB estimation with the optimization of the MDS, the MAQ/MMQ estimation has to take both these two sides into accounts in the implementation.

Firstly, the issue of construction of prior pdfs involved in the MARMAX-QB/ARMMAX-QB estimations is briefly considered.

Then, we discuss the general issues of the implementation in multidirectional search, such as initial simplex, scaling factors and stopping criteria. We mainly follow the rules of the standard MDS procedure and further specify some of them to fit our parameter estimation setting. The options listed serve us mainly for references as they are used in the illustrative examples of the next chapter.

Prior and Flattening

The good choice of *a priori* pdf is critical for the estimation of MARMAX-QB/ARMMAX-QB. It is solved in the following way:

- In the very initial iteration of the MDS, *a priori* pdf is chosen by incorporating *a priori* knowledge on the ARX parts, see [34].
- In its other generic iterations, *a posteriori* pdf from the previous iteration on ARX parts is used as *a priori* one of the current iteration after a flattening.
- Note that iterative (batch) MARMAX-QB/ARMMAX-QB estimation is almost always used in practice. Thus, within each iterative (batch) estimation, *a priori* pdf is constructed in an iterative way Chapter 4.3.4 and 4.3.5.

Initial Simplex

As a simplex-based method, multi-directional search begins by choosing an initial simplex. Given the order n_c of the C-polynomials and the initial guess

$$C_1^0 = [c_{1,1}^0, \cdots, c_{n_c,1}^0], \tag{7.8}$$

then the n_c new points can be generated

$$C_p^0 = [c_{1,p}^0, \cdots, c_{n_c,p}^0], \quad p = 2, \cdots, n_c + 1 \tag{7.9}$$

to form an $n_c + 1$-vertex simplex

$$< C_1^0, \cdots, C_{n_c+1}^0 > . \tag{7.10}$$

General multi-directional search algorithm requires only that the initial simplex to be *non-degenerated* so that the n_c edges adjacent to any given vertex in the simplex spans the space R_c^n. Otherwise, with a degenerate simplex, the algorithm can only optimize over the subspace spanned by the degenerate simplex.

Shape, Size and Orientation

As we mentioned in Chapter 2.3, *regular* simplex and *right-angled* simplex are the most often used two types of simplex. In addition, some other shape non-degenerate simplex could also be used, for instance, the simplex defined in Matlab toolbox for the routine of the NM simplex.

Although the shape of simplex may not influence the MMQ/MAQ procedure significantly, the use of *right-angled* simplex may be more recommendable for our problem, since the C-parameters are scale dependent.

With a given initial guess C_1^0 (7.8), specifying the definition of *right-angled* simplex (2.26) to our case means to determine the n_c vertices (7.9) by some fixed distance β_p in each of n_c coordinate directions from the initial guess C_1^0

$$C_p^0 = C_1^0 + \beta_p \mathbf{1}_p, \;\; p = 2, \cdots, n_c + 1, \tag{7.11}$$

where $\mathbf{1}_p$ denotes the unit coordinate vector.

Now let us inspect how to determine the non-zero scalars β_p, $p = 2, \cdots, n_c+1$. The stability requirements on the C-polynomial implies a loose condition we could exploit to determined a

rough scaling of the C-parameters $C = [c_1, \cdots, c_{n_c}]$: its i-th entry element c_i in magnitude cannot exceed the combination number,

$$|c_i| \leq \begin{pmatrix} n_c \\ i \end{pmatrix}, i = 1, \cdots, n_c$$

where n_c is the order of MA part.

Therefore we can define β_p as follows

$$\beta_p = \pm h \begin{pmatrix} n_c \\ p \end{pmatrix}, \quad p = 1, \cdots, n_c \tag{7.12}$$

with its magnitude $|\beta_p| = h \begin{pmatrix} n_c \\ p \end{pmatrix}$ determining the *size* of the simplex. The positive scalar h becomes the only tuning parameter to control the size of the simplex as the order n_c is fixed. A proper choice of h is important in practice. Although if the initial simplex is either too small or too big, the MDS method can rescale accordingly with the operations of the expansion and contraction, it may cost a significant number of iterations to expand or contract simplex before any real progress can be made, particularly in the case of too big initial simplex.

The signs of β_p determine the important *orientation* of the initial simplex. There is no universal rule how to select them. Thus, whenever possible, it makes sense to try several initial options differing just in orientation.

Scaling Factors and Termination

Here we shall go for the usual choices on the scaling factors [15]:

$$\chi = 2, \quad \xi = 1/2$$

for expansion and contraction operations respectively, with unit reflection factor.

For termination, standard stopping rules are adopted and enriched by a specific one:

- To inspect *the relative size of simplex*, by measure the length of longest edge adjacent to the best vertex C_1^j

$$\frac{1}{\Delta} \max_{1 \leq i \leq n} \left|\left| C_i^j - C_1^j \right|\right| \leq \epsilon, \ \ \epsilon \in (0, 1), \tag{7.13}$$

  where $\Delta = \max\left(1, \left|\left| C_1^j \right|\right|\right)$ and ϵ is a pre-selected tolerance. Instead of $\epsilon = 1e-008$, approximately the square root of machine zero, expensive function evaluations in our estimation context enforces us to set this tolerance around $1e-004$.

- To limit the number of iterations j by a total bound J.

  The appropriate value of the bound J can be determined by the affordable computational time. Here, we benefit from the fact that the QB estimations have fixed *a priori* known computational demands.

- To check the *increments of the global log-likelihood of the mixture.*

  Availability of the global log-likelihood of mixture $\mathcal{L} = \mathcal{L}(d(t), \Theta_c)$, (7.6) or (7.7), allows us to add another possible rule to stop the evaluation when the increment among iteration steps is smaller than a pre-specified threshold η

$$\left| \frac{\mathcal{L}^j - \mathcal{L}^{j-1}}{\mathcal{L}^j} \right| \leq \eta, \quad \eta \in (0, 1) \tag{7.14}$$

7.2 PEB Method

To improve Bayesian estimation of ARMA(X) models, another natural idea worth to be inspected is to make use of the existing point estimation methods, such as PE, ELS, since only a point estimate of the C-parameters is sought for.

Here we consider the most popular PE method [1] or [2] for this task and combine it with Bayesian method for the rest of estimation. Similarly to the last section, the trick behind is to use a hybrid, which leads to a so-call PEB method,

- point estimation on C-parameters is provided by the PE method.
- the rest of estimation is provided as Bayesian solution. It is based on the extended LD filter for prewhitening.

With the nice properties of the PE method, the hybrid PEB method is able to deal efficiently with the unknown C-parameters in high dimension (≥ 2). However, this method has potential difficulties in the stability on the C-polynomial. As pointed out by Peterka, it is often a rule rather than an exception that the C-polynomial of an ARMA(X) model has its roots lying close to/on the unit circle, especially in case of fast sampling. Thus being free from potential difficulties in the stability on the C-polynomial, MMQ and MAQ approaches may appear to be more favorable than PEB in practice.

Remarks 7.2.1

1. *Note that PEB estimation imposes restrictive constraint on the stability of the C-polynomial, since estimates of C-parameters is returned from the PE method. Thus in the context of PEB estimation, the extended LD filter makes no differences from some other types of filter in this aspect.*

2. *A routine is available in system identification toolbox of Matlab [38] for the implementation of PE estimation and detailed discussion on the issue can be found in [1] or [2].*

Chapter 8

Illustrative Examples

After the theoretical study in the previous chapters, we present some illustrative experiments in this chapter to verify the practical properties of the theory and to inspect the performance of the proposed algorithms. The focus is put on the performance of the proposed MMQ and MAQ algorithms for the estimation of ARMA(X) models. They indicate the efficiency of the MARMAX-QB and ARMMAX-QB algorithms and confirm the underlying theory such as approximated parallelism of ARMMAX and MARMAX as well.

Essentially three groups of experiments were designed with the aims:

- to examine the basic properties of the MAQ and MMQ algorithm by estimating a simulated ARMA and a simulated OE (output error model), respectively;
- to provide the comparisons of the methods among the MMQ estimation, the MAQ estimation and the PEB estimation;
- to illustrate the performance of the three methods using real data.

In addition, some sensitivity tests are carried out with the varying implementation options, namely

- starting point of the MDS search;
- size and orientation of the initial simplex;

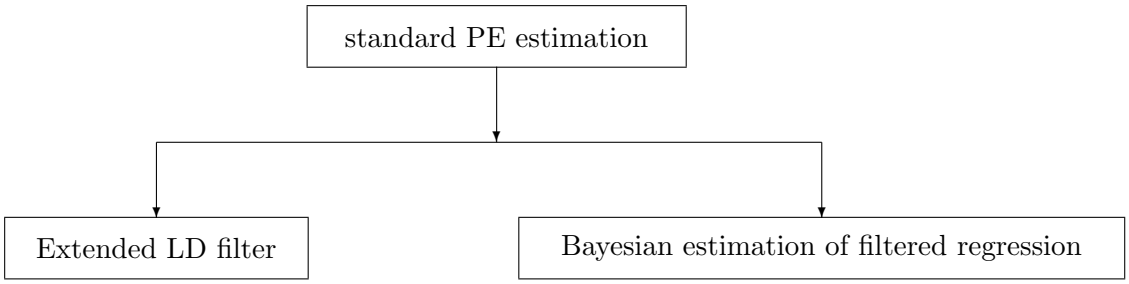
8.1 Preliminaries

8.1.1 Software Aspects

This subsection gives a few comments on the software implementations.

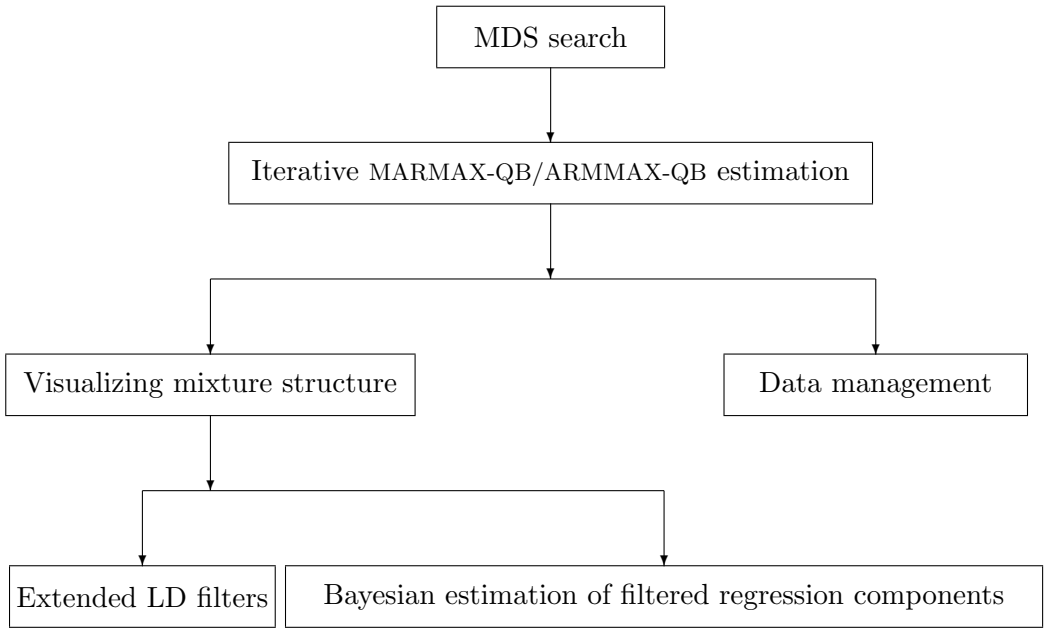
- For the PEB estimation, we use the routine in system identification toolbox of Matlab [38] for PE point estimation of C-parameters and the rest of estimation relies on the use of the extended LD filter. Therefore the estimation has a two-layer software architecture:

  Recall that the main idea of the PEB estimation is to apply standard PE estimation for point estimation of unknown C-parameters and use it in the rest of Bayesian estimation. Therefore, the first layer is a routine of standard PE estimation. In the next layer, (iterative) Bayesian estimation of the resulting filtered regression is based on the extended LD filter.



Software architecture of the PEB estimation algorithm

- For the MMQ and MAQ estimations, a software package is developed as m-files of Matlab. It is based on Mixtools [39], a toolbox has been developed in ÚTIA to provide Bayesian estimation and prediction of finite mixtures, mainly Markov and ARX types. Using its algorithmic basis, the MMQ/MAQ estimation is developed with a four-layer software architecture:



Software architecture of the MMQ/MAQ estimation algorithm

Recall that the main idea of the MMQ/MAQ estimation is to apply the MDS procedure to search for unknown C-parameters and exploit the mixture estimation algorithms for its function evaluations. Therefore, the first layer is a routine of the MDS search fitting to the estimation settings. In the next layer, iterative mixture estimations are then called within the MDS routine. The MMQ/MAQ algorithm described in the last chapter shows how these two layers interact to each other.

To be able to implement the mixture estimation, software representation of mixture structure and data management have to be considered as an intermediate layer to build a bridge with the estimation theory. As showed in Chapter 6 and Chapter 7, the mixture estimation relies on running several extended LD filters in parallel and Bayesian estimation of the corresponding filtered regression components. They serve as the basic layer of the procedure.

Note that sometimes it may be advantageous to use a batch mixture estimation instead in the second layer, see Chapter 4.3.5.

8.1.2 Measures of Performance

This subsection presents some measures which help us to judge performance of the methods.

Criterions of Estimation Quality

As hybrid methods, the MMQ, MAQ and PEB estimations provide a point estimation of the C-parameters for an ARMA(X) model and Bayesian estimation for the AR(X) part. To access and compare the quality of the estimations, we present all point estimates of unknown characteristics of the models studied.

In the simulation studies, it may be also useful to inspect the ratio

$$R_{pdf} = \frac{f\left({}^{[o]}\theta,\, {}^{[o]}r\right)}{f(\hat{\theta}, \hat{r})} \tag{8.1}$$

as another criterion. Here $f({}^{[o]}\theta,\, {}^{[o]}r)$ is the posterior pdf of the true regression coefficients ${}^{[o]}\theta$ and the true noise variance ${}^{[o]}r$, while $f(\hat{\theta}, \hat{r})$ give the resulting posterior pdf of the estimates.

Numerically, we evaluate

$$R_{pdf} = \exp\left\{\ln f({}^{[o]}\theta,\, {}^{[o]}r) - \ln f(\hat{\theta}, \hat{r})\right\} \tag{8.2}$$

instead. According to the form of GiW pdf in Chapter 4.2.2, (8.2) can then be specified as follows

$$R_{pdf} = \exp\left\{-0.5(\nu + n_\psi + 2)\ln\left(\frac{{}^{[o]}r}{\hat{r}}\right) - \frac{\left({}^{[o]}\theta - \hat{\theta}\right)' C^{-1}\left({}^{[o]}\theta - \hat{\theta}\right) + D_y}{2\,{}^{[o]}r} - \frac{D_y}{2\hat{r}}\right\}. \tag{8.3}$$

Generally, $0 \leq R_{pdf} \leq 1$. Obviously, the larger R_{pdf} is, the better quality the estimation achieve, or in another words, the closer to the true values the estimates are. Meanwhile the closer to zero R_{pdf} is, the sharper the pdf $f(\theta, r)$ of parameters is shaped.

Criterions of prediction Quality

On-step-ahead prediction is used to capture the dynamics. To judge the quality of the prediction, besides histogram prediction errors, we inspect the relative standard deviation of prediction errors

$$S_{PE} = \sqrt{\frac{\sum_{t=1}^{\mathring{t}} \hat{e}_t^2}{\sum_{t=1}^{\mathring{t}} (y_t - \bar{y})^2}}. \tag{8.4}$$

Where $\hat{e}_t \equiv y_t - \hat{y}_t$ defines prediction error, $\hat{y}$ is conditional expectation of output y_t. $\bar{y}_t \equiv \frac{1}{\mathring{t}} \sum_{t=1}^{\mathring{t}} y_t$ is sample mean of y_t and $\mathring{t}$ is the number of data samples.

Other used criterions

The MMQ/MAQ algorithm may be relatively slow in some cases. Therefore, the time needed for estimations and the number of iterations used (for MMQ and MAQ) are also measured and compared during experiments.

8.2 Basic Properties of MMQ and MAQ

This section aims at examining the basic properties of the proposed MMQ and MAQ algorithms together with their sensitivity to some implementation options. They are demonstrated by estimating a simulated ARMA model and a simulated output error model OE, respectively.

8.2.1 Estimating Simulated ARMA

Our first example uses MMQ and MAQ to estimate a simulated time series ARMA(2,2) with output part and MA part having orders $n_a = 2$ and $n_c = 2$, respectively. We also want to examine the influence of a starting point. Thus, several estimations are performed differing just in the starting points.

Simulated Process

The data $d(\mathring{t})$ of the length $\mathring{t} = 2000$ were simulated by the following SO ARMA process

$$y_t = 1.5y_{t-1} - 0.7y_{t-2} + e_t - 0.8e_{t-1} + 0.6e_{t-2}. \tag{8.5}$$

The variance $r = 0.1$ of the driving white Gaussian noise e_t was chosen. The plot of the outputs is given in Figure 8.1.(a).

Estimated Model and Implementation

The estimated model is selected as an ARMA(2,2) with the correct structure

$$y_t = a_1y_{t-1} + a_2y_{t-2} + e_t + c_1e_{t-1} + c_2e_{t-2}, \tag{8.6}$$

where e_t is a white noise sequence with zero mean and unknown variance r. Thus, regression coefficients are $\theta = [a_1, a_2]'$ and C-parameters vector is $C = [c_1, c_2]$,

Firstly, the MAQ method has been used for its estimation. This implies that ARMMA mixture structure is used in function evaluations. Since the order of the MA term is $n_c = 2$, a two-dimensional optimization problem is under consideration in the search of the C-parameters. Consequently, $n_c + 1$-vertex evolving simplex of the MDS search and $n_c + 1$-component ARMMA mixtures are used.

Then, the MMQ method has been used for its estimation by making use of $n_c + 1$-component MARMA mixture structure.

To test the sensitivity to starting points, 5 different representative starting points were selected. Therefore 5 estimations were performed for both methods, respectively. The tuning factors of the algorithms are specified as follows:

- Starting points

$$C_I^0 = (0 \quad 0)\,,\ C_{II}^0 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \end{pmatrix},\ C_{III}^0 = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \end{pmatrix},\ C_{IV}^0 = \begin{pmatrix} \frac{1}{3} & -\frac{1}{3} \end{pmatrix},\ C_V^0 = \begin{pmatrix} -\frac{1}{3} & -\frac{1}{3} \end{pmatrix}$$

- The option $\beta = [-0.2 \quad -0.1]$ with $h = 0.1$ is used for all of the ten experiments to generate the corresponding ten right-angle initial simplex according to (7.11)–(7.12).
- For stopping, we measure the relative size of simplex (7.13) with tolerance $\epsilon = 1e-004$. No limitations on the number of iterations and the likelihood of mixture were imposed in order to illustrate convergence properties of the algorithms.

Results and Analysis

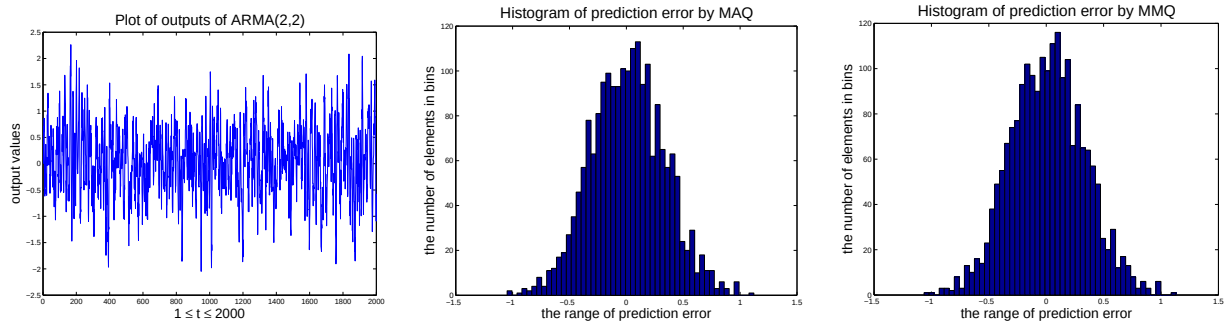
With one row describing the true parameters, the remaining 10 rows of Table 8.1 reflect the estimation and prediction results of the methods for the considered starting points. Since the different starting points lead to the similar results, we give histogram of prediction errors only for one of them. Figure 8.1 shows the histograms of the cases with the starting point (0 0) together with the plot of outputs.

	C^0		$\hat{\theta}$		$\hat{r}$	$\hat{C}$		N	T (min.)	R_{pdf}	S_{PE}
True	–		1.5	-0.7	0.1	-0.8	0.6	–	–	–	–
MAQ	0	0	1.4703	-0.6769	0.1035	-0.7674	0.6043	31	19.6017	0.0289	0.3953
MAQ	$\frac{1}{2}$	$\frac{1}{2}$	1.4703	-0.6769	0.1035	-0.7674	0.6043	26	15.6338	0.0289	0.3953
MAQ	$\frac{-1}{2}$	$\frac{1}{2}$	1.4703	-0.6769	0.1035	-0.7674	0.6043	31	19.3175	0.0289	0.3953
MAQ	$\frac{1}{3}$	$\frac{-1}{3}$	1.4704	-0.6770	0.1035	-0.7674	0.6043	30	19.5815	0.0293	0.3953
MAQ	$\frac{-1}{3}$	$\frac{-1}{3}$	1.4703	-0.6768	0.1035	-0.7672	0.6042	27	16.9677	0.0288	0.3953
MMQ	0	0	1.4721	-0.6815	0.1036	-0.7799	0.6206	23	14.2308	0.0262	0.3955
MMQ	$\frac{1}{2}$	$\frac{1}{2}$	1.4746	-0.6812	0.1035	-0.7741	0.6066	37	25.7372	0.0545	0.3953
MMQ	$\frac{-1}{2}$	$\frac{1}{2}$	1.4724	-0.6817	0.1036	-0.7799	0.6199	24	14.7896	0.0282	0.3955
MMQ	$\frac{1}{3}$	$\frac{-1}{3}$	1.4719	-0.6814	0.1036	-0.7799	0.6209	39	24.9709	0.0251	0.3955
MMQ	$\frac{-1}{3}$	$\frac{-1}{3}$	1.4729	-0.6819	0.1036	-0.7798	0.6188	23	14.4872	0.0315	0.3954

Table 8.1: MAQ and MMQ estimations and predictions of ARMA(2,2) with the different starting point C^0. $\hat{C}$ denotes the best vertex of the final simplex, i.e., the estimates of the C-parameters. $\hat{\theta} = E[\theta|d(t)]$ and $\hat{r} = E[r|d(t)]$ denote the estimates of the rest parameters. N is the number of iterations used. $T(min.)$ is the time needed in estimations. R_{pdf} is the ratio (8.3). S_{PE} is the relative standard deviation of prediction errors (8.4).

These results illustrate the promising properties and efficiency of the MAQ method and the MMQ method. Here, the results of the two methods are similar. However, the differences may become more obvious sometimes, we shall show one of such samples in the next group of experiments.

They also demonstrate that the convergence of the procedure is not significantly sensitive to the starting point and suggest that a good choice on the starting point may substantially speed up the search.



(a) (b) (c)

Figure 8.1: (a) Plot of outputs of the ARMA(2,2) process. (b) Histogram of prediction errors of MMQ estimation on ARMA(2,2) starting at (0 0). (c) Histogram of prediction errors of MAQ estimation on ARMA(2,2) starting at (0 0).

8.2.2 Estimating Simulated OE

This section applies the MMQ and MAQ estimations on an output error (OE) model. At the same time, we intend to examine the influence that the size of initial simplex may bring. Thus, two experiments are performed for each method with the size of the initial simplex varying.

Simulated OE process

2000 simulated data samples are available from the process

$$y_t = 1.4y_{t-1} - 0.49y_{t-2} + 0.9u_{t-1} + 0.7u_{t-2} + e_t - 1.4e_{t-1} + 0.49e_{t-2}, \tag{8.7}$$

where e_t is a Gaussian white noise sequence with zero mean and variance $r = 0.1$. The input u_t is modelled as a Gaussian white noise with variance $r_u = 1$. The plot of the outputs is shown in Figure 8.2.(a).

Estimated Model and Implementation

An ARMAX model with the correct structure is used in the estimation

$$y_t = a_1y_{t-1} + a_2y_{t-2} + b_1u_{t-1} + b_2u_{t-2} + e_t + c_1e_{t-1} + c_2e_{t-2}, \tag{8.8}$$

where e_t is a white noise sequence with zero mean and unknown variance r. It means that regression coefficients are $\theta = [a_1, b_1, a_2, b_2]'$ and C-parameters vector is $C = [c_1, c_2]$,

MAQ and MMQ were used to estimate its parameters, respectively. With the order of the MA $n_c = 2$, the number of vertices of simplex is therefore $n_c + 1 = 3$ and the number of components of the mixture used is $n_c+1 = 3$ as well. The tuning factors of the algorithms in the experiments were set as follows

- All experiments start at $C^0 = (0\ \ 0)$;
- For each method, different initial right-angle simplices with different size are created according to (7.11)–(7.12).

  One is generated by $\beta = [1\ \ 0.5]$ with $h = 0.5$, while the other smaller one is generated by $\beta = [0.4\ \ 0.2]$ with $h = 0.2$.

- Stopping is based on measuring the relative size of simplex (7.13) with tolerances $\epsilon = 1e-004$, without limitations on the number of iterations and likelihood of mixture.

Result and Analysis

Tables 8.2 and 8.3 display the estimation and prediction results for the OE process (8.7) by the MAQ estimations and MMQ estimations. In the MMQ estimation with $h = 0.5$, the estimates of C-parameters actually returned are $\hat{C} = [-1.8986 \quad 0.7881]$. In Table 8.2, the stable reflection $[-1.3909 \quad 0.4769]$ are listed instead to facility the comparisons. Again for simplicity, only the histograms of prediction errors for both estimation with $h = 0.5$ are showed in Figure 8.2 together with the plot of outputs..

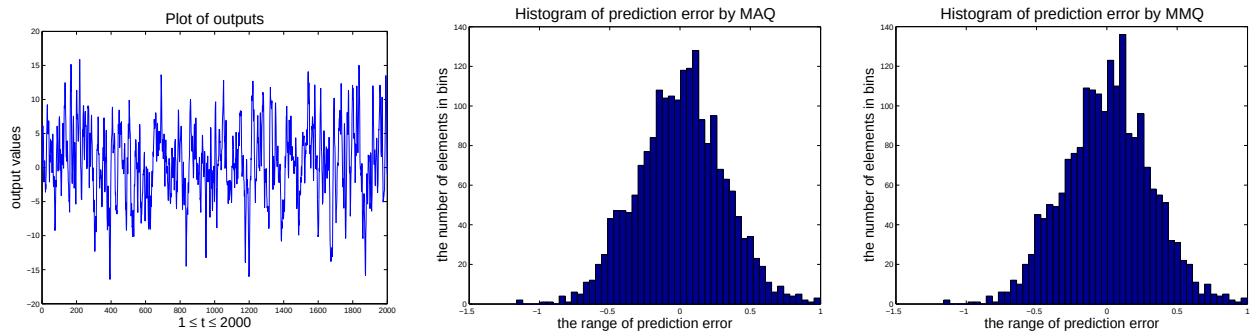
	C^0	h	$\hat{\theta}$				$\hat{r}$	$\hat{C}$	
True	–	–	1.4000	-0.4900	0.9000	0.7000	0.1000	-1.4000	0.4900
MAQ	0 0	0.5	1.4005	-0.4905	0.9041	0.6957	0.0942	-1.3901	0.4805
MAQ	0 0	0.2	1.4005	-0.4905	0.9042	0.6956	0.0942	-1.3914	0.4818
MMQ	0 0	0.5	1.4005	-0.4905	0.9043	0.6955	0.0943	-1.3909	0.4769
MMQ	0 0	0.2	1.4005	-0.4905	0.9042	0.6956	0.0953	-1.3927	0.4834

Table 8.2: Estimates of OE by MAQ and MMQ methods with the size of the initial simplex varying. The size is tuned by h. All estimations start from the point $C^0 = (0 \quad 0)$. $\hat{C}$ denotes the best vertex of the final simplex, i.e., the estimates of C-parameters. $\hat{\theta} = E[\theta|d(t)]$ and $\hat{r} = E[r|d(t)]$ denote the estimates of regression coefficients and noise variance.

	h	N	T (min.)	R_{pdf}	S_{PE}
MAQ	0.5	73	49.9124	0.0360	0.1323
MAQ	0.2	52	32.5655	0.0358	0.1323
MMQ	0.5	203	119.6252	0.0364	0.1323
MMQ	0.2	60	37.6186	0.0356	0.1323

Table 8.3: Other MAQ and MMQ estimation and prediction results on OE with the size of the initial simplex varying. The size is determined by h. N is the number of iterations used. $T(min.)$ is the time needed in estimations. R_{pdf} is the ratio (8.3). S_{PE} is the relative standard deviation of prediction errors (8.4).

This example shows that the MMQ and MAQ methods behave well on the estimations of the above OE model. It also demonstrates that the size of the initial simplex may directly affect the speed of the search. For example, Table 8.2 shows that the MMQ estimation with $h = 0.5$ consumed the time which is around three times longer than the others do.



(a) (b) (c)

Figure 8.2: (a) Plot of outputs. (b) Histograms of prediction errors of OE by MAQ estimation. (c) Histograms of prediction errors of OE by MMQ estimation.

8.3 Comparisons of MMQ, MAQ and PEB Methods

The example of this section considers a simulated SO ARMAX(3,3,3) with all roots of the C-polynomial lying on the unit circle.

The purpose here is to demonstrate the modelling properties of MARMAX models and ARMMAX models in their ability to provide the parallelism and compare the efficiency of their corresponding estimations with the PEB method.

8.3.1 Simulated ARMAX Process

The data samples $d(\mathring{t})$ of the length $\mathring{t} = 2000$ were simulated by the following SO ARMAX(3,3,3) process,

$$y_t = -1.8y_{t-1} + 1.5y_{t-2} - 0.5y_{t-3} + u_{t-1} + 0.7u_{t-2} + 0.4u_{t-3} + e_t - 3e_{t-1} + 3e_{t-2} - 1e_{t-3}. \quad (8.9)$$

The variance $r = 0.1$ of the driving white Gaussian noise e_t was chosen. All three roots of its cC-polynomial are located at the stability boundary. Figure 8.3 shows the plot of its outputs.

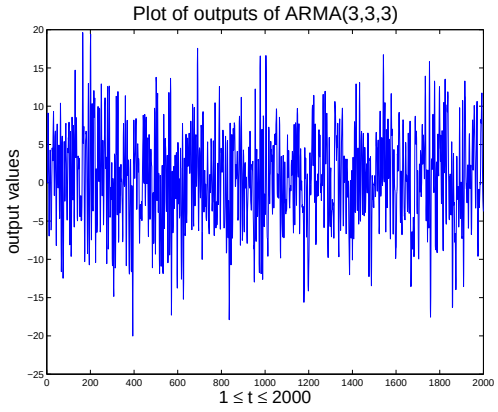


Figure 8.3: Plot of the output of the simulated ARMAX(3,3,3) process.

8.3.2 Estimated Model and Implementation

The following ARMAX model with the correct structure

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + a_3 y_{t-3} + b_1 u_{t-1} + b_2 u_{t-2} + b_3 u_{t-3} + e_t + c_1 e_{t-1} + c_2 e_{t-2} + c_3 e_{t-3} \quad (8.10)$$

is used in all estimations of this section.

For implementation of the MMQ and MMQ estimations, simplex with $n_c + 1 = 4$ vertices and mixtures with $n_c + 1 = 4$ components are defined. The tuning factors are specified as follows:

- Both estimations start at the point $C^0 = (0 \;\; 0 \;\; 0)$;
- For both cases, the option $\beta = [1.5 \;\; 1.5 \;\; 0.5]$ with $h = 0.5$ is used to generate a right-angle 4-vertex initial simplex, according to (7.11) – (7.12).
- Stopping is based on measuring the relative size of simplex (7.13) with tolerances $\epsilon = 1e-004$, without the limitations on the number of iterations and likelihood of mixture.

while for implementation of the PEB estimation

- the routine `arma.m` in Matlab system identification toolbox of is used to perform PE estimation of C-parameters,
- the default initial parameter values constructed in a special four stage LS-IV algorithm are used.
- iterative Bayesian estimation method is used for the rest estimation based on the extended LD filter.

8.3.3 Result and Analysis

The estimates are listed in Table 8.4, whilst the rest estimation and prediction results are depicted in Table 8.5 and Figure 8.4. In the MAQ estimation, the estimates of C-parameters actually returned are $\hat{C} = [-3.5625 \;\; 3.9375 \;\; -1.3754]$, we list its stable reflection $[-2.3206 \;\; 1.7462 \;\; -0.4254]$ instead to facility the comparisons.

	Initial C^0	$\hat{\theta}$						$\hat{r}$	$\hat{C}$		
True	–	1.8000	-1.5000	0.5000	1.0000	0.7000	0.4000	0.1	-3	3	-1
MAQ	0 0 0	1.7922	-1.4946	0.4992	0.9975	0.7070	0.4291	0.1486	-2.3206	1.7462	-0.4254
MMQ	0 0 0	1.7982	-1.4989	0.5001	0.9976	0.7079	0.4021	0.0940	-3.0000	3.0000	-1.0000
PEB	–	1.7924	-1.4947	0.4988	0.9990	0.7006	0.4355	0.1930	-2.1525	1.6504	-0.4470

Table 8.4: Estimates of ARMAX(3,3,3) by the MMQ, MAQ and PEB methods. Both MMQ and MAQ methods start from the point $C^0 = (0 \;\; 0 \;\; 0)$. $\hat{C}$ denotes the best vertex of the final simplex, i.e., the estimates of C-parameters. $\hat{\theta} = E[\theta|d(t)]$ and $\hat{r} = E[r|d(t)]$ denote the estimates of regression coefficients and noise variance.

	N	T	R_{pdf}	S_{PE}
MAQ	24	38.6873 (min.)	2.8360e-074	0.1605
MMQ	20	33.0786 (min.)	4.6015e-004	0.1300
PEB	–	22.8569 (sec.)	3.1017e-189	0.1823

Table 8.5: Other estimation and prediction results of MMQ, MAQ and PEB methods on ARMAX(3,3,3). N is the number of iterations used. T is the time needed in estimations. R_{pdf} is the ratio (8.3). S_{PE} is the relative standard deviation of prediction errors (8.4).

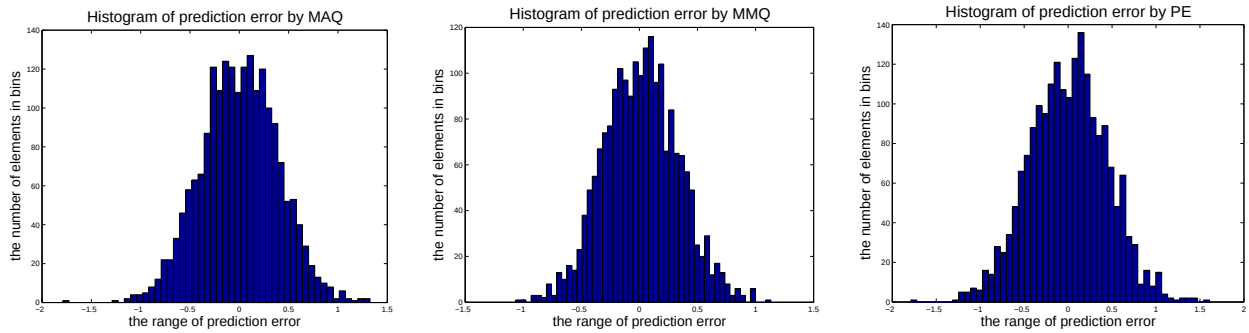


Figure 8.4: Histograms of prediction errors of MMQ, MAQ and PEB methods on ARMAX(3,3,3).

These results further confirm good properties of the MMQ estimation and reveal the limitation of the MAQ estimation. It reflects the influence of the joint updating in the ARMMAX-QB estimation and indicates that the stable replacement operation may be not efficient enough.

This example also shows the limitation of the PEB method on the stability boundary and illustrates the slowness of the MMQ and MAQ as well. However, it has to be stressed here that the software used for implementation of the MMQ is currently in m-files of Matlab, the memory and speed of the computation is therefore limited. A significant improvement in this aspect can be expected when using mex-files instead.

In addition, the extremely small values of the radio R_{pdf} (8.3) indicate that the pdf $f(\theta, r)$ is sharply shaped with its maximum $f(\hat{\theta}, \hat{r})$ sitting at a quite high value. For instance, the MMQ estimation leads to $\ln f(\hat{\theta}, \hat{r}) = 1.5649e + 003$. Thus it is not surprising than such a small radio as $R_{pdf} = 2.8360e - 074$ is obtained.

8.4 Transportation Problem

The performance of MMQ, MAQ and PEB methods using real data is illustrated in this example. With an increasing number of cars, the traffic transportation problem become urgent. The process under analysis here is the density of a traffic flow of city crossroads.

8.4.1 Data Description

The data are measured by detectors under the road surface. Each detector gives information about time periods when it is occupied by cars and periods when it is free. Using the information provided by a detector and an average length of cars, the number of cars per kilometer of the traffic flow, i.e., density ρ, can be computed. This variable describes the traffic state at the detector position and will finally be used in traffic control for a better utilization of the available traffic lights.

Outlier filtration, normalization to zero mean and standard deviation one were applied on the source data. Figure 8.5 displays preprocessing data samples ρ of the length 2000, which are measured along the traffic lanes of the Strahov tunnel in Prague.

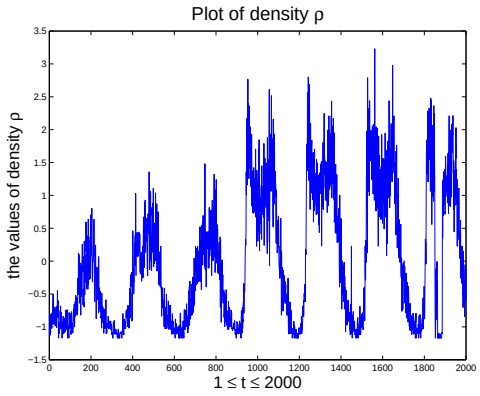


Figure 8.5: Plot of traffic density ρ.

8.4.2 Estimated Model and Implementation

An ARMA(1,1) model

$$y_t = a_1 y_{t-1} + e_t + c_1 e_{t-1} \tag{8.11}$$

is used in the estimations by means of MMQ method, MAQ method and PEB method, respectively.

The MMQ and MAQ estimations are implemented as follows

- start at the point $C^0 = 0$;
- using simplex with $n_c + 1 = 2$ vertices and mixtures with $n_c + 1 = 2$ components.
- The option $\beta = 0.2$ with $h = 0.2$ is used to generate a right-angle 2-vertex initial simplex according to (7.11) – (7.12) for each method, respectively.
- Stopping is based on measuring the relative size of simplex (7.13) with tolerances $\epsilon = 1e - 004$, without the limitations on the number of iterations and likelihood of mixture.

While for implementation of the PEB estimation

- the routine `arma.m` in the Matlab toolbox is used to perform PE estimation of C-parameters,
- the default initial parameter values constructed in a special four stage LS-IV algorithm are used,
- iterative Bayesian estimation method is used for the remaining estimation based on the extended LD filter.

8.4.3 Results and Analysis

The estimation and prediction results of MMQ, MAQ and PEB methods are listed in Table 8.6, whilst histograms of prediction errors are depicted in Figure 8.6.

	$\hat{C}$	N	T	s_{PE}
MAQ	-0.6492	16	2.6663 min.	0.3583
MMQ	-0.6749	16	2.5296 min.	0.3587
PEB	-0.6502	–	11.9810 sec.	0.3583

Table 8.6: Estimation and prediction results of MMQ, MAQ and PEB methods on the real data. $\hat{C}$ is the point estimates of C-parameters. N is the number of iterations used. T is the time needed in estimations. S_{PE} is relative standard deviation of prediction errors (8.4).

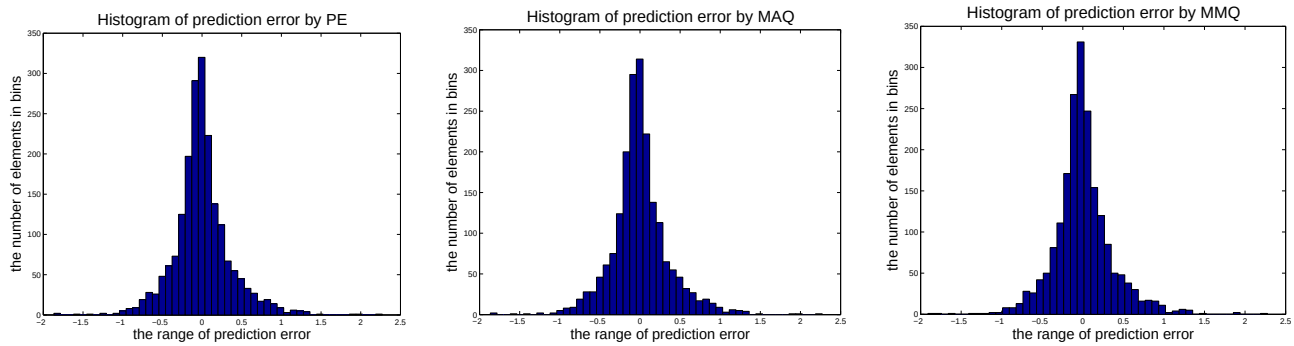


Figure 8.6: Histograms of prediction errors of MMQ, MAQ and PEB methods on the real data.

The results of Table 8.6 show that the C-polynomial is well located within the stability area, it is then not surprising that the three methods give similar results.

Remarks 8.4.1

Note that in the estimations we used the simplest first order ARMA(1,1) model, which was found to be no worse than higher order ARMA models. However, the plot of prediction errors shown in Figure 8.7 indicates that a richer model structure, such as mixture models, has a space for further improvements.

8.5 Summary

In summary, our experiments of this chapter show that

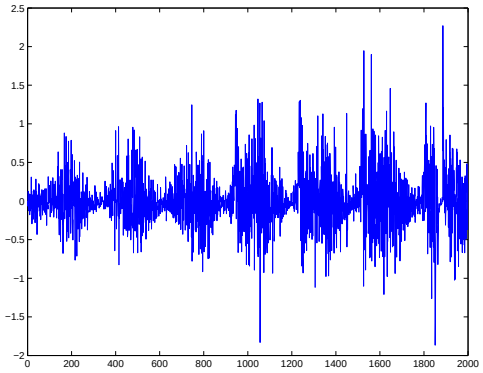


Figure 8.7: Plot of the prediction errors $\hat{e}_t$ in the MAQ estimation.

- MMQ method performs well generally.
- MAQ method has the limitations with respect to its joint updating.
- PEB method has the limitations with respect to its stability requirement on C-parameters. Both the MMQ and MAQ method perform better than PEB method does around the stability boundary.
- MAQ and PEB methods, despite the limitations, still deserves attention with their certain efficiency confirmed by the experiments.
- MMQ and MMQ methods are not significantly sensitive to initial simplex with respect to convergence. However, a poor choice on the initial simplex may have a negative impact, possibly significant, on their performance in terms of time and iteration number.

Chapter 9

Conclusions and Future Work

The thesis concerns dynamic mixture modelling and Bayesian estimation in the presence of colored stochastic disturbances. Although some progress has been made in the past two decades, Bayesian solution to ARMA(X) models remains a long standing open question. Consequently, the use of ARMA(X) mixtures is hindered as well.

The thesis made further steps towards a Bayesian scheme covering ARMA(X) models and the corresponding mixtures. It extends the established scheme of AR(X) models and AR(X) mixtures.

9.1 Main Results

Preliminary experimental results demonstrated and confirmed the promising and respectable properties of the underlying theory and related algorithms. The success is due to the efforts in theoretical and practical aspects as follows:

- A reexamination on Bayesian modelling from the view of probabilistic dynamic mixtures.

  A restricted description of dynamic mixtures was introduced and interpreted. We showed its feasibility and limitation.

  All models considered in the thesis were then described within such a unifying mixture modelling framework. In particular,

  - We stressed how to understand relaxing stability on the MA part for an ARMA(X) model in modelling.
  - We not only considered MARMAX model as a natural mixture generalization of ARMAX model but also introduced ARMMAX models as a novel system description tool.

- A reexamination on the LD type filters.

  The idea of LD type filters is not new. However, the results of Peterka were mainly based on a proper chosen state-space representation. For our purpose, we reexamined the results based on regression form of ARMA(X) models and clarified a few related issues which are important in the estimations of Chapter 6 and 7.

- An investigation on the MARMAX and ARMMAX models.

  Efficient MARMAX-QB and ARMMAX-QB estimations have been developed to estimate these two ARMAX mixtures when the C-parameters are given. Benefited from the extended LD

filter, the estimations do not impose any constraint on the stability of the MA parts. Meanwhile, they require feasible computation load, which increases linearly with the number of components and is close to several recursive least-squares estimations.

We also showed the flexibility and limitations of these two mixtures in system description,

- We proved an ARMMA(X) model is indeed richer than an ARMA(X) model.
- We showed their approximate parallelism, i.e., they provide a quantitative measure of descriptive quality of ARMAX components in parallel. Preliminary analysis was provided to justify this claim.
- The limited parallelism of ARMMAX models with its joint updating in the estimation was revealed and the stable replacement operation was proposed as a possible solution.

- MMQ, MAQ and PEB methods were proposed to provide improved Bayesian estimation of an ARMA(X) model. They preserve Bayesian estimation setting up to the ARX part of the model and reduce the uncertainty of the MA part by searching for a point estimate of the C-parameters.

  In particular, due to the successful choices of the essential ingredients, namely MDS optimization method and ARMA(X) mixtures, the MMQ and MAQ methods have several favorable properties:

  - They are able to cope with C-polynomial with its roots even at stability boundary. This property of the adopted extended LD filter makes them outperform the PEB method at this aspect.
  - They are able to cope with C-parameters even in high dimensions (≥ 2). The strong convergence analysis of the adopted MDS search makes them outperform alternative methods. For instance, they are more reliable than a method based on the NM simplex search at this aspect.
  - They enable parallel search acceptable on a single-processor machine by taking advantage of the approximate "algorithmic" parallelism of the mixtures. Thus, they can be considered as one of off-line applications of ARMAX mixtures. It actually opens up a novel possible application of mixtures in parallel computing field as well.
  - The richness of ARMAX mixtures may bring potential benefit to provide a valuable model even if the search in the C-parameters space is stopped before it converges finally. Stopping may be enforced by a slow terminal convergence of the MDS method or by computational demands implied by the extensive data set processed.

- The thesis clarified the relationships among all models considered: AR(X), ARMA(X), mixture of AR(X), MARMA(X) and ARMMA(X).

- Illustration and Experiments.

  A significant amount of work was devoted to implement the proposed algorithms.

  - A few implementation rules were proposed to complement the rules of the standard MDS method in our parameter estimation setting.
  - A software package was developed in the Matlab language.

9.2 Future Work

The above discussion also highlights where further efforts should be made. We list some of them below as the subjects of future work.

General Dynamic Mixture Modelling

Although our restricted description of dynamic mixtures (3.6) is acceptable, it is desirable to remove its assumption on constant mixing weights for asymptotic goodness.

Consider all input-output data contained in the observed data vector $\Psi_t = [y_t, \psi_t']$ as a whole

$$f(\Psi_t|\Theta), \tag{9.1}$$

and approximate it by a static mixture using the universal approximation property

$$f(\Psi_t|\Theta) \quad = \quad \sum_{p=1}^{n_p} \alpha_p f(\Psi_t|\Theta_p, p), \tag{9.2}$$

with constant weights α_p and static components $f(\Psi_t|\Theta_p, p)$.

Using chain rule (2.4), the following relation then holds

$$f(y_t|\psi_t, \Theta) \quad = \quad \frac{\sum_{p=1}^{n_p} \alpha_p f(y_t|\psi_{p;t}, \Theta_p, p) f(\psi_t|\Theta_p, p)}{f(\psi_t|\Theta)}, \tag{9.3}$$

If we denote

$$\bar{\alpha}_p(\psi_t) = \frac{\alpha_p f(\psi_t|\Theta_p, p)}{f(\psi_t|\Theta)}, \tag{9.4}$$

then (9.3) can be rewritten in terms of data-dependent weights $\bar{\alpha}_p(\psi_t)$

$$f(y_t|\psi_t, \Theta) \quad = \quad \sum_{p=1}^{n_p} \bar{\alpha}_p(\psi_t) f(y_t|\psi_{p;t}, \Theta_p, p). \tag{9.5}$$

Obviously, with the data-dependent weights $\bar{\alpha}_p(\psi_t)$ and the dynamic components $f(y_t|\psi_{p;t}, \Theta_p, p)$, (9.5) gives a description of dynamic mixtures. What's more, such a modelling also opens up a promising prospect to tackle the corresponding estimation.

Extended MMQ/MAQ Algorithm

In order to have the MMQ/MAQ algorithm fully ready for routine real applications, more elaborated implementation and to further take advantage of the power of mixtures are needed.

The recent progress in optimization area could help use to improve the MDS method, for example to used sequential multidirectional search or some hybrid methods.

By exploring the power of mixture, we mean to increase algorithmic parallelism by using ARMAX mixtures with more than $n_c + 1$ components. It is similar to the attempts to use all available processors or scaling the algorithm to fit the properties of a given machine in parallel optimization problems. In 1991, Denis [40] investigated how to generate algorithms which is able to use any number of processors for direct search, in particular multidirectional search. The idea there would throw light on our attempt to increase algorithmic parallelism of ARMMAX.

However, a significant amount of work is then added in each iteration. For parallel processors, it is distributed among the available processors, thus it speeds up the procedure and finally one can benefit from reducing the execution time. In our case, we are doing parallel computing on a single processor. Thus, if we want to employ the above idea to increase the parallel ability of mixtures, it requires caution since we are not sure how much we can benefit in execution time.

Generally it would be more reasonable for our mixture to visit the proper number of points instead of as many points as possible. For instance, we could use one $2n_c + 1$-component mixtures for the evaluation of likelihoods in both reflection and expansion steps instead of two $n_c + 1$-component mixtures for each step respectively.

Refining MAQ Estimation

Based on the preliminary analysis, the parallelism of ARMA(X) mixtures appears to be feasibly successful in the experiments. However, the full theoretical analysis on their parallelism is still lacking. Especially a more efficient way to deal with the influence of the joint updating in the MAQ estimation is preferable.

Rather than the proposed stable replacement operation in Chapter 6.3, another potential solution is to use a re-scaling as following:

As defined earlier, a partial covariance matrix S of an ARMA(X) model is

$$S_{t,t\pm i} \quad = \quad s_i = \sum_{k=i}^{n_c} c_i c_{k-i}, \qquad\qquad \text{for} \quad i = 0, 1, \cdots, n_c, \quad c_0 = 1$$

It can always be normalized so that its principal diagonal elements equal to unit, i.e.,

$$s_0 = 1 \;\text{ or }\; S_{i,i} = 1, \;\; i = 1, \cdots, t$$

using a scaling

$$S \to S/s_0 \tag{9.6}$$

In effect, it amounts to a re-scaling in noise term, more exactly in noise variance r_e

$$r_e \to r_e s_0 \tag{9.7}$$

or re-scaling in the C-parameters $C = [1, c_1, \cdots, c_{n_c}]$

$$C \to C/\sqrt{s_0} \tag{9.8}$$

Note that, in the context of the multidirectional search, such a re-scaling may create danger because it alerts the angles of the simplex during the searching in C-parameters Cartesian space. Thus it destroys one of important features of MDS method that the shape (the angles) of the simplex keeps unchanged during the moving.

To preserve this property, a polar transformation could be used here. The main idea is to represent the C-parameters in the polar coordinates and perform the MDS search in the polar coordinates space instead of the Cartesian space.

Recall that in multivariate integral analysis, see for example [37], any point $x = [x_1, \cdots, x_m]$ in a m-dimensional Cartesian space can be identified by its distance from the origin, ρ, together with the angles $\phi_1, \cdots, \phi_{m-1}$ as a polar representation $(\rho, \phi_1, \cdots, \phi_{m-1})$, according to the

following relations:

$$\begin{aligned} x_1 &= \rho\cos\phi_1 \\ x_2 &= \rho\sin\phi_1\cos\phi_2 \\ &\vdots \\ x_{m-1} &= \rho\sin\phi_1\cos\phi_2\cdots\sin\phi_{m-2}\cos\phi_{m-1} \\ x_m &= \rho\sin\phi_1\cos\phi_2\cdots\sin\phi_{m-2}\cos\phi_{m-1}, \end{aligned}$$

where $\rho \geq 0$, $\phi_i \in (\frac{-\pi}{2}, \frac{\pi}{2})$ for $i = 1, \cdots, \phi_{m-2}$ and $\phi_{m-1} \in (-\pi, \pi)$.

With these relations, the above re-scaling can be equivalently expressed by polar representation, with only n_c unknown angles ϕ needed to represent the $n_c + 1$ dimensional vector $C/\sqrt{s_0}$, since the norm is 1 such that $\rho = 1$. Thus we can perform the MDS search in the the space n_c-dimensional ϕ space instead of the $n_c + 1$-dimensional Cartesian space. In this way, we preserve the shape of simplex unchanged.

The use of the polar representation can be easily embedded into the the procedure of MAQ described in Chapter 7.1.3:

- Give an initial guess of C-parameters in terms of polar coordinates , i.e., the angles $\phi^0 = (\phi_1, \cdots, \phi_{m-1})$.
- Create an initial simplex in terms of polar coordinates. It is then transformed into Cartesian coordinates according to the formula given above.
- Feed the simplex in Cartesian representation into the ARMMAX-QB estimation to make function evaluations.
- Determine the best vertex according the component log-likelihoods.
- Generate new trial points, such like reflection, expansion, contraction points, in terms of polar coordinates ϕ^r, ϕ^e, ϕ^c.
- Transform these trial points into the Cartesian coordinates according to the formula given above and feed them into the ARMMAX-QB estimation to make function evaluations.
- Determine the best vertex according the results of estimation.
- Determine the best vertex according the component log-likelihoods.
- If the stopping rules are not satisfied, repeat the procedure.

Dealing with Multivariate ARMAX Models

As remarked earlier, for the estimation of ARMAX models, when the assumption that the MA part is known can be made, the majority results of Chapter 5 are directly applicable on its MO extension, see [6]. When such an assumption cannot be made, the MO extension of Chapter 6 could be obtained by an approach studied in [16], where a MO linear system was described by a collection of SO models and thereafter the identification was made based on such kind of entry-wise predicting models.

Generalized Use of ARMMAX Models

Another important issue is the generalized use of ARMMAX model. Although it is not required by our definition of ARMMAX model, the study of ARMMAX models in the thesis has indeed been mainly based on the Peterka filters to realize the temporal variations of the stochastic MA part.

Actually, the modelling power of ARMMAX and its gained algorithmic parallelism can be extended further on. Different choices of the types of filters leaded to various filtering properties. For instance, the Peterka filters can be partially or fully replaced by other types of filters [41]. It opens a way to a wide set of novel adaptive filters dealing with outliers, with temporarily varying measurement noise etc. Practical impact of such possibilities can hardly be over-stressed.

Appendix A

Mixtures at Factor Level

By chain rule, the distribution of data $y(\mathring{t})$ with multiple modes can be expressed as

$$f(y(\mathring{t})|\Theta) \equiv \prod_{t\in t^*} f(y_t|u_t,\phi_{t-1},\Theta), \quad t^* \equiv \{1,2,...,\mathring{t}\}$$

with finite parametric mixtures forming each individual parameterized models

$$f(y_t|u_t,\phi_{t-1},\Theta) \equiv \sum_{p\in p^*} \alpha_p f(y_t|u_t,\phi_{p;t-1},\Theta_p,p), \quad p^* = \{1,\ldots,n_p\},\ n_p<\infty, \tag{A.1}$$

where, $\alpha_p \geq 0,\ \sum_{p\in p^*}\alpha_p = 1$, and

$$f(y_t|u_t,\phi_{p;t-1},\Theta_p,p) \tag{A.2}$$

describes the p-th component of the mixture. Here the state $\phi_{p;t}$ is newly introduced and recursively updated using new data d_t

$$\phi_{p;t} = \Phi_p(\phi_{p;t-1},d_t). \tag{A.3}$$

The overall parameters Θ of the mixture are formed by the component weight α and the parameters of the individual components Θ_p

$$\Theta \equiv \left\{\alpha=[\alpha_1,\cdots,\alpha_{n_p}],\{\Theta_p\}_{p=1}^{n_p}\right\}.$$

When multi-output are considered, the chain rule implies that the above component level description of mixtures could be refined further. For a n_y-dimensional output y_t, each component (A.2) can actually be further split as a product of pdfs predicting its individual entries,

$$f(y_t|\phi_{p;t-1},\Theta_p,p) \equiv \prod_{i\in i^*} f(y_{ip;t}|\psi_{ip;t},\Theta_{ip},p), \tag{A.4}$$

where the pdfs, for $p\in p^*=\{1,\ldots,n_p\}$, $i\in i^*=\{1,2,\cdots,n_y\}$

$$f(y_{ip;t}|\psi_{ip;t},\Theta_{ip},p) \tag{A.5}$$

are so-called *parameterized factors*. In particular, a factor occurring in several components is called *common parameterized factor*. The additional subscript i of the parameter Θ_{ip} here

indicates that only some entries of Θ_p may occur in i-th factor. Similarly, the vector $\psi_{ip;t}$ is generally a sub-vector of the vector

$$[y_{(i+1)p;t}, y_{(i+2)p;t}, \cdots, y_{n_y p;t}, u_t, \phi'_{p;t-1}]'. \tag{A.6}$$

It is obvious that description of dynamic mixtures at factor level could bring some additional freedom, namely

- to provide more flexibility in the parametric description,
- to open a way to use different models for different entries of output y_t,
- to describe jointly continuous and discrete valued variables,
- to respect common dependencies reflected in several components.

Appendix B

QB Estimation at Factor Level

As a complement of Section 4.3.3, here we describe the corresponding QB estimation of mixtures at factor level. The more detailed discussion and the full treatment on the topic, see for example [18].

Algorithm B.0.1 (QB estimation at factor level)

Initial (off line) mode

- *Select the complete structure of the mixture, i.e. specify the number of components n_p and the ordered lists of factors allocated to the considered components. The structure of the factor labelled by ip is determined by the structure of the corresponding data vector Ψ_{ip}.*
- *Select prior pdfs $f(\Theta_{ip})$, $p \in p^*$ of the individual factors, ideally, in the conjugate form, with respect to the parameterized factors $f(y_{ip;t}|\psi_{ip;t}, \Theta_{ip}, p)$.*
- *Select prior pdfs of component weights α in the form of Dirichlet,*

$$f(\alpha) = Di_{\alpha}(\kappa)$$

and specify the initial values $\kappa_{p;0} > 0$.

Sequential (on line) mode,

1. *Evaluate the point estimates of the mixing weights α_p of previous time instant $t-1$,*

$$\hat{\alpha}_{p;t-1} \equiv \mathcal{E}[\alpha_p|d(t)] = \frac{\kappa_{p;t-1}}{\sum_{\tilde{p}\in p^*} \kappa_{\tilde{p};t-1}}.$$

2. *Acquire the data record d_t.*
3. *Compute the values of the predictive pdfs for each individual factor $i \in i^* = \{1, \ldots, n_y\}$ in all components $p \in p^*$*

$$f(y_{ip;t}|\psi_{ip;t}, d(t-1), p) = \int f(y_{ip;t}|\psi_{ip;t}, \Theta_{ip}) f(\Theta_{ip}|d(t-1))\, d\Theta_{ip} = \frac{\mathcal{I}(d(t)|ip)}{\mathcal{I}(d(t-1)|ip)}, \quad \text{(B.1)}$$

using (2.21) and the data d_t.

A trial updating of statistics with neither data weighting nor stability forgetting is performed in the evaluation here. In the case of normal, a partitioned $L'DL$ decomposition of the statistic $V_{p;t}$ has to be used to counteract the numerical troubles.

4. *Compute values of the predictive pdfs for each individual component* $p \in p^*$

$$f(y_t|u_t, d(t-1), p) = \prod_{i \in i^*} f(y_{ip;t}|\psi_{ip;t}, d(t-1), p). \tag{B.2}$$

5. *Compute the probabilistic weights*

$$w_{p;t} = \frac{\hat{\alpha}_{p;t-1} \prod_{i \in i^*} f(d_{ip;t}|\psi_{ip;t}, d(t-1), p)}{\sum_{\tilde{p} \in p^*} \hat{\alpha}_{\tilde{p};t-1} \prod_{i \in i^*} f(d_{i\tilde{p};t}|\psi_{i\tilde{p};t}, d(t-1), \tilde{p})}, \tag{B.3}$$

   they assign the weights to the data, which are used in the updating the statistics for each component.

6. *Update a posteriori* Dirrichlet *pdf of mixing weights by the evolution of the scalars* $\kappa_{p;t+1}$

$$\kappa_{p;t} = \kappa_{p;t-1} + w_{p;t}.$$

7. *Update a posteriori* pdfs $f(\Theta_p|d(t+1))$ *of the parameters associated with individual factors according to the weighted Bayes rule. Here two situations may encountered:*

   - *When there are no common factors in components,*

$$f(\Theta_{ip}|d(t)) \propto [f(y_{ip;t}|\psi_{ip;t}, \Theta_{ip}, p)]^{w_{p;t}} f(\Theta_{ip}|d(t-1)). \tag{B.4}$$

   - *When there are common factors in components,*

$$f(\Theta_{ip}|d(t)) \quad \propto \quad [f(y_{ip;t}|\psi_{ip;t}, \Theta_{ip}, p)]^{w_{ip;t}} f(\Theta_{ip}|d(t-1)), \tag{B.5}$$

     where $w_{ip;t} = \sum_{\tilde{p} \in p_i^*} w_{\tilde{p};t}$, *with* p_i^* *is the set of pointers, which label the components containing the* i*-th factor.*

8. *Evaluate, if need be, the characteristics of* $f(\Theta_{ip}|d(t))$ *describing other parameters* Θ_{ip}.

9. *Repeat the sequential mode, when* $t \leq \mathring{t}$.

Bibliography

[1] L. Ljung. *System Identification-Theory for the User*. D. van Nostrand Company Inc., Prentice-hall. Englewood Cliffs, N.J, 1987.

[2] T. Söderström and P. Stoica. *System Identification*. Prentice-hall. Englewood Cliffs, N.J, 1989.

[3] B. Bercu. Weighted estimation and tracking for ARMAX models. *SIAM Journal on Control and Optimization*, 33(1):89–106, 1995. ISBN 0306-0129.

[4] V. Peterka. Bayesian approach to system identification. In P. Eykhoff, editor, *Trends and Progress in System identification*, pages 239–304. Pergamon Press, Oxford, 1981.

[5] M. Kárný and R. Kulhavý. Structure determination of regression-type models for adaptive prediction and control. In J.C. Spall, editor, *Bayesian Analysis of Time Series and Dynamic Models*. Marcel Dekker, New York, 1988. chapter 12.

[6] V. Peterka. Algorithm for LQG self-tuning control based on input-output delta models. *IFAC Workshop on Adaptive Systems in Control and Signal Processing*, 1986.

[7] V. Peterka. Self–tuning control with alternative sets of uncertain process models. In *Proceedings of IFAC Symposium on Adaptive Systems in Control and Signal Processing*, pages 409–414. Glasgow, UK, 1989.

[8] D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixtures*. John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1985. ISBN 0 471 90763 4.

[9] R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26, 1984.

[10] A. P. Dempster, N. M. Lair, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39, 1977.

[11] S. Richardson and P.J. Green. On Bayesian analysis of mixtures with an unknown number of components, with discussion. *Journal of the Royal Statistical Society, Series B*, 59(4):731–792, 1997.

[12] M. Kárný, J. Kadlec, and E. L. Sutanto. Quasi-Bayes estimation applied to normal mixture. In *Preprints of the 3rd European IEEE workshop on Computer-Intensive Methods in Control and Data Processing, J. Rojíček, M. Valečková, M. Kárný and K. Warwick, Eds.*, pages 77–82. ÚTIA AV ČR, Prague, September 1998.

[13] S. Haykin. *Neural Networks – A comprehensive Foundation*. Macmillam College Publishing Company Inc., New York, Toronto, Oxford, Singapore, Sydney, 2000.

[14] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM journal on optimization*, 1:123–145, 1991.

[15] V. Torczon. *Multi-directional Search: A Direct Search Algorithm for Parallel Machines*. Ph.D thesis, Rice University, Houston, Texas, USA, 1989.

[16] M. Kárný. Parametrization of multi-output autoregressive regressive models for self-tuning control. *Kybernetika*, 28(5):402–412, 1998.

[17] V. Peterka. Control of uncertain processes: applied theory and algorithms. *Kybernetika*, 22, 1986. Supplement to No. 3, 4 ,5, 6.

[18] M. Kárný, J. Böhm, T.V. Guy, L. Jirsa, I. Nagy, P. Nedoma, L. Tesař, and M. Tichý. Prodactool background – theory, algorithms and software. Technical report, UTIA AV CR, 2003. Draft of the report, 401 pp.

[19] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–87, 1951.

[20] E. L. Sutanto and K. Warwick. Cluster analysis: An intelligent system for the process industries. In Robert Trappl, editor, *Cybernetics and Systems '94*, volume 1, pages 327–344, Vienna, 1994. World Scientific.

[21] M. H. Wright. Direct search methods: Once scorned, now respectable. *Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis*, pages 191–208, 1995.

[22] R. M. Lewis, V. Torczon, and M. W. Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124:191–207, 2000.

[23] S. L. S. Jacoby, J. S. Kowalik, and J. T. Pizzo. *Iterative Methods for Nonlinear Optimization Problems*. Prentive-Hall, Inc., Englewood Cliffs, New Jersaey, 1972.

[24] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[25] G. R. Hext W. Spendley and F. R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4:441–461, 1962.

[26] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[27] The Math works. *Matlab Optimization Toolbox*. Natick, Massachusetts, USA, 2001.

[28] J. L. Doob. *Stochastic Processes*. John Wiley & Sons, New York, London, 1953.

[29] M. Loeve. *Probability Theory, 4th edition*. Springer Verlag, 1977.

[30] R. Kulhavý and M. B. Zarrop. On general concept of forgetting. *International Journal of Control*, 58(4):905–924, 1993.

[31] R. Koopman. On distributions admitting a sufficient statistic. *Transactions of American Mathematical Society*, 39:399, 1936.

[32] E. F. Daum. New exact nonlinear filters. In J.C. Spall, editor, *Bayesian Analysis of Time Series and Dynamic Models*. Marcel Dekker, New York, 1988.

[33] G.J. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York, 1977.

[34] M. Kárný, N. Khailova, J. Böhm, and P. Nedoma. Quantification of prior information revised. *International Journal of Adaptive Control and Signal Processing*, 15(1):65–84, 2001.

[35] M. Kárný, P. Nedoma, and M. Nagy I. Valečková. Initial description of multi-modal dynamic models. In V. Kůrková, R. Neruda, M. Kárný, and N. C. Steele, editors, *Artificial Neural Nets and Genetic Algorithms. Proceedings*, pages 398–401, Wien, April 2001. Springer.

[36] Z. Vostrý. A numerical method of matrix spectral factorization. *Kybernetika*, 8(5):448–470, 1972.

[37] H. B. Aasnaes and T. Kailath. An innovations approach to least-squares estimation–part vii: Some applications of vector autoregressive-moving average models. *IEEE Trans. Autom. Control AC*, 13(6):646–655, 1973.

[38] The Math works. *Matlab System Identification Toolbox*. Natick, Massachusetts, USA, 2001.

[39] P. Nedoma, M. Kárný, I. Nagy, and M. Valečková. Mixtools. MATLAB Toolbox for Mixtures. Technical Report 1995, ÚTIA AV ČR, Praha, 2000.

[40] J. E. Dennis. Jr and V. Torczon. Direct search methods on parallel machines. *SIAM journal on optimization*, 1(4):448–474, 1991.

[41] V. Šmídl, A. Quinn, M. Kárný, and T. V. Guy. Robust estimation of autoregressive processes using a mixture based filter bank. *System & Control Letters*, 2003. Submitted.

Index

$L'DL$ decomposition, 35, 50, 60
LDL' decomposition, 52, 53, 59, 63
a posteriori pdf, 10
a priori pdf, 10
regular simplex, 15
right-angled simplex, 15

algebraic recursive updating, 33
Algorithm dydr, 32
algorithm of Dyadic reduction, 32, 35
Alternative Description of Mixtures, 39
AR(X) part, 27
ARMAX Models, 26
ARMAX models in regression form, 26
ARMMAX Models, 29
ARMMAX-QB algorithm, 64
ARX Mixtures, 28
ARX Models, 25

batch QB estimation, 42, 44
Bayes rule, 7
Bayesian estimation and prediction, 11

C-parameters, 27
C-polynomial, 27
Calculus with pdfs, 6
Chain rule, 7
colored noise, 26
common parameterized factor, 99
complete recursion, 34
Completion of squares, 36
component, 12, 22, 99
component log-likelihoods, 73
conditioning, 7
conjugate form, 33

data vector, 33
Delta model, 28
description of dynamic mixtures at factor level, 100
Description of Static Mixtures, 12
Direct Search Methods, 14
Dirichlet, 40, 41, 44, 47, 60, 63, 101
discrete random pointer, 24, 39, 59, 62
dynamic component, 22, 23
Dynamic Mixture Modelling, 22

Estimation and prediction in exponential family, 33
Estimation with Forgetting, 30
Expectation-Maximization (EM), 2
exponential family, 33, 34, 40, 42
exponential forgetting, 31
extended information matrix, 35, 64
Extended LD Filter, 52

factor, 99
finite mixture density function, 12
finite mixture distribution, 12
finite parametric mixture, 12, 99
flattening, 42, 77
flattening mapping, 43
flattening rate, 43
forgetting factor, 31

Gauss-inverse-Wishart (GiW) distribution, 35
general description of dynamic mixtures, 23
GiW, 36, 44, 47, 54, 60, 63

Independence, 7
Iterative Construction of Prior, 42
Iterative QB estimation, 44
iterative QB estimation, 42

KL distance, 30, 43, 67, 68
Kullback-Leibler (KL) distance, 7

LD Filter, 50
LD type filtering, 2
least-squares (LS) estimates, 38
likelihood function, 11
linear regression, 25

MA, 26
Marginalization, 7
Markov chain, 40
Markov Chain Monte Carlo (MCMC), 2
MARMAX Models, 28
MARMMAX-QB algorithm, 61
MDS, 18, 71
mixture log-likelihood, 73
MMQ/MAQ Algorithm, 74
MMQ/MAQ Estimation, 72
moving-average, 26
multidirectional search, 18, 71

natural conditions of control, 11
Nelder-Mead (NM) simplex method, 15
NM algorithm, 16, 71
non-degenerate simplex, 14

parallelism, 18, 66, 68, 72
parametric system model, 9, 22
pdf, 6
pf, 6
point estimate, 10
prediction error, 38
predictive pdf, 10
prior, 10, 33, 35, 42, 76
prior statistics, 33
probability density function, 6
probability function, 6

QB estimation, 40, 59, 62
QB estimation at factor level, 101
Quasi-Bayes (QB) estimation, 3

random, 6
regression, 26
regression coefficients, 26
restricted description of dynamic mixtures, 23

simplex, 14
simplex-based direct search, 14
stabilized forgetting, 30, 43
Structure of a parameterized mixture, 25
structures of parameterized components, 25
Student distribution, 38
Student form, 60, 63
sufficient statistics, 33
system model, 9

uncertainty, 6
universal approximation property of mixtures, 13, 22
universal approximation theorem of neural networks, 13
Updating of $L'DL = \lambda L'DL + \beta\Psi\Psi'$, 35

whiteness, 9