INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

**COMPARATIVE STUDY
OF TWO METHODS FOR RECURSIVE
PARAMETERS IDENTIFICATION:
KALMAN FILTER WITH HYPOTHESIS TESTING
AND
DIRECTIONAL FORGETTING**

by

Mariusz Janusz

A thesis submitted to

The Faculty of Graduate Studies and Research

In partial fulfillment of the requirements

for the degree of Masters of Applied Science

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

2001, Mariusz Janusz ©

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-66892-4

Canada

The undersigned hereby recommend to
The Faculty of Graduate Studies and Research
Acceptance of the thesis,

COMPARATIVE STUDY OF TWO METHODS FOR RECURSIVE
PARAMETERS IDENTIFICATION:
KALMAN FILTER HYPOTHESIS TESTING
AND
DIRECTIONAL FORGETTING ALGORITHM

Submitted by:
Mariusz Janusz, BEng.
In partial fulfillment of the requirements for
The degree of Master of Applied Science

Thesis Supervisor

Chair, Department of Systems and Computer Engineering

Carleton University
September 19, 2001

Abstract

The objective of this thesis is to compare the performance of the Kalman filtering algorithm with hypothesis testing to the directional forgetting algorithm based on a decomposition of the information matrix. The former was introduced by Chowdhury [2] and latter by Cao and Schwartz [1]. Residual analysis is used as a comparison measure. All assumptions, results, and conclusions are presented in the main sections of this thesis. Some recommendations and general conclusions are listed in the final discussion.

Acknowledgement

The author thanks Professor Howard Schwartz for his help and encouragement during the research and preparation of the theses. I valued the honest feedback and many constructive suggestions that proved to be essential in completing my work.

Table of Contents

Abstract	iii
Acknowledgement	iv
List of Figures	vii
List of Abbreviations and Acronyms	ix
1. Introduction	1
2. Fundamental issues in Parameters Estimation	3
2.1 The Least Square Algorithm	3
2.2 The Recursive Least Square Algorithm	5
2.3 Parameters bursting phenomena	7
3. Kalman Filter with hypothesis testing	10
3.1 Introduction to Kalman Filter	10
3.2 Kalman Filter with Hypothesis Testing	14
3.3 Example of implementing Kalman Filter with Hypothesis Testing	19
4. Directional forgetting	27
5. Comparison of two methods	31
5.1 Experiment 1	33
5.2 Experiment 2	43
5.3 Experiment 3	50
5.4 Experiment 4	58
5.5 Experiment 5	66

5.6 Experiment 6	73
5.7 Experiment 7	81
5.8 Experiment 8	88
5.9 Experiment 9	93
6. Conclusions	99
Appendix A	103
References	117

List of Figures

Figure 2.3.1 A) Input Function	B) Parameters burst after step 2000	9
Figure 3.3.1 Parameter estimation for $Tr=0.1$ and $Ta=0.05$		22
Figure 3.3.2 Residuals for $Tr=0.1$ and $Ta=0.05$		22
Figure 3.3.3 Parameter estimation for $Tr=0.1$ and $Ta=0.05$		24
Figure 3.3.4 Residuals for $Tr=0.1$ and $Ta=0.05$		24
Figure 3.3.5 Parameter estimation for $Tr=0.1$ and $Ta=0.05$ after noise conditions Changed		26
Figure 5.0.1 Typical plot of MSE as a function of forgetting factor		32
Figure 5.1.1 Actual input of the simulated system		35
Figure 5.1.2 Actual output of the simulated system		35
Figure 5.1.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA		36
Figure 5.1.4 Actual (dashed line) and estimated (dotted line) system outputs for DF		36
Figure 5.1.5 Actual (dashed line) and simulated (dotted line) system output at initial steps for CFA method		37
Figure 5.1.6 Actual (dashed line) and simulated (dotted line) system output at initial steps for DF method		37
Figure 5.1.7 Actual (dashed line) and estimated (dotted line) output around 600 step for CFA method		38
Figure 5.1.8 Actual (dashed line) and estimated (dotted line) output around 600 step for DF method		38
Figure 5.1.9 Residuals for simulated system using CFA method		39
Figure 5.1.10 Residual for simulated system using DF method		39
Figure 5.1.11 Estimates of parameters for simulated system using CFA method		40
Figure 5.1.12 Estimates of parameters for simulated system using DF method		40
Figure 5.1.13 Estimate of parameters around step 600 for CFA method		41
Figure 5.1.14 Estimate of parameters around step 600 for DF method		41
Figure 5.1.15 Estimate of parameters at the initial steps for CFA method		42
Figure 5.1.16 Estimate of parameters at the initialization stage for DF method		42
Figure 5.2.1 Actual input of the simulated system		45

Figure 5.2.2 Actual output of the simulated system	45
Figure 5.2.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	46
Figure 5.2.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	46
Figure 5.2.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method	47
Figure 5.2.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method	47
Figure 5.2.7 Residual for simulated system using CFA method	48
Figure 5.2.8 Residual for simulated system using DF method	48
Figure 5.2.9 Estimates of parameters for simulated system using CFA method	49
Figure 5.2.10 Estimates of parameters for simulated system using DF method	49
Figure 5.3.1 Actual input of the simulated system	52
Figure 5.3.2 Actual output of the simulated system	52
Figure 5.3.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	53
Figure 5.3.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	53
Figure 5.3.5 Actual (dashed line) and estimated (dotted line) output around 600 step for CFA method	54
Figure 5.3.6 Actual (dashed line) and estimated (dotted line) output around 600 step for DF method	54
Figure 5.3.7 Residuals for simulated system using CFA method	55
Figure 5.3.8 Residual for simulated system using DF method	55
Figure 5.3.9 Estimates of parameters for simulated system using CFA method	56
Figure 5.3.10 Estimates of parameters for simulated system using DF method	56
Figure 5.3.11 Estimates of parameters at initial stage of simulation for CFA method	57
Figure 5.3.12 Estimates of parameters at initial stage of simulation for DF method	57
Figure 5.4.1 Actual input of the simulated system	60
Figure 5.4.2 Actual output of the simulated system	60
Figure 5.4.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	61
Figure 5.4.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	61

Figure 5.4.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method	62
Figure 5.4.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method	62
Figure 5.4.7 Actual (dashed line) and estimated (dotted line) output around step 200 for CFA method	63
Figure 5.4.8 Actual (dashed line) and estimated (dotted line) output around step 200 for DF method	63
Figure 5.4.9 Residuals for simulated system using CFA method	64
Figure 5.4.10 Residual for simulated system using DF method	64
Figure 5.4.11 Estimates of parameters for simulated system using CFA method	65
Figure 5.4.12 Estimates of parameters for simulated system using DF method	65
Figure 5.5.1 Actual input of the simulated system	68
Figure 5.5.2 Actual output of the simulated system	68
Figure 5.5.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	69
Figure 5.5.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	69
Figure 5.5.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method	70
Figure 5.5.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method	70
Figure 5.5.7 Residuals for simulated system using CFA method	71
Figure 5.5.8 Residual for simulated system using DF method	71
Figure 5.5.9 Estimates of parameters for simulated system using CFA method	72
Figure 5.5.10 Estimates of parameters for simulated system using DF method	72
Figure 5.6.1 Actual input of the simulated system	75
Figure 5.6.2 Actual output of the simulated system	75
Figure 5.6.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	76
Figure 5.6.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	76
Figure 5.6.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method	77

Figure 5.6.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method	77
Figure 5.6.7 Residuals for simulated system using CFA method	78
Figure 5.6.8 Residual for simulated system using DF method	78
Figure 5.6.9 Estimates of parameters for simulated system using CFA method	79
Figure 5.6.10 Estimates of parameters for simulated system using DF method	79
Figure 5.6.11 Estimated parameters at initial steps for CFA method	80
Figure 5.6.12 Estimated parameters at initial steps for DF method	80
Figure 5.7.1 Actual input of the simulated system	83
Figure 5.7.2 Actual output of the simulated system	83
Figure 5.7.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	84
Figure 5.7.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	84
Figure 5.7.5 Actual (dashed line) and estimated (dotted line) output around 600 step for CFA method	85
Figure 5.7.6 Actual (dashed line) and estimated (dotted line) output around 600 step for DF method	85
Figure 5.7.7 Residuals for simulated system using CFA method	86
Figure 5.7.8 Residual for simulated system using DF method	86
Figure 5.7.9 Estimates of parameters for simulated system using CFA method	87
Figure 5.7.10 Estimates of parameters for simulated system using DF method	87
Figure 5.8.1 Actual input of the simulated system	89
Figure 5.8.2 Actual output of the simulated system	89
Figure 5.8.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	90
Figure 5.8.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	90
Figure 5.8.5 Residuals for simulated system using CFA method	91
Figure 5.8.6 Residual for simulated system using DF method	91
Figure 5.8.7 Estimates of parameters for simulated system using CFA method	92
Figure 5.8.8 Estimates of parameters for simulated system using DF method	92
Figure 5.9.1 Actual input of the simulated system	95
Figure 5.9.2 Actual output of the simulated system	95
Figure 5.9.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA	96

Figure 5.9.4 Actual (dashed line) and estimated (dotted line) system outputs for DF	96
Figure 5.9.5 Residuals for simulated system using CFA method	97
Figure 5.9.6 Residual for simulated system using DF method	97
Figure 5.9.7 Estimates of parameters for simulated system using CFA method	98
Figure 5.9.8 Estimates of parameters for simulated system using DF method	98

List of Abbreviations and Acronyms

CFA: Combined Filter Algorithm

DF: Directional Forgetting

ARMA: Auto Regressive Moving Average

FF: Forgetting Factor

MSE: Mean Square Error

LS: Least Square

LSE: Least Square Error

Chapter 1

Introduction

On-line estimation of system parameters is a well-researched and documented subject. One particular group of ideas is directed at a special class of systems: poorly excited systems that can be described as a system having input data with insignificant content of new information. The main drawback observed in those systems is the so-called estimator wind-up problem (also known as parameter bursting phenomena) corresponding to a sudden exponential growth of the covariance matrix. The origins of this phenomenon are well described in Kulhavy & Karny [3]. As very little new information is delivered to the system and the old information gets gradually forgotten the system eventually becomes under-trained and susceptible to instabilities. Any disturbance in the system (even insignificant) can result in the burst of parameter estimates. It is important to understand that this kind of poor excitation is very typical for chemical processes and as such constitutes a significant barrier for a successful implementation of adaptive controllers in industrial environments. A number of different solutions and approaches were presented over the last few years, each addressing the issue to a different extend and with different degree of success.

Methods proposed such as Selective Forgetting (SF) described by Parkum, Poulsen, Holst [8] or Directional Forgetting (DF) described in Khulavy & Karny [7], suffer from excessive computational requirements [1], which in turn limit the practical value of those algorithms. Two of the algorithms analyzed in this paper introduce a very attractive alternative from the performance and ease of implementation point of view.

A Directional Forgetting Algorithm Based on decomposition of the Information matrix introduced by Cao and Schwartz in [1] performs exponential forgetting according to

the direction of new information. In this method only one part of information matrix is allowed to forget information at each update. Even in cases of poor excitation, parameter windup does not occur because the algorithm shelters appropriate part of information matrix. This is a simple and effective method to eliminate major shortcomings of the Least Square Algorithm.

Kalman Filtering with Hypothesis Testing introduced by Chowdhury in [2] leverages the traditional discrete Kalman Filter theory and statistical single-sample hypothesis testing. The algorithm introduces a three-state decision rule (yes, no, maybe) to determine if an unexpected parameter jump occurred. This method is capable of handling all different behaviours of a given system because the algorithm switches back and forth between techniques that allow tracking slowly varying parameters (fading memory filter) and constant parameters. Also in this case, simplicity and effectiveness are main attributes of this method.

The thesis paper is divided into seven chapters. After initial introduction of the subject Chapter 2 provides general background information about parameter estimation. Some main subjects elaborated include, The Least Square Method (in both discrete and recursive versions) and parameter bursting phenomena. In Chapter 3, the details and properties of Kalman filter with the hypothesis testing are discussed. Chapter 4 contains similar analysis for Directional Forgetting by the Information Matrix Decomposition algorithm. Chapter 5 contains details and results of simulations conducted on both methods. The final conclusions and findings are reported in Chapter 6. Source-codes for all Matlab simulations used in the paper are listed in Appendix 1.

Chapter 2

Fundamental issues in Parameters Estimation

2.1 The Least Square Algorithm

If we denote the system input and output at time t as $u(t)$ and $y(t)$ respectively, and the system is linear, the relationship between input and output could be written as,

$$y(t) + a_1 \cdot y(t-1) + \dots + a_n \cdot y(t-n) = b_1 \cdot u(t-1) + \dots + b_m \cdot u(t-m) + \mu(t), \quad (2.1)$$

where $\mu(t)$ is a white gaussian noise.

With this notation $y(t)$ can be derived as,

$$\hat{y}(t) = -a_1 \cdot y(t-1) - \dots - a_n \cdot y(t-n) + b_1 \cdot u(t-1) + \dots + b_m \cdot u(t-m) + \mu(t), \quad (2.2)$$

where, $\hat{y}(t)$ is an estimate of $y(t)$ based on selected parameters and given data set. For a more compact notation we can use the following representation,

$$\Theta = [a_1, a_2, a_3, \dots, a_n, b_1, b_2, \dots, b_m]^T, \quad (2.3)$$

$$\varphi(t) = [-y(t-1), -y(t-2), \dots, -y(t-n), u(t-1), u(t-2), \dots, u(t-m)]^T. \quad (2.4)$$

Now $\hat{y}(t)$ can be represented as,

$$\hat{y}(t) = \varphi^T(t) \cdot \Theta + \mu(t). \quad (2.5)$$

Such a model is very often referred to as a linear regression. It is very powerful because having $y(t)$ and $\varphi(t)$ one can estimate the parameter vector Θ and use it as a prediction model. In order to further describe a prediction of $y(t)$ we can adopt the following notation, as per [4] p.203,

$$\hat{y}(t / \Theta) = \varphi^T(t) \cdot \Theta + \mu(t), \quad (2.6)$$

which means that the prediction of y is based on selection of the parameter vector, Θ .

The prediction error can be calculated using the following formula, as per [4] p.198

$$\varepsilon(t, \Theta) = y(t) - \hat{y}(t / \Theta). \quad (2.7)$$

This equation can be further transformed using 2.6, as per [4] p.204,

$$\varepsilon(t, \Theta) = y(t) - \varphi^T(t) \cdot \Theta \quad (2.7A)$$

One method of optimizing the solution for parameters vector Θ is to minimize the value of $\varepsilon(t, \Theta)$. The standard choice is to use a quadratic norm, as per [4] p.201

$$V_N(\Theta, Z^N) = \frac{1}{N} \sum_{t=1}^N \frac{1}{2} \cdot \varepsilon(t, \Theta)^2. \quad (2.8)$$

Substituting for $\varepsilon(t, \Theta)$ from equation 2.7 we get, as per [4] p.204

$$V_N(\Theta, Z^N) = \frac{1}{N} \sum_{t=1}^N \frac{1}{2} \cdot [y(t) - \varphi^T(t) \cdot \Theta]^2. \quad (2.9)$$

This is a relatively simple function to minimize using standard analytical methods, as per [4] p. 204,

$$\hat{\Theta}_N^{LS} = \arg \min V_N(\Theta, N), \quad (2.10)$$

$$\hat{\Theta}_N^{LS} = \left[\frac{1}{N} \sum_{t=1}^N \varphi(t) \cdot \varphi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^N \varphi(t) \cdot y(t). \quad (2.11)$$

Introducing new notation, as per [4] p.204,

$$R(N) = \frac{1}{N} \sum_{t=1}^N \varphi(t) \cdot \varphi^T(t), \quad (2.12)$$

$$f(N) = \frac{1}{N} \sum_{t=1}^N \varphi(t) \cdot y(t), \quad (2.13)$$

we can represent parameters vector as,

$$\hat{\Theta}_N^{LS} = R^{-1}(t) \cdot f(N), \quad (2.14)$$

where the necessary assumption is that $R(t)$ is non-singular.

2.2 The Recursive Least Square Algorithm

The Least Square method can be further generalized by allowing different weights to different measurements. This results in so called Weighted Least Square method and can be represented by the following expression,

$$\hat{\Theta}_t = \left[\sum_{k=1}^t \beta(t, k) \cdot \varphi(t) \cdot \varphi^T(t) \right]^{-1} \sum_{k=1}^t \beta(t, k) \cdot \varphi(t) \cdot y(t), \quad (2.15)$$

where $\beta(t, k)$ is the weight of k -th measurement.

Again, this formula can be expressed in a more compact form by substitutions, as per [4] p.363,

$$\bar{R}(t) = \sum_{k=1}^t \beta(t, k) \cdot \varphi(t) \cdot \varphi^T(t), \quad (2.16)$$

$$f(t) = \sum_{k=1}^t \beta(t, k) \cdot \varphi(t) \cdot y(t). \quad (2.17)$$

The final representation of the Weighted Least Square method is,

$$\hat{\Theta}_t = \bar{R}^{-1}(t) \cdot f(t). \quad (2.18)$$

Assuming that the weighting formula satisfies the following equation, as per [4] p.363,

$$\beta(t, k) = \lambda(t) \cdot \beta(t-1, k), \quad (2.19)$$

where, $0 \leq k \leq t-1$ and

$$\beta(t, t) = 1, \quad (2.20)$$

we may write,

$$\beta(t, k) = \prod_{j=k+1}^t \lambda(j). \quad (2.21)$$

Using the following results, as per [4] p.364,

$$\bar{R}(t) = \lambda(t) \cdot \bar{R}(t-1) + \varphi(t) \cdot \varphi^T(t), \quad (2.22)$$

$$f(t) = \lambda(t) \cdot f(t-1) + \varphi(t) \cdot y(t), \quad (2.23)$$

one can write,

$$\hat{\Theta}_t = \bar{R}^{-1}(t) \cdot f(t), \quad (2.24)$$

$$\hat{\Theta}_t = \bar{R}^{-1}(t) \cdot [\lambda(t) \cdot f(t-1) + \varphi(t) \cdot y(t)], \quad (2.25)$$

$$\hat{\Theta}_t = \bar{R}^{-1}(t) \cdot [\lambda(t) \cdot \bar{R}(t-1) \cdot \hat{\Theta}_{t-1} + \varphi(t) \cdot y(t)], \quad (2.26)$$

$$\hat{\Theta}_t = \bar{R}^{-1}(t) \cdot \{[\bar{R}(t) - \varphi(t) \cdot \varphi^T(t)] \cdot \hat{\Theta}_{t-1} + \varphi(t) \cdot y(t)\}. \quad (2.27)$$

In conclusion, as per [4] p.364,

$$\begin{cases} \hat{\Theta}_t = \hat{\Theta}_{t-1} + \bar{R}^{-1}(t) \cdot \varphi(t) \cdot [y(t) - \varphi^T(t) \cdot \hat{\Theta}_{t-1}], \\ \bar{R}(t) = \lambda(t) \cdot \bar{R}(t-1) + \varphi(t) \cdot \varphi^T(t). \end{cases} \quad (2.28)$$

$$(2.29)$$

Equations 2.28 and 2.29 constitute a recursive algorithm of the Least Square Method.

In order to avoid inverting the R matrix at each step of the computation, we can define, as per [4] p.364,

$$P(t) = \bar{R}^{-1}(t). \quad (2.30)$$

Using the matrix inversion lemma,

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1}, \quad (2.31)$$

and making the following substitutions,

$$A = \lambda(t)\bar{R}(t-1), \quad (2.32)$$

$$B = D^T = \varphi(t), \quad (2.33)$$

$$C = 1, \quad (2.34)$$

one gets,

$$P(t) = \frac{1}{\lambda(t)} \left[P(t-1) - \frac{P(t-1) \cdot \varphi(t) \cdot \varphi^T(t) \cdot P(t-1)}{\lambda(t) + \varphi^T(t) \cdot P(t-1) \cdot \varphi(t)} \right]. \quad (2.35)$$

Finally we can summarize the recursive Least Squares Method as,

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + L(t) \cdot [y(t) - \varphi^T(t) \cdot \hat{\Theta}(t-1)], \quad (2.36)$$

$$L(t) = \frac{P(t-1) \cdot \varphi(t)}{\lambda(t) + \varphi^T(t) \cdot P(t-1) \cdot \varphi(t)}, \quad (2.37)$$

$$P(t) = \frac{1}{\lambda(t)} \left[P(t-1) - \frac{P(t-1) \cdot \varphi(t) \cdot \varphi^T(t) \cdot P(t-1)}{\lambda(t) + \varphi^T(t) \cdot P(t-1) \cdot \varphi(t)} \right]. \quad (2.38)$$

$\bar{R}(t)$ is referred to as the information matrix and plays a very important role in determining knowledge about the system. In the equation,

$$\theta(t) = \lambda(t) \cdot \theta(t-1) + \varphi(t) \cdot \varphi^T(t), \quad (2.39)$$

one can see that the forgetting factor (typically 0.95 to 0.99 as per Andersson [6]) causes the system data to be forgotten after each time step.

2.3 Parameters bursting phenomena

Tracking slowly varying parameters poses some additional difficulties attributed to the phenomena of covariance matrix windup. This subject has been described in a number of papers [Cao & Schwartz [1], Fortescue & Kershenbaun & Ydstie [3], Kulhuavy & Karny [7]]. When a system is poorly excited and no new information is introduced to parameter estimation, a significant amount of system knowledge is lost due to forgetting mechanisms

(dependent on forgetting factor). If no advanced algorithms are used to prevent it, parameters will eventually burst causing a regulator to become very sensitive to disturbances and may even result in loss of stability. It proved to be a significant factor limiting the use of adaptive controllers in industrial applications. As already indicated, the covariance matrix is very useful in analyzing this phenomenon. In the Least Square Method, the covariance matrix is defined as,

$$\text{cov } \theta = P(t) |_{\lambda(t)=1} . \quad (2.40)$$

Using the definition of $P(t)$ represented by equation 2.38 and equations 2.29 and 2.30, we get,

$$\text{cov } \theta = \bar{R}^{-1}(t) |_{\lambda=1} = [\bar{R}(t-1) + \varphi(t) \cdot \varphi^T(t)]^{-1} . \quad (2.41)$$

Initially, after system identification process is started, the discrepancy between estimated and real parameters is large and therefore the information matrix is small. As more information is collected and parameter estimation becomes more accurate, the covariance matrix $P(t)$ becomes small. However, as control becomes better and very little or no new information about the system's dynamics is introduced, the information matrix $R(t)$ can become very small, especially for near deterministic systems, (Fortescue, Kershenbaun, and Ydstie [3]). At this point the system identification algorithm becomes very sensitive to disturbances or numerical errors and the parameter bursting phenomena is likely to occur.

In order to demonstrate how parameters blow-up a simulation was conducted for LSA identification of the following ARMA system,

$$y(t) + 2 \cdot y(t-1) + 2 \cdot y(t-2) = 2 \cdot u(t-1) + u(t-2) + e(t) , \text{ where } e(t) \text{ is white noise.}$$

The input at the initial stage is a sinusoidal function but after step 1000 it becomes just a small white noise signal, see figure 2.3.1A. As one can see in the figure 2.3.1B, after step

1000 the system does not get any new information and around step 2000 parameters burst causing instability.

Figure 2.3.1 **A) System Input Function**

B) System parameters. The parameter burst phenomena occurs after after step 2000

Chapter 3

Kalman Filter with hypothesis testing

3.1 Introduction to Kalman Filter

Details regarding the Kalman Filter were published in 1960 [9] and since that time it has been a subject for extensive research and applications. The Kalman filter provides a very efficient solution to the Least Square Method and is very powerful due to its computational simplicity.

The main objective of Kalman filter is to estimate the state $x \in R^n$ governed by the following equation [9],

$$x_{k+1} = A_k \cdot x_k + B_k \cdot u_k + w_k. \quad (3.1)$$

Using a measurement $z \in R^m$

$$z_k = H_k \cdot x_k + v_k, \quad (3.2)$$

where, w_k -Process white noise, and $p(w)$ -N(0,Q)

v_k -Measurement white noise, and $p(v)$ -N(0,R).

If we define [9],

$\bar{x}_k \in R^n$ as 'a priori' state estimate at step k given knowledge of the process prior to step k,

$\hat{x}_k \in R^n$ as 'a posteriori' state estimate given all measurements up to z_k .

We can define errors [9],

$$\bar{e}_k = x_k - \bar{x}_k \quad \text{'a priori' estimate error,} \quad (3.3)$$

$$e_k = x_k - \hat{x}_k \quad \text{'a posteriori' estimate error.} \quad (3.4)$$

Covariance of the two estimates can be expressed as [9],

$$\bar{P}_k = E[\bar{e}_k \bar{e}_k^T] \quad \text{'a priori',} \quad (3.5)$$

$$P_k = E[e_k e_k^T] \text{ 'a posteriori',} \quad (3.6)$$

where E symbolizes the expected value.

The following equation (3.7) is true [7] assuming that 'a posteriori' estimation is a linear combination of 'a priori' estimate and weighted difference between measurement (z_k) and measurement prediction ($H_k \cdot \bar{x}_k$),

$$\hat{x}_k = \bar{x}_k + K \cdot (z_k - H_k \cdot \bar{x}_k), \quad (3.7)$$

where, $z_k - H_k \cdot \bar{x}_k$ is called a measurement innovation or residual and K is the gain factor minimizing 'a posteriori' error covariance. After the minimization process (done similarly to Least Square Method) we get [9],

$$K_k = \bar{P}_k \cdot H_k^T \cdot (H_k \cdot \bar{P}_k \cdot H_k^T + R_k)^{-1} = \frac{\bar{P}_k \cdot H_k^T}{H_k \cdot \bar{P}_k \cdot H_k^T + R_k}. \quad (3.8)$$

Equation 3.8 is correct only if the denominator is a scalar.

The Kalman filter estimates the process by using a dual step update: time update, and measurement update. The time update provides the 'a priori' estimate and in this sense the filter makes a projection forward of the current state. The measurement update ('a posteriori') provides feedback measurement to verify 'a priori' state estimate. This two-step cycle can be expressed by the following set of equation as presented by Welch and Bishop in [9],

$$\text{time step:} \quad \begin{cases} \bar{x}_{k+1} = A_k \cdot \hat{x}_k + B_k \cdot u_k, \end{cases} \quad (3.9)$$

$$\begin{cases} \bar{P}_{k+1} = A_k \cdot P_k \cdot A_k^T + Q_k, \end{cases} \quad (3.10)$$

$$\text{measurement update: } \begin{cases} K_k = \bar{P}_k \cdot H_k^T \cdot (H_k \cdot \bar{P}_k \cdot H_k^T + R_k)^{-1}, & (3.11) \\ \hat{x}_k = \bar{x}_k + K_k (z_k - H_k \cdot \bar{x}_k), & (3.12) \\ P_k = (I - K_k \cdot H_k) \cdot \bar{P}_k. & (3.13) \end{cases}$$

This is also the most widely used form of Kalman filter equations.

The principle of Kalman filtering can be also applied to recursive parameter identification. Assuming that $A_k = I$ and $B_k = 0$ in equations 3.9 and 3.10 we get,

$$\bar{x}_{k+1} = \hat{x}_k, \quad (3.14)$$

$$\bar{P}_{k+1} = P_k + Q_k. \quad (3.15)$$

Substituting equation 3.13 into 3.15 we get [9],

$$\bar{P}_{k+1} = \bar{P}_k - K_k \cdot H_k \cdot \bar{P}_k + Q_k. \quad (3.16)$$

Substituting the expression for K_k (eg. 3.11) in equation 3.16 we get [9],

$$\bar{P}_{k+1} = \bar{P}_k - \frac{\bar{P}_k \cdot H_k^T \cdot H_k \cdot \bar{P}_k}{R_k + H_k \cdot \bar{P}_k \cdot H_k^T} + Q_k. \quad (3.17)$$

By rearranging the equations order and adopting notation ($\rightarrow$) from Ljung [4], page 367,

$$\bar{x}_k \rightarrow \hat{\theta}(t-1),$$

$$H_k \rightarrow \varphi^T(t),$$

$$Q_k \rightarrow R_1(t),$$

$$R_k \rightarrow R,$$

we get,

$$\left\{ \begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + K(t) \cdot [y(t) - \varphi^T(t) \cdot \hat{\theta}(t-1)], & (3.18) \\ K(t) &= \frac{P(t-1) \cdot \varphi(t)}{R(t) + \varphi^T(t) \cdot P(t-1) \cdot \varphi(t)}, & (3.19) \\ P(t) &= P(t-1) - \frac{P(t-1) \cdot \varphi(t) \cdot \varphi^T(t) \cdot P(t-1)}{R(t) + \varphi^T(t) \cdot P(t-1) \cdot \varphi(t)} + R_1(t). & (3.20) \end{aligned} \right.$$

As stated by Andersson in [6], the common problem with this method is that in order to make the algorithm reasonably responsive to changes, the value of $R_1(t)$ needs to be large and this will have a negative effect on performance when no changes occur. This situation forces a designer to compromise between responsiveness and stability for poor excitation. Several suggestions have been made to solve this problem, one of them, the Kalman Filter with hypothesis testing, is introduced in the next section.

3.2 Kalman Filter with Hypothesis Testing

Identification of linear systems is a well-explored area with a variety of tools available for different circumstances. It remains difficult however, to develop a single technique that could address all possible behaviours of a given system. Kalman Filter with Hypothesis Testing or CFA (Combined Filter Algorithm) as Chowdhury [2] describes it, is an attempt to create such a universal method.

For a given time-variant system, parameter tracking can be divided into three different cases:

- 1) sudden and significant change in parameters (jump); the jump could be a part of system dynamics or some abnormal situation like system fault or failure,
- 2) very little fluctuation of parameters that can be considered negligible; this state is most likely to exist when we deal with near-deterministic systems that are poorly excited,
- 3) continuous and reasonably bounded change in parameters that can be tracked; this state corresponds to a typical system that demonstrates small parameters variations and reasonable external noise.

Case #3 can be considered as typical and most desirable behaviour of a system while cases #1 and #2 are exceptional circumstances that require special measures.

All three cases described above can be formalized in the test of hypothesis, where two hypotheses are defined:

- 1) H_0 - null hypothesis; no jump,
- 2) H_1 - alternate hypothesis; jump.

The important issue becomes how to define and measure the rate of parameter change. The author of [2] defines two thresholds called the rejection and the acceptance thresholds. The null hypothesis is false if the parameter change is greater than the rejection threshold and is true if the parameter change is below acceptance threshold.

T_r - Rejection threshold; the threshold of parameter change above which hypothesis H_0 is rejected and a parameter jump has occurred.

T_a - Acceptance threshold; the threshold of parameter change below which hypothesis H_0 is accepted.

T_r And T_a can be defined based on type I and II errors respectively, widely used in the statistical literature for hypothesis tests.

The probability of a bad decision (false alarm) is dependent on selection of the rejection threshold,

$$\Pr(H1_is_accepted / H0_is_true) = 1 - \int_{-T_r}^{T_r} f(x)dx, \quad (3.21)$$

where $f(x)$ is the probability density function of the normal distribution.

The acceptable level of false alarms therefore determines the value of rejection threshold.

One needs to be careful however, because too aggressive an approach in minimizing error I without consideration to the noise level can cause some significant problems for this algorithm.

Similarly, the maximum probability of a missed jump can be used to determine the value of the acceptance threshold,

$$\Pr((H0_is_accepted / H1_is_true) = \int_{-T_a}^{T_a} f(x)dx. \quad (3.22)$$

Also in this case caution with selection of acceptable type error II is advised and has to be considered in conjunction with noise levels.

If we assume that $\tilde{Z}_{nk}$ is the normalized residual [2] than there are three possible outcomes of hypothesis testing (equivalent to three different cases described earlier),

- 1) Reject H_o if $|\tilde{Z}_{nk}| > T_r$ Jump
- 2) Accept H_o if $|\tilde{Z}_{nk}| < T_a$ No jump
- 3) No decision if $T_a \leq |\tilde{Z}_{nk}| \leq T_r$.

Each outcome of hypothesis testing has a distinct technique to deal with parameters tracking:

- 1) If a significant change in parameter value occurs and the old accumulated knowledge is no longer valid, the filter has to be reinitialized with a new initial guess. A more sophisticated technique can also be applied where the magnitude of the jump is assessed and initial values can be set based on this information. After the initial guess the filter will eventually converge to the new parameters. This is how the CFA deals with a jump.
- 2) No change in residuals is detected so it should be concluded that the parameters remain steady and no additional action is necessary. CFA does nothing.
- 3) This event is called ‘no decision’ and corresponds to a situation where parameters change very slowly but significantly enough so that changes need to be captured. In this case a different model is applied, called the Random Walk Kalman Filter and is represented by the following mathematical model,

$$X_{k+1} = F_k \cdot X_k + B_k \cdot U_k + W_k, \quad (3.23)$$

where, X_k is the state vector at step k , F_k is the state transition matrix, B_k is the input transition matrix, and W_k is the random walk element (or process noise).

This filter can effectively track small changes in parameters as long as residuals fall between the two thresholds defined above. The decision making process is based on a normalized residual denoted in [2] as Z_{nk} and derived in the following analysis,

$$\tilde{Z}_k = Z_k - \hat{Z}_k = Z_k - H_k \cdot \hat{X}_{k/k-1}, \quad (3.24)$$

where, $\hat{X}_{k/k-1}$, is the estimate of X at time k using measurements up to time $k-1$ and,

$$\hat{X}_{k/k} = X_{k/k-1} + K_k \cdot \tilde{Z}_k, \text{ is the Kalman estimate.} \quad (3.25)$$

$$Q_{-k} = E[(Z - H \cdot \hat{X}_{k/k-1}) \cdot (Z - H \cdot \hat{X}_{k/k-1})^T], \quad (3.26)$$

where, Q_{-k} , is the covariance matrix of residuals, H is the observation matrix, and $\hat{X}$ is the estimate of state X at time k using the measurement history up to time $k-1$ [2].

Measurement Z is given by [2],

$$Z = H \cdot X + V, \quad (3.27)$$

where, V is uncorrelated zero-mean Gaussian observation noise and X the actual state.

Substituting 3.25 into 3.24 we get,

$$Q_{-k} = E[(H \cdot X_k + V - H \cdot \hat{X}_{k/k-1})(H \cdot X_k + V - H \cdot \hat{X}_{k/k-1})^T]. \quad (3.28)$$

Predicted estimation error e^- can be expressed as [2],

$$e^- = X_k - \hat{X}_{k/k-1}. \quad (3.29)$$

Grouping similar terms in 3.28 we get [2],

$$Q_{zk} = E[(H \cdot (X_k - \hat{X}_{k/k-1}) + V)(H \cdot (X_k - \hat{X}_{k/k-1}) + V)^T]. \quad (3.30)$$

Substituting 3.29 into 3.30 we get [2],

$$Q_{zk} = E[(He^- + V) \cdot (He^- + V)^T], \quad (3.31)$$

$$Q_{zk} = H \cdot P^- \cdot H^T + R, \quad (3.32)$$

$$P^- = E(e^- \cdot e^{-T}), \quad (3.33)$$

where, P^- is the covariance matrix of the prediction errors and R is the covariance matrix of the observation noise.

Kalman gain K_k as [2],

$$K_k = P_{k/k-1} \cdot H_k^T \cdot [H_k \cdot P_{k/k-1} \cdot H_k^T + R_k]^{-1}, \quad (3.34)$$

The normalized residual is given as [2],

$$\tilde{Z}_{nk} = \tilde{Z}_k / \sqrt{Q_{zk}}, \quad (3.35)$$

where, $\tilde{Z}_{nk}$ is the normalized residual. This technique is also sometimes referred to as fading memory model.

Chowdhury [2] presented several examples of simulations and applications of this algorithm to different dynamic systems. All demonstrate that this is an effective and a fast method for the detection of parameter jumps in a variety of situations. If parameters remain

unchanged and stable for an extended period of time, no covariance matrix blow-up phenomena is observed. Another significant feature of Kalman Filter with Hypothesis Testing method is computational simplicity.

3.3 Example of implementing Kalman Filter with Hypothesis Testing

In order to illustrate the procedure for practical implementation of the Kalman Filter with Hypothesis Testing algorithm, the following system has been simulated,

$$y(t) + a \cdot y(t-1) + 0.8 \cdot y(t-2) = u(t-1) + 0.7 \cdot u(t-2) + e(t),$$

where,

$$a = \begin{cases} 0.1 & \text{for } t \leq 400, \\ 0.2 & \text{for } 400 < t \leq 700, \\ 0.3 & \text{for } 700 < t \leq 1000, \\ 0.4 & \text{for } 1000 < t \leq 1300, \\ 0.5 & \text{for } 1300 < t \leq 1600, \\ 0.6 & \text{for } 1600 < t \leq 2000. \end{cases}$$

The input function is graphically presented in Figure 5.8.1 and represents a white noise waveform with mean of $\bar{x} = 0$ and a variance of $\sigma = 0.1$.

The most challenging task in practical implementation of this algorithm is to determine proper values for rejection and acceptance thresholds represented by Tr and Ta respectively. Rejection threshold Tr represents such a value of the normalized residual $\tilde{Z}_n$ that any value higher than that threshold is considered to be a parameter jump. Acceptance threshold Ta represents in turn such a value of the normalized residual $\tilde{Z}_n$ that any value below it should

be considered as ‘no jump’ and consequently no action or parameter adjustment is necessary. It is obvious that the selection of both thresholds is strongly dependent on simulated system characteristic and noise levels.

The first guess for values of Tr and Ta depends on system noise, input noise levels and is mostly based on experience. Based on my practice, the general rule is to start from $Tr=0.1$, $Ta=0.05$ when noise levels are relatively low and $Tr=0.5$, $Ta=0.1$ when noise levels are high. In order to identify the number of occurrences for each range of interest, $err2 < Ta$, $Ta < err2 < Tr$, and $err2 > Tr$, (where $err2$ is the normalized residual and was defined in equation 3.35) three counters had been added to Kalman.m Matlab function (listed in Appendix A) using the following logic,

```
if |err2| > Tr than increment counter1
if |err2| ≤ Tr and |err2| ≥ Ta than increment counter2
if |err2| < Ta than increment counter3,
```

where, counter1 is the number of occurrences when the normalized residual $\tilde{Z}_n$ exceeded Tr (accept hypothesis H_o), counter2 is the number of occurrences when the normalized residual $\tilde{Z}_n$ exceeded Ta but was below Tr (reject hypothesis H_o), and counter3 is the number of occurrences when the normalized residual $\tilde{Z}_n$ was less than Ta (no decision for hypothesis H_o).

It is important to identify the number of parameter jumps for a simulated system. When running a simulation, one should look at the *counter1* and try to make sure that it matches the number of expected parameter jumps. This is the main indicator if Tr has been selected correctly as its value is directly related to the number of parameter jumps. On the top of existing (anticipated) jumps one should typically add one or two additional occurrences to

account for the initial stage of the simulation when system knowledge is poor and parameters behave erratically.

For the selection of $Tr=0.1$ and $Ta=0.05$ the following results were received,

$MSE = 0.0014$, $counter1 = 0$, $counter2 = 1$, and $counter3 = 1999$,

where, MSE is defined in equation 5.1 (page 42).

It is clear that all parameter jumps were missed because $counter1$ for $err2 > Tr$ range was equal to zero. Looking at $counter2$ and $counter3$ it is also obvious that the value of Ta was selected too large because almost all occurrences happened in the range $err2 < Ta$. In such a situation the Kalman Filter algorithm remains in the 'do nothing' decision throughout the entire simulation and consequently the system is not identified at all. The conclusion at this point should be that both Tr and Ta need to be substantially reduced in order to better reflect noise levels and characteristics of the system in question.

More important learning can be obtained from analyzing graphs. The figure below presents estimated parameters and it is obvious that jumps were poorly detected.

Figure 3.3.1 Parameter estimation for $Tr=0.1$ and $Ta=0.05$

Figure 3.3.2 Residuals for $Tr=0.1$ and $Ta=0.05$

The graph of residual leads to similar conclusions. One will easily notice that the value of residual grows this time. This indicates that the algorithm cannot keep up with the new system knowledge and the quality of identification diminishes over time.

The next step is to try $Tr=0.05$ and $Ta=0.01$. The following results were obtained, $MSE = 3.9912e-004$, $counter1 = 1$, $counter2 = 6$, and $counter3 = 1993$.

Using the identical reasoning, it is obvious that Tr and Ta need to be further reduced.

The next step is to try $Tr=0.01$ and $Ta=0.003$. The following results were obtained,

$MSE = 3.4282e-004$, $counter1 = 7$, $counter2 = 7$, and $counter3 = 1986$.

This is a much better result because one can start to distinguish individual parameter jumps (see figure 3.3.3).

Figure 3.3.3 Parameter estimation for $Tr=0.01$ and $Ta=0.003$

Figure 3.3.4 Residuals for $Tr=0.01$ and $Ta=0.003$

It is obvious that the algorithm detects major jumps and performs much better than in any of the previous attempts. Figure 3.3.4 proves that residual is consistent across the entire simulation.

Vast majority of events (1990) are still concentrated in the 'no action' range where Kalman Filter algorithm remains practically inactive. This indicates that Ta needs to be further reduced in order to include more events in both dynamic ranges and make the algorithm more active. Also, it is preferred to see around ten jumps altogether rather than seven to account for the initial stage when system knowledge is poor and parameters behave erratically.

The next step is to try $Tr=0.007$ and $Ta=0.0032$. The following results were obtained, $MSE = 3.0005e-004$, $counter1 = 8$, $counter2 = 16$, and $counter3 = 1976$.

Again, in this case better results were obtained and system identification is very satisfactory. One can perform some final tuning trying to reduce the value of MSE. The best results were obtained for $Tr=0.0068$ and $Ta= 0.0015$,

$MSE = 2.9106e-004$, $counter1 = 8$, $counter2 = 35$, and $counter3 = 1957$.

The final results are graphically represented in Chapter 5.9 where identical system was simulated.

A given selection of threshold values is only valid for a specific set of conditions. If anything changes then the same process has to be started again. For the illustration purposes the level of system noise was increased from 0.01 to 0.05, both thresholds were left unchanged. The results are presented below, $MSE = 0.0032$ (*large!*), $counter1 = 77$, $counter2 = 1198$, and $counter3 = 725$. It is obvious that the selected thresholds need to be tuned in order to get full capabilities of this method (see figure 3.2.5).

Based on the presented example one should conclude that it is rather complicated to determine the desirable values of the thresholds. This algorithm is very dependent on system characteristics and noise levels so the presented procedure needs to be followed every time a new system is analyzed. This is the major disadvantage of CFA algorithm that influences its practical value.

Figure 3.3.5 Parameter estimation $Tr=0.0068$ and $Ta=0.0015$ after noise conditions changed. Lines from the top: coefficients of $u(t-1)$, $y(t-2)$, $u(t-2)$, $y(t-1)$

Chapter 4

Directional forgetting

In order to prevent the estimator wind-up phenomena one has to identify when and where there is a lack of new information and prevent forgetting in that region of the system's knowledge. If this is neglected, the system identification engine will lose a significant portion of its knowledge base and result in unstable behaviour. Several algorithms have been introduced in the past that can alleviate this problem but typically they suffer from some limitations that force users to trade benefits. Cao and Schwartz [1] presented a method that addresses most of the shortcomings that other algorithms introduce, such as high computational demand or poor tracking capabilities. The main idea adopted in the paper is to decompose the information matrix into two components: R_2 that carries new information and should be forgotten and R_1 that does not contain new information and should be preserved. Both are dynamic and dependent on the excitation of the system.

If we assume that R is the information matrix used in the exponential forgetting of recursive least squares method then the recursive equation for $R(t)$ can be expressed as,

$$R(t) = \lambda \cdot R(t-1) + \varphi(t) \cdot \varphi(t)^T, \quad (4.1)$$

where, λ is the forgetting factor; $0 \leq \lambda \leq 1$. From the equation 4.1 it is obvious that all knowledge accumulated in the information matrix is forgotten evenly in all directions and continuously. If the input does not supply new information in a certain direction but old information is lost then the model will lack necessary knowledge about the system. This is the origin of the covariance windup and the parameter bursting problem associated with the standard recursive least squares method. In order to eliminate this problem the information

matrix should only forget information in the direction where there is a fresh supply of new information and retain information in the direction where no new information exists.

In order to achieve this objective, the information matrix $R(t-1)$ can be decomposed as follows [1],

$$R(t-1) = R_1(t-1) + R_2(t-1). \quad (4.2)$$

$R_1(t)$ has to satisfy the condition ensuring that no new information is introduced by updates of $\varphi(t)$ and $R_1(t)$ cannot be forgotten. Cao & Schwartz in [1] prove that $R_1(t-1)$ and $\varphi(t)$ are orthogonal, which implies that,

$$R_1(t-1) \cdot \varphi(t) = 0, \quad (4.3)$$

where $\varphi(t) \neq 0$. The second component, $R_2(t)$ has to satisfy condition (4.4) ensuring that new information is introduced by the update of $\varphi(t)$ and $R_2(t)$ should be forgotten [1],

$$R_2(t-1) \cdot \varphi(t) = R(t-1) \cdot \varphi(t). \quad (4.4)$$

In order to determine $R_1(t-1)$ and $R_2(t-1)$ uniquely we need to restrict ranks of both components. Cao and Schwartz in [1] proved that the rank of $R_1(t-1)$ is $n-1$ and the rank of $R_2(t-1)$ is 1 and both are nonnegative definite.

Using the two previous equations, R_2 can be expressed as [1],

$$R_2(t-1) = \alpha(t) \cdot [R(t-1) \cdot \varphi(t)] \cdot [R(t-1) \cdot \varphi(t)]^T, \quad (4.5)$$

where,

$$\alpha(t) = [\varphi^T(t) \cdot R(t-1) \cdot \varphi(t)]^{-1}. \quad (4.6)$$

Obviously, for $\alpha(t)$ to exist, $\varphi(t)$ cannot be a zero vector. To ensure the existence of equation (4.6) the following condition should be adopted [1],

$$\alpha(t) = 0 \quad \text{if} \quad |\varphi(t)| < \varepsilon, \quad (4.7)$$

where ϵ , is a dead zone; has to be defined for an individual system based on the noise level.

Applying these ideas to the original equations for the information matrix we get [1],

$$R(t) = R_1(t-1) + \lambda R_2(t-1) + \varphi(t)\varphi^T(t) = \bar{R}(t-1) + \varphi(t)\varphi^T(t), \quad (4.8)$$

where, $\bar{R}(t-1)$ is the modified information matrix given by,

$$\bar{R}(t-1) = (I - M(t))R(t-1), \quad (4.9)$$

$$M(t) = (1 - \lambda)\alpha(t)R(t-1)\varphi(t)\varphi^T(t). \quad (4.10)$$

Assuming $N(t) = I - M(t)$ as the forgetting factor the new information matrix equation can be expressed as follows [1],

$$\bar{R}(t-1) = N(t) \cdot R(t-1) = R(t-1) - M(t)R(t-1). \quad (4.11)$$

Knowing that $M(t)R(t-1)$ is nonnegative definite one can conclude that [1],

$$\bar{R}(t-1) \leq R(t-1). \quad (4.12)$$

Inequality 4.12 means that each update causes old information to be forgotten.

Using the definition of the two component matrices and some fundamental matrix conversion lemmas a new algorithm can be developed for parameter estimation. It is defined by the following set of equations,

$$\Theta(t) = \Theta(t-1) + K(t) \cdot [y(t) - \varphi^T(t) \cdot \Theta(t-1)] \quad (4.13)$$

$$K(t) = P(t) \cdot \varphi(t) = \frac{\bar{P}(t-1) \cdot \varphi(t)}{1 + \varphi^T(t) \cdot P(t-1) \cdot \varphi(t)} \quad (4.14)$$

$$\bar{P}(t-1) = P(t-1) + \frac{1-\mu}{\mu} \cdot \frac{\varphi(t) \cdot \varphi^T(t)}{\varphi^T(t) \cdot R(t-1) \cdot \varphi(t)}, \quad |\varphi(t)| > \varepsilon \quad (4.15)$$

$$\bar{P}(t-1) = P(t-1), \quad |\varphi(t)| \leq \varepsilon \quad (4.16)$$

$$P(t) = \bar{P}(t-1) - \frac{\bar{P}(t-1) \cdot \varphi(t) \cdot \varphi^T(t) \cdot \bar{P}(t-1)}{1 + \varphi^T(t) \cdot \bar{P}(t-1) \cdot \varphi(t)} \quad (4.17)$$

$$R(t) = [I - M(t)] \cdot R(t-1) + \varphi(t) \cdot \varphi^T(t) \quad (4.18)$$

$$M(t) = (1-\mu) \cdot \frac{R(t-1) \cdot \varphi(t) \cdot \varphi^T(t)}{\varphi^T(t) \cdot R(t-1) \cdot \varphi(t)}, \quad |\varphi(t)| > \varepsilon \quad (4.19)$$

$$M(t) = 0, \quad |\varphi(t)| \leq \varepsilon \quad (4.20)$$

Authors of [1] present a simulation proving that this algorithm behaves well in a variety of situations, such as during parameter jumps or during prolonged periods of poorly excited data. This algorithm is not only effective in preventing parameter covariance wind-up but it is also simple and fast, making it a good candidate for on-line applications.

Chapter 5

Comparison of two methods

In order to understand the performance aspects of each method and decide what factors should be considered in selecting a superior one, a series of simulations was conducted. Each experiment is done under similar conditions (same number of samples, identical input function, process and input noise). As the primary measure of performance the residual factor MSE (Mean Square Error) as defined in [4] was used,

$$MSE = \frac{\sum (residuals)^2}{No_samples}. \quad (5.1)$$

The MSE measures how closely the predicted output follows the actual one, over the entire duration of the simulation. The input function changes in each experiment in such a way that its noise contents increases gradually. The plant output is simulated using the second order ARMA model and is represented by the following equation,

$$y(t) + a_1 \cdot y(t-1) + a_2 \cdot y(t-2) = b_1 \cdot u(t-1) + b_2 \cdot u(t-2) + e(t), \quad (5.2)$$

where,

$$\left\{ \begin{array}{l} a_1 = -1.4 \text{ for } 0 \leq t \leq 600 \\ a_1 = -1.0 \text{ for } t > 600 \\ a_2 = 0.8 \\ b_1 = 1.0 \\ b_2 = 0.2 \text{ for } t \leq 200 \\ b_2 = 0.1 \text{ for } t > 200, \end{array} \right.$$

and $e(t)$ is the process noise.

In order to ensure that no parameter wind-up happens each simulation is conducted for 4500 samples. Convergence of estimated parameters takes place after approximately 2500 samples in both cases. A most favorable value of the forgetting factor was determined in each case experimentally. It is very typical of forgetting factor to have just a single value for which MSE approaches a minimum. Selection of the forgetting factor has a significant impact on the quality of system estimation, the closer it is to the most favorable value the better estimation is achieved. Figure 5.0.1 is hand drawn plot of a typical MSE value against the value of forgetting factor for given noise conditions. One can observe an existence of a single minimum of MSE in each experiment.

Figure 5.0.1 Typical plot of MSE as a function of forgetting factor

5.1 Experiment 1

In this experiment, the input function is graphically presented in Figure 5.1.1 and represents a 'comb' binary waveform with amplitude +1 and -1 and period of 20 steps. After step 2000 the input settles at +1.0 and does not change for the duration of the experiment. Noise is kept at a very low level (0.05) and is represented by the normal distribution with mean of $\bar{x} = 0$ and a variance of $\sigma = 1$.

The output is given by equation 5.2 and is represented graphically in figure 5.1.2 Figures 5.1.3 and 5.1.4 represent the actual versus predicted outputs for Kalman Filter and Directional Forgetting respectively. Darker spots represent areas of discrepancy where estimated output is not overlapped by the actual output symbolized by the lighter waveform. Figures 5.1.9 and 5.1.10 represent residuals for both methods. They are very similar and reveal almost identical behaviour at the time of parameter jump. One behaviour that makes a difference is the fact that the CFA identified the system quicker at the beginning while DF takes approximately 150 samples. Finally, identified parameters are plotted in fig. 5.1.11 and 5.1.12 and show that DF is slower although CFA overshoots at parameter jump points. This could be even better observed in figures 5.1.13 and 5.1.14 where only 100 points are presented around step 600. The sudden change in parameters is almost instantaneously detected and corrected by CFA method while DF took approximately 50 steps. This difference could be explained by the fact of limited forgetting (by virtue of forgetting factor) for DF method that takes more time and is gradual. Figures 5.1.5, 5.1.6, 5.1.15, 5.1.16 show the initial steps of identification for both methods. In this case performance is very similar since the DF method had no previous knowledge to forget. It is also worth noting that CFA method smoothly detected parameter values while DF overshoots at first but quickly

converged to the proper parameters. Some interesting observation can be made from figures 5.1.7 and 5.1.8 where the simulated and actual outputs of the system are presented in the narrow range around step 600 where parameters suddenly jumped. In the case of the CFA method the identification became very good after approximately 5 steps while DF needed at least 50 steps to accomplish the same level of accuracy.

The numerical residual analyses indicate that the Kalman filter method is more accurate than Directional Forgetting algorithm in this case (MSE= 0.0058 versus 0.0085). The important fact to remember is that results obtained for DF method are strongly dependent on the choice of forgetting factor. In all experiments the most favorable (or close to it) forgetting factor was used. The value of the most favorable forgetting factor is presented in each method and that value was found experimentally. In this case the value was set to 0.65. Similarly, in the case of the CFA method the values of rejection and acceptance thresholds are determined experimentally. Although Chowdhury in [2] defines the two parameters, that definition is of limited practical value. One has to consider the levels of noise to ensure proper performance of the algorithm. In this case rejection threshold (Tr) was set to 0.06 and acceptance threshold to 0.01 to provide most favorable performance. Analyzing the convergence of parameters one might notice that the Kalman method performed better than DF method. The changes were very quick for CFA method although it results in greater overshooting than DF method.

Figure 5.1.1 Actual input of the simulated system

Figure 5.1.2 Actual output of the simulated system

Figure 5.1.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.1.4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.1.5 Actual (dashed line) and simulated (dotted line) system output at initial steps for CFA method

Figure 5.1.6 Actual (dashed line) and simulated (dotted line) system output at initial steps for DF method

Figure 5.1.7 Actual (dashed line) and estimated (dotted line) output around 600 step for CFA method

Figure 5.1.8 Actual (dashed line) and estimated (dotted line) output around 600 step for DF method

Figure 5.1.9 Residuals for simulated system using CFA method

Figure 5.1.10 Residual for simulated system using DF method

Figure 5.1.11 Estimates of parameters for simulated system using CFA method

Figure 5.1.12 Estimates of parameters for simulated system using DF method

Figure 5.1.13 Estimate of parameters around step 600 for CFA method

Figure 5.1.14 Estimate of parameters around step 600 for DF method

Figure 5.1.15 Estimate of parameters at the initial steps for CFA method

Figure 5.1.16 Estimate of parameters at the initialization stage for DF method

5.2 Experiment 2

In this experiment, the input function is graphically presented in Figure 5.2.1 and it is essentially a white noise waveform with the mean $\bar{x} = 0$ and variance $\sigma = 0.01$.

System noise is kept at relatively high level and is represented by the normal distribution with a mean of $\bar{x} = 0$ and variance of $\sigma = 0.1$.

The output is given by equation 5.2 and is represented graphically in figure.5.2.2. Figures 5.2.3 and 5.2.4 represent the estimated versus actual outputs for the CFA and the DF respectively. Darker spots represent areas of discrepancy where estimated output is not overlapped by actual output symbolized by lighter waveform. The estimated output has been plotted with a greater degree of detail in figures 5.2.5 and 5.2.6. Both methods deliver very comparable results with respect to accuracy and speed of convergence after a jump change in parameters that occur in step 600.

Figures 5.2.7 and 5.2.8 represent residuals for both methods. They are both very similar and it is hard to see any differences from that graphical representation. The numerical residual analyses indicate that both methods are essentially even in this case (MSE = 0.0108 versus 0.0109). Again, the results obtained for DF method are strongly dependent on the choice of forgetting factor. The value of the most favorable forgetting factor in this case was determined experimentally to be 0.93. Similarly, in the case of the Kalman Filter the values of rejection and acceptance thresholds have important impact on the final results. For this simulation the most favorable results were received for rejection threshold (Tr) of 0.08 and the acceptance threshold (Ta) of 0.02. This choice is based on the performance of numerous simulation trials. Analyzing the convergence of parameters, one might notice that both methods reveal a slight problem associated with drifting parameters.

A prolonged simulation (25,000 points) has been conducted on both systems. It appears that both methods exhibit similar trouble behaviour for two parameters. After step 5000 both parameter converge but still remained noisy (wondering around the proper value). This was the case for both, CFA and DF methods. In this particular example the system noise is relatively high (white noise with 0.1 variance) while input is very poorly excited (white noise with 0.01 variance) and this fact can contribute to this behaviour. Despite this problem both methods predicted the system with relatively high degree of accuracy.

Figure 5.2 1 Actual input of the simulated system

Figure 5.2.2 Actual output of the simulated system

Figure 5.2.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.2.4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.2.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method

Figure 5.2.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method

Figure 5.2.7 Residual for simulated system using CFA method

Figure 5.2.8 Residual for simulated system using DF method

Figure 5.2.9 Estimates of parameters for simulated system using CFA method

Figure 5.2.10 Estimates of parameters for simulated system using DF method

5.3 Experiment 3

In this experiment, the input function is graphically presented in Figure 5.3.1 and constitutes a white noise waveform with mean $\bar{x} = 0$, and variance $\sigma = 0.1$.

Noise is kept at relatively high level and is represented by the normal distribution with a mean of $\bar{x} = 0$ and a variance of $\sigma = 0.1$.

The output is depicted by equation 5.2 and is represented graphically in figure 5.3.2. Figures 5.3.3 and 5.3.4 represent the actual versus predicted outputs for CFA and DF methods respectively. Darker spots represent areas of discrepancy where the estimated output is not overlapped by actual output symbolized by the lighter waveform. Estimated output has been further observed with a greater degree of detail in figures 5.3.5 and 5.3.6. Both methods deliver very comparable results with respect to accuracy and speed of convergence after a jump change in the parameters that occurs at step 600.

Figures 5.3.7 and 5.3.8 represent residuals for both methods. They are both very similar and it is hard to see any differences from that graphical representation.

The numerical residual analyses indicate that both methods are essentially even in this case, MSE= 0.0110 for DF versus 0.0113 for CFA. Again, the results obtained for the DF method are strongly dependent on the choice of forgetting factor. The value of the most favorable forgetting factor in this case was determined to be 0.86. Similarly, in the case of the Kalman Filter, the values of rejection and acceptance thresholds have an important impact on the final results. For this simulation the most favorable results were received for rejection threshold (T_r) set at 0.05 and the acceptance threshold (T_a) set at 0.02. This choice is based on the performance of numerous simulation trials. Estimated parameters are graphically represented in figures 5.3.9 and 5.3.10 for the CFA and the DF methods

respectively. Analyzing the convergence of the parameters, one will notice that the Kalman Filter method was somewhat faster than the Directional Forgetting method. Also, in case of DF, the parameters had a tendency to higher noise content throughout the duration of this simulation. Figures 5.3.11 and 5.3.12 compare the behaviour of estimated parameters at initial stage of this simulation for both methods. Both methods expose very similar behaviour with the exception of slight overshooting for the DF algorithm. This results in a slightly elevated residual during the first 10 steps and contributes to the overall result.

Figure 5.3. 1 Actual input of the simulated system

Figure 5.3.2 Actual output of the simulated system

Figure 5.3. 3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.3. 4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.3.5 Actual (dashed line) and estimated (dotted line) output around 600 step for CFA method

Figure 5.3.6 Actual (dashed line) and estimated (dotted line) output around 600 step for DF method

Figure 5.3. 7 Residuals for simulated system using CFA method

Figure 5.3. 8 Residuals for simulated system using DF method

Figure 5.3.9 Estimates of parameters for simulated system using CFA method

Figure 5.3.10 Estimates of parameters for simulated system using DF method

Figure 5.3.11 Estimates of parameters at initial stage of simulation for CFA method

Figure 5.3.12 Estimates of parameters at initial stage of simulation for DF method

5.4 Experiment 4

In this experiment, the input function is graphically presented in Figure 5.4.1 and it represents a white noise waveform with mean $\bar{x} = 0$, and variance $\sigma = 1.0$.

System noise is kept at relatively high level and is represented by the normal distribution with a mean of $\bar{x} = 0$ and variance of $\sigma = 0.1$.

The output is depicted by equation 5.2 and is represented graphically in figure 5.4.2. Figures 5.4.3 and 5.4.4 represent the actual versus predicted outputs for the CFA and the DF methods respectively. Darker spots represent areas of discrepancy where estimated output is not overlapped by actual output symbolized by the lighter waveform. Estimated output has been further observed with a greater degree of detail in figures 5.3.5 and 5.3.6. Both methods deliver very comparable results with respect to accuracy and speed of convergence after a jump change in parameters that occur at step 600. A similar situation exists for the jump occurring at step 200 that was illustrated in figures 5.4.7 and 5.4.8 respectively for CFA and DF methods.

The graphical representation of residuals does not indicate any method being superior; see figures 5.4.9 and 5.4.10 for the CFA and the DF methods respectively. Numerical analysis of residuals indicate that the Kalman Filter algorithm in this case is more accurate, $MSE = 0.0119$ for the CFA versus 0.0204 for the DF method. Again in this case, the results obtained for the DF method are strongly dependent on the choice of forgetting factor. The value of the most favorable forgetting factor in this case was determined to be 0.56 . Similarly, in case of the Kalman Filter, the values of rejection and acceptance thresholds have important impact on final results. For this simulation the most favorable results were received for rejection threshold (Tr) of 0.05 and the acceptance threshold (Ta) of 0.02 .

Estimated parameters are graphically represented in figures 5.3.11 and 5.3.12 for the CFA and the DF methods respectively. Analyzing the convergence of parameters one might notice that the Kalman filter method generally was faster in establishing stable parameters although each 'jump' point resulted in overshooting. This was not true for the DF method where parameter jump points are smoother.

Figure 5.4.1 Actual input of the simulated system

Figure 5.4.2 Actual output of the simulated system

Figure 5.4. 3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.4. 4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.4.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method

Figure 5.4.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method

Figure 5.4.7 Actual (dashed line) and estimated (dotted line) output around step 200 for CFA method

Figure 5.4.8 Actual (dashed line) and estimated (dotted line) output around step 200 for DF method

Figure 5.4.9 Residual for simulated system using CFA method

Figure 5.4.10 Residual for simulated system using DF method

Figure 5.4.11 Estimates of parameters for simulated system using CFA method

Figure 5.4.12 Estimates of parameters for simulated system using DF method

5.5 Experiment 5

In this experiment the objective is to compare performance of both methods in the condition of high system noise. The input function is graphically presented in Figure 5.5.1 and it represents a white noise waveform with amplitude 0.1, mean $\bar{x} = 0$, and variance $\sigma = 1$. System noise is kept at very high level (1.0 amplitude) and is represented by the normal distribution with a mean of $\bar{x} = 0$ and a variance of $\sigma = 1$.

The output is depicted by equation 5.2 and is represented graphically in figure 5.5.2. Figures 5.5.3 and 5.5.4 represent the actual versus predicted outputs for the CFA and the DF methods respectively. In both cases the prediction is rather poor and discrepancies are easy to identify graphically (dark spots on the graphs). Estimated output has been further observed with a greater degree of detail in figures 5.3.5 and 5.3.6. Both methods deliver very comparable results with respect to accuracy and speed of convergence after a jump change in parameters that occurs at step 600. It should also be noticed that because of poor performance step 600 is not clearly visible as a parameter jump.

Figures 5.5.7 and 5.5.8 represent residuals for both methods. They are both very similar although the DF's method seems to produce slightly more accurate parameter prediction. Step 600 is visible for both methods as a disturbance in the system but step 200 blends in with the noise.

The identified parameters are plotted in figures 5.5.7 and 5.5.8 and show that the DF is much smoother. Numerical residual analyses indicate that both estimators perform rather poorly but the Directional Forgetting is more accurate than the Kalman filter algorithm in this case $MSE = 1.0895$ for the DF versus 1.0938 for the CFA. Again in this case results obtained for the DF method are strongly dependent on the choice of forgetting factor. The value of the

most favorable forgetting factor in this case was determined to be 0.98. Similarly, in case of the Kalman Filter, the values of rejection and acceptance thresholds have important impact on final results. For this simulation the most favorable results were received for rejection threshold (T_r) of 0.3 and the acceptance threshold (T_a) of 0.17

Figure 5.5.1 Actual input of the simulated system

Figure 5.5.2 Actual output of the simulated system

Figure 5.5. 3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.5. 4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.5.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method

Figure 5.5.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method

Figure 5.5.7 Residuals of the simulated system for CFA method

Figure 5.5.8 Residuals of the simulated system for DF method

Figure 5.5.9 Estimates of parameters for simulated system using CFA method

Figure 5.5.10 Estimates of parameters for simulated system using DF method

5.6 Experiment 6

In this experiment, the input function is graphically presented in Figure 5.6.1 and represents a white noise waveform with mean $\bar{x} = 0$ and variance $\sigma = 0.1$.

Noise is kept at relatively high level and is represented by the normal distribution with a mean of $\bar{x} = 0$ and variance of $\sigma = 0.5$.

The output is depicted by equation 5.2 and is represented graphically in figure 5.6.2. Figures 5.6.3 and 5.6.4 represent the actual versus predicted outputs for the CFA and the DF methods respectively. In both cases prediction is poor and discrepancies are easy to identify graphically. Estimated output has been further observed with a greater degree of detail in figures 5.6.5 and 5.6.6 around step 600. Both methods deliver very comparable results with respect to accuracy and speed of convergence after a jump change in parameters that occur at step 600.

Figures 5.6.5 and 5.6.6 represent residuals for both methods. They are both very similar and it is hard to conclude on any differences from that graphical representation. The numerical residual analyses indicate that the Kalman filter algorithm is slightly more accurate than the Directional Forgetting method in this case (MSE = 0.2722 versus 0.2772). Again in this experiment results obtained for the DF method are strongly dependent on the choice of forgetting factor. The value of the most favorable forgetting factor in this case was determined experimentally to be 0.95. Similarly in case of the Kalman Filter the value of rejection and acceptance thresholds have important impact on the final results. For this simulation the most favorable results were received for rejection threshold (Tr) of 0.19 and the acceptance threshold (Ta) of 0.1. Analyzing the convergence of parameters one might notice that the Kalman method generally was faster in establishing stable parameters

although each 'jump' point would result in significant initial overshooting. Figures 5.6.11 and 5.6.12 compare the behaviour of estimated parameters at initial stage of this simulation for both methods. Both methods exhibited very similar initial convergence characterized by slight overshooting but then quick stabilization.

Figure 5.6.1 Actual input of the simulated system

Figure 5.6.2 Actual output of the simulated system

Figure 5.6. 1 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.6. 2 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.6.5 Actual (dashed line) and estimated (dotted line) output around step 600 for CFA method

Figure 5.6.6 Actual (dashed line) and estimated (dotted line) output around step 600 for DF method

Figure 5.6.7 Residual for simulated system using CFA method

Figure 5.6.8 Residual for simulated system using DF method

Figure 5.6.9 Estimates of parameters for simulated system using CFA method

Figure 5.6.10 Estimates of parameters for simulated system using DF method

Figure 5.6.11 Estimated parameters at initial steps for CFA method

Figure 5.6.12 Estimated parameters at initial steps for DF method

5.7 Experiment 7

In this experiment, the input function is graphically presented in Figure 5.1.1 and represents a ‘comb’ binary waveform with amplitude +1 and –1 and period of 20 steps. After step 2000 function settles at +1.0 and does not change for the duration of the experiment. Noise is kept at a constant level and is represented by the normal distribution with mean of $\bar{x} = 0$ and variance of $\sigma = 1$.

The plant output is simulated using the second order ARMA model and is represented by the following equation,

$$y(t) + a_1 \cdot y(t-1) + a_2 \cdot y(t-2) = b_1 \cdot u(t-1) + b_2 \cdot u(t-2) + e(t), \quad (5.17)$$

where, $e(t)$ represents process noise and,

$$\left\{ \begin{array}{ll} a_1 = -1.4 & \text{for } 0 \leq t \leq 600 \\ a_1 = -0.6 & \text{for } t > 600 \\ a_2 = 0.8 \\ b_1 = 1.0 \\ b_2 = 0.2 & \text{for } t \leq 200 \\ b_2 = 0.9 & \text{for } t > 200. \end{array} \right.$$

This system is very similar to the one simulated in the Experiment 1 except for higher noise content and more dramatic parameters jump.

Figures 5.7.3 and 5.7.4 represent the actual versus predicted outputs for the Kalman Filter and the Directional Forgetting methods respectively. Darker spots represent areas of discrepancy where estimated output is not overlapped by actual output symbolized by the lighter waveform. Estimated output has been further observed with a greater degree of detail

in figures 5.7.5 and 5.7.6. Both methods deliver comparable results with respect to accuracy and speed of convergence after a jump change in parameters that occur at step 600.

Figures 5.7.7 and 5.7.8 represent residuals for both methods. They are very similar and reveal almost identical behaviour. One difference may be observed in the regions following sudden parameters jump e.g. after step 600. The Directional Forgetting method takes slightly longer to converge and this behaviour results in a sloping residual characteristic on the stretch of at least 50 steps.

Finally, identified parameters are plotted in fig. 5.1.9 and 5.1.10 and show that the DF is slower although the CFA slightly overshoots at rapid transition points.

The residual analyses indicate that the Kalman filter method is more accurate than the Directional Forgetting algorithm in this case (MSE= 0.0136 versus 0.0182). The important fact to remember is that results obtained for the DF method are strongly dependent on the choice of forgetting factor. In the experiments the most favorable (or close to it) factor was used. In this case that value was set at 0.5. Similarly, in case of the CFA method the values of rejection and acceptance thresholds are determined experimentally. Although Chowdhury in [2] gives definitions of the two parameters (equations 3.20 and 3.21 in this thesis) they are of limited practical value. One has to consider the level of noise to ensure proper performance of the algorithm. In this case the rejection threshold (Tr) was set at 0.08 and the acceptance threshold at 0.02 in order to provide most favorable performance.

Figure 5.7.1 Actual input of the simulated system

Figure 5.7.2 Actual output of the simulated system

Figure 5.7.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.7.4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.7.5 Actual (dashed line) and estimated (dotted line) output around 600 step for CFA method

Figure 5.7.6 Actual (dashed line) and estimated (dotted line) output around 600 step for DF method

Figure 5.7.7 Residual for simulated system using CFA method

Figure 5.7.8 Residual for simulated system using DF method

Figure 5.7.9 Estimates of parameters for simulated system using CFA method

Figure 5.7.10 Estimates of parameters for simulated system using DF method

5.8 Experiment 8

The objective of this experiment is to demonstrate how quickly a given algorithm can learn a system. In this case there is no parameter jumps or excessive system noise present.

The output function is given by the following relationship,

$$y(t) + 0.8 \cdot y(t-1) + y(t-2) = u(t-1) + 0.7 \cdot u(t-2) + e(t). \quad (5.18)$$

The input function is graphically presented in Figure 5.8.1 and represents a white noise waveform with amplitude 0.1, mean 0, and variance 1.

System noise is kept at a very low level (0.01 amplitude) and is represented by the normal distribution with a mean of $\bar{x} = 0$ and a variance of $\sigma = 1$.

Figures 5.8.3 and 5.8.4 represent the actual versus predicted outputs for the CFA and the DF methods respectively. It is obvious that the CFA method performs better at the initial stage of identification. Full recognition and accurate prediction is accomplished after approximately 10 steps while the DF required at least 50 to match that performance.

Figures 5.8.5 and 5.8.6 represent residuals for both methods. Also, in this case it is obvious that the CFA algorithm performs better at the initial stage of identification. A significant portion of error that contributed to overall result for MSE originates from this stage. The residual analyses indicate that the Kalman filter algorithm performed very well and its MSE was only $2.8115 \cdot 10^{-4}$ versus $3.900 \cdot 10^{-3}$ for the DF method. Again in this case results obtained for the DF method are strongly dependent on the choice of forgetting factor. The value of the most favorable forgetting factor in this case was determined to be 0.35. Figures 5.8.7 and 5.8.8 represent estimated parameters for the CFA and the DF methods. The CFA parameters converged very quickly while the DF parameters required almost 50 steps to get to that level.

Figure 5.8.1 Actual input of the simulated system

Figure 5.8.2 Actual output of the simulated system

Figure 5.8.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.8.4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.8.5 Residual for simulated system using CFA method

Figure 5.8.6 Residual for simulated system using DF method

Figure 5.8.7 Estimates of parameters for simulated system using CFA method

Figure 5.8.8 Estimates of parameters for simulated system using DF method

5.9 Experiment 9

The objective of this experiment is to demonstrate how effectively a given method is in identifying systems containing frequent parameter jumps.

The input function is graphically presented in Figure 5.8.1 and represents a white noise waveform with amplitude 0.1, mean 0, and variance 1.

System noise is kept at a very low level (0.01 amplitude) and is represented by the normal distribution with mean of $\bar{x} = 0$ and a variance of $\sigma = 1$.

The simulated system output is represented in figure 5.9.2 and given by the following equation,

$$y(t) + a \cdot y(t-1) + 0.8 \cdot y(t-2) = u(t-1) + 0.7 \cdot u(t-2) + e(t). \quad (5.23)$$

where

$$a = \begin{cases} 0.1 & \text{for } t \leq 400 \\ 0.2 & \text{for } 400 < t \leq 700 \\ 0.3 & \text{for } 700 < t \leq 1000 \\ 0.6 & \text{for } 1000 < t \leq 1300 \\ 0.7 & \text{for } 1300 < t \leq 1600 \\ 0.8 & \text{for } 1600 < t \leq 2000 \end{cases}$$

Figures 5.9.3 and 5.9.4 represent the actual versus predicted outputs for the Kalman filter and the Directional Forgetting methods respectively. Darker spots represent areas of discrepancy where estimated output is not overlapped by actual output symbolized by the lighter waveform.

Figures 5.9.5 and 5.9.6 represent residuals for both methods. It is obvious that the CFA method performs better in this case. The speed of convergence is slow using the DF

algorithm causing significant disturbances after each parameter jump. Very much the same behaviour is revealed from parameter graphs presented in figures 5.9.7 and 5.9.8. In the case of the DF algorithm only jumps in steps 1000 and 1300 are clearly distinct. All other jumps can be identified from residuals but not parameters.

Numerical analysis of residuals confirms observations from graphical representation. The CFA method had MSE of $2.843 \cdot 10^{-4}$ while the DF method $8.343 \cdot 10^{-4}$.

Figure 5.9.1 Actual input of the simulated system

Figure 5.9.2 Actual output of the simulated system

Figure 5.9.3 Actual (dashed line) and estimated (dotted line) system outputs for CFA

Figure 5.9.4 Actual (dashed line) and estimated (dotted line) system outputs for DF

Figure 5.9.5 Residual for simulated system using CFA method

Figure 5.9.6 Residual for simulated system using DF method

Figure 5.9.7 Estimates of parameters for simulated system using CFA method

Figure 5.9.8 Estimates of parameters for simulated system using DF method

Chapter 6

Conclusions

Nine experiments completed in the scope of this thesis explore a wide range of different systems and circumstances. In the majority of those the CFA method proved to be more accurate than DF method. In general, for low content of noise in the input and process, the Kalman filter is faster and more accurate than the DF algorithm. This is especially evident for systems simulated in experiments 8 and 9. As system noise content was increased, the differences in performance of both methods slowly diminished. For high level of noise, for example in experiment #5, the DF method proved to be superior. Summary of the results is presented in the table 6.1.

One of the serious challenges in comparing the two methods was optimization of parameters that greatly affect performance and accuracy. In case of DF algorithm it is the necessity of finding the most favorable setting of forgetting factor. Arbitrarily selecting the forgetting factor causes the method to perform below its full capabilities. All known practices of choosing the value of forgetting factor (like the one presented in [4]) become of limited use in this case. For the benefit of the practical value of the DF algorithm one should conduct at least some form of optimization of the forgetting factor. My experience shows that this value heavily depends on the content of noise but it is independent of the input function. This leads to the hypothesis that probably the most appropriate variable to use in the selection criteria of Forgetting Factor is noise/signal ratio. Another very important conclusion is that for the higher noise content, the most favorable value of forgetting factor shifts towards 1.0 while for lower noise it could be significantly smaller (closer to 0). This is fairly obvious considering the fact that if the forgetting factor is small, the algorithm starts following the

noise and the parameters estimated become excessively sensitive. If the forgetting factor is selected to be very high, parameter jumps are poorly detected and convergence is sluggish. This opens a concept of a dynamical forgetting factor that could be adjusted on-line depending on existing conditions. Generally, the forgetting factor should be kept at a relatively high value (close to 1.0) at steady condition but when some abrupt transition exists it should be dynamically switched to some low value to allow the system to forget the past that is no longer valid.

The Kalman Filter method also requires proper selection of parameters but in this case these are, T_r – rejection threshold and T_a – acceptance threshold. The selection of thresholds affects accuracy of the system and is strongly dependent on the noise content. The rejection threshold has to be selected in such a way that only major parameter jumps are interpreted as such and all noise related residuals are filtered out. A similar situation applies to the acceptance threshold where its excessively low value may force overreaction of the algorithm to the system noise resulting in very sensitive parameters. Also in this case, adaptive regulation of threshold levels could provide a significant improvement to the accuracy of CFA method.

The idea of using the hypothesis testing proved to be very advantageous to the Kalman Filter method. Judging from similarities of the two tested methods it is obvious that the DF algorithm could also benefit from adopting this technique. This should be considered as one of the possible recommendations to improve performance of DF algorithm in detecting the parameter jumps.

With respect to speed of calculation, the DF algorithm proves to be faster than the CFA. This conclusion can be supported by the count of operations conducted on both methods. In order

to complete a single step of the recursive calculation the CFA method needed the following matrix operations: thirteen multiplications, and five additions. In order to accomplish the same, the DF method required only eleven multiplications and six additions. It was experimentally observed that DF calculations were completed approximately 30% faster.

Overall, it is very hard to determine what method was better. Different criteria favor a different method as a preferred one. Neither of the two methods suffered from the parameter blow-up phenomena. Depending on circumstances and the computation environment one should select a method that better satisfies existing needs. As indicated in the thesis, further improvement of performance can be accomplished by adopting dynamic factor adjustment for both methods.

Table 6.1 Summary of results

EXP. #	SYSTEM DESCRIPTION	PROCESS NOISE LEVEL	INPUT NOISE LEVEL	METHOD	METHOD PARAMETER	MSE
1	Output: eq. 5.2 Input: 'comb' binary waveform settling at +1 after step 2000	0.05	0.0	CFA	$T_r = 0.06$ $T_a = 0.01$	0.0058
				DF	$F.F. = 0.65$	0.0085
2	Output: eq. 5.2 Input: white noise	0.1	0.01	CFA	$T_r = 0.08$ $T_a = 0.02$	0.0108
				DF	$F.F. = 0.93$	0.0109
3	Output: eq. 5.2 Input: white noise	0.1	0.1	CFA	$T_r = 0.05$ $T_a = 0.02$	0.0110
				DF	$F.F. = 0.86$	0.0113
4	Output: eq. 5.2 Input: white noise	0.1	1.0	CFA	$T_r = 0.05$ $T_a = 0.02$	0.0199
				DF	$F.F. = 0.56$	0.0304
5	Output: eq. 5.2 Input: white noise	1.0	0.1	CFA	$T_r = 0.3$ $T_a = 0.17$	1.0938
				DF	$F.F. = 0.96$	1.0895
6	Output: eq. 5.2 Input: white noise	0.5	0.1	CFA	$T_r = 0.19$ $T_a = 0.1$	0.2722
				DF	$F.F. = 0.95$	0.2772
7	Output: eq. 5.2 Input: 'comb' binary waveform settling at +1 after step 2000	0.1	0.0	CFA	$T_r = 0.08$ $T_a = 0.02$	0.0136
				DF	$F.F. = 0.5$	0.0182
8	Output: eq. 5.18 Input: white noise	0.01	0.1	CFA	$T_r = 0.08$ $T_a = 0.001$	0.00028
				DF	$F.F. = 0.35$	0.0039
9	Output: eq. 5.23 Input: white noise	0.01	0.1	CFA	$T_r = 0.3$ $T_a = 0.001$	0.000284
				DF	$F.F. = 0.1$	0.00083

Appendix A

M-Files listings

7.1 Input1.m

```
clear
%binary comb (-1;1) for input
%y(t)+a1y(t-1)+a2y(t-2)=b1u(t-1)+b2u(t-2)+e(t)
%The parameters are given by:
%a1 = -1.4                                for 0 <= t <=600
%a2 = -1.0                                for t > 600
%b1 = 0.2                                  for t <= 200
%b2 = 0.1                                  for t > 200
% a second-order system example:y(t)+a1*y(t-1)+a2*y(t-2)=b1*u(t-1)+b2*u(t-
2)
% b-parameter and then a-parameter changes suddenly

N=5000;N2=1200;

%input data
u=1.0*ones(N,1); %just a bunch of ones

%the following loop makes every 20 elements =-1 for 20 elements (makes a
comb function)
for i=1:2000/40;
    u(40*(i-1)+1:40*(i-1)+20)=-1.0;
end

randn('seed',127); %defines a random function generator
epsin=0.05*randn(N,1);% white noise
a1=-1.4; a2=-1.0;
b1=0.2; b2=0.1;
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th2=arx2th([1 a1 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
th3=arx2th([1 a2 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.

% output data
%Simulates a given dynamic system;
y(1:200)=idsim([u(1:200) epsin(1:200)],th1);
y(201:600)=idsim([u(201:600) epsin(201:600)],th2);
y(601:N)=idsim([u(601:N) epsin(601:N)],th3);
y(201)=y(200);
y(601)=y(600);

tmax=N;
t=1:tmax;
z=[y' u];
n = 2;                                % the order of ARMA gain
```

```
%***** Plotting the results
```

```
figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')
```

```
figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')
```

7.2 Input2.m

```
clear
% random noise with 0.01 variance for input
%y(t)+a1y(t-1)+a2y(t-2)=b1u(t-1)+b2u(t-2)+e(t)
%The parameters are given by a1 = -1.4 for 0 <= t <=600

%a1= -1.0 for t > 600
%a2=0.8, b1=1 b2 = 0.2 for t <= 200
% b2= 0.1 for t > 200
% a second-order system example:y(t)+a1*y(t-1)+a2*y(t-2)=b1*u(t-1)+b2*u(t-2)
% b-parameter and then a-parameter changes suddenly

N=5000;N2=1200;

%input data
u=0.01*randn(N,1);

randn('seed',127); %defines a random function generator
epsin=0.1*randn(N,1);% white noise
a1=-1.4; a2=-1.0;
b1=0.2; b2=0.1;
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th2=arx2th([1 a1 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
th3=arx2th([1 a2 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.

% output data
%Simulates a given dynamic system;
y(1:200)=idsim([u(1:200) epsin(1:200)],th1);
y(201:600)=idsim([u(201:600) epsin(201:600)],th2);
y(601:N)=idsim([u(601:N) epsin(601:N)],th3);
y(201)=y(200);
y(601)=y(600);

tmax=N;
t=1:tmax;
z=[y' u];
n = 2; % the order of ARMA gain
```

```
%***** Plotting the results
```

```
figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')
```

```
figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')
```

7.3 Input3.m

```
clear
% random noise with 0.1 variance for input
% $y(t)+a_1y(t-1)+a_2y(t-2)=b_1u(t-1)+b_2u(t-2)+e(t)$ 
%The parameters are given by  $a_1 = -1.4$  for  $0 \leq t \leq 600$ 

% $a_1 = -1.0$  for  $t > 600$ 
% $a_2 = 0.8$ ,  $b_1 = 1$  b2 = 0.2 for  $t \leq 200$ 
% b2 = 0.1 for  $t > 200$ 
% a second-order system example: $y(t)+a_1y(t-1)+a_2y(t-2)=b_1u(t-1)+b_2u(t-2)$ 
% b-parameter and then a-parameter changes suddenly

N=5000;N2=1200;

%input data
u=0.1*randn(N,1);

randn('seed',127); %defines a random function generator
epsin=0.1*randn(N,1);% white noise
a1=-1.4; a2=-1.0;
b1=0.2; b2=0.1;
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th2=arx2th([1 a1 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
th3=arx2th([1 a2 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.

% output data
%Simulates a given dynamic system;
y(1:200)=idsim([u(1:200) epsin(1:200)],th1);
y(201:600)=idsim([u(201:600) epsin(201:600)],th2);
y(601:N)=idsim([u(601:N) epsin(601:N)],th3);
y(201)=y(200);
y(601)=y(600);

tmax=N;
t=1:tmax;
z=[y' u];
n = 2; % the order of ARMA gain

%***** Plotting the results
```



```
figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')
```

```
figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')
```

7.4 Input4.m

```
clear
% random noise with 1 variance for input
%y(t)+a1y(t-1)+a2y(t-2)=b1u(t-1)+b2u(t-2)+e(t)
%The parameters are given by
%a1 = -1.4                                for 0 <= t <=600
%a1 = -1.0                                for t > 600
%a2=0.8, b1=1    b2 = 0.2                  for t <= 200
%          b2= 0.1                          for t > 200
% a second-order system example:y(t)+a1*y(t-1)+a2*y(t-2)=b1*u(t-1)+b2*u(t-
2)
% b-parameter and then a-parameter changes suddenly

N=5000;N2=1200;

%input data
u=1*randn(N,1);

randn('seed',127); %defines a random function generator
epsin=0.1*randn(N,1);% white noise
a1=-1.4; a2=-1.0;
b1=0.2; b2=0.1;
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th2=arx2th([1 a1 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
th3=arx2th([1 a2 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.

% output data
%Simulates a given dynamic system;
y(1:200)=idsim([u(1:200) epsin(1:200)],th1);
y(201:600)=idsim([u(201:600) epsin(201:600)],th2);
y(601:N)=idsim([u(601:N) epsin(601:N)],th3);
y(201)=y(200);
y(601)=y(600);

tmax=N;
t=1:tmax;
z=[y' u];
n = 2;                                % the order of ARMA gain

%***** Plotting the results

figure(1)
plot(u)
title('Actual input')
```

```
xlabel('Time Steps')
```

```
figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')
```

7.5 Input5.m

```
clear
% random noise with 0.1 variance for input
%y(t)+a1y(t-1)+a2y(t-2)=b1u(t-1)+b2u(t-2)+e(t)
%The parameters are given by a1 = -1.4                for 0 <= t <=600

%a1= -1.0                for t > 600
%a2=0.8, b1=1            b2 = 0.2        for t <= 200
%                b2= 0.1                for t > 200
% a second-order system example:y(t)+a1*y(t-1)+a2*y(t-2)=b1*u(t-1)+b2*u(t-2)
% b-parameter and then a-parameter changes suddenly

N=5000;N2=1200;

%input data
u=0.1*randn(N,1);

randn('seed',127); %defines a random function generator
epsin=1*randn(N,1);% white noise
a1=-1.4; a2=-1.0;
b1=0.2; b2=0.1;
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an ARX model.
th2=arx2th([1 a1 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an ARX model.
th3=arx2th([1 a2 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an ARX model.

% output data
%Simulates a given dynamic system;
y(1:200)=idsim([u(1:200) epsin(1:200)],th1);
y(201:600)=idsim([u(201:600) epsin(201:600)],th2);
y(601:N)=idsim([u(601:N) epsin(601:N)],th3);
y(201)=y(200);
y(601)=y(600);

tmax=N;
t=1:tmax;
z=[y' u];
n = 2;                % the order of ARMA gain

%***** Plotting the results

figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')
```

```
figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')
```

7.6 Input6.m

```
clear
%binary comb (-1;1) for input
%y(t)+a1y(t-1)+a2y(t-2)=b1u(t-1)+b2u(t-2)+e(t)
%The parameters are given by a1 = -1.4                      for 0 <= t <=600

%a1= -1.0                                                    for t > 600
%a2=0.8, b1=1          b2 = 0.2                             for t <= 200
%                      b2= 0.1                             for t > 200
% a second-order system example:y(t)+a1*y(t-1)+a2*y(t-2)=b1*u(t-1)+b2*u(t-
2)
% b-parameter and then a-parameter changes suddenly

N=5000;N2=1200;

%input data
u=0.1*randn(N,1); %just a bunch of ones

%the following loop makes every 20 elements ==-1 for 20 elements (makes a
comb function)
for i=1:2000/40;
    u(40*(i-1)+1:40*(i-1)+20)=-1.0;
end

randn('seed',127); %defines a random function generator
epsin=0.5*randn(N,1);% white noise
a1=-1.4; a2=-1.0;
b1=0.2; b2=0.1;
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th2=arx2th([1 a1 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
th3=arx2th([1 a2 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
%th1=arx2th([1 a1 a3],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
%th2=arx2th([1 a1 0.9],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
%th3=arx2th([1 a2 a3],[0 b3 b2],1,1); %Constructs the THETA-format for an
ARX model.

% output data
%Simulates a given dynamic system;
y(1:200)=idsim([u(1:200) epsin(1:200)],th1);
y(201:600)=idsim([u(201:600) epsin(201:600)],th2);
y(601:N)=idsim([u(601:N) epsin(601:N)],th3);
y(201)=y(200);
y(601)=y(600);

tmax=N;
```

```

t=1:tmax;
z=[y' u];
n = 2;

```

```

% the order of ARMA gain

```

```

%***** Plotting the results

```

```

figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')

```

```

figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')

```

7.7 Input7.m

```

clear
%ARMA plant:  $y(t)+a_1y(t-1)+a_2y(t-2)=b_1u(t-1)+b_2u(t-2)+e(t)$ 
%The parameters are given by
% a=[1.0 -1.4 0.8]; b=[0 1 0.2]
% a=[1.0 -1.4 0.8]; b=[0 1 0.9]
% a=[1.0 -0.1 0.8]; b=[0 1 0.9]
for t <=200
for 600 >=t > 200
for t > 600

N=5000;N2=1400;

%input data
u=0.1*randn(N,1);

%the following loop makes every 20 elements ==-1 for 20 elements (makes a
comb function)
for i=1:N2/40;
    u(40*(i-1)+1:40*(i-1)+20)=-1.0;
end

%
randn('seed',127); %defines a random function generator
epsin=0.1*randn(N,1);% white noise

% Simulation of ARMA plant:
a1=-1.4; a2=-0.1;
b1=0.2; b2=0.9;
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th2=arx2th([1 a1 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.
th3=arx2th([1 a2 0.8],[0 1 b2],1,1); %Constructs the THETA-format for an
ARX model.

% Output data
y(1:200)=idsim([u(1:200) epsin(1:200)],th1);
y(201:600)=idsim([u(201:600) epsin(201:600)],th2);
y(601:N)=idsim([u(601:N) epsin(601:N)],th3);
y(201)=y(200);
y(601)=y(600);

```

```

tmax=N;
t=1:tmax;
z=[y' u];    % Phi function
n = 2;       % The order of ARMA gain

```

```

%***** Plotting the results

```

```

figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')

```

```

figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')

```

7.8 Input8.m

```

clear
%ARMA plant:  $y(t)+a_1y(t-1)+a_2y(t-2)=b_1u(t-1)+b_2u(t-2)+e(t)$ 
%The parameters are given by
% a=[1.0 0.8 1.0]; b=[0.0 1.0 0.7]
% The objective is to demonstrate how quickly an algorithm can learn the
system
% There is no parameters jumps
% Recommended for Kalman: TR=0.08;TA=0.01
% Recommended for dir_forg: ff=0.35

N=200;

%input data

u=0.1*randn(N,1);

randn('seed',127); %defines a random function generator
epsin=0.01*randn(N,1); % white noise

% Simulation of ARMA plant:
    th1=arx2th([1.0 0.8 1.0],[0 1.0 0.7],1,1); %Constructs the THETA-format
for an ARX model.

    % Output data
    y(1:N)=idsim([u(1:N) epsin(1:N)],th1);

tmax=N;
t=1:tmax;
z=[y' u];    % Phi function
n = 2;       % The order of ARMA gain

%***** Plotting the results

```

```
figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')
```

```
figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')
```

7.9 Input9.m

```
clear
% ARMA plant:  $y(t) + a_1y(t-1) + a_2y(t-2) = b_1u(t-1) + b_2u(t-2) + e(t)$ 
% The parameters step 6 times
%
% The objective is to demonstrate how frequent changes affect the system.
% Recommended for Kalman: TR=0.35;TA=0.001; very good oerformance.
% Recommended for dir_forg: ff=0.65

N=2000;

%input data

u=0.1*randn(N,1);

randn('seed',127); %defines a random function generator
epsin=0.01*randn(N,1); % white noise

%Simulation of ARMA plant:

a1=0.1; a2=0.2; a3=0.3; a4=0.4; a5=0.5; a6=0.6; b1=0.7
th1=arx2th([1 a1 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th2=arx2th([1 a2 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th3=arx2th([1 a3 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th4=arx2th([1 a4 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th5=arx2th([1 a5 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.
th6=arx2th([1 a6 0.8],[0 1 b1],1,1); %Constructs the THETA-format for an
ARX model.

% output data
%Simulates a given dynamic system;
y(1:400)=idsim([u(1:400) epsin(1:400)],th1);
y(401:700)=idsim([u(401:700) epsin(401:700)],th2);
y(701:1000)=idsim([u(701:1000) epsin(701:1000)],th3);
y(1001:1300)=idsim([u(1001:1300) epsin(1001:1300)],th4);
y(1301:1600)=idsim([u(1301:1600) epsin(1301:1600)],th5);
y(1601:2000)=idsim([u(1601:2000) epsin(1601:2000)],th6);

tmax=N;
```

```

t=1:tmax;
z=[y' u];    % Phi function
n = 2;       % The order of ARMA gain

```

```

%***** Plotting the results

```

```

figure(1)
plot(u)
title('Actual input')
xlabel('Time Steps')

```

```

figure(2)
plot(y)
title('Actual output')
xlabel('Time Steps')

```

7.10 Kalman.m

```

N=200;
N2=1200;
TR=0.08;
TA=0.001;
counter1=0;
counter2=0;
counter3=0;

randn('seed',127);           % Defines a random function
generator

tmax=N;                       % Max time step
t=1:tmax;                     % Discrete time domain
n = 2;                         % The order of ARMA gain

theta_0=zeros(2*n,1);
Q_0 = eye(2*n)*0.0001;       % The covariance of the process
noise
R_0 = 0.0025*ones(1,length(t)); % The covariance of the
measurement noise
P_0 = 5*eye(2*n);            % Initialize the riccatti
(Identity)

theta=theta_0;P=P_0;R=R_0;P=P_0;

xhat(:,1) = theta;           % Initialize the ARMA gain
yhat(1) = 0;                  % Intialize the estimated output
inp(1,:) = zeros(1,n*2);      % Initialize input vector

%*****MAIN
LOOP*****
for l = 1:length(t);

    % This part is for the Kalman / ARMA
    inpl = plantin(u,y,l,n);    % Get the input to the Kalman
    Filteru (u(l-1)...y(l-1)...)
    inpl(l,:) = inpl;
    yhatl(l) = inpl*xhat(:,l);  % Find the estimated output

```

```

    err1(1) = y(1)-yhat1(1);          % Get the error between actual and
estimated output

    A = eye(2*n);                    % the 'A' Matrix for the Kalman
Filter
    C = inpl;                        % the 'C' matrix of the Kalman =
input to the Kalman Filter

    nor = inv(C*P*C' + R(1));
    err2(1) = err1(1)/sqrt(nor);

    if abs(err2(1))>TR; Q = eye(2*n)*0.0001; counter1=counter1+1; end
    if and(abs(err2(1))<=TR , abs(err2(1))>=TA);counter2=counter2+1;Q =
eye(2*n)*0.001; end
    if abs(err2(1))<TA; Q = eye(2*n)*0; counter3=counter3+1; end

    L = A*P*C'*nor;                  % calculate the Kalman Gain

    xhat(:,1+1) = xhat(:,1) + L*err1(1);

    P = A*P*A' + Q - L*C*P*A';      % update the riccatti equation

end
%*****
%*****

mse = (err1*err1')^2/length(t)      % overall mean square error
counter1
counter2
counter3

%***** Plotting the results

figure(1)
plot(t,yhat1,t,y,'g')
title('Actual and estimated outputs')
xlabel('Time Steps')
print nasal.ps

figure(2)
plot(xhat')
title('ARMA Coefficients')
xlabel('Time Steps')
print nasa2.ps

figure(3)
plot(err1)
title('Residuals')
xlabel('Time Steps')
print nasa3.ps

```

7.11 DF.m

```

function [thm,yhat,mse,p,phi,psi] = df(z,nn,adm,adg,th0,p0,phi,psi)
%Directional Forgetting algorithm
% [THM,YHAT,mse] =df(z,NN,adm,adg)
%

```



```

% Z: The output-input data z=[y' u] (single input only!)
% NN : NN=[na nb nc nk], The orders and delay of an ARMAX
%      input-output model (see also ARMAX)
%
% adm: Adaptation mechanism; 'df' directional forgetting
% THM: The resulting estimates. Row k contains the estimates "in alpha-
%      betic order" corresponding to data up to time k (row k in Z)
% YHAT: The predicted values of the output. Row k corresponds to time k.
% TYPICAL: [THM,YHAT,mse] =df(z,[2 2 0 1],'df',0.95)
if nargin < 4
    disp('Usage: MODEL_PARS = df(DATA,ORDERS,ADM,ADG)')
return
end

y=z(:,1);

adm=lower(adm(1:2));
if ~(adm=='df')
    error('The argument ADM should be ''df''')
end

[nz,ns]=size(z);[ordnr,ordnc]=size(nn);
t=1:nz;

if ns>2,error('This routine is for single input only.'),end

if ns==1, if ordnc~=2,error('For a time series nn should be [na nc]!'),end
else if ordnc~=4, error('the argument nn should be [na nb nc
nk]!'),end,end
if ns==1,na=nn(1);nb=0;nc=nn(2);nk=1;
else na=nn(1);nb=nn(2);nc=nn(3);nk=nn(4);nu=1;
end

if nk<1,error('Sorry, this routine requires nk>0; Shift input sequence if
necessary!'),end

d=na+nb+nc;

if ns==1,nb=0;end
if nb==0,nk=1;end
nam=max([na,nc]);
nbm=max([nb+nk-1,nc]);
tic=na+nb+1:na+nb+nc;
ia=1:na;iac=1:nc;
ib=nam+nk:nam+nb+nk-1;
ibc=nam+1:nam+nc;
ic=nam+nbm+1:nam+nbm+nc;

iia=1:nam-1;
iib=nam+1:nam+nbm-1;
iic=nam+nbm+1:nam+nbm+nc-1;
dm=nam+nbm+nc;
if nb==0,iib=[];end
ii=[iia iib iic];
i=[ia ib ic];

if nargin<8, psi=ones(dm,1);end

```

```

if nargin<7, phi=ones(dm,1);end
if nargin<6, p0=10000*eye(d);end
if nargin<5, th0=eps*ones(d,1);end
if isempty(psi),psi=ones(dm,1);end
if isempty(phi),phi=zeros(dm,1);end
if isempty(p0),p0=10000*eye(d);end
if isempty(th0),th0=eps*ones(d,1);end

R=ones(d);

if length(th0)~=d, error('The length of th0 must equal the number of
estimated parameters!'),end
[th0nr,th0nc]=size(th0);if th0nr<th0nc, th0=th0';end
p=p0;th=th0;

R1=zeros(d,d);lam=adg;

for kcou=1:nz

    yh=phi(i)'*th;
    epsi=z(kcou,1)-yh;

    % DF equations
    if epsi>=0.1 M=(1-lam)*(R*psi(i)*psi(i)')/(psi(i)'*R*psi(i));else
M=0;end
    R=(eye(d)-M)*R+psi(i)*psi(i)';
    if epsi>=0.1 p_dash=p+(1-
lam)/lam*(psi(i)*psi(i)')/(psi(i)'*R*psi(i));else p_dash=p;end
    p=p_dash-(p_dash*psi(i)*psi(i)'*p_dash)/(1+psi(i)'*p_dash*psi(i));
    K=p*psi(i);
    th=th+K*epsi;

    if nc>0,c=fstab([1;th(tic)])';else c=1;end
    th(tic)=c(2:nc+1);

    epsilon=z(kcou,1)-phi(i)'*th;
    if nb>0,zb=[z(kcou,2),-psi(1bc)'];else zb=[];end
    ztil=[[z(kcou,1),psi(1ac)'];zb;[epsilon,-psi(1c)']]*c;

    phi(ii+1)=phi(ii);psi(ii+1)=psi(ii);
    if na>0,phi(1)=-z(kcou,1);psi(1)=-ztil(1);end
    if nb>0,phi(nam+1)=z(kcou,2);psi(nam+1)=ztil(2);end
    if nb==0,zc=ztil(2);else zc=ztil(3);end
    if nc>0,phi(nam+nbm+1)=epsilon;psi(nam+nbm+1)=zc;end

    thm(kcou,:)=th';
    yhat(kcou)=yh;
    err(kcou)=yhat(kcou)-z(kcou,1);
end

mse = (err*err')^2/nz; % overall mean square error

figure(1)
plot(t,yhat,t,y','g')
title('Actual and estimated outputs')

```

```
xlabel('Time Steps')
```

```
figure(2)  
plot(thm)  
title('ARMA Coefficients')  
xlabel('Time Steps')
```

```
figure(3)  
plot(err)  
title('Residuals')  
xlabel('Time Steps')
```

References

- [1] L.Cao and H. Schwartz, "A Novel Recursive Algorithm for Directional Forgetting", *Proceedings of the American Control Conference*, San Diego, California, June 1999, pp. 1334-1338
- [2] F.Chawdhury, "Kalman Filter with Hypothesis testing: a tool for estimating uncertain parameters", *Circuits Systems Signal Processing*, Vol. 15, No3, 1996, pp. 291-311
- [3] T.Fortescue and L.Kershenbaum and B.Ydstie, "Implementation of self-tuning regulators with variable forgetting factor", *Automatica*, Vol. 17, No. 6, 1981, pp. 831-835
- [4] L.Ljung, "System Identification, Theory for the User", Prentice Hall, Second Edition, 1999
- [5] F. Chowdhury, "Input-output Modeling Of Nonlinear Systems With Time-varying Linear Models", *IEEE Transactions on Automatic Control*, Vol. 45 No. 7, July 2000, pp.1355-8
- [6] P. Andersson, "Adaptive forgetting in recursive identification through multiple models", *Int. J. Control*, 1985, Vol. 42, No. 5, pp.1175-1193
- [7] R. Kulhavy and M. Karny, "Tracking of Slowly Varying Parameters By Directional Forgetting", *IFAC Triennial World Congress*, Budapest, Hungary, 1984
- [8] J. Parkum and N. Poulsen and J. Holst, "Recursive Forgetting Algorithm", *Int. J. Control*, 1992, Vol. 55, No. 1, pp.109-128
- [9] Greg Welch and Gary Bishop, "An Introduction To Kalman Filter", published on www.cs.unc.edu/~welch/kalman/kalman.html
- [10] J.K.Astrom and G.C.Goodwin and P.R.Kumar, "Adaptive Control, Filtering, and Signal Processing", Springer-Verlag, 1995