



National Library  
of Canada

Acquisitions and  
Bibliographic Services Branch

395 Wellington Street  
Ottawa, Ontario  
K1A 0N4

Bibliothèque nationale  
du Canada

Direction des acquisitions et  
des services bibliographiques

395, rue Wellington  
Ottawa (Ontario)  
K1A 0N4

Your file / Votre référence :

Our file / Notre référence :

## NOTICE

**The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.**

**If pages are missing, contact the university which granted the degree.**

**Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.**

**Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.**

## AVIS

**La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.**

**S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.**

**La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.**

**La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.**

**UNIVERSITY OF ALBERTA**

**AUGMENTED UD IDENTIFICATION  
FOR PROCESS CONTROL**

**BY**



**SHAOHUA NIU**

**A thesis submitted to the Faculty of Graduate Studies and Research in  
partial fulfillment of the requirements for the degree of**

**DOCTOR OF PHILOSOPHY**

**IN**

**PROCESS CONTROL**

**DEPARTMENT OF CHEMICAL ENGINEERING**

**EDMONTON, ALBERTA**

**SPRING, 1994**



National Library  
of Canada

Bibliothèque nationale  
du Canada

Acquisitions and  
Bibliographic Services Branch

Direction des acquisitions et  
des services bibliographiques

395 Wellington Street  
Ottawa, Ontario  
K1A 0N4

395, rue Wellington  
Ottawa (Ontario)  
K1A 0N4

*Author's file - Auteur, résumés, etc.*

*Author's file - Auteur, résumés, etc.*

**The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.**

**L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.**

**The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.**

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

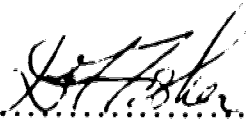
ISBN 0-612-11312-4

**Canada**

UNIVERSITY OF ALBERTA

**FACULTY OF GRADUATE STUDIES AND RESEARCH**

The undersigned certify that they have read, and recommended to the FACULTY OF GRADUATE STUDIES AND RESEARCH for acceptance, a thesis entitled THE AUGMENTED UD IDENTIFICATION FOR PROCESS CONTROL submitted by SHAOHUA NIU in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY in PROCESS CONTROL



.....  
Dr. D. G. Fisher (Supervisor)



.....  
Dr. S. L. Shah



.....  
Dr. M. Rao



.....  
Dr. R. K. Wood



.....  
Dr. V. G. Gourishankar



.....  
Dr. L. Ljung (External Examiner)

Dated Jan..18., 1994



UNIVERSITY OF ALBERTA

**RELEASE FORM**

NAME OF AUTHOR:

**SHAOHUA NIU**

TITLE OF THESIS:

**AUGMENTED UD IDENTIFICATION**

**FOR PROCESS CONTROL**

DEGREE:

**DOCTOR OF PHILOSOPHY**

YEAR THIS DEGREE GRANTED:

**1994**

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

..........

.....#3-10824-85 AVE.....

.....EDMONTON, ALBERTA.....

.....CANADA, T6E 2L1.....

Dated

*Jan 18/94*

D. Grant Fisher, S.L. Shah and P. Banerjee  
Dept of Chemical Engineering  
University of Alberta  
Edmonton, AB T6G 2G6  
January 18, 1994

**To Whom It May Concern**

Dear Sir/Madam,


We hereby certify that Mr. Shaohua Niu has our permission to include in his thesis the material contained in the following papers that we have co-authored. We understand that the acceptance of Mr. Niu's thesis includes permission for microfilming of the thesis by the National Library of Canada and for the University of Alberta Library to reproduce single copies of his thesis and to lend or sell such copies for private, scholarly or scientific research purpose only.

1. Shaohua Niu, D. Xiao and D. Grant Fisher, "A Unified Form of the Instrumental Variable Identification Algorithm", *International Journal of Adaptive Control and Signal Processing*, Vol. 7, No. 4, 261-277, 1993.
2. Shaohua Niu, D. Grant Fisher and D. Xiao, "An Augmented UD Identification Algorithm", *International Journal of Control*, Vol. 56, No. 1, 93-211, 1992.
3. Shaohua Niu, D. Xiao and D. Grant Fisher, "A New Algorithm for Simultaneous Identification of Model Order and Parameters", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 38, No. 5, 884-886, 1990.
4. Shaohua Niu, D. Grant Fisher, "Simultaneous Structure Identification and Parameter Estimation of Multivariable Systems", accepted by *International Journal of Control*.
5. Shaohua Niu and D. Grant Fisher, "Multivariable System Identification Using Augmented UD Factorization", *Proceedings of the 1991 American Control Conference*, 699-703, Boston, 1991.
6. Shaohua Niu, D. Grant Fisher, "Information Forgetting in Recursive Identification", in *Proceedings of the 1993 American Control Conference*, 796-800, San Francisco, 1993.
7. Shaohua Niu, D. Grant Fisher, "On-line Identification of Noise Variance and Signal-to-Noise Ratio", accepted by *IEEE Transactions on Signal Processing*.
8. P. Banerjee., S. L. Shah, Shaohua Niu and D. Grant Fisher, "Identification of the Process Dynamics for the Shell Benchmark Problem", submitted to *Shell Identification Workshop*, 1993.

We trust that this letter provides the permission necessary for acceptance of Mr. Niu's thesis.

Sincerely Yours

  
D. Grant Fisher,

  
S. L. Shah,

  
P. Banerjee

Prof. Deyun Xiao  
Dept. of Automation  
Tsinghua University  
Beijing 100084, P.R.China  
January 1, 1994

**To Whom It May Concern**

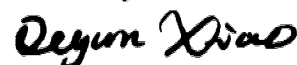
Dear Sir/Madam,

I hereby certify that Mr. Shaohua Niu has my permission to include in his thesis the material contained in the following papers that I have co-authored. I understand that the acceptance of Mr. Niu's thesis includes permission for microfilming of the thesis by the National Library of Canada and for the University of Alberta Library to reproduce single copies of his thesis and to lend or sell such copies for private, scholarly or scientific research purpose only.

1. Shaohua Niu, D. Xiao and D. Grant Fisher, "A Factored Form of the Instrumental Variable Identification Algorithm", *International Journal of Adaptive Control and Signal Processing*, Vol.7, No.4, 261-273, 1993.
2. Shaohua Niu, D. Grant Fisher and D. Xiao, "An Augmented UD Identification Algorithm", *International Journal of Control*, Vol. 56, No. 1, 193-211, 1992.
3. Shaohua Niu, D. Xiao and D. Grant Fisher, "A Recursive Algorithm for Simultaneous Identification of Model Order and Parameters", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.38, No. 5, 884-886, 1990.
4. D. Xiao and Shaohua Niu, "A Recursive Algorithm for Simultaneous Identification of Model Order and Parameters", *Chinese Journal of Automation*, Vol. 1, No. 3, 257-264, Allerton Press, New York, 1989.
5. Shaohua Niu and D. Xiao, "The UD factorization Form of the IV algorithm", *Proceedings of the 8th IFAC Symposium on Identification and System Parameter Identification*, 1060-1063, Beijing, 1988.

I trust that this letter provides the permission necessary for acceptance of Mr. Niu's thesis.

Sincerely Yours



Deyun Xiao

**EVERYTHING SHOULD BE AS SIMPLE  
AS POSSIBLE — BUT NOT SIMPLER**

**ALBERT EINSTEIN**

**TO MY SWEETHEART  
WHO HAS SHARED MY UPS AND DOWNS**

**TO MY PARENTS  
WHO ARE SO FAR AWAY AND ALSO SO CLOSE**

# ABSTRACT

Process identification is a very important branch of automatic control because it generates the dynamic models required for successful prediction, control, real-time optimization, fault detection etc. Successful identification and control provide economic gains, dependable quality, optimal production, improved safety and reduced environmental compact, etc, in industrial applications.

Currently the most widely used identification methods are the Recursive Least-Squares (RLS) type algorithms which include the basic RLS algorithm and its variants. However, because the covariance matrix in RLS type algorithms is updated in a potentially unstable manner, RLS algorithms may have serious numerical problems in real industrial applications, especially when implemented on digital computers with finite data length, and/or with large multivariable applications which are increasingly common in industry.

This thesis focuses on a new approach to process identification called Augmented UD Identification (AUDI). By properly rearranging and augmenting the data (regressor) and parameter vectors normally used in RLS identification algorithms, a structure is developed which shows that all the information required to determine the model parameters and cost functions for all models from order 1 to a user-specified value  $n$  are explicitly contained in the Augmented Covariance Matrix (ACM) and can be easily extracted by applying a  $UDU^T$  type decomposition. The Augmented UD Identification approach presents the least-squares estimator in a form which is easier to interpret and leads to a simpler, more compact and convenient structure that provides much more information than the least-squares estimator. The UD factorization inherent in the AUDI approaches provides the algorithm with superior numerical performance. In addition, the AUDI algorithms provide some useful features that the RLS type algorithms do not have, such as simultaneous identification of model order and parameters with no extra computation, on-line estimation of the process noise variance and signal-to-noise ratio.

A family of AUDI identification algorithms are developed which are analogous to the RLS type algorithms in principle but superior to RLS in almost every respect. The Recursive Instrumental Variable version (AUDI-IV) and the recursive Extended Least-squares version (AUDI-ELS) of the AUDI family of algorithms are also presented in this thesis. Multivariable control is a major concern in industry, but multivariable identification is a difficult step. The AUDI structure has been extended to the MIMO case and a MIMO version of the AUDI algorithm is presented.

Careful analysis of the AUDI structure leads to new insights into the information forgetting principle and some easy and convenient guidelines are proposed to facilitate the design and interpretation of information forgetting mechanisms for recursive identification.

The AUDI algorithms have been programmed into a software package and tested with industrial data sets as well as a number of specially constructed simulation examples. The results produced by the AUDI software were compared with those by commercial products such as Dynamic Matrix Identification (DMI) and proved that the AUDI algorithms are very promising for industrial applications. The AUDI approach is recommended as a replacement for the RLS type algorithms in real applications.

## **ACKNOWLEDGEMENT**

**I wish to express my sincere gratitude to my supervisor Dr. D. Grant Fisher for his enthusiastic supervision, help and encouragement during the course of my PhD program.**

**I would also like to thank Dr. S. L. Shah for his many valuable suggestions and help with my research work.**

**Thanks are also due to all my student colleagues in the process control group for stimulating discussions and to the staff of the DACS center, especially Mr. Bob Barton, for their help with the computing facility.**

**Financial support from the Natural Science and Engineering Research Council of Canada (NSERC) in the form of the postgraduate scholarship, from the University of Alberta in the form of the Water Johns Graduate Scholarship and the Andrew Steward Memorial Graduate Prize, and from the Department of Chemical Engineering in the form of the Captain Thomas Farrell Greenhalgh Graduate Memorial Scholarship are also gratefully acknowledged.**

# Contents

Abstract .....	
Contents .....	
List of Figures .....	
List of Tables .....	
List of Symbols and Abbreviations .....	
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Introduction to Recursive Identification .....	1
1.2 Contributions of This Thesis .....	4
1.2.1 Contributions to Identification Theory .....	5
1.2.2 Contributions to Industrial Application .....	6
1.3 Organization of the Thesis .....	6
Bibliography .....	7
<b>2 THE AUGMENTED UD IDENTIFICATION METHOD</b>	<b>8</b>
2.1 Introduction .....	8
2.2 Process Models .....	9
2.3 Least-squares Estimator .....	10
2.4 The Augmented UD Identification Structure .....	12
2.5 Decomposition .....	14
2.6 Model Order Determination .....	16
2.7 Simulation Examples .....	18
2.8 Conclusions .....	21
Bibliography .....	23
<b>3 THE AUDI-LS ALGORITHM</b>	<b>24</b>
3.1 Introduction .....	24
3.2 Recursive Least-Squares Algorithms .....	25
3.2.1 The Basic RLS Algorithm .....	26
3.2.2 RLS with UD Factorization .....	27
3.3 The Augmented UD Identification Algorithm .....	28
3.4 The Multiple Model Structure .....	30
3.5 Comparison of AUDI and RLS .....	32
3.6 Simulation Examples .....	33
3.7 Conclusions .....	33
Bibliography .....	39



<b>4 THE AUDI-ELS ALGORITHM</b>	<b>40</b>
4.1 Introduction	40
4.2 The Pseudo-linear Regression Model	41
4.3 The Extended Least-Squares Estimator	42
4.4 The AUDI Form of the ELS Algorithm	43
4.4.1 The AUDI structure for ELS	43
4.4.2 The Recursive AUDI Algorithm for ELS	44
4.4.3 The simplified AUDI-ELS algorithm	45
4.5 Simulation Examples	47
4.6 Conclusions	50
Bibliography	53
<b>5 THE AUDI-IV ALGORITHM</b>	<b>54</b>
5.1 Introduction	54
5.2 The Basic Instrumental Variable Algorithm	55
5.3 The AUDI Form of IV Algorithm	57
5.4 Recursive Implementation of AUDI-IV	60
5.4.1 The Recursive AUDI-IV Algorithm	60
5.4.2 Efficient Implementation of AUDI-IV	60
5.5 Simulation Examples	62
5.6 Conclusion	65
Bibliography	67
<b>6 IDENTIFICATION OF MULTIVARIABLE SYSTEMS</b>	<b>68</b>
6.1 Introduction	68
6.2 Model Representations for MIMO Systems	69
6.2.1 State-Space Representation	69
6.2.2 Input-Output Difference Equation Model	70
6.2.3 Transformations	71
6.3 The Multivariable AUDI Algorithm	72
6.3.1 The Augmented UD Identification Structure	72
6.3.2 The Parameter and Loss Function Matrices	74
6.3.3 Recursive AUDI Algorithm	77
6.4 Simulation Example	78
6.5 Conclusion	80
Bibliography	83
<b>7 IDENTIFICATION OF TIME VARYING PROCESS</b>	<b>84</b>
7.1 Introduction	84
7.2 Information Accumulation	85
7.3 Information Forgetting with AUDI	87
7.3.1 Relative Forgetting	88
7.3.2 Absolute Forgetting	90
7.3.3 Matrix Regularization	93
7.4 Improved Variable Forgetting Method	94
7.4.1 The Algorithm	94
7.4.2 On/off Decision for Identification	95
7.4.3 Regularization	96
7.4.4 Simulation Example	96

7.5	Conclusions	97
	Bibliography	100
<b>8</b>	<b>IDENTIFICATION OF NOISE CHARACTERISTICS</b>	<b>101</b>
8.1	Introduction	101
8.2	Estimation of Noise Variance	102
8.3	Estimation of Signal-to-Noise Ratio	105
8.3.1	The Augmented UD Identification Algorithm	108
8.3.2	Estimation of Signal-to-Noise Ratio with AUDI	108
8.4	Conclusion	110
	Bibliography	111
<b>9</b>	<b>PRACTICAL CONSIDERATIONS</b>	<b>112</b>
9.1	Practical Considerations	112
9.1.1	Model Set	114
9.1.2	Initial Conditions for Identification	116
9.1.3	Model Conversion	117
9.2	Identification of a Distillation Column	119
9.2.1	Preliminary Analysis	120
9.2.2	Identification of Parameters	121
9.2.3	Discussion	125
9.3	Identification of a Real Industrial Process	127
9.4	Conclusions	132
	Bibliography	133
<b>10</b>	<b>CONCLUSIONS AND RECOMMENDATIONS</b>	<b>134</b>
10.1	Conclusions	134
10.2	Recommendations	135
	Bibliography	136
<b>A</b>	<b>DERIVATION OF THE AUDI STRUCTURE</b>	<b>137</b>
<b>B</b>	<b>DERIVATION OF THE AUDI-IV ALGORITHM</b>	<b>141</b>
B.1	Proof of the AUDI-IV structure	141
B.2	The $UDV^T$ Decomposition	142
<b>C</b>	<b>A CHECKLIST OF "PRACTICAL FACTORS"</b>	<b>144</b>
C.1	Project Definition and Planning	144
C.2	Process Analysis and Modeling	145
C.3	Experimental Design	146
C.4	Experimental Plant Tests	147
C.5	Model Validation	148
C.6	Integrating Model into Application	149
C.7	Maintenance and Enhancement	149

# List of Figures

1.1 The Principle of Process Identification . . . . .	1
1.2 The Procedure of Process Identification . . . . .	3
1.3 A Model-Based Adaptive Controller . . . . .	4
2.1 The Process . . . . .	9
2.2 Batch and Recursive Implementation Methods of the AUDI Method . . . . .	17
2.3 The Loss Functions versus Model Orders . . . . .	20
3.1 The order recursion of the AUDI algorithm . . . . .	31
3.2 The parameter trajectory using recursive AUDI algorithm . . . . .	36
3.3 The loss functions using recursive AUDI algorithm . . . . .	36
3.4 The gain trajectory using recursive AUDI algorithm . . . . .	37
4.1 Parameter Trajectories Using AUDI-LS (Biased Results) . . . . .	48
4.2 Parameter Trajectories Using AUDI-ELS (Unbiased Results) . . . . .	49
4.3 Loss Functions versus Model Orders Using AUDI-ELS . . . . .	50
4.4 Trajectories of Estimated Variance of Process Noise . . . . .	52
4.5 Parameter Trajectories using the Simplified AUDI-ELS . . . . .	52
5.1 Construction of Instrumental Variables . . . . .	56
5.2 Parameter trajectories using different IV algorithms . . . . .	63
6.1 Partitioning of the Parameter Matrix $U(t)$ . . . . .	75
6.2 Partitioning of the Loss Function Matrix $\mathcal{D}$ . . . . .	76
7.1 Input and Output data . . . . .	97
7.2 Parameter Estimates with AUDI (solid) and EF (dashed) for $\lambda=0.90$ . . . . .	98
7.3 Parameter Estimates with AUDI (solid) and EF (dashed) for $\lambda=0.98$ . . . . .	98
8.1 Trajectories of Loss Functions . . . . .	104
8.2 Trajectories of Estimated Standard Deviation $\sigma$ of Noise . . . . .	104
8.3 Trajectories of Estimated Signal-to-Noise Ratios . . . . .	107
9.1 Diagram of A Distillation Column . . . . .	119
9.2 Input/output Data of the Distillation Column . . . . .	121
9.3 Step Responses of the Distillation Column . . . . .	122
9.4 Predicted Output, Noise-free Output and Measured Output . . . . .	122
9.5 Step Responses of the Distillation Column (Differenced Data) . . . . .	124
9.6 Predicted Output, Noise-free Output and Measured Output (differenced data) . . . . .	124

<b>9.7 Step Responses from a Commercial Identification Package (original data)</b>	<b>126</b>
<b>9.8 Predicted Output, Noise-free Output and Measured Output (from the commercial package)</b>	<b>126</b>
<b>9.9 Step Responses from a Commercial Identification Package (differenced data)</b>	<b>127</b>
<b>9.10 Input/output data from a Real Plant (Raw Data)</b>	<b>129</b>
<b>9.11 Input/output data from a Real Plant (after pretreatment)</b>	<b>130</b>
<b>9.12 Step Response produced by ALDI (dashed lines) and DMI (solid lines)</b>	<b>131</b>

# List of Tables

<b>2.1 The LU and <math>UDU^T</math> decompositions</b>	<b>15</b>
<b>2.2 The Augmented Information Matrix</b>	<b>19</b>
<b>2.3 Parameter Matrix Obtained by <math>LDL^T</math> Decomposition</b>	<b>19</b>
<b>2.4 Loss Function Matrix Obtained by <math>LDL^T</math> Decomposition</b>	<b>19</b>
<b>3.1 The Recursive Least-Squares Algorithm</b>	<b>26</b>
<b>3.2 Bierman's UD Factorization Algorithm</b>	<b>27</b>
<b>3.3 The Recursive AUDI Algorithm</b>	<b>28</b>
<b>3.4 The compact form of the recursive AUDI algorithm</b>	<b>30</b>
<b>3.5 Comparison of AUDI and RLS</b>	<b>34</b>
<b>3.6 Parameter matrix <math>U(t)</math></b>	<b>35</b>
<b>3.7 Loss function matrix <math>D(t)</math></b>	<b>35</b>
<b>4.1 The recursive extended least-squares algorithm</b>	<b>42</b>
<b>4.2 The Recursive AUDI Algorithm</b>	<b>45</b>
<b>4.3 multiple Model Structure of AUDI-ELS</b>	<b>46</b>
<b>4.4 Noise Models in AUDI-ELS</b>	<b>46</b>
<b>4.5 The Simplified AUDI-ELS Algorithm</b>	<b>47</b>
<b>4.6 Biased Parameters Obtained Using AUDI-LS</b>	<b>48</b>
<b>4.7 Converged Parameter Matrix By AUDI-ELS</b>	<b>51</b>
<b>4.8 Converged Parameter Matrix By Simplified AUDI-ELS</b>	<b>51</b>
<b>5.1 The Recursive AUDI-IV Algorithm</b>	<b>61</b>
<b>5.2 The Efficient AUDI-IV Algorithm</b>	<b>62</b>
<b>5.3 Parameter Matrix in the Recursive AUDI-IV Example</b>	<b>64</b>
<b>5.4 Parameter Matrix in the Efficient AUDI-IV Example</b>	<b>64</b>
<b>5.5 Auxiliary Parameter Matrix in the Recursive AUDI-IV Example</b>	<b>64</b>
<b>6.1 The Multivariable Recursive AUDI Algorithm</b>	<b>77</b>
<b>6.2 Loss Function Matrix of the <math>2 \times 2</math> Multivariable System (Compare Figure 2)</b>	<b>79</b>
<b>6.3 Loss Functions of all the Subsystems</b>	<b>79</b>
<b>6.4 The Parameter Matrix of the Multivariable System (Compare Figure 1)</b>	<b>81</b>
<b>6.5 Rearranged Parameter Matrix</b>	<b>81</b>
<b>7.1 The Recursive AUDI Algorithm</b>	<b>89</b>
<b>8.1 Loss Functions and Estimated Noise Variance (with <math>\lambda=1.0</math>)</b>	<b>106</b>
<b>8.2 Loss Functions and Estimated Noise Variance (with <math>\lambda=0.99</math>)</b>	<b>106</b>
<b>8.3 Estimated SNR (with <math>\lambda=1.0</math>)</b>	<b>107</b>

<b>8.4</b>	<b>Estimated SNR (with <math>\lambda=0.98</math>)</b>	<b>107</b>
<b>8.5</b>	<b>Loss functions and squared sum of outputs</b>	<b>108</b>
<b>9.1</b>	<b>Dependence of Identification on Application Purpose</b>	<b>113</b>
<b>9.2</b>	<b>Loss Functions Using Original Data</b>	<b>123</b>
<b>9.3</b>	<b>Loss Functions Using Differenced Data</b>	<b>123</b>
<b>9.4</b>	<b>Variance of Prediction Errors</b>	<b>125</b>

# List of Symbols and Abbreviations

## 1. Symbols

$C(t)$	augmented covariance matrix (ACM)
$S(t)$	augmented information matrix (AIM)
$P(t)$	covariance matrix
$R(t)$	information matrix
$K(t)$	Kalman gain vector
$U$	parameter matrix
$D$	loss function matrix
$V$	auxiliary parameter matrix
$\theta_0$	the true parameter vector of the process
$\hat{\theta}(t)$	parameter estimate for process model at time $t$
$\hat{\theta}(t)$	parameter estimate for auxiliary model at time $t$
$\hat{\alpha}(t)$	parameter estimate for backward model
$h(t)$	data vector
$\varphi(t)$	augmented data vector
$\zeta(t)$	instrumental vector
$\eta(t)$	augmented instrumental vector
$z(t)$	process output (corrupted by noise)
$y(t)$	process output (noise free)
$u(t)$	process input
$v(t)$	zero-mean white noise
$e(t)$	zero-mean colored noise
$\tilde{z}(t)$	innovation sequence for process model
$\tilde{z}(t)$	innovation sequence for auxiliary model
$\sigma_v^2$	noise variance
$\epsilon(t)$	residual of process model
$\epsilon(t)$	residual of auxiliary model
$J(t)$	loss function of process model
$L(t)$	loss function of backward model
$t$	time step
$d$	dimension of the information accumulation matrix
$n$	model order

## 2. Abbreviations

ACM	Augmented Covariance Matrix
AIC	Akaike's Information Criterion
AIM	Augmented Information Matrix
ADM	Augmented Data Matrix
ARMA	AutoRegressive and Moving Average
ARIMA	AutoRegressive and Integral Moving Average
AUDI	Augmented UD Identification
DMC	Dynamic Matrix Control
ELS	Extended Least-Squares
FIR	Finite Impulse Response
GMV	Generalized Minimum Variance Control
GPC	Generalized Predictive Control
IAM	Information Accumulation Matrix
IV	Instrumental Variable
LS	Least-Squares
MA	Moving Average
MIMO	Multi-Input Multi-Output
MISO	Multi-Input Single-Output
MOCCA	Multivariable Optimal Constrained Control Algorithm
PID	Proportional-Integral-Derivative (control)
PRBS	Pseudo-Random Binary Sequence
RBS	Random Binary Sequence
RLS	Recursive Least-Squares
SISO	Single-Input Single-Output



# Chapter 1

## INTRODUCTION

**T**his thesis focuses on the development and implementation of a new family of identification methods called the augmented UD identification (AUDI) methods. The AUDI method is an efficient reformulation of the familiar least squares estimator, but are superior to the existing recursive least-squares (RLS) type methods in almost all aspects, especially for real-time applications. This chapter provides a brief introduction to recursive identification and an overview of this thesis.

### 1.1 Introduction to Recursive Identification

Process identification is the field of modeling dynamic systems from experimental data. More precisely, identification is "the determination on the basis of input/output, of a system within a specified class of systems, to which the system under test is equivalent" (Zadach 1962). The basic concepts of process identification are illustrated by Figure 1.1. In both batch and recursive applications, identification of the model are based on measurements of the process input and output variables.

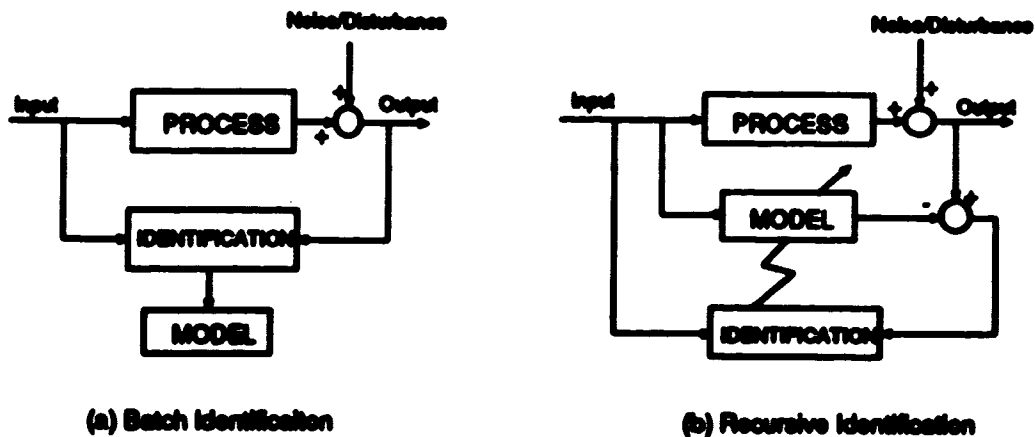


Figure 1.1: The Principle of Process Identification

An accurate and reliable dynamic model is necessary for applications such as model based control, adaptive filtering, prediction and fault detection. In the field of process control, more and more industrial control systems are changing from the "traditional"

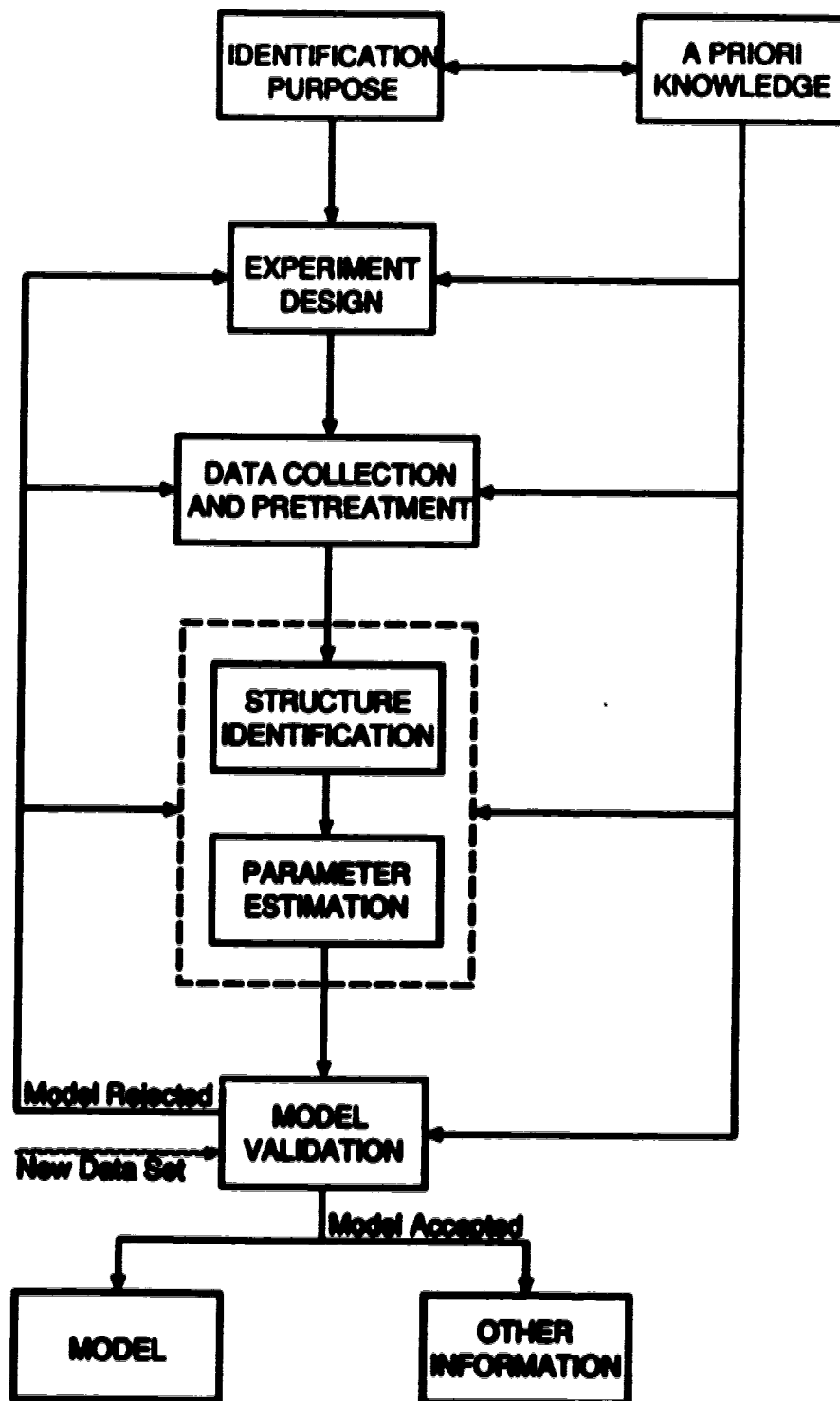
PID feedback loops to more advanced model-based controllers such as Dynamic Matrix Control (DMC) (Cutler & Ramaker 1980) and Generalized Predictive Control (Clarke & Mohtadi 1987). However, in most cases, developing the process model is the most important and also most difficult and time-consuming part in developing a model-based controller. Therefore this thesis addresses an important current problem in industrial process control.

The typical procedure for process identification is illustrated in Figure 1.2, where it is seen that process identification usually consists of the following steps (Islermann 1980, Fang & Xiao 1988, Söderström & Stoica 1989):

1. Experiment design. This may include the determination of input/output variables, input excitation signal(s), sampling time, duration of identification, open-loop versus closed-loop identification, on-line or off-line identification, etc.
2. Data collection and pretreatment. The main task of data pretreatment is to remove any drift or bias (D.C. component) and filter out undesirable components such as high frequency noises. Proper pretreatment of input/output data can significantly improve the identification results.
3. Structure identification. A suitable model structure must be assumed before the structural parameters are identified. The structural parameter for a single-input single-output difference equation model is just the model order, while for a multi-input multi-output difference equation model, the structural parameters are a set of invariants, called the structural indices.
4. Parameter estimation. After the model structure is determined, the model parameters can then be estimated. The most widely used method for parameter estimation has been the recursive least-squares (RLS) method. However, as discussed later, the least-squares method has serious drawbacks in real applications. The ALDI family of methods developed in this thesis are recommended as replacements for the RLS type methods for real applications.
5. Model validation. In practice, the identified model is always only an approximation to the real process. Therefore the identified model must be validated to see whether it is an adequate representation of the actual process. If the model is not acceptable, then the identification procedure must be repeated.

The classical approach to identification is the so-called batch or off-line method, in which all the input/output data are collected first and then used simultaneously to find the parameter estimates. However, in process application, the process model is usually required for calculations or decisions that must be made on-line. Obviously, the model should always be up-to-date, and thus it is necessary to update the process model every time a new set of input/output is obtained, which leads to the concept of on-line recursive identification. Recursive identification has many advantages over batch methods, the most important of which are that recursive identification methods have lower computational and smaller memory requirements. One disadvantage of most recursive methods is that the decision regarding model structure must be made *a priori*, which is usually not possible. However, the ALDI method proposed in this thesis solves (at least partially) this problem.

Recursive identification is the cornerstone of almost all model based adaptive controller where the control action is based on the most recent model. A typical adaptive



**Figure 1.2: The Procedure of Process Identification**

control configuration is shown in Figure 1.3, which makes it obvious that the parameter estimates are a prerequisite for the control action calculation. The success of any

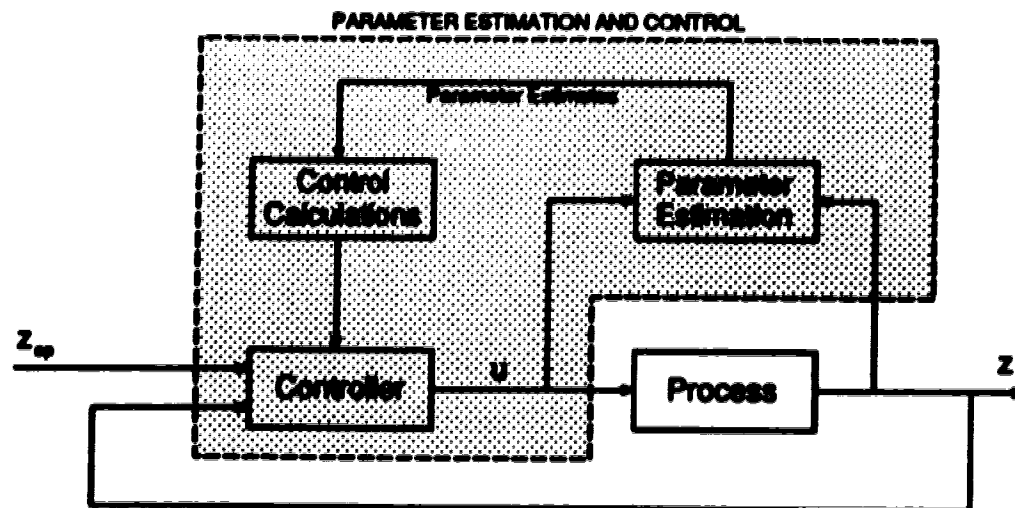


Figure 1.3: A Model-Based Adaptive Controller

adaptive controller in practice is highly dependent on the success of the recursive identification part. At present the complexities and uncertainties associated with recursive identification prevent adaptive control from being widely used in industry. Numerous publications on improving the robustness of the control law are simply aimed at reducing the effects of model inaccuracies (Clarke & Mohtadi 1987). In other words, if the accuracy of the process model can be improved, much of the work on controller design will be simplified.

Currently the most popular identification methods are the RLS method and its many variants, such as extended least-squares (ELS) method, the instrumental variable (IV) method, etc. However, a common problem with all these LS type methods is that they may have serious numerical problems when implemented on digital computers with finite data length or applied to large-dimension problems such as multivariable processes, since the LS type methods all involve (implicitly or explicitly) matrix inversion (Shah & Chueti 1991). The inconvenience of handling problems such as estimator turnoff, covariance blowup, parameter drift, variable excitation, the existence of output feedback, also add difficulties to RLS type methods in real applications. Because of the importance of process identification and the many practical problems associated with industrial applications a great deal of effort has been devoted to improving the least-squares type methods.

## 1.2 Contributions of This Thesis

The main contribution of this thesis is a family of new identification methods called the ALDI methods which are superior to the widely used least-squares methods in almost all respects. The ALDI methods open a new area in process identification and provide a set of versatile, reliable and efficient identification methods for practical applications.

### 1.2.1 Contributions to Identification Theory

The AUDI methods are based on least-squares principles but have a different formulation that leads to a special and interesting structure called the AUDI structure. The AUDI structure explicitly contains all the parameter estimates and corresponding loss functions of all models from order 1 to a user-specified maximum order  $n$ . The AUDI methods based on the AUDI structure possess the following features which are superior to their RLS counterparts:

1. Simultaneous identification of model parameters and loss functions for multiple models of different orders from 1 to a user-specified maximum possible order  $n$ , with computational effort equivalent to an  $n$ th-order RLS method.
2. A compact, convenient structure for easy interpretation and implementation. The implementation of the AUDI approach consists of the following straightforward steps: collecting data from process, constructing and decomposing the augmented covariance matrix (ACM), then all the required parameter estimates and the corresponding loss functions of all models from 1 to a maximum  $n$  are obtained from the parameter matrix  $U$  and loss function matrix  $D$ .
3. Excellent numerical performance. The AUDI structure, which is inherently a UD factorization, is numerically very stable. It overcomes the numerical inferiority of the RLS type methods, and thus can provide consistent estimates of high accuracy under most circumstance.
4. A convenient basis for practical integrated extensions. The information provided by the AUDI structure provides the basis for extensions such as an on/off criterion, estimation of signal-to-noise ratio, information forgetting. This is represented by the box other information in Figure 1.2.

In summary, *The AUDI type methods are superior to RLS type methods in almost every respect, and are thus recommended for use in place of RLS type methods for all applications.* The AUDI type methods are expected to take over the monopolistic position that has been held by RLS type methods for so many decades.

The AUDI family of methods includes AUDI-LS, AUDI-ELS, AUDI-IV and the MIMO-AUDI methods. AUDI-LS is the basic version of the AUDI method which is used for identification of processes with white noise. It is also the method that is used for interpretation and analysis of the AUDI family methods. The AUDI-ELS and AUDI-IV methods are the RLS counterparts of the extended least-squares method and the instrumental variable method and hence can handle colored process noise. The MIMO-AUDI method is an extension of the AUDI method to multivariable process. The AUDI structure makes the very complicated problem of identifying the multivariable structural indices and parameters much easier. The extension of AUDI methods to other forms is also straightforward.

Adaptive methods for tracking time-varying systems are usually equipped with a suitable information forgetting mechanism. Information forgetting has been a very active but also very difficult field. New insights into the principles of information forgetting for time varying process are gained from the AUDI methods. Simpler and more convenient guidelines for information forgetting are provided and a generalized information forgetting scheme is given in this thesis.

Noise statistics of the process is very important for many applications such as Kalman filtering and minimum variance control. Simultaneous estimation of noise variance,

signal-to-noise ratio, parameters and loss functions can be easily achieved using the AUDI methods.

### **1.2.2 Contributions to Industrial Application**

The AUDI type methods are practically oriented and have great potential for industrial applications. The AUDI methods have been programmed into a prototype software package designed for real industrial use. The package was evaluated with real industrial data, as well as some specially designed benchmark test data. The results were very encouraging and in most cases equalled or exceeded those from other identification packages including the commercial DMI package.

Some "practical factors" that must be included in industrial applications, such as input data design, input/output data filtering, "bad" data handling, are also investigated and some guidelines are provided.

## **1.3 Organization of the Thesis**

This thesis is organized as follows. Chapter 2 is devoted to the derivation and discussion of the AUDI structure, which is the basis of the whole thesis. Model order determination and numerical properties are also briefly discussed. In Chapter 3, the basic least-squares form of the AUDI method (AUDI-LS) is developed and compared with the recursive least-squares.

The extended least-squares version (AUDI-ELS) and the instrumental variable version (AUDI-IV) of the AUDI method are developed in Chapters 4 and 5 respectively. Comparison with AUDI-LS are provided in each chapter. In Chapter 6, the AUDI method for identifying multivariable process is developed based on the model representation of Guidorzi (1975).

In Chapter 7, the special structure of the AUDI method is used to investigate information forgetting mechanisms for time-varying processes. Guidelines for designing new information forgetting schemes are provided. Chapter 8 extends the AUDI method for simultaneous estimation of noise characteristics and parameters, and Chapter 9 discusses some practical implementation issues and presents some practical application examples. Finally Chapter 10 presents some overall conclusions and recommendations for future extensions.

# Bibliography

- Clarke, D. W. & Mohtadi, C. (1987), 'Generalized predictive control - part I. the basic algorithm. part II. extensions and interpretations', *Automatica* 23(2), 137 - 160.
- Cutler, C. R. & Ramaker, B. L. (1980), Dynamic matrix control — a computer control algorithm, in 'Proceedings of the 1980 Joint Automatic Control Conference', San Francisco, pp. WP5-B.
- Fang, C. & Xiao, D. (1988), *Process Identification*, Tsinghua University Press, Beijing, China.
- Guidorzi, R. P. (1975), 'Canonical structure in the identification of multivariable systems', *Automatica* 11(4), 361-374.
- Isermann, R. (1980), 'Practical aspects of process identification', *Automatica* 16(5), 575 - 587.
- Shah, S. L. & Chueti, W. R. (1991), 'Recursive least-squares based estimation schemes for self-tuning control', *The Canadian Journal of Chemical Engineering* 69, 89 - 96.
- Söderström, T. & Stoica, P. (1989), *System Identification*, Prentice Hall, Englewood Cliffs, New Jersey.
- Zadach, L. A. (1962), 'From circuit theory to system theory', *Proceedings of the IRE* 50(5), 856 - 865.

## Chapter 2

# THE AUGMENTED UD IDENTIFICATION METHOD

**L**east-squares parameter estimation methods involve matrix inversion and thus are subject to numerical problems when the matrices involved are not well conditioned. In this chapter, augmented UD identification (AUDI) method is proposed which avoids the inversion of ill-conditioned matrices and simultaneously produces the parameter estimates for all model orders from 1 to a user-specified value  $n$ , plus the corresponding loss functions. Multiple models are identified simultaneously and therefore the correct model order can be easily estimated and the correct model can be selected among the multiple models.

## 2.1 Introduction

Least-squares parameter estimation is basically a problem of solving linear simultaneous equations as in numerical analysis, or the problem of solving normal equations as in linear regression. Least-squares estimation, which has been widely used for many years, is the dominant estimation method in many fields. Although simple and well-founded in theory, the least-squares method may have fatal numerical problems in real applications because it may involve the inversion of an ill-conditioned matrix. Many variants and improvements of the least-squares algorithm exist in the literature, but most of them are just *ad hoc* modifications of the basic least-squares estimator. A detailed survey paper on the different modifications of the basic least-squares method can be found in Shah & Chett (1991).

The LU-decomposition (Doolittle's method) is a stable and efficient way for solving linear simultaneous equations and is widely used in numerical analysis (Dahlquist & Björck 1974, Gerald & Wheatley 1984). The concept has been successfully used by Bierman (1977) to derive a numerically more stable, recursive least-squares estimator, i.e., the UD Factorization method. The augmented UD identification method presented in this chapter is inspired by Doolittle's LU-decomposition, Bierman's UD factorization and the "shift structure" in Ljung, Morf & Falconer (1978). The key step is to construct the

---

<sup>1</sup>A version of this chapter has been published as: S. Ma, D. Grant Fisher and D. Xiao, 1992, An Augmented UD Identification Algorithm, *International Journal of Control*, Vol. 55, No. 1, 199-211.



data and parameter vectors used in identification in a different manner than in conventional methods. Analysis of the resulting AUDI structure shows that all the information on model parameters and loss functions is stored in the augmented covariance matrix (ACM), and can be made explicit by decomposing the ACM into a  $UDU^T$  factored form. The AUDI structure is simple and compact and simplifies the interpretation and implementation of least-squares identification. Also, as shown in the following chapters, it provides the basis for a whole family of recursive algorithms which are superior or equal to the recursive least-squares algorithms in almost every respects.

## 2.2 Process Models

A model is "a representation of the essential aspects of an existing system (or a system to be constructed) which presents knowledge of that system in a usable form" (Eykhoff 1974). Different models, for example conceptual models, physical models or mathematical models, can be used depending on the objective of the application. Mathematical models are quantitative representations of the characteristics of the real processes. They can be further classified into many different types, such as linear/nonlinear, dynamic/static, deterministic/stochastic and continuous/discrete-time. The focus of this thesis will be on linear, discrete-time, dynamic, difference equation models.

The process to be modeled can be depicted as in Figure 2.1, where  $u(t)$  represents the process input and  $y(t)$  is the process output. The noise/disturbance present in the output is represented by  $e(t)$  and the noise-corrupted output is represented by  $z(t)$ . Obviously

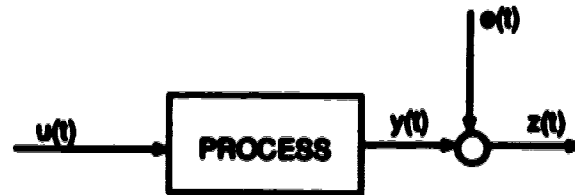


Figure 2.1: The Process

$$z(t) = y(t) + e(t)$$

In practice, the measurable output is always corrupted by noise, thus the processes investigated are stochastic processes.

Depending on the objective of the identification, different mathematical models can be used to describe the same process. For adaptive control, two kinds of models are common, the Autoregressive and Moving Average (ARMA) model and the Finite Impulse Response (FIR) model. For instance, the ARMA model, or one of its many variants, is widely used in Generalized Minimum Variance (GMV) control (Clarke & Gawthrop 1979) and Generalized Predictive Control (GPC) (Clarke & Mohtadi 1987) algorithms. The FIR model, also called a Moving Average (MV) model, is widely used in Model Process Control (MPC) algorithms such as Dynamic Matrix Control (DMC) (Cutler & Ramaker 1980) and the Multivariable Optimal Constrained Control Algorithm (MOCCA) (Sripada & Fisher 1985). The following general least squares format, also called the linear regression format in statistics, can be used for both models

$$z(t) = h^T(t) \theta + v(t) \quad (2.1)$$

where  $h(t)$  is the data vector and  $\theta$  is the parameter vector,  $v(t)$  is assumed to be zero-mean white noise. The variance of the process outputs and noise are represented by  $\sigma_s, \sigma_v$ , respectively, and are usually assumed to be unknown a priori. Most linear

difference equation models can be written in this general form and hence it will be used throughout this thesis. For example, for the ARMA model representation

$$z(t) + a_1 z(t-1) + \dots + a_n z(t-n) = b_1 u(t-1) + \dots + b_n u(t-n) + v(t) \quad (2.2)$$

the data and parameter vectors in (2.1) are defined as

$$h(t) = [-z(t-1), -z(t-2), \dots, -z(t-n), u(t-1), u(t-2), \dots, u(t-n)]^T \quad (2.3)$$

$$\theta_0 = [a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n]^T \quad (2.4)$$

For the FIR model representation

$$z(t) = s_1 u(t-1) + s_2 u(t-2) + \dots + s_n u(t-n) + v(t) \quad (2.5)$$

the data and parameter vectors in (2.1) are defined as

$$h(t) = [u(t-1), u(t-2), \dots, u(t-n+1), u(t-n)]^T \quad (2.6)$$

$$\theta = [s_1, s_2, \dots, s_{n-1}, s_n]^T \quad (2.7)$$

In this chapter, the ARMA model (2.2) is used to introduce the augmented UD identification structure for the sake of simplicity in notation. However, the AIDI structure applies to all models that can be represented by the least-squares format (2.1). For instance, as shown in Chapter 4, the CARMA or ARMAX model representation, in which the noise term is no longer white noise, can be rewritten into a pseudo-linear regression form, which is then in the same form as the least-squares format (2.1). Other model representations, plus features such as time delays, will be discussed in later chapters.

## 2.3 Least-squares Estimator

Least-Squares Estimation (LSE) can be traced back almost 200 years to Gauss (1809) who used this method to calculate the orbits of the planets. It is simple in concept and easy to implement and thus remains the most popular estimation algorithm.

Consider the ARMA model (2.2) and assume that the process noise  $v(t)$  is zero-mean white noise, with variance  $\sigma_v^2$ . Given the time series data representing the process input  $u(t)$  and output  $z(t)$ , the purpose of the parameter estimation is to find the best estimate of  $\theta$  that satisfies the equation (2.1) in the sense of minimizing a specific loss function (cost function). Different loss functions lead to different methods. For the least-squares method, the following quadratic function is used, i.e., the least-squares estimate is defined as the vector  $\hat{\theta}_{LS}$  that minimizes the following loss function

$$J(i, \hat{\theta}) = \sum_{j=1}^i \epsilon^2(j) = \sum_{j=1}^i [z(j) - h^T(j) \hat{\theta}]^2 \quad (2.8)$$

where  $\hat{\theta}$  is one of the parameter estimates and  $\epsilon(t) = z(t) - h^T(t) \hat{\theta}$  is called the residual, estimation error or equation error. Obviously, the loss function  $J(i, \hat{\theta})$  is a function of both time  $i$  and parameter estimate  $\hat{\theta}$ , and the residual  $\epsilon(t)$  is a linear function of the parameter estimate  $\hat{\theta}$ . The minimum loss function corresponding to  $\hat{\theta}_{LS}$  is represented by  $J(i)$  and is a function of time only

$$J(i) = \sum_{j=1}^i [z(j) - h^T(j) \hat{\theta}_{LS}]^2$$

**Theorem 2.1 (Least-squares estimate)** For the loss function  $J(t, \hat{\theta})$  given by (2.8), the unique minimum point of  $J(t, \hat{\theta})$  is given by

$$\hat{\theta}_{LS}(t) = \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1} \sum_{j=1}^t h(j) z(j) \quad (2.9)$$

under the condition that  $\sum_{j=1}^t h(j) h^T(j)$  is positive definite. The corresponding minimum value of the cost function is given by

$$J(t) = \sum_{j=1}^t z^2(j) - \hat{\theta}_{LS}^T(t) \left( \sum_{j=1}^t h(j) h^T(j) \right) \hat{\theta}_{LS}(t) \quad (2.10)$$

**Proof:** Based on formula (2.8), assuming that  $\hat{\theta}_{LS}(t)$  minimizes  $J(t, \hat{\theta})$ , then

$$\frac{\partial J(t, \hat{\theta})}{\partial \hat{\theta}} \Big|_{\hat{\theta}_{LS}} = \frac{\partial}{\partial \hat{\theta}} \left( \sum_{j=1}^t (z(j) - h^T(j) \hat{\theta}(t))^2 \right) = 0$$

or

$$\left( \sum_{j=1}^t h(j) h^T(j) \right) \hat{\theta}_{LS}(t) = \sum_{j=1}^t h(j) z(j) \quad (2.11)$$

Equation (2.11) is also called the normal equation in linear regression (Gerald & Wheatley 1984, Söderström & Stoica 1989). If  $\sum_{j=1}^t h(j) h^T(j)$  is positive-definite, equation (2.11) immediately gives (2.9) by a inverse. Substituting (2.9) into (2.8) and then expanding it leads to (2.10).  $\sum_{j=1}^t h(j) h^T(j)$  is usually called the information matrix or data product moment matrix and is represented by  $R(t)$ .

From now on, for simplicity of notation, we will drop the subscript 'LS' in  $\hat{\theta}_{LS}$  and directly use  $\hat{\theta}$  for the least-squares estimate.

**Remark 1.** Assuming that the input  $u(t)$  and output  $z(t)$  satisfy (2.1) and  $v(t)$  is white noise with zero-mean and variance  $\sigma_v^2$ , then

1.  $\hat{\theta}(t)$  is the best linear unbiased estimate (BLUE) of the parameter vector  $\theta$  in (2.4).
2. The covariance matrix of the parameter estimates  $\hat{\theta}(t)$  is given by

$$\text{cov}(\hat{\theta}(t)) = \sigma_v^2 \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1}$$

Define  $P(t) = \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1}$ . Since  $P(t)$  is proportional to the covariance matrix, it is usually called the covariance matrix.

3. An unbiased estimate of  $\sigma_v^2$  is given by (Isermann 1981, Goodwin & Payne 1977)

$$\hat{\sigma}^2(t) = \frac{J(t)}{t-d}$$

where  $d$  is the dimension of  $\hat{\theta}(t)$ .

**Remark 2.** With the best linear unbiased estimate  $\hat{\theta}(t)$  for model (2.2), the residual term

$$e(t) = z(t) - h^T(t) \hat{\theta}(t)$$

is orthogonal to the space spanned by the data vector  $h(t)$ .

For proof of the above remarks see Söderström & Stoica (1989).

## 2.4 The Augmented UD Identification Structure

The least-squares estimator is simple and clear in concept, and gives an unique parameter estimate for process (2.1) when the matrix  $\sum_{j=1}^i h(j) h^T(j)$  is positive-definite. The augmented UD identification structure in this section reformulates and interprets the least-squares estimator in a different way which leads to a simpler, more compact and convenient structure and provides much more information than the least-squares estimator.

Starting with the process model (2.2), define the augmented data vector as

$$\varphi(t) = [-z(t-n), u(t-n), \dots, -z(t-1), u(t-1), -z(t)]^T \quad (2.12)$$

Compared with the data vector  $h(t)$  (2.3) in the least-squares estimator, the elements of the augmented data vector  $\varphi(t)$  are grouped in  $\{z, u\}$  pairs and the current process output  $z(t)$  is included in the augmented data vector. As a result, the augmented data vector exhibits the following "shift structure" (Ljung et al. 1978, Ljung & Söderström 1983)

$$\varphi(t) = \begin{pmatrix} h(t) \\ -z(t) \end{pmatrix} \quad (2.13)$$

A more detailed discussion of the "shift structure" is included in Appendix A. The data vector defined by (2.12) is the basis of the AUEI structure developed in this section. Many of the AUEI variants differ mainly in the formulation of the augmented data vector.

Now define a new matrix, the augmented covariance matrix (ACM), as follows

$$C(t) = \left[ \sum_{j=1}^i \varphi(j) \varphi^T(j) \right]^{-1}_{(2n+1) \times (2n+1)} \quad (2.14)$$

Note that the augmented covariance matrix (2.14) has a structure similar to the covariance matrix  $P(t)$  in least-squares estimation, but with an augmented dimension.

Decomposing the ACM into a  $UDU^T$  factored form produces (proof see Appendix A)

$$C(t) = U(t) D(t) U^T(t) \quad (2.15)$$

where  $U(t)$  is an upper triangular matrix with all the diagonal elements equal to unity, i.e.,

$$U(t) = \begin{bmatrix} 1 & \hat{d}_1^{(0)} & \hat{d}_1^{(1)} & \hat{d}_1^{(2)} & \dots & \hat{d}_1^{(n-1)} & \hat{d}_1^{(n)} \\ & 1 & \hat{d}_2^{(1)} & \hat{d}_2^{(2)} & \dots & \hat{d}_2^{(n-1)} & \hat{d}_2^{(n)} \\ & & 1 & \hat{d}_3^{(2)} & \dots & \hat{d}_3^{(n-1)} & \hat{d}_3^{(n)} \\ & & & 1 & \dots & \hat{d}_4^{(n-1)} & \hat{d}_4^{(n)} \\ & & & & \ddots & \vdots & \vdots \\ & & & & & 1 & \hat{d}_n^{(n)} \\ & & & & & & 1 \end{bmatrix}_{(2n+1) \times (2n+1)} \quad (2.16)$$

$D(t)$  is a diagonal matrix, with the form

$$[D(t)]^{-1} = \text{diag} \left[ J^{(0)}(t), L^{(0)}(t), \dots, J^{(n-1)}(t), L^{(n-1)}(t), J^{(n)}(t) \right] \quad (2.17)$$

The bracketed superscripts represent the model orders, *e.g.*,  $\hat{\theta}^{(n)}$  represents the estimated parameter vector of the  $n$ th order model. Further information on the  $U(t)$  and  $D(t)$  matrices is summarized in the following remarks.

1. The  $U(t)$  matrix is called the *parameter matrix* and represented by  $U$ . Each column of the parameter matrix  $U$  contains the parameter estimates of a specific model. For instance, the last column

$$\hat{\theta}^{(n)}(t) \triangleq \left[ \hat{\theta}_1^{(n)}, \hat{\theta}_2^{(n)}, \dots, \hat{\theta}_{2n}^{(n)} \right]^T$$

is the estimated parameter vector of (2.4), which is exactly the least-squares estimate given by (2.9). Similarly, the vector

$$\hat{\theta}^{(j)}(t) \triangleq \left[ \hat{\theta}_1^{(j)}, \hat{\theta}_2^{(j)}, \dots, \hat{\theta}_{2j}^{(j)} \right]^T, \quad j \in [1, n]$$

has a form similar to (2.4) and is the estimated parameter vector of the  $j$ th order model of process (2.2).

2. The  $\alpha$  elements in the  $U$  matrix are parameter estimates of another model type and will be discussed later.
3. Define  $\mathcal{D} \triangleq D^{-1}(t)$ . The  $\mathcal{D}$  matrix is called the *loss function matrix* since each diagonal element is the loss function of a specific model corresponding to those in the parameter matrix  $U$ . For example, the last element,  $J^{(n)}(t)$ , is the loss function of the  $n$ th order model that is defined by the parameter estimate  $\hat{\theta}^{(n)}(t)$  in the parameter matrix  $U$ .  $J^{(n)}(t)$  is calculated in exactly the same way as (2.10), see Appendix A for a proof. Similarly, the other elements of the loss function matrix,  $J^{(j)}(t)$ ,  $j \in [1, n]$ , are loss functions of the corresponding  $j$ th order models.
4. The  $L$  elements ( $L^{(0)}(t)$  to  $L^{(n)}(t)$ ) in the loss function matrix  $\mathcal{D}$  are loss functions of the  $\alpha$ -models mentioned in Remark 2, and will be discussed together with the  $\alpha$  parameters of the  $U$  matrix in a later chapter.

For system models with different numbers of  $a$  parameters and  $b$  parameters, this augmented data vector should be defined differently and some part of the data may not be paired. For example, the augmented data vector for a system with two  $a$  parameters and four  $b$  parameters can be defined as

$$\varphi(t) = [u(t-4), u(t-3), \underline{-s(t-2)}, \underline{u(t-2)}, \underline{-s(t-1)}, \underline{u(t-1)}, \underline{-s(t)}]^T$$

Accordingly, the parameter and loss function matrices should also be interpreted differently.

From the above remarks it is seen that the augmented covariance matrix implicitly "accumulates" all the information about the model parameters and loss functions and thus this matrix is also called the *information accumulation matrix (IAM)* which is more meaningful in later chapters where recursive identification algorithms are discussed. After applying the  $UDU^T$  decomposition technique, all the desired information is explicitly

contained in the parameter matrix  $U$  and the loss function matrix  $\mathcal{D}$ . This decomposed ACM is called the *augmented UD identification structure*, where UD indicates that this structure comes from a  $UDU^T$  decomposition. The recursive implementation of the AUDI structure, *i.e.*, the recursive AUDI algorithm (Chapter 3), is similar to Bierman's UD Factorization algorithm (Bierman 1977). Since  $U$  is the parameter matrix and  $\mathcal{D}$  is the loss function matrix, the name augmented UD identification also stands for simultaneous identification of model parameters and loss functions.

## 2.5 Decomposition

The augmented covariance matrix (ACM) contains all the information about the parameter estimates and loss functions. Therefore, all that is required for augmented UD identification is to collect the process input and output data and construct the ACM. Whenever needed, a  $UDU^T$ -decomposition can be applied to extract the required information.

The inverse of the ACM, called the augmented information matrix (AIM) or data product moment matrix (DPMM) in consistence with the notations in conventional least-squares, is defined as

$$S(i) = \left[ \sum_{j=1}^i \varphi(j) \varphi^T(j) \right]_{(2n+1) \times (2n+1)} \quad (2.18)$$

It contains the same information as the ACM in (2.14). The information about process parameters and loss functions can be extracted using the well-known LU-decomposition (Dahlquist & Björck 1974). When the AIM is symmetrical, the  $LDL^T$  decomposition (Dahlquist & Björck 1974) can also be used. In this thesis, the ACM and the  $UDU^T$  decomposition will be used because the  $UDU^T$  decomposition can take advantage of Bierman's UD Factorization algorithm (Bierman 1977) in the recursive AUDI algorithms described later. For analysis purpose, either the ACM or the AIM can be used. This section will discuss the LU- and  $UDU^T$ -decomposition techniques and show that these two representations are equivalent.

**Theorem 2.2 (LU-Decomposition)** 1. Let  $S$  be a given  $n \times n$  matrix, and denote by  $S_k$  the  $k \times k$  matrix formed by the intersection of the first  $k$  rows and columns in  $S$ . If  $\det(S_k) \neq 0$  for  $k=1, 2, \dots, n$ , then there exists a unique unit-lower-triangular matrix  $L$  and a unique upper-triangular matrix  $U$ , such that  $S=LU$ .

2. Under the same hypothesis,  $S$  can also be uniquely decomposed into  $S=LDU$  form where  $D$  is a diagonal matrix and  $U$  is a unit-upper-triangular matrix.

3. If  $S$  is symmetrical as well, then the  $S$  matrix can be uniquely decomposed into the form  $S=LDL^T$  where  $L$  and  $D$  are the same as above.

**Proof:** The above theorem can be easily proved by induction. A detailed proof can be found in Dahlquist & Björck (1974).

Theorem 2.2 immediately leads to the following theorem.

**Theorem 2.3 ( $UDU^T$  Decomposition)** Let  $C(i)$  be the symmetrical augmented covariance matrix in (2.14) with dimension  $d \times d$ , where  $d=2n+1$ . Denote by  $C_k(i)$  the  $k \times k$

matrix formed by the intersection of the first  $k$  rows and columns in  $C(t)$ . Under the condition that  $\det(C_k) \neq 0$  for  $k=1, 2, \dots, d-1$ ,  $C(t)$  can be uniquely decomposed into  $C(t) = U(t) D(t) U^T(t)$  form where  $U(t)$  is a unit-upper-triangular matrix, and  $D(t)$  is diagonal.

**Proof:** The proof of Theorem 2.3 is similar to the proof of Theorem 2.2 and thus is omitted here.

The steps required to carry out the LU- and/or UDU<sup>T</sup>-decomposition are summarized in Table 2.1, where  $C_{ij}(t)$  is the  $(i, j)$ th element of the ACM and  $S_{ij}(t)$  is the  $(i, j)$ th element of the AIM matrix.  $I$  stands for the identity matrix. Notice that the ACM is the inverse

Table 2.1: The LU and UDU<sup>T</sup> decompositions

The LU-Decomposition	The UDU <sup>T</sup> Decomposition
$L(t) = I, U(t) = I$ <b>for</b> $i=1$ <b>to</b> $n$ <b>for</b> $j=1$ <b>to</b> $i-1$ $\alpha=0$ <b>for</b> $k=1$ <b>to</b> $j-1$ $\alpha = \alpha + L_{ik}(t) U_{kj}(t)$ $L_{ij}(t) = (S_{ij}(t) - \alpha) / U_{jj}(t)$ $\beta=0$ <b>for</b> $k=1$ <b>to</b> $i-1$ $\beta = \beta + L_{ik}(t) U_{kj}(t)$ $U_{ij}(t) = S_{ij}(t) - \beta$	$U(t) = I, D(t) = I$ <b>for</b> $i=n$ <b>downto</b> $2$ $D_{ii}(t) = C_{ii}(t)$ $\alpha = 1/D_{ii}(t)$ <b>for</b> $j=1$ <b>to</b> $i-1$ $\beta = C_{ji}(t)$ $U_{ji}(t) = \alpha \beta$ <b>for</b> $k=1$ <b>to</b> $j$ $C_{kj}(t) = C_{kj}(t) - \beta U_{ki}(t)$ $D_{11}(t) = C_{11}(t)$

of the AIM

$$C(t) = S^{-1}(t)$$

The UDU<sup>T</sup>-decomposition of the augmented covariance matrix and the LDL<sup>T</sup>-decomposition of the augmented information matrix are given by

$$\begin{array}{ll}
 \text{UD:} & C(t) = U(t) D(t) U^T(t), \Rightarrow \begin{cases} U = U(t) \\ D = D^{-1}(t) \end{cases} \\
 \text{LDL}^T & S(t) = L(t) D(t) L^T(t), \Rightarrow \begin{cases} U = L^{-T}(t) \\ D = D(t) \end{cases} \\
 \text{LU} & S(t) = L(t) U(t), \Rightarrow \begin{cases} U = L^{-T}(t) \\ D = \text{diag} U(t) \end{cases} \\
 \text{Cholsky} & S(t) = L(t) L^T(t), \Rightarrow \begin{cases} U = \text{diag} L(t) \cdot L^{-T}(t) \\ D = [\text{diag} L(t)]^2 \end{cases}
 \end{array}$$

where "<sup>-T</sup>" stands for transpose and inverse. The inverse of the lower-diagonal matrix  $L(t)$  is always available and stable, and inverting  $L(t)$  requires much less computation than inverting a full matrix. A formula for inverting a lower-triangular matrix is given below and can be used to calculate the inverse in  $U = L^{-T}(t)$  (Dahlgren & Björck 1974)

$$u_{ji} = \frac{L_{ij}(t) - \sum_{k=j}^{i-1} L_{ik}(t) u_{jk}}{L_{ii}(t)}, \quad \begin{matrix} j=1, 2, \dots, n \\ i=j, j+1, \dots, n \end{matrix} \quad (2.19)$$

where  $\delta_{ij}$  stands for the element of identity matrix, *i.e.*,

$$\delta_{ij} = \begin{cases} 1 & \text{when } i=j. \\ 0 & \text{otherwise.} \end{cases}$$

The inverse of upper-triangular matrix is similar and the formula can also be found in Dahlquist & Björck (1974).

The above results are summarized as follows

1. All the information about the process parameters and loss functions is contained in the augmented covariance matrix or its inverse, the augmented information matrix. Therefore, the purpose of system identification is to produce the parameter matrix  $U$  and loss function matrix  $\mathcal{D}$  by collecting the input/output data, formulating and decomposing the augmented covariance matrix or augmented information matrix.
2. The implementation of the AUDI method can be in three different ways
  - (a) constructing the augmented covariance matrix and using the UD factorization to produce the parameter and loss function matrices. This is the recommended method for recursive implementation.
  - (b) constructing the augmented information matrix and using the Cholsky/LU/LDU/LDL<sup>T</sup> decomposition method to produce the parameter and loss function matrices. This method is recommended for batch implementation where the augmented information matrix  $S(i)$  is more readily available.
  - (c) the third method is to directly decompose the augmented data matrix with a QR type decomposition. The results of the QR decomposition is the same as the Cholsky decomposition and thus are not discussed here. For batch implementation, the QR decomposition is expected to have better numerical properties than the above mentioned two methods. For recursive implementation, however, QR decomposition has about the same numerical performance as the recursive UD factorization method which is discussed in next chapter.

A schematic diagram of the different implementation methods of the AUDI approach is depicted in Figure 2.2. Note that every method starts with the input/output data and produces the parameter matrix  $U$  and loss function matrix  $\mathcal{D}$ . The parameter and loss function matrices have standard structure and interpretation no matter which type of matrix and/or decomposition method is used.

## 2.6 Model Order Determination

In system identification, it is usually assumed that the model order is known *a priori* and fixed, so that the system identification is merely a problem of determining the coefficients of the difference equation. In practice, however, the order of the process is seldom exactly known *a priori*, or the input/output relationship may change from time to time. Therefore, order determination is an very important part of system identification. Order identification is also referred to as structure identification in the multivariable case since a set of structural indices, rather than model order, is used to describe the structure of the process (Goulden 1975, Goulden 1981). Incorrect model order (*e.g.*,



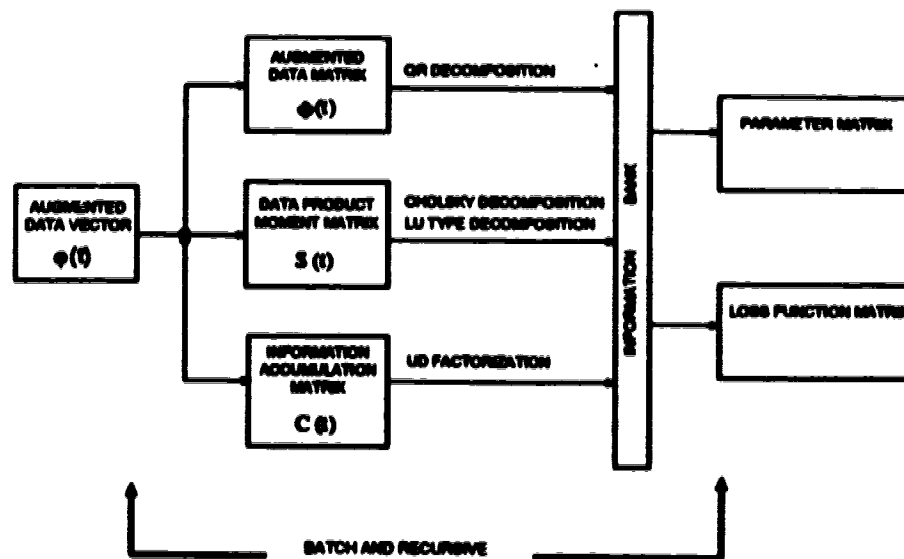


Figure 2.2: Batch and Recursive Implementation Methods of the AUDI Method

over-parameterization or under-parameterization) can cause serious problems in control system design. Hence it is very important to test frequently the adequacy of the model order during process identification.

A simple but very efficient and widely used method for order determination is to compare the goodness-of-fit of the model to the observed data for different model orders  $\hat{n}$ . The goodness-of-fit is usually measured by the loss function  $J(t)$  as in (2.10). It can be proven (Fang & Xiao 1988) that when the estimated model order  $\hat{n}$  is greater than the real model order  $n_0$ , the variance of the residual (estimation error) tends to be a constant, under the condition of zero-mean white noise. In general, for different model orders  $\hat{n}$ , the loss function  $J^{(\hat{n})}(t)$  decreases as  $\hat{n}$  increases. However, the decrease of the loss function  $J$  ceases to be significant when  $\hat{n}$  becomes greater than the true model order  $n_0$ .

The common procedure to use this principle is to simply compute the least-squares estimates  $\hat{\theta}^{(\hat{n})}(t)$  and the corresponding loss functions  $J^{(\hat{n})}(t)$  for all models from order 1 to a user-specified maximum possible order  $n$ . The appropriate model order is then chosen as the one for which that the loss function is not significantly greater than the one for the next higher order.

In the previous sections, it was shown that the AUDI structure contains all the model parameters and corresponding loss functions for all model orders from 1 to  $n$ . This is exactly the information required for model order determination as discussed above. Therefore, order determination with the AUDI structure is very simple and convenient. This is actually one of the significant advantages of using the AUDI structure.

Obviously, given the loss functions for different model orders, the order determination becomes a problem of statistics: when  $\hat{n}$  is increased from  $\hat{n}_1$  to  $\hat{n}_2$ , determine whether  $J^{(\hat{n}_2)}(t)$  is significantly smaller than  $J^{(\hat{n}_1)}(t)$ . The significance can be checked using Åström's F-test method (Åström & Eykhoff 1971). A detailed description of the F-test method can be found in many books, e.g., Fang & Xiao (1988), Söderström & Stoica (1989).

Other widely used order-determination methods include Akaike's Information Crite-

tion (AIC) (Akaike 1974, Akaike 1981), and the Final Prediction Error (FPE) method, etc. The principle of these methods is different than the F-test method, but it can be proven (Söderström & Stoica 1989) that they are equivalent to each other and can be interpreted in terms of the relationship between loss functions and model orders.

Another important consideration is time delay determination. The determination of process time delay is quite similar to the determination of model order. Assuming a possible range of the value of the process time delay, by explicitly identifying all model with different time delays in this range, the delay that leads to the smallest loss function is the estimated process time delay.

If the time delay changes in the vicinity of a nominal value and this nominal value is known, then an explicit search for the changed time delay is not necessary. The AUDI method can automatically compensate the small changes in time delay by adjusting the model parameters and even the model order. This is an important advantage of the AUDI method over the conventional least-squares method.

## 2.7 Simulation Examples

Consider a process represented by the following SISO linear difference equation model

$$z(t) - 1.5z(t-1) + 0.7z(t-2) = u(t-1) + 0.5u(t-2) + v(t)$$

where  $z(t)$  and  $u(t)$  are the process output and input respectively;  $v(t)$  is zero-mean white noise with variance  $\sigma_v^2=0.25$  (or standard deviation  $\sigma_v=0.5$ ). A random binary sequence is used as the process input. Assuming the maximum possible model order is  $n=4$ , the following data vector can be constructed

$$\varphi(t) = \begin{bmatrix} -z(t-4), u(t-4), -z(t-3), u(t-3), \\ -z(t-2), u(t-2), -z(t-1), u(t-1), -z(t) \end{bmatrix}^T$$

Note the paired variables in  $\varphi(t)$  and the fact that it has been augmented with the current output  $z(t)$ . The augmented information matrix

$$S(t) = \sum_{j=1}^t \varphi(j) \varphi^T(j)$$

is then constructed using 500 data points and shown in Table 2.2.

This is a symmetrical positive-definite matrix and can be uniquely decomposed into the form

$$S(t) = L(t) D(t) L^T(t)$$

The parameter matrix  $U=L^{-T}(t)$  is then calculated and shown in Table 2.3. The corresponding loss functions are obtained from the loss function matrix  $\mathcal{D}=D^{-1}(t)$ , shown in Table 2.4.

In the parameter matrix  $U$ , the fifth column is the parameter estimate of the second order AFMA parameter vector (2.4). The third column is the parameter estimate of the first order model of (2.2) which is

$$z(t) + a_1 z(t-1) = b_1 u(t-1) + v(t)$$

**Table 2.2: The Augmented Information Matrix**

9382.8	101.1	8308.3	493.2	5354.1	-9.7	1980.5	-54.3	-1580.0
101.1	500.0	-373.4	0.8	-885.4	-42.7	-1029.4	-18.4	-848.8
8308.3	-373.4	9359.9	492.3	7913.3	47.9	4995.7	-38.0	1024.4
493.2	0.8	492.3	500.2	-128.0	-2.4	-810.5	-44.8	-1133.6
5354.1	-885.4	7913.3	-128.0	8824.0	93.4	7853.5	34.3	4412.8
-9.7	-42.7	47.9	-2.4	93.4	500.1	-385.8	-3.9	-907.9
1980.5	-1029.4	4995.7	-810.5	7853.5	-385.8	8909.6	95.0	7598.9
-54.3	-18.4	-38.0	-44.8	34.3	-3.9	95.0	500.3	-370.2
-1580.0	-848.8	1024.4	-1133.6	4412.8	-907.9	7598.9	-370.2	9171.5

**Table 2.3: Parameter Matrix Obtained by LDL<sup>T</sup> Decomposition**

1	-0.0108	-0.8974	-0.0237	0.7019	0.0588	-0.0108	0.0023	0.0612
	1	0.9284	-0.0210	0.5088	0.1104	0.0167	0.0422	0.0570
		1	-0.0324	-1.5003	-0.1155	0.7145	0.0504	-0.0384
			1	0.9891	0.0748	0.5211	0.1296	0.0881
				1	0.0898	-1.5029	-0.1134	0.7991
					1	0.9873	0.0890	0.5244
						1	0.0785	-1.5036
							1	0.9893
								1
	order 1		order 2		order 3		order 4	

**Table 2.4: Loss Function Matrix Obtained by LDL<sup>T</sup> Decomposition**

9382.8								
	498.9							
		1557.3						
			498.1					
				134.3				
					495.6			
						134.2		
							494.9	
								134.1
order 0	order 1	order 2	order 3	order 4				

Similarly, the seventh column is the parameter estimates of the third order model which can be represented by

$$z(t) + a_1 z(t-1) + a_2 z(t-2) + a_3 z(t-3) = b_1 u(t-1) + b_2 u(t-2) + b_3 u(t-3) + v(t)$$

In general, column  $(2i+1)$  of the parameter matrix  $U$  contains parameter estimates for the  $i$ th order model, with  $i \in [1, n]$ .

Similarly, the loss function corresponding to the parameter estimates of the 2nd order model can be obtained directly from the 5th diagonal element of the loss function matrix  $D$  (Table 2.4), i.e., 134.3. In general, the loss function of the  $i$ th order model is the  $(2i+1)$ st diagonal element in the loss function matrix.

Since the maximum possible order was specified as  $n=4$ , all the parameter estimates and loss functions for models from order 1 to 4 are produced simultaneously in the parameter matrix  $U$  and loss function matrix  $D$  respectively. This is the most distinguishing feature of the ALDI structure and is obviously convenient for many applications, as will be discussed in later chapters. The relationship between the loss functions and the model orders is plotted in Figure 2.3, from which it is seen that the loss functions decrease as the model order goes from 0 to 1 and from 1 to 2. However, increasing the

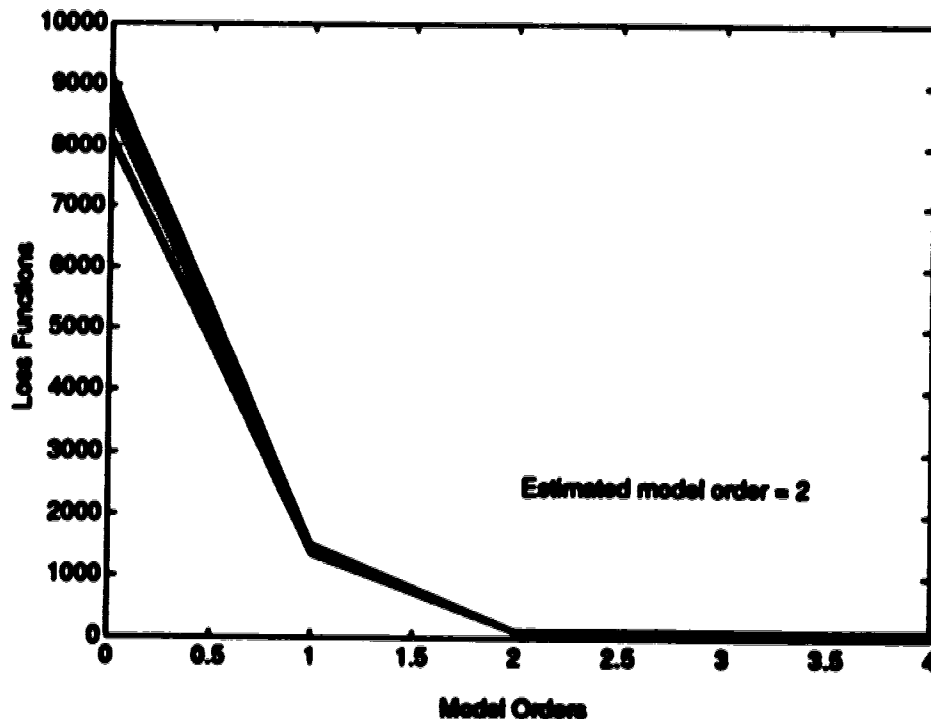


Figure 2.3: The Loss Functions versus Model Orders

model order from 2 to 3 or from 3 to 4 does not decrease the loss function further. This clearly indicates that the process has second order dynamics. Criteria such as the F-test and the AIC can be used for analysis and both of them result in the same estimate of 2.

## **2.8 Conclusions**

The AUDI algorithm simplifies the interpretation and implementation of the least-squares estimator, and also provides much more information than the basic least-squares estimator. Because the model order can be easily determined based on the loss function matrix, it is obvious that the AUDI structure and the AUDI algorithms possess the ability for simultaneous order identification and parameter estimation. The AUDI structure developed in this chapter is the basis for the other AUDI algorithms and analysis presented in the following chapters.

# Bibliography

- Akaike, H. (1974), 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control* 19(6), 716 - 723.
- Akaike, H. (1981), Modern development of statistical methods, in P. Eykhoff, ed., 'Trends and Progress in System Identification', Pergamon Press, Elmsford, N.Y.
- Åström, K. J. & Eykhoff, P. (1971), 'System identification — a survey', *Automatica* 7(2), 123 - 162.
- Bierman, G. J. (1977), *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York.
- Clarke, D. W. & Gawthrop, P. J. (1979), 'Self-tuning control', *IEE Proceedings, Part D*, 128, 663 - 640.
- Clarke, D. W. & Mohtadi, C. (1987), 'Generalized predictive control - part I. the basic algorithm. part II. extensions and interpretations', *Automatica* 23(2), 137 - 160.
- Cutler, C. R. & Ramaker, B. L. (1980), Dynamic matrix control — a computer control algorithm, in 'Proceedings of the 1980 Joint Automatic Control Conference', San Francisco, pp. WP5-B.
- Dahlquist, G. & Björck, A. (1974), *Numerical Methods*, Prentice Hall, New Jersey. (Translated by N. Anderson).
- Eykhoff, P. (1974), *System Identification*, John Wiley & Sons, New York.
- Fang, C. & Xiao, D. (1988), *Process Identification*, Tsinghua University Press, Beijing, China.
- Gauss, K. F. (1963), *Theoria motus corporum coelestium in Sectionibus Conicis Solem Ambientium (1809)*, English Translation: *Theory of the Motion of Heavenly Bodies Moving About the Sun in Conic Sections*, Dover, New York.
- Gerald, C. F. & Wheatley, P. (1984), *Applied Numerical Analysis*, Third edn, Addison-Wesley.
- Goodwin, G. C. & Payne, R. L. (1977), *Dynamic System Identification: Experimental Design and Data Analysis*, Academic Press, New York.
- Guidorzi, R. P. (1975), 'Canonical structure in the identification of multivariable systems', *Automatica* 11(4), 361-374.

- Guidorzi, R. P. (1981), 'Invariant and canonical forms for systems structural and parameter identification', *Automatica* 17(1), 117-133.
- Isermann, R. (1981), *Digital Control Systems*, Springer-Verlag, New York.
- Ljung, L. & Söderström, T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass.
- Ljung, L., Morf, M. & Falconer, D. (1978), 'Fast calculation of gain matrices for recursive estimation schemes', *International Journal of Control* 27(1), 1 - 19.
- Niu, S., Fisher, D. G. & Xiao, D. (1992), 'An augmented UD identification algorithm', *International Journal of Control* 56(1), 193 - 211.
- Söderström, T. & Stoica, P. (1989), *System Identification*, Prentice Hall, Englewood Cliffs, New Jersey.
- Sripada, N. R. & Fisher, D. G. (1985), Multivariable Optimal Constrained Control Algorithm (MOCCA): Part 1. formulation and application, in 'Proceedings of International Conference on Industrial Process Modeling and Control', Vol. 1, Hangzhou, China.

## Chapter 3

# THE AUDI-LS ALGORITHM

**T**he least-squares estimator and the AUDI structure discussed in Chapter 2 are often referred to as the batch processing method, since the process input/output data are recorded first and all the recorded data are then used simultaneously to find the parameter estimates for the process model. For many applications, however, a dynamic model is needed to support decisions which must be made on-line, i.e., during the operation of the process. For these applications, it is often necessary to estimate the model parameters at the same time as the input/output data are collected from the process. Such an approach is usually called recursive identification, sequential identification or on-line identification.

### 3.1 Introduction

Recursive identification of dynamic models is the cornerstone of many adaptive systems such as adaptive control and adaptive filtering. The success of an adaptive control scheme is usually highly dependent on how well the recursive identification performs. Much of the reluctance to apply adaptive control to real industrial processes is due to the complexities and uncertainties associated with recursive identification (Tjokro 1984). Numerous papers on improving the robustness of control law (Clarke & Mohtadi 1987, Chueti 1986) are aimed at reducing the reliance on model accuracy.

The basic idea of recursive identification is that, assuming the parameter estimate  $\hat{\theta}(i-1)$  based on the data up to time  $i-1$ , is known, then  $\hat{\theta}(i)$  for the current time step can be computed by some "simple modification" to  $\hat{\theta}(i-1)$ . The general updating formula is as follows

$$\begin{bmatrix} \text{New} \\ \text{Estimate} \end{bmatrix} = \begin{bmatrix} \text{Previous} \\ \text{Estimate} \end{bmatrix} + \begin{bmatrix} \text{Gain} \\ \text{Vector} \end{bmatrix} \times \begin{bmatrix} \text{Prediction} \\ \text{Error} \end{bmatrix} \quad (3.1)$$

The new estimate is obtained from the previous estimate by adding a correction term, which is based on the prediction error and controlled by a gain vector. Recursive identification has some obvious advantages over batch methods. It requires a modest amount of memory and the requirement does not increase with time, since not all data are stored.

---

<sup>1</sup>A version of this chapter has been published as: S. Niu, D. Grant Fisher and D. Xing, 1992, An Augmented UD Identification Algorithm, *International Journal of Control*, Vol. 55, No. 1, 109-211.



The computational requirement at each time interval is relatively low and fixed and thus can be converted into a real-time algorithm to track time-varying parameters.

In this chapter, a recursive implementation of the AUDI structure is developed by modifying Bierman's UD factorization technique. The resulting algorithm is called the AUDI algorithm, or AUDI-LS algorithm to distinguish it from the AUDI-IV, AUDI-ELS algorithms developed in later chapters. The AUDI algorithm has the following features

1. **Simultaneous identification of model parameters and loss functions**, for all models from order 1 to a user specified order  $n$ . In other words, multiple models can be identified at every time interval. This provides the basis for simultaneous identification of model parameters and model structure (order and/or time delay).
2. **Robust performance**. The AUDI structure guarantees that the augmented covariance matrix is always well-conditioned so that the algorithm is numerically sound. Consistent identification results can be obtained under most process conditions.
3. **Low computational effort**. The AUDI algorithm simultaneously identifies  $n$  different models from order 1 to  $n$ , with a computational effort equivalent to that of the  $n$ th order RLS. In this sense, the AUDI algorithm is quite efficient in implementation.
4. **Simple and compact structure for interpretation and implementation**. The AUDI algorithm uses the AUDI structure and thus it is clear and compact.
5. **Easy extensions**. The informative AUDI structure provides the basis for many useful extensions such as estimation of signal-to-noise ratio (Chapter 8), recursive information forgetting and on/off criteria (Chapter 7). In comparison with the lattice type algorithms (Friedlander 1983, Samson 1982), which can also provide simultaneous order identification and parameter estimation, the AUDI-IV algorithm is simpler both in interpretation and implementation.

This chapter is organized as follows. The recursive least-squares method and Bierman's UD factorization algorithm are briefly reviewed in Section 3.2. Then the AUDI algorithm is developed in sections 3.3. The properties of the recursive AUDI algorithm are discussed in section 3.4. In section 3.5, the AUDI algorithm and RLS algorithm are compared and finally in Section 3.6, some simulation examples are presented.

## **3.2 Recursive Least-Squares Algorithms**

The recursive implementation of the LS estimator (2.9), i.e., the Recursive Least-Squares algorithm (RLS) (Plett 1960, Ljung & Söderström 1983), is the most commonly used recursive parameter estimation scheme. Although widely accepted both in theory and in practice, the RLS is also well-known to have numerical difficulties. Poor numerical properties can destroy the performance of the RLS especially when being implemented on digital computers with finite numerical accuracy (Bierman 1977, Clarke & Gawthrop 1979, Shah & Chett 1991). One of the most successful approaches to improve the numerical performance of the RLS algorithm is Bierman's UD factorization method (Bierman 1977, Thornton & Bierman 1980). Bierman's method is mathematically equivalent to the RLS method. However, because of the different formulation, it is numerically much better than the RLS algorithm (Bierman 1977, Thornton & Bierman 1980, Ljung & Ljung 1986). Surprisingly, the UD factorization algorithm has not been as widely used

as the RLS algorithm in spite of its superior performance over RLS. One of the main reasons may be that the UD factorization algorithm appears to be much more complicated to interpret and implement.

### 3.2.1 The Basic RLS Algorithm

The recursive least-squares algorithm is one of the recursive implementations of the least-squares estimator (2.9) and has been widely accepted in many fields. Detailed derivations and discussion of the RLS algorithm can be found in almost every text book on parameter estimation (Goodwin & Sin 1984, Ljung 1987, Ljung & Söderström 1983, Söderström & Stoica 1989). Therefore only a brief review is included here for comparison with the AUDI algorithm.

Starting from the normal equation (2.11) in Chapter 2, remember that the covariance matrix

$$P(t) = \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1} \quad (3.2)$$

trivially

$$P^{-1}(t) = P^{-1}(t-1) + h(t) h^T(t) \quad (3.3)$$

It then follows from the least-squares estimator (2.9) that

$$\begin{aligned} \hat{\theta}(t) &= P(t) \left[ \sum_{j=1}^{t-1} h(j) z(j) + h(t) z(t) \right] \\ &= P(t) \left[ P^{-1}(t-1) \hat{\theta}(t-1) + h(t) z(t) \right] \\ &= \hat{\theta}(t-1) + P(t) h(t) [z(t) - h^T(t) \hat{\theta}(t-1)] \end{aligned}$$

Using the matrix inversion (rank one update) formula (Söderström & Stoica 1989)

$$(A + B B^T)^{-1} = A^{-1} - A^{-1} B (I + B^T A^{-1} B)^{-1} B^T A^{-1}$$

$P(t)$  can be calculated as

$$P(t) = P(t-1) - \frac{P(t-1) h(t) h^T(t) P(t-1)}{1 + h^T(t) P(t-1) h(t)} \quad (3.4)$$

Denoting  $s(t) = 1 + h^T(t) P(t-1) h(t)$  and  $K(t) = P(t) h(t)$ , the complete RLS algorithm is then given in Table 3.1, where  $K(t)$  is the Kalman gain vector.  $s(t)$  is called the

Table 3.1: The Recursive Least-Squares Algorithm

$s(t) = 1 + h^T(t) P(t-1) h(t)$	: scaling factor
$P(t) = P(t-1) - P(t-1) h(t) s^{-1}(t) h^T(t) P(t-1)$	: covariance update
$K(t) = P(t) h(t)$	: Kalman gain
$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t) [z(t) - h^T(t) \hat{\theta}(t-1)]$	: parameter update

scaling factor in this thesis for reasons stated later. Initial conditions are usually taken as  $P(0) = \sigma^2 I$  and  $\hat{\theta}(0) = 0$ , where  $\sigma$  is a large integer. Clearly the parameter update in

Table 3.1 follows the general formula of (3.1), where the gain vector is  $K(t)$  and the prediction error (innovation) is given by  $\tilde{z}(t) = z(t) - h^T(t) \hat{\theta}(t-1)$ .

Investigating the RLS algorithm in Table 3.1, it is found that the updating of the covariance matrix may involve a subtraction of two almost equal matrices with very small magnitudes. This can easily result in poor numerical performance when implemented on digital computers with round off errors (Clarke & Gawthrop 1979). Problems occur most frequently due to the covariance matrix becoming ill-conditioned, or non-positive definite. In order to achieve more robust numerical performance, Bierman (1977) and Thornton & Bierman (1980) proposed the UD factorization algorithm. It is also discussed at length by Ljung & Söderström (1983). In Ljung (1985b) and Ljung (1985), comparison of the numerical properties of the RLS and the UD-factorization algorithms are presented.

### 3.2.2 RLS with UD Factorization

Bierman's UD factorization algorithm (Bierman 1977, Thornton & Bierman 1980) successfully overcomes the numerical problems described above. The UD factorization method uses a different technique for updating the covariance matrix than that presented in the RLS method (Table 3.1). Instead of directly updating the covariance matrix using equation (3.4), a  $UDU^T$  factored form of the covariance matrix,  $P(t) = U(t) D(t) U^T(t)$ , is used. At every time interval,  $U(t)$  and  $D(t)$  are updated, instead of directly updating  $P(t)$ . The UD factorization preserves the positive-definiteness of the covariance  $P(t)$  matrix, and thus better numerical performance is attained. A stepwise implementation of Bierman's UD factorization algorithm is summarized in Table 3.2, and a detailed derivation and discussion can be found in, e.g., Ljung &

Table 3.2: Bierman's UD Factorization Algorithm

$f = U^T(t-1) \varphi(t), g = D(t-1) f, \beta_0 = 1$	: innovation sequence
For $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j$	: scaling factor
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$	: loss function update
$\mu_j = -f_j / \beta_{j-1}, \nu_j = g_j$	
For $i=1$ to $j-1$ , do	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i \mu_j$	: covariance update
$\nu_i = \nu_i + U_{ij}(t-1) \nu_j$	
$\tilde{K}(t) = \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_d \end{pmatrix}, K(t) = \tilde{K}(t) / \beta_d$	: Kalman Gain update
$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t) [z(t) - h^T(t) \hat{\theta}(t-1)]$	: nth order parameter

Söderström (1983) or Thornton & Bierman (1980).

Clearly, the UD factorization algorithm is a variant of the RLS algorithm in Table 3.1 where the UD factorization technique is used to replace formula (3.4) for stable updating of the covariance matrix  $P(t)$ . Experiments shows that for digital computer implementations, to obtain the same numerical accuracy, the UD algorithm can use about half

the word length required by the RLS algorithm (Hägglund 1983, Thornton & Bierman 1980, Morris, Nazor & Wood 1989).

### 3.3 The Augmented UD Identification Algorithm

The AUDI structure in Chapter 2 permits a novel interpretation the Least-squares estimator (2.9). In this section, the implementation of the AUDI structure is discussed and the AUDI-LS algorithm is developed.

In the recursive implementation of AUDI, all that is required at each time interval is updating of the augmented covariance matrix since all the information on model parameters and loss functions is implicitly contained in the ACM. Expressed in terms of the ACM, the update equation is

$$C^{-1}(t) = C^{-1}(t-1) + \varphi(t) \varphi^T(t) \quad (3.5)$$

Obviously, (3.5) has the same form as (3.3), except that  $C(t)$  is defined differently than  $P(t)$ . Therefore, the UD factorization technique can be directly used for updating  $C(t)$ . Since  $C(t) = U(t) D(t) U^T(t)$ , the ACM updating formula (3.5) can then be rewritten as (Bierman 1977, Ljung & Söderström 1983)

$$U(t) D(t) U^T(t) = U(t-1) \left[ D(t-1) - \frac{g g^T}{\beta} \right] U^T(t-1) \quad (3.6)$$

where

$$\begin{aligned} f &= U^T(t-1) \varphi(t) \\ g &= D(t-1) f \\ \beta &= 1 + f^T g \end{aligned}$$

The updating of the ACM can therefore be accomplished by solving the above equations, at every time step. Following the derivation of Bierman's UD factorization algorithm, the final recursive AUDI algorithm is obtained as shown in Table 3.3.  $C(0) = U(0) D(0) U^T(0) = \sigma^2 I$  is used for algorithm initialization where  $\sigma$  is a large in-

Table 3.3: The Recursive AUDI Algorithm

$f = U^T(t-1) \varphi(t)$ , $g = D(t-1) f$ , $\beta_0 = 1$	: innovation sequence
for $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j$	: scaling factor
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$	: loss function update
$\mu_j = -f_j / \beta_{j-1}$ , $\nu_j = g_j$	
for $i=1$ to $j-1$ , do (skip for $j=1$ )	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i \mu_j$	: parameter update
$\nu_i = \nu_i + U_{ij}(t-1) \nu_j$	: Kalman gain matrix
$U = U(t)$ , $D = D^{-1}(t)$	: result matrices

tegr. This is equivalent to setting  $P(0) = \sigma^2 I$  and  $\theta(0) = 0$  in the basic RLS algorithm

(Table 3.1). Note that the AUDI algorithm (Table 3.3) omits the last two steps of Bierman's algorithm (Table 3.2), because essentially the last two steps in Bierman's method repeat previous steps for a higher order model. In the AUDI algorithm, this is done by including the current output data  $z(t)$  in the data vector (2.12) and thus increases the dimensions of variables such as  $f, g, U(t), D(t)$  by 1. The updated parameter vector  $\theta$  in the last step of Bierman's method is included in the last column of the  $U(t)$  matrix in the AUDI algorithm. This is explained more fully in next section.

Although the AUDI algorithm and Bierman's algorithm have a very similar form, their interpretation is quite different. Bierman's method is used as a stable way to perform the update of covariance matrix  $P(t)$  with formula (3.4). The  $U(t)$  and  $D(t)$  matrices are just temporary, intermediate matrices for calculation and do not have any physical significance. For parameter estimation using RLS, all the steps in Tables 3.1 and 3.2 must be carried out. However, in the AUDI algorithm, the  $U(t)$  and  $D(t)$  matrices explicitly contain all the information on parameter estimates and loss functions they are the purpose of identification rather than temporary matrices for computation. Thus updating the factored form of the ACM matrix using (3.6) is all that is required for recursive identification.

Some special features of the AUDI method can be summarized as follows

1. The most obvious and useful feature of the AUDI algorithm is that it simultaneously produces multiple models of  $n$  different orders at every time interval, with approximately the same amount of computation required for identifying a single  $n$ th order model with conventional instrumental variable algorithm. The most appropriate model can be selected by applying specific order-determining criteria that the loss functions provided in the loss function matrix.
2. Another significant advantage of the AUDI algorithm over conventional LS algorithm is that it is not sensitive to over-parameterization. The AUDI method is order recursive, from lower order to high order. Numerical problems are always associated with elements of the diagonal loss function matrix being too large or too small. Over-parameterized models lead to very small loss functions, which implies very small elements in the loss function matrix which may cause numerical problems. However, this does not affect the accuracy of the lower order models since the loss functions of lower order models are not close to zero. That is, unlike the conventional LS type algorithms, numerical problems due to over-parameterization do not destroy the entire augmented covariance matrix.
3. A simple matrix regularization technique such as that suggested by Ljung & Söderström (1983) can keep the augmented covariance matrix well-conditioned. The only cost is lower accuracy in the over-parameterized part of the models since only that part needs regularization. Matrix regularization can be easily done with the AUDI algorithm by applying an upper and a lower bounds to the elements of the loss function matrix. This requires very little extra computation and thus is always recommended (Stu & Fisher 1984).
4. A forgetting factor can be very easily introduced into the AUDI algorithm to track time-varying parameters. Given the forgetting factor  $\lambda(t)$ , the AUDI algorithm with forgetting factor is obtained by making the following changes to Table 3.3

$$\begin{aligned} \beta_0 &= 1.0 \Rightarrow \beta_0 = \lambda(t) \\ D_{jj}(t) &= D_{jj}(t-1) \beta_{j-1} / \beta_j \Rightarrow D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j / \lambda(t) \end{aligned}$$

where  $\Rightarrow$  means "change to". (Forgetting factors are discussed further in Chapter 7).

### 3.4 The Multiple Model Structure

The recursive ALM algorithm presented in Table 3.3 is quite implementation oriented. If the algorithm is rewritten using more compact matrix forms, the result is as in Table 3.4, where  $K$  is the Kalman gain matrix.

Table 3.4: The compact form of the recursive ALM algorithm

$f = U^T(t-1) \varphi(t)$ , $g = D(t-1) f$ , $\beta_0 = 1$	: innovation
for $j=1$ to $d$	
$\beta_j = \beta_{j-1} + f_j g_j$	: scaling factor
$K_{jj} = -g_j / \beta_j$	
for $i=1$ to $j-1$ , (skip $j=1$ )	
$U_{ij}(t) = U_{ij}(t-1) + K_{ij} f_j$	: parameter update
$K_{i,j+1} = K_{ij} + U_{ij}(t-1) K_{jj}$	: gain update
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$	: loss function update
$U = U(t)$ , $D = D^{-1}(t)$	: result matrices

Notice that the parameter matrix is updated by

$$U_{ij}(t) = U_{ij}(t-1) + K_{ij} f_j$$

which is in the same general formula as (3.1). This means that in the ALM algorithm, the parameter matrix  $U$  is updated at every time interval, instead of the parameter vector  $\hat{\theta}(t)$  in RLS (Table 3.1). The Kalman gain vector in RLS corresponds to the Kalman gain matrix in the ALM algorithm. Actually, the ALM algorithm simultaneously identifies the following  $d$  models

$$\begin{bmatrix} 1 \\ \hat{\alpha}_1^{(0)} \\ \hat{\theta}_1^{(1)} & 1 \\ \vdots & \vdots \\ \hat{\alpha}_1^{(n-1)} & \hat{\alpha}_2^{(n-1)} & \hat{\alpha}_3^{(n-1)} & \dots & 1 \\ \hat{\theta}_1^{(n)} & \hat{\theta}_2^{(n)} & \hat{\theta}_3^{(n)} & \dots & \hat{\theta}_{d-1}^{(n)} & 1 \end{bmatrix} \begin{bmatrix} -x(t-n) \\ u(t-n) \\ \vdots \\ -x(t-1) \\ u(t-1) \\ -x(t) \end{bmatrix} = 0 \quad (3.7)$$

The odd-numbered models use the past inputs and outputs to fit the current output and thus are defined as the *forward models*. For example, the 3rd model in (3.7) can be rewritten as

$$[\hat{\theta}_1^{(1)}, \hat{\theta}_2^{(1)}, 1] \begin{bmatrix} -x(t-n) \\ u(t-n) \\ -x(t-n+1) \end{bmatrix} = 0$$

or in a more explicit form as

$$x(t-n+1) + \hat{\theta}_1^{(1)} x(t-n) = \hat{\theta}_2^{(1)} u(t-n)$$

for stationary input/output, this is equivalent to

$$z(t) + \hat{\theta}_1^{(1)} z(t-1) = \hat{\theta}_2^{(1)} u(t-1)$$

This is clearly the first order model of our process (2.2).

The even-numbered models in (3.7) uses the past inputs and output to predict the future input. This is the inverse of conventional model definitions, and is thus called *the backward model*. For example, the 4th model in (3.7) can be rewritten as

$$[\hat{\alpha}_1^{(1)}, \hat{\alpha}_2^{(1)}, \hat{\alpha}_3^{(1)}, 1] \begin{bmatrix} -z(t-n) \\ u(t-n) \\ -z(t-n+1) \\ u(t-n+1) \end{bmatrix} = 0$$

which is equivalent to, under the condition of stationary input/output

$$u(t) + \hat{\alpha}_2^{(1)} u(t-1) = \hat{\alpha}_3^{(1)} z(t) + \hat{\alpha}_1^{(1)} z(t-1)$$

The backward model is not used in most applications and hence can be regarded simply as auxiliary information. However, under closed-loop, the backward model gives the dynamics of the controller.

By comparing the AUDI algorithm in Table 3.3 with RLS in Table 3.1, it is clear that the AUDI algorithm is an order-recursive method. The order recursion is shown in

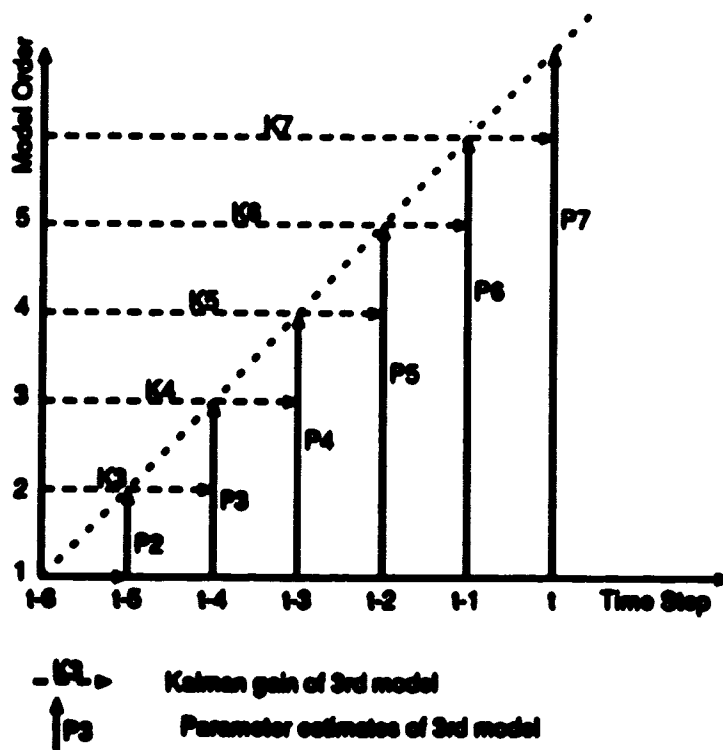


Figure 3.1: The order recursion of the AUDI algorithm

Figure 3.1 and can be explained as follows. The first model, i.e., the first row in (3.7), has a single parameter that is fixed to 1. Therefore, the parameter update is omitted

by including "skip for  $j=1$ " in the AUDI algorithm. Based on information of the first model, the Kalman gain (KG in Figure 3.1) for updating the second order model in (3.7) can then be calculated as in Table (3.4). The parameter estimates (PE in Figure 3.1) of the second model are obtained by using this Kalman gain to update the parameter matrix  $U$  as shown in Table (3.4). After the parameter estimates of the second model are obtained, the Kalman gain (KG) for updating the third model is calculated, then the parameter estimates (PE) of the third model is updated using the Kalman KG. This recursion carries on until the parameters of the last model are updated. This order recursion is done at every time interval, thus the AUDI algorithm is both time recursive and order-recursive. The recursion in Figure 3.1 also shows that the last two steps in Bierman's UD algorithm (Table 3.2) simply repeat previous steps at a higher order level and are thus redundant.

### 3.5 Comparison of AUDI and RLS

The AUDI algorithm is a simultaneous order and time recursive algorithm. At every time step, the order recursion applies to every model in the multiple model structure, from lower order model to the highest order model. However, for each order, the method used for parameter estimation is inherently a recursive least squares estimator. The information implicitly contained in the ACM matrix is made explicit through the order-recursive implementation. This is seen through the following comparison between the AUDI algorithm in Table 3.3 and the familiar RLS algorithm in Table 3.1.

1. The vector  $-f$  in AUDI algorithm (Table 3.3) is the innovation sequence of the parameter estimates in  $U(t)$ , *i.e.*,

$$-f = U^T(t-1) \varphi(t)$$

This is an analogous form of  $\hat{z}(t) = z(t) - h^T(t) \hat{\theta}(t-1)$  in the least squares algorithm (Table 3.1), but for all orders from 1 to  $n$ .

*Proof:* This is obvious from the definition of  $U(t-1)$  and  $\varphi(t)$  in the AUDI algorithm.

2. The variables  $\beta_j$  in the AUDI algorithm (Table 3.3) are called scaling factors, and are analogous to the variable  $s(t)$  in RLS (Table 3.1), with  $j \in [1, n]$ . They determine the convergence rate of the ACM or  $P(t)$  matrix.
3. The parameter updating in the AUDI algorithm (Table 3.3) is done by updating the matrix

$$U_{ij}(t) = U_{ij}(t-1) + v_i \mu_j U_{ij}(t-1) + \frac{v_i}{\beta_{j-1}} f_j$$

or

$$U(t) = U(t-1) + K(t) \hat{z}(t)$$

in the compact AUDI algorithm (Table 3.4), is in the same form as that in RLS, with  $v_i/\beta_j$  or  $K(t)$  being the Kalman gain.

4. In the AUDI algorithm, the updating of  $D(t)$  using

$$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1}/\beta_j$$



is the same as updating of the loss function  $J(t)$  in RLS, *i.e.*,

$$J(t) = J(t-1) + \tilde{z}(t) e(t)$$

where  $\tilde{z} = z(t) - h^T(t) \hat{\theta}(t-1)$  is the innovation (prediction error) sequence and  $e(t) = z(t) - h^T(t) \hat{\theta}(t)$  is the residual.

A comparison of the features of the AUDI algorithm and the RLS algorithm is summarized in Table 3.5 (some of the features summarized in Table 3.5 are not discussed until later).

### 3.6 Simulation Examples

Consider a process represented by the following SISO linear difference equation model

$$z(t) - 1.5z(t-1) + 0.7z(t-2) = u(t-1) + 0.5u(t-2) + v(t)$$

where  $z(t)$  and  $u(t)$  are the process output and input respectively;  $v(t)$  is white noise with zero-mean and variance  $\sigma^2 = 0.25$ . A random binary sequence (RBS) is used as the process input. Assume the maximum possible model order is 4, then the following augmented data vector is constructed

$$\varphi(t) = \begin{bmatrix} -z(t-4), u(t-4), -z(t-3), u(t-3), \\ -z(t-2), u(t-2), -z(t-1), u(t-1), -z(t) \end{bmatrix}^T$$

Assume the initial condition to be  $C(0) = U(0) D(0) U^T(0) = 10^3 I$ , which is equivalent to setting  $P(0) = 10^3 I$  and  $\hat{\theta}(0) = 0$  in the RLS algorithm. Using the recursive AUDI algorithm presented in Table 3.3, the converged parameter matrix is shown in Table 3.6. The loss function matrix is shown in Table 3.7. Note that the results are exactly the same as the results of the batch AUDI method in Chapter 2 (Tables 2.3 and 2.4).

The trajectories of the parameter estimates of the 2nd order model (which has the correct model order) are plotted in Figure 3.2. It is seen that the estimated parameters (solid line) rapidly converge to their true values (dashed line). The loss functions for different model orders are plotted in Figure 3.3. It is seen that significant decreases in the loss functions can be observed when the model order increases from 0 to 1 and from 1 to 2. However, a model higher than 2nd order does not reduce the loss function further. This indicates that correct the model order is 2. Criteria such as the AIC and the F-test give the same estimate of  $k=2$ .

Figure 3.4 gives the trajectories of the steady-state gains for different models. It is seen that the first order model does not provide a good gain estimate. When the model order is higher than the true process order, the steady-state gains are almost identical for different models, in other words, over-parameterized models can provide gain estimates and loss functions as good as the model with the correct order.

### 3.7 Conclusions

1. The AUDI algorithm is a reformulation of the least-squares estimator, but implemented in a more efficient manner.

Table 3.5: Comparison of AUDI and RLS

features	AUDI	RLS
model structure	multiple model structure	single model structure
interpretation	simple, clear, compact and informative	not so easy
order identification	simultaneous order/parameter identification	parameter estimation only
deadline estimation	simultaneous deadline/parameter identification	parameter estimation only
covariance update	positive-definiteness automatically guaranteed	sensitive to roundoff errors
numerical properties	excellent	sometimes unstable
matrix regularization	by applying upper and lower bound to $D(t)$	more difficult
computation	$\approx$ nth order RLS for single model	$\approx$ nth order AUDI for $n$ models
storage requirement	$\approx$ nth order RLS for single model	$\approx$ nth order AUDI for $n$ models

**Table 3.6: Parameter matrix  $U(t)$**

1	-0.0108	-0.8974	-0.0237	0.7019	0.0586	-0.0106	0.0023	0.0612
	1	0.9284	-0.0210	0.5068	0.1104	0.0167	0.0422	0.0570
		1	-0.0324	-1.5003	-0.1155	0.7145	0.0504	-0.1384
			1	0.9891	0.0746	0.5211	0.1296	0.0981
				1	0.0696	-1.5029	-0.1134	0.7991
					1	0.9873	0.0690	0.5244
						1	0.0785	-1.5036
							1	0.9893
								1

**Table 3.7: Loss function matrix  $D(t)$**

9362.8								
	496.9							
		1557.3						
			496.1					
				134.3				
					495.6			
						135.5		
							494.9	
								134.1

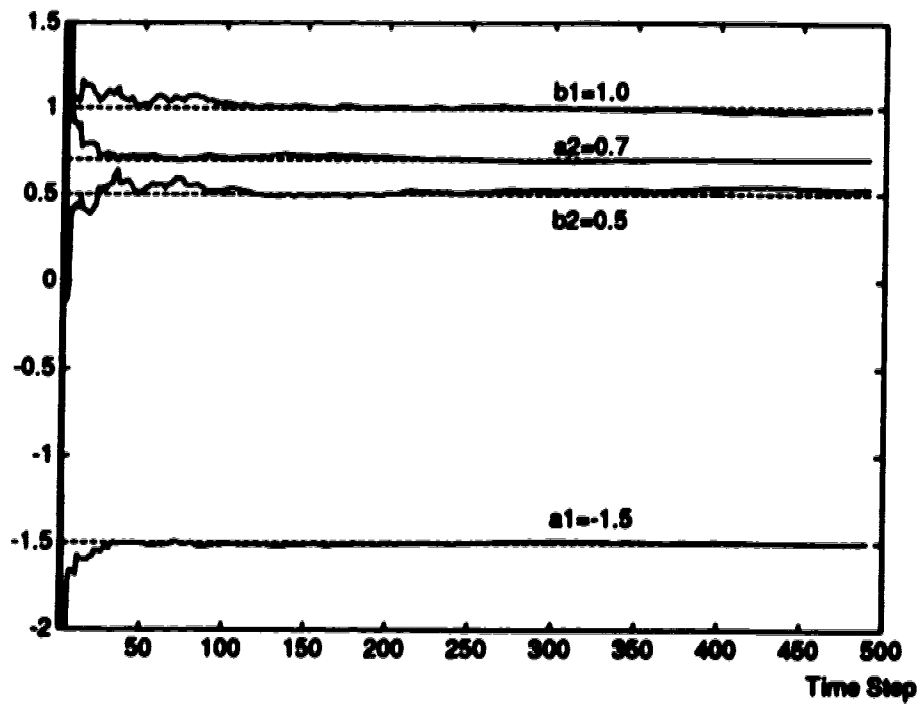


Figure 3.2: The parameter trajectory using recursive AUDI algorithm

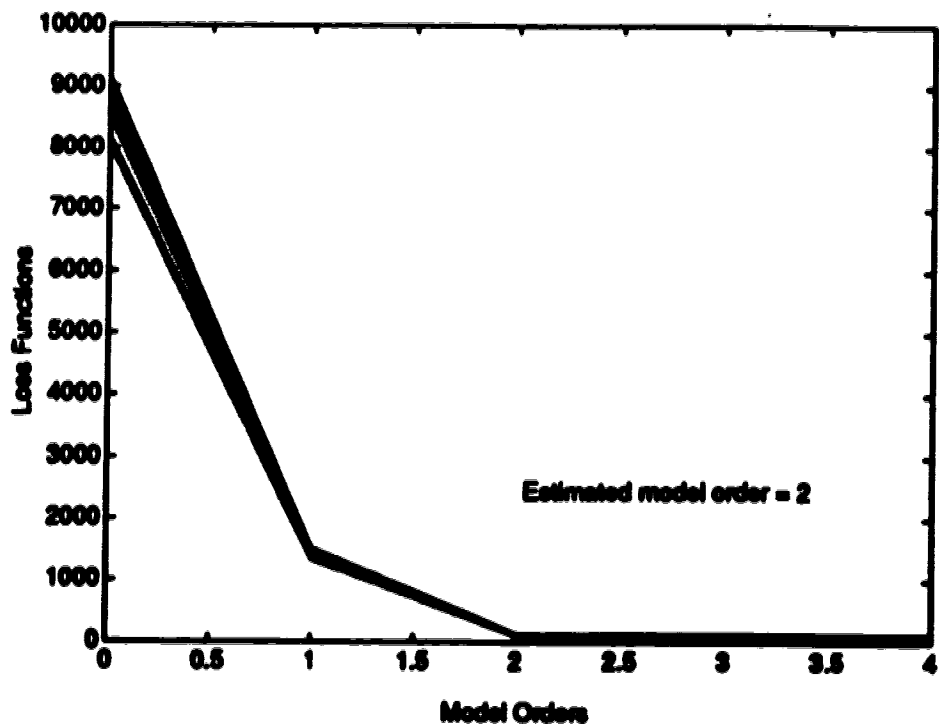
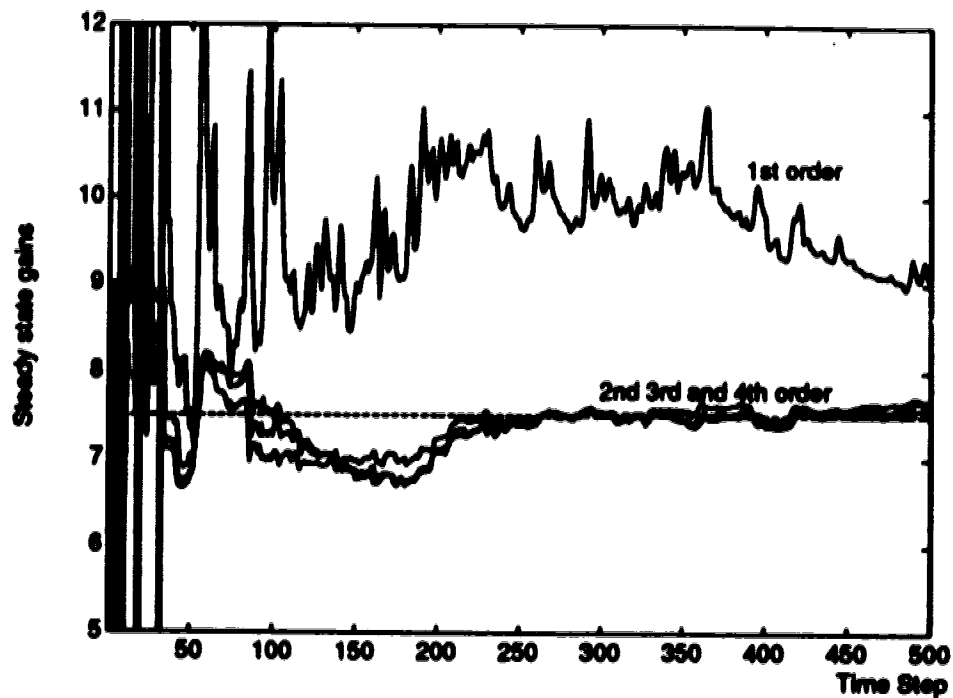


Figure 3.3: The loss functions using recursive AUDI algorithm



**Figure 3.4: The gain trajectory using recursive AUDI algorithm**

- 2. The AUDI algorithm provides simultaneous estimation of the parameters and loss functions for all model orders from 1 to  $n$  with a computational load equivalent to  $n$ th order RLS.**
- 3. The structure and implementation of the AUDI algorithm is more straightforward and easier to analyze than Bierman's UD algorithm.**
- 4. The AUDI algorithm, as shown in later chapters, provides a convenient and efficient basis for additional features such as an on/off criterion, estimation the noise variance or N/S ratio, and stability/convergence analysis.**

# Bibliography

- Bierman, G. J. (1977), *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York.
- Clarke, D. W. & Gawthrop, P. J. (1979), 'Self-tuning control', *IEE Proceedings, Part D*, **126**, 663 - 640.
- Clarke, D. W. & Mohtadi, C. (1987), 'Generalized predictive control - part I. the basic algorithm. part II. extensions and interpretations', *Automatica* **23**(2), 137 - 160.
- Cluett, W. R. (1986), *Stable Adaptive Control in the Presence of Unmodeled Dynamics*, PhD thesis, University of Alberta, Edmonton, Canada.
- Friedlander, B. (1983), 'Lattice implementations of some recursive parameter estimation algorithms', *International Journal of Control* **37**(4), 661 - 684.
- Goodwin, G. C. & Sin, K. S. (1984), *Adaptive Filtering Prediction and Control*, Prentice Hall.
- Hägglund, T. (1983), The problem of forgetting old data in recursive estimation, in 'Prepr. IFAC Workshop Adaptive System Control, Signal Processing', San Francisco. Paper SAC-6.
- Ljung, L. (1987), *System Identification: Theory for the User*, Prentice Hall, Englewood Cliffs, New Jersey.
- Ljung, L. & Söderström, T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass.
- Ljung, S. (1985), *Fast Algorithms for Integral Equations and Least-Squares Identification Problems*, PhD thesis, Linköping University, Sweden. Dissertation No. 93.
- Ljung, S. & Ljung, L. (1985), 'Error propagating properties of recursive least-squares adaptive algorithms', *Automatica* **21**(2), 157-167.
- Morris, A. J., Nasor, Y. & Wood, R. K. (1989), Single and multivariable application of self-tuning controllers, in C. J. Harris & S. A. Billings, eds, 'Self-Tuning and Adaptive Control: Theory and Applications', IEE Control Engineering Series 15.
- Niu, S. & Fisher, D. G. (1994), 'Multiple model least squares method'. Submitted to 1994 American Control Conference.
- Niu, S., Fisher, D. G. & Xiao, D. (1992), 'An augmented UD identification algorithm', *International Journal of Control* **56**(1), 193 - 211.

- Plackett, R. L. (1950), 'Some theorems in least-squares', *Biometrika* **37**, 149 - 157.
- Samson, C. (1982), 'A unified treatment of fast algorithms for identification', *International Journal of Control* **35**(5), 909 - 934.
- Shah, S. L. & Cluett, W. R. (1991), 'Recursive least-squares based estimation schemes for self-tuning control', *The Canadian Journal of Chemical Engineering* **69**, 89 - 96.
- Söderström, T. & Stoica, P. (1989), *System Identification*, Prentice Hall, Englewood Cliffs, New Jersey.
- Thornton, C. L. & Bierman, G. J. (1980), 'UDU<sup>T</sup> covariance factorization for Kalman filtering', *Control and Dynamic Systems*. New York: Academic Press.
- Tjokro, S. (1984), Derivation and evaluation of an adaptive PID controller, Master's thesis, University of Alberta, Edmonton, Canada.

## Chapter 4

# THE AUDI-ELS ALGORITHM

**T**he AUDI form of the Extended Least-squares (ELS) algorithm, or the AUDI-ELS algorithm, is an extension of the AUDI-LS algorithm presented in Chapter 3. The AUDI-LS algorithm, just like RLS, gives unbiased model parameter estimates only if the process noise is white. For the case of colored process noise, a modified algorithm, such as the ELS algorithm (or the instrumental variable algorithm discussed in next chapter) should be used. The AUDI-ELS algorithm has the same features as the AUDI-LS algorithm, *i.e.*, clear and compact structure, excellent numerical properties, simultaneous identification of model parameters and loss functions, but gives unbiased parameter estimates in the presence of colored noise. (Unfortunately, like ELS, it requires more computation and the performance is not always as good as RLS).

### 4.1 Introduction

The AUDI-LS algorithm developed in Chapter 3 is appropriate for model (2.2) where the process noise  $v(i)$  is zero-mean white noise. The AUDI-LS algorithm is easy to apply and also has many excellent properties, but it has a serious restriction for real implementations: the parameter estimates are consistent only under the condition that the process noise is zero-mean white noise. If the process noise is colored, the AUDI-LS method will give biased parameter estimates (Söderström & Stoica 1989). However, the AUDI-LS or LS algorithms can be modified in different ways to overcome this drawback. The extended least-squares method and the instrumental variable method are two commonly used modifications. The extended least-squares method (Panasika 1968, Young 1968, Ljung & Söderström 1983) introduces a pseudo-linear structure for estimation which results in unbiased estimates. It is actually an extension of the basic least-squares method (2.9) in Chapter 3. Since a pseudo-linear regression structure is used, the ELS method suffers from some additional numerical problems as well as the numerical problems inherent in the basic least-squares method (see Chapter 3).

---

<sup>1</sup>A version of this chapter has been published as: S. Niu, D. Xiao and D. Grant Fisher, 1999, A Recursive Algorithm For Simultaneous Identification of Model Order and Parameters, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.46, No.5, pp894-903.



## 4.2 The Pseudo-linear Regression Model

The following linear difference equation model

$$z(t) + a_1 z(t-1) + \dots + a_n z(t-n) = b_1 u(t-1) + \dots + b_n u(t-n) + e(t) \quad (4.1)$$

$$e(t) = v(t) + d_1 v(t-1) + \dots + d_n v(t-n) \quad (4.2)$$

or its polynomial form

$$A(q^{-1}) z(t) = B(q^{-1}) u(t-1) + D(q^{-1}) v(t) \quad (4.3)$$

is widely used in many implementations, where

$$A(q^{-1}) = 1 + a_1 q^{-1} + \dots + a_n q^{-n}$$

$$B(q^{-1}) = b_1 q^{-1} + \dots + b_n q^{-n}$$

$$D(q^{-1}) = 1 + d_1 q^{-1} + \dots + d_n q^{-n}$$

$u(t)$  and  $z(t)$  are process input and output respectively.  $e(t)$  is colored process noise which is represented by the moving-average of a zero-mean white noise sequence  $v(t)$  with variance  $\sigma_v^2$ . The parameters of the noise model (4.2), are  $d_i$ , with  $i \in [1, n]$ . This model is usually called ARMAX model, standing for AutoRegressive and Moving Average with an eXogenous signal, or CARMA model for Controlled AutoRegressive and Moving Average model. The ARIMAX (or CARIMA) model used in generalized predictive control (GPC) (Clarke & Mohtadi 1987)

$$A(q^{-1}) z(t) = B(q^{-1}) u(t-1) + \frac{D(q^{-1}) v(t)}{\Delta} \quad (4.4)$$

falls into this category if we rewrite it as

$$A(q^{-1}) \Delta z(t) = B(q^{-1}) \Delta u(t-1) + D(q^{-1}) v(t)$$

Obviously, model (4.1) can be written in the least-squares format as follows

$$z(t) = h^T(t) \theta_0 + v(t)$$

where  $v(t)$  is zero-mean white noise and the data and parameter vectors are defined as

$$h(t) = [-z(t-1), \dots, -z(t-n), u(t-1), \dots, u(t-n), v(t-1), \dots, v(t-n)] \quad (4.5)$$

$$\theta_0 = [a_1, \dots, a_n, b_1, \dots, b_n, d_1, \dots, d_n]^T \quad (4.6)$$

The least-squares estimator (2.9) can be used directly to obtain an unbiased estimate  $\hat{\theta}(t)$  of the parameter vector  $\theta_0$ . However, the problem is that the noise terms  $v(t-1), v(t-2), \dots, v(t-n)$  in the data vector are assumed to be white noise sequence and are un-measurable. To get around with this problem, the white noise terms are replaced by the estimated prediction errors or residuals. This leads to a pseudo-linear regression in which the prediction error or residual is dependent on the parameter estimates of the previous step. The method to implement this pseudo-linear regression is called the extended least-squares method which is a special form of the more general Pseudo-Linear Regression (PLR) method (Söderström & Stoica 1989). The ELS method is also called Furusaker's method (Furusaker 1968, Furusaker 1969), or the extended matrix method (Tahmon & Ben Den Boom 1973).

### 4.3 The Extended Least-Squares Estimator

The extended data vector in (4.5) is rewritten as

$$h(t) = [-z(t-1), \dots, -z(t-n), u(t-1), \dots, u(t-n), \hat{v}(t-1), \dots, \hat{v}(t-n)]^T \quad (4.7)$$

Using the basic least-squares algorithm (2.9), the unbiased parameter estimates of  $\theta_0$  can be obtained as

$$\hat{\theta}(t) = \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1} \sum_{j=1}^t h(j) z(j) \quad (4.8)$$

which is in exactly the same form as (2.9) except the data vector  $h(t)$  is constructed differently. The estimated noise term  $\hat{v}(t)$  in the data vector is replaced by the residual

$$\hat{v}(t) = e(t) \triangleq z(t) - h^T(t) \hat{\theta}(t) \quad (4.9)$$

Clearly  $\hat{v}(t)$  is a function of both time  $t$  and parameter estimates  $\hat{\theta}(t)$ , therefore it is no longer a linear problem, but rather, a pseudo-linear problem.

The recursive implementation of the extended least-squares algorithm, i.e., the Recursive Extended Least-Squares algorithm (RELS) takes a form very close to the RLS method (Table 3.1). Detailed derivation and discussion can be found in, for example, Solo (1979), or Ljung & Söderström (1983). Compared with RLS, the RELS algorithm identifies the noise model parameters  $d_i$  as well as the process model parameters  $a_i$  and  $b_i$  with  $i \in [1, n]$ . The complete algorithm is summarized in Table 4.1.

Table 4.1: The recursive extended least-squares algorithm

$h(t) = [-z(t-n), \hat{v}(t-n), u(t-n), \dots, -z(t-1), \hat{v}(t-1), u(t-1)]^T$	: data vector
$\theta(t) = \theta(t-1) + K(t) [z(t) - h^T(t) \hat{\theta}(t-1)]$	: parameter update
$S(t) = 1 + h^T(t) P(t-1) h(t)$	: scaling factor
$K(t) = P(t) h(t)$	: Kalman gain
$P(t) = P(t-1) - P(t-1) h(t) S^{-1}(t) h^T(t) P(t-1)$	: covariance update
$\hat{v}(t) = z(t) - h^T(t) \hat{\theta}(t)$	: noise estimate

The estimated parameters asymptotically converge to their true values under the (sufficient) condition that the filter

$$\frac{1}{D(q^{-1})} - \frac{1}{2}$$

is strictly positive real (SPR) and the input/output data are described by the model (4.1) (Ljung & Söderström 1983). Further discussions on the convergence properties can be found in, e.g., Ljung (1977), Moore (1980) or Solo (1979).

## 4.4 The AUDI Form of the ELS Algorithm

### 4.4.1 The AUDI structure for ELS

Similar to AUDI-LS, the AUDI structure can be developed for the ELS case to attain the multiple model structure and the excellent numerical performance of AUDI-LS.

Define the augmented data vector as

$$\varphi(t) = \begin{bmatrix} -z(t-n), \hat{v}(t-n), u(t-n), \dots, \\ -z(t-1), \hat{v}(t-1), u(t-1), -z(t) \end{bmatrix}^T \quad (4.10)$$

Note that the components of the regressor vector,  $h(t)$ , are arranged in triplets  $\{z, v, u\}$ , and the parameter vector  $\theta_0$  is also arranged in triplets  $\{a, d, b\}$ . In comparison with (4.7)

$$\varphi(t) = \begin{pmatrix} h(t) \\ -z(t) \end{pmatrix}$$

The augmented covariance matrix for the extended least-squares method can then be defined as follows

$$C(t) \triangleq \left[ \sum_{j=1}^t \varphi(j) \varphi^T(j) \right]_{(2n+1) \times (2n+1)}^{-1} \quad (4.11)$$

Note the special structure of the augmented data vector  $\varphi(t)$  and the augmented dimension of the augmented covariance matrix. Decomposing  $C(t)$  into the  $UDU^T$  factored form yields

$$C(t) = U(t) D(t) U^T(t) \quad (4.12)$$

where the parameter matrix  $U=U(t)$  has the form of

$$U = U(t) = \begin{bmatrix} 1 & \hat{\beta}_1^{(0)} & \hat{\alpha}_1^{(0)} & \hat{\theta}_1^{(1)} & \hat{\beta}_1^{(1)} & \hat{\alpha}_1^{(1)} & \dots & \hat{\beta}_1^{(n-1)} & \hat{\alpha}_1^{(n-1)} & \hat{\theta}_1^{(n)} \\ & 1 & \hat{\alpha}_2^{(0)} & \hat{\theta}_2^{(1)} & \hat{\beta}_2^{(1)} & \hat{\alpha}_2^{(1)} & \dots & \hat{\beta}_2^{(n-1)} & \hat{\alpha}_2^{(n-1)} & \hat{\theta}_2^{(n)} \\ & & 1 & \hat{\theta}_3^{(1)} & \hat{\beta}_3^{(1)} & \hat{\alpha}_3^{(1)} & \dots & \hat{\beta}_3^{(n-1)} & \hat{\alpha}_3^{(n-1)} & \hat{\theta}_3^{(n)} \\ & & & 1 & \hat{\beta}_4^{(1)} & \hat{\alpha}_4^{(1)} & \dots & \hat{\beta}_4^{(n-1)} & \hat{\alpha}_4^{(n-1)} & \hat{\theta}_4^{(n)} \\ & & & & 1 & \hat{\alpha}_5^{(1)} & \dots & \hat{\beta}_5^{(n-1)} & \hat{\alpha}_5^{(n-1)} & \hat{\theta}_5^{(n)} \\ & & & & & 1 & \dots & \hat{\beta}_6^{(n-1)} & \hat{\alpha}_6^{(n-1)} & \hat{\theta}_6^{(n)} \\ & & & & & & \ddots & \vdots & \vdots & \vdots \\ & & & & & & & 1 & \hat{\alpha}_{2n-2}^{(n-1)} & \hat{\theta}_{2n-1}^{(n)} \\ & & & & & & & & 1 & \hat{\theta}_{2n}^{(n)} \\ & & & & & & & & & 1 \end{bmatrix} \quad (4.13)$$

and the loss function matrix  $D=D^{-1}(t)$  has the diagonal form

$$D = \text{diag} \left[ J^{(0)}(t), I^{(0)}(t), L^{(0)}(t), \right. \\ \left. J^{(1)}(t), I^{(1)}(t), L^{(1)}(t), \dots, J^{(n-1)}(t), L^{(n-1)}(t), J^{(n)}(t) \right] \quad (4.14)$$

Again, the bracketed superscripts represent the model orders. The following remarks correspond to those for the AUDI-LS method in Chapter 3.

1. The parameter matrix  $U$  now contains three types of model parameters, namely,  $\hat{\theta}$ ,  $\hat{\alpha}$  and  $\hat{\beta}$ . The process model parameters, from first order ( $\hat{\theta}^{(1)}(t)$ ) to  $n$ th order ( $\hat{\theta}^{(n)}(t)$ ), are contained in the parameter matrix  $U$ . They have a form similar to (4.6). To be more specific, the parameter estimates of the  $i$ th order model are contained in the  $(3i + 1)$ st column of the parameter matrix, with  $i \in [1, n]$ , e.g., the  $n$ th order parameter estimates

$$\begin{aligned}\hat{\theta}^{(n)}(t) &= [\hat{\theta}_1^{(n)}, \hat{\theta}_2^{(n)}, \dots, \hat{\theta}_{3n}^{(n)}]^T \\ &= \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1} \sum_{j=1}^t h(j) z(j)\end{aligned}$$

are contained in the  $(3n + 1)$ st (last) column in the parameter matrix with dimension  $3n$ , which corresponds to (4.6).

2. The  $\hat{\alpha}$  parameters are the parameter estimates of the *backward models* (see Chapter 3). The  $\hat{\beta}$  parameters are estimates of yet another kind of model and will be discussed shortly.
3. The *loss function matrix*  $\mathcal{D}$  contains the corresponding loss functions of the above three types of model parameter estimates. The  $J$  elements are of most interest because they are the loss functions of the process models. For example,

$$J^{(n)}(t) = \sum_{j=1}^t [z(j) - h^T(j) \hat{\theta}(t)]^2 = \sum_{j=1}^t e^2(t)$$

is the loss function of the  $n$ th order model.  $J^{(i)}(t)$ , with  $i \in [1, n]$ , is the loss function of the  $i$ th order model and is contained in  $\mathcal{D}$  as the  $(3i + 1)$ st diagonal element.

4. The  $L$  and  $I$  elements in the loss function matrix are the loss functions for the other two kinds of models mentioned in Remark 2, and will be discussed later with the  $\hat{\alpha}$  and  $\hat{\beta}$  parameters.

#### 4.4.2 The Recursive ALDI Algorithm for ELS

The recursive ALDI algorithm that implements the ALDI-ELS structure is similar to the ALDI-LS algorithm, but with some added complexity. The differences are in the construction of the augmented data vector, the estimation of process noise and the interpretation of the parameter and loss function matrices. The derivation of the ALDI-ELS algorithm is almost exactly the same as that of the ALDI-LS except that the augmented data vector has a different structure and a higher dimension. The complete algorithm is summarized in Table 4.2.

Note that the estimation of the noise term  $\hat{\epsilon}(t)$  is done in a different and more efficient way than that in (4.9), i.e.,

$$\hat{\epsilon}(t) = -f_t / \beta_{t-1} \quad (4.15)$$

A brief proof is as follows. From Chapter 3, it is known that the  $f$  vector in the ALDI algorithms (Tables 3.3 and 4.2) is the prediction error vector. It is then not difficult to show, based on

$$f = U^T(t-1) \varphi(t)$$

Table 4.2: The Recursive AUDI Algorithm

$\varphi(t) = [-z(t-n), \hat{\theta}(t-n), u(t-n), \dots,$	
$\quad -z(t-1), \hat{\theta}(t-1), u(t-1), -z(t)]^T$	: data vector
$f = U^T(t-1)\varphi(t), \quad g = D(t-1)f, \quad \beta_0 = 1$	: innovation sequence
for $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j$	: scaling factor
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$	: loss function update
$\mu_j = -f_j / \beta_{j-1}, \quad \nu_j = g_j$	
for $i=1$ to $j-1$ , do	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i \mu_j$	: parameter update
$\nu_i = \nu_i + U_{ij}(t-1) \nu_j$	: Kalman gain matrix
$\hat{\theta}(t) = -f_d / \beta_{d-1}$	: noise estimation
$U = U(t), \quad D = D^{-1}(t)$	: result matrices

that

$$f_d = U_{\cdot d}^T(t) \varphi(t) = -[x(t) - h^T(t) \hat{\theta}(t-1)] = -\tilde{z}(t)$$

where  $U_{\cdot d}$  represents the last column of the parameter matrix and  $\tilde{z}(t)$  is the innovation sequence. In LS estimation, the residual  $\epsilon(t)$  and innovation  $\tilde{z}(t)$  have the following relationship (Ljung & Söderström 1983),

$$\epsilon(t) = \frac{\tilde{z}(t)}{1 + h^T(t) P(t-1) h(t)}$$

since  $\beta_{d-1} = 1 + h^T(t) P(t-1) h(t)$  (see Remark 2 in section 3.5),

$$\hat{\theta}(t) = \epsilon(t) = -f_d / \beta_{d-1}$$

This provides an efficient method to calculate the residual, or the estimated noise term.

#### 4.4.3 The simplified AUDI-ELS algorithm

Now let us investigate the  $\hat{\beta}$  parameters in the parameter matrix  $U$ . As described earlier, they are parameter estimates of a specific type of model. Rewriting the AUDI-ELS algorithm into a multiple model structure, it is found that the AUDI-ELS is actually identifying the following  $d$  ( $d \hat{=} 3n + 1$ ) models simultaneously

$$U^T(t) \varphi(t) = 0$$

or in an explicit form as shown in Table 4.3. The models related to the  $\hat{\beta}$  parameters are shown in Table 4.4. In least-squares type estimation, the converged estimates of  $\hat{\theta}(t)$  are white noise, i.e.,

$$E\{v(i) v(j)\} = \delta_{ij} = \begin{cases} 1 & \text{when } i=j, \\ 0 & \text{otherwise.} \end{cases}$$

In addition, from the process model (4.1), it follows that the noise term  $\hat{\theta}(t)$  is uncorrelated with the past inputs and outputs. This indicates that the estimates of all the

Table 4.3: multiple Model Structure of AUDI-ELS

$$\begin{bmatrix} 1 \\ \hat{\beta}_1^{(0)} \\ \hat{\alpha}_2^{(0)} \\ \hat{\theta}_3^{(1)} \\ \hat{\beta}_4^{(1)} \\ \hat{\alpha}_5^{(1)} \\ \vdots \\ \hat{\beta}_1^{(n-1)} \\ \hat{\alpha}_2^{(n-1)} \\ \hat{\theta}_3^{(n)} \end{bmatrix} \begin{bmatrix} 1 \\ \hat{\alpha}_2^{(0)} \\ \hat{\theta}_3^{(1)} \\ \hat{\beta}_4^{(1)} \\ \hat{\alpha}_5^{(1)} \\ \vdots \\ \hat{\beta}_2^{(n-1)} \\ \hat{\alpha}_3^{(n-1)} \\ \hat{\theta}_4^{(n)} \end{bmatrix} \begin{bmatrix} 1 \\ \hat{\theta}_3^{(1)} \\ \hat{\beta}_4^{(1)} \\ \hat{\alpha}_5^{(1)} \\ \vdots \\ \hat{\beta}_3^{(n-1)} \\ \hat{\alpha}_4^{(n-1)} \\ \hat{\theta}_5^{(n)} \end{bmatrix} \begin{bmatrix} 1 \\ \hat{\alpha}_5^{(1)} \\ \vdots \\ \hat{\beta}_4^{(n-1)} \\ \hat{\alpha}_5^{(n-1)} \\ \hat{\theta}_6^{(n)} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ -x(t-1) \\ \hat{\theta}(t-1) \\ u(t-1) \\ -z(t) \end{bmatrix} = 0$$

Table 4.4: Noise Models in AUDI-ELS

$$\begin{aligned} \hat{\theta}(t-n+1) + \hat{\beta}_2^{(1)} \hat{\theta}(t-n) - \hat{\beta}_1^{(1)} z(t-n) - \hat{\beta}_4^{(1)} z(t-n+1) - \hat{\beta}_3^{(1)} u(t-n) &= 0 \\ \hat{\theta}(t-1) + \hat{\beta}_2^{(n-1)} \hat{\theta}(t-n) + \dots + \hat{\beta}_{2n-4}^{(n-1)} \hat{\theta}(t-2) - \hat{\beta}_1^{(n-1)} z(t-n) + \hat{\beta}_3^{(n-1)} u(t-n) + \dots + \hat{\beta}_{2n-3}^{(n-1)} u(t-n) &= 0 \end{aligned}$$

$\hat{\beta}$  parameters should be zero. Correspondingly, all the loss functions for these models should be the sum of squares of the estimated noise, *i.e.*,

$$J^{(0)} \approx J^{(1)} \approx \dots \approx J^{(n)} = \sum_{j=1}^n \hat{e}^2(j)$$

They are not exactly equal to each other because they each have a time shift of one time step, see the data vector (4.10).

Eliminating the  $\hat{\beta}$ -related computations by substitution of the known (converged) values can effectively reduce the computational burden by almost 1/3. The resulting simplified AUDI-ELS algorithm is listed in Table 4.5. Note that it is the same as the AUDI-

Table 4.5: The Simplified AUDI-ELS Algorithm

$\varphi(t) = [-z(t-n), \hat{e}(t-n), u(t-n), \dots,$	
$\quad -z(t-1), \hat{e}(t-1), u(t-1), -z(t)]^T$	: data vector
$f = U^T(t-1)\varphi(t), g = D(t-1)f, \beta_0 = 1$	: innovation sequence
for $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j$	: scaling factor
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$	: loss function update
$\mu_j = -f_j / \beta_{j-1}, \nu_j = g_j$	
for $i=1$ to $j-1$ , if $\text{mod}(j,3) \neq 2$ , do	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i \mu_j$	: parameter update
$\nu_i = \nu_i + U_{ij}(t-1) \nu_j$	: Kalman gain matrix
$\hat{e}(t) = -f_d / \beta_{d-1}$	: noise estimation
$U = U(t), D = D^{-1}(t)$	: result matrices

ELS algorithm in Table 4.2 except that the computation of the innovation sequence and the updating of the parameter and gain matrices are different.

## 4.5 Simulation Examples

A simulation is carried out assuming that the process can be described by the following ARMAX model (Ljung & Söderström 1983)

$$\begin{aligned} z(t) &= 1.5z(t-1) + 0.7z(t-2) + u(t-1) + 0.5u(t-2) + e(t) \\ e(t) &= v(t) - v(t-1) + 0.2v(t-2) \end{aligned}$$

where  $u(t)$  and  $z(t)$  are process input and outputs,  $v(t)$  is zero-mean white noise with variance  $\sigma_v^2 = 0.25$ . The input signal is a random binary sequence with magnitude 1.0. The signal-to-noise ratio is about 6.7.

First the AUDI-LS algorithm is used to identify this process and the parameter trajectories are plotted in Figure 4.1. The converged parameter estimates are shown in Table 4.6 and the loss functions are included in Table 4.6 as the last row. Clearly the parameter estimates are biased from their true values.

Next, the AUDI-ELS algorithm is used under exactly the same condition as AUDI-LS. The identified parameter trajectories are shown in Figure 4.2, and the converged

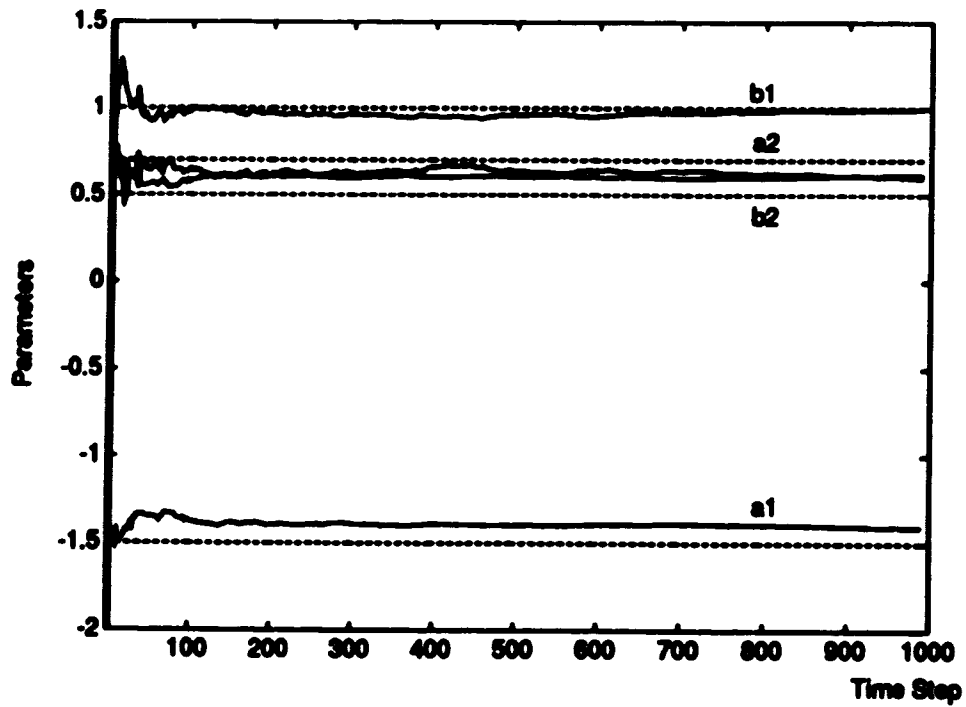


Figure 4.1: Parameter Trajectories Using AUDI-LS (Biased Results)

Table 4.6: Biased Parameters Obtained Using AUDI-LS

1	-0.0053	-0.8939	-0.0016	0.6147	0.0156	0.9085
	1	1.0053	-0.0083	0.8980	0.0028	0.3378
		1	-0.0048	-1.4083	-0.0429	-0.1364
			1	0.9989	0.0277	1.0880
				1	0.0285	-0.9120
					1	1.0087
						1
18408	991.48	2990.4	991.28	489.30	990.01	348.88



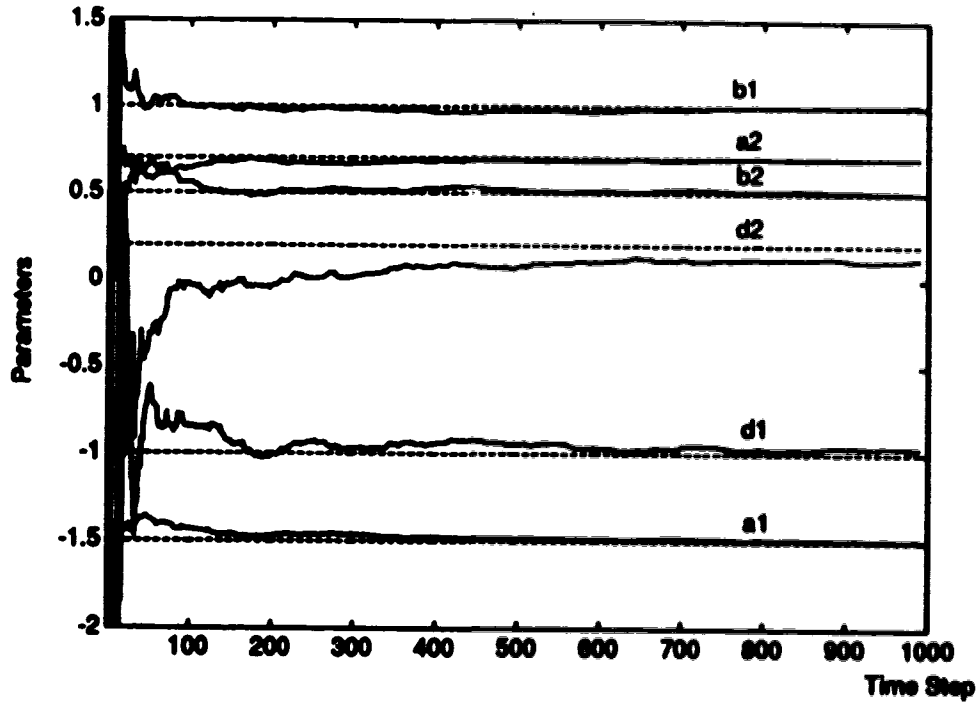


Figure 4.2: Parameter Trajectories Using AUDI-ELS (Unbiased Results)

parameter estimates are shown in Table 4.7, together with the loss functions (in the last row). It is seen that using the AUDI-ELS method, the estimated parameters converge to their true values. The parameter estimates of the noise model converge more slowly than those of process model, as can be expected (Ljung & Söderström 1983).

The relationship between the loss functions and model orders is plotted in Figure 4.3, from which it is seen that the loss function exhibits an abrupt change at the true model order and then becomes flat, just as in the AUDI-LS case. Therefore, the model order can be easily estimated to be 2.

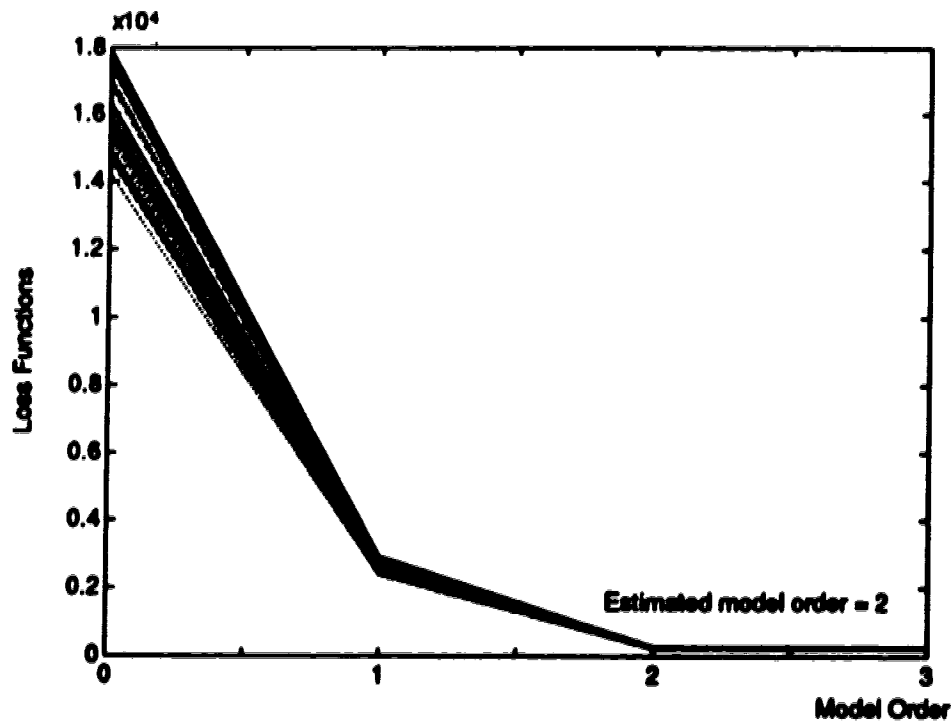
Now investigate the variance of noise. From Section 2.3, it is known that an unbiased estimate of the noise variance is given by

$$\hat{\sigma}_v^2 = \frac{J(t)}{t - 3n}$$

where  $n$  is the model order and  $J(t)$  is the loss function. Define

$$\hat{\sigma}_{ls} = \frac{1}{t - 3n} J_{ls}(t), \quad \hat{\sigma}_{els}(t) = \frac{1}{t - 3n} J_{els}(t)$$

where the subscripts "ls" and "els" stands for the AUDI-LS and AUDI-ELS method respectively. For recursive identification, we can plot the trajectories of  $\hat{\sigma}_{ls}$  and  $\hat{\sigma}_{els}$  versus time steps as shown in Figure 4.4, from which it is found that the estimated variance of the noise using AUDI-ELS converges to a value close to its true value (0.25). The lower limit of the estimated variance is its true value and can be achieved only when the estimated parameters (including the parameter estimates of the noise model) exactly equal to their true values. On the other hand, the AUDI-LS algorithm gives a larger estimate for the



**Figure 4.3: Loss Functions versus Model Orders Using AUDI-ELS**

noise variance, which indicates that AUDI-LS results in a poorer description of the real process than AUDI-ELS for models with the same order.

The simplified AUDI-ELS in Table 4.5 was then used for the same process under the same conditions, the estimated parameters are listed in Table 4.8, together with loss functions, while the parameter trajectories are plotted in Figure 4.5. From the table and figure, it can be seen that the simplified AUDI-ELS gives almost the same results as the full AUDI-ELS algorithm in Table 4.2, but with less computation.

## **4.6 Conclusions**

The AUDI form of the extended least-squares algorithm developed in this chapter is useful for identifying process with colored noise, or, more specifically, for identifying processes that can be represented by an ARMAX model. The AUDI-ELS has the same features as AUDI-LS, such as simultaneous identification of parameters and loss functions, excellent numerical properties and simple, compact structure.

Table 4.7: Converged Parameter Matrix By AUDI-ELS

1	0.0180	-0.0061	-0.9007	-0.0723	0.0027	0.6967	0.6109	0.0968	0.3919
1	-0.0425	-0.3746	-0.0284	0.0166	0.1238	0.0648	0.0648	0.0710	0.0903
1	1.0128	1	0.0850	0.0751	-0.0177	0.5016	0.4572	0.1138	0.2828
1	0.0850	1	0.0850	1	-0.0094	-1.4912	-1.3303	-0.2152	-0.1484
1	-0.0460	1	-0.0460	1	-0.9847	-0.8584	-0.8584	-0.1207	-0.4207
1	1.0067	1	1.0067	1	1	0.8960	0.8960	0.1440	1.0759
1	0.9004	1	0.9004	1	0.9004	1	0.9004	0.1448	-0.9337
1	0.1096	1	0.1096	1	0.1096	1	0.1096	1	-0.4189
1	1.0114	1	1.0114	1	1.0114	1	1.0114	1	1.0114
1	1	1	1	1	1	1	1	1	1
18408	243.89	968.04	2950.5	228.12	987.76	274.34	27.102	965.59	263.99

Table 4.8: Converged Parameter Matrix By Simplified AUDI-ELS

1	0	-0.0079	-0.9133	0	0.0082	0.6955	0	-0.0329	0.5305
1	-0.0848	-0.6097	0	-0.0122	0.0959	0	0	0.0372	0.0934
1	1.0118	1	0	-0.0247	0.5100	0	0	-0.0025	0.3877
1	0	-0.0181	-1.4827	0	-0.0181	-1.4827	0	0.0702	-0.4597
1	-0.1120	-0.9397	1	-0.1120	-0.9397	0	0	0.0294	-0.6103
1	1.0150	1	1.0150	1	1.0150	0	0	-0.0602	1.2935
1	0	-0.0539	-0.7205	0	-0.0539	-0.7205	0	-0.0539	-0.7205
1	-0.0990	-0.1918	1	-0.0990	-0.1918	1	1	-0.0990	-0.1918
1	1.0212	1	1.0212	1	1.0212	1	1.0212	1	1.0212
1	1	1	1	1	1	1	1	1	1
18408	254.45	968.54	2905.0	250.65	985.26	273.83	245.722	985.89	271.77

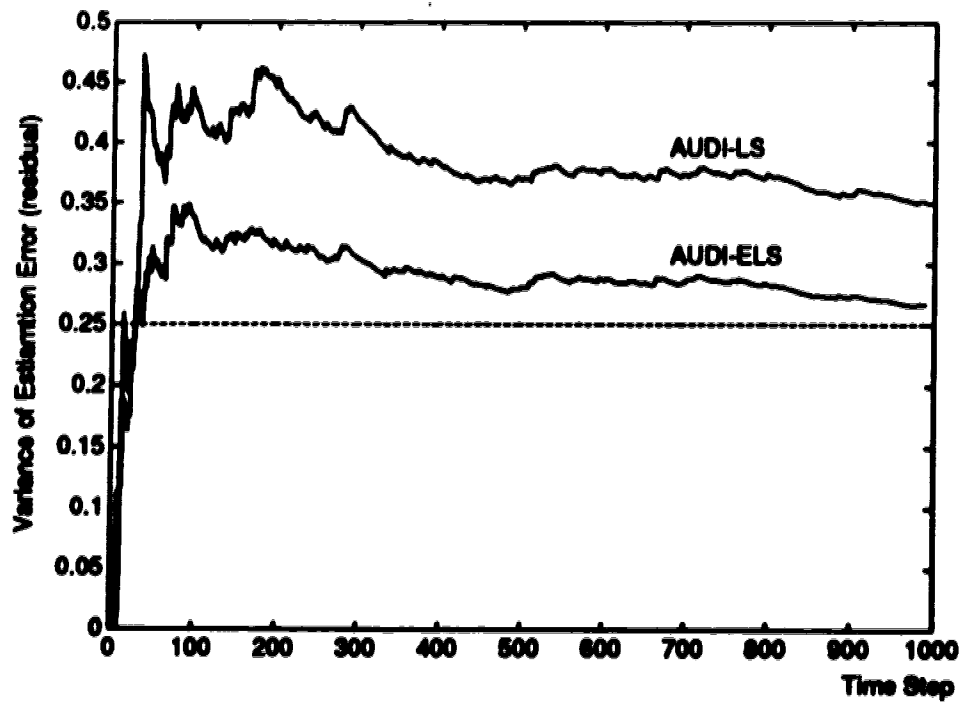


Figure 4.4: Trajectories of Estimated Variance of Process Noise

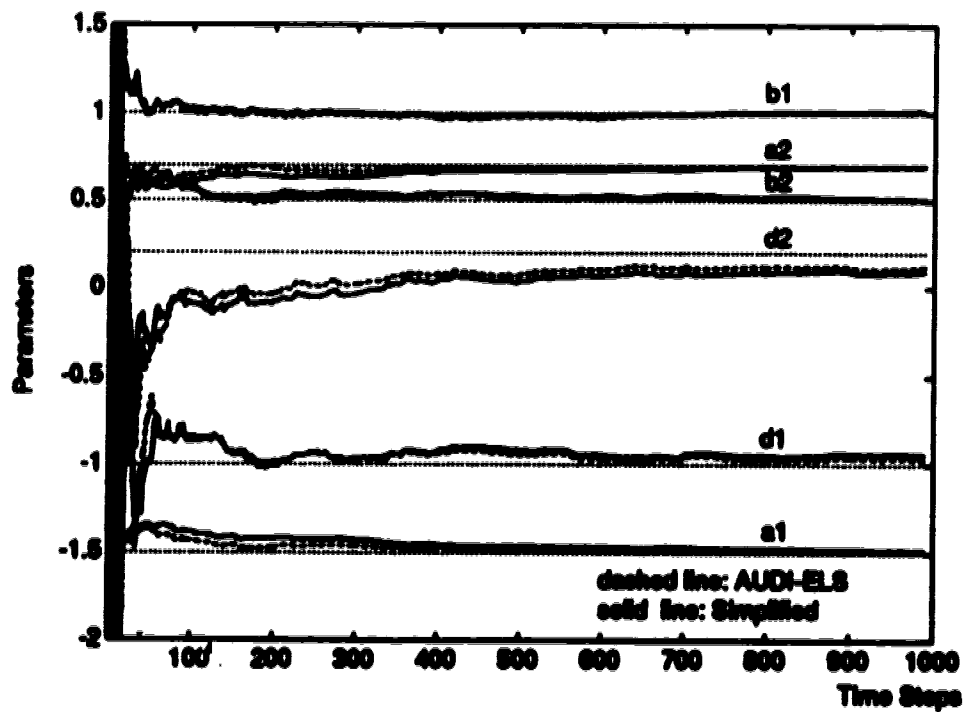


Figure 4.5: Parameter Trajectories using the Simplified AUDI-ELS

# Bibliography

- Clarke, D. W. & Mohtadi, C. (1987), 'Generalized predictive control – part I. the basic algorithm. part II. extensions and interpretations', *Automatica* 23(2), 137 – 160.
- Ljung, L. (1977), 'On positive real transfer functions and the convergence of some recursive schemes', *IEEE Transactions on Automatic Control* 22(4), 539 – 551.
- Ljung, L. & Söderström, T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass.
- Moore, J. B. & Ledwich, G. (1980), 'Multivariable adaptive parameter and state estimators with convergence analysis', *Journal of Australian Mathematics Society* 21, 176 – 197.
- Niu, S., Xiao, D. & Fisher, D. G. (1990), 'A recursive algorithm for simultaneous identification of model order and parameters', *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38(5), 884 – 886.
- Panuska, V. (1968), A stochastic approximation method for identification of linear systems using adaptive filtering, in 'Proceedings of the 1968 Joint Automatic Control Conference', University of Michigan, Ann Arbor, Michigan, pp. 1014 – 1021.
- Panuska, V. (1969), An adaptive recursive least-squares identification algorithm, in 'Proceedings of the IEEE Symposium on Adaptive Process Decision and Control'.
- Söderström, T. & Stoica, P. (1989), *System Identification*, Prentice Hall, Englewood Cliffs, New Jersey.
- Solo, V. (1979), 'The convergence of AML', *IEEE Transactions on Automatic Control* 24(6), 958 – 963.
- Takmon, J. L. & Ben Den Boom, A. J. W. (1973), On the estimation of transfer function parameters of process and noise dynamics using a single stage estimator, in 'IFAC Symposium on Identification and System Parameter Estimation', The Hague.
- Young, P. C. (1968), The use of linear regression and related procedures for the identification of dynamic processes, in 'Proc. 7th IEEE Symposium on Adaptive Processes', San Antonio, U.S.A., pp. 501 – 505.

## Chapter 5

# THE AUDI-IV ALGORITHM

A  $UDV^T$ -factored form of the Instrumental Variable (IV) identification algorithm is developed in this chapter for identification of systems with colored noise. The algorithm uses a  $UDV^T$ -factorization technique to reformulate the conventional instrumental variable method and thus simultaneously produces all the parameter estimates and loss functions for all models from order 1 to a user specified number  $n$ , with approximately the same computational effort as the conventional IV algorithm. The algorithm possesses a special compact structure which facilitates the interpretation and implementation of the IV-type algorithms. The  $UDV^T$  factorization technique used in this algorithm also results in better numerical performance than the conventional instrumental variable method.

### 5.1 Introduction

The instrumental variable algorithm was developed to solve the problem of parameter estimation for systems with colored noise where the commonly used recursive least-squares algorithm gives biased estimates (Söderström & Stoica 1983, Wong & Polak 1967, Young 1968, Young 1984). The IV algorithm has been proven to be an effective and efficient method and has found many successful applications (Wong & Polak 1967, Young 1976). However, like the recursive least-squares algorithm, a vital step in the recursive instrumental variable algorithm, namely, the updating of the augmented covariance matrix, may involve the subtraction of two almost equal matrices with very small magnitudes. In practical applications, especially when the system is over-parameterised, the algorithm can become very sensitive to computer round-off errors and noise, which can completely destroy the identification results (Niu, Fisher & Xiao 1992, Ljung & Söderström 1983, Shah & Chett 1991).

In this chapter, the main idea of the AUDI algorithms in Chapters 3 and 4 are extended to the instrumental variable method. Since the IV algorithm has a non-symmetrical augmented covariance matrix, the proposed algorithm is significantly different than the

---

<sup>1</sup>A version of this chapter has been published as: S. Niu and D. Xiao, 1992, The UD Factorization Form of the IV Algorithm, *Preprints of the 35th IFAC Symposium on Identification and System Parameter Estimation*, Beijing, 1990-1993. A revised version has been published as: S. Niu, D. Grant Fisher and D. Xiao, 1993, A Factored Form of the Instrumental Variable Algorithm, *International Journal of Adaptive Control and Signal Processing*, Vol. 7, No. 4, 361-372.

AUDI-LS algorithm in Chapter 3, although the basic idea is similar. A  $UDV^T$  decomposition technique, derived and extended from Bierman's well-known  $UDU^T$  factorization technique (Bierman 1977, Thornton & Bierman 1980), is used to reformulate the IV method. The end result is a factored instrumental variable algorithm (AUDI-IV) which possesses a multiple model structure similar to AUDI-LS. In comparison with the ordinary IV algorithm, the AUDI-IV method has the advantage of simultaneously identifying the model order plus parameters, and also possesses better numerical properties. This algorithm is especially useful in applications where the model order is not known *a priori* or where it changes with time, *e.g.*, in practical application of adaptive control. The multiple model structure makes the algorithm non-sensitive to overparameterization in the sense that the correct model is always available.

## 5.2 The Basic Instrumental Variable Algorithm

Since the conventional instrumental variable method is thoroughly covered in many books and papers such as Söderström & Stoica (1983), Söderström & Stoica (1989), Young (1976) and Young (1984), only a brief review is given here.

Assume that the system to be investigated is represented by the following SISO difference equation model:

$$z(t) + a_1 z(t-1) + \dots + a_n z(t-n) = b_1 u(t-1) + \dots + b_n u(t-n) + e(t) \quad (5.1)$$

or in a more compact form

$$z(t) = h^T(t) \theta_0 + e(t) \quad (5.2)$$

where

$$h(t) = [-z(t-1), \dots, -z(t-n), u(t-1), \dots, u(t-n)]^T \quad (5.3)$$

$$\theta_0 = [a_1, \dots, a_n, b_1, \dots, b_n]^T \quad (5.4)$$

$$(5.5)$$

$u(t)$  and  $z(t)$  are the observed input and output sequences;  $a_i$  and  $b_i$  ( $i=1,2,\dots,n$ ) are model parameters;  $e(t)$  is the process noise sequence which can be either white or colored. This model will be referred to as the *process model* in this chapter.

From the least-squares estimator (2.9), the least-squares estimate of the parameter vector  $\theta_0$  for model (5.1) is given by

$$\hat{\theta}(t) = \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1} \sum_{j=1}^t h(j) z(j) \quad (5.6)$$

Using the true system description (5.1), the above formula can be rewritten as

$$\begin{aligned} \hat{\theta}(t) &= \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1} \left[ \sum_{j=1}^t h(j) (h^T(j) \theta_0 + e(j)) \right] \\ &= \theta_0 + \left[ \sum_{j=1}^t h(j) h^T(j) \right]^{-1} \sum_{j=1}^t h(j) e(j) \end{aligned} \quad (5.7)$$

When  $e(t)$  is white noise sequence, the last term in (5.7) is

$$\lim_{t \rightarrow \infty} \sum_{j=1}^t h(j) e(j) = 0$$

Thus the LS estimate (5.6) is an asymptotically unbiased estimate of  $\theta_0$  if the noise is zero-mean and white. However, if  $e(t)$  is colored noise, then  $e(t)$  is a function of its past values. While  $h(t)$  is a function of the process output  $z(t)$  and thus implicitly depends on the past values of  $e(t)$  through (5.1), then it is clear that

$$\lim_{t \rightarrow \infty} \sum_{j=1}^t h(j) e(j) \neq 0$$

and therefore  $\hat{\theta}(t)$  from (5.7) is a biased estimate.

To obtain unbiased estimates under colored process noise, the instrumental variable method is proposed. The main idea is to introduce an instrumental data vector based on the process representation in Figure 5.1

$$\zeta(t) = [-z(t-1), -z(t-2), \dots, -z(t-n), u(t-1), u(t-2), \dots, u(t-n)]^T \quad (5.8)$$

the IV estimate of the model parameters is then given by

$$\hat{\theta}(t) = \left[ \sum_{j=1}^t \zeta(j) h^T(j) \right]^{-1} \sum_{j=1}^t \zeta(j) z(j) \quad (5.9)$$

$\hat{\theta}(t)$  is a consistent estimate of the model parameter  $\theta_0$  under the following conditions.

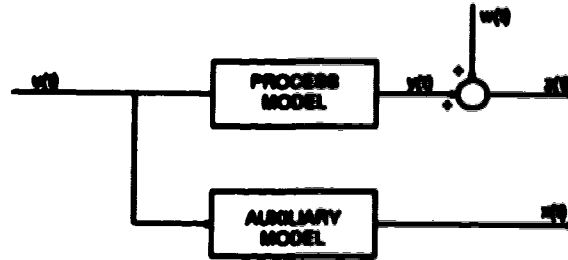


Figure 5.1: Construction of Instrumental Variables

$$\begin{cases} E\{\zeta(t) e(t)\} = 0 \\ \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{j=1}^t \zeta(j) h^T(j) \text{ has full rank.} \end{cases} \quad (5.10)$$

The instruments,  $s(t)$ , can be selected in many different ways. The most widely used instruments are calculated using the following auxiliary regression model (Söderström & Stoica 1981, Wong & Polak 1967, Young 1968)

$$s(t) = \zeta^T(t) \theta_0 \quad (5.11)$$

which is called the auxiliary model. In the batch least squares approach, the instrumental variable estimator (5.9) is applied in iterations until a presumed parameter accuracy



is attained, and  $\phi_0$  is usually chosen as the parameter estimates of the process model from the previous iteration. For recursive implementations,  $\phi_0$  is chosen as the latest available parameter estimates (usually from previous time interval) of the process model and is updated at every time step. Detailed derivations of the recursive IV algorithm can be found in, e.g., Söderström & Stoica (1989), and convergence analysis can be found in Ljung & Söderström (1983) or Söderström & Stoica (1981). The key equations for the recursive IV method are

$$\begin{cases} s(t) &= 1 + h^T(t) P(t-1) \zeta(t) \\ P(t) &= P(t-1) - P(t-1) h(t) s^{-1}(t) \zeta^T(t) P(t-1) \\ K(t) &= P(t) \zeta(t) \\ \hat{\theta}(t) &= \hat{\theta}(t-1) + K(t) [z(t) - h^T(t) \hat{\theta}(t-1)] \end{cases} \quad (5.12)$$

where  $P(t) = \left[ \sum_{j=1}^t \zeta(j) \zeta^T(j) \right]^{-1}$  is the covariance matrix and  $K(t)$  is the Kalman gain vector.

### 5.3 The AUDI Form of IV Algorithm

The instrumental variable estimator (5.9) provides a simple and elegant way of improving the basic least-squares estimator to produce unbiased parameter estimates for systems with colored noise. However, by introducing a  $UDV^T$  factorization technique, it is found that the IV estimator (5.9) can be reformulated in a much different and more efficient way. In addition, in the recursive implementation of the IV estimator in (5.12), after the parameters converge, the updating of the  $P(t)$  matrix may involve the subtraction of two matrices with very small magnitudes and thus can be very sensitive to round-off errors, especially when model (5.1) is over-parameterized. This may violate the second condition in (5.10) and thus lead to poor identification results or even divergence. The AUDI-IV method can preserve the second condition in (5.10) and thus can achieve improved performance than the conventional IV algorithm.

Assume the process model can be represented by model (5.1). And assume for simplicity, that the number of  $a$  parameters and  $b$  parameters in (5.1) are the same, then the augmented data vector can be defined as

$$\varphi(t) = [-s(t-n), u(t-n), \dots, -s(t-1), u(t-1), -s(t)]^T \quad (5.13)$$

Notice the arrangement of the elements in the data and parameter vector are different from traditional IV methods in order to generate the AUDI structure. The elements in (5.13) are grouped as  $\{s(\cdot), u(\cdot)\}$  pairs and the current output variable  $s(t)$  is appended to the data vector. The dimension of the data vector is  $d \triangleq 2n + 1$ .

By defining the corresponding augmented instrumental variable vector as

$$\eta(t) = [-s(t-n), u(t-n), \dots, -s(t-1), u(t-1), -s(t)]^T \quad (5.14)$$

the augmented covariance matrix can then be defined as

$$C(t) = \left[ \sum_{j=1}^t \eta(j) \eta^T(j) \right]_{d \times d}^{-1} \quad (5.15)$$

which is analogous to the  $P(t)$  matrix in (5.12) but with an augmented dimension and a different arrangement of its elements.

By decomposing the augmented covariance matrix into a  $UDV^T$  form (see Appendix A), it is found that

$$C(t) = U(t) D(t) V^T(t) \quad (5.16)$$

where  $U=U(t)$  is the parameter matrix with an unit-upper-triangular form

$$U = U(t) = \begin{bmatrix} 1 & \# & \hat{\theta}_1^{(1)}(t) & \# & \hat{\theta}_1^{(2)}(t) & \dots & \hat{\theta}_1^{(n-1)}(t) & \# & \hat{\theta}_1^{(n)}(t) \\ & 1 & \hat{\theta}_2^{(1)}(t) & \# & \hat{\theta}_2^{(2)}(t) & \dots & \hat{\theta}_2^{(n-1)}(t) & \# & \hat{\theta}_2^{(n)}(t) \\ & & 1 & \# & \hat{\theta}_3^{(2)}(t) & \dots & \hat{\theta}_3^{(n-1)}(t) & \# & \hat{\theta}_3^{(n)}(t) \\ & & & 1 & \hat{\theta}_4^{(2)}(t) & \dots & \hat{\theta}_4^{(n-1)}(t) & \# & \hat{\theta}_4^{(n)}(t) \\ & & & & 1 & \dots & \hat{\theta}_5^{(n-1)}(t) & \# & \hat{\theta}_5^{(n)}(t) \\ & & & & & \ddots & \vdots & \vdots & \vdots \\ & & & & & & 1 & \# & \hat{\theta}_{\frac{n-1}{2}}^{(n)}(t) \\ & 0 & & & & & & 1 & \hat{\theta}_{\frac{n-1}{2}}^{(n)}(t) \\ & & & & & & & & 1 \end{bmatrix} \quad (5.17)$$

The '#' represents elements that are not of interest in this chapter and the bracketed superscripts represent model orders. The columns of  $\hat{\theta}$  elements are the parameter estimates of model (5.1) as discussed below.

Matrix  $V=V(t)$  is defined as the *auxiliary parameter matrix* and has a structure similar to (5.17) and differs only by the presence of  $\hat{\theta}$  instead of  $\hat{\theta}$  in the odd numbered columns.

$$V = V(t) = \begin{bmatrix} 1 & \# & \hat{\theta}_1^{(1)}(t) & \# & \hat{\theta}_1^{(2)}(t) & \dots & \hat{\theta}_1^{(n-1)}(t) & \# & \hat{\theta}_1^{(n)}(t) \\ & 1 & \hat{\theta}_2^{(1)}(t) & \# & \hat{\theta}_2^{(2)}(t) & \dots & \hat{\theta}_2^{(n-1)}(t) & \# & \hat{\theta}_2^{(n)}(t) \\ & & 1 & \# & \hat{\theta}_3^{(2)}(t) & \dots & \hat{\theta}_3^{(n-1)}(t) & \# & \hat{\theta}_3^{(n)}(t) \\ & & & 1 & \hat{\theta}_4^{(2)}(t) & \dots & \hat{\theta}_4^{(n-1)}(t) & \# & \hat{\theta}_4^{(n)}(t) \\ & & & & 1 & \dots & \hat{\theta}_5^{(n-1)}(t) & \# & \hat{\theta}_5^{(n)}(t) \\ & & & & & \ddots & \vdots & \vdots & \vdots \\ & & & & & & 1 & \# & \hat{\theta}_{\frac{n-1}{2}}^{(n)}(t) \\ & 0 & & & & & & 1 & \hat{\theta}_{\frac{n-1}{2}}^{(n)}(t) \\ & & & & & & & & 1 \end{bmatrix} \quad (5.18)$$

The loss function matrix  $D=D^{-1}(t)$  is a diagonal matrix with the form

$$\begin{aligned} D &= [D(t)]^{-1} \\ &= \text{diag} \left[ J^{(0)}(t), L^{(0)}(t), \dots, J^{(n-1)}(t), L^{(n-1)}(t), J^{(n)}(t) \right] \end{aligned} \quad (5.19)$$

The multiple model structure that results from this decomposition is similar to that in AUDI-LS and can be described as follows.

1. The vectors,  $\hat{\theta}^{(1)}(t), \dots, \hat{\theta}^{(n)}(t)$ , which are above the diagonal of the odd-numbered columns in  $U$ , have dimensions of  $2, 4, \dots, 2n$ . That is,  $\hat{\theta}^{(i)}(t)$  has dimension  $2i$

with  $i \in [1, n]$ , and

$$\begin{aligned}\hat{\theta}^{(1)}(t) &= [\hat{\theta}_1^{(1)}(t), \hat{\theta}_2^{(1)}(t)]^T \\ &\vdots \\ \hat{\theta}^{(i)}(t) &= [\hat{\theta}_1^{(i)}(t), \hat{\theta}_2^{(i)}(t), \dots, \hat{\theta}_{2i}^{(i)}(t)]^T \\ &\vdots \\ \hat{\theta}^{(n)}(t) &= [\hat{\theta}_1^{(n)}(t), \hat{\theta}_2^{(n)}(t), \dots, \hat{\theta}_{2n}^{(n)}(t)]^T\end{aligned}$$

2. Each vector mentioned above, is the instrumental variable estimate of a particular model. That is,  $\hat{\theta}^{(1)}(t)$  is the IV estimate of the first order model of process (5.1) at time step  $t$ ;  $\hat{\theta}^{(i)}(t)$  is the IV estimate of the  $i^{\text{th}}$  order model. Specifically,  $\hat{\theta}^{(n)}(t)$  is the  $n^{\text{th}}$  order IV estimate, which is exactly the same as that given by (5.9), with a format defined in (5.4). See Appendix A for a proof.
3. The elements  $J^{(1)}(t), \dots, J^{(n-1)}(t), J^{(n)}(t)$  in the diagonal  $\mathcal{D}$  matrix are all scalars. They are the *generalized loss functions* corresponding to the models mentioned in Remark 2 and have the form of

$$J^{(n)}(t) = \sum_{j=1}^i \bar{z}(j) \bar{z}(j) \quad (5.20)$$

where

$$\begin{aligned}\bar{z}(j) &= z(j) - \hat{z}(j) = z(j) - \zeta^T(j) \hat{\theta}(t) \\ \bar{z}(j) &= z(j) - \hat{z}(j) = z(j) - h^T(j) \hat{\theta}(t)\end{aligned}$$

**Proof** see Appendix A.

Since the generalized loss functions of all models are available from (5.19), the model order can be determined using some order-determining criteria such as those suggested by Young (1980) .

4. All the other elements in the  $\mathcal{U}$  and  $\mathcal{D}$  matrices are intermediate variables and are not of interest in this chapter. The  $\mathcal{V}$  matrix, which turns out to be the parameter matrix of the auxiliary model, is discussed below.

To summarize the above discussion: after the input/output data is obtained, the augmented covariance matrix is formulated according to (5.15) and decomposed according to equation (5.16) into three matrices  $\mathcal{U}$ ,  $\mathcal{D}$  and  $\mathcal{V}$ . The odd-numbered columns of the  $\mathcal{U}$  matrix contain the IV estimates for all process models from order 1 to  $n$ , while the odd-numbered diagonal elements in  $\mathcal{D}$  matrix contain all the corresponding generalized loss functions. This  $\mathcal{U}\mathcal{D}\mathcal{V}^T$ -factored form is referred to as a multiple model structure because all the models are produced simultaneously.

The  $\mathcal{U}\mathcal{D}\mathcal{V}^T$  batch decomposition of the  $C(t)$  matrix is usually by using LU or LDU decomposition on the augmented information matrix similar to those in Chapter 3, that is

$$S(t) = \sum_{j=1}^i \varphi(j) \varphi^T(j) = L(t) D(t) U(t) \Rightarrow \begin{cases} \mathcal{U} = L^{-T}(t) \\ \mathcal{D} = D(t) \\ \mathcal{V} = U^{-1}(t) \end{cases} \quad (5.21)$$

or

$$S(t) = \sum_{j=1}^t \varphi(j) \varphi^T(j) = L(t) U(t) \Rightarrow \begin{cases} U = L^{-T}(t) \\ D = \text{diag} U(t) \\ V = D \cdot U^{-1}(t) \end{cases} \quad (5.22)$$

The batch implementations are very straightforward and therefore are not presented here.

## 5.4 Recursive Implementation of AUDI-IV

The AUDI form of the the instrumental variable method can also be implemented in a recursive manner analogous to recursive AUDI-LS.

### 5.4.1 The Recursive AUDI-IV Algorithm

Since the augmented covariance matrix  $C(t)$  in (5.15) is non-symmetrical, Bierman's UD factorization algorithm can no longer be used. Instead, a  $UDV^T$  decomposition technique, which is a modified version of Bierman's algorithm, is derived to decompose the non-symmetrical matrix. The updating formula is as follows

$$C^{-1}(t) = C^{-1}(t-1) + \eta(t) \varphi^T(t) \quad (5.23)$$

or

$$[U(t) D(t) V^T(t)]^{-1} = [U(t-1) D(t-1) V^T(t-1)]^{-1} + \eta(t) \varphi^T(t) \quad (5.24)$$

Since the three matrices  $U$ ,  $D$ , and  $V$  contains all the information about all the parameter estimates of the process model, parameter estimates of the auxiliary model and all the corresponding generalized loss functions, formulae (5.23) or (5.24) is the only calculation that needs to be carried out during recursive implementation of the AUDI-IV algorithm. The Kalman gain calculation and the explicit parameter update steps in the conventional RIV algorithm (5.12) are no longer necessary. This simplifies the interpretation and implementation of the recursive instrumental variable estimator. The complete recursive AUDI-IV algorithm is summarized in Table 5.1. A more detailed derivation of the recursive AUDI-IV algorithm is included in Appendix B. The derivation of Bierman's UD factorization algorithm, which is the basis of the  $UDV^T$  decomposition, can be found in Ljung & Söderström (1983) or Thornton & Bierman (1980).

The recursive AUDI-IV algorithm is mathematically equivalent to the conventional RIV algorithm. However, because of the different formulation, the AUDI-IV algorithm is numerically better than the conventional RIV algorithm when implemented on digital computers with finite word length. It has similar properties as those pertaining to AUDI-LS. See Chapter 3 for these properties.

### 5.4.2 Efficient Implementation of AUDI-IV

Examination of the AUDI-IV algorithm in Table 5.1 shows that the parameter matrix  $V$  for the auxiliary model is used only for the calculation of the auxiliary variables. If the computation related to this matrix could be avoided, the total computational effort would then be reduced considerably and a more efficient implementation of the AUDI-IV algorithm would be obtained.

Table 5.1: The Recursive AUDI-IV Algorithm

$\varphi(t) = [-z(t-n), u(t-n), \dots, -z(t-1), u(t-1), -z(t)]^T$	: data vector
$\eta(t) = [-z(t-n), u(t-n), \dots, -z(t-1), u(t-1), -z(t)]^T$	: instruments
$f = U^T(t-1) \varphi(t), f^* = V^T(t-1) \eta(t)$	: innovations
$g = D(t-1) f, g^* = D(t-1) f^*, \beta_0 = 1.0$	
for $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j^*$	: scaling factor
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$	: loss functions
$\mu_j = -f_j / \beta_{j-1}, \mu_j^* = -f_j^* / \beta_{j-1}$	
$\nu_j = g_j, \nu_j^* = g_j^*$	
for $i=1$ to $j-1$ , do	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i^* \mu_j$	: process model
$V_{ij}(t) = V_{ij}(t-1) + \nu_i \mu_j^*$	: auxiliary model
$\nu_i = \nu_i + U_{ij}(t-1) \nu_j$	: gain vector
$\nu_i^* = \nu_i^* + V_{ij}(t-1) \nu_j^*$	: gain vector
$U = U(t), D = D^{-1}(t), V = V(t)$	: result matrices

Recall that the instruments  $z(t)$  in the instrumental variable vector  $\eta(t)$  (5.14) are generated using the auxiliary model (5.11) where  $\zeta(t)$  is the IV vector (before augmentation) and  $\theta_0$  is the parameter vector of the auxiliary model. As shown in Ljung & Söderström (1983), the most widely used instrumental vector is the one generated using the following filter

$$z(t) = \zeta^T(t) \hat{\theta}(t-1) \quad (5.25)$$

where  $\hat{\theta}(t-1)$  is the parameter estimates of the process model from the last time interval. As shown in section 5.3 and Appendix A, the AUDI-IV algorithm simultaneously generates the parameter estimates of both the process model (5.1) and the auxiliary model (5.11). The parameter estimates of the auxiliary model are contained in the matrix  $V$ . Since the true parameters of the auxiliary model are, by definition in (5.25), the latest available parameter estimates  $\hat{\theta}(t-1)$  of the process model, then the estimated parameters of the auxiliary model  $\hat{\theta}(t)$  are given by

$$\hat{\theta}(t) = \hat{\theta}(t-1)$$

That is, the elements in the  $V$  matrix are estimates of the parameters contained in matrix  $U$ , which are used as the parameters of the auxiliary model. Obviously, the estimates of the auxiliary model parameters in  $V(t)$  are not necessary since the parameters of the auxiliary model are already known to be  $\hat{\theta}(t-1)$ , from (5.25). Therefore, we can directly set the estimate  $\hat{\theta}(t)$  to its true value  $\hat{\theta}(t-1)$  i.e.,

$$V(t) = U(t-1) \quad (5.26)$$

All the computation related to the updating of  $V$  is thereby eliminated, and the computational effort required at every time interval is reduced considerably. Removal of the computation related to  $V$  from the recursive AUDI-IV algorithm in Table 5.1 leads to the more efficient AUDI-IV algorithm summarized as in Table 5.2.

Table 5.2: The Efficient AUDI-IV Algorithm

$\varphi(t) = [-x(t-n), u(t-n), \dots, -x(t-1), u(t-1), -x(t)]^T$	: data vector
$\eta(t) = [-x(t-n), u(t-n), \dots, -x(t-1), u(t-1), -x(t)]^T$	: instruments
$f = U^T(t-1) \varphi(t), f^* = V^T(t-1) \eta(t)$	: innovations
$g^* = D(t-1) f^*, \beta_0 = 1.0$	
for $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j^*$	: scaling factor
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$	: loss functions
$\mu_j = -f_j / \beta_{j-1}, \nu_j^* = g_j^*$	
for $i=1$ to $j-1$ , do	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i^* \mu_j$	: parameter update
$\nu_i^* = \nu_i^* + U_{ij}(t-1) \nu_j^*$	
$U = U(t), D = D^{-1}(t)$	: result matrices

Because of the better numerical performance, multiple model structure, easy interpretation and implementation, the AUDI-IV algorithms (Tables 5.1 and 5.2) are recommended as replacement for the conventional RIV algorithm in practical applications.

## 5.5 Simulation Examples

The following SISO difference equation model is used as the representation of the simulated process

$$\begin{cases} x(t) - 1.5x(t-1) + 0.7x(t-2) = u(t-1) + 0.5u(t-2) + e(t) \\ e(t) = v(t) - v(t-1) + 0.2v(t-2) \end{cases}$$

Where,  $u(t)$  and  $x(t)$  are the observed input and output of the model respectively. The noise sequence  $e(t)$  is a second-order moving average of the zero-mean white noise sequence  $v(t)$  with variance  $\sigma_v^2 = 0.25$ . Assume that the maximum possible order of this "unknown" process is 3.

First, the AUDI-LS algorithm is used to identify this process. As expected, the parameter estimates are biased. For example, the trajectory of the  $a_1$  parameter is plotted in Figure 5.2, from which it is clear that there is quite a significant bias from its true value  $-1.5$ . This confirms that for the case of colored noise, the AUDI-LS and conventional RLS algorithms produce biased estimates.

Now, the conventional RIV, the recursive AUDI-IV algorithm in Table 5.1, and the efficient AUDI-IV algorithm in Table 5.2 are used to identify the same process. Unbiased estimates are obtained from all these algorithms. The trajectories of the  $a_1$  parameter are shown in Figure 5.2. The simulation is carried out in MATLAB (Matlab 1990), which uses double precision for calculations and thus no numerical problems were encountered. However, other experiments by the authors and related experiments by Hågglund (1993) and Ljung (1986) shows that, for single-precision simulation, UD factorization based algorithms have better numerical performance than the conventional RIV algorithms.

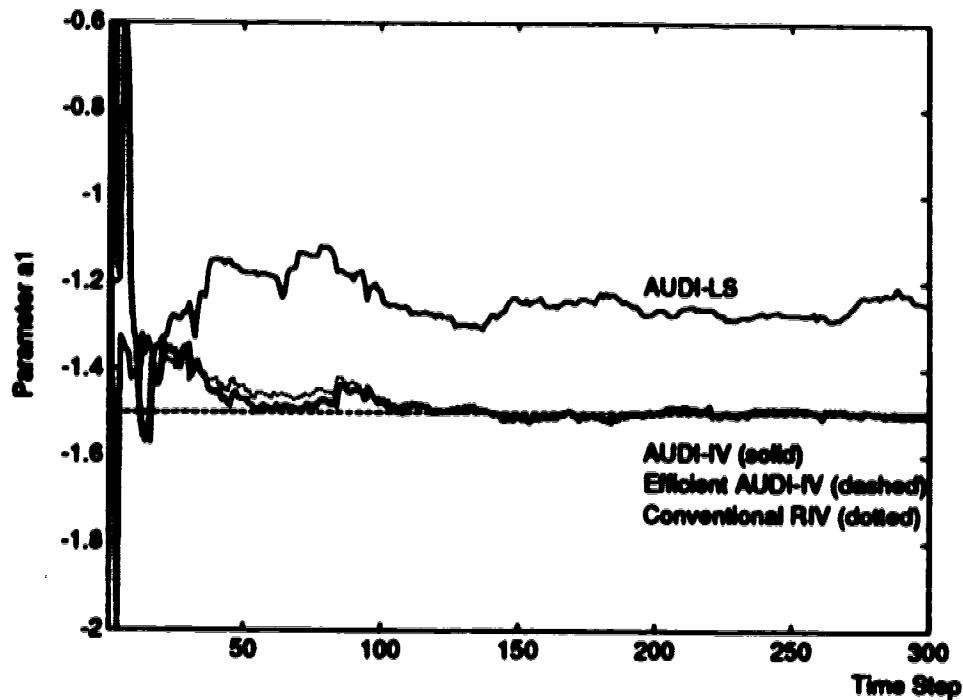


Figure 5.2: Parameter trajectories using different IV algorithms

The converged parameter matrices, using the recursive AUDI-IV method (Table 5.1) and the efficient AUDI-IV method (Table 5.2) are shown in Table 5.3 and Table 5.4 respectively. They have exactly the same structure as in equation (5.17) and (5.18), which means that the 3rd columns in Tables 5.3 and 5.4 are the parameter estimates of the 1st order process models; the 5th columns are the parameter estimates of the 2nd order models and the 7th columns are the parameter estimates of the 3rd order models and so on. Comparison of Table 5.3 and Table 5.4 shows that the converged parameters using the AUDI-IV algorithm and the efficient AUDI-IV algorithm are quite close. See also Figure 5.2. The parameter matrix of the auxiliary model, using the recursive AUDI-IV algorithm in Table 5.1, is shown in Table 5.5, and is found to be quite close to the parameter matrix in Table 5.3. This experimentally confirms equation (5.26).

The loss functions are provided in the loss function matrix  $\mathcal{D}$ , which is

$$\begin{aligned} \mathcal{D} &= D^{-1}(t) \\ &= \text{diag} \left[ \boxed{4040.9} \quad 292.6 \quad \boxed{675.3} \quad 292.6 \quad \boxed{11.2} \quad 291.0 \quad \boxed{6.21} \right] \end{aligned}$$

Remember that the 3rd element is the loss function of the 1st order model, the 5th element is the loss function of the 2nd order model and the last element is the loss function of the 3rd order model. The loss function values decrease rapidly as the model order increases from 0 to 1 and from 1 to 2, but tends to be flat after order 2. This suggests an order of 2. However, since the generalized loss function defined in (5.20) is a special form of the conventional loss function, further investigation is being conducted for a suitable criteria for order determination with them.

**Table 5.3: Parameter Matrix in the Recursive AUDI-IV Example**

$$\begin{bmatrix} 1 & 0.0086 & -0.8891 & 0.0276 & 0.7110 & -0.2036 & 0.2537 \\ & 1 & 0.9074 & -0.0036 & 0.4700 & -0.0258 & 0.3181 \\ & & 1 & -0.0170 & -1.5016 & 0.4345 & 0.1360 \\ & & & 1 & 0.9865 & -0.2733 & 0.8896 \\ & & & & 1 & -0.2716 & -1.0867 \\ & & & & & 1 & 0.9996 \\ & & & & & & 1 \end{bmatrix}$$

**Table 5.4: Parameter Matrix in the Efficient AUDI-IV Example**

$$\begin{bmatrix} 1 & 0.0088 & -0.8875 & 0.0299 & 0.7131 & -0.0763 & 0.3881 \\ & 1 & 0.9041 & -0.0076 & 0.4673 & 0.0483 & 0.4051 \\ & & 1 & -0.0196 & -1.5020 & 0.1552 & -0.1407 \\ & & & 1 & 0.9866 & -0.0725 & 1.0544 \\ & & & & 1 & -0.0810 & -0.9050 \\ & & & & & 1 & 1.0033 \\ & & & & & & 1 \end{bmatrix}$$

**Table 5.5: Auxiliary Parameter Matrix in the Recursive AUDI-IV Example**

$$\begin{bmatrix} 1 & 0.0166 & -0.8861 & 0.0320 & 0.6932 & 0.0411 & 0.2902 \\ & 1 & 0.9862 & -0.0071 & 0.4141 & 0.1747 & 0.3802 \\ & & 1 & -0.0125 & -1.4782 & -0.1016 & 0.0156 \\ & & & 1 & 1.0494 & 0.1158 & 0.9512 \\ & & & & 1 & 0.1091 & -0.9807 \\ & & & & & 1 & 1.0580 \\ & & & & & & 1 \end{bmatrix}$$



## 5.6 Conclusion

In this chapter, a  $UDV^T$  factored form of the standard IV algorithm is derived which simultaneously identifies all the parameters and corresponding loss functions of  $n$  process models from order 1 to a user-specified maximum order  $n$ , for processes with colored noise. The algorithm has a clear and compact structure which facilitate interpretations and implementation while at the same time possesses better numerical properties than conventional IV algorithms. This algorithm is especially suitable for the identification of processes with unknown model structure and unknown noise statistics.

# Bibliography

- Bierman, G. J. (1977), *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York.
- Ljung, L. & Söderström, T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass.
- Ljung, S. (1985), Fast Algorithms for Integral Equations and Least-Squares Identification Problems, PhD thesis, Linköping University, Sweden. Dissertation No. 93.
- Matlab (1989), *386-MATLAB for 80386 MS-DOS Personal Computer: Users's Guide*, The Math Works, Inc.
- Niu, S. & Xiao, D. (1988), The UD factorization form of IV algorithm, in 'Prepr. 8Th IFAC Symposium on Identification and System Parameter Estimation', Beijing, China, pp. 1060 - 1063.
- Niu, S., Fisher, D. G. & Xiao, D. (1992), 'An augmented UD identification algorithm', *International Journal of Control* 59(1), 193 - 211.
- Niu, S., Fisher, D. G. & Xiao, D. (1993), 'A factored form of the Instrumental Variable identification algorithm', *International Journal of Adaptive Control and Signal Processing* 7(4), 261-273.
- Shah, S. L. & Chueti, W. R. (1991), 'Recursive least-squares based estimation schemes for self-tuning control', *The Canadian Journal of Chemical Engineering* 69, 89 - 96.
- Söderström, T. & Stoica, P. (1981), 'Comparison of some instrumental variable methods - consistency and accuracy aspects', *Automatica* 17(1), 101-115.
- Söderström, T. & Stoica, P. (1983), *Instrumental Variable Methods for System Identification*, Springer-Verlag, New York.
- Thornton, C. L. & Bierman, G. J. (1980), 'UDU<sup>T</sup> covariance factorization for Kalman filtering', *Control and Dynamic Systems*. New York: Academic Press.
- Wong, K. Y. & Polak, E. (1967), 'Identification of linear discrete time systems using the instrumental variable approach', *IEEE Transactions Automatic Control* 12(6), 707 - 718.
- Young, P. C. (1968), The use of linear regression and related procedures for the identification of dynamic processes, in 'Proc. 7th IEEE Symposium on Adaptive Processes', San Antonio, U.S.A., pp. 501 - 505.

- Young, P. C. (1976), 'Some observations on instrumental variable methods of time series analysis', *International Journal of Control* 23(5), 593 - 612.
- Young, P. C. (1984), *Recursive Estimation and Time Series Analysis*, Springer-Verlag, New York.
- Young, P., Jakeman, A. & McMurtrie, R. (1980), 'An instrumental variable method for model order identification', *Automatica* 16(3), 281 - 194.

## Chapter 6

# IDENTIFICATION OF MULTIVARIABLE SYSTEMS

A recursive identification algorithm for multivariable systems is developed by extending the SISO augmented UD identification algorithm in Chapter 3 to the multivariable case. The resulting MIMO-AUDI algorithm has a simple and compact structure and provides simultaneous identification of model structure and parameters, with excellent numerical performance. A multivariable input/output difference equation model is used as the model representation for identification, and it is shown how the corresponding (unique) canonical state-space representation can also be generated by a straightforward transformation. The algorithm is easier to interpret and more straightforward to implement than the corresponding RLS approaches.

### 6.1 Introduction

Structure identification and parameter estimation of multivariable systems is a very important but rather difficult issue in system identification. Depending on the choice for the structure of the investigated system, the multivariable system can be represented in more than one way (Eldem & Yildizbayrak 1988). It is often assumed that there is enough *a priori* knowledge about the system structure so that only parameter estimation is required (Elliott & Wolovich 1982, Goodwin, Ramadge & Caines 1980, Morne 1981, Dion & Lamare 1984). However, the determination of a suitable structure is a critical step in system identification and is equal in importance to parameterization. Pioneering work on structure identification can be found in Ho & Kalman (1966) and Tether (1970). In Ackermann & Bucy (1971) a canonical realization procedure is developed using an input-output description. Guidorzi (1978) gives an invariant, canonical structure for linear multivariable system representation, which directly and uniquely links the state-space model with the corresponding input-output difference equation representation.

---

<sup>1</sup>A version of this chapter has been published as: S. Niu and D. Grant Fisher, 1991, MIMO System Identification Using An Augmented UD Identification Algorithm, *Proceedings 1991 American Control Conference*, Vol. 1, 689 - 703, Boston. A revised version has also been accepted for publication as: S. Niu and D. Grant Fisher, Simultaneous Structure Identification and Parameter Estimation of Multivariable Systems, *International Journal of Control*.

The augmented UD identification algorithms developed in previous chapters can simultaneously identify the model order and parameters of SISO process, with excellent numerical performance and high computational efficiency. The special structure of the AUDI algorithm also makes interpretation and implementation easier than the conventional RLS algorithm (Niu et al. 1992). In this chapter, the AUDI algorithm is extended to the multivariable case, and modified to perform structure identification and parameter estimation of multivariable systems simultaneously and recursively. The algorithm uses an input-output difference equation as its model, but it is shown that the corresponding (unique) canonical state space model can also be easily generated using a straightforward transformation based on the work of Guidorzi (1975, 1981).

## 6.2 Model Representations for MIMO Systems

Multivariable systems can be represented by state-space models, input-output difference equations, transfer function matrices or Markov parameters. A state space model is frequently used because there is strong mathematical and theoretical support for state-space applications. In practice, however, an input-output difference model is much easier to identify because the data available for identification in practical applications are the actual input-output data sequences. Guidorzi (1975) provides a link between state space canonical models and input-output difference equation models that can be used to transform one form to the other. These transformations are reviewed in this section and the MIMO AUDI algorithm is developed in Section 6.3. The example in section 6.4 illustrates how to identify an input-output model and/or state-space model. For readers who are interested in input-output model identification only, they can skip this section and go directly to Section 6.3.

### 6.2.1 State-Space Representation

Linear multivariable systems, if observable, can always be represented by the following state-space form,

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{s}(t) = \mathbf{C}\mathbf{x}(t) \end{cases} \quad (6.1)$$

where,  $\mathbf{x}(t)$  is an  $n$ -dimensional state-space variable vector;  $\mathbf{u}(t)$  and  $\mathbf{s}(t)$  are  $r$ -dimensional input and  $m$ -dimensional output variables respectively;  $\mathbf{A}$ ,  $\mathbf{B}$  and  $\mathbf{C}$  are coefficient matrices with the following structures (Guidorzi 1975).

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1n} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{n1} & \mathbf{A}_{n2} & \cdots & \mathbf{A}_{nn} \end{bmatrix} \quad (6.2)$$

$\mathbf{A}_{ii}$ , ( $i=1,2,\dots,m$ ), have dimension  $(\nu_i \times \nu_i)$  and  $\mathbf{A}_{ij}$ , ( $i,j=1,2,\dots,m$ ), have dimension  $(\nu_i \times \nu_j)$ , with the following forms

$$\mathbf{A}_{ii} = \begin{bmatrix} 0 & & \\ \vdots & \mathbf{I}_{\nu_i-1} & \\ 0 & & \end{bmatrix} \quad (6.3)$$

$$\mathbf{A}_{ij} = \begin{bmatrix} a_{ij,1} & \cdots & a_{ij,\nu_j} \end{bmatrix}$$

$$A_{ij} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ a_{ij,1} & \cdots & a_{ij,\nu_i} & 0 & \cdots & 0 \end{bmatrix} \quad (6.4)$$

where  $a_{ij,k}$  indicates element  $k$  of the last row of matrix  $A_{ij}$ . The  $B$  and  $C$  matrix are given as follows

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & \cdots & b_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nr} \end{bmatrix} \quad (6.5)$$

$$C = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \quad (6.6)$$

$\uparrow \qquad \qquad \qquad \uparrow \qquad \qquad \qquad \uparrow$   
 $1 \qquad \qquad \qquad (\nu_1 + 1) \qquad \qquad \qquad (\nu_1 + \cdots + \nu_{m-1} + 1)$

Any  $r$ -input  $m$ -output linear multivariable observable system is equivalent to the above Luenberger canonical structure (Luenberger 1967). The integers  $\nu_1, \nu_2, \dots, \nu_m$  are called the structural indices, because they completely and uniquely define the structure of the model.  $\nu_i, \nu_{ij}$  are a set of invariant quantities (Guidorzi 1975). The structure indices are related to the dimension of the state vector in the following way

$$\sum_{i=1}^m \nu_i = n$$

$$\nu_{ij} \leq \begin{cases} \nu_i + 1 & j < i \\ \nu_i & j \geq i \end{cases}$$

### 6.2.2 Input-Output Difference Equation Model

An input-output difference equation description which is structurally and parametrically equivalent to (6.1) can be defined as

$$P(z) z(t) = Q(z) u(t) \quad (6.7)$$

where  $P(z)$  and  $Q(z)$  are polynomial matrices,

$$P(z) = \begin{bmatrix} p_{11}(z) & p_{12}(z) & \cdots & p_{1m}(z) \\ p_{21}(z) & p_{22}(z) & \cdots & p_{2m}(z) \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1}(z) & p_{m2}(z) & \cdots & p_{mm}(z) \end{bmatrix}_{m \times m} \quad (6.8)$$

$$Q(z) = \begin{bmatrix} q_{11}(z) & q_{12}(z) & \cdots & q_{1r}(z) \\ q_{21}(z) & q_{22}(z) & \cdots & q_{2r}(z) \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1}(z) & q_{m2}(z) & \cdots & q_{mr}(z) \end{bmatrix}_{m \times r} \quad (6.9)$$

and  $z$  is the unitary forward shift operator. The elements of  $P(z)$  and  $Q(z)$  are polynomials with the following structures

$$\begin{cases} p_{ii}(z) = z^{\nu_i} - a_{ii,\nu_i} z^{\nu_i-1} - \dots - a_{ii,2} z - a_{ii,1} \\ p_{ij}(z) = a_{ij,\nu_{ij}} z^{\nu_{ij}-1} - \dots - a_{ij,2} z - a_{ij,1}, \quad (i \neq j) \\ q_{ij}(z) = \beta_{(\nu_1+\nu_2+\dots+\nu_i),j} z^{\nu_i-1} + \beta_{(\nu_1+\dots+\nu_{i-1}+2),j} z + \beta_{(\nu_1+\dots+\nu_{i-1}+1),j} \end{cases} \quad (6.10)$$

The degree of each polynomial is given by

$$\begin{cases} \deg\{p_{ii}(z)\} > \deg\{p_{ij}(z)\}, & j > i \\ \deg\{p_{ii}(z)\} \geq \deg\{p_{ij}(z)\}, & j < i \\ \deg\{p_{ii}(z)\} > \deg\{p_{ji}(z)\}, & j \neq i \\ \deg\{p_{ii}(z)\} > \deg\{q_{ij}(z)\} \end{cases} \quad (6.11)$$

This means that for a given row, the degree of the diagonal element in (6.8) is always higher than the degrees of the elements to the right of them, and not lower than the degrees of the elements to the left of them. This characteristic leads to the special structure of  $A_0$  in (6.16) of section 3 which is essential for the implementation of the augmented UD identification structure.

### 6.2.3 Transformations

The canonical state-space representation in (6.1) and the difference equation representation in (6.7) are equivalent and can be uniquely converted from one form to another (Guidorzi 1975). From (6.10), it is clear that the coefficients of the  $p_{ij}$ , ( $i, j=1, 2, \dots, m$ ) are directly related to the elements of the  $A$  matrix in (6.3) and (6.4). The coefficients of  $q_{ij}$  in (6.10) are defined by

$$B = M B = \begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1r} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{n1} & \beta_{n2} & \dots & \beta_{nr} \end{bmatrix} \quad (6.12)$$

where the matrix  $M$  is given by

$$\begin{bmatrix} -a_{11,2} & -a_{11,3} & \dots & -a_{11,\nu_1} & 1 & \dots & -a_{1m,2} & \dots & \dots & -a_{1m,\nu_{1m}} & 0 \\ -a_{11,3} & -a_{11,4} & \dots & 1 & & \dots & -a_{1m,3} & \dots & \dots & 0 & \\ \vdots & & & & & \dots & \vdots & & & & \\ \vdots & & & & & \dots & -a_{1m,\nu_{1m}} & & & & \\ -a_{11,\nu_1} & 1 & & & & \dots & 0 & & & & \\ 1 & & & & & \dots & 0 & & & & \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline -a_{m1,2} & \dots & \dots & -a_{m1,\nu_{m1}} & 0 & \dots & -a_{mm,2} & \dots & \dots & -a_{mm,\nu_m} & 1 \\ -a_{m1,3} & \dots & \dots & 0 & & \dots & -a_{mm,3} & \dots & \dots & 1 & \\ \vdots & & & & & \dots & \vdots & & & & \\ -a_{m1,\nu_{m1}} & 0 & & & & \dots & -a_{mm,\nu_m} & 1 & & & \\ 0 & & & & & \dots & 1 & & & & \end{bmatrix} \quad (6.13)$$

These transformations are very useful for system identification. Input/output difference equation models (6.7) are easier and more logical to use for discrete-time recursive identification, since they can be identified directly from the input/output data sequence of the system. The corresponding state-space model, if required, can then be easily and uniquely obtained from the identified input-output model (6.7) through the above transformations. One example is presented in Section 6.4. Note that if the input/output model is minimal order then so is the state space model.

## 6.3 The Multivariable AUDI Algorithm

The SISO augmented UD identification algorithm developed in Chapter 3 is a recursive, least-squares algorithm with improved numerical properties and provides simultaneous identification of model order and parameters. This algorithm is superior to the widely used recursive least-squares algorithm (Ljung & Söderström 1983, Söderström & Stojica 1989) in almost all aspects and is thus recommended for use in place of RLS for all applications (Niu et al. 1992). In this section, the AUDI algorithm is extended to the multivariable case and provides simultaneous identification of *model structure* (i.e. identification of the structural indices  $\nu_i, \nu_{ij}$ ) as well as the model parameters.

### 6.3.1 The Augmented UD Identification Structure

The  $r$ -input,  $m$ -output multivariable process described by (6.7) can be rearranged into the following input-output difference equation model

$$A_0 z(t) + A_1 z(t-1) + \dots + A_n z(t-n) = B_1 u(t-1) + \dots + B_n u(t-n) + v(t) \quad (6.14)$$

where

$$z(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_m(t) \end{bmatrix}, \quad u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_r(t) \end{bmatrix} \quad (6.15)$$

are the system output and input vectors, and  $v(t)$  is  $m$ -dimensional uncorrelated white noise with zero-mean.

The system model defined by (6.14) is described as having  $m$  subsystems corresponding to the  $m$  output variables. The model orders of each subsystem determine the structural indices of this multivariable process.

For the identification approach described below the user specifies the maximum order  $n$  that is required to adequately describe the largest of the  $m$  subsystems. The augmented UD identification identifies the parameters of all models from order 1 to  $n$  for each of the  $m$  subsystems plus the loss functions corresponding to each model. The loss functions are then used to select the model orders required for each subsystem (which in turn defines the structural indices,  $\nu_i$ , of the MIMO system). In spite of the large number of identified parameters, the computational load is comparable to using conventional RLS to identify  $n$ th-order MISO systems.

Based on (6.11) and the comments following it, it is clear that the  $A_0$  matrix has the



following special form

$$A_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_{21,0} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1,0} & a_{m2,0} & \cdots & 1 \end{bmatrix} \quad (6.16)$$

The element  $a_{ij,0}$  equals zero if the inequality in the second formula of (6.11) holds, that is

$$a_{ij,0} = 0, \quad \text{if } \deg\{p_{ii}(z)\} > \deg\{p_{ij}(z)\}, \text{ for } j < i$$

and  $a_{ij,0}$  does not equal to zero when the equality holds. *I.e.*,

$$a_{ij,0} \neq 0, \quad \text{if } \deg\{p_{ii}(z)\} = \deg\{p_{ij}(z)\}, \text{ for } j < i$$

$A_i, B_i, (i=1, \dots, n)$  have the forms

$$A_i = \begin{bmatrix} a_{11,i} & a_{12,i} & \cdots & a_{1m,i} \\ a_{21,i} & a_{22,i} & \cdots & a_{2m,i} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1,i} & a_{m2,i} & \cdots & a_{mm,i} \end{bmatrix} \quad (i=1, 2, \dots, n) \quad (6.17)$$

$$B_i = \begin{bmatrix} b_{11,i} & b_{12,i} & \cdots & b_{1r,i} \\ b_{21,i} & b_{22,i} & \cdots & b_{2r,i} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1,i} & b_{m2,i} & \cdots & b_{mr,i} \end{bmatrix} \quad (i=1, 2, \dots, n) \quad (6.18)$$

Similar to the SISO case, the augmented data vector for the AUDI algorithm is defined as

$$\varphi(t) = \begin{bmatrix} -s(t-n) \\ u(t-n) \\ \vdots \\ -s(t-1) \\ u(t-1) \\ -s(t) \end{bmatrix} \quad (6.19)$$

or in a more explicit form

$$\begin{aligned} \varphi(t) = & [-s_1(t-n), -s_2(t-n), \dots, -s_m(t-n), u_1(t-n), u_2(t-n), \dots, u_r(t-n), \\ & \dots \dots \\ & -s_1(t-1), -s_2(t-1), \dots, -s_m(t-1), u_1(t-1), u_2(t-1), \dots, u_r(t-1), \\ & -s_1(t), -s_2(t), \dots, -s_m(t)]^T \end{aligned}$$

where  $s$  and  $u$  are defined by (6.15).

By defining the coefficient matrix as

$$\Theta(t) = [A_n, B_n, \dots, A_1, B_1, A_0]^T \quad (6.20)$$

the regression equation can then be rewritten as

$$\Theta^T(t) \varphi(t) = v(t), \quad \text{or} \quad \varphi^T(t) \Theta(t) = v^T(t) \quad (6.21)$$

where  $\Theta(t)$  is matrix with dimension  $m \times [(m+r)n + m]$ , and  $\varphi(t)$  is a vector of length  $(m+r)n + m$ .

Define the augmented covariance matrix as

$$C(t) = \left[ \sum_{j=1}^t \varphi(j) \varphi^T(j) \right]^{-1} \quad (6.22)$$

with dimension  $[(m+r)n+m] \times [(m+r)n+m]$ , and decompose it into  $UDU^T$  form

$$C(t) = U(t) D(t) U^T(t) \quad (6.23)$$

where  $U(t)$  is a unit-upper-triangular matrix and  $D(t)$  is a diagonal matrix. This factored form of the augmented covariance matrix is called the *multivariable augmented UD identification structure (AUDI)* and contains all the information required for recursive identification.  $U=U(t)$  is the parameter matrix and has the form

$$U = U(t) = \begin{bmatrix} 1 & \theta_{12} & \theta_{13} & \theta_{14} & \cdots & \theta_{1,d-1} & \theta_{1d} \\ & 1 & \theta_{23} & \theta_{24} & \cdots & \theta_{2,d-1} & \theta_{2d} \\ & & 1 & \theta_{34} & \cdots & \theta_{3,d-1} & \theta_{3d} \\ & & & 1 & \cdots & \theta_{4,d-1} & \theta_{4d} \\ & & & & \ddots & \vdots & \vdots \\ & & & & & 1 & \theta_{d-1,d} \\ & 0 & & & & & 1 \end{bmatrix}_{d \times d} \quad (6.24)$$

where  $d \triangleq (m+r)n+m$ .  $D=D(t)$  is called the loss function matrix and has the form

$$\begin{aligned} D &= D^{-1}(t) \\ &= \text{diag} \left[ J^{(0)}(t), L^{(0)}(t), J^{(1)}(t), L^{(1)}(t), \dots, L^{(n-1)}(t), J^{(n)}(t) \right] \end{aligned} \quad (6.25)$$

where

$$\begin{aligned} J^{(j)}(t) &= \text{diag} \left[ J_1^{(j)}(t), J_2^{(j)}(t), \dots, J_m^{(j)}(t) \right] \\ L^{(j)}(t) &= \text{diag} \left[ L_1^{(j)}(t), L_2^{(j)}(t), \dots, L_r^{(j)}(t) \right] \end{aligned}$$

The superscript in parentheses represents the model order, e.g.,  $J^{(j)}(t)$  is the loss function block of the  $j$ th order system at time step  $t$ . As shown below, the columns in (6.24) contain the parameter estimates for all models of order 1 to  $n$  and the loss functions in (6.25) are used to select an appropriate model order.

### 6.3.2 The Parameter and Loss Function Matrices

For multivariable systems, the dimensions of the parameter matrix  $U$  and loss function matrix  $D$  can become very large (i.e.,  $(m+r)n+m$ ) and thus careful interpretation is needed. Bearing in mind that our system is assumed to be a  $m$ -input,  $r$ -output multivariable system, the  $U$  matrix is partitioned into blocks as shown in Figure 6.1. The following remarks apply

1. The  $U$  matrix is a unit-upper-triangular matrix so the nonzero part of each column (the upper triangular part) has a different length. i.e., the non-zero part of the first column has a length of 1 and the non-zero part of the last column has a length of  $d=(m+r)n+m$ .

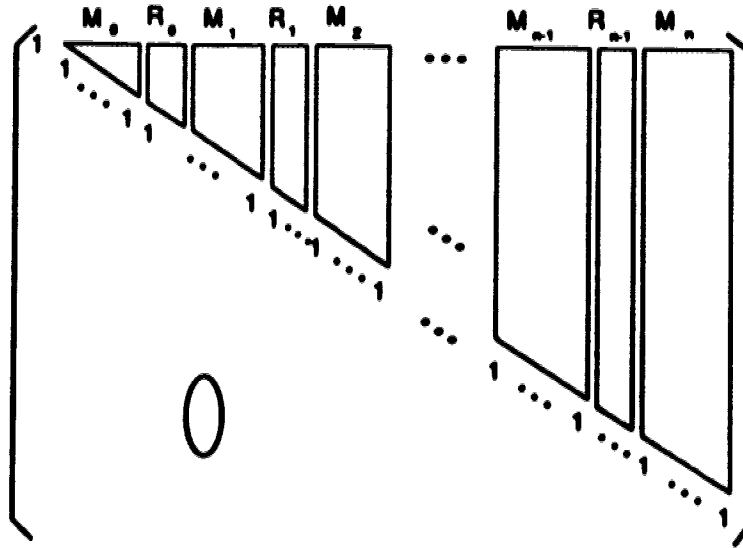


Figure 6.1: Partitioning of the Parameter Matrix  $U(t)$

2. The blocks denoted by  $M_0, M_1, \dots, M_n$  consists of  $m$  columns each, and are called *M-Blocks* thereafter. The blocks denoted by  $R_0, R_1, \dots, R_{n-1}$  consists of  $r$  columns each and are called the *R-Blocks*.
3. Each block fills the space above the unitary diagonal elements and has a different dimension. For example, the  $M_0$ -block shown in Figure 6.1, is triangular with  $m$  columns which contain  $1, 2, \dots, m$  elements, i.e. the number of elements equals the column number. The  $R_0$ -block has  $r$  columns containing  $m+1$  to  $m+r$  elements from left to right, the  $M_1$ -block has  $m$  columns containing  $m+r+1$  to  $m+r+m$  elements from left to right, and so on so forth.
4. The parameter matrix  $U$  contains the parameter estimates of all models from order 1 to order  $n$  (the user-specified maximum possible order) for each of the  $m$  sub-systems of (6.14). In other words, the  $m$  columns in each *M-block* correspond to the  $m$  subsystems in the multivariable model defined by (6.16), (6.17) and (6.18). More specifically, the  $M_1$ -block contains the parameter estimates for the first order models of each of the  $m$  subsystems of (6.14). The  $M_n$ -block contains the parameter estimates of  $n$ th order models. The identity of the parameters in each *M-block* is defined by (6.20).

Similarly, the loss function matrix  $D$  is also partitioned into blocks, as shown in Figure 6.2. The following remarks apply

1. The loss function matrix is a block diagonal matrix with a dimension  $d \times d$ , where  $d = (m+r)n + m$ , and each element is the loss function of a specific model.
2. The blocks denoted by  $M_0, M_1, \dots, M_n$  in Figure 6.2 are diagonal matrices with  $m$  elements, and are called *M-Blocks*. (The same name as in the parameter matrix is used to indicate the close connection with the *M-blocks* in the parameter matrix). The blocks denoted by  $R_0, R_1, \dots, R_{n-1}$  are diagonal with  $r$  elements, and are called the *R-blocks*.

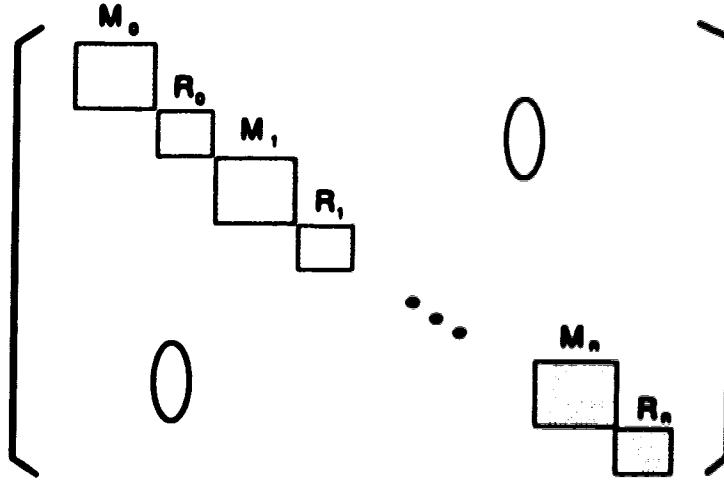


Figure 6.2: Partitioning of the Loss Function Matrix  $\mathcal{D}$

3. The  $M$ -blocks of  $\mathcal{D}$  are the loss functions corresponding to the parameter estimates ( $M$ -blocks) in the parameter matrix  $\mathcal{U}$ . That is, the  $m$  diagonal elements of the  $M_1$ -block of  $\mathcal{D}$  are the loss functions corresponding to the first order models, contained in the  $M_1$ -block of the parameter matrix. The  $M_n$ -block of  $\mathcal{D}$  contains the  $m$  loss functions for the  $n$ th order models of the  $m$  subsystems of (6.14).
4. The structural indices  $\nu_1, \nu_2, \dots, \nu_n$  of the multivariable system correspond to the model orders of each subsystem in (6.14), and hence can be determined by finding the appropriate model order for each subsystem. Since the loss functions of all  $n$  models for each of the  $m$  subsystems are available in the loss function matrix, it is easy, using the AIC or F-test, to determine the order of each subsystem. For instance, for the  $i$ th subsystem,  $i \in [1, m]$ , of the input-output models defined by (6.14) through (6.18), the loss functions of the 1st order model through  $n$ th order models are contained in the  $i$ th diagonal elements of blocks  $M_1, M_2$  up to  $M_n$  respectively of  $\mathcal{D}$  as shown in Figure 6.2.

The  $R$ -blocks in both the parameter matrix and the loss function matrix are intermediate variables and are not discussed further in this chapter.

After the model order of each of the  $m$ -subsystems (i.e., the structural indices) are determined, the difference equation model can be established by selecting the corresponding parameter blocks from the parameter matrix  $\mathcal{U}$ . If the state-space model is required, the input-output difference equation model can be uniquely transformed as explained in Section 6.2.

In summary, by decomposing the augmented covariance matrix (6.22) into  $UDU^T$  form, the parameter estimates from order 1 to  $n$  for each of the  $m$ -subsystems are generated simultaneously and contained in the  $M$ -blocks shown in Figure 6.1. If not known *a priori*, the appropriate structural indices of this multivariable system can also be easily determined from the  $M$ -blocks of the loss function matrix illustrated in Figure 6.2.

### 6.3.3 Recursive AUDI Algorithm

The multivariable recursive AUDI algorithm is almost identical to the AUDI algorithm for SISO system in Chapter 3 except that the data vector is constructed in a different manner and the parameter/loss function matrices are interpreted differently. The complete algorithm is outlined in Table 6.1. Details can be found in Chapter 3 with related discussion in Bierman (1977) or Ljung & Söderström (1983). The algorithm is initialized as  $C(0) = U(0) D(0) U^T(0) = \delta^2 I$  where  $\delta$  is a large integer.

Table 6.1: The Multivariable Recursive AUDI Algorithm

At every time step $t$ ,	
Construct $\varphi(t)$ according to (6.19), and do	
$f = U^T(t-1) \varphi(t)$ , $g = D(t-1) f$ , $\beta_0 = \lambda(t)$	: innovation sequence
for $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j$	
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j \lambda(t)$	: loss function update
$\mu_j = -f_j / \beta_{j-1}$ , $\nu_j = g_j$	
for $i=1$ to $j-1$ , do (skip for $j=1$ )	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i \mu_j$	: parameter update
$\nu_i = \nu_i + U_{ij}(t-1) \nu_j$	: Kalman gains
$U = U(t)$ , $D = D^{-1}(t)$	: result matrices

The following comments can be made about the AUDI algorithms.

1. The most significant advantage of the AUDI algorithm is that it is simultaneously order and time recursive, and can produce multiple models of different orders at every time interval. The most appropriate model can always be selected by investigating the corresponding loss functions provided in the loss function matrix (Niu et al. 1992).
2. The  $UDU^T$  factorization technique used by the AUDI algorithm is very stable and thus the numerical performance of the AUDI algorithm is excellent. A rule of thumb is, the AUDI algorithm can achieve the same accuracy with half the word length used by the RLS-type algorithms and thus it is a very good candidate for real-time applications, especially for multivariable systems which usually have very large matrix dimensions.
3. Another significant feature of the AUDI algorithm is that it is not sensitive to over-parameterization. More specifically, numerical problems occur first in the higher-order, over-parameterized models and do not affect the accuracy of the lower order models. The models from order 1 to the correct model order are always well-conditioned as long as the numerical problems with the higher order models do not cause math overflow in calculations. This is an obvious advantage of the AUDI algorithm over traditional least-squares algorithms.
4. In the AUDI algorithms, numerical problems are always associated with elements of the diagonal loss function matrix being too large or too small. A simple matrix regularization technique such as that suggested by Ljung & Söderström (1983)

can keep the augmented covariance matrix well-conditioned. The only cost is lower accuracy in the over-parameterized part of the models since only that part needs regularization. Matrix regularization can be easily done with the AUDI algorithm by applying an upper and a lower bound to the elements of the loss function matrix. This requires very little extra computation and thus is always recommended (Niu & Fisher 1994).

## 6.4 Simulation Example

The AUDI structure and algorithm are illustrated by the following example. Consider a process represented in a state space form as in (6.1) where

$$A = \left[ \begin{array}{cc|cc} 0 & 1 & 0 & 0 \\ -0.1 & 0.65 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \frac{1}{3} & -\frac{5}{6} & -0.25 & 1 \end{array} \right] \quad (6.26)$$

$$B = \left[ \begin{array}{cc} 1 & 2 \\ 0.25 & 0.8 \\ \hline 2 & 0 \\ 0.5 & 4 \end{array} \right], \quad C = \left[ \begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right] \quad (6.27)$$

Clearly the structure indices are  $\nu_1=2$ ,  $\nu_2=2$ ,  $\nu_{12}=0$  and  $\nu_{21}=1$ . From equations (6.12) and (6.13), the corresponding difference equation model can be uniquely determined as

$$P(z)z(t) = Q(z)u(t) + v(t)$$

where

$$P(z) = \begin{bmatrix} z^2 - 0.65z + 0.1 & 0 \\ \frac{5}{6}z - \frac{1}{3} & z^2 - z + 0.25 \end{bmatrix}$$

$$Q(z) = \begin{bmatrix} z - 0.4 & 2z - 0.5 \\ 2z - \frac{2}{3} & \frac{17}{3} \end{bmatrix}$$

There are two MISO subsystems corresponding to  $s_1$  and  $s_2$ . In difference equation form

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} s(t) + \begin{bmatrix} -0.65 & 0 \\ 0.83 & -1 \end{bmatrix} s(t-1) + \begin{bmatrix} 0.1 & 0 \\ -0.33 & 0.25 \end{bmatrix} s(t-2) =$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} u(t-1) + \begin{bmatrix} -0.4 & -0.5 \\ -0.67 & 5.67 \end{bmatrix} u(t-2) + v(t) \quad (6.28)$$

where  $s(t) = [s_1(t), s_2(t)]^T$  is the output vector,  $u(t) = [u_1(t), u_2(t)]^T$  is the input vector. Input excitation  $u_1(t)$  and  $u_2(t)$  are two uncorrelated random binary sequences;  $v(t) = [v_1(t), v_2(t)]^T$ , where  $v_1(t)$  and  $v_2(t)$  are two uncorrelated white noise sequences with zero means.

Assume the maximum possible order of the largest subsystem is  $n=3$  and the data length is 1000. The augmented data vector (6.19) is constructed as follows

$$\varphi(t) = [-s_1(t-3), -s_2(t-3), u_1(t-3), u_2(t-3),$$

$$\begin{aligned} & -z_1(t-2), -z_2(t-2), u_1(t-2), u_2(t-2), \\ & -z_1(t-1), -z_2(t-1), u_1(t-1), u_2(t-1), \\ & -z_1(t), -z_2(t) \end{aligned}^T$$

The coefficient matrix in the form of (6.20) is

$$\Theta_0 = \begin{bmatrix} 0.1 & 0 & -0.4 & -0.5 & -0.65 & 0 & 1 & 2 & 1 & 0 \\ -0.33 & 0.25 & -0.67 & 5.67 & 0.83 & -1 & 2 & 0 & 0 & 1 \end{bmatrix}^T \quad (6.29)$$

which corresponds to the  $M_2$  block in the parameter matrix shown in Figure 6.1. Note that the structure of the data and parameter vector are a simple extension of the SISO case (Niu et al. 1992) and constructed directly from (6.28).

Using the multivariable AUDI algorithm to identify this process, with a forgetting factor  $\lambda=0.99$ , produces the loss function matrix at step 1000 given in Table 6.2 along with the identifiers  $M_i, R_i, i=1, 2, 3$  that links it to Figure 6.2.

Now examine the M-blocks and R-blocks shown in Table 6.2. The first diagonal elements of each block  $M_i, i=0, 1, 2, 3$ , give the loss functions for models with orders from 0 to 3 for subsystem 1. The second diagonal elements of each block  $M_i, i=0, 1, 2, 3$ , give the loss functions for models with orders from 0 to 3, for subsystem 2. For convenience these loss function values are rearranged and shown in Table 6.3. Obviously, both the loss functions of subsystem 1 and 2 exhibit sharp decreases when the model order is increased from 1 to 2, and become flat for model orders higher than 2. This suggests an estimated order of 2 for both subsystems, which is consistent with the structural indices of the true system, i.e.,  $\nu_1=\nu_2=2$ .

Table 6.2: Loss Function Matrix of the  $2 \times 2$  Multivariable System (Compare Figure 2)

$M_0$	$R_0$	$M_1$	$R_1$	$M_2$	$R_2$	$M_3$
595.15	99.59	3.397	99.29	0.001	99.11	0.004
24040	99.79	299.6	99.17	0.946	99.21	0.003

Table 6.3: Loss Functions of all the Subsystems

Model Order	0	1	2	3
Subsystem 1	595.15	3.397	0.001	0.004
Subsystem 2	24040	299.6	0.946	0.003

The converged parameter matrix  $U$  is shown in Table 6.4. It has the same form as (6.24) and is partitioned into blocks corresponding to Figure 6.1.

The data from the  $M$ -blocks of the parameter matrix are rearranged and shown in Table 6.5 in a form corresponding to (6.14). Since the estimated structural indices are  $\nu_1 = \nu_2 = 2$ , the estimated parameters of the system model are found by picking up the  $M_2$ -block in the parameter matrix, which from (6.20) is:

$$\begin{aligned}\hat{\theta} &= [\hat{\lambda}_2, \hat{\beta}_2, \hat{\lambda}_1, \hat{\beta}_1, \hat{\lambda}_0]^T \\ &= \begin{bmatrix} 0.107 & 0.003 & -0.400 & -0.456 & -0.633 & -0.004 & 0.997 & 1.999 & 1 & 0 \\ -0.328 & 0.250 & -0.658 & 5.661 & 0.810 & -1.000 & 1.946 & -0.079 & -0.046 & 1 \end{bmatrix}^T\end{aligned}$$

Comparison of these estimated parameters with (6.29) shows that the estimated parameters are quite close to their true values. This can be converted into the input/output equation form with

$$\begin{aligned}P(z) &= \begin{bmatrix} z^2 - 0.633z + 0.107 & -0.004z + 0.003 \\ 0.810z - 0.328 & z^2 - z + 0.25 \end{bmatrix} \\ Q(z) &= \begin{bmatrix} z - 0.4 & 1.999z - 0.456 \\ 1.946z - 0.658 & 5.661 \end{bmatrix}\end{aligned}$$

If a state-space model is required then it can be obtained by using the transformation in section 6.2.3, which gives

$$\begin{aligned}A &= \left[ \begin{array}{cc|cc} 0 & 1 & 0 & 0 \\ -0.107 & 0.633 & -0.003 & 0.004 \\ \hline 0 & 0 & 0 & 1 \\ 0.328 & 0.810 & -0.25 & 1.00 \end{array} \right] \\ B &= \left[ \begin{array}{cc} 0.997 & 1.999 \\ 0.239 & 0.809 \\ \hline 1.946 & -0.079 \\ 0.461 & 3.964 \end{array} \right], \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\end{aligned}$$

Obviously, this is a good estimate for (6.26) and (6.27).

Note that if subsystem #1 were assumed to be first-order rather than second order in the above example, then the parameters for the  $s_1$  subsystem would be extracted from column 1 of the  $M_1$ -block in Table 6.4. The parameter estimates for the second order subsystem  $s_2$  would be extracted from column 2 of the  $M_2$ -block as described above.

## 6.5 Conclusion

The augmented UD identification algorithm is extended from single-input, single-output systems to multivariable systems. This provides a method for simultaneous identification of the model structure (i.e., the order of the MISO subsystems) and the parameters of the MISO systems. The algorithm is simple in structure, easy to interpret and implement, and has excellent numerical properties. The MISO AUEI algorithm can also be easily formulated into an extended least-squares version similar to chapter 4, or the instrumental variable version similar to that in Chapter 5, and thus provides the basis for a family of algorithms for multivariable system identification.



Table 6.4: The Parameter Matrix of the Multivariable System (Compare Figure 1)

	$M_0$	$R_0$	$M_1$	$R_1$	$M_2$	$R_2$	$M_3$
1	-2.5072	-0.0109 0.0191	-0.3754 1.1712	0.0345 0.1270	0.1072 -0.3283	-0.3968 -0.0738	-0.0011 -0.0430
	1	0.0041 -0.0017	0.0002 0.6998	-0.0192 -0.0228	0.0032 0.2495	0.2170 -0.0673	0.0169 0.0197
		1 -0.0064	0.9941 -9.6636	-0.3132 -0.5874	-0.4000 -0.6578	-0.3718 1.0772	-0.0812 -0.0799
			1.9973 -23.3808	-0.7365 -1.3220	-0.4558 5.6613	5.3609 -0.5814	0.2791 0.5234
			1 -11.6911	-0.3791 -0.6399	-0.6332 0.8097	-0.4287 1.4601	0.0117 -0.4367
			1	0.0327 0.0323	-0.0043 -1.0000	-0.9082 0.3267	-0.0596 0.1568
				1 -0.0128	0.9969 1.9461	1.1005 -1.8293	-0.2428 -0.4457
					1.9986 -0.0785	-1.4778 -2.2696	-0.3827 5.6898
					1 -0.0461	-0.7521 -1.1160	-0.5975 0.7923
						0.9337 -0.3562	0.0561 -0.0965
						1 -0.0027	0.9956 1.9358
							1.9994 -0.0965
							1 -0.0555
							1

Table 6.5: Rearranged Parameter Matrix

$n$	$A_0$	$A_1$	$B_1$	$A_2$	$B_2$	$A_3$	$B_3$
3	1	-0.5008	0.0366	0.9999	1.9999	0.012 -0.060	-0.243 -0.389
	-0.066	1	0.782	-0.097	1.896	-0.457 0.157	-0.446 5.690
2	1	-0.653	-0.004	0.997	1.999	0.107 0.003	-0.400 -0.456
	-0.046	1	0.810	-1.000	1.946	-0.079 -0.528	0.250 -0.688
1	1	-0.575	0.000	0.994	1.997		
	-11.69	1	0.700	1.171	-23.38	-9.664	
*	1	-0.65	0	1	2	0.1 0	-0.4 -0.5
	0 1	0.53	-1	2	0	-0.33 0.25	-0.67 5.67

$n$  = order of estimated model      \* = actual parameter values ( $n=2$ )

# Bibliography

- Ackermann, J. E. & Bucy, R. S. (1971), 'Canonical minimal realization of a matrix of impulse response sequences', *Inform. Control* 19, 224 - 231.
- Bierman, G. J. (1977), *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York.
- Dion, J. M. & Lamare, H. (1984), Direct model reference adaptive control for linear multivariable systems, in 'Proceedings of the 9th IFAC World Congress', Budapest, p. 87.
- Eldem, V. & Yildizbayrak, N. (1988), 'Parameter and structure identification of linear multivariable systems', *Automatica* 24(3), 365-373.
- Elliott, H. & Wolovich, W. A. (1982), 'A parameter adaptive control structure for linear multivariable systems', *IEEE Transactions on Automatic Control* 27(2), 340 - 352.
- Goodwin, G. C., Ramadge, P. J. & Caines, P. E. (1980), 'Discrete-time multivariable adaptive control', *IEEE Transactions on Automatic Control* 25(3), 449 - 456.
- Guidorzi, R. P. (1975), 'Canonical structure in the identification of multivariable systems', *Automatica* 11(4), 361-374.
- Guidorzi, R. P. (1981), 'Invariant and canonical forms for systems structural and parameter identification', *Automatica* 17(1), 117-133.
- Ho, B. L. & Kalman, R. E. (1966), 'Effective construction of linear state variable models from input-output functions', *Regelungstechnik* 14, 545.
- Ljung, L. & Söderström, T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass.
- Lucenberger, D. G. (1987), 'Canonical forms for linear multivariable systems', *IEEE Transactions on Automatic Control* 32(3), 290-293.
- Morse, A. S. (1981), Parameterization for multivariable adaptive systems, in 'IEEE Proc. Yale Workshop on Adaptive Control'.
- Niu, S. & Fisher, D. G. (1991), MIMO system identification using an augmented UD identification algorithm, in 'Proceedings 1991 American Control Conference', Vol. 1, Boston, U.S.A., pp. 699 - 703.
- Niu, S. & Fisher, D. G. (1993), 'Simultaneous structure identification and parameter estimation of multivariable systems', *International Journal of Control*. Accepted for publication.

- Niu, S. & Fisher, D. G. (1994), 'Multiple model least squares method'. Submitted to 1994 American Control Conference.
- Niu, S., Fisher, D. G. & Xiao, D. (1992), 'An augmented total least squares algorithm', *International Journal of Control* 56(1), 193 - 211
- Niu, S., Fisher, D. G. & Xiao, D. (1993), 'A factored form total least squares instrumental Variable identification algorithm', *International Journal of Adaptive Control and Signal Processing* 7(4), 261-273.
- Niu, S., Xiao, D. & Fisher, D. G. (1990), 'A recursive least squares simultaneous identification of model order and parameters', *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38(5), 884 - 886.
- Söderström, T. & Stoica, P. (1989), *System Identification*, Prentice Hall, Englewood Cliffs, New Jersey.
- Tether, A. J. (1970), 'Construction of minimal linear state variable models from finite input-output data', *IEEE Transactions on Automatic Control* 15(4), 427 - 436.

## Chapter 7

# IDENTIFICATION OF TIME VARYING PROCESS

**T**he mechanism of information forgetting for recursive identification is investigated and discussed from a very general point of view, based on the augmented UD identification algorithm developed in previous chapters. It is shown that all the information pertinent to identification is contained in the augmented covariance matrix. Recursive information forgetting is simply a question of appropriately controlling the weight taken by new process information in the augmented covariance matrix. Information forgetting can be classified into *relative forgetting* and *absolute forgetting* and can both be controlled by manipulating a simple diagonal matrix. Convenient guidelines are provided for analyzing and designing information forgetting schemes. An improved information forgetting algorithm that implements these guidelines is also presented.

### 7.1 Introduction

Tracking the time varying dynamics of a process is a fundamental problem in control and signal processing. Since most real processes are nonlinear in nature, linear process models provide an accurate description of the process dynamics only in the vicinity of the current operating points. The parameters of the linear model change as the process moves from one operating condition to another and hence a recursive forgetting mechanism is needed to appropriately discard old data and put more emphasis on the recently obtained information which represents the latest process dynamics. Recursive identification with an appropriate information forgetting mechanism is a very important but also very difficult area of recursive identification.

Many different information forgetting schemes exist in the literature, including exponential forgetting (EF) (Åström, Borison, Ljung & Wittenmark 1977, Ljung & Söderström 1983), variable forgetting factor (Forbes, Keruehbaum & Ydell 1981, Cordero & Mayne 1981), directional forgetting (Saeid & Foss 1983, Haggund

---

<sup>1</sup>A version of this chapter has been published as : S. Niu and D. Grant Fisher, 1983, Information Forgetting in Recursive Identification, proceedings of the 1983 American Control Conference, San Francisco, 788-800. A revised version is submitted for publication as: S. Niu and D. Grant Fisher, Recursive Information Forgetting Based on AUDI Algorithm, International Journal of Control.

1983, Kulhávy & Kárny 1984), covariance resetting (Goodwin & Teoh 1983, Vogel & Edgar 1982) and the constant trace method (Sripada & Fisher 1987, Irving 1979). However, with the AUDI notation, all these methods can be conveniently classified into two categories: *relative forgetting* and *absolute forgetting*. Relative forgetting methods use a forgetting factor to scale the augmented covariance matrix to proportionally forget old information. Absolute forgetting schemes forcibly modify the augmented covariance matrix/augmented information matrix with user-specified values, which may not necessarily be related to the previous augmented covariance matrix. For example, the exponential forgetting method, which falls into the relative forgetting category, uses a constant forgetting factor to scale the ACM at every time step, i.e., the new ACM is a scalar multiple of the previous ACM. On the other hand, the covariance resetting algorithm, which is an absolute forgetting method, simply resets all or part of the ACM to a value, which is not necessarily related to the previous ACM.

In this chapter, information forgetting is discussed from a very general point of view, based on the AUDI algorithms. Deeper insight into the information forgetting principle is provided by taking advantage of the AUDI structure. By using the concept of *relative forgetting* and *absolute forgetting*, information forgetting in recursive identification is simplified into the problem of controlling the weight that the new information takes in the updated augmented covariance matrix. Simpler and more convenient guidelines are derived for analyzing and designing new information forgetting schemes. An improved information forgetting method is proposed as an implementation example and simulation shows excellent results.

## 7.2 Information Accumulation

Consider a process described by the following linear difference equation model

$$z(t) + a_1 z(t-1) + \dots + a_n z(t-n) = b_1 u(t-1) + \dots + b_n u(t-n) + v(t) \quad (7.1)$$

Define the augmented data vector as

$$\varphi(t) = [-z(t-n), u(t-n), \dots, -z(t-1), u(t-1), -z(t)]^T \quad (7.2)$$

As shown by the AUDI structure in Chapter 2, all the information on the process parameters and loss functions for all models from order 1 to a user-specified maximum number  $n$  are contained, implicitly, in the augmented covariance matrix

$$C(t) = \left[ \sum_{j=1}^t \varphi(j) \varphi^T(j) \right]^{-1} \quad (7.3)$$

or equivalently, in the augmented information matrix

$$S(t) = \sum_{j=1}^t \varphi(j) \varphi^T(j) = C^{-1}(t) \quad (7.4)$$

Batch identification methods collect all the input/output data first, then use them simultaneously to construct the augmented covariance matrix or the augmented information matrix. In recursive algorithms, the new information from the input/output data is accumulated in the ACM/AIM as it is obtained, that is at time  $t$ ,

$$S(t) = S(t-1) + \varphi(t) \varphi^T(t) \quad \text{or} \quad C^{-1}(t) = C^{-1}(t-1) + \varphi(t) \varphi^T(t) \quad (7.5)$$

In the AUDI algorithm, the concept of recursive identification becomes the recursive accumulation of process information into the augmented covariance matrix using (7.5). As will be shown later, this simplifies the interpretation and implementation of the existing information forgetting mechanism, and also makes it easier to design new information forgetting methods.

Notice that the diagonal elements of the rank-one matrix  $\varphi(t)\varphi^T(t)$  are always non-negative, therefore, the augmented information matrix  $S(t)$  is always non-decreasing. The new information  $\varphi(t)\varphi^T(t)$  from the process at each different time interval  $t$  is accumulated into the AIM with exactly the same weight. As time goes on, the AIM will eventually go to infinity, and the new information will be "buried" in the AIM, i.e.,

$$\lim_{t \rightarrow \infty} S(t) \approx \lim_{t \rightarrow \infty} S(t-1) \rightarrow \infty$$

The identification algorithm no longer responds to any new information in  $\varphi(t)\varphi^T(t)$  and effectively shuts itself off. This is the main reason that an information forgetting mechanism is needed for tracking time varying process. To keep the AIM from going to infinity (or, in other words, to keep the ACM from going to zero), the updating formula (7.5) must be modified. A general formula is given as follows

$$S(t) = \mathcal{F}(S(t-1), \varphi(t)) \quad (7.6)$$

where  $\mathcal{F}$  represents a particular relationship or function. All existing information forgetting methods are based on this principle and they differ only in the choice of the function  $\mathcal{F}$ . In essence, information forgetting is simply a problem of controlling the weight that the new information  $\varphi(t)\varphi^T(t)$  will take in the ACM/AIM matrix. In other words, the mechanism for information forgetting is to appropriately control the relative importance of the new information  $\varphi(t)\varphi^T(t)$  as compared with what is already contained in the augmented covariance matrix. A less general formula deduced from (7.6) is

$$\begin{cases} \tilde{S}(t) = S(t-1) + \varphi(t)\varphi^T(t) \\ S(t) = \mathcal{F}(\tilde{S}(t)) \end{cases} \quad (7.7)$$

which still covers most of the existing methods as its special cases. The numerous techniques for information forgetting in recursive identification differ only in the criteria used to determine the weight of the new information  $\varphi(t)\varphi^T(t)$  added to the ACM and can all be interpreted with formula (7.6) or (7.7). For example,

1. **NO INFORMATION FORGETTING.** This is equivalent to using a function  $\mathcal{F}$  which basically equates  $S(t)$  to  $\tilde{S}(t)$ , i.e.,

$$S(t) = \tilde{S}(t)$$

As discussed earlier, with no information forgetting mechanism, all the data go into the augmented information matrix  $S(t)$  with exactly the same weights (or importance).  $S(t)$  eventually goes to infinity and loses the ability to track time varying parameters.

2. **EXPONENTIAL FORGETTING METHOD.** A constant forgetting factor  $\lambda$  is introduced to exponentially forget old information, and put more emphasis on newly obtained data. That is, the older the data, the less important it is and the less weight it will have in the augmented covariance matrix. The information accumulation formula for the exponential forgetting method can be represented as

$$S(t) = \lambda \tilde{S}(t) \quad \text{or} \quad C(t) = \tilde{C}(t) / \lambda$$

Exponential forgetting incorporates the important idea of information forgetting. However, this method does not work well in practice since complications such as "covariance windup" or "bursting" may occur (Shah & Chuet 1991, Sripada & Fisher 1987, Åström & Wittenmark 1980, Morris et al. 1989).

3. **VARIABLE FORGETTING METHOD.** This category includes the variable forgetting factor method (Fortescue et al. 1981), the constant trace/determinant method (Rogers 1989), the directional forgetting method (Hägglund 1983, Kulhávy 1987), etc. All of them use a variable forgetting factor  $\lambda(i)$  instead of a fixed value to discard obsolete information. The more information the matrix  $\varphi(i) \varphi^T(i)$  contains, the bigger the weight it is given when it is accumulated into the AIM. The information accumulation formula is given by

$$S(i) = \lambda(i) \tilde{S}(i) \quad \text{or} \quad C(i) = \tilde{C}(i) / \lambda(i) \quad (7.8)$$

The main problem with this class of methods is that it is usually difficult to determine how much information is contained in the new information matrix  $\varphi(i) \varphi^T(i)$  relative to the current AIM. There is no clear answer to this question. The methods commonly used to judge the information content of the  $\varphi(i) \varphi^T(i)$  matrix involve maintaining a selected information criterion of the AIM/ACM at a constant value, as new information from the process is accumulated. All algorithms in this category differ only in how this information criterion is chosen. For example, the constant trace method keeps the trace of the ACM at a constant while the constant determinant method maintains the determinant of the ACM at an appropriate value. All these algorithms are quite *ad hoc*. However, if an appropriate information criterion can be found, this type of method can give very good results in real applications.

4. **COVARIANCE RESETTING.** This method modifies the current augmented covariance matrix and thereby changes the relative weight of the new information in the updated augmented covariance matrix. The two most commonly used covariance resetting schemes are

$$C(i) = \tilde{C}(i) + Q \quad \text{or} \quad C(i) = Q \quad (7.9)$$

where  $Q$  is a constant (usually diagonal) matrix. Again, the difficulty with covariance resetting method is deciding when and how much the covariance should be reset in order for the new information to have an appropriate weight in the augmented covariance matrix.

Based on the above discussion, it is seen that the information forgetting problem actually reduces to the problem of appropriately constructing the augmented covariance matrix with (7.6) or (7.7). A general information forgetting method requires some means of determining the information content of the new matrix  $\varphi(i) \varphi^T(i)$  relative to that contained in the current AIM/ACM. This is the key step towards a good information forgetting mechanism and is discussed below.

## 7.3 Information Forgetting with AUI

From the last section, it is seen that information forgetting is basically a problem of appropriately updating the augmented covariance matrix or augmented information matrix. In this section, the concept of information forgetting is further simplified based on the augmented UD identification algorithm.

Using AUDI notation and noting that  $C(t) = U(t) D(t) U^T(t)$ , the recursive information accumulation formula (7.5) can be rewritten as

$$[U(t) D(t) U^T(t)]^{-1} = [U(t-1) D(t-1) U^T(t-1)]^{-1} + \varphi(t) \varphi^T(t) \quad (7.10)$$

Successful identification implies that the parameter matrix  $U=U(t)$  converges to a constant matrix, and the inverse of the loss function matrix  $D(t)=\mathcal{D}^{-1}$  converges to a zero matrix, i.e.,

$$\lim_{t \rightarrow \infty} \{U(t) D(t) U^T(t)\} = 0, \Rightarrow \lim_{t \rightarrow \infty} D(t) = 0$$

That is, the tracking ability of the AUDI identification algorithm (i.e. the relative importance of the "old" information stored in the  $U D U^T$  term of (7.10) compared to the "new" information in the  $\varphi(t) \varphi^T(t)$  term) can be controlled by modifying the diagonal loss function matrix  $\mathcal{D}$ . Consequently, a general information forgetting formula for AUDI algorithms can be written very simply as

$$\begin{cases} [U(t) \tilde{D}(t) U^T(t)]^{-1} = [U(t-1) D(t-1) U^T(t-1)]^{-1} + \varphi(t) \varphi^T(t) \\ D(t) = \mathcal{G}(\tilde{D}(t)) \end{cases} \quad (7.11)$$

where  $\mathcal{G}$  is a function analogous to  $\mathcal{F}$  in (7.7). Since  $D$  is diagonal, it is much easier to interpret and work with than the full ACM/AIM matrices. Therefore, from now on, the emphasis in the discussion of information forgetting will be shifted from the ACM/AIM to the loss function matrix  $\mathcal{D}$ .

### 7.3.1 Relative Forgetting

The relative forgetting approach for AUDI algorithms is much simpler than the variable forgetting methods discussed above. The main idea is to scale the loss function matrix with a diagonal matrix  $\Lambda(t)$

$$D(t) = \tilde{D}(t) \Lambda(t) \quad (7.12)$$

where the key question is the selection of the  $\Lambda(t)$  matrix. A special case of (7.12) is

$$\Lambda(t) = \frac{1}{\lambda(t)} I \quad (7.13)$$

which is equivalent to the forgetting factor method in (7.8)

$$\begin{aligned} C(t) &= U(t) D(t) U^T(t) = U(t) \left[ \frac{\tilde{D}(t)}{\lambda(t)} \right] U^T(t) \\ &= [U(t) \tilde{D}(t) U^T(t)] / \lambda(t) = C(t-1) / \lambda(t) \end{aligned}$$

or in terms of AIM

$$S(t) = \lambda(t) S(t-1)$$

that is, scaling the matrix  $D(t)$ , which is a simple diagonal matrix with clear physical meanings, has the same effect as scaling the augmented covariance matrix  $C(t)$ . The new loss function matrix is simply a multiple of the previous loss function matrix and is therefore referred to as *relative forgetting*. Relative forgetting takes exponential forgetting, variable forgetting and directional forgetting as its special cases, since they all use a forgetting factor  $\lambda(t)$  as in (7.13). The AUDI-LS algorithm with a variable forgetting factor  $\lambda(t)$  is shown in Table 7.1.



Table 7.1: The Recursive AUDI Algorithm

$\varphi(t) = [-z(t-n), u(t-n), \dots, -z(t-1), u(t-1), -z(t)]^T$	: data vector
$f = U^T(t-1) \varphi(t), g = D(t-1) f, \beta_0 = \lambda(t)$	: innovations
for $j=1$ to $d$ , do	
$\beta_j = \beta_{j-1} + f_j g_j$	: scaling factor
$D_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j / \lambda(t)$	: loss functions
$\mu_j = -f_j / \beta_{j-1}, \nu_j = g_j$	
for $i=1$ to $j-1$ , do (skip for $j=1$ )	
$U_{ij}(t) = U_{ij}(t-1) + \nu_i \mu_j$	: parameter update
$\nu_i = \nu_i + U_{ij}(t-1) \nu_j$	: Kalman gain
$U = U(t), \quad \mathcal{D} = \mathcal{D}^{-1}(t)$	: result matrices

where it is seen that the loss function is updated by

$$D_{jj}(t) = D_{jj}(t-1) \beta_j / \beta_{j-1} / \lambda(t) \quad (7.14)$$

or after rearrangement

$$D_{jj}(t) = \tilde{D}_{jj}(t) / \lambda(t), \quad \text{and} \quad \tilde{D}_{jj}(t) = D_{jj}(t-1) \beta_{j-1} / \beta_j$$

which is exactly the same as formula (7.12) or (7.13). The relative forgetting method is a generalization of the variable forgetting methods. Since the forgetting factor  $\lambda(t)$  is calculated based on the diagonal  $D(t)$  matrix rather than the full matrix  $C(t)$ , it is much more convenient. However, the problem of how to evaluate the information content of the AIM/ACM matrix still exists.

Consider the following properties of the AIM/ACM that are affected by the introduction of a forgetting factor defined by (7.13)

1. **DETERMINANT.** Without information forgetting, the determinant of the augmented covariance matrix  $C(t)$  would converge to zero. Note that the determinant of the parameter matrix  $U(t)$  is always unity, thus the determinant of the ACM equals the determinant of the  $D(t)$  matrix, which is a diagonal matrix, i.e.,

$$\det\{C(t)\} = \det\{U(t) D(t) U^T(t)\} = \det\{D(t)\}$$

Introducing a forgetting factor is equivalent to

$$C(t) = \frac{\tilde{C}(t)}{\lambda(t)} \Rightarrow \det\{C(t)\} = \frac{\det\{\tilde{C}(t)\}}{\lambda^n(t)} \Rightarrow \det\{D(t)\} = \frac{\det\{\tilde{D}(t)\}}{\lambda^n(t)}$$

The constant determinant method maintains the determinant of the ACM at a constant value by using a forgetting factor  $\lambda(t)$ , which is equivalent to maintaining the determinant of the  $D(t)$  matrix constant. From the AUDI structure, it is known that the  $D(t)$  matrix contains the loss functions for all models, and thus the determinant of the loss function matrix is the product of all the loss functions of all models. Clearly, there is no solid theoretical basis for using this product as the information criterion.

2. **TRACE.** The forgetting factor scales the trace of the ACM/AIM by a factor of  $\lambda(t)$ , that is

$$\text{tr}\{C(t)\} = \frac{\text{tr}\{\tilde{C}(t)\}}{\lambda(t)} \quad \text{or} \quad \text{tr}\{S(t)\} = \lambda(t) \text{tr}\{S(t-1)\}$$

the constant trace method tries to maintain the trace of the ACM at a constant by using a forgetting factor. However, for least-squares estimators, if the process noise is white with zero mean and variance  $\sigma_v^2$ , then  $\sigma_v^2 C(t)$  is the covariance matrix of the parameter estimates, which implies that the trace of the  $C(t)$  matrix equals to the sum of the variance of the parameter estimates of all the  $n$  models. Clearly, the trace can be affected by many factors such as the noise level and model order. Using the trace of  $C(t)$  as the information criterion is therefore *ad hoc* and a great deal of experience would be required to select an appropriate value for this criterion.

3. **CONDITION NUMBER.** The condition number of the ACM is approximately a linear function of the condition number of the loss function matrix since the parameter matrix converges to a constant matrix, that is

$$\text{cond}\{C(t)\} = \text{cond}\{U(t) D(t) U'(t)\} \approx \kappa \cdot \text{cond}\{D(t)\}$$

where  $\kappa$  is a constant positive number determined by the condition number of  $U(t)$ . In another words, the condition number of the ACM is mainly determined by the condition of the loss function matrix. Since the loss function matrix is diagonal, its condition number can be conveniently calculated as

$$\text{cond}\{D(t)\} = \frac{\max_{1 \leq i \leq d} \{D_{ii}(t)\}}{\min_{1 \leq i \leq d} \{D_{ii}(t)\}}$$

Introducing a forgetting factor by  $C(t) = \tilde{C}/\lambda(t)$  is equivalent to scaling  $D(t)$  by  $D(t) = \tilde{D}(t)/\lambda(t)$ , and since

$$\text{cond}\{D(t)\} = \text{cond}\left\{\frac{\tilde{D}(t)}{\lambda}\right\} = \text{cond}\{\tilde{D}(t)\}$$

this shows that in theory, introducing a forgetting factor can not improve the condition of the AIM/ACM. However, the forgetting factor can prevent the ACM from going to zero, thus round-off errors are less serious, and hence gives the impression that introducing a forgetting factor makes the identification numerically more stable.

### 7.3.2 Absolute Forgetting

As mentioned earlier, the information forgetting problem is simply a problem of determining the relative weight given to the new information  $\varphi(t) \varphi'(t)$  when it is accumulated into the ACM in (7.10). Relative forgetting schemes scale the ACM in order to give the new information an appropriate weight. The absolute forgetting schemes, however, simply reassign the augmented covariance matrix or augmented information matrix to a value, which can be completely unrelated to its previous values. The covariance resetting method (Goodwin & Sin 1984, Xie & Evans 1984, Vogel & Edgar 1982) is an example of the absolute forgetting approaches.

The main idea behind absolute resetting method is to regularly reset the augmented information matrix/augmented covariance matrix to appropriate values, in accordance

with a user-specified criterion. This keeps the ACM from going to zero and also assigns an appropriate weight for the new information from the process. With the ACM updating formula as

$$[\hat{C}(t)]^{-1} = [C(t-1)]^{-1} + \varphi(t) \varphi^T(t)$$

usually the absolute forgetting schemes take one of the two forms in (7.9), i.e.,

$$C(t) = \hat{C}(t) + Q, \quad \text{e.g., } Q = \sigma I \quad (7.15)$$

or

$$C(t) = Q, \quad \text{e.g., } Q = \sigma I \quad (7.16)$$

The above ACM resetting schemes can be very effective but are *ad hoc* as to when and how much the ACM should be reset. This is the major problem with the covariance resetting methods in the literature. Many approaches exist but they are more or less based on trial and error or application-specific results.

Another problem with ACM resetting is that the parameters exhibit a drift towards the origin (Ljung & Gunnarsson 1990), i.e., ACM resetting can also produce unexpected parameter resetting. This is made clear by the following simple example.

Consider the ACM updating formula (7.15), and rewrite it in UDU<sup>T</sup> form as

$$U(t) D(t) U^T(t) = \bar{U}(t) \bar{D}(t) \bar{U}^T(t) + \sigma I$$

where  $\sigma I$  is the resetting term. Assuming the augmented covariance matrix at time  $t$ , which is after being updated and before being reset, is given by

$$\begin{aligned} \hat{C}(t) &= \bar{U}(t) \bar{D}(t) \bar{U}^T(t) \\ &= \begin{bmatrix} 1 & \theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 & \\ & d_2 \end{bmatrix} \begin{bmatrix} 1 & \theta \\ 0 & 1 \end{bmatrix}^T \\ &= \begin{bmatrix} d_1 + d_2 \theta^2 & \theta d_2 \\ \theta d_2 & d_2 \end{bmatrix} \end{aligned}$$

Then resetting  $\hat{C}(t)$  with  $\sigma I$  by using (7.15) leads to

$$\begin{aligned} C(t) &= \hat{C} + \sigma I = \begin{bmatrix} d_1 + d_2 \theta^2 + \sigma & \theta d_2 \\ \theta d_2 & d_2 + \sigma \end{bmatrix} \\ &= \begin{bmatrix} 1 & \frac{\theta d_2}{d_2 + \sigma} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 + d_2 \theta^2 (1 - \frac{d_2}{d_2 + \sigma}) + \sigma & \\ & d_2 + \sigma \end{bmatrix} \begin{bmatrix} 1 & \frac{\theta d_2}{d_2 + \sigma} \\ 0 & 1 \end{bmatrix}^T \end{aligned}$$

which indicates that resetting the ACM by  $\sigma I$  causes the parameter  $\theta$  to drift from  $\theta$  to  $\theta d_2 / (d_2 + \sigma)$ . After the parameters converge,  $d_2$ , the inverse of a loss function term, becomes a small positive number. Thus  $\theta d_2 / (d_2 + \sigma)$  becomes a small number, which indicates that the parameter  $\theta$  drifts towards zero.

If the resetting of the second kind (7.16) is used, then

$$\begin{aligned} C(t) &= \sigma I \quad (\text{not related to } \hat{C}(t)) \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma & \\ & \sigma \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T \end{aligned}$$

This resets the parameters to 0. This is obviously not desirable for the general case.

In the AUDI algorithm, information accumulation into the ACM is controlled by the loss function matrix and thus ACM resetting can be replaced by loss function matrix resetting. This has obvious advantages over ACM resetting. The resetting, from  $\hat{C}(t)$  to  $C(t)$ , is accomplished by setting

$$U(t) = \bar{U}(t) \quad \text{and} \quad D(t) = \bar{D}(t) + Q \quad (7.17)$$

this means that

$$\begin{aligned} \hat{C}(t) &= \begin{bmatrix} 1 & \theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 & \\ & d_2 \end{bmatrix} \begin{bmatrix} 1 & \theta \\ 0 & 1 \end{bmatrix}^T \\ C(t) &= \begin{bmatrix} 1 & \theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 + \sigma & \\ & d_2 + \sigma \end{bmatrix} \begin{bmatrix} 1 & \theta \\ 0 & 1 \end{bmatrix}^T \end{aligned}$$

In this way,  $C(t)$  is prevented from approaching zero, and at the same time, the tracking ability is effectively restored and parameter drift is avoided.

The following properties of the  $D(t)$  matrix provide theoretical guidelines for resetting the loss function matrix

1. With no information forgetting, the elements of the loss function matrix increase linearly with time step  $t$ , that is

$$\lim_{t \rightarrow \infty} D^{-1}(t) = \lim_{t \rightarrow \infty} (t \cdot R_0) = \infty$$

where  $R_0$  is a constant diagonal matrix. The first element of the  $D^{-1}(t)$  matrix equals to the sum of squared process outputs (Niu & Fisher 1994),

$$\lim_{t \rightarrow \infty} J^{(0)}(t) = \lim_{t \rightarrow \infty} \sum_{j=1-n}^{t-n} x^2(j) = t \operatorname{E} x^2(t) \quad (7.18)$$

where  $\operatorname{E}$  stands for expectation. The last element, which is the loss function of the  $n$ th order model, is the squared sum of the residuals

$$\lim_{t \rightarrow \infty} J^{(n)}(t) = \lim_{t \rightarrow \infty} \sum_{j=1}^t \varepsilon^2(j) = \lim_{t \rightarrow \infty} \sum_{j=1}^t (x(j) - \varphi^T(j) \hat{\theta}(t))^2 = t \sigma_\varepsilon^2$$

where  $\varepsilon(t)$  represents the residual at time  $t$  and  $\sigma_\varepsilon^2$  is the variance of the process noise.

2. With a constant forgetting factor  $0 < \lambda < 1$ , the loss function matrix  $D = D^{-1}(t)$  converges to a constant matrix which is dependent on the forgetting factor, that is

$$\lim_{t \rightarrow \infty} D^{-1}(t) = \frac{1}{1-\lambda} R_0$$

Here  $N = \frac{1}{1-\lambda}$  is called the "asymptotic memory length", or the "asymptotic data window". It indicates that the information dies away with a time constant of approximately  $N$  sampling intervals. This principle can be extended to variable forgetting factors, where  $\lambda(t)$  means that at time interval  $t$ , the identification algorithm is implementing a data window with an asymptotic length  $N = 1/(1 - \lambda(t))$ . The first element,  $J^{(0)}(t)$ , of the loss function matrix becomes

$$\lim_{t \rightarrow \infty} J^{(0)}(t) = \frac{1}{1-\lambda} \operatorname{E} x^2(t) = N \operatorname{E} x^2(t) \quad (7.19)$$

and the last element,  $J^{(n)}(t)$ , becomes

$$\lim_{t \rightarrow \infty} J^{(n)}(t) = \frac{1}{1-\lambda} E e^2(t) = N E e^2(t)$$

3. From (7.18) and (7.19), it is seen that a forgetting factor is directly related to the values of the loss function matrix. Equation (7.18) can provide an estimate of the expectation value of outputs, which is  $E r^2(t)$ . Setting the first element of  $D^{-1}(t)$  matrix to  $N E r^2(t)$  is equivalent to setting the asymptotic memory length to  $N$  which in turn is equivalent to setting the current forgetting factor value to  $\lambda(t) = 1 - 1/N$ , regardless of its previous value. This leads to remark 4.
4. For ALDI algorithms, the following actions have the same effects in terms of information forgetting
  - (a) introducing a forgetting factor  $\lambda(t)$ .
  - (b) setting the "asymptotic memory length" to  $N = 1/(1 - \lambda(t))$ .
  - (c) setting the loss function matrix to  $D = N R_0$ .

### 7.3.3 Matrix Regularization

Matrix regularization is a way to keep the augmented covariance matrix  $C(t)$  well conditioned (Ljung & Söderström 1983). This is done by specifying an upper and a lower bound on the ACM

$$\alpha_{\min} I \leq C(t) \leq \alpha_{\max} I \quad (7.20)$$

where  $I$  is the identity matrix and  $\alpha_{\min}$  and  $\alpha_{\max}$  are positive constants, with  $0 < \alpha_{\min} < \alpha_{\max} < \infty$ . The lower bound maintains the algorithm's ability to track time-varying parameters, and the upper bound prevents blowing up. Normally the objective of matrix regularization is to improve numerical performance. For RLS algorithms, matrix regularization results in boundedness of the estimation errors. If the input signal  $\varphi(t)$  is persistently exciting and bounded as well, then the convergence of the parameter estimates is guaranteed and the convergence rate is at least exponentially fast (Parkman, Poulsen & Holst 1992). The ALDI algorithm is inherently a LS algorithm, and hence the above results also apply.

In the ALDI algorithm, with the decomposition of  $C(t) = U(t) D(t) U^T(t)$ , only the diagonal loss function matrix needs to be regularized. The parameter matrix  $U = U(t)$  is an upper triangular matrix with all the diagonal elements being unity. Its determinant is always unity and thus does not need to be regularized. Matrix regularization of  $D(t)$  is very trivial

$$\alpha_{\min} \leq D_{ii}(t) \leq \alpha_{\max} \quad \text{for } i=1, \dots, d \quad (7.21)$$

Actually, matrix regularization of the  $D(t)$  matrix can be expressed in an even simpler way, i.e., apply an upper bound to the largest element of the loss function matrix and a lower bound to the smallest element. For AFMA models, the largest element of the loss function matrix is  $J^{(s)}$  or  $L^{(s)}$ , depending on whether  $\sum_{j=1}^d s^2(j)$  is larger than  $\sum_{j=1}^d v^2(j)$ . The smallest element is usually the last element in the loss function matrix, i.e.,  $J^{(n)}(t)$  (Stu & Fisher 1994).

The ALDI formulation/analysis shows that the upper and lower bounds on the loss function matrix, set by matrix regularization, also determine the maximum and minimum "asymptotic memory length" of the algorithm. Once these maxima/minima are

exceeded, the algorithm resets the memory length to the upper/lower bounds which has the same effect as the loss function resetting discussed in Section 7.3.2 and thus *matrix regularization has the dual function of improving matrix condition and limiting maximum memory length of the identification algorithm.*

Matrix regularization puts an upper bound on the largest element of the loss function matrix and a lower bound on the smallest element and thus sets the maximum condition number of the loss function matrix which in turn determines the maximum condition number of the augmented covariance matrix. It thereby guarantees that the condition number of the augmented covariance matrix is always smaller than that specified by matrix regularization. In this sense, matrix regularization improves the condition of the matrices.

## 7.4 Improved Variable Forgetting Method

The relative and absolute forgetting principles discussed in the last two sections provide deeper insight into the information forgetting mechanism in recursive estimator. Information forgetting is equivalent to the problem of appropriately controlling the weight of the new information  $\varphi(t)\varphi^T(t)$  in the augmented covariance matrix or augmented information matrix. The diagonal loss function matrix can be conveniently used to actually control the information assimilation. The above analysis and discussions provide a guideline for the design and interpretation of new and better information forgetting schemes. In this section, the principles discussed in the last two sections are used to improve the existing variable forgetting factor method.

### 7.4.1 The Algorithm

The key to effective information forgetting is to appropriately evaluate the information contained in the new information matrix  $\varphi(t)\varphi^T(t)$ , by comparing it with the accumulated information already contained in the ACM. Different algorithms in the literature use different criteria. However, one of the widely used, also the most obvious, criterion is based on the prediction errors and loss functions. Note that the loss function of the  $n$ th order process model is defined as

$$J^{(n)}(t) = \sum_{j=1}^t \rho(t, j) e^2(j)$$

with

$$\rho(t, j) = \prod_{k=j+1}^t \lambda(k)$$

The recursive form of this formula is given by Fang & Xiao (1988) or Söderström & Stoica (1989) as

$$\begin{aligned} J^{(n)}(t) &= \lambda(t) J^{(n)}(t-1) + e(t) I(t) \\ e(t) &= I(t) / \beta_{t-1}(t) \end{aligned} \quad (7.22)$$

where  $e(t) = z(t) - \varphi^T(t)\hat{\theta}(t)$  is the residual and  $I(t) = z(t) - \varphi^T(t)\hat{\theta}(t-1)$  is the innovation.

As discussed in last section, the information forgetting ability of the algorithm can be characterized by its "asymptotic memory length". If the "asymptotic memory length" is to be kept at  $N$  sampling intervals, which indicates

$$J^{(n)}(t) = N \cdot \sigma_v^2 = J^{(n-1)}(t) \quad (7.23)$$

then from (7.22),

$$N \cdot \sigma_v^2 = N \cdot \lambda(t) \cdot \sigma_v^2 + \frac{\bar{z}(t)}{\beta_{d-1}(t)}$$

which leads to

$$\lambda(t) = 1 - \frac{\bar{z}^2(t)}{N \beta_{d-1} \sigma_v^2} \quad (7.24)$$

This is exactly the result of the variable forgetting factor method proposed by Fortescue et al. (1981). That is, the variable forgetting factor method by Fortescue et al. (1981) maintains a fixed "asymptotic memory length" in order to keep the identification algorithm alert to time varying processes. However, it is obvious that a fixed "asymptotic memory length" may not always give enough emphasis to the newly obtained process information. An improvement which uses a variable memory length is proposed as follows. When the residual  $\bar{z}(t)$  contains more new information than the average, which is represented by the noise variance  $\sigma_v^2$ , then a smaller memory length is chosen to give more weight to the new information. The variable memory length is chosen as

$$N = K \frac{\sigma_v^2}{\bar{z}^2(t)}$$

which leads to

$$\lambda(t) = 1 - \frac{\bar{z}^2(t)}{K \beta_{d-1} \sigma_v^2} \quad (7.25)$$

The noise variance  $\sigma_v^2$  is usually unknown but its estimated value  $\hat{\sigma}_v^2$  can be used instead. A method for estimating the noise variance on-line is given in Chapter 8 or Niu & Fisher (1993).

A typical value of  $K$  is 100, which is equivalent to using an maximum "asymptotic memory length" of 100, or maximum forgetting factor 0.99.

### 7.4.2 On/off Decision for Identification

One situation where information forgetting schemes such as (7.24) and (7.25) would run into a problem is when the process is at steady state and there is no more useful information coming from the process. If the identification were allowed to continue, severe problems like "bursting", "covariance windup", etc., could occur (Åström & Wittenmark 1990, Shah & Chett 1991, Morris et al. 1993). Therefore, a logical procedure is to shut off the identification algorithm and keep the existing parameter estimates.

Monitoring the prediction error is a reasonable mean to decide whether the estimation should be stopped (Seborg, Edgar & Shah 1993). Several methods have been proposed for stopping identification, such as those proposed by Sripada & Fisher (1987) and Yin (1988). However, most of these methods either do not have a sound theoretical basis, or require extensive calculations. The following method is proposed which works together with (7.25) and is very simple to interpret and implement.

The prediction error is a measure of the goodness of the estimated model parameters. When the prediction error is smaller than the standard deviation of the process noise, or

$$\hat{e}^2(t) \leq \sigma_v^2$$

it implies that the data at the current step does not contain new information for updating the parameter estimates. In this case, the identification algorithm should be stopped to prevent "covariance windup". From (7.25), it is seen that  $\hat{e}^2(t) = \sigma_v^2$  is equivalent to a forgetting number 0.99. Therefore the stopping rule simply becomes:

**Stopping Rule.** The identification algorithm in Table 7.1 should be stopped for the current sampling interval when the forgetting factor calculated by (7.25) is greater than  $1 - \frac{1}{\pi^{1/2-1}}$ .

### 7.4.3 Regularization

Matrix regularization in the AUDI algorithm has the dual function of limiting the maximum "asymptotic memory length" and keeping the information accumulation matrix well conditioned. Since the sum of the squared process outputs,  $\hat{\sigma}_t^2$ , can be easily calculated as

$$\hat{\sigma}_t^2 = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{j=1}^t \hat{e}^2(j) \approx \frac{1}{t} \sum_{j=1}^t \hat{e}^2(j)$$

it is assumed to be known. An upper bound on the first element,  $J^{(0)}(t)$ , which is usually the largest element in the loss function matrix (Niu & Fisher 1984), is imposed as

$$J^{(0)}(t) = K \hat{\sigma}_t^2(t), \quad \text{when } J^{(0)}(t) > K \hat{\sigma}_t^2(t) \quad (7.26)$$

where  $K$  is a positive integer and specifies the "asymptotic memory length" of the algorithm. The same value of  $K$  as in formula (7.25) can be used.

The last element,  $J^{(n)}(t)$ , which is usually the smallest element in the loss function matrix, is bounded by

$$J^{(n)}(t) = K \hat{\sigma}_t^2, \quad \text{when } J^{(n)}(t) < K \hat{\sigma}_t^2$$

### 7.4.4 Simulation Example

The following first order SISO difference equation model is used as the representation of the simulated process

$$x(t) + ax(t-1) = bu(t-1) + v(t)$$

where  $\{v(t)\}$  is zero-mean white noise sequence with variance  $\sigma_v^2 = 0.1^2$ . Assume that the parameters are time varying

$$\begin{aligned} a &= \begin{cases} -0.8 & 0 \leq t \leq 100 \\ -0.4 & t > 100 \end{cases} \\ b &= 1.0, \quad t \geq 0 \end{aligned}$$

This is the same example as in Parkum et al. (1982). Using a random binary sequence as the input signal for the first 350 data points and a constant input for the rest of time, the input/output data was obtained and plotted in Figure 7.1. The magnitude of  $v(t)$



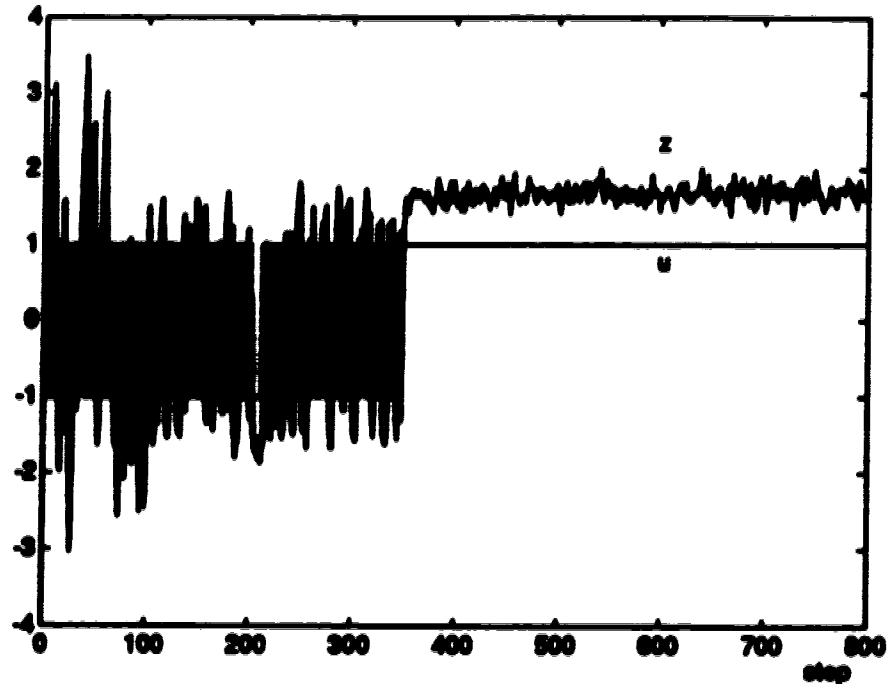


Figure 7.1: Input and Output data

is 1.0. The *ADI* forgetting algorithm (7.25) and the ordinary least-squares algorithm exponential forgetting algorithm are used to identify the same process and the results were shown in Figures 7.2 and 7.3. From Figure 7.2 it is seen that ordinary LS with a forgetting factor  $\lambda=0.90$  gives approximately the same tracking ability as the *ADI* algorithm, but the parameter estimates are very noise sensitive and blow up when no input excitation is present. The *ADI* algorithm, on the other hand, tracks the parameters very quickly and keeps a smooth trajectory. When there is no excitation present, the algorithm just shuts itself down to prevent blow up. The exponential forgetting method with  $\lambda=0.98$  gives a smoother trajectory, but responds very slowly to parameter changes. Note that the forgetting factors calculated by (7.25) are smoothed by a first order filter to avoid false alarms caused by noise spikes.

## 7.5 Conclusions

The *ADI* formulation shows that information forgetting in recursive identification reduces to the simple problem of controlling the weight of the new process information relative to that already included in the augmented information matrix or augmented covariance matrix. This can be done conveniently by manipulating the diagonal loss function matrix. Information forgetting is classified into two categories as relative forgetting and absolute forgetting. This simplifies the interpretation and provides convenient guidelines for designing new information forgetting schemes. An improved information forgetting method is proposed that successfully takes advantage of these guidelines.

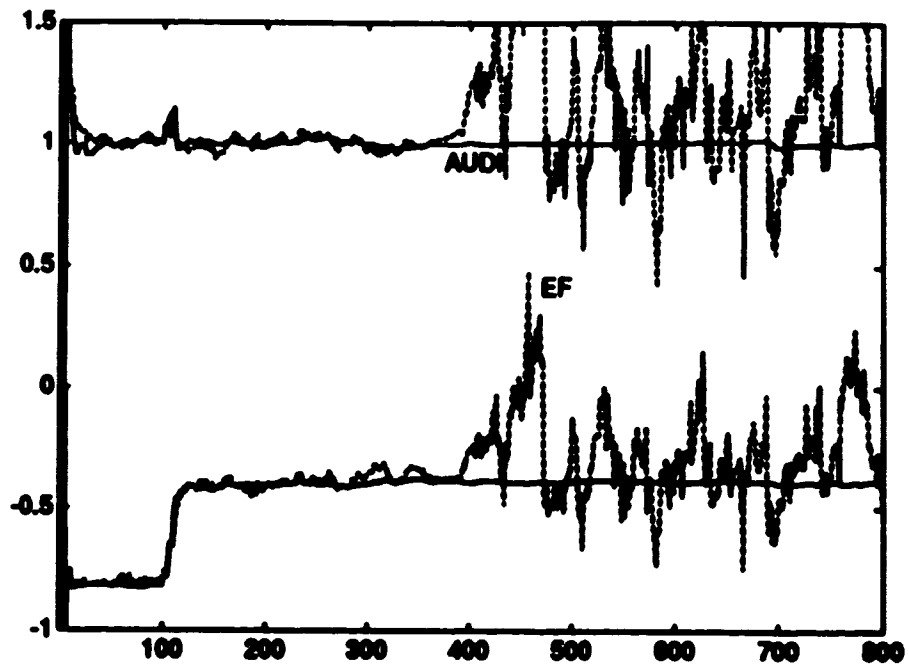


Figure 7.2: Parameter Estimates with AUDI (solid) and EF (dashed) for  $\lambda=0.90$

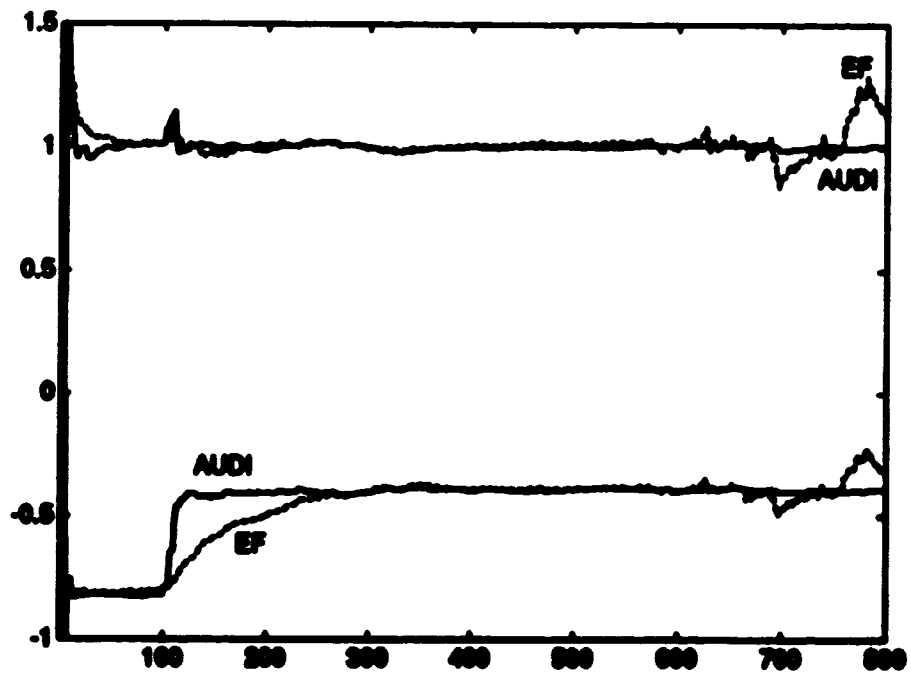


Figure 7.2: Parameter Estimates with AUDI (solid) and EF (dashed) for  $\lambda=0.98$

# Bibliography

- Åström, K., Borisson, U., Ljung, L. & Wittenmark, B. (1977), 'Theory and application of self-tuning regulators', *Automation* 13(5), 457-476.
- Åström, K. J. & Wittenmark, B. (1980), 'Self-tuning controllers based on pole-zero placement', in *IEE Proceedings, Part D*, Vol. 127, pp. 120 - 123.
- Cordero, A. & Mayne, D. Q. (1981), 'Deterministic convergence of a self-tuning regulator with variable forgetting factor', 128(1), 19 - 23.
- Fortescue, T. R., Kershenbaum, L. S. & Ydstie, B. E. (1981), 'Implementation of self-tuning regulator with variable forgetting factor', *Automation* 17(6), 631-635.
- Goodwin, G. C. & Sin, K. S. (1984), *Adaptive Filtering Prediction and Control*, Prentice Hall.
- Goodwin, G. C. & Teoh, E. K. (1983), 'Adaptive control of a class of linear time-varying systems', in *Prepr. IFAC Workshop Adaptive System Control, Signal Processing*, San Francisco.
- Hägglund, T. (1983), 'The problem of forgetting old data in recursive estimation', in *Prepr. IFAC Workshop Adaptive System Control, Signal Processing*, San Francisco. Paper SAC-6.
- Irving, E. (1979), 'New developments in improving power network stability with adaptive control', in *Yale Workshop Application Adaptive Control*, Yale.
- Kulháry, R. (1987), 'Restricted exponential forgetting in real-time identification', *Automation* 23(5), 589-600.
- Kulháry, R. & Kármay, M. (1984), 'Tracking of slowly varying parameters by directional forgetting', in *Proceedings of the 9th IFAC Triennial World Congress*, Vol. 2, Budapest, Hungary, pp. 657 - 662.
- Ljung, L. & Gunnarsson, S. (1990), 'Adaption and tracking in system identification -- a survey', *Automation* 26(1), 7-21.
- Ljung, L. & Söderström, T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass.
- Morris, A. J., Naser, Y. & Wood, R. K. (1986), 'Single and multivariable application of self-tuning controllers', in C. J. Harris & S. A. Billings, eds, 'Self-Tuning and Adaptive Control: Theory and Applications', IEE Control Engineering Series 15.

- Niu, S. & Fisher, D. G. (1993), Information forgetting in recursive identification, in 'Proc. 1993 American Control Conference', San Francisco, pp. 796 - 800.
- Niu, S. & Fisher, D. G. (1994), 'Multiple model least squares method'. Submitted to 1994 American Control Conference.
- Niu, S. & Fisher, D. G. (n.d.), 'Recursive information forgetting based on audi algorithm'. Under Review for Publication in *International Journal of Control*.
- Parkum, J. E., Poulsen, N. K. & Holst, J. (1992), 'Recursive forgetting algorithms', *International Journal of Control* 55(1), 109 - 128.
- Rogers, M. D. (1989), Models and methods for recursive identification, Master's thesis, The University of Alberta, Edmonton, Canada.
- Sælid, S. & Foss, B. (1983), Adaptive controllers with a vector variable forgetting factor, in 'Proceedings of the 22Nd IEEE Conference on Decision and Control', San Antonio, Texas, pp. 1488 - 1494.
- Seborg, D. E., Edgar, T. F. & Shah, S. L. (1986), 'Adaptive control strategies for process control - a survey', *AIChE Journal* 32(6), 881 - 913.
- Shah, S. L. & Cluett, W. R. (1991), 'Recursive least-squares based estimation schemes for self-tuning control', *The Canadian Journal of Chemical Engineering* 69, 89 - 96.
- Sripada, N. R. & Fisher, D. G. (1987), 'Improved least-squares identification', *International Journal of Control* 46(6), 1889 - 1913.
- Vogel, E. F. & Edgar, T. F. (1982), Application of an adaptive pole-zero placement controller to chemical process with variable deadline, in 'Proceedings of the 1982 American Control Conference', Washington, D.C., U.S.A., pp. 536-544.
- Xie, X. & Evans, R. J. (1984), 'Discrete time adaptive control for deterministic time-varying systems', *Automation* 20(3), 309 - 319.
- Yin, G. (1989), 'A stopping rule for Least-Squares identification', *IEEE Transactions on Automatic Control* 34(6), 659-662.

## Chapter 8

# IDENTIFICATION OF NOISE CHARACTERISTICS

**A**n efficient and convenient method is developed for the on-line estimation of the noise variance and the signal-to-noise ratio (SNR) of the process based on the AUDI algorithm developed in previous chapters. The method can be incorporated into most parameter estimation algorithms but is particularly appropriate for the augmented UD identification algorithm. The on-line estimates of the variance, SNR and parameters can be used directly in applications such as filtering and adaptive control.

### 8.1 Introduction

Variance and signal-to-noise ratio (SNR) are widely used to describe the noise characteristics of a process (Isermann 1981). Many applications in process identification and control require knowledge of the process noise, but a suitable on-line measure is not always available. For example, the variable forgetting identification method (Fortescue et al. 1981) calculates the forgetting factor based on the noise variance and prediction error of the parameter estimates, but assumes that the noise variance is constant and known *a priori*. Kalman filter design and minimum variance controller design also require noise statistics and assume they are provided by the user.

In this chapter, a convenient and efficient method is developed for on-line estimation of both the noise variance (or standard deviation) and signal-to-noise ratio. The approach used can be incorporated into most identification algorithms, but it is particularly appropriate for the augmented UD identification algorithm. The AUDI algorithm simultaneously provides the loss functions as well as the parameter estimates for all models from order 1 to a user-specified maximum order  $n$ . With only a little extra calculation, the noise variance and signal-to-noise ratio can also be estimated simultaneously and used directly in adaptive filtering and control applications.

---

<sup>1</sup>A version of this chapter has been accepted for publication as: S. Wu and D. Grant Fisher, 1988, Simultaneous Identification of Process Parameters, Noise Variance and Signal-to-noise Ratio, *IEEE Transactions on Signal Processing*.

## 8.2 Estimation of Noise Variance

Assume the process to be investigated can be represented by the following ARMA model

$$z(t) + a_1 z(t-1) + \dots + a_n z(t-n) = b_1 u(t-1) + \dots + b_n u(t-n) + v(t) \quad (8.1)$$

where  $u(t)$  and  $z(t)$  are the process input and output, respectively;  $v(t)$  is assumed to be white noise and  $n$  is the model order.

Model (8.1) can be rewritten in the least-squares format as

$$z(t) = h^T(t) \theta(t) + v(t) \quad (8.2)$$

where  $h(t)$  is the data vector and  $\theta(t)$  is the parameter vector

$$h(t) = [-z(t-n), u(t-n), \dots, -z(t-1), u(t-1)]^T \quad (8.3)$$

$$\theta(t) = [a_n, b_n, \dots, a_1, b_1]^T \quad (8.4)$$

For least-squares type algorithms, the parameter estimates of process (8.1) are obtained by minimizing the loss function (see Chapter 2)

$$J(t) = \sum_{j=1}^t \epsilon^2(j) = \sum_{j=1}^t [z(j) - h^T(j) \hat{\theta}(t)]^2 \quad (8.5)$$

where  $\epsilon(\cdot)$  is the residual of the estimation and  $\hat{\theta}(t)$  is the model parameter estimate at time  $t$ . The least-squares estimate  $\hat{\theta}(t)$  is given by (2.9) in Chapter 2. When least-squares estimation is exact, the residual  $\epsilon(t)$  asymptotically approaches a zero-mean white noise sequence, and it is shown that

$$\hat{\sigma}^2(t) = \frac{J(t)}{t - \dim \theta} \quad (8.6)$$

is an unbiased estimate of the noise variance. Here  $\dim \theta$  stands for the dimension of the parameter vector and  $t$  is the data length. For a proof see Ljung (1987) or Isermann (1981).

For identification of time-varying processes, a forgetting factor is often introduced to limit the memory length of the identification algorithm. For example, the exponential forgetting algorithm uses a constant forgetting factor and some other algorithms use a variable forgetting factor, see Chapter 7. When a constant forgetting factor,  $\lambda$ , is used, the loss function is redefined as

$$J(t) = \sum_{j=1}^t \lambda^{t-j} \epsilon^2(j) = \sum_{j=1}^t \lambda^{t-j} [z(j) - h^T(j) \hat{\theta}(t)]^2 \quad (8.7)$$

then the statistical property of the loss function is given by

$$\begin{aligned} E\{J(t)\} &= E\left\{\sum_{j=1}^t \lambda^{t-j} \epsilon^2(j)\right\} \\ &= \sum_{j=1}^t \lambda^{t-j} E\{\epsilon^2(j)\} \\ &= \text{Var}(\epsilon(t)) \sum_{j=1}^t \lambda^{t-j} \end{aligned}$$

$$= \frac{1 - \lambda^t}{1 - \lambda} \text{Var}(\varepsilon(t))$$

where  $E$  stands for expectation. For infinite data length, i.e.,  $t \rightarrow \infty$ , we have

$$E\{J(t)\} = \frac{1}{1 - \lambda} \text{Var}(\varepsilon(t))$$

or

$$\hat{\sigma}^2 \approx (1 - \lambda) J(t) \quad \text{for large } t$$

The above results can be summarized as follows

**Summary.** For process (8.1) with zero-mean white noise, using a least-squares estimator for identification

1. With no data forgetting, an unbiased estimates of the noise variance is given by (8.6). In other words, the loss function converges to a linear function of the noise variance, i.e.,

$$J(t) = (t - \dim \theta) \hat{\sigma}^2 \propto t$$

2. With a constant forgetting factor  $0 < \lambda < 1$ , the estimate of the noise variance is given by

$$\hat{\sigma}^2 \approx (1 - \lambda) J(t) \quad \text{for large } t \quad (8.8)$$

In other words, the loss function converges to a constant, which is determined by the noise variance and the forgetting factor. That is

$$J(t) \approx \frac{\hat{\sigma}^2}{1 - \lambda} = \text{const}$$

3. With a variable forgetting factor, the loss function is indefinite and the estimation of the noise variance is much more complicated. However, an approximate method is developed in Section 8.3.1 for use with the augmented UD identification algorithm.

**Example Noise Variance** The following example illustrates the above conclusions, using different noise levels and forgetting factors. Assume the process is represented by the following difference equation model

$$z(t) - 1.5z(t-1) + 0.7z(t-2) = u(t-1) + 0.5u(t-2) + v(t) \quad (8.9)$$

where  $v(t)$  is zero-mean white noise with variance  $\sigma_v^2$ . A data sequence of 1000 points is used. For different noise levels, using a least-squares-type estimator (such as the recursive least-squares algorithm or the AUDI-LS algorithm), the estimated noise variance, calculated using the equation (8.6) and (8.8), are shown in Table 8.1 (with no data forgetting) and Table 8.2 (with data forgetting). From the tables, it is clear that the estimated noise variance is close to its true value.

Since the noise variance is closely related to the loss functions, the trajectory of the loss functions for  $\lambda=1.0$  and  $\lambda=0.98$  with noise variance  $0.5^2$  are plotted in Figure 8.2. It is clear that when  $\lambda=1.0$ , the loss function converges to a linear function of time, while for  $\lambda=0.98$ , the loss function converges to a constant value, which is approximately  $\frac{0.5^2}{1-\lambda} = 2.5$ . In both case, an asymptotic estimate of the noise variance can be obtained by applying (8.6) and (8.8). The estimated standard deviation of the noise (the square root of the noise variance) is shown in Figure 8.1 for both cases. It is found that they are quite close to the true value  $0.5^2$  when  $\lambda=1.0$  but is noisier when  $\lambda=0.98$ . However, both estimates are satisfactory for most applications.

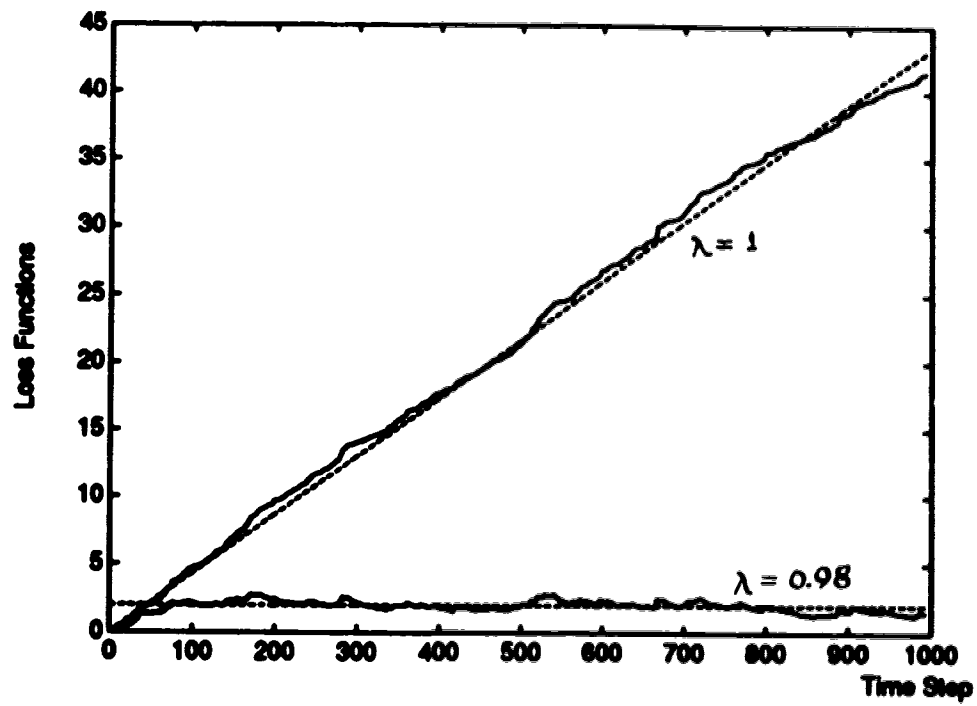


Figure 8.1: Trajectories of Loss Functions

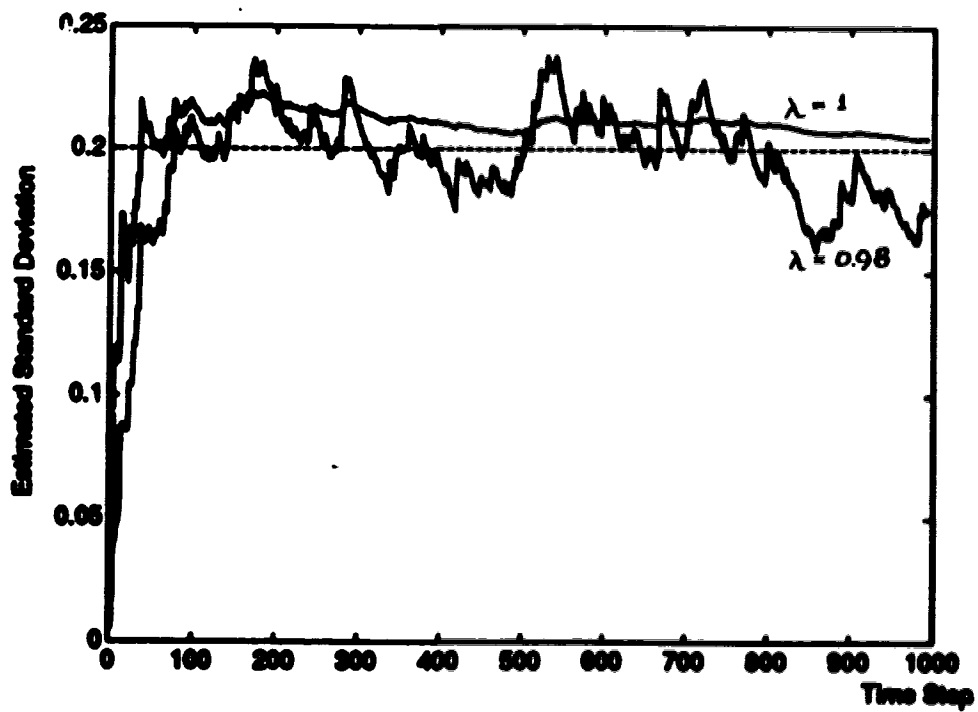


Figure 8.2: Trajectories of Estimated Standard Deviation  $\sigma$  of Noise



Table 8.1: Loss Functions and Estimated Noise Variance (with  $\lambda=1.0$ )

$\sigma^2$ (true value)	0.0	0.1 <sup>2</sup>	0.2 <sup>2</sup>	0.5 <sup>2</sup>	1.0 <sup>2</sup>	2.0 <sup>2</sup>
$\hat{\sigma}^2 = J(t) / (t - \dim \theta)$	0.0022 <sup>2</sup>	0.1023 <sup>2</sup>	0.2046 <sup>2</sup>	0.5114 <sup>2</sup>	1.0230 <sup>2</sup>	2.0461 <sup>2</sup>
$J(t) = \sum \epsilon^2(t)$	0.0060	10.393	41.556	259.66	1038.1	4149.1

Table 8.2: Loss Functions and Estimated Noise Variance (with  $\lambda=0.98$ )

$\sigma^2$ (true value)	0.0	0.1 <sup>2</sup>	0.2 <sup>2</sup>	0.5 <sup>2</sup>	1.0 <sup>2</sup>	2.0 <sup>2</sup>
$\hat{\sigma}^2 = (1 - \lambda) J(t)$	0.0	0.0880 <sup>2</sup>	0.1760 <sup>2</sup>	0.4400 <sup>2</sup>	0.8801 <sup>2</sup>	1.7599 <sup>2</sup>
$J(t) = \sum \lambda^{t-1} \epsilon^2(t)$	0.0	0.3835	1.5488	9.6818	38.728	154.86

### 8.3 Estimation of Signal-to-Noise Ratio

Rewrite the process (8.1) into the following equation error form

$$\hat{A}(q^{-1}) z(t) = \hat{B}(q^{-1}) u(t) + \epsilon(t)$$

or in output error form

$$z(t) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} u(t) + \frac{\epsilon(t)}{\hat{A}(q^{-1})} = \hat{y}(t) + \delta(t) \quad (8.10)$$

where

$$\begin{aligned} \hat{A}(q^{-1}) &= 1 + a_1 q^{-1} + \dots + a_n q^{-n} \\ \hat{B}(q^{-1}) &= b_1 q^{-1} + \dots + b_n q^{-n} \end{aligned}$$

The estimated process output (without noise) and the estimated process noise are  $\hat{y}(t) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} u(t)$  and  $\delta(t) = \frac{\epsilon(t)}{\hat{A}(q^{-1})}$  respectively. The signal-to-noise-ratio is then given by (Ljung 1987)

$$\widehat{\text{SNR}} = \sqrt{\frac{\text{Var}\left(\frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} u(t)\right)}{\text{Var}\left(\frac{\epsilon(t)}{\hat{A}(q^{-1})}\right)}} = \sqrt{\frac{\text{Var}(\hat{y}(t))}{\text{Var}(\delta(t))}} \quad (8.11)$$

To evaluate the  $\widehat{\text{SNR}}$ , we need to first calculate  $\text{Var}(\hat{y}(t))$  and  $\text{Var}(\delta(t))$ . Suppose  $\text{Var}(\epsilon(t))$  is known from identification (e.g., AUD), then  $\delta(t)$  can be expressed in terms of  $\epsilon(t)$ , as

$$\begin{aligned} \delta(t) &= \frac{\epsilon(t)}{\hat{A}(q^{-1})} \\ &= (1 + b_1 q^{-1} + b_2 q^{-2} + \dots) \epsilon(t) \\ &= \epsilon(t) + b_1 \epsilon(t-1) + b_2 \epsilon(t-2) + \dots \end{aligned}$$

With unbiased estimates of  $\hat{\theta}(t)$ ,  $\varepsilon(t)$  is zero-mean white noise, which is identically, independently distributed (i.i.d.). Therefore

$$\begin{aligned}\text{Var}(\hat{\varepsilon}(t)) &= \text{Var}(\varepsilon(t) + h_1\varepsilon(t-1) + h_2\varepsilon(t-2) + \dots) \\ &= \text{Var}(\varepsilon(t)) + \text{Var}(h_1\varepsilon(t-1)) + \text{Var}(h_2\varepsilon(t-2)) + \dots \\ &= (1 + h_1^2 + h_2^2 + h_3^2 + \dots) \text{Var}(\varepsilon(t))\end{aligned}\quad (8.12)$$

This gives an estimate of  $\text{Var}(\hat{\varepsilon}(t))$ . Now consider  $\text{Var}(\hat{y}(t))$ . From (8.10), and according to the orthogonal property of least-squares estimation (Ljung & Söderström 1983), it is known that  $\varepsilon(t)$  is orthogonal to  $h^T(t)\hat{\theta}(t)$ , i.e., the residuals of the parameter estimates are orthogonal to the predicted outputs. Therefore  $\hat{\varepsilon}(t)$  and  $\hat{y}(t)$  can be treated as being independent of each other, which gives

$$\begin{aligned}\text{Var}(z(t)) &= \text{Var}(\hat{y}(t) + \hat{\varepsilon}(t)) \\ &= \text{Var}(\hat{y}(t)) + \text{Var}(\hat{\varepsilon}(t))\end{aligned}$$

or

$$\text{Var}(\hat{y}(t)) = \text{Var}(z(t)) - \text{Var}(\hat{\varepsilon}(t)) \quad (8.13)$$

And combining (8.12) and (8.13), the desired estimate of the signal-to-noise ratio is given by

$$\widehat{\text{SNR}} = \sqrt{\frac{\text{Var}(\hat{y}(t))}{\text{Var}(\hat{\varepsilon}(t))}} = \sqrt{\frac{\text{Var}(z(t)) - \text{Var}(\hat{\varepsilon}(t))}{\text{Var}(\hat{\varepsilon}(t))}} = \sqrt{\frac{\text{Var}(z(t))}{\text{Var}(\hat{\varepsilon}(t))}} - 1 \quad (8.14)$$

where

$$\begin{aligned}\text{Var}(\hat{\varepsilon}(t)) &= (1 + h_1^2 + h_2^2 + h_3^2 + \dots) \text{Var}(\varepsilon(t)) \\ \text{Var}(\varepsilon(t)) &= \frac{J(t)}{t - \dim\theta} \\ \text{Var}(z(t)) &= \frac{\sum z^2(t)}{t - 1}\end{aligned}$$

Note that calculation of the signal-to-noise ratio involves the deconvolution of  $1/\hat{A}(q^{-1}) = 1 + h_1q^{-1} + h_2q^{-2} + \dots$ . Actually, after the parameters converge, the  $\hat{A}(q^{-1})$  polynomial doesn't change much, so that the sum  $1 + h_1^2 + h_2^2 + \dots$  is approximately constant and hence doesn't need to be calculated at every time step. Alternatively, the change in the SNR during periods when  $\hat{A}$  does not significantly is given by  $\sqrt{\frac{\text{Var}(z(t))}{\text{Var}(\hat{\varepsilon}(t))}} - 1$  useful for process monitoring and fault detection.

**Example Estimation of SNR** Consider the estimation of the SNR of the same process as in Example 1. Assume that  $\sum_{j=1}^t z^2(j)$  and the loss function  $J(t) = \sum_{j=1}^t \varepsilon^2(j) \sum_{j=1}^t [z(j) - h^T(j)\hat{\theta}(t)]^2$  are provided. The calculated  $\widehat{\text{SNR}}$  values, for different noise level and forgetting factors, are given in Table 8.3 ( $\lambda=1.0$ ) and Table 8.4 ( $\lambda=0.98$ ). The estimated variance  $\hat{\sigma}$  and  $\widehat{\text{SNR}}$ s are satisfactorily close to their true values in all cases.

Figure 8.3 shows the trajectory of the estimated SNR. Again, it converges to the true SNR value but is noisy for  $\lambda < 1$ . In many applications it would be desirable to smooth the SNR by heavy filtering or averaging.

Table 8.3: Estimated SNR (with  $\lambda=1.0$ )

$\sigma$	$\hat{\sigma}$	$a_1$	$a_2$	$b_1$	$b_2$	$J_0(t)$	$J_n(t)$	SNR	SNR
0.0	0.0022	-1.5000	0.7000	1.0000	0.5000	17831	0.0050	$\infty$	635.28
0.1	0.1023	-1.4998	0.6998	1.0025	0.5030	17663	10.393	14.480	13.962
0.2	0.2046	-1.4994	0.6996	1.0050	0.5057	18399	41.556	7.2402	7.0004
0.5	0.5114	-1.4975	0.6984	1.0123	0.5165	20537	259.66	2.8961	2.8166
1.0	1.0230	-1.4936	0.6978	1.0242	0.5342	27530	1036.1	1.4480	1.4125
2.0	2.0441	-1.4888	0.6960	1.0462	0.5668	54323	4149.1	0.7240	0.6930

Table 8.4: Estimated SNR (with  $\lambda=0.98$ )

$\sigma$	$\hat{\sigma}$	$a_1$	$a_2$	$b_1$	$b_2$	$J_0(t)$	$J_n(t)$	SNR	SNR
0.0	0.0002	-1.5000	0.7000	1.0000	0.5000	839.35	0.0000	$\infty$	115100
0.1	0.0880	-1.4977	0.6947	1.0142	0.4908	864.33	0.3635	14.480	18.470
0.2	0.1760	-1.4974	0.6970	1.0280	0.4682	1180.2	1.5466	7.2402	9.2328
0.5	0.4400	-1.4958	0.6942	1.0701	0.4180	1246.8	9.6618	2.8961	3.6602
1.0	0.8801	-1.4958	0.6792	1.1367	0.3820	1507.3	38.728	1.4480	1.8280
2.0	1.7599	-1.5056	0.6901	1.2647	0.3376	2691.3	154.86	0.7240	0.9416

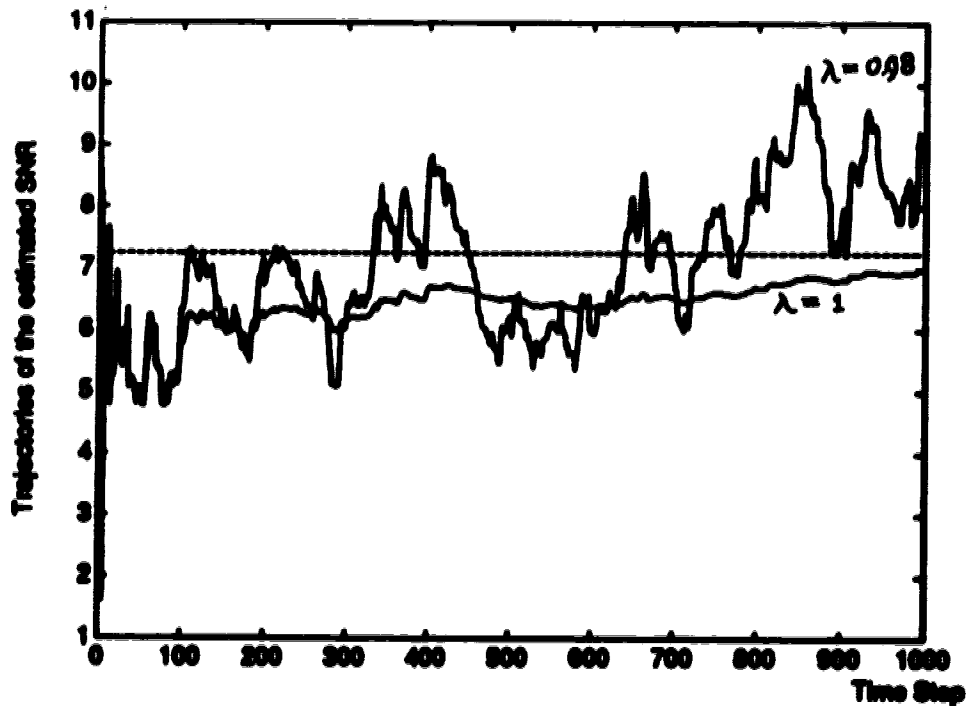


Figure 8.3: Trajectories of Estimated Signal-to-Noise Ratios

### 8.3.1 The Augmented UD Identification Algorithm

For recursive identification, if a variable forgetting factor were used, variance estimation would be much more difficult than the cases discussed above. In addition, for SNR estimation, the sum of the squared residuals (loss function) and the sum of the squared outputs would have to be calculated at the same time. This represents extra computation if the recursive least-squares algorithm (Ljung & Söderström 1983, Söderström & Stoica 1989) is used. However, the augmented UD identification algorithm requires very few extra calculations to include estimation of the noise variance and SNR.

The augmented UD identification algorithm (see Chapters 2 and 3) is a simultaneous order and time recursive algorithm with excellent numerical properties and low computational requirements. The most significant feature of the AUDI algorithm is that it simultaneously identifies the model parameters and the corresponding loss functions of the models from order 1 to a user-specified maximum order  $n$  (the multiple model structure). This feature provides the basis for simultaneous model order and/or time delay identification and also provides the basis for variance and SNR estimation. Refer to Chapters 2 and 3 for detailed information on the AUDI algorithm. The information related to variance and signal-to-noise ratio estimation can be summarized as follows

- The process parameter estimates of all model orders from 1 to  $n$  are contained in the parameter matrix  $U$  and can be found in columns 3, 5, 7 up to  $2n + 1$ . *I.e.*, the parameter estimates for the  $i^{\text{th}}$  order model are found in the  $(2i + 1)^{\text{th}}$  column of the  $U$  matrix, with  $i \in [1, n]$ .
- The loss functions for models from order 1 to  $n$  are given by diagonal elements 3, 5, 7 up to  $2n + 1$  of the matrix  $D$ . *I.e.*, the loss function of the  $i^{\text{th}}$  order model is the  $(2i + 1)^{\text{th}}$  diagonal element of the matrix  $D$ .
- The first element in  $D$ , *i.e.*,  $J^{(0)}(t)$ , is the sum of the squared process outputs up to time  $t$ .

### 8.3.2 Estimation of Signal-to-Noise Ratio with AUDI

With different forgetting factors, the  $n^{\text{th}}$  order loss function  $J^{(n)}(t)$  and the sum of the squared outputs  $J^{(0)}(t)$  are given by the expressions in Table 8.5. The loss function

Table 8.5: Loss functions and squared sum of outputs

$\lambda$	$J^{(n)}(t)$ (loss function)	$J^{(0)}(t)$ (sum of squared outputs)
$\lambda=1$	$J^{(n)}(t) = \sum_{j=1}^t e^2(j)$	$J^{(0)}(t) = \sum_{j=1}^t x^2(j)$
$0 < \lambda < 1$	$J^{(n)}(t) = \sum_{j=1}^t \lambda^{t-j} e^2(j)$	$J^{(0)}(t) = \sum_{j=1}^t \lambda^{t-j} x^2(j)$
$\lambda=\lambda(t)$	$J^{(n)}(t) = \sum_{j=1}^t \left( \prod_{i=j}^{t-1} \lambda(i) \right) e^2(j)$	$J^{(0)}(t) = \sum_{j=1}^t \left( \prod_{i=j}^{t-1} \lambda(i) \right) x^2(j)$

matrix  $\mathcal{D}$  provides all the values required for estimation of the noise variance and signal-to-noise ratio.

First, reconsider the signal-to-noise ratio in Section 8.3

$$\widehat{\text{SNR}} = \sqrt{\frac{\text{Var}(x(t))}{\text{Var}(\hat{e}(t))} - 1} = \sqrt{\frac{1}{1 + h_1^2 + h_2^2 + \dots} \cdot \frac{\text{Var}(x(t))}{\text{Var}(e(t))} - 1}$$

From Table 8.5,

$$\frac{\text{Var}(x(t))}{\text{Var}(e(t))} \approx \frac{J^{(o)}(t)}{J^{(n)}(t)} \quad \text{for large } t$$

Therefore, for all values of the forgetting factor  $\lambda$ , an approximate estimate of the signal-to-noise ratio is conveniently given by

$$\widehat{\text{SNR}} = \sqrt{\frac{1}{1 + h_1^2 + h_2^2 + \dots} \cdot \frac{J^{(o)}(t)}{J^{(n)}(t)} - 1} \quad (8.15)$$

In situations where only a rough approximation of the signal-to-noise ratio is needed or where the objective is to detect changes in the signal-to-noise ratio, the following approximation can be used

$$\widehat{\text{SNR}} \approx \sqrt{\frac{J^{(o)}(t)}{J^{(n)}(t)} - 1} \quad (8.16)$$

Now consider estimation of the noise variance as discussed in Section 8.2. For a forgetting factor  $\lambda=1$  and constant forgetting factor  $0 < \lambda < 1$ , equation (8.6) and (8.8) can still be used and the loss function  $J^{(o)}(t)$  is the appropriate loss function obtained from the loss function matrix  $\mathcal{D}$ . For identification with variable forgetting factors  $\lambda(t)$ , the calculation is more complicated but an approximation can be calculated as follows, using the information provided in the loss function matrix.

For large data length  $t$ , from Table 8.5,

$$\begin{aligned} \frac{J^{(o)}(t)}{J^{(n)}(t)} &= \frac{\mathbb{E} \left( \sum_{j=1}^t \left[ \left( \prod_{i=j}^{t-1} \lambda(i) \right) e^2(j) \right] \right)}{\mathbb{E} \left( \sum_{j=1}^t \left[ \left( \prod_{i=j}^{t-1} \lambda(i) \right) x^2(j) \right] \right)} \\ &= \frac{\text{Var}(x(t)) \sum_{j=1}^t \left[ \prod_{i=j}^{t-1} \lambda(i) \right]}{\text{Var}(e(t)) \sum_{j=1}^t \left[ \prod_{i=j}^{t-1} \lambda(i) \right]} \\ &= \frac{\text{Var}(x(t))}{\text{Var}(e(t))} \end{aligned}$$

which means

$$\text{Var}(x(t)) \approx \frac{J^{(n)}(t)}{J^{(o)}(t)} \text{Var}(e(t)) \quad (8.17)$$

Note that  $\text{Var}(z(t))$  can be estimated using the formula  $\frac{1}{t - \text{dim} \theta} \sum_{j=1}^t z^2(j)$ , where the calculation of  $\sum_{j=1}^t z^2(j)$  is added to AUDI algorithm.

The two general formulae (8.15) and (8.17) are used with the AUDI algorithm to estimate the noise variance and signal-to-noise ratio. With no data forgetting, or with constant and variable forgetting factors close to 1.0, the two formulae give accurate estimates as shown by the results shown in Table 8.1 to Table 8.4. For small forgetting factors, the two formulae provide only an approximation to the noise level.

## 8.4 Conclusion

This chapter presents a simple but general method for adding the simultaneous estimation of noise variance and signal-to-noise ratio to a parameter estimation algorithm. Very little extra computation is required if the AUDI algorithm is used and the method is useful for on-line identification using constant or variable forgetting factors. The on-line estimates of the variance and S/N ratio can then be used in applications such as identification, filtering and control.

# Bibliography

- Fortescue, T. R., Kershenbaum, L. S. & Ydstie, B. E. (1981), 'Implementation of self-tuning regulator with variable forgetting factor', *Automatica* 17(6), 831-835.
- Isermann, R. (1981), *Digital Control Systems*, Springer-Verlag, New York.
- Ljung, L. (1987), *System Identification: Theory for the User*, Prentice Hall, Englewood Cliffs, New Jersey.
- Ljung, L. & Söderström, T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass.
- Niu, S. & Fisher, D. G. (1993), 'Simultaneous estimation of process parameters, noise variance and signal-to-noise ratio', *IEEE Transactions on Signal Processing*. Accepted for publication.
- Söderström, T. & Stoica, P. (1989), *System Identification*, Prentice Hall, Englewood Cliffs, New Jersey.

## Chapter 9

# PRACTICAL CONSIDERATIONS

**A**lthough the theory of process identification is becoming more and more mature, there are still many difficulties encountered in its applications to actual processes. This chapter provides an overview of some practical considerations and two application examples are presented.

### 9.1 Practical Considerations

As defined in Chapter 1, the objective of process identification is to construct, based on input/output data from the system, a mathematical model which reproduces the external characteristics of the true process. The general procedure for implementing process identification was discussed briefly in Chapter 1 and depicted in Figure 1.2. After suitable plant I/O data has been obtained, it is possible to use an identification algorithm such as *ALDI* to produce estimates of the model parameters, model order etc. As indicated by the discussion in this thesis, the conversion of plant I/O data into a plant model is non-trivial. Fortunately, however, for the engineers in the plant, the details of the identification algorithm can be "hidden" in a well defined software package. Given the availability of a suitable identification package, the role of the plant engineer then focuses on a number of related activities such as process analysis, experimental design and implementation, data analysis and model verification. A partial list of "practical factors", which can be used as a checklist for preparing and conducting the actual process identification, is attached in Appendix C. In general, factors such as the following require careful consideration.

1. **PURPOSE OF IDENTIFICATION.** The purpose determines which model set should be used, how accurate the model should be and what type of identification algorithm is required. A brief summary for some common applications is given in Table 9.1.
2. **KNOWLEDGE OF THE PROCESS.** *A priori* knowledge is very helpful in the design of identification experiments. Knowledge of the process may include exact or approximate values (or bounds) for: the steady-state gain, integral components, dominant time constant, the cut-off frequency, time delay, time-to-steady-state and so on. In

---

<sup>1</sup>Some of the material in this chapter has been included in a paper: P. Banerjee, S. Shah, S. Niu and D. Grant Fisher, Identification of Process Dynamics for the Shell Benchmark Problem, 1992 Shell Workshop on Process Identification.



**Table 9.1: Dependence of Identification on Application Purpose**

<b>Purpose</b>	<b>Model Set</b>	<b>Accuracy</b>	<b>Identification Method</b>
<b>Model Verification</b>	linear, continuous	medium/high	off-line, correlation
<b>Self-tuning of controller parameters</b>	parametric/nonparametric		
	linear, nonparametric	medium	off-line, correlation
<b>Adaptive control</b>	continuous-time		
	linear, parametric	medium	on-line, closed-loop
	discrete-time		
<b>Process Monitoring &amp; fault diagnosis</b>	linear/nonlinear	high	on-line
	parametric		
<b>Prediction</b>	linear/nonlinear	high	on-line/off-line
	parametric		

addition, some characteristics of the process such as the process linearity, time-variance, noise characteristics (colored/white noise, signal-to-noise ratio) etc., may also be needed. Knowledge of the environment where the identification is to be conducted is also very important, e.g., what external disturbances may be present.

3. **DESIGN AND IMPLEMENTATION OF THE EXPERIMENT.** Careful consideration must also be given to important factors such as the selection of input signal, sampling time, data length, on-line or off-line, open-loop or closed-loop, elimination of bad-data, data storage.

It is not difficult to identify a process under "ideal" (textbook) conditions especially in simulation studies which are characterized by:

- linear process (often known to user), i.e., no structural mismatch.
- strong and persistent input excitation.
- no feedback loop.
- single input and single output.
- low noise level (high signal-to-noise ratio).
- no measurement errors or biases.

Unfortunately, many important practical applications do not have these ideal characteristics. The "goodness" of a process identification methodology and algorithm depends on how difficult a problem it can handle. In other words, how robust it is in the presence of non-idealities such as noise and nonlinearity.

### 9.1.1 Model Set

Choosing the appropriate model set is an important decision. In practice, model set selection is quite application specific and there is no generally applicable approach. However, some guidelines are provided as follows:

1. **IDENTIFIABILITY.** The identifiability of a process is usually dependent on the model set selected. Therefore, in choosing the model set for a specific process, the identifiability, including both open-loop identifiability and closed-loop identifiability, must be evaluated.
2. **FLEXIBILITY.** The model set must be flexible enough to be able to represent the system dynamics over the required range of operating conditions. The flexibility of a model set is often related to the number of model parameters and how the parameters enter the model. For example, the following FIR model

$$z(t) = B(z^{-1}) u(t) + v(t)$$

is not suitable for systems with slow dynamics because of the large number of parameters required, i.e., it is not a flexible model set for slow processes.

3. **PARSIMONY.** The model set should be selected in such a way that an excessive number of parameters is not required. As shown in Chapter 4, if an increase in the number of model parameters does not result in significant decrease in the

corresponding loss functions, then more parameters are not justified. In another words, once the model set is large enough to contain the true system, it is usually not worthwhile to further increase the number of parameters to obtain further small reductions in the loss functions.

4. **ALGORITHM COMPLEXITY.** Generally speaking, identification algorithms for real applications should not be overly complicated, especially for real-time applications. The complexity of the calculation of the output prediction  $\hat{z}(t)$  and its gradient  $\varphi(t)$  is determined by the number of model parameters and model structure. Therefore a simple model set is often preferred. Usually this is a compromise between model simplicity and model accuracy.

For single-input single-output systems, a general linear, discrete-time model structure is given by

$$A(z^{-1})z(t) = \frac{B(z^{-1})}{F(z^{-1})}u(t) + \frac{D(z^{-1})}{C(z^{-1})}v(t) \quad (9.1)$$

where  $u(t)$  and  $z(t)$  are the system input and output respectively, and  $v(t)$  is zero-mean white noise. The polynomials are defined as

$$\begin{cases} A(z^{-1}) &= 1 + a_1 z^{-1} + \dots + a_{n_a} z^{-n_a} \\ B(z^{-1}) &= b_1 z^{-1} + \dots + b_{n_b} z^{-n_b} \\ C(z^{-1}) &= 1 + c_1 z^{-1} + \dots + c_{n_c} z^{-n_c} \\ D(z^{-1}) &= 1 + d_1 z^{-1} + \dots + d_{n_d} z^{-n_d} \\ F(z^{-1}) &= 1 + f_1 z^{-1} + \dots + f_{n_f} z^{-n_f} \end{cases}$$

Selection of a SISO model set then reduces to determining which of the  $A(z^{-1})$ ,  $B(z^{-1})$ ,  $C(z^{-1})$ ,  $D(z^{-1})$  and  $F(z^{-1})$  polynomials should be used. For example, the famous Box-Jenkins model (Box & Jenkins 1970) is

$$z(t) = \frac{B(z^{-1})}{F(z^{-1})}u(t) + \frac{D(z^{-1})}{C(z^{-1})}v(t) \quad (9.2)$$

The following three models are widely used in applications.

#### 1. LEAST-SQUARES MODEL or EQUATION ERROR MODEL.

$$A(z^{-1})z(t) = B(z^{-1})u(t) + v(t) \quad (9.3)$$

This is the most widely used model set, and is particularly appropriate when the process noise term is small. In addition to model parsimony, this model set also has many other advantages such as robustness, simplicity, fast and global convergence. When the system noise is large, the model order can be increased to compensate for the effect of process noise. This is also the model that was used to derive the ALDI algorithm in previous chapters. Once the model set is selected, the model structure (delay and order) can be determined using the ALDI algorithm. The ALDI algorithm can be used to determine an appropriate model order to best represent the process dynamics, including the compensation of noise dynamics, for processes with either large or small noise components.

#### 2. OUTPUT ERROR MODEL.

$$z(t) = \frac{B(z^{-1})}{F(z^{-1})}u(t) + v(t) \quad (9.4)$$

This is another widely used model set. The advantage of this model set is that, as long as the noise is uncorrelated with the system input, the identification of  $B(z^{-1})$  and  $F(z^{-1})$  is not affected by the noise  $v(t)$ .

3. In control applications, the following CARMA model (See Chapter 4)

$$A(z^{-1})x(t) = B(z^{-1})u(t) + D(z^{-1})v(t) \quad (9.5)$$

or CARIMA model

$$A(z^{-1})\Delta x(t) = B(z^{-1})\Delta u(t) + D(z^{-1})v(t) \quad (9.6)$$

is very popular since this model set has more flexibility, *e.g.*, to represent different process disturbances. In addition, the order of the polynomial  $D(z^{-1})$  does not affect the convergence of the parameters of  $A(z^{-1})$  and  $B(z^{-1})$ .

Model selection requires experience and often some trial and error. If the selected model set can not meet the practical requirements, some other model set should then be used.

### 9.1.2 Initial Conditions for Identification

All recursive identification algorithms need some initial conditions for start up. For example, the least-squares type algorithms need initial conditions on  $\hat{\theta}(0)$  and  $P(0)$ , while the AUDI type algorithms require initial conditions on  $C(0)$ . Usually, the initial values of the variables are unknown. In this case, the initial conditions are chosen as

$$\begin{cases} \hat{\theta}(0) = \epsilon \\ P(0) = \sigma^2 I \end{cases} \quad (9.7)$$

for least-squares type algorithms and

$$C(0) = \sigma^2 I \quad (9.8)$$

for AUDI-type algorithms. Usually,  $\epsilon$  is chosen as a small positive number and  $\sigma$  a large integer. Common practice is to use 1000 for  $\sigma$  and zero for  $\epsilon$ , but the best choice depends on the application.

The initialization of the AUDI algorithm in (9.8) is equivalent to choosing

$$U(0) = I, \quad D(0) = \sigma^2 I$$

which means that the initial values of all the model parameters from order 1 to  $n$  are taken as zeros and the initial loss functions very small values. An interpretation is as follows: a larger value of  $\sigma$  implies less confidence in the initial parameter estimate  $\hat{\theta}(0)$ , which causes the algorithm to put more weight on the incoming input/output data when updating the parameter estimates. In terms of information forgetting as discussed in Chapter 7, increasing  $\sigma$  (which determines the value of  $D(0)$ ) implies that the algorithm forgets old information (including  $\hat{\theta}(0)$ ) quickly and updates the parameter estimates quickly based on the new input/output data.

For algorithms that are not globally convergent, poor initial conditions may affect the convergence of the algorithm. For example, the convergence of the RIV and AUDI-IV algorithms (chapter 8) is very sensitive to initial conditions. Therefore in practice, the FLS or AUDI-LS method is usually used for the first 10 — 20 steps to provide reasonable initial values for RIV or AUDI-IV.

### 9.1.3 Model Conversion

The model obtained by system identification is usually in discrete-time form. In practice, it is sometimes necessary to convert the discrete-time model into its continuous-time equivalent and vice versa. This conversion is usually done using a bi-linear transformation and can be carried out easily on a computer (Fang & Xiao 1988).

#### 1. DISCRETE-TIME TO CONTINUOUS-TIME.

Assume that the discrete-time model (9.3) is in the following general form

$$G_d(z^{-1}) = \frac{b_0 + b_1 z^{-1} + \dots + b_{m-1} z^{-(m-1)} + b_m z^{-m}}{1 + a_1 z^{-1} + \dots + a_{n-1} z^{-(n-1)} + a_n z^{-n}} z^{-d}, \quad (m \leq n) \quad (9.9)$$

and use the bi-linear transformation

$$z^{-1} = \frac{2 - T_0 s}{2 + T_0 s}$$

The following continuous-time transfer function model is then obtained

$$G_c(s) = \frac{\beta_n s^n + \beta_{n-1} s^{n-1} + \dots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1} s^{n-1} + \dots + \alpha_1 s + \alpha_0} e^{-rs} \quad (9.10)$$

where  $r = dT_0$  and  $T_0$  is the sampling interval. The coefficients are given by

$$\begin{cases} \beta_{n-j} = \frac{1}{a_n} \sum_{i=0}^m b_i \omega_{ij}, & j=0, 1, \dots, n \\ \alpha_{n-j} = \frac{1}{a_n} \sum_{i=0}^n a_i \omega_{ij}, & j=1, 2, \dots, n \\ \alpha_n = \sum_{i=0}^n a_i \omega_{i0} \\ \alpha_0 \triangleq 1 \end{cases} \quad (9.11)$$

where the  $\omega_{ij}$  terms are calculated as follows

$$\begin{cases} \omega_{ij} = \sum_{k=0}^j (-1)^{i-k} C_{n-i}^{k-1} C_i^{k-1} 2^j T_0^{n-j}, & i, j=0, 1, \dots, n \\ C_P^Q \triangleq \begin{cases} 0, & Q > P \text{ or } Q < 0 \\ 1, & Q = P \text{ or } Q = 0 \\ \frac{P(P-1)\dots(P-Q+1)}{Q!}, & Q < P \end{cases} \end{cases}$$

**Example Model Conversion From Discrete to Continuous** Assume the discrete-time model is

$$G_d(z^{-1}) = \frac{0.0950226 + 0.00804977z^{-1} - 0.0889728z^{-2}}{1 - 1.8000z^{-1} + 0.819004z^{-2}}$$

and the sampling interval  $T_0 = 0.1$  second. Then using equation (9.11), the continuous-time transfer function is

$$G_c(s) = \frac{9.54794 \times 10^{-3} s^2 + 2.0s + 2.0}{s^2 + 2.00001s + 2.00049}$$

which is very close to the true continuous-time model

$$G_c(s) = \frac{2s + 2}{s^2 + 2s + 2}$$

## 2. CONTINUOUS-TIME TO DISCRETE-TIME

Assume that the continuous-time transfer function model is

$$G_c(s) = \frac{\beta_m s^m + \beta_{m-1} s^{m-1} + \dots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1} s^{n-1} + \dots + \alpha_1 s + \alpha_0} e^{-\tau s}, \quad (m \leq n) \quad (9.12)$$

With the following bi-linear transformation

$$s = \frac{2}{T_0} \frac{1 - z^{-1}}{1 + z^{-1}}$$

the discrete-time transfer function model is

$$G_d(z^{-1}) = \frac{b_0 + b_1 z^{-1} + \dots + b_{n-1} z^{-(n-1)} + b_n z^{-n}}{1 + a_1 z^{-1} + \dots + a_{n-1} z^{-(n-1)} + a_n z^{-n}} z^{-d}, \quad (m \leq n) \quad (9.13)$$

where  $d = \tau/T_0$  and  $T_0$  is the sampling interval. The coefficients are given by

$$\begin{cases} b_{n-j} = \frac{1}{a_0} \sum_{i=0}^n \beta_i \left(\frac{2}{T_0}\right)^j \omega_{ij}^*, & j=0, 1, \dots, n \\ a_{n-j} = \frac{1}{a_0} \sum_{i=0}^n \alpha_i \left(\frac{2}{T_0}\right)^j \omega_{ij}^*, & j=0, 1, \dots, n-1 \\ a_0 = \sum_{i=0}^n \alpha_i \left(\frac{2}{T_0}\right)^i \omega_{in}^* \\ a_n \triangleq 1 \end{cases}$$

and

$$\begin{cases} \omega_{ij}^* = \sum_{k=0}^j (-1)^{i-k} C_{n-i}^{j-k} C_i^k, & i, j=0, 1, 2, \dots, n \\ C_i^0 = \text{same as above.} \end{cases}$$

An example of the conversion from continuous-time to discrete-time is as follows.

**Example Model Conversion From Continuous to Discrete** Assume the continuous-time model is given by

$$G_c(s) = \frac{-0.15625s^2 - 6.25s + 187.5}{s^3 + 3.75s + 25}$$

and set the sampling interval to  $T_0 = 0.1$  second. The corresponding discrete-time model is

$$G_d(z^{-1}) = \frac{1.49012 \times 10^{-8} + 1.0z^{-1} + 0.5z^{-2}}{1 - 1.5z^{-1} + 0.7z^{-2}}$$

The exact discrete-time model is

$$G_d = \frac{z^{-1} + 0.5z^{-2}}{1 - 1.5z^{-1} + 0.7z^{-2}}$$

From the above two examples, it is seen that the bi-linear transformation of models from discrete-time to continuous-time and back is straightforward and very accurate. The conversion can be easily implemented in a computer and thus both continuous-time model and discrete-time model can easily be made available if either one of them is known.

## 9.2 Identification of a Distillation Column

This example is from the 1992 Canadian Society of Chemical Engineering (CSChE) model identification workshop which focused on the identification of the distillation column as depicted in Figure 9.1. The problem was prepared by industrial personnel and is based on an actual industrial process.

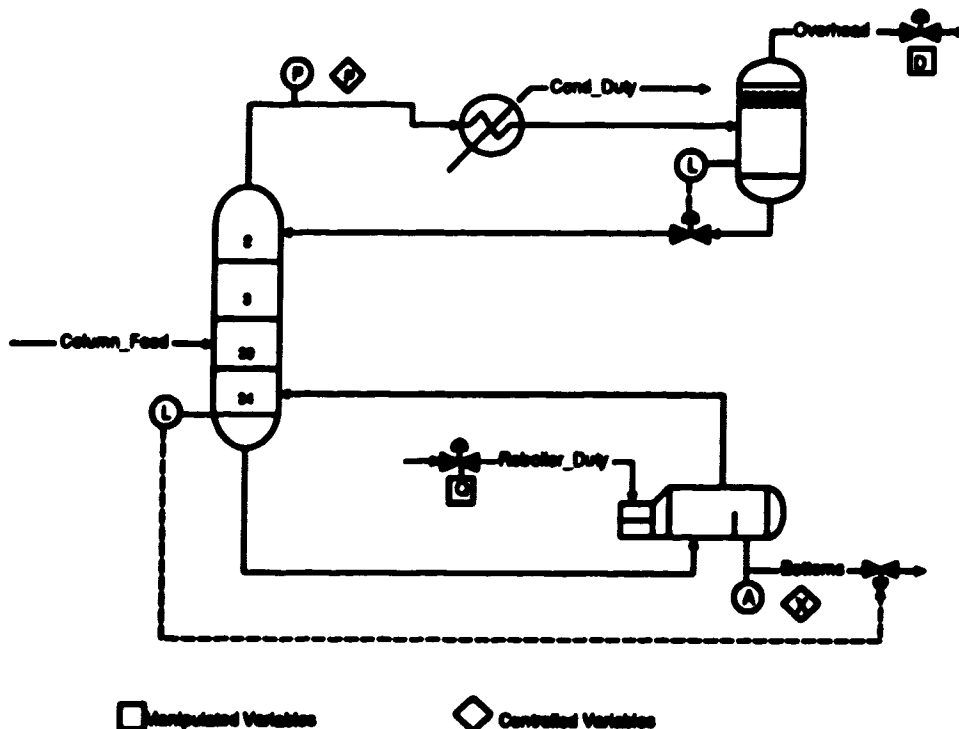


Figure 9.1: Diagram of A Distillation Column

The main purpose of the distillation unit is to remove a light key impurity from the feed stream so that it does not reach the reaction system downstream of the column. The controlled variables are

1. Impurity level in bottoms  $X$ . Since the purpose of the column is to remove the impurity in the feed, it is necessary to maintain the impurity level at a value acceptable by the downstream reactor. An on-line analyser is available to provide  $X(t)$  at every control interval.
2. Column pressure  $P$ . The condenser heat removal rate is limited due to cooling water temperature constraints. The column pressure will float with changing cooling water temperature and supply pressure. If the column pressure is higher than 3200, a pressure relief system will kick in and dump the column contents to a flare system. To prevent this undesirable event from happening, the column pressure should be maintained below this release pressure.

To achieve the desired control objective, the following two variables can be manipulated

1. The re-boiler duty,  $Q$ . Heat input to the column is provided by a steam-heated reboiler, and is controlled by a reboiler heat-duty controller which manipulates the

steam flowrate to achieve the desired duty given the temperature and pressure of the steam.

2. The overhead vapor flow-rate  $D$ . Heat removal from the column is provided by a partial overhead condenser, which uses process cooling water to remove the heat.

Typical operating conditions are  $Q=2500$ ,  $D=20$  and  $P=2800$ ,  $X=500$ . Constraints on  $D$  and  $Q$  are 2000 - 3000 and 10 - 30, and the typical setpoint range of  $P$  and  $X$  are 2700 - 2900 and 250 - 1000.

### 9.2.1 Preliminary Analysis

The problem is formulated as a two-input two-output multivariable system. The purpose of identification is to provide a mathematical model for model-based control of this system or for tuning of controller parameters. Therefore the main purpose of the identification is to find a stable model that can adequately represent the process input/output relationship. The internal mechanics of the process are not of interest for this control application. The order of the actual industrial process might be very high, but it is not necessary that the model be the same order as the process. An appropriate model order must be obtained by identification, e.g., by investigating the loss functions of models with different model order. Since AUDI can simultaneously identify the model parameters and determine the model order, it is especially convenient for use in this example.

A MIMO process is usually broken into several MISO processes for convenience of identification and thus the following model can be used for this application.

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} A_1(z^{-1}) & 0 \\ 0 & A_2(z^{-1}) \end{bmatrix}^{-1} \begin{bmatrix} B_{11}(z^{-1})z^{-d_{11}} & B_{12}(z^{-1})z^{-d_{12}} \\ B_{21}(z^{-1})z^{-d_{21}} & B_{22}(z^{-1})z^{-d_{22}} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad (9.14)$$

where  $x_1(t)=P$ ,  $x_2(t)=X$ ,  $u_1(t)=D$  and  $u_2(t)=Q$ .  $A_1(z^{-1})$  is the common denominator of  $B_{11}(z^{-1})$  and  $B_{12}(z^{-1})$ ,  $A_2(z^{-1})$  is the common denominator of  $B_{21}(z^{-1})$  and  $B_{22}(z^{-1})$ . This MISO model is equivalent to the following true MIMO model

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \frac{\tilde{B}_{11}(z^{-1})}{\tilde{A}_{11}(z^{-1})}z^{-d_{11}} & \frac{\tilde{B}_{12}(z^{-1})}{\tilde{A}_{12}(z^{-1})}z^{-d_{12}} \\ \frac{\tilde{B}_{21}(z^{-1})}{\tilde{A}_{21}(z^{-1})}z^{-d_{21}} & \frac{\tilde{B}_{22}(z^{-1})}{\tilde{A}_{22}(z^{-1})}z^{-d_{22}} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} + \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix} \quad (9.15)$$

if the following is assumed

$$\begin{cases} A_1(z^{-1}) = \tilde{A}_{11}(z^{-1}) \tilde{A}_{12}(z^{-1}) \\ A_2(z^{-1}) = \tilde{A}_{21}(z^{-1}) \tilde{A}_{22}(z^{-1}) \end{cases}$$

and

$$\begin{cases} B_{11}(z^{-1}) = \tilde{B}_{11}(z^{-1}) \tilde{A}_{12}(z^{-1}), & B_{12}(z^{-1}) = \tilde{B}_{12}(z^{-1}) \tilde{A}_{11}(z^{-1}) \\ B_{21}(z^{-1}) = \tilde{B}_{21}(z^{-1}) \tilde{A}_{22}(z^{-1}), & B_{22}(z^{-1}) = \tilde{B}_{22}(z^{-1}) \tilde{A}_{21}(z^{-1}) \end{cases}$$

Clearly, the orders of the denominators and the numerators are all increased when the MIMO process is broken into MISO processes. The rationale behind this is:

- most real processes have essentially infinite order, thus any model with finite order only approximates the real process. For control purposes, the best model is the one that gives the best output prediction, or equivalently the smallest loss function.



- theoretically, a MIMO process can always be broken into MISO processes without losing its multivariable characteristics, by increasing the orders of the transfer functions.
- the ALDI algorithm simultaneously identifies  $n$  different models with model orders from 1 up to  $n$ , so the most appropriate model can be easily selected.

### 9.2.2 Identification of Parameters

Using two uncorrelated random binary sequences as the input excitation signals for flowrate  $D$  and reboiler duty  $Q$ , the input/output data were recorded and a portion of the data plotted in Figure 9.2.

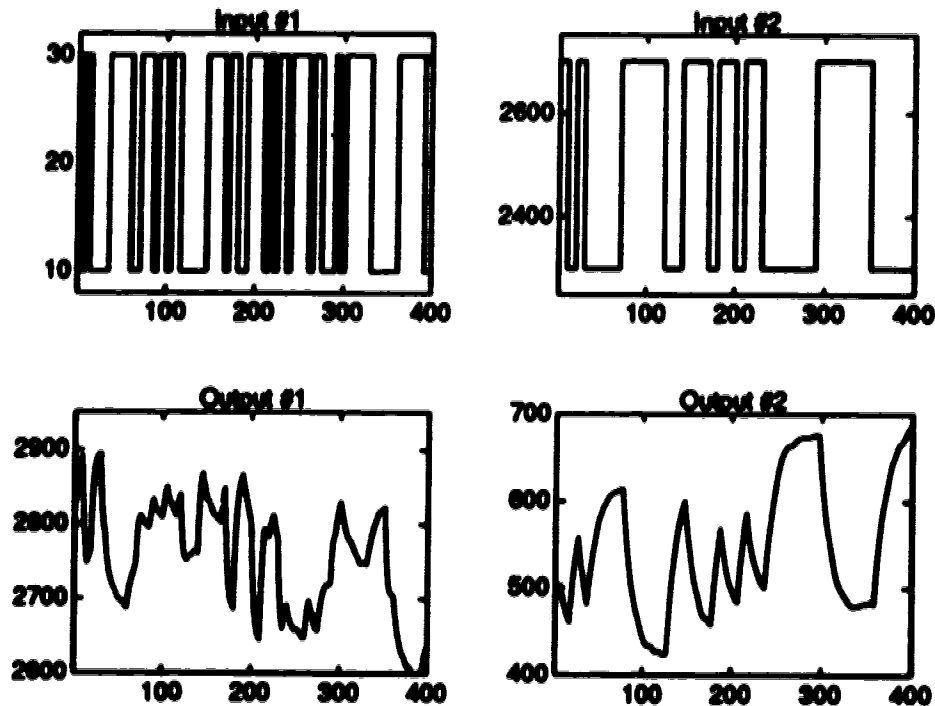
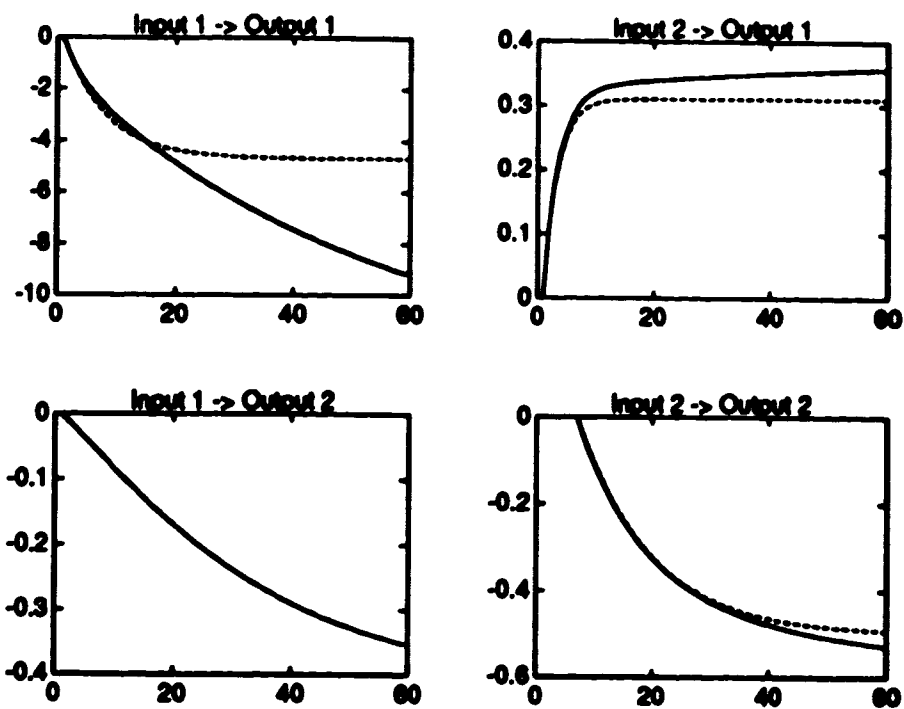


Figure 9.2: Input/output Data of the Distillation Column

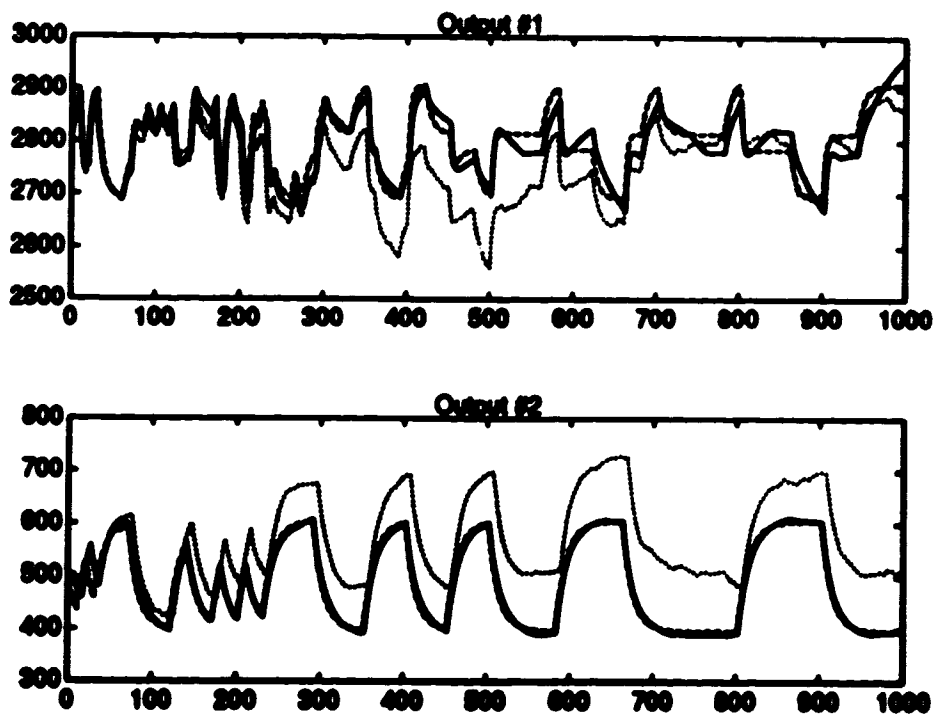
First, the ALDI algorithm is applied without any pretreatment of the data. The resulting parametric model is

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 1 - 1.659s^{-1} + 0.655s^{-2} & 0 \\ 0 & 1 - 1.867s^{-1} + 0.870s^{-2} \end{bmatrix}^{-1} \cdot \begin{bmatrix} -0.618s^{-1} + 0.537s^{-2} & 0.105s^{-1} - 0.103s^{-2} \\ -0.0086s^{-1} - 0.00723s^{-2} & -0.0396s^{-7} + 0.0376s^{-8} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

The corresponding step responses are plotted in Figure 9.3, where the solid lines are the identified step responses and the dashed lines are the true step responses. Due to the process noise, the identified step responses are not the same as the true step responses, especially for the first channel (input 1 to output 1). The process output



**Figure 9.3: Step Responses of the Distillation Column**



**Figure 9.4: Predicted Output, Noise-free Output and Measured Output**

predictions shown in Figure 9.4 show that the predicted process outputs (solid lines) using the identified model are very close to the noise-free process outputs (the dashed lines) simulated using the true process model (see next section). However, the measured (noise-contaminated) outputs (dotted lines) show large deviations from both the predicted output and the noise-free output due to noise.

Data pretreatment is a very important consideration in system identification and typically includes filtering of the input/output data. The process noise was investigated by running the process at steady state and recording the process output. Analysis of this output, which consists of mainly the noise dynamics, shows that there is a first order autoregressive term with the disturbance dynamics. This suggests the use of a first order filter to pretreat the data prior to identification. The process input/output data were therefore differenced and the AUDI identification algorithm is applied to the differenced data. The identified parameteric model is

$$\begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} = \begin{bmatrix} 1 - 0.699z^{-1} & 0 \\ 0 & 1 - 0.894z^{-1} \end{bmatrix}^{-1} \begin{bmatrix} -0.591z^{-1} & 0.105z^{-1} \\ -0.0066z^{-1} & -0.0395z^{-7} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

The identified step responses are shown in Figure 9.5 and comparison with Figure 9.3 shows that they are closer to the true step responses. However, the first channel (input 1 to output 1) still has a large mismatch in the steady state gain. The predicted process output (solid lines) using the identified model, the noise-free process output (dashed lines) simulated using the true model and the measured process output are plotted in Figure 9.6 and can be compared with Figure 9.4. This comparison shows that the output prediction of the column pressure (output 1) has improved slightly, while the prediction of the impurity level remains about the same. Using different parameters for the first order data filter results in approximately the same model as before.

The AUDI program also produces other information that is helpful in the analysis of the process. For example, the loss functions corresponding to these two data sets are listed in Table 9.2 and Table 9.3 respectively and show that the identified model using the original data has smaller loss functions (for the converged orders) than the model using the differenced data. The variance of the prediction error shown in Table 9.4

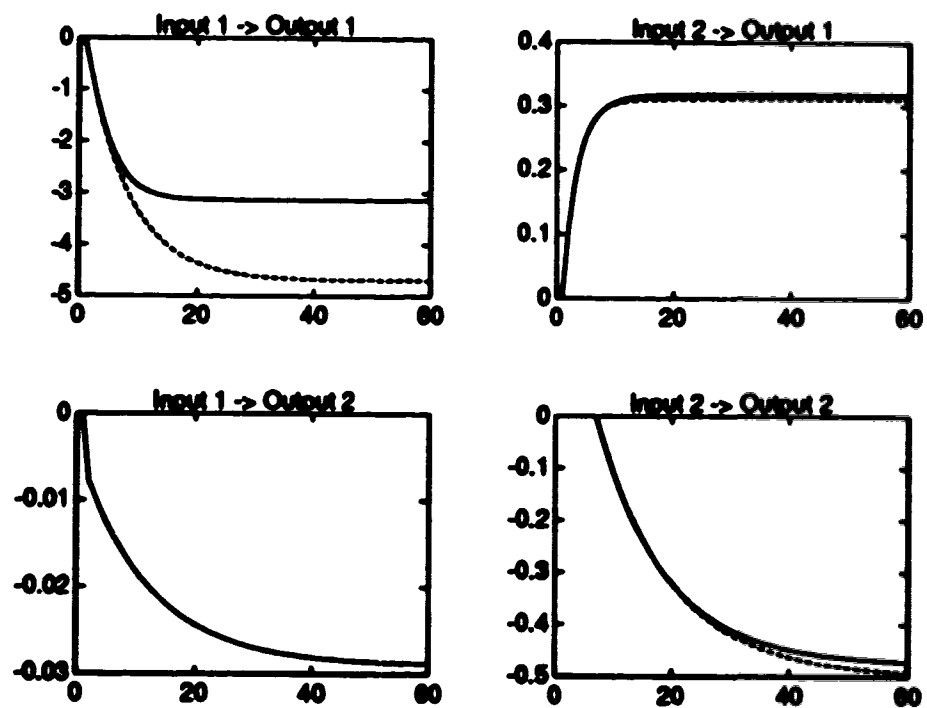
Table 9.2: Loss Functions Using Original Data

Order	0	1	2	3	4
Subsystem #1	5.517E6	4.827E4	1.828E3	1.750E3	1.714E3
Subsystem #2	6.631E6	4.860E3	5.323E2	5.229E2	5.103E2

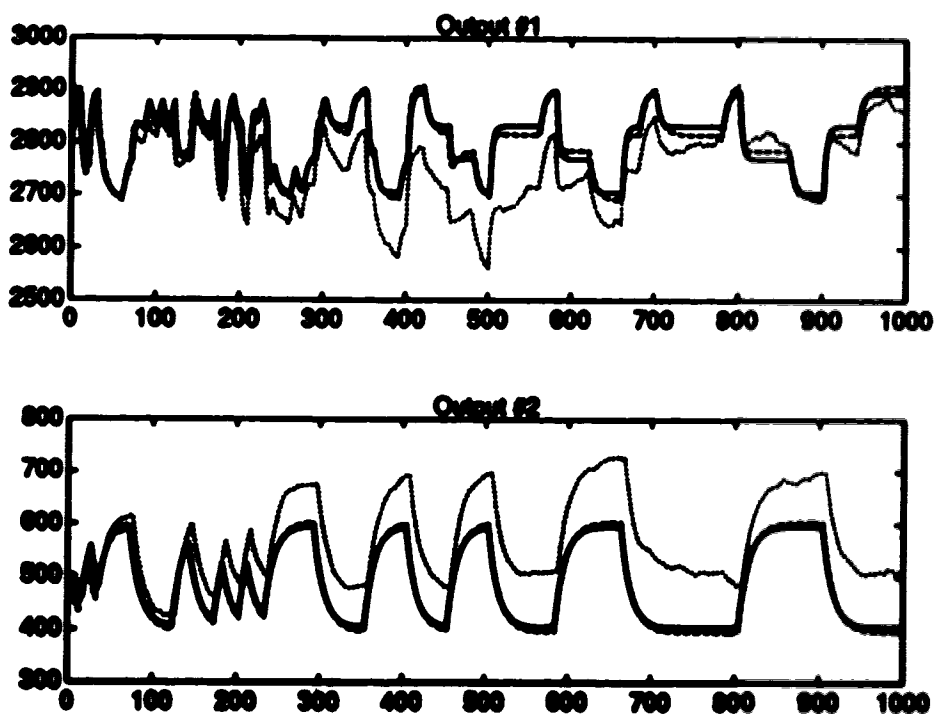
Table 9.3: Loss Functions Using Differenced Data

Order	0	1	2	3	4
Subsystem #1	0.464E4	2.187E3	1.971E3	1.943E3	1.750E3
Subsystem #2	2.666E4	5.662E3	5.483E2	5.355E2	5.211E2

shows that the variance estimated using the original data is closer to the true noise variance (which is the minimum possible value for the variance of prediction error) than the variance estimated using differenced data. This suggests that differencing of the



**Figure 9.5: Step Responses of the Distillation Column (Differenced Data)**



**Figure 9.6: Predicted Output, Noise-free Output and Measured Output (differenced data)**

Table 9.4: Variance of Prediction Errors

	original data	differenced data	true value
Subsystem #1	1.2938	1.2999	1.231
Subsystem #2	0.7059	0.7133	0.677

process I/O data is not necessary even though the process noise has integral dynamics. This is a topic that needs further investigation.

An identification package developed for commercial use in industry by a third party was then applied to the same set of data. The identification results are as follows: without any data pretreatment, the step responses shown in Figure 9.7 were obtained. The predicted process output (solid line), the noise-free output (dashed line) and the measured output (dotted line) are plotted in Figure 9.8. Compared with Figure 9.3 and Figure 9.5, the identified step responses using the commercial package are less accurate than the AUDI results. This can also be seen from the predictions.

If the input/output data are differenced and fed into the commercial identification package, the identified step response models are completely useless. Even the sign of the steady state gains are wrong in the first two channels as shown in Figure 9.9. This illustrates the importance of data pretreatment and the selection of an appropriate identification algorithm.

### 9.2.3 Discussion

Since this is a simulation rather than a real process, the process models are exactly known. Assuming that  $P$  is the column pressure,  $D$  is the overheads flowrate,  $Q$  is the reboiler duty and  $X$  is the impurity level in the bottom stream, the process models are given as follows.

(1) the column pressure,  $P(t)$

$$\begin{cases} P'(t) = 1.5296P'(t-1) + 0.5740P'(t-2) + \\ \quad - 0.6096D'(t-1) + 0.4022D'(t-1) + 0.1055Q'(t-1) - 0.0918Q'(t-2) \\ w(t) = 1.5945w(t-1) - 0.5945w(t-2) + ew \\ P(t) = P'(t) + P(0) + w(t) \end{cases}$$

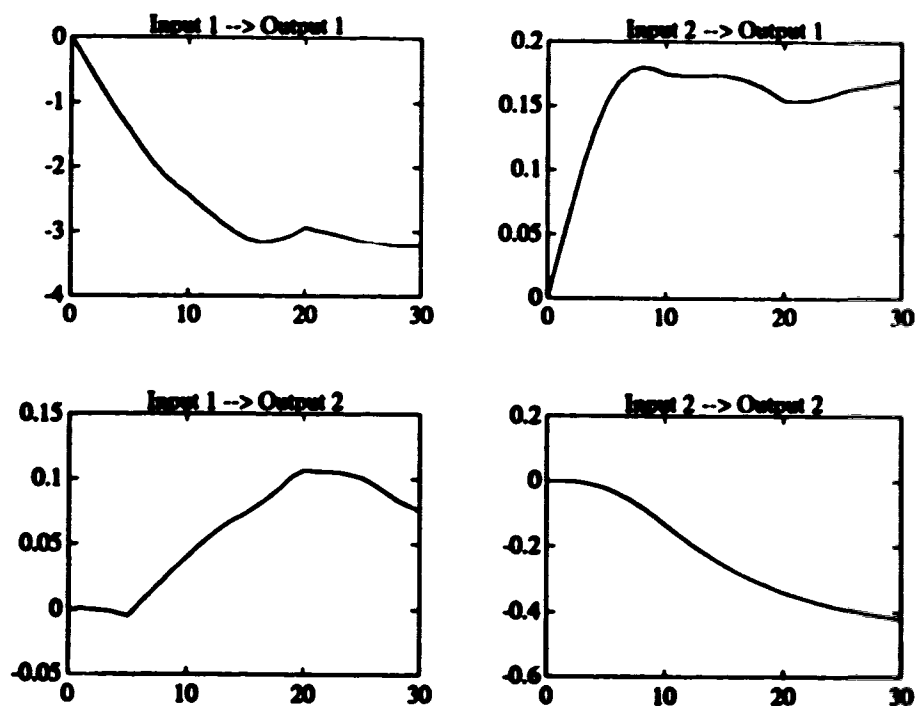
where  $ew$  is zero-mean white noise with variance  $\sigma_w^2 = 1.231^2$ .

(2) the impurity level,  $X(t)$

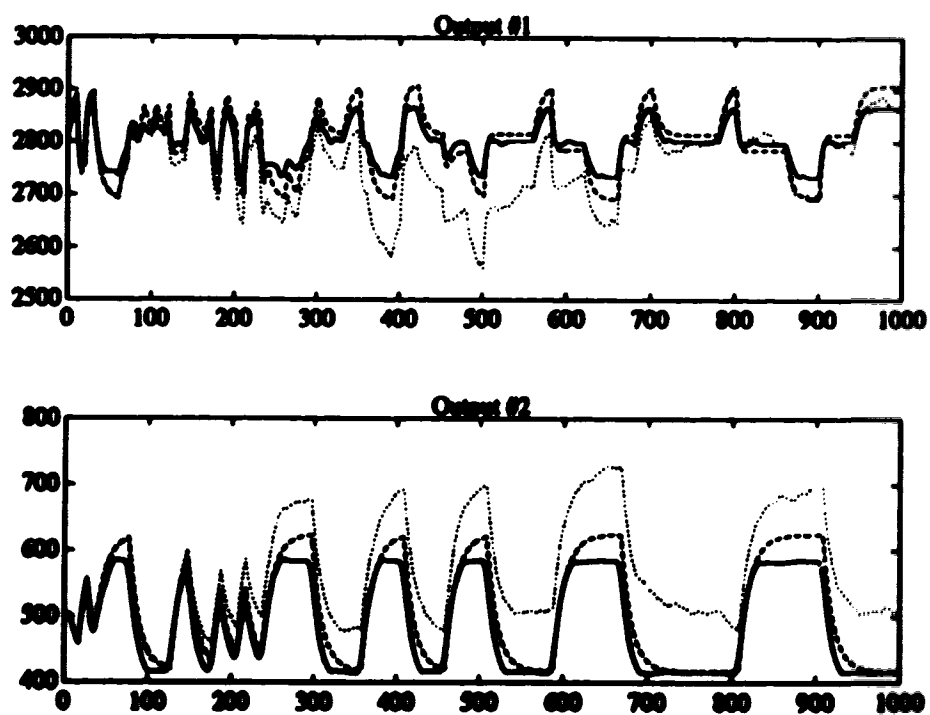
$$\begin{cases} s_{ss} = 5.0 \times 10^5 / (Q(t-7) - 1500) \\ s(t) = 0.9235s(t-1) + 0.0765s_{ss} + v(t) \\ v(t) = 1.6365v(t-1) - 0.6365v(t-2) + ev \end{cases}$$

where  $ev$  is zero-mean white noise with variance  $\sigma_v^2 = 0.677^2$ .

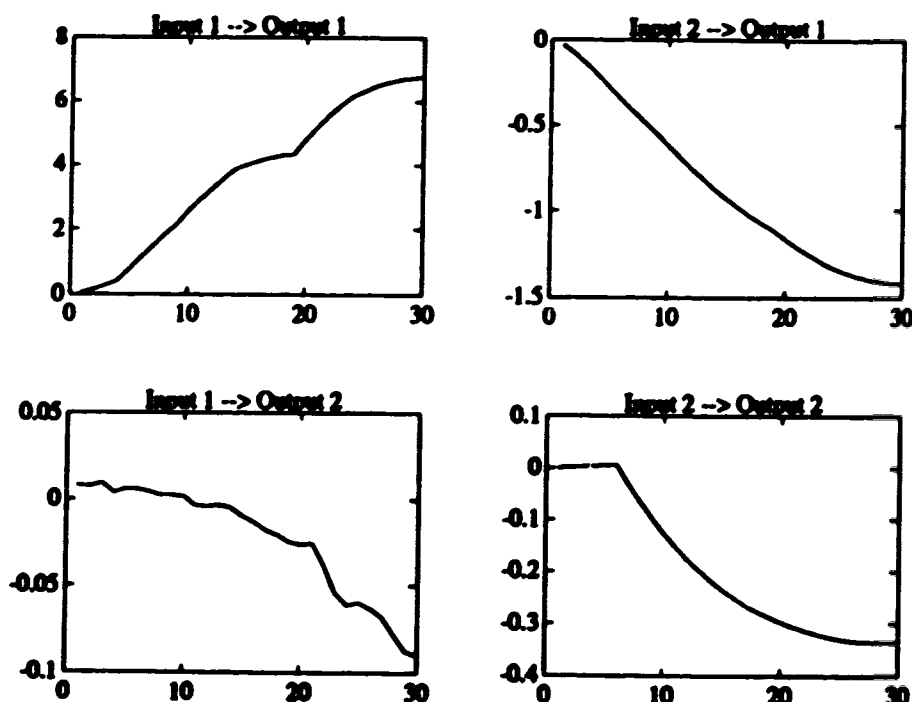
Clearly the second channel is nonlinear in nature so there is no linear model that corresponds to it. The identification purpose is to find a linear discrete-time model that can represent the external characteristics of the impurity, even though it might be only valid in the vicinity of this operating point. If the ordinary least-squares (OLS) method is used to identify this channel, the model order should be determined first. This is



**Figure 9.7: Step Responses from a Commercial Identification Package (original data)**



**Figure 9.8: Predicted Output, Noise-free Output and Measured Output (from the commercial package)**



**Figure 9.9: Step Responses from a Commercial Identification Package (differenced data)**

usually done by fitting the data to different models from a lowest possible order to a highest possible order. Then the loss functions of these models are compared and the appropriate order is selected by using the order-determination method discussed before. Obviously this is very tedious. However, using the ALDI algorithm, this procedure is implicitly included in the calculation and the computation requirement is equivalent to a single run of RLS (with the highest order).

If on-line identification is required, the linear model that best fits the performance of the actual process at its current operating point may be time-varying in both parameters and order. In this case, it would be very difficult to use RLS for the identification. The ALDI algorithm would have very obvious advantages over RLS.

From the identification results, it is seen that the linear models produced by ALDI are quite acceptable and give fairly good predictions. This implies that although most of real applications are nonlinear in nature, a low-order linear model can often be used to represent the process dynamics near the operating point.

### **9.3 Identification of a Real Industrial Process**

A real application was carried out in a local chemical plant (Wu 1991). Both the ALDI algorithm and the commercial identification package DMI (Dynamic Matrix Identification from DMC company in U.S.A.) (Cutler & Ramaker 1980, Cutler & Yocum 1980) were used on the same sets of data taken from actual production units. The DMI package is widely used in industry to produce the mathematical models required by DMC (Dynamic Matrix Control from DMC company) (Cutler & Ramaker 1980). DMI is an off-line

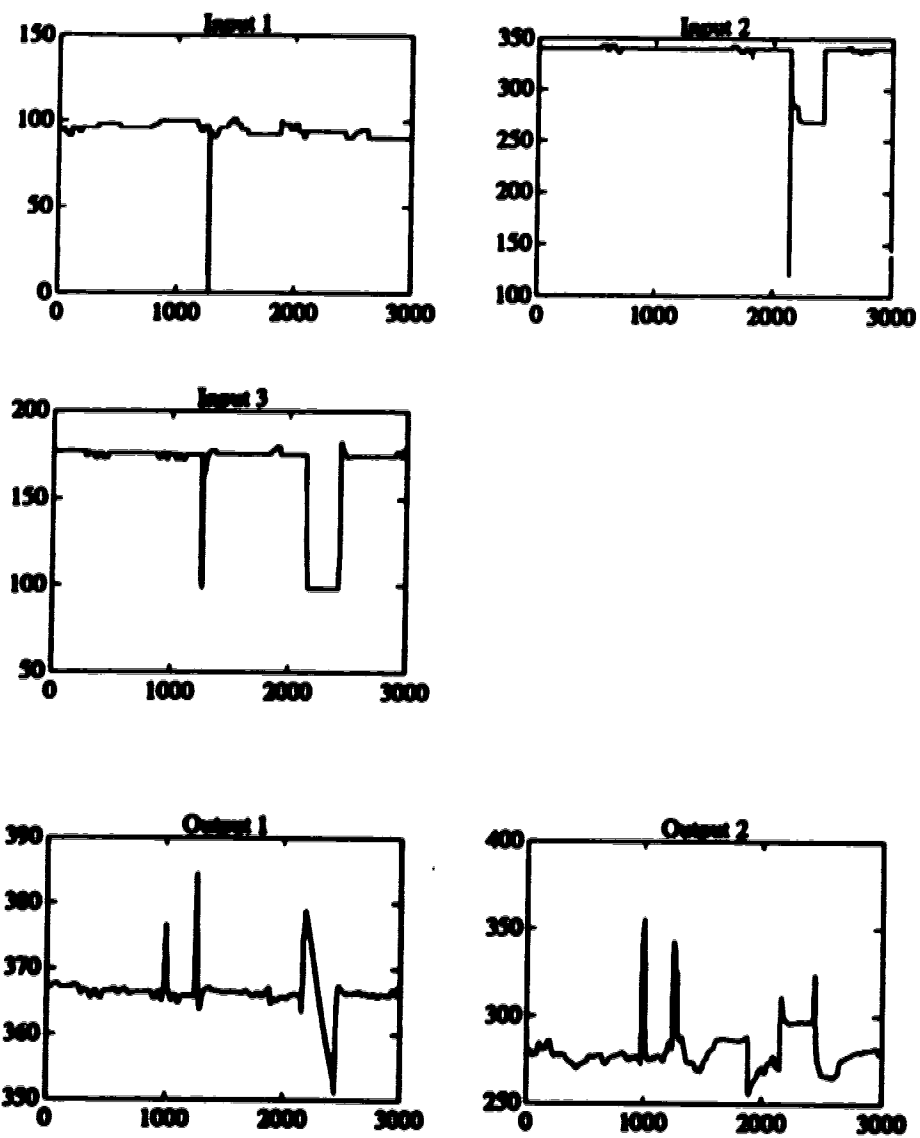
identification algorithm. To obtain a reasonably good process model, it often requires up to ten 24-hour days of plant tests plus careful manual checking of the plant data. This is very time and money consuming and requires a great deal of knowledge and experience with the process and process dynamics. An identification software package based on the ALDI algorithm was developed as a replacement or alternative to the DMI package. The ALDI package was designed to be simpler to use and more efficient with the objective of saving both manpower and money. The ALDI package took full advantage of the ALDI algorithm and could be easily extended to handle many practical requirements such as eliminating "bad data", data filtering, on/off control and detection of low excitation.

The ALDI package was tested with both simulated data and real plant data, and was compared with the DMI package. The ALDI algorithm produced results which appeared to be better than or equal to those produced by the DMI package. One of the comparisons was made based on a set of real plant data taken from a plant, which was assumed to be a 3-input 2-output MIMO process. The data, which consisted of 3850 points, were recorded during a 3-day plant test for DMI. The raw data is plotted in Figure 9.10, from which it is seen that the data have many outliers or "bad data points". With DMI, the data must be visually inspected by an experienced engineer to remove the outliers. When the "bad data" was removed, the process I/O data were in several slices. These slices were then fed into DMI for identification. The data after pretreatment by the control engineer in the plant are plotted in Figure 9.11. Obviously, the inspection of the input/output data for "bad data" requires considerable time and experience, and the result is subjective based on the engineer's personal judgement. Using the pretreated data, DMI produced the process step response data as shown by the solid lines in Figure 9.12.

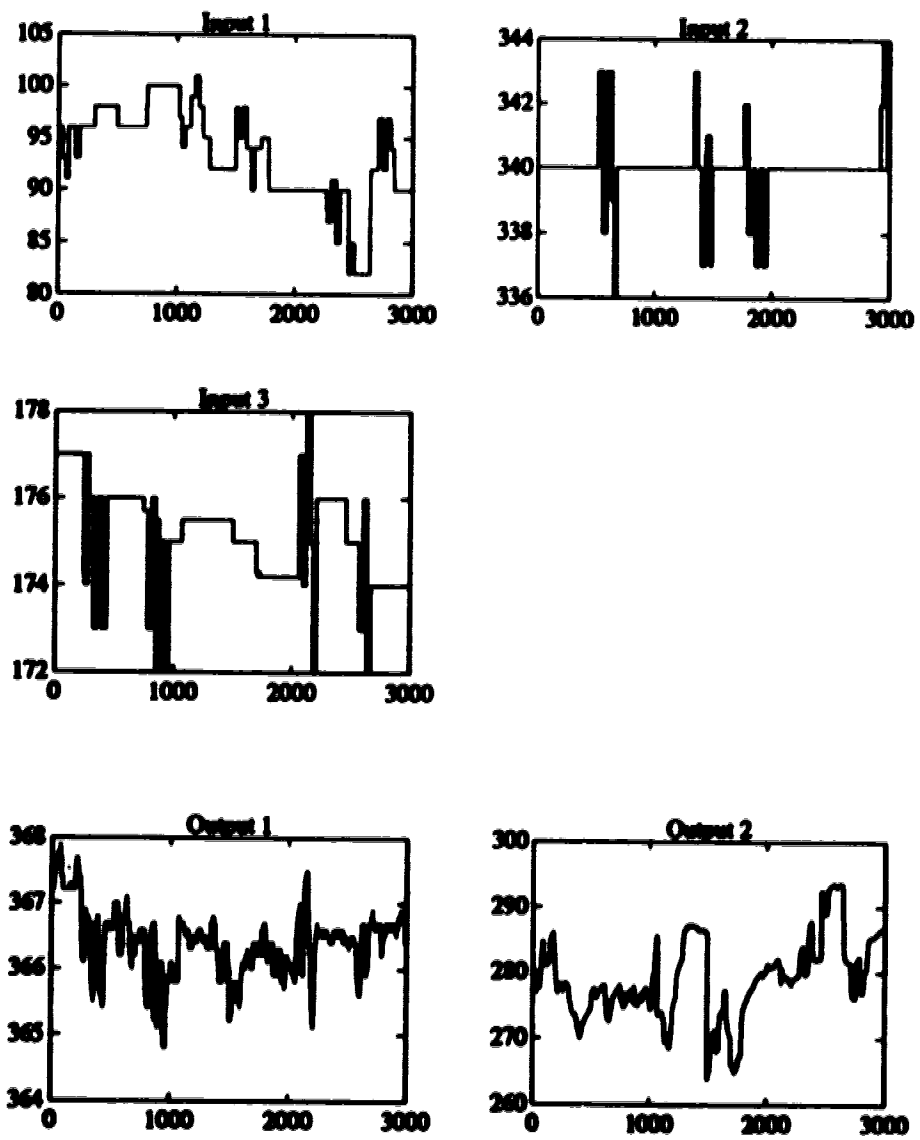
The ALDI package, which has the ability to automatically detect and eliminate the "bad data", used the raw data in Figure 9.10 for identification. The results are shown as the dashed lines in Figure 9.12. The ALDI and DMI results are quite close to each other. It should be noted that the "true" step response of the actual process is unknown so it is impossible to say whether DMI or ALDI is "best" in the sense of minimum output error. However, as some preliminary investigation showed, the "wiggles" (oscillations) in the step response trajectory produced by DMI, which are characteristics of the DMI type non-parametric identification methods, may cause problems for predictive control. In terms of real applications, the ALDI package had many advantages over the DMI package, such as

1. ALDI is on-line algorithm and thus can be configured to automatically and periodically update the process model. DMI is an off-line method and hence the time-consuming plant tests need to be repeated whenever an updated model is desired.
2. ALDI can automatically detect and eliminate "bad data" such as "outliers", and thus manual data pretreatment is not usually necessary or becomes very simple. For implementing DMI, considerable experience is required to check the plant data.
3. ALDI has an on/off control mechanism, and thus the identification is automatically stopped when there is no fresh information contained in the upcoming input/output data, e.g., during steady state operation or when the current model adequately represents the process behaviour.

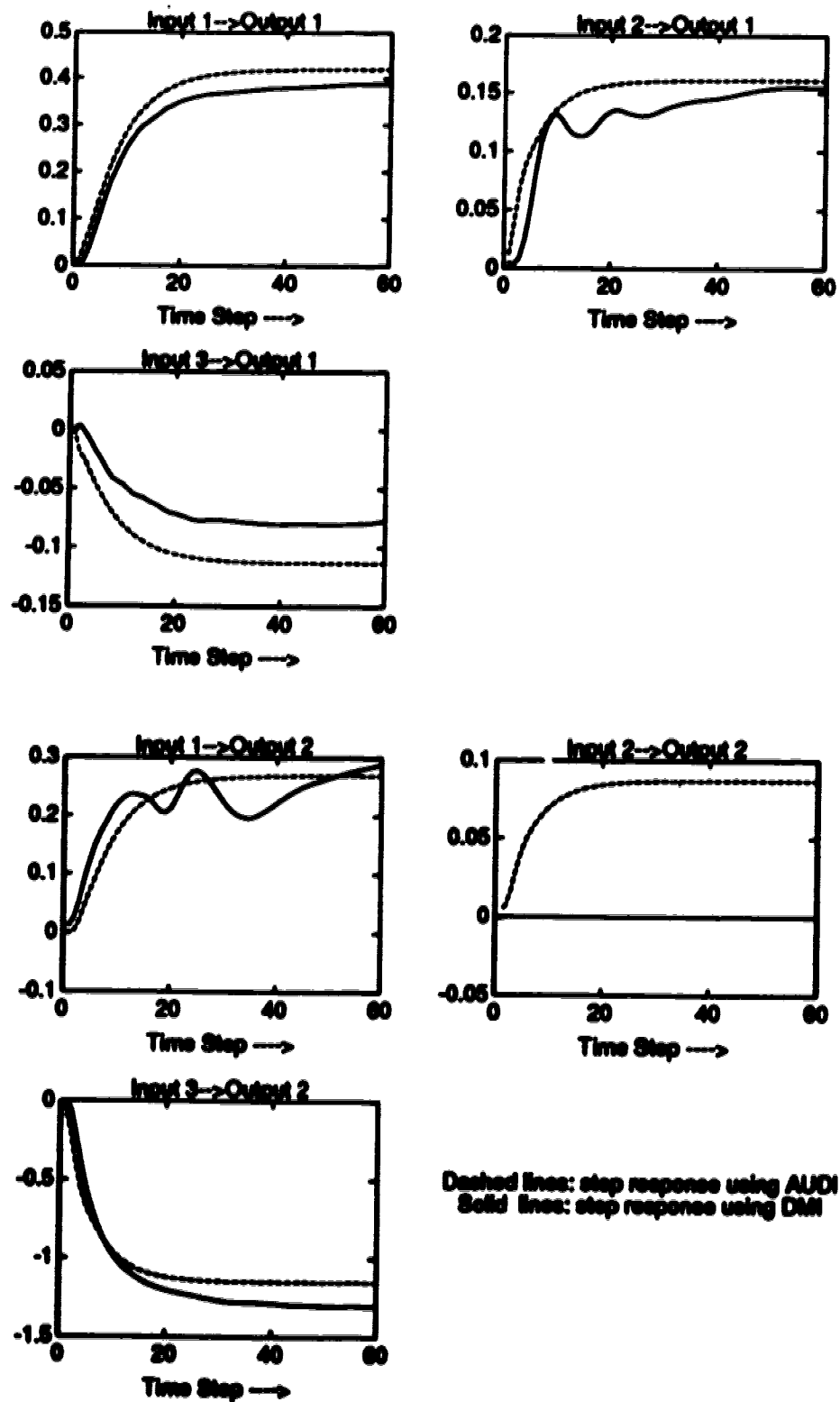




**Figure 9.10: Input/output data from a Real Plant (Raw Data)**



**Figure 9.11: Input/output data from a Real Plant (after pretreatment)**



**Figure 9.12: Step Response produced by AUDI (dashed lines) and DMI (solid lines)**

4. AUDI can provide more information, such as the process noise variance, signal-to-noise ratio, than DMI.

The AUDI software project is still being carried on for improved performance and more features. At the same time, more real plant data will be used to test the AUDI algorithm.

## **9.4 Conclusions**

Since many industrial control systems are currently being upgraded to more advanced model-based control, the need for simpler, more reliable and more accurate identification algorithms is urgent. It is also very important to augment identification theory with the practical considerations required for real applications. The development of the AUDI algorithm is an important step in this direction. The AUDI approach provides a rigorous and convenient basis for the analysis and design of least-squares type identification algorithms. It can also be extended to include many practical requirements such as information forgetting, on-off control, estimation of signal-to-noise ratio.

# Bibliography

- Banerjee, P., Shah, S. L., Niu, S. & Fisher, D. G. (1993), 'Identification of process dynamics for the shell benchmark problem'. Presented in 1992 Shell Workshop.
- Box, G. E. P. & Jenkins, G. M. (1970), *Time Series Analysis, Forecasting and Control*, Holden-Day, San Francisco.
- Cutler, C. R. & Ramaker, B. L. (1980), Dynamic matrix control — a computer control algorithm, in 'Proceedings of the 1980 Joint Automatic Control Conference', San Francisco, pp. WP5-B.
- Cutler, C. R. & Yocum, F. H. (1990), 'Experience with the DMC inverse for identification'. Technical Report.
- Fang, C. & Xiao, D. (1988), *Process Identification*, Tsinghua University Press, Beijing, China.
- Niu, S. (1991), Project report I, II and III on AUDI software program, Technical report, Department of Chemical Engineering, University of Alberta, Edmonton, Canada. Submitted to Esso Chemical Alberta Ltd.

## Chapter 10

# CONCLUSIONS AND RECOMMENDATIONS

### 10.1 Conclusions

The main contribution of this thesis is the development of the AUDI family of identification algorithms, which are characterised by their efficiency, simplicity, reliability and versatility. The AUDI approach represents a significant advance in process identification and brings the theory of process identification a step closer to real applications.

By rearranging and augmenting the data vector used in the ordinary least-squares method, an augmented information matrix (AIM) or augmented covariance matrix (ACM) can be constructed. The AIM/ACM contains all the information on process parameter estimates and loss functions for all models from order 1 to a user-specified maximum order. This information can be easily extracted by decomposition of the AIM/ACM, which results in a parameter matrix  $\hat{\theta}$  containing the parameter estimates for all models and a loss function matrix  $\mathcal{D}$  containing all the corresponding loss functions. This decomposed structure is called the augmented UD identification structure.

Recursive implementation of the AUDI structure can be done by taking advantage of Bierman's UD factorization algorithm. The main advantages of the AUDI algorithm can be summarised as follows

1. **VERSATILE.** The AUDI algorithm simultaneously produces all the information on the parameter estimates and corresponding loss functions for all models from 1 to a user-specified value  $n$ . Therefore the user can easily determine the appropriate model order by examining the loss functions and then pick the appropriate model from the multiple models provided in the parameter matrix. Problems caused by over- and/or under-parameterization can thereby be avoided. This is very useful for processes with unknown or time-varying orders. The simple structure of the  $\mathcal{D}$  matrix (diagonal) makes some features such as information forgetting more convenient and more theoretically sound.
2. **EFFICIENT.** The AUDI algorithm is an efficient implementation of the widely-used least-squares estimator. It identifies  $n$  models (from order 1 to  $n$ ) with approximately the same amount of computation as recursive least-squares algorithm for identifying only an  $n$ th order model.

3. **ROBUST.** The AUDI structure is a very stable structure. It preserves the positive-definiteness of the augmented information matrix or augmented covariance matrix when accumulating process information and thus can achieve high identification accuracy.
4. **SIMPLE.** The AUDI structure is a very compact structure with a clear interpretation. Process identification with the AUDI approach becomes very simple: construct the augmented information matrix or augmented covariance matrix, decompose it into factored form. Then all the information about the model parameters are contained in the parameter matrix  $U$  and all the information about the corresponding loss functions are contained in the loss function matrix  $D$ .
5. **INTEGRATED.** The AUDI approach provides a complete and efficient integration of a number of important features and extensions required by practical identification.

*Because of the features discussed above, the augmented UD identification methods are equal or superior to the widely used least-squares methods in all respects, and are therefore recommended for use in place of the least-squares type methods in all applications.*

The AUDI algorithm has also been extended to include several extensions of the basic AUDI-LS algorithm. These extensions include

1. The AUDI-ELS version. This is the AUDI form of the extended least-squares algorithm for identifying processes with colored process noise. The process model and the noise model are produced simultaneously. However, the noise model is assumed to have a specific structure.
2. The AUDI-IV version. The instrumental variable (IV) version of the AUDI algorithm can handle a more general form of colored process noise since it is not necessary to assume any structure for the noise model.
3. The MIMO-AUDI methods. The SISO algorithms are extended to the MIMO domain and thus simplify the identification of multivariable processes.
4. Although not presented in this thesis, extensions to some other forms (e.g., the Generalized Least-Squares (GLS) form) are also possible and straightforward.

All the above variants of the AUDI algorithm preserve the properties of efficiency, simplicity, reliability and versatility, and thus form a complete set of identification algorithms which can be used in place of the ELS-family algorithms for all applications.

The informative AUDI structure provides the basis for many very important implementations. On-line estimation of noise variance and signal-to-noise-ratio provide useful information required for many applications. Deeper insight into the principle of information accumulation is gained and thus makes the selection and/or design of information forgetting mechanism much easier and more theoretically sound.

## 10.2 Recommendations

Model-based control represents the future of industrial process control. More and more industrial control systems are changing from the "traditional" PID feedback loops to more advanced model-based controllers such as DMC and GPC. Obviously, an accurate and reliable dynamic model of the process is essential for model based control to be

successful. Therefore the first step (and as it turns out in most cases, the most important and difficult) step in developing a model based controller is to identify the process model.

The AUDI algorithms developed in this thesis are superior to most of the currently available identification algorithms and can provide many integrated features (e.g., simultaneous order and parameter estimation, online estimation of the signal-to-noise ratio) required for industrial process control. The AUDI algorithms have already provided the basis for significant improvements to industrial applications. Future extension with the AUDI algorithm can be focused on integrating them with the following capacities

1. Automatic data analysis. Process measurements plus information from the AUDI algorithm provide the necessary information required for statistical analysis of the noise, "automatic" filter design, detection of "bad" data, etc. These are all critical first steps for industrial process control but currently they are all open questions.
2. Simultaneous identification of model order, time delay and model parameters. Time delay identification remains a very difficult component of process identification. The AUDI algorithm can simultaneously identify model order and parameters. By appropriate formulation, the AUDI algorithm is also able to identify variable process time delays. Thus AUDI will minimize the *a priori* knowledge required for implementation which is usually not easy to obtain.
3. Closed-loop identification. This is an extremely difficult issue in real-time identification. However, the AUDI algorithm has the ability to automatically detect the existence of a feedback loop, and can provide simultaneous identification of the order and parameters of both the feedback channel and forward channel. Once again AUDI can provide the basis to overcome a major shortcoming of existing methods for closed-loop identification of industrial systems.
4. Integrated adaptive modeling and control design. The AUDI algorithm can provide much more information than the currently available identification algorithms, such as the signal-to-noise ratio, the degree of persistent excitation, the existence of a feedback loop. This information can be passed directly to the controller. Thus modeling and control can be more tightly integrated and control performance can be expected to improve significantly.
5. Software Design. The AUDI algorithms are being coded into a versatile, general-purpose package for both educational demonstrations and practical industrial applications. Cooperation with industry is helpful in evaluating the theoretical results in a real industrial environment to make them both theoretically and practically complete.



## Appendix A

# DERIVATION OF THE AUDI STRUCTURE

This appendix shows very briefly how the AUDI structure is derived from the least-squares estimator, and provides some insight into the least-squares principle.

First, consider following theorem

**Theorem A.1 (LDL<sup>T</sup>-Decomposition of partitioned matrix)** Given a positive-definite symmetrical matrix  $S$  of dimension  $d \times d$ , the LDL<sup>T</sup>-decomposition of its partitioned matrix is given by

$$S = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix} = \begin{bmatrix} I_1 & 0 \\ B^T A^{-1} & I_2 \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ B^T A^{-1} & I_2 \end{bmatrix}^T \quad (\text{A.1})$$

where  $A$  is the  $k \times k$  submatrix of  $S$ , with  $0 < k < d$ , and

$$\Delta = D - B^T A^{-1} B$$

**Proof:** Since  $S$  is positive-definite, any submatrix  $A$  of  $S$  is positive-definite, hence  $A^{-1}$  always exists. Direct multiplication of the right-hand side gives the left hand side of (A.1).

Represent the  $n$ th order augmented data vector  $\varphi(t)$  (2.12) by  $\varphi^{(n)}(t)$  (the bracketed superscript is used for the derivation in this appendix. In the context of this thesis, it is omitted for simplicity of notation). A "shift structure" (Ljung et al. 1978, Ljung & Söderström 1983) then follows as

$$\varphi^{(n)}(t) = \begin{bmatrix} -s(t-n), u(t-n), \dots, -s(t-3), u(t-3), -s(t-2), u(t-2), -s(t-1), u(t-1), -s(t) \end{bmatrix}^T$$

From which it can be shown that

$$\varphi^{(n)}(t) = \begin{pmatrix} h^{(n)}(t) \\ -s(t) \end{pmatrix}, \quad h^{(n)}(t) = \begin{pmatrix} \varphi^{(n-1)}(t) \\ u(t-1) \end{pmatrix}, \quad \varphi^{(n-1)}(t) = \begin{pmatrix} h^{(n-1)}(t) \\ -s(t-1) \end{pmatrix}, \quad \dots$$

and so on. Define the augmented information matrix (AIM) for different model orders as follows

$$\begin{aligned}
 S^{(n)}(t) &= \left[ \sum_{j=1}^i \varphi^{(n)}(j) \varphi^{(n)r}(j) \right]_{(2n+1) \times (2n+1)} \\
 R^{(n)}(t) &= \left[ \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) \right]_{2n \times 2n} \\
 S^{(n-1)}(t) &= \left[ \sum_{j=1}^i \varphi^{(n-1)}(j) \varphi^{(n-1)r}(j) \right]_{(2n-1) \times (2n-1)} \\
 R^{(n-1)}(t) &= \left[ \sum_{j=1}^i h^{(n-1)}(j) h^{(n-1)r}(j) \right]_{(2n-2) \times (2n-2)} \\
 &\vdots \\
 S^{(0)}(t) &= \left[ \sum_{j=1}^i \varphi^{(0)}(j) \varphi^{(0)r}(j) \right]_{1 \times 1}
 \end{aligned} \tag{A.2}$$

Obviously  $S^{(n)}(t)$  can be partitioned and Theorem A.1 is applied as

$$\begin{aligned}
 S^{(n)}(t) &= \sum_{j=1}^i \varphi^{(n)}(j) \varphi^{(n)r}(j) \\
 &= \begin{bmatrix} \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) & -\sum_{j=1}^i h^{(n)}(j) z(j) \\ -\sum_{j=1}^i z(j) h^{(n)r}(j) & \sum_{j=1}^i z^2(j) \end{bmatrix} \\
 &= \begin{bmatrix} -\left(\sum_{j=1}^i z(j) h^{(n)r}(j)\right) \left(\sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j)\right)^{-1} & 0 \\ 0 & I_1 \end{bmatrix} \begin{bmatrix} \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) & 0 \\ 0 & \Delta \end{bmatrix} \\
 &\quad \cdot \begin{bmatrix} -\left(\sum_{j=1}^i z(j) h^{(n)r}(j)\right) \left(\sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j)\right)^{-1} & 0 \\ 0 & I_1 \end{bmatrix}^T \\
 &= \begin{bmatrix} -\sum_{j=1}^i z(j) h^{(n)r}(j) & 0 \\ 0 & I_1 \end{bmatrix} \begin{bmatrix} R^{(n)}(t) & 0 \\ 0 & J^{(n)}(t) \end{bmatrix} \begin{bmatrix} -\sum_{j=1}^i z(j) h^{(n)r}(j) & 0 \\ 0 & I_1 \end{bmatrix}^T
 \end{aligned} \tag{A.3}$$

this structure is often called the "nested structure" (Söderström & Stoica 1980) since  $R^{(n)}(t)$  is "nested" in the  $S^{(n)}(t)$ . Equation (A.3) is given by

$$\begin{aligned}
 &\begin{bmatrix} \sum_{j=1}^i z(j) h^{(n)r}(j) \\ \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) \end{bmatrix} \begin{bmatrix} \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) \end{bmatrix}^{-1} \\
 &= \left[ \left( \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) \right)^{-1} \left( \sum_{j=1}^i h^{(n)}(j) z(j) \right) \right]^T
 \end{aligned}$$

$$= [\hat{\theta}^{(n)}(t)]^T$$

and

$$\begin{aligned} \Delta &= \sum_{j=1}^i z^2(j) - \left[ \sum_{j=1}^i z(j) h^{(n)r}(j) \right] \left[ \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) \right]^{-1} \left[ \sum_{j=1}^i h^{(n)}(j) z(j) \right] \\ &= \sum_{j=1}^i [\epsilon^{(n)}(j)]^2 \\ &= J^{(n)}(t) \end{aligned}$$

where  $\hat{\theta}^{(n)}(t)$  contains the parameter estimates of the  $n$ th order model at time  $t$ , and  $J^{(n)}(t)$  is the loss function for the  $n$ th order estimates at the same time step.

Now decompose the nested part  $R^{(n)}(t)$  in a similar manner as (A.3), we have

$$\begin{aligned} R^{(n)}(t) &= \sum_{j=1}^i h^{(n)}(j) h^{(n)r}(j) \\ &= \begin{bmatrix} \sum_{j=1}^i \varphi^{(n-1)}(j) \varphi^{(n-1)r}(j) & -\sum_{j=1}^i \varphi^{(n-1)}(j) u(j) \\ -\sum_{j=1}^i u(j) \varphi^{(n-1)}(j) & \sum_{j=1}^i u^2(j) \end{bmatrix} \\ &= \begin{bmatrix} I_{2n-1} & 0 \\ -\hat{a}^{(n-1)r}(t) & I_1 \end{bmatrix} \begin{bmatrix} S^{(n-1)}(t) & 0 \\ 0 & L^{(n-1)}(t) \end{bmatrix} \begin{bmatrix} I_{2n-1} & 0 \\ -\hat{a}^{(n-1)}(t) & I_1 \end{bmatrix}^T \end{aligned}$$

Recursively decomposing all the nested matrices in the same manner leads to

$$\begin{aligned} S^{(n)}(t) &= \begin{bmatrix} I_{2n} & 0 \\ -\hat{\theta}^{(n)r}(t) & I_1 \end{bmatrix} \begin{bmatrix} I_{2n-1} & 0 & 0 \\ -\hat{a}^{(n)}(t) & I_1 & 0 \\ 0 & & I_1 \end{bmatrix} \begin{bmatrix} I_{2n-2} & 0 & 0 \\ -\hat{\theta}^{(n-1)r}(t) & I_1 & 0 \\ 0 & & I_2 \end{bmatrix} \dots \\ &\quad \begin{bmatrix} I_2 & 0 & 0 \\ -\hat{\theta}^{(1)r}(t) & I_1 & 0 \\ 0 & & I_{2n-2} \end{bmatrix} \begin{bmatrix} I_1 & 0 & 0 \\ -\hat{a}^{(0)}(t) & I_1 & 0 \\ 0 & & I_{2n-1} \end{bmatrix} \\ &\quad \begin{bmatrix} J^{(0)}(t) & & & \\ & L^{(1)}(t) & & \\ & & \dots & L^{(n-1)}(t) \\ & & & J^{(n)}(t) \end{bmatrix} \\ &= \begin{bmatrix} I_{2n} & 0 \\ -\hat{\theta}^{(n)r}(t) & I_1 \end{bmatrix}^T \begin{bmatrix} I_{2n-1} & 0 & 0 \\ -\hat{a}^{(n)}(t) & I_1 & 0 \\ 0 & & I_1 \end{bmatrix}^T \begin{bmatrix} I_{2n-2} & 0 & 0 \\ -\hat{\theta}^{(n-1)r}(t) & I_1 & 0 \\ 0 & & I_2 \end{bmatrix}^T \dots \\ &\quad \begin{bmatrix} I_2 & 0 & 0 \\ -\hat{\theta}^{(1)r}(t) & I_1 & 0 \\ 0 & & I_{2n-2} \end{bmatrix}^T \begin{bmatrix} I_1 & 0 & 0 \\ -\hat{a}^{(0)}(t) & I_1 & 0 \\ 0 & & I_{2n-1} \end{bmatrix}^T \\ &= [L_n \ L_{n-1} \ \dots \ L_2 \ L_1] D [L_1^T \ L_2^T \ \dots \ L_{n-1}^T \ L_n^T] \\ &= LDL^T \end{aligned}$$

Inverting the AIM  $S^{(n)}(t)$ , we obtain the augmented covariance matrix (ACM) as

$$C^{(n)}(t) = [S^{(n)}(t)]^{-1}$$

$$\begin{aligned}
&= [L_n L_{n-1} \cdots L_2 L_1 D L_1^T L_2^T \cdots L_{n-1}^T L_n^T]^{-1} \\
&= L_n^{-T} L_{n-1}^{-T} \cdots L_2^{-T} L_1^{-T} D^{-1} L_1^{-T} L_2^{-T} \cdots L_{n-1}^{-T} L_n^{-T} \\
&= U_n U_{n-1} \cdots U_2 U_1 D^{-1} U_1^T U_2^T \cdots U_{n-1}^T U_n^T \\
&= U^{(n)}(t) D^{-(n)}(t) U^{(n)T}(t)
\end{aligned}$$

where  $U^{(n)}(t)$  is the parameter matrix  $U$  in (2.16) and  $D^{(n)}(t)$  is the loss function matrix  $D$  in (2.17).

## Appendix B

# DERIVATION OF THE AUDI-IV ALGORITHM

### B.1 Proof of the AUDI-IV structure

The proof of (5.16) is similar to the  $UDU^T$  decomposition in Appendix A. From the data vector (5.4) and instrumental variable vector (5.8)

$$\varphi(t) = \begin{pmatrix} h(t) \\ -z(t) \end{pmatrix}, \quad \eta(t) = \begin{pmatrix} \zeta(t) \\ -s(t) \end{pmatrix}$$

the augmented information matrix can be formulated and decomposed as

$$\begin{aligned} S(t) &= \sum_{j=1}^t \eta(j) \varphi^T(j) = \begin{bmatrix} \sum_{j=1}^t \zeta(j) h^T(j) & -\sum_{j=1}^t \zeta(j) z(j) \\ -\sum_{j=1}^t h^T(j) s(j) & \sum_{j=1}^t s(j) z(j) \end{bmatrix}_{(2n+1) \times (2n+1)} \\ &= \begin{bmatrix} I_{2n} & 0 \\ -\hat{\theta}^T(t) & 1 \end{bmatrix} \begin{bmatrix} R(t) & \\ & J^{(n)}(t) \end{bmatrix} \begin{bmatrix} I_{2n} & -\hat{\theta}(t) \\ 0 & 1 \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned} R(t) &\triangleq \sum_{j=1}^t \zeta(j) h^T(j) \\ \hat{\theta}(t) &= \left[ \sum_{j=1}^t \zeta(j) h^T(j) \right]^{-1} \sum_{j=1}^t \zeta(j) z(j) \\ \hat{\theta}(t) &= \left[ \sum_{j=1}^t h(j) \zeta^T(j) \right]^{-1} \sum_{j=1}^t h(j) s(j) \\ J^{(n)}(t) &= \sum_{j=1}^t s(j) z(j) - \sum_{j=1}^t h(j) s(j) \left[ \sum_{j=1}^t \zeta(j) h^T(j) \right]^{-1} \sum_{j=1}^t \zeta(j) z(j) \\ &= \sum_{j=1}^t s(j) I(j) \end{aligned}$$

Similarly,

$$R(t) = \begin{bmatrix} I_{2n-1} & 0 \\ \hat{\alpha}^T(t-1) & 1 \end{bmatrix} \begin{bmatrix} \hat{S}(t-1) & \\ & L^{(n-1)}(t-1) \end{bmatrix} \begin{bmatrix} I_{2n-1} & -\hat{\beta}(t-1) \\ 0 & 1 \end{bmatrix}$$

where  $\hat{S}$  equals the matrix  $S(t)$  after its last two columns and rows are removed.  $\hat{\alpha}(t-1)$  and  $\hat{\beta}(t-1)$  are the elements represented by '0' in (5.17) and (5.18).

Repeat the above decomposition with  $\hat{S}(t-1)$ , until  $S(t)$  is in the form of the product of a series of matrices. Taking the inverse of the factored  $S(t)$ , produces equation (5.16).

## B.2 The $UDV^T$ Decomposition

The derivation of the recursive  $UDV^T$  decomposition is adapted from Bierman's UD decomposition technique (Bierman 1977, Thornton & Bierman 1980), and can be stated as follows.

From equation (5.15)

$$\begin{aligned} C(t) &= \left[ \sum_{j=1}^t \eta(j) \varphi^T(j) \right]^{-1} \\ &= \left[ \sum_{j=1}^{t-1} \eta(j) \varphi^T(j) + \eta(t) \varphi^T(t) \right]^{-1} \\ &= [C(t-1) + \zeta(t) \varphi^T(t)]^{-1} \end{aligned}$$

Using the rank one update formula (Ljung & Söderström 1983)

$$C(t) = C(t-1) - C(t-1) \eta(t) \varphi^T(t) C(t-1) \beta^{-1}(t) \quad (B.1)$$

where

$$\beta(t) = 1 + \varphi^T(t) C(t-1) \eta(t) \quad (B.2)$$

and by defining

$$\begin{cases} f = U^T(t-1) \varphi(t), & f^* = V^T(t-1) \eta(t) \\ g = D(t-1) f, & g^* = D(t-1) f^* \end{cases}$$

equation (B.2) can be rewritten as

$$\beta(t) = 1 + f^T g^* = 1 + g^T f^*$$

Equation (B.1) can then be rewritten as

$$\begin{aligned} C(t) &= U(t) D(t) V^T(t) \\ &= U(t-1) \left[ D(t-1) - \frac{f^* f^T}{\beta(t)} \right] V^T(t-1) \end{aligned} \quad (B.3)$$

Following the derivation in Bierman (1977), the bracketed part of equation (B.3) is decomposed through a series of transformations to obtain

$$D(t-1) - \frac{f^* f^T}{\beta(t)} = \hat{D}(t) \hat{V}(t) \quad (B.4)$$

where  $\tilde{U}(t)$ ,  $\tilde{D}(t)$  and  $\tilde{V}(t)$  have a structure similar to  $U(t-1)$ ,  $D(t-1)$  and  $V(t-1)$ , respectively. From (B.3) and (B.4)

$$U(t) = U(t-1) \tilde{U}(t)$$

$$V(t) = V(t-1) \tilde{V}(t)$$

$$D(t) = \tilde{D}(t)$$

and the final algorithm then follows as Table 5.1.

## **Appendix C**

# **A CHECKLIST OF "PRACTICAL FACTORS"**

In real industrial applications, plant tests are usually very expensive, and it is often quite time-consuming to obtain a set of "good" plant data for identification. Careful planning and preparation are required at all stages of the project. A systematic approach to the design, specification, engineering, testing, commissioning and auditing of the process identification is essential to the success of the project. This appendix briefly (and partially) lists the steps that are needed to conduct a successful process identification project in industry. This is based on the literature and the authors' experience in actual applications of the ALIM algorithm to industrial processes.

The checklist of "practical factors" which follows is broken into the following sections for ease of reference. The order of the sections is essentially chronological but there is considerable variation between projects.

### **C.1 Project Definition and Planning**

As much as possible of the following information should be collected:

1. Goal of identification and the intended application of the identified model. This determines the type of model, its accuracy requirements and identification method to be used. Refer to Table 9.1 for guidelines.
2. Project specifications. This may include performance specifications, acceptance criteria, etc.
3. Manpower requirements. Depending on the size of the project, a process engineer plus control engineers who are familiar with the design, specification, engineering and execution of process identification applications should work together through every stage of the project.
4. Determine the software package that is going to be used for identification. Check the hardware/software requirements for running the package. Check the capacity (e.g., how many input/output variables and what file sizes/formats can be handled) to see whether it is suitable for the project.



5. Determine the hardware platform that is going to be used for process identification. Determine the computer systems for collecting and processing the input/output data. Check the computer power (speed, memory space, sampling rate, storage space) to ensure they are adequate for the project.
6. Determine the ease, acceptability and cost of doing formal or informal experimental tests on the process, *e.g.*, on some plants it is convenient and acceptable to do plant tests at short notice which means a more informal, iterative schedule can be used. For remote or critical plants, a carefully designed, minimal set of (expensive) plant tests may be necessary.

## **C.2 Process Analysis and Modeling**

1. Determine the input/output/disturbance (feedforward) variables of interest. Check whether the number of variables is within the limits of the capacity of the process control equipment and the data-acquisition/identification software.
2. Determine the auxiliary or intermediate variables that may be useful in the specification, implementation or assessment of the control/optimization applications.
3. Determine the possible sources of disturbances, both fast and slow with respect to process dynamics, *e.g.*, ambient temperature changes might cause "drift" in the operating conditions with a regular cycle of 24 hours; feedstock changes could produce "abrupt" step changes.
4. Obtain the following basic information about each input-output variable of the process
  - dynamic characteristics, *e.g.*, integrating dynamics, stability, non-minimum phase characteristics.
  - dominant time constants plus the range of significant dynamics (for determining the sampling interval).
  - time-to-steady-state or settling time (for determining the input excitation signal and the data length).
  - steady state gains (for determining the magnitude of the input excitation signals and defining the "steady state" optimum).
  - maximum possible time delays (to provide the range for searching the correct time delay) plus the range of typical variations.
  - approximate process order, *e.g.*, low-order/high-order dynamics. For MIM, the maximum possible process order is needed to provide the range of the multiple models.

It may be possible to determine the above information from past files or process data and/or informal tests on the process unit. In other cases it may be desirable to carry out a series of formal preliminary process tests and/or develop dynamic models of parts of the process.

5. Determine the approximate cut-off frequency of the process dynamics for input excitation signal and input/output data filter design.

6. Decide whether the process parameters and/or structure is time-invariant/time-variant.
7. Determine whether the process is non-linear. If non-linear, whether a linear approximation is sufficient for the application. Note that some control schemes can handle a variable order and/or delay if the values are known.
8. Determine the relative gain and cost of each input variable, *e.g.*, 3 units of change in input #1 is equivalent to 1 unit of change in input #2 in their effect on process outputs.
9. Determine relative importance of each output, *e.g.*, 2.5 units of error in output #1 is as serious as 1.5 units of error in output #2 in their influence on process performance/product quality. "Importance" may be a function of economics, safety, process operation, product specification, etc.
10. Obtain the constraints/bounds on all input/output/disturbance variables, including the operating constraints, process constraints (off-grade), equipment constraints and safety constraints.
11. Determine the noise characteristics (white/colored, mean, variance, signal-to-noise ratio). This is helpful for choosing the appropriate model set and designing data filters.

### **C.3 Experimental Design**

Make the following decisions in accordance with the project definition and based on the *a priori* information about the process,

1. Is on-line recursive identification required or is off-line batch identification (which provides more opportunity for manual intervention) sufficient?
2. Open-loop versus close-loop identification. *e.g.*, can the process be safely operated with the existing control loops open. Is there significant output feedback in the process itself which might cause difficulty?
3. The model set to be used (discrete/continuous, state-space/difference equation, linear/nonlinear, etc.). Use simulation plus any data already available to determine the best model set.
4. Using knowledge of the time constant, time-to-steady state, cut-off frequency etc, determine the process sampling time for each variable to be recorded. The selection of sampling time also depends on the sampling time of the final application (*e.g.*, digital control), and affects the accuracy of the resulting model. Check whether they can be obtained with existing computer hardware/software and plant instrumentation?
5. Input signal design.
  - select the form of input signal, *e.g.*, step changes, random binary sequences, pseudo-random binary sequence (PRBS), "normal" process changes, etc.

- using the knowledge of the steady state gains and noise level of the process, determine the magnitude of the input signals required to produce a reasonable signal-to-noise ratio at the process output.
  - using the knowledge of the sampling time, time constant of the process and the cut-off frequency, determine the period and duration of the input signal in order to stimulate all the frequencies of interest.
  - for MIMO applications, avoid correlated inputs and include combinations of inputs (e.g., a formal or informal factorial design).
6. The duration or data length of identification.
  7. The data format for storing the data or for communication between the process and the identification package.
  8. Is analog anti-alias filtering of measured data required? Is there excessive filtering in the existing instrumentation/data acquisition system?
  9. Using knowledge of the cut-off frequency of the process, determine whether a (low-pass) filter is needed to eliminate high frequency noise/disturbances which are above the process frequencies of interest.
  10. Whether a (high-pass) filter, e.g., differencing, is needed to eliminate trends, drift, offset or low frequency disturbances such as ambient temperature changes.
  11. For closed-loop identification, should a probing signal be added and what kind of probing signal be added. Where and when it should be added?

## **C.4 Experimental Plant Tests**

1. Hardware inspection, which includes the equipment for signal generation, data storage and computation, the sensors and actuators, data communication between the process control unit and the identification computer, etc.
2. Identification software configuration. Configure the data acquisition software in accordance with the process to be identified, the hardware environment under which the identification is being carried out, and the requirements of the identification software.
3. Check the media and data format that is used to store input/output data.
4. Check the span of the instruments for acceptable sensitivity. e.g., avoid trying to measure  $\pm 1$  when the measurement span is 1000.
5. Check the mass balance (MB) and energy balance (EB) during steady-state for input/output consistency.
6. Record the normal operating conditions and the normal operating range of the input/output/disturbance/auxiliary variables.
7. Check the relative and absolute accuracy of process I/O data.
8. Check for outliers and bursts in the input/output data.

9. Determine what should be printed out, displayed or monitored during the identification tests.
10. Identification of time delays for each individual channel.
11. Identification of structural indices (in SISO cases, the model order).
12. Re-examine this model structure using the rules such as flexibility, parsimony, complexity.
13. Selection of parameter estimation method, *e.g.*, AUDI.
14. determine what parameters are fixed? what parameters are bounded?
15. For recursive identification, choose appropriate initial conditions.
16. For recursive identification, choose forgetting mechanism or forgetting factor.
17. For closed-loop identification, add probing signals if needed to ensure the identifiability of the closed-loop process.
18. Add an on/off control mechanism for recursive identification and select parameters, *e.g.*, level of excitation for switching.
19. Use matrix regularization for better numerical performance.
20. Monitor plant operations and actions by the process operators that could make the data collected invalid or non-representative, *e.g.*, manual addition of ingredients; controller temporarily changed from automatic to manual; equipment changes (*e.g.*, temporary shutdown of a pump for a maintenance check); changes to parameters in the computer-control data acquisition system (*e.g.*, filter constants or limits); "unusual" or atypical conditions in related processes (*e.g.*, upstream or utility units).

## **C.5 Model Validation**

1. Consistency check. Use engineering insights, physical laws and *a priori* knowledge to validate model.
2. Auto-validation of model. A check of auto-correlations of the residuals at the desired confidence level should result in no more than 1 in 20 outside the limit.
3. Cross-validation of model. A check of cross correlations of the residuals versus the input time series at the desired confidence level should result in no more than 1 in 20 outside the limit.
4. If an independent estimate of measurement error is available then an F-test can be made with the variance of the residuals.
5. Run the identified model in parallel with process and compare model versus actual process outputs. Note that this is an open-loop test.
6. Model validation by comparison of control performance.

## **C.6 Integrating Model into Application**

If the AUDI algorithm is used for on-line identification, some additional information can be passed directly to the application. For example,

- parameter estimates and loss functions of multiple models.
- estimates of the process noise variance and signal-to-noise ratio.
- parameter estimates and loss functions of the backward models.
- level of input excitation.
- presence of output feedback loops.

Possible applications of process identification include

- model-based process control.
- prediction.
- adaptive filtering.
- real time optimization.
- process monitoring and fault detection.

The integration of the model into the specific application strongly depends on the specific requirements of the application. This is beyond the scope of this list.

## **C.7 Maintenance and Enhancement**

If off-line identification is used, the process model may need to be updated and improved periodically. The information that has been accumulated since the last identification should be incorporated into the next identification to give more accurate models and predictions. (Obviously, there must have been changes in the process or re-identification would not be necessary! These process changes may justify changes in the new identification procedures). This is quite application specific and therefore is not discussed in details.