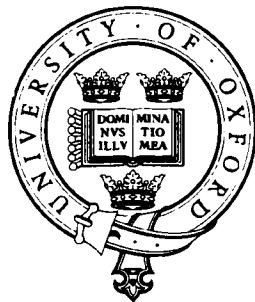


Multiple order models in predictive control

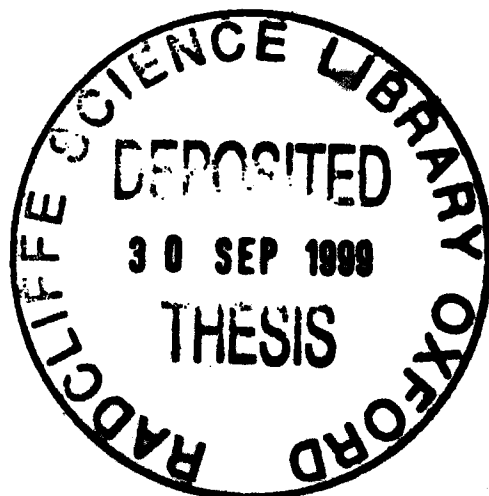
Robert O. Bowyer

Trinity College, Oxford.



Submitted in partial fulfilment of the requirements for the Degree of
Doctor of Philosophy at the University of Oxford.

Hilary 1999



*To my dearest Mum and Dad, and
Family.*

Multiple order models in predictive control

Robert O. Bowyer

Trinity College, Oxford.

Submitted in partial fulfilment of the requirements for the Degree of
Doctor of Philosophy at the University of Oxford.

Hilary 1999

Abstract

Predictive control has attracted much attention from both industry and academia alike due to its intuitive time domain formulation and since it easily affords adaption. The time domain formulation enables the user to build in prior knowledge of the operating constraints and thus the process can be controlled more efficiently, and the adaptive mechanism provides tighter control for systems whose behaviour changes with time. This thesis presents a fusion of technologies for dealing with the more practical aspects of obtaining suitable models for predictive control, especially in the adaptive sense.

An accurate model of the process to be controlled is vital to the success of a predictive control scheme, and most the of work to date has assumed that this model is of fixed order, a restriction which can lead to poor controller performance associated with under/over-parameterisation of the estimated model. To overcome this restriction a strategy which estimates both the parameters and the order of a linear model of the time-varying plant *on-line* is suggested. This Multiple Model Least-Squares technique is based on the recent work of Niu and co-workers who have ingeniously extended Bierman's method of UD updating so that, with only a small change to the existing UD update code, a wealth of additional information can be obtained directly from the U and D matrices including estimates of all the lower order models and their loss functions. The algorithm is derived using Clarke's Lagrange multiplier approach leading to a neater derivation and possibly a more direct understanding of Niu's Augmented UD Identification algorithm. An efficient and robust forgetting mechanism is then developed by analysing the properties of the continuous-time differential equations corresponding to existing parameter tracking methods. The resulting Multiple Model Recursive Least-Squares estimator is also ported to the δ -domain in order to obtain models for predictive controllers that employ fast sampling. The MMRLS estimator is then used in an adaptive multiple model based predictive controller for a coupled tanks system to compare performance with the fixed model order case.

Acknowledgements

I would like to thank my supervisor Prof. David Clarke for his guidance and encouragement over the last few years. Many a time has his wide experience and vision restored direction to my research. I am also greatly indebted to all my colleagues in the control group, especially the Self-Tuning Control crew: Alexander “Sasha” Kuznetsov, Chi-Ming “Billy” Chow and Arthur Jutan with whom I have shared many fruitful discussions in this wide and exciting field.

Individually, I would like to thank Billy for introducing me to the world of Linux, Sasha for interpreting whilst on conferences abroad, Mamdouh Atia and Martin Leahy for punting mayhem, Justin Lowe for his advice on job markets, Anisio Braga for teaching me how to “identify myself”, and Andrew Fry for helping me keep a perspective on things. Outside of work I owe a lot to the University Cross-Country Club and to my College friends, especially David Sykes, Eddie Rame and Euain Drysdale for all the fun times in The Coach House.

Above all I am grateful to my parents whose care, understanding and faith in me have kept me going through thick and thin.

Finally I would like to acknowledge the generous support of the Rhodes’ Trust.

Rob Bowyer
22.ix.98

*It was the best of times, it was the worst of times,
it was the age of wisdom, it was the age of foolishness,
it was the epoch of belief, it was the epoch of incredulity,
it was the season of Light, it was the season of Darkness,
it was the spring of hope, it was the winter of despair,
we had everything before us, we had nothing before us,
we were all going direct to heaven, we were all doing direct the other way
– in short, the period was so far like the present period,
that some of its noisiest authorities insisted on its being received,
for good or for evil,
in the superlative degree of comparison only.*

Charles Dickens, A Tale of Two Cities.

Contents

Glossary	ix
1 Introduction	1
1.1 Model-based predictive control	2
1.2 Adaptive predictive control	12
1.3 The need for multiple order models in predictive control	15
1.4 Existing methods for the estimation of multiple order models	16
1.5 Organisation of the thesis	26
2 Simultaneous estimation of multiple order models on-line	28
2.1 Multiple Model Least-Squares	28
2.2 Rapprochement with existing order recursive algorithms	34
2.3 Reliable estimator jacketing	39
2.4 Test simulation and results	52
2.5 Conclusions	56
3 The parameter tracking problem: a continuous-time approach	62
3.1 Continuous-time recursive exponentially weighted Least-Squares	62
3.2 Comparison with the Least Mean Squares algorithm	74

3.3	Simulation results	77
3.4	Conclusions	80
4	Multiple Model Least-Squares in the δ-domain	86
4.1	Introduction	86
4.2	Batch Multiple Model Least-Squares in the δ -domain	88
4.3	Recursive Multiple Model Least-Squares in the δ -domain	91
4.4	Estimation of multiple order δ -domain ARX models	95
4.5	The forgetting strategy	96
4.6	Simulations	98
4.7	Conclusions	108
5	Application of multiple model based predictive control	110
5.1	The coupled tanks experimental rig	110
5.2	The control law	113
5.3	Simulation-based comparison with an optimal non-linear predictive controller	125
5.4	Experimental results	128
5.5	Discussion	132
5.6	Conclusions	135
6	Conclusions and future work	147
6.1	Conclusions	147
6.2	Future work	149

A	Change of basis in the regressor space	151
B	Bierman’s recursive U-D estimator	153
B.1	Derivation of the recursive U-D filter	154
B.2	Regularisation of U-D filter	156
C	Proof of the $\mathcal{AUDI}$ method	157
D	Least-Squares in the δ-domain	161
E	Derivation of the coupled tanks model	164
	References	166

Glossary

Abbreviations

ACM	Augmented Covariance Matrix
AIC	Akaike’s Information Criterion A
AIM	Augmented Information Matrix
AR	Auto-Regressive
ARX	Auto-Regressive with an eXogenous input
ARMAX	Auto-Regressive Moving-Average with eXogenous input
AUDZ	Augmented UD Identification
BIC	Akaike’s Information Criterion B
BJ	Box–Jenkins
BLUE	Best Linear Unbiased Estimate
CARIMA	Controlled Auto-Regressive Integrated Moving Average
CGPC	Continuous-time Generalised Predictive Control
CRHPC	Constrained Receding-Horizon Predictive Control
CSGPC	Constrained Stable Generalised Predictive Control
DCGPC	Delta-domain approximation of CGPC
DGPC	Delta-operator Generalised Predictive Control

DQGPC	Delta-domain approximation of q -domain GPC
DMC	Dynamic Matrix Control
EE	Equation Error
EWLS	Exponentially Weighted Least-Squares
EVN	Error Variance Norm
FIR	Finite Impulse Response
HIECON	Hierarchical Constraint Control
IDCOM	Identification-Command
IIR	Infinite Impulse Response
GPC	Generalised Predictive Control
GPCW	Generalised Predictive Control with end-point state
LDA	Levinson-Durbin Algorithm
LMS	Least Mean Squares
LS	Least-Squares
LSL	Least-Squares Lattice
MA	Moving Average
MBPC	Model-Based Predictive Control
MCC	Multiple Correlation Coefficient
MDL	Minimum Description Length
MIMO	Multi-Input, Multi-Output
MISO	Multi-Input, Single-Output
MMLS	Multiple Model Least-Squares

MMGPC	Multiple Model Generalised Predictive Control
MMRLS	Multiple Model Recursive Least-Squares
MV	Minimum Variance
MWAC	Model Weighting Adaptive Control
MWLS	Mixed Weights Least-Squares
NARMAX	Non-linear ARMAX
OE	Output Error
PI	Proportional plus Integral
PRBS	Pseudo Random Binary Sequence
QP	Quadratic Program
RLS	Recursive Least-Squares
RMPCT	Robust Model Predictive Control Technology
RMS	Root Mean-Square
SGPC	Stable Generalised Predictive Control
SISO	Single-Input, Single-Output
SVD	Singular Value Decomposition
WGPC	Weighted Generalised Predictive Control

Nomenclature

$x(a b)$	prediction of the signal x at time a given information up to and including time b
q^{-1}	backward shift operator

$A(q^{-1}), A$	polynomial in q^{-1}
$\deg(A)$	degree of polynomial A
$\boldsymbol{a}$	a vector
$\boldsymbol{A}$	a matrix
$\boldsymbol{a}^T$	transpose of $\boldsymbol{a}$
$\boldsymbol{a}^{-T}$	transpose of $\boldsymbol{a}^{-1}$
Δ	differencing operator $1 - q^{-1}$

Principal symbols

i, j, k	integer indices
$u(t), u$	scalar control/manipulated variable
$v(t), v$	scalar auxiliary/constrained variable
$y(t), y$	scalar output variable
$w(t), w$	scalar setpoint variable
$f(t), f$	scalar free-response
$\xi(t), \xi$	scalar white noise sequence
$\boldsymbol{u}(t), \boldsymbol{u}$	control/manipulated variable vector
$\boldsymbol{v}(t), \boldsymbol{v}$	auxiliary/constrained variable vector
$\boldsymbol{y}(t), \boldsymbol{y}$	output variable vector
$\boldsymbol{w}(t), \boldsymbol{w}$	setpoint vector
$\boldsymbol{\xi}(t), \boldsymbol{\xi}$	white noise vector
$\boldsymbol{f}(t), \boldsymbol{f}$	free-response vector

A, B, T	polynomials in the scalar CARIMA model
$A y(t) = B u(t-1) + \frac{T}{\Delta} \xi(t)$	
$\mathcal{A}, \mathcal{B}, \mathcal{T}$	polynomials in the multivariable CARIMA model
$\mathcal{A} y(t) = \mathcal{B} u(t-1) + \frac{\mathcal{T}}{\Delta} \xi(t)$	
N_1	lower costing horizon
N_2	upper costing horizon
N_u	control horizon
$e(t), e$	linear regression model measurement noise
$\epsilon(t), \epsilon$	linear regression model prediction error
$\theta(t), \theta$	linear regression model parameter vector
$\hat{\theta}(t), \hat{\theta}$	estimated regression model parameter vector
$x(t), x$	regression vector
$\phi(t), \phi$	augmented regression vector
$k(t), k$	Kalman gain vector
$S(t), S$	information and augmented information matrix
$P(t), P$	covariance and augmented covariance matrix
$\mathcal{U}(t), \mathcal{U}$	MMLS parameter matrix
$\mathcal{D}(t), \mathcal{D}$	MMLS inverse loss function matrix
$\alpha(t), \alpha$	continuous-time exponential forgetting-factor
$\beta(t), \beta$	discrete-time exponential forgetting-factor
m	Maximum ARX model order
n	Number of estimated parameters

τ	Plant dead-time
$\underline{\tau}$	Minimum expected dead-time
$\overline{\tau}$	Maximum expected dead-time

Chapter 1

Introduction

Model-Based Predictive Control (MBPC) refers to a control methodology that optimises the future behaviour of a process plant using a model of that process. Indeed, even when very simple linear models are used, this powerful and reliable method has yielded significant economic benefits within the process control industry (Clarke, 1994; Camacho and Bordons, 1995; Qin and Badgwell, 1996). The reason for this success can be attributed to the fact that MBPC is perhaps the most general way of posing the control problem in the *time* domain and hence constraints can be easily incorporated into the control law (Scokaert, 1994; Rossiter and Kouvaritakis, 1993). Clearly the better the model can predict the future behaviour of the process the better the performance of the MBPC scheme and thus system identification (Eykhoff, 1974; Ljung, 1987; Söderström and Stoica, 1989) plays a vital rôle in the design of MBPC systems. This thesis concentrates on the more practical aspects of modelling for MBPC, and in particular, the application of multiple model techniques to adaptive predictive control.

There are many commercial packages available for MBPC design including Connoisseur (Sandoz, 1996) from Predictive Control Ltd., AspenTech's DMC+ which is a combination of Dynamic Matrix Control (DMC) (DMC, 1994) and Identification-Command (IDCOM) (Setpoint, 1993), Robust Model Predictive Control Technology (RMPCT) from Honeywell (Honeywell, 1995) and Adersa's Hierarchical Constraint Control (HIECON). These packages all employ a fixed order linear-in-the-parameters model so that Least-Squares (LS) methods (dating back to Gauss, 1809) can be used to automatically obtain the best-fit parameters from a batch of appropriate input-output data. Thus the order of the model has to be chosen

by the designer before the predictive controller is commissioned. This can be limiting for two reasons: firstly because the process behaviour might change with age and secondly because the batch of data used to choose the model order takes time to accumulate hence delaying the commissioning date and thus wasting money. Clearly adaption (Åström and Wittenmark, 1989) could be employed to address the first issue, and as Qin and Badgwell (1996) point out, it is remarkable that there is only one industrial adaptive MBPC package available (STAR from Dot Products (Dollar et al., 1993)) despite a strong market incentive for developing a self-tuning predictive controller.

For these reasons a selection and fusion of the most appropriate techniques is presented in this thesis to enable the practical exploitation of multiple order models in an adaptive multivariable constrained predictive controller. The adaptive mechanism is based on the elegant Augmented UD Identification (AUDI) algorithm of Niu et al. (1992) which simultaneously estimates a set of multiple order models with corresponding measures of model fit to automatically obtain an estimate of appropriate model order. This Multiple Model Least-Squares algorithm (Niu et al., 1995a; Bowyer and Clarke, 1998) is shown to be numerically robust and is especially suited to adaptive systems.

1.1 Model-based predictive control

This section gives a brief introduction to predictive control. Although there are many predictive controllers in the literature they generally involve the following: predicting the future responses from a suitable model, choosing a cost function to reflect the control objectives, and an optimisation routine to generate the future controls. The finite-horizon framework of Yoon (1994) (which includes the Generalised Predictive Control (GPC) algorithm of Clarke et al. (1987) and Constrained Receding-Horizon Predictive Control (CRHPC) of Clarke and Scattolini (1991)) has been chosen here to highlight the main ingredients, and since it easily lends itself to adaption. It should be noted however that more recent algorithms using infinite-horizons (Rawlings and Muske, 1993; Rossiter et al., 1996; Scokaert, 1997) have been developed to deal with nominal stability issues associated with finite-horizon predictive controllers.

1.1.1 Long range prediction

Consider the general plant description:

$$\begin{aligned} y(t) &= \mathcal{M}(q^{-1})u(t) + \mathcal{N}(q^{-1})\xi(t), \\ \psi(t) &= \mathcal{P}(q^{-1})y(t), \end{aligned}$$

where $u(t)$ and $y(t)$ are the plant input and output signals, $\psi(t)$ is the auxiliary or ‘shaped’ output to be predicted, $\xi(t)$ is an uncorrelated random white noise sequence with zero mean, $\mathcal{M}(q^{-1})$, $\mathcal{N}(q^{-1})$ and $\mathcal{P}(q^{-1})$ are Infinite Impulse Response (IIR) models and q^{-1} is the backward shift operator. The optimal i -step ahead predictor for the shaped output is found by splitting the disturbance model $\mathcal{P}(q^{-1})\mathcal{N}(q^{-1})$ into terms which only affect past (hence known) noise signals and those which affect the future, i.e.

$$\mathcal{P}(q^{-1})\mathcal{N}(q^{-1}) = \mathcal{N}_i^*(q^{-1}) + q^{-i}\mathcal{N}_i(q^{-1}), \quad (1.1)$$

where $\mathcal{N}_i^*(q^{-1})$ is an $(i-1)^{th}$ order polynomial formed from the first i terms of $\mathcal{P}(q^{-1})\mathcal{N}(q^{-1})$.

Thus the i -step predictor is given by:

$$\begin{aligned} \hat{\psi}(t+i) &= \mathcal{P}(q^{-1})\mathcal{M}(q^{-1})u(t+i) + \frac{\mathcal{N}_i(q^{-1})}{\mathcal{N}(q^{-1})}[y(t) - \mathcal{M}(q^{-1})u(t)] \\ &= \frac{\mathcal{N}_i^*(q^{-1})}{\mathcal{N}(q^{-1})}\mathcal{M}(q^{-1})u(t+i) + \frac{\mathcal{N}_i(q^{-1})}{\mathcal{N}(q^{-1})}y(t), \end{aligned} \quad (1.2)$$

where the prediction error is

$$\psi(t+i) - \hat{\psi}(t+i) = \mathcal{N}_i^*(q^{-1})\xi(t+i)$$

which is orthogonal to the prediction of eqn.(1.2) and thus leads to optimality. Notice that the use of the disturbance model $\mathcal{N}(q^{-1})$ effectively removes the influence of the disturbances on the prediction via the term $\frac{\mathcal{N}_i(q^{-1})}{\mathcal{N}(q^{-1})}[y(t) - \mathcal{M}(q^{-1})u(t)]$. Thus the selection of $\mathcal{N}(q^{-1})$ plays a *crucial* rôle in robust predictive control design.

The IIR plant $\mathcal{M}$ and disturbance $\mathcal{N}$ models in the general prediction equation of eqn.(1.2) can be approximated by truncating to get Finite Impulse Response (FIR) models as in IDCOM of Richalet et al. (1978) and DMC of Cutler and Ramaker (1980), or by using finite-order rational transfer functions as in GPC and CRHPC. The latter approach will be

adopted here as it requires fewer model parameters to be estimated and copes with open-loop unstable models. Hence we now consider the following Controlled Auto-Regressive Integrated Moving-Average (CARIMA) model:

$$\begin{aligned} A(q^{-1})y(t) &= B(q^{-1})u(t-1) + \frac{T_c(q^{-1})}{\Delta}\xi(t), \\ \psi(t) &= \frac{P_n(q^{-1})}{P_d(q^{-1})}y(t). \end{aligned} \quad (1.3)$$

where A , B , T_c , P_n and P_d are polynomials in q^{-1} and Δ is the difference operator $(1 - q^{-1})$. This model is suitable for describing low frequency, step-like disturbances found in practice, and gives integral action in control by means of the internal model principle. The denominator of the noise model (Δ) can be replaced by other polynomials if exact prior knowledge is available about the disturbance characteristics (e.g. ramps, sinusoids).

Thus substituting $\mathcal{M} = q^{-1}\frac{B}{A}$, $\mathcal{N} = \frac{T_c}{A\Delta}$ and $\mathcal{P} = \frac{P_n}{P_d}$ into eqn.(1.1) we get the following Diophantine equation:

$$P_n T_c = P_d A \Delta E_i + q^{-i} F_i \quad (1.4)$$

where $E_i \equiv \mathcal{N}_i^*$ and $\frac{F_i}{P_d A \Delta} \equiv \mathcal{N}_i$ and thus

$$\deg(E_i) = i - 1, \quad \deg(F_i) = \max(\deg(P_d A), \deg(P_n T_c) - i).$$

Thus the predictor can be rewritten as:

$$\hat{\psi}(t+i) = \frac{B E_i}{T_c} \Delta u(t+i-1) + \frac{F_i}{P_d T_c} y(t).$$

This can be divided into two terms: one depending on the future control moves (forced response) and the other depending only on the past input-output data (free response), and hence the i -step ahead shaped output prediction is given by:

$$\hat{\psi}(t+i) = G_i \Delta u(t+i-1) + f(t+i), \quad (1.5)$$

where

$$f(t+i) = \frac{H_i}{T_c} \Delta u(t-1) + \frac{F_i}{P_d T_c} y(t) \quad (1.6)$$

is the free response, and the polynomials G_i and H_i satisfy the Diophantine equation:

$$E_i B = G_i T_c + q^{-i} H_i, \quad (1.7)$$

with

$$\deg(G_i) = i - 1, \quad \deg(H_i) = \max(\deg(B), \deg(T_c)) - 1.$$

Note that since we have assumed that the future noise signal, $\xi(t+i)$ for $i > 0$ is zero, the parameters $G_i(q^{-1})$ are simply the step response coefficients of the auxiliary system $\frac{P_n B}{P_d A}$, i.e.

$$\frac{P_n B}{P_d A \Delta} = \sum_{k=1}^{\infty} g_k q^{-k+1}, \quad G_i = \sum_{k=1}^i g_k q^{-k+1},$$

where g_k is the k^{th} step response of the shaped output $\psi(t)$.

1.1.2 The predictive control law

The MBPC control objective is to drive the predicted future shaped outputs close to their desired values as shown in Figure 1.1. This can be achieved by minimising the following quadratic cost function:

$$J = \sum_{i=N_1}^{N_y-1} \mu(i) \left\{ r(t+i) - \hat{\psi}(t+i) \right\}^2 + \sum_{i=N_y}^{N_2} \frac{\mu(N_y)}{\gamma} \left\{ w(t+N_y) - \hat{\psi}(t+i) \right\}^2 + \sum_{i=0}^{N_u-1} \rho(i) \Delta u^2(t+i), \quad (1.8)$$

subject to the constraint:

$$\Delta u(t+i) = 0 \quad \text{for } i \geq N_u,$$

where $\mu(i)$ and $\rho(i)$ are non-negative weighting sequences, N_1 and N_2 are the lower and upper prediction horizons, N_u is the control horizon and γ is a non-negative number (≤ 1) introduced to place heavier weighting on the errors further ahead than N_y .

Remark 1 The cost function in eqn.(1.8) encompasses GPC ($\gamma = 1$) and CRHPC ($\gamma = 0$) as special cases.

Given the future reference trajectory $\mathbf{w}$ (see Figure 1.1) and the plant free response $\mathbf{f}$ of eqn.(1.6), the cost function can be written in matrix form:

$$J = [\hat{\psi}_1 - \mathbf{w}_1]^T \mathbf{M} [\hat{\psi}_1 - \mathbf{w}_1] + \frac{\mu(N_y)}{\gamma} [\hat{\psi}_2 - \mathbf{w}_2]^T [\hat{\psi}_2 - \mathbf{w}_2] + \Delta \mathbf{u}^T \Lambda \Delta \mathbf{u}$$

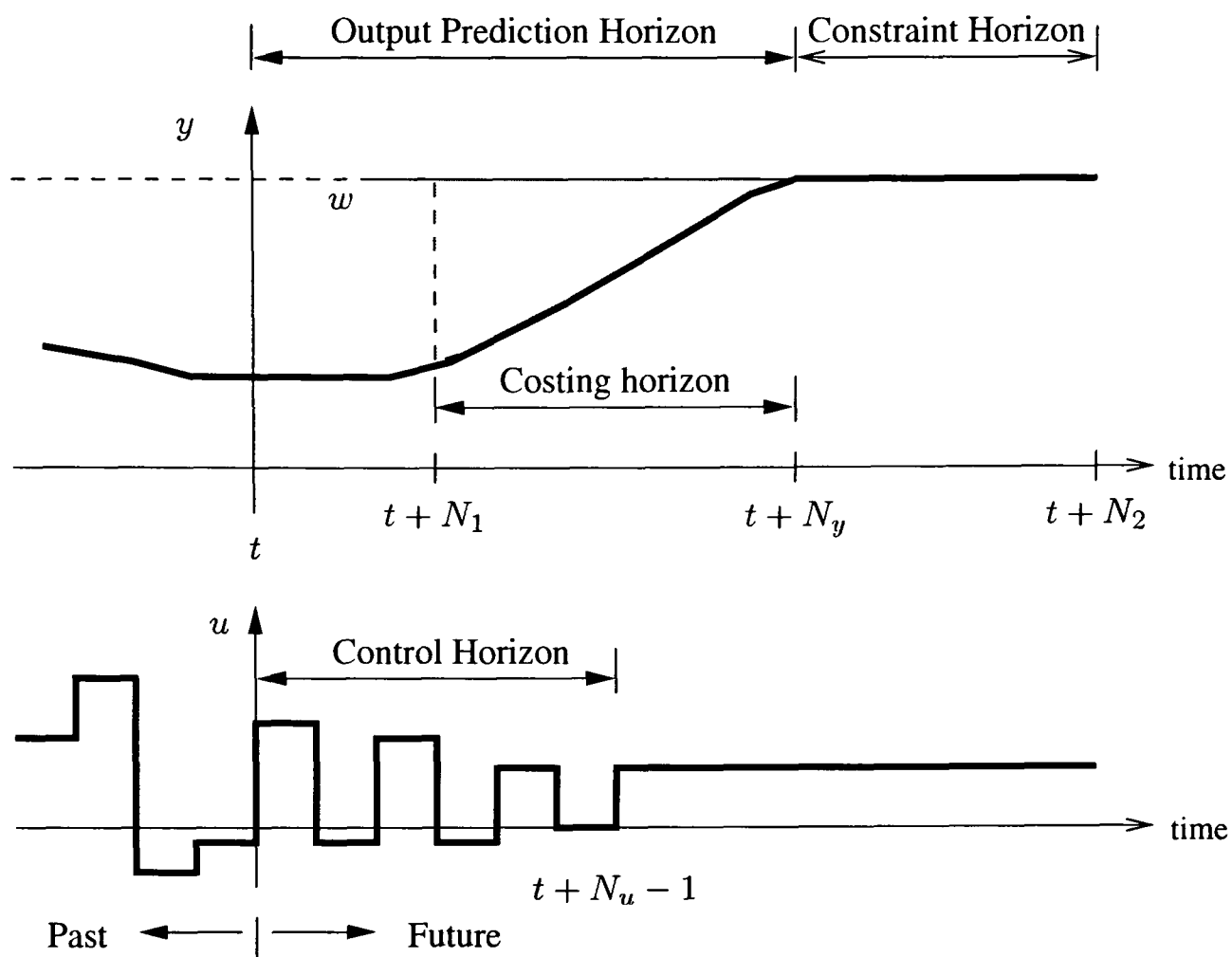


Figure 1.1: Prediction horizons for MBPC

where the vectors are constructed as follows

$$\begin{aligned}
 \Delta \mathbf{u} &= [\Delta u(t), \Delta u(t+1), \dots, \Delta u(t+N_u-1)]^T \\
 \mathbf{w}_1 &= [w(t+N_1), w(t+N_1+1), \dots, w(t+N_y-1)]^T \\
 \hat{\psi}_1 &= [\hat{\psi}(t+N_1), \hat{\psi}(t+N_1+1), \dots, \hat{\psi}(t+N_y-1)]^T \\
 \mathbf{f}_1 &= [f(t+N_1), f(t+N_1+1), \dots, f(t+N_y-1)]^T \\
 \mathbf{w}_2 &= [w(t+N_y), w(t+N_y+1), \dots, w(t+N_y)]^T \\
 \hat{\psi}_2 &= [\hat{\psi}(t+N_y), \hat{\psi}(t+N_y+1), \dots, \hat{\psi}(t+N_2)]^T \\
 \mathbf{f}_2 &= [f(t+N_y), f(t+N_y+1), \dots, f(t+N_2)]^T
 \end{aligned}$$

with the diagonal weighting matrices $\mathbf{M}$ and Λ being:

$$\begin{aligned}
 \mathbf{M} &= \text{diag}[\mu(N_1), \mu(N_1+1), \dots, \mu(N_y-1)] \\
 \Lambda &= \text{diag}[\rho(0), \rho(1), \dots, \rho(N_u-1)]
 \end{aligned}$$

The cost function is minimised given the predictions:

$$\begin{aligned}\hat{\psi}_1 &= \mathbf{G}_1 \Delta \mathbf{u} + \mathbf{f}_1 \\ \hat{\psi}_2 &= \mathbf{G}_2 \Delta \mathbf{u} + \mathbf{f}_2\end{aligned}$$

where the matrices $\mathbf{G}_1$ and $\mathbf{G}_2$ are given by:

$$\mathbf{G}_1 = \begin{bmatrix} g_{N_1} & g_{N_1-1} & \cdots & 0 \\ g_{N_1+1} & g_{N_1} & & 0 \\ \vdots & & \ddots & \vdots \\ g_{N_y-1} & g_{N_y-2} & \cdots & g_{N_y-N_u} \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} g_{N_y} & g_{N_y-1} & \cdots & g_{N_y-N_u+1} \\ g_{N_y+1} & g_{N_y} & & g_{N_y-N_u+2} \\ \vdots & & \ddots & \vdots \\ g_{N_2} & g_{N_2-1} & \cdots & g_{N_2-N_u+1} \end{bmatrix}.$$

The control is found by solving the augmented linear equation

$$\begin{bmatrix} \mathbf{G}_1^T \mathbf{M} \mathbf{G}_1 + \Lambda & \mathbf{G}_2^T \\ \mathbf{G}_2 & -\frac{\gamma}{\mu(N_y)} \mathbf{I} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_1^T \mathbf{M} (\mathbf{w}_1 - \mathbf{f}_1) \\ \mathbf{w}_2 - \mathbf{f}_2 \end{bmatrix} \quad (1.9)$$

where the vector $\mathbf{p}$ is given by

$$\mathbf{p} = \frac{\mu(N_y)}{\gamma} [\mathbf{G}_2 \Delta \mathbf{u} - (\mathbf{w}_2 - \mathbf{f}_2)].$$

Note that when $\gamma = 0$, eqn.(1.9) is identical to the reformulation of CRHPC in Clarke (1994) where the $\mathbf{p}$ is the vector of Lagrange multipliers for the equality constraints $\mathbf{G}_2 \Delta \mathbf{u} + \mathbf{f}_2 = \mathbf{w}_2$.

Applying the block matrix inversion lemma¹ to eqn.(1.9) we get the future control moves:

$$\begin{aligned} \Delta \mathbf{u} &= \left[\mathbf{I} - \tilde{\mathbf{G}} \mathbf{G}_2^T \left(\frac{\gamma}{\mu(N_y)} \mathbf{I} + \mathbf{G}_2 \tilde{\mathbf{G}} \mathbf{G}_2^T \right)^{-1} \mathbf{G}_2 \right] \tilde{\mathbf{G}} \mathbf{G}_1^T \mathbf{M} (\mathbf{w}_1 - \mathbf{f}_1) \\ &\quad + \tilde{\mathbf{G}} \mathbf{G}_2^T \left(\frac{\gamma}{\mu(N_y)} \mathbf{I} + \mathbf{G}_2 \tilde{\mathbf{G}} \mathbf{G}_2^T \right)^{-1} (\mathbf{w}_1 - \mathbf{f}_1), \end{aligned} \quad (1.10)$$

where the matrix $\tilde{\mathbf{G}}$ is:

$$\tilde{\mathbf{G}} = (\mathbf{G}_1^T \mathbf{M} \mathbf{G}_1 + \Lambda)^{-1}.$$

¹Block matrix inversion lemma:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1} B (D - C A^{-1} B)^{-1} C A^{-1} & -A^{-1} B (D - C A^{-1} B)^{-1} \\ -(D - C A^{-1} B)^{-1} C A^{-1} & (D - C A^{-1} B)^{-1} \end{bmatrix}$$

Adopting the *receding horizon* control strategy, only the first element of the control vector is used at time t , i.e.

$$\Delta u(t) = [1 \ 0 \ 0 \ \dots \ 0] \Delta \mathbf{u},$$

and the whole procedure is repeated at the next sample instant.

1.1.3 Model choice

The performance of a MBPC scheme is ultimately governed by the choice of model: the better the chosen model emulates the true system, the better the predictions and hence the tighter the control. The following paragraphs give a brief discussion of what the control engineer needs to consider in order to obtain a good model.

Domain

The first choice to consider is the model's domain. Traditionally this has been either the continuous-time (s -domain) or the discrete-time (z -domain). Since most real systems are continuous-time phenomena, the natural choice would be to use the s -domain to accurately model their behaviour since there would be no loss of information due to sampling. However almost all real-time control systems are implemented using digital computers due to the ease in which the data can be monitored, manipulated and stored for later analysis. This has meant that most predictive controllers in practice employ shift operator (z -domain) models whose coefficients do not generally bear any physical significance to the real plant. This is because the coefficients of a shift operator model do not equal the true system differential equation coefficients, even as the sample time tends to zero. Thus prior knowledge of the plant cannot be used to bound key parameters to lie within user defined limits, and thereby improve the quality of the estimated model. Middleton and Goodwin (1990) addressed this dichotomy by proposing the δ -domain which gives a unified framework for control and estimation theory in the continuous and discrete domains by approximating s by a simple function of z and sampling time h . Indeed the survey paper of Goodwin et al. (1992) shows that the δ -domain formulation can greatly improve the conditioning of specific problem

areas such as filtering, system identification and time-series modelling when implementing these methods on finite word-length digital computers and when high sampling rates are used.

Model structure

Ideally the best model structure would be that based on a detailed physical model of the process. However often a detailed model is not available due to lack of fundamental understanding of the process, or due to economic reasons. Thus the control engineer usually looks for a minimum set of (orthogonal) functions (i.e. a basis) which can adequately explain (i.e. span) the observed behaviour of the given process. The following lists a few typical model structures which have been successfully used in practice:

- Linear time series models, e.g. Auto-Regressive (AR), Auto-Regressive with an eXogenous input (ARX), Auto-Regressive Moving-Average with eXogenous input (ARMAX), CARIMA, Box–Jenkins (BJ), ... (e.g. see Ljung, 1987).
- Non-linear time series, e.g. Non-linear ARMAX (NARMAX) (Chen and Billings, 1989) which includes Hammerstein, Volterra, Weiner and bilinear models as special cases.
- Neural Networks, e.g. Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), Kohonen, ... (e.g. see Bishop, 1995).
- Fuzzy sets (e.g. see Zadeh, 1994).

where each tend to suit particular application areas. Expressing this another way: let us propose that, due to the practically limitless variety of phenomena that exist in the world, there can be no universal model structure which is a basis for them all. Therefore the choice of model structure must be *application driven*. This is where the rôle of the control engineer is paramount. The more insight (from physics/experience/intuition) that the control engineer uses when deciding on the model structure, the closer one gets to the basis set, and hence fewer parameters are needed to accurately describe the plant's behaviour. Having a smaller parameter search space makes it easier to obtain a good model

from the data and hence one would expect better performance from the predictive controller, especially in the adaptive sense. This grey-box modelling philosophy has been used very successfully by Hinton (1992) and Clarke and Hinton (1997) for the adaptive control of materials-testing machines.

Model complexity

In practice it is very rare to have an exact physical model of the process and thus we never get a true basis set for the model and a sub-optimal model structure (which requires more parameters) has to be used. Then the question of: “Exactly how many parameters do we need ?” arises. This will be referred to as *model complexity* in this sequel. If LS is used to obtain the model coefficients, the condition number of the matrix inverse involved determines whether the data can uniquely identify the model coefficients or not. Thus a high model complexity places a tougher requirement on the data (i.e. the conditioning of the estimator deteriorates with model complexity). So, in order to obtain good conditioning of the estimator, only “relevant” data should be presented to it. This conditioning problem can be partitioned into:

Static conditioning: Given a plant which has a large number of actuators (inputs) and sensors (outputs) we need to find out which inputs affect which outputs by using Relative Gain Array (RGA)/Singular Value Decomposition (SVD)/Eigen methods.

Dynamic conditioning: Given the input/output pairings from the static conditioning, which parameters are identifiable using the current window of past data ? For time series modelling this question is equivalent to asking: “Up to what order is the data sufficiently exciting ?”

In this thesis multiple order models will be used in an attempt to improve the dynamic conditioning of the on-line estimation of the GPC CARIMA model.

1.1.4 Predictive control using the δ -domain

Gawthrop (1987) proposes that the continuous-time approach to the *design* of self-tuning controllers has a number of advantages over discrete-time. The first being that no information is lost due to the sampling process, and secondly, continuous-time differential equation models bear some physical significance to the real system to be controlled. Hence Demircioğlu and Gawthrop (1991) developed the Continuous-time Generalised Predictive Control (CGPC) which displays similar features to the discrete-time GPC algorithm such as the conversion to ‘polynomial’ RST form (i.e. $R(s)u + S(s)y = T(s)w$) for *implementation* on digital microprocessors. The sampling involved in digital systems means that the final continuous-time controller design has to be transformed to the shift domain before implementation.

This dichotomy of continuous-time design and digital implementation of control systems can be *unified* by using the δ domain (Middleton and Goodwin, 1990) since the continuous-time results are obtained as a particular case of the discrete-time ones by letting the sampling time approach zero. Use of the δ -domain removes the need for *explicit* continuous-time to discrete-time conversion when implementing the control design. In addition the δ -operator offers a numerical advantage over the shift operator when implementing the controller filters R , S and T on restricted word-length embedded microprocessors that employ fast sampling (i.e. if the plant input/output data is sampled quickly with respect to the plant dynamics) in that the filter outputs lose numerical accuracy since all the poles tend towards the $(1 + 0j)$ in the z -plane. Thus the δ -operator is most suited to the polynomial form of predictive controllers such as Delta-operator Generalised Predictive Control (DGPC) (Neumann et al., 1992, 1993), Weighted Generalised Predictive Control (WGPC) of Thygesen (1993), and Delta-domain approximation of Continuous-time Generalised Predictive Control (DCGPC) and Delta-domain approximation of q -domain Generalised Predictive Control (DQGPC) of Lauritsen et al. (1997), which require a δ -domain model of the plant, and this motivated the derivation of batch (off-line) and recursive (on-line) multiple model LS in the δ -domain in chapter 4.

1.2 Adaptive predictive control

The automatic tuning of controllers to obtain a specified closed-loop behaviour for systems that are time-varying has excited both theoretical and practical interest for many years. Typically the LS method has been used in the calculation of the controller parameters, either directly or indirectly, leading to the implicit or explicit self-tuning control schemes respectively. This thesis considers the explicit self-tuning case shown in Figure 1.2 where the plant parameters are estimated using a recursive parameter estimator (Clarke, 1988).

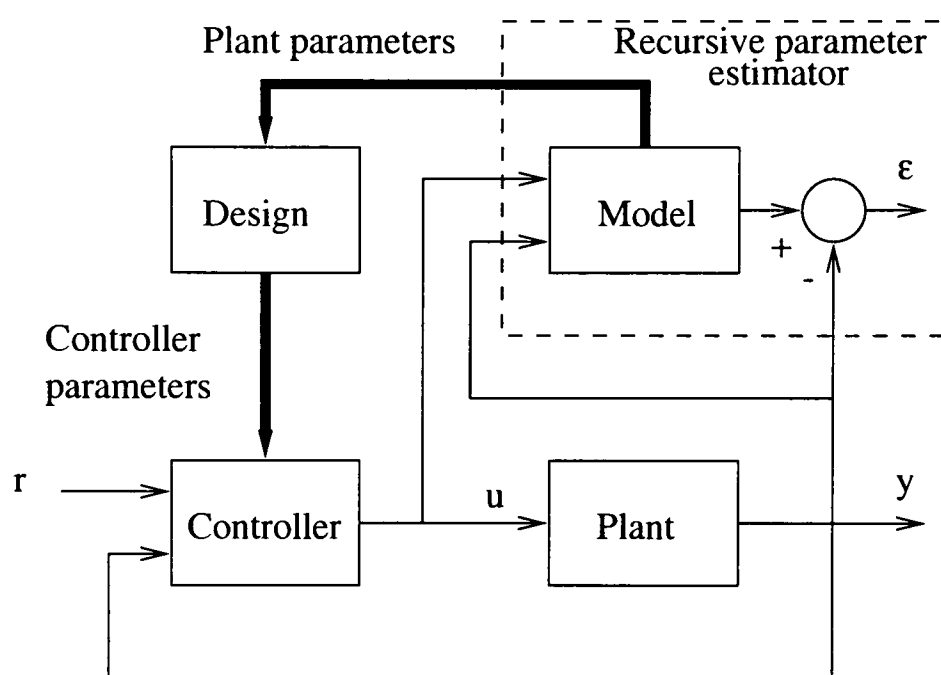


Figure 1.2: Block diagram of an explicit self-tuning control scheme

The basic equations for the recursive parameter estimator are now derived and some of the problems associated with them are discussed.

1.2.1 Ordinary Least-Squares

The classical LS problem lies in estimating the n parameters θ_n from input/output data $\{x(t), y(t)\}$ and additive noise $\{e(t)\}$ using the regression relation:

$$y(t) = \theta_1 x_1(t) + \theta_2 x_2(t) + \dots + \theta_n x_n(t) + e(t), \quad (1.11)$$

where $e(t)$ is the measurement error. This relation can be written in a compact vector form:

$$y(t) = \theta_n^T x(t) + e(t).$$

where $\boldsymbol{\theta}_n \in \mathbb{R}^n$ is a vector of unknown parameters and $\boldsymbol{x}(t) \in \mathbb{R}^n$ is a vector of known data.

The N stacked equations corresponding to successive time samples can be written:

$$\boldsymbol{y}_N = \boldsymbol{X}_N \boldsymbol{\theta}_n + \boldsymbol{e}_N,$$

where $\boldsymbol{y}_N$ and $\boldsymbol{e}_N$ are $N \times 1$ vectors, and $\boldsymbol{X}_N$ is a $N \times n$ matrix.

Given an estimate $\hat{\boldsymbol{\theta}}_n$ the model in eqn.(1.11) can be written:

$$y(t) = \hat{\boldsymbol{\theta}}_n^T \boldsymbol{x}(t) + \epsilon(t), \quad (1.12)$$

where $\epsilon(t)$ is the model's *prediction error*.

The LS solution minimises the loss (or cost) function:

$$J_n = \sum_{t=1}^N \epsilon^2(t) = \boldsymbol{\epsilon}_N^T \boldsymbol{\epsilon}_N \quad (1.13)$$

using $\nabla_{\hat{\boldsymbol{\theta}}_n} J_n = 0$, yielding:

$$\hat{\boldsymbol{\theta}}_n = \boldsymbol{S}_n^{-1} \boldsymbol{X}_N^T \boldsymbol{y}_N, \quad (1.14)$$

where the *information matrix* $\boldsymbol{S}_n = \boldsymbol{X}_N^T \boldsymbol{X}_N$.

The model's predictions are given by $\hat{\boldsymbol{y}}_N = \boldsymbol{X}_N \hat{\boldsymbol{\theta}}_n$, and if the parameter errors are denoted $\tilde{\boldsymbol{\theta}}_n = \boldsymbol{\theta}_n - \hat{\boldsymbol{\theta}}_n$, we get the *residual sequence*:

$$\begin{aligned} \boldsymbol{\epsilon}_N &= \boldsymbol{y}_N - \hat{\boldsymbol{y}}_N \\ &= \boldsymbol{X}_N \boldsymbol{\theta}_n + \boldsymbol{e}_N - \boldsymbol{X}_N \hat{\boldsymbol{\theta}}_n = \boldsymbol{X}_N \tilde{\boldsymbol{\theta}}_n + \boldsymbol{e}_N \\ &= \boldsymbol{X}_N \boldsymbol{\theta}_n + \boldsymbol{e}_N - \boldsymbol{X}_N \boldsymbol{S}_n^{-1} \boldsymbol{X}_N^T [\boldsymbol{X}_N \boldsymbol{\theta}_n + \boldsymbol{e}_N] \\ &= [\boldsymbol{I} - \boldsymbol{X}_N \boldsymbol{S}_n^{-1} \boldsymbol{X}_N^T] \boldsymbol{e}_N. \end{aligned}$$

Multiplication by $\boldsymbol{X}_N^T$ gives:

$$\boldsymbol{X}_N^T \boldsymbol{\epsilon}_N = (\boldsymbol{X}_N^T - \boldsymbol{X}_N^T) \boldsymbol{e}_N = \mathbf{0} \quad (1.15)$$

and hence we see that $\boldsymbol{X}_N \perp \boldsymbol{\epsilon}_N$, i.e. the data are *orthogonal* to the residuals which indicates that the estimate is the *best* we can do given the data set $\boldsymbol{X}_N$.

1.2.2 Recursive Least-Squares

The Recursive Least-Squares (RLS) equations are obtained by updating the information matrix with the new data $\mathbf{x}(t)$ and observation $y(t)$ at each successive time step as follows:

$$\mathbf{S}_n(t) = \beta \mathbf{S}_n(t-1) + \mathbf{x}(t)\mathbf{x}^T(t),$$

where β is the exponential forgetting-factor (Clarke, 1981a). The inverse of $\mathbf{S}_n(t)$ needed in the computation of the estimates in eqn.(1.14) can be computed recursively using the matrix inversion lemma, and leads to the standard RLS equations:

$$\text{Kalman gain vector:} \quad \mathbf{k}(t) = \frac{\mathbf{P}_n(t-1)\mathbf{x}(t)}{\beta + \mathbf{x}^T(t)\mathbf{P}_n(t-1)\mathbf{x}(t)} \quad (1.16)$$

$$\text{Parameter update:} \quad \hat{\boldsymbol{\theta}}_n(t) = \hat{\boldsymbol{\theta}}_n(t-1) + \mathbf{k}(t)\epsilon(t) \quad (1.17)$$

$$\text{Covariance update:} \quad \mathbf{P}_n(t) = [\mathbf{I} - \mathbf{k}(t)\mathbf{x}^T(t)] \mathbf{P}_n(t-1)/\beta \quad (1.18)$$

where $\epsilon(t) = y(t) - \hat{\boldsymbol{\theta}}_n^T(t-1)\mathbf{x}(t)$ is the *a priori* prediction error.

It is well known that implementing the standard RLS equations (1.16–1.18) for a standard LS regression model such as that of eqn.(1.11) leads to poor numerical performance. Graupe et al. (1980) shows that the standard RLS equations can give estimates that are meaningless when the covariance matrix becomes ill-conditioned due to the finite word-length inherent to computer implementation. In (Clarke, 1981b) it is shown that instability may even result when using the standard RLS equations in which the covariance matrix becomes negative definite. Thus in practice most self-tuning and adaptive systems use Bierman's UD filter (Bierman, 1976; Thornton and Bierman, 1978) to implement the recursive parameter estimator due to its good stability properties and numerical accuracy (see Bierman (1977) chapter 5, Mohtadi (1988a), and Åström and Wittenmark (1989) chapter 3.6). It has also been shown by Graupe et al. (1980) and Ljung and Ljung (1985) that the UD filter approach compares favourably, in terms of execution time as well as numerical performance, with other recursive implementations such as lattice/ladder algorithms.

1.3 The need for multiple order models in predictive control

Most of the work in self-tuning control to date has assumed that the model of the plant is of fixed order which often leads to problems associated with under/over-parameterisation of the assumed plant model. Poor modelling associated with under-parameterisation has obvious drawbacks for use in an adaptive control scheme due to the model's inability to capture the full behaviour of the process under control. Thus only the problem of over-parameterising the model structure is addressed in this section.

Consider a special case where the model is an AR:

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = e(t).$$

Using the past output measurements as regressors is natural, however one disadvantage is that the coefficients a_i depend on the model order n . For example, consider the system:

$$y(t) + a_1 y(t-1) + a_2 y(t-2) = e(t),$$

where $\{e(t)\}$ is a sequence of independent random variables with unit variance and zero mean. The output sequence $\{y(t)\}$ has a covariance function

$$r_k = E y(t) y(t-k),$$

where elementary calculation gives

$$r_0 = \frac{1 + a_2}{((1 + a_2)^2 - a_1^2)(1 - a_2)}, \quad r_1 = -\frac{a_1}{1 + a_2} r_0 \quad \text{and} \quad r_2 = -a_2 r_0 - a_1 r_1.$$

The *first order* model will be

$$\hat{y}(t) = -\hat{a}_1^{(1)} y(t-1),$$

with asymptotic estimate

$$\hat{a}_1^{(1)} = \lim_{t \rightarrow \infty} \left\{ \left(\sum_{k=1}^t y^2(k-1) \right)^{-1} \sum_{k=1}^t -y(k-1)y(k) \right\} = -r_1/r_0 = \frac{a_1}{1 + a_2}.$$

A *second order* model

$$\hat{y}(t) = -\hat{a}_1^{(2)} y(t-1) - \hat{a}_2^{(2)} y(t-2)$$

will asymptotically give the true values, i.e.

$$\hat{a}_1^{(2)} \rightarrow a_1 \quad \text{and} \quad \hat{a}_2^{(2)} \rightarrow a_2.$$

In particular, we notice that the asymptotic values of $\hat{a}_1^{(1)}$ and $\hat{a}_1^{(2)}$ are not equal if $a_2 \neq 0$. Indeed, even if the true value of a_2 is zero (so that the limit of $\hat{a}_1$ is independent on the model order) the accuracy is affected, as we will now demonstrate.

We know that the asymptotic covariance matrix for

$$\sqrt{t}(\hat{a}_1^{(1)}(t) - a_1) \quad \text{is} \quad (Ey^2(t))^{-1} = 1/r_0 \quad (\text{First order model}).$$

For the second order model the covariance matrix for

$$\sqrt{t} \begin{pmatrix} \hat{a}_1^{(2)}(t) - a_1 \\ \hat{a}_2^{(2)}(t) \end{pmatrix} \quad \text{is} \quad \begin{pmatrix} r_0 & r_1 \\ r_1 & r_0 \end{pmatrix}^{-1},$$

when $a_2 = 0$ and thus the asymptotic variance

$$\sqrt{t}(\hat{a}_1^{(2)}(t) - a_1) \quad \text{is} \quad \frac{1}{r_0} \left(\frac{1}{1 - a_1^2} \right) \quad (\text{Second order model})$$

which is strictly greater than that of the first order model. Hence there is indeed a penalty for going beyond the true model order.

One way to avoid these problems would be to estimate a set of models of different order and to choose the model which manages to fit the data without using an excessive number of parameters, i.e. the model is *parsimonious* (Ljung and Söderström, 1983) with respect to the model parameterisation.

1.4 Existing methods for the estimation of multiple order models

The simultaneous estimation of reduced order models for automatic order selection is a well-known concept which has lead to the development of Lattice (or Ladder) structures and Levinson-type algorithms for block AR models, and more recently the *AUDZ* (Niu et al., 1992) family of algorithms. These algorithms are now briefly discussed to highlight the state-of-the-art and also to set the stage for a comparison between them in section 2.2.

1.4.1 Least-Squares Lattice algorithm

The reason why the estimates of an AR model change when the model order is increased can be described as follows. Suppose we have estimated an n^{th} order AR model

$$\hat{y}^{(n)}(t) = \hat{\boldsymbol{\theta}}^{(n)T} \mathbf{x}(t)$$

where $\hat{\boldsymbol{\theta}}^{(n)} = (\hat{a}_1 \dots \hat{a}_n)^T$ are the parameter estimates and $\mathbf{x}(t)$ contains the past measurements $y(t-i)$, $1 \leq i \leq n$. Now if we increase the order to $n+1$ and add the regressor $y(t-n-1)$ we get:

$$\hat{y}^{(n+1)}(t) = \hat{\boldsymbol{\theta}}^{(n)T} \mathbf{x}(t) - \hat{a}_{n+1}y(t-n-1)$$

where $y(t-n-1)$ is typically correlated with $\mathbf{x}(t)$, but contains some new information that is not present in $\mathbf{x}(t)$ i.e.

$$y(t-n-1) = -\mathbf{b}^T \mathbf{x}(t) + r_n(t-1) \quad (1.19)$$

where $r_n(t-1)$ is the *innovation*, or new piece of information contained in $y(t-n-1)$ which is not present in $\mathbf{x}(t)$, and thus $r_n(t-1)$ and $\mathbf{x}(t)$ are not correlated. Thus

$$\hat{y}^{(n+1)}(t) = \left(\hat{\boldsymbol{\theta}}^{(n)} + \hat{a}_{n+1}\mathbf{b} \right)^T \mathbf{x}(t) - \hat{a}_{n+1}r_n(t-1),$$

and so

$$\hat{\boldsymbol{\theta}}^{(n+1)} = \begin{pmatrix} \hat{\boldsymbol{\theta}}^{(n)} + \hat{a}_{n+1}\mathbf{b} \\ \hat{a}_{n+1} \end{pmatrix}$$

where $\hat{\boldsymbol{\theta}}^{(n+1)}$ are the estimates for the $n+1^{th}$ order AR model. Thus the higher order model estimates differ from the lower order estimates by $\hat{a}_{n+1}\mathbf{b}$.

If, when we extended the model to account for the variable $y(t-n-1)$ we had added only the new information $r_n(t-1)$ instead of $y(t-n-1)$, then the first n estimates of $\boldsymbol{\theta}^{(n+1)}$ would be unaffected by this order increase. Thus we should use the innovations $r_k(t-1)$, ($k = 0, \dots, n$) as the regressors instead of the past observations $y(t-k-1)$. From eqn.(1.19) we see that $r_n(t-1)$ can be interpreted as the *backward prediction error* associated with predicting $y(t-n-1)$ from past observations $\mathbf{x}(t)$.

In addition, we see from eqn.(1.19) that $r_k(t-1)$ is a linear combination of $\{y(t-n), \dots, y(t-n+k-1), y(t-n+k) \text{ (with coefficient 1)}\}$, and hence

$$\tilde{\mathbf{x}}(t) = \begin{pmatrix} r_0(t-1) \\ \vdots \\ r_{n-1}(t-1) \end{pmatrix} = \begin{pmatrix} 1 & & & \\ b_1^{(1)} & 1 & & \\ \vdots & & \ddots & \\ b_n^{(n)} & \dots & b_1^{(n)} & 1 \end{pmatrix} \begin{pmatrix} y(t-n) \\ y(t-n+1) \\ \vdots \\ y(t-1) \end{pmatrix} = \mathbf{L}_b(t)\mathbf{x}(t), \quad (1.20)$$

where $\mathbf{L}_b(t)$ is a unit lower triangular matrix.

The transformation from the original regressors $y(t-k)$ to the backward innovations $r_k(t-1)$ is therefore a linear change of basis in the regressor space (see Appendix A).

The corresponding change of coefficients is:

$$\tilde{\boldsymbol{\theta}} = \begin{pmatrix} K_0^b \\ \vdots \\ K_{n-1}^b \end{pmatrix} = \mathbf{L}_b^{-T}(t)\boldsymbol{\theta}, \quad (1.21)$$

where K_i^b are usually known as *reflection coefficients* (from an analogy with transmission line theory) or *partial correlation coefficients* for the backward predictions $\mathbf{r}$. The n^{th} order model is thus:

$$\hat{y}^{(n)}(t) = \tilde{\boldsymbol{\theta}}^T \tilde{\mathbf{x}}(t) = K_0^b r_0(t-1) + \dots + K_{n-1}^b r_{n-1}(t-1). \quad (1.22)$$

Looking at the residuals $e_n(t) = y(t) - \hat{y}^{(n)}(t)$, it is evident from eqn.(1.22) that

$$\hat{y}^{(n+1)}(t) = \hat{y}^{(n)}(t) + K_n^b r_n(t-1),$$

hence we get an order recursive formula for computing the residuals:

$$e_{n+1}(t) = e_n(t) + K_n^b r_n(t-1). \quad (1.23)$$

We also see that:

$$\begin{aligned} e_0(t-n) &= y(t-n) \\ e_1(t-n+1) &= y(t-n+1) - \hat{y}^{(1)}(t-n+1) \\ &\vdots \\ e_n(t) &= y(t) - \hat{y}^{(n)}(t) \end{aligned}$$

i.e.

$$\tilde{\mathbf{x}}(t) = \begin{pmatrix} e_0(t-n) \\ e_1(t-n+1) \\ \vdots \\ e_n(t) \end{pmatrix} = \begin{pmatrix} 1 & & & \\ a_1^{(1)} & 1 & & \\ \vdots & & \ddots & \\ a_n^{(n)} & \dots & a_1^{(n)} & 1 \end{pmatrix} \begin{pmatrix} y(t-n) \\ y(t-n+1) \\ \vdots \\ y(t) \end{pmatrix} = \mathbf{L}_f(t)\mathbf{x}(t). \quad (1.24)$$

Again, this is a linear change of basis in the regressor space (from output measurements $y(t-k)$ to forward residuals $e_k(t-n+k)$) with corresponding parameters:

$$\tilde{\boldsymbol{\theta}} = \begin{pmatrix} K_0^f \\ \vdots \\ K_{n-1}^f \end{pmatrix} = \mathbf{L}_f^{-T}(t)\boldsymbol{\theta}, \quad (1.25)$$

and so the n^{th} order model is:

$$\hat{y}^{(n)}(t-n-1) = \tilde{\boldsymbol{\theta}}^T \tilde{\mathbf{x}}(t) = K_0^f e_0(t-n) + \dots + K_{n-1}^f e_{n-1}(t-1). \quad (1.26)$$

It is clear from eqn.(1.26) that

$$\hat{y}^{(n)}(t-n-1) = \hat{y}^{(n-1)}(t-n-1) + K_n^f e_n(t-1),$$

hence we get an order recursive formula for computing the backward residuals:

$$r_{n+1}(t) = r_n(t-1) + K_n^f e_n(t). \quad (1.27)$$

Equations (1.23) and (1.27) are known as the lattice equations as their implementation of an AR model has a lattice structure shown in Figure 1.3.

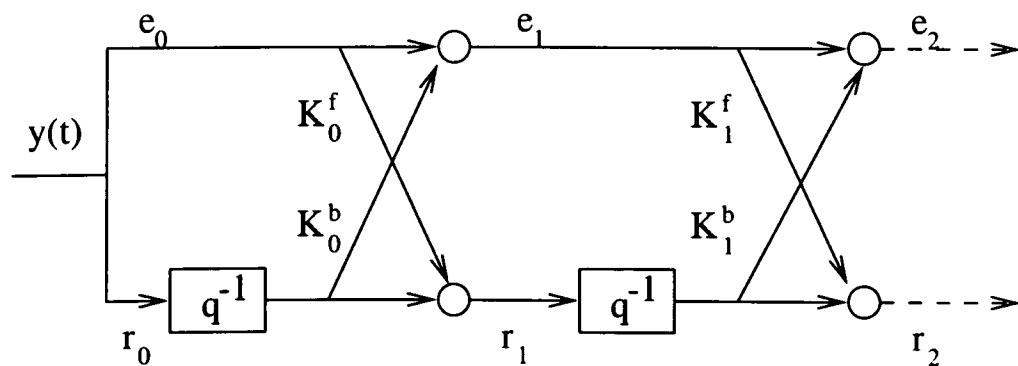


Figure 1.3: The lattice structure

The covariance matrix of the asymptotic distribution for $\sqrt{t}(\hat{\boldsymbol{\theta}}(t) - \tilde{\boldsymbol{\theta}})$ is proportional to:

$$\tilde{\mathbf{P}} = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{k=1}^t E(\tilde{\mathbf{x}}(t) \tilde{\mathbf{x}}^T(t))^{-1},$$

which is diagonal and hence the estimates of the different components of $\tilde{\boldsymbol{\theta}}$ are asymptotically uncorrelated. Therefore neither the value nor the accuracy of the reduced order reflection coefficients are affected by the estimation of the higher order reflection coefficients. This order-independence property gives rise to the Least-Squares Lattice (LSL) algorithm given in Table 1.1 (e.g. see Ljung and Söderström, 1983, chapter 6.4). This is in contrast to the

Table 1.1: The Least-Squares Lattice algorithm

$e_0 = -y(t)$	
$r_0 = -y(t-1)$	
$\lambda = 0$	
for $i = 1$ to m	
$K_i^f = -R_i^{re} / R_i^r$	reflection coefficients
$K_i^b = -R_i^{re} / R_i^e$	
$e_i = e_{i-1} + K_i^b r_{i-1}$	forward residuals
$r_i = r_i + K_i^f e_{i-1}$	backwards innovations
$R_i^r = \beta R_i^r + (1 - \lambda) r_{i-1}^2$	sample covariances update
$R_i^e = \beta R_i^e + (1 - \lambda) e_{i-1}^2$	
$R_i^{re} = \beta R_i^{re} + (1 - \lambda) r_{i-1} e_{i-1}$	
$\lambda = \lambda + \mu^2 r_{i-1}^2 / R_i^r$	
end	

standard LS approach described earlier.

1.4.2 Levinson type algorithms

Consider the following set of linear equations:

$$\left. \begin{aligned} \mathbf{R}_k \boldsymbol{\theta}_k &= -\mathbf{r}_k \\ r_0 + \mathbf{r}_k^T \boldsymbol{\theta}_k &= \sigma_k^2 \end{aligned} \right\} \quad k = 1, \dots, n \quad (1.28)$$

where σ_k^2 is the mean-square error,

$$\mathbf{R}_k = \begin{pmatrix} r_0 & r_1 & \cdots & r_{k-1} \\ r_1 & r_0 & & r_{k-2} \\ \vdots & & \ddots & \\ r_{k-1} & r_{k-2} & & r_0 \end{pmatrix} \quad \mathbf{r}_k = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{pmatrix} \quad \boldsymbol{\theta}_k = \begin{pmatrix} a_{k1} \\ a_{k2} \\ \vdots \\ a_{kk} \end{pmatrix}$$

and where $\{r_i\}_{i=0}^n$ are given. Such systems of equations appear, for example, when fitting AR models of orders $k = 1, 2, \dots, n$ to given (or estimated) sample covariances $\{r_i\}$ of the data by the Yule-Walker method. These equations can be written in the following compact form:

$$\begin{pmatrix} r_0 & \mathbf{r}_k^T \\ \mathbf{r}_k & \mathbf{R}_k \end{pmatrix} \begin{pmatrix} 1 \\ \boldsymbol{\theta}_k \end{pmatrix} = \mathbf{R}_{k+1} \begin{pmatrix} 1 \\ \boldsymbol{\theta}_k \end{pmatrix} = \begin{pmatrix} \sigma_k^2 \\ 0 \end{pmatrix} \quad k = 1, \dots, n$$

Premultiplying by the vector $(1 \ \boldsymbol{\theta}_k^T)$ gives:

$$(1 \ \boldsymbol{\theta}_k^T) \begin{pmatrix} r_0 & \mathbf{r}_k^T \\ \mathbf{r}_k & \mathbf{R}_k \end{pmatrix} \begin{pmatrix} 1 \\ \boldsymbol{\theta}_k \end{pmatrix} = (1 \ \boldsymbol{\theta}_k^T) \mathbf{R}_{k+1} \begin{pmatrix} 1 \\ \boldsymbol{\theta}_k \end{pmatrix} = \sigma_k^2 \quad k = 1, \dots, n$$

where σ_k^2 is a *scalar* and hence we can rewrite the set of equations as a single matrix equation:

$$\underbrace{\begin{pmatrix} 1 & a_{n1} & \cdots & a_{nn} \\ & 1 & & \vdots \\ & & \ddots & a_{11} \\ & & & 1 \end{pmatrix}}_{\mathbf{U}_{n+1}} \underbrace{\begin{pmatrix} r_0 & r_1 & \cdots & r_n \\ r_1 & r_0 & & r_{n-1} \\ \vdots & & \ddots & \\ r_n & r_{n-1} & & r_0 \end{pmatrix}}_{\mathbf{R}_{n+1}} \underbrace{\begin{pmatrix} 1 & & & \\ a_{n1} & 1 & & \\ \vdots & & \ddots & \\ a_{nn} & \cdots & a_{11} & 1 \end{pmatrix}}_{\mathbf{U}_{n+1}^T} = \underbrace{\begin{pmatrix} \sigma_n^2 & & & \\ & \ddots & & \\ & & \sigma_1^2 & \\ & & & r_0 \end{pmatrix}}_{\mathbf{D}_{n+1}}$$

due to the fact that all the lower order $\{\mathbf{R}_i\}$ matrices are contained in $\mathbf{R}_{n+1}$. Taking the inverse and re-arranging we get that

$$\mathbf{R}_{n+1}^{-1} = \mathbf{U}_{n+1}^T \mathbf{D}_{n+1}^{-1} \mathbf{U}_{n+1}$$

i.e. the parameters for each order model and their associated mean-square errors can be obtained by decomposing the inverse of the sample covariance matrix $\mathbf{R}_{n+1}^{-1}$ into LDL^T form. This will be referred to as the *direct LDU method* and is an $O(n^3)$ algorithm as shown later in section 2.2.3.

We now derive the Levinson-Durbin algorithm (LDA) (Levinson, 1947; Durbin, 1960) which is a “fast”, $O(n^2)$, algorithm by exploiting both the Toeplitz structure of $\mathbf{R}_{n+1}$ and the *nested* structure:

$$\mathbf{R}_{n+1}^{-1} = \begin{pmatrix} 1 & 0 \\ \boldsymbol{\theta}_n & 1 \end{pmatrix} \begin{pmatrix} \sigma_n^{-2} & 0 \\ 0 & \mathbf{R}_n^{-1} \end{pmatrix} \begin{pmatrix} 1 & \boldsymbol{\theta}_n^T \\ 0 & 1 \end{pmatrix}$$

Substituting $\mathbf{R}_n^{-1} = \mathbf{U}_n^T \mathbf{D}_n^{-1} \mathbf{U}_n$ into the equation above gives the block LDL^T form:

$$\mathbf{R}_{n+1}^{-1} = \begin{pmatrix} 1 & 0 \\ \boldsymbol{\theta}_n & \mathbf{U}_n^T \end{pmatrix} \begin{pmatrix} \sigma_n^{-2} & 0 \\ 0 & \mathbf{D}_n^{-1} \end{pmatrix} \begin{pmatrix} 1 & \boldsymbol{\theta}_n^T \\ 0 & \mathbf{U}_n \end{pmatrix}.$$

Now using the Toeplitz structure of the matrices $\mathbf{R}_k$,

$$\boldsymbol{\theta}_{k+1}^R = -\mathbf{R}_{k+1}^{-1} \mathbf{r}_{k+1}^R$$

where the operator $\{\cdot\}^R$ signifies the elements in the vector are reversed, and inserting the factorisation of $\mathbf{R}_{k+1}^{-1}$ gives

$$\begin{aligned} \boldsymbol{\theta}_{k+1}^R &= - \begin{pmatrix} 1 & 0 \\ \boldsymbol{\theta}_k & \mathbf{U}_k^T \end{pmatrix} \begin{pmatrix} \sigma_k^{-2} & 0 \\ 0 & \mathbf{D}_k^{-1} \end{pmatrix} \begin{pmatrix} 1 & \boldsymbol{\theta}_k^T \\ 0 & \mathbf{U}_k \end{pmatrix} \begin{pmatrix} r_{k+1} \\ \mathbf{r}_k^R \end{pmatrix} \\ &= - \begin{pmatrix} 1/\sigma_k^2 & \boldsymbol{\theta}_k^T/\sigma_k^2 \\ \boldsymbol{\theta}_k/\sigma_k^2 & \mathbf{R}_k^{-1} + \boldsymbol{\theta}_k \boldsymbol{\theta}_k^T/\sigma_k^2 \end{pmatrix} \begin{pmatrix} r_{k+1} \\ \mathbf{r}_k^R \end{pmatrix} \\ &= \begin{pmatrix} -(r_{k+1} + \boldsymbol{\theta}_k^T \mathbf{r}_k^R)/\sigma_k^2 \\ -\boldsymbol{\theta}_k(r_{k+1} + \boldsymbol{\theta}_k^T \mathbf{r}_k^R)/\sigma_k^2 + \boldsymbol{\theta}_k^R \end{pmatrix} = \begin{pmatrix} a_{k+1,k+1} \\ \boldsymbol{\theta}_k^R + a_{k+1,k+1} \boldsymbol{\theta}_k \end{pmatrix}. \end{aligned}$$

We also have that

$$\begin{aligned} \sigma_{k+1}^2 &= r_0 + \boldsymbol{\theta}_{k+1}^T \mathbf{r}_{k+1} = r_0 + \boldsymbol{\theta}_{k+1}^{RT} \mathbf{r}_{k+1}^R \\ &= r_0 + \begin{pmatrix} a_{k+1,k+1} \\ \boldsymbol{\theta}_k^R + a_{k+1,k+1} \boldsymbol{\theta}_k \end{pmatrix}^T \begin{pmatrix} r_{k+1} \\ \mathbf{r}_k^R \end{pmatrix} \\ &= \underbrace{r_0 + \boldsymbol{\theta}_k^{RT} \mathbf{r}_k^R}_{\sigma_k^2} + a_{k+1,k+1} \underbrace{(r_{k+1} + \boldsymbol{\theta}_k^T \mathbf{r}_k^R)}_{-\sigma_k^2 a_{k+1,k+1}} \end{aligned}$$

$$= \sigma_k^2 (1 - a_{k+1,k+1}^2)$$

Thus the LDA can be implemented as shown in Table 1.2 (e.g. see Ljung, 1987, chapter 10). where the value of ρ_k can be shown to be equal to the reflection coefficient $K_i^b = K_i^f$

Table 1.2: The Levinson-Durbin algorithm

$\rho_0 = -r_1/r_0$
$a_{1,1} = \rho_0$
$\sigma_1^2 = r_0 (1 - \rho_0^2)$
for $k = 1$ to $n - 1$
$\rho_k = -(r_{k+1} + a_{k,1}r_k + \dots + a_{k,k}r_1) / \sigma_k^2$
$a_{k+1,k+1} = \rho_k$
$a_{k+1,i} = a_{k,i} + \rho_k a_{k,k+1-i} \quad i = 1, \dots, k$
$\sigma_{k+1}^2 = \sigma_k^2 (1 - \rho_k^2)$
end

if $\{y(t)\}$ is a wide-sense-stationary random process (Goodwin and Sin, 1984).

1.4.3 The Augmented UD Identification method

Consider the classical LS relation of eqn.(1.11). Now suppose that the regression vector $\mathbf{x}(t)$ contains past outputs *interleaved* with past inputs, for example

$$\mathbf{x}(t) = [\underline{-y(t-2) \ u(t-2)}, \underline{-y(t-1) \ u(t-1)}]^T$$

and moving the observation to the right hand side, we get a 2^{nd} order (4-parameter) ARX model:

$$\begin{aligned} 0 &= a_2^{(2)}(-y(t-2)) + b_2^{(2)}u(t-2) + a_1^{(2)}(-y(t-1)) + b_1^{(2)}u(t-1) + (-y(t)) + e(t) \\ &= [\boldsymbol{\theta}_4^T \ 1] \begin{bmatrix} \mathbf{x}(t) \\ -y(t) \end{bmatrix} + e(t) \end{aligned}$$

where $\boldsymbol{\theta}_4^T = [a_2^{(2)} \ b_2^{(2)}, \ a_1^{(2)} \ b_1^{(2)}]$.

The interleaving of past outputs provides reduced order models by simply truncating the parameter vector; in our example the 1st order (2-parameter) ARX can be obtained as follows:

$$\begin{aligned} 0 &= a_1^{(1)}(-y(t-2)) + b_1^{(1)}u(t-2) + (-y(t-1)) + e(t) \\ &= [\theta_2^T \ 1 \ 0 \ 0] \begin{bmatrix} x(t) \\ -y(t) \end{bmatrix} + e(t) \end{aligned}$$

where $\theta_2^T = [a_1^{(1)} \ b_1^{(1)}]$.

A common approach for obtaining parameters θ_{2i} for models of order $i = 0, \dots, m$ simultaneously is via the LDA (Söderström and Stoica, 1989) which assumes a block AR form and exploits the Toeplitz structure of the correlation matrix, hence leading to a fast order-recursive solution. The requirement of Toeplitz structure implies that for finite data sets the LDA solution is not a LS solution. In particular it precludes the minimum requirement of n -samples to solve for n parameters in the noise-free case.

Recently, Niu and co-workers (Niu et al., 1992; Niu, 1993; Niu and Fisher, 1993; Niu et al., 1995a) have developed the *AUDI* family of algorithms which is an extension to LS and is able to handle more general model structures than the LDA. In addition, since the *AUDI* method relies on a UD factorisation, it is also inherently numerically superior to the LDA. In particular, since the time-recursive version is based on Bierman's UD filter Bierman (1977) which is the method of choice for parameter estimation in self-tuning applications, the lower order models with their associated loss function values can be obtained automatically at no extra computational cost.

The *AUDI* method is based on the following reformulation of the ordinary LS regression vector for an m^{th} -order ARX model:

$$\begin{aligned} \phi(t) &= \left[\underline{-y(t-m) \ u(t-m)} \ \dots \ \underline{-y(t-1) \ u(t-1)} \ -y(t) \right]^T \\ \theta(t) &= [a_m \ b_m \ \dots \ a_1 \ b_1 \ 1]^T \end{aligned}$$

where the past inputs and outputs are *paired* and the latest observation is included as the last element of the regression, and hence $\phi(t)$ is known as the *augmented data vector*.

Then by constructing the Augmented Information Matrix (AIM):

$$\mathbf{S}_\phi(t) = \sum_{k=0}^t \phi(k)\phi^T(k)$$

and decomposing the inverse of the AIM which is called the Augmented Covariance Matrix (ACM):

$$\mathbf{P}_\phi(t) = \mathbf{S}_\phi^{-1}(t)$$

into $\mathbf{UDU}^T$ form one obtains the $\mathcal{AUDI}$ structure:

$$\mathbf{U} = \begin{bmatrix} 1 & \hat{\alpha}_1^{(0)} & \hat{\theta}_1^{(1)} & \dots & \hat{\alpha}_1^{(m)} & \hat{\theta}_1^{(m)} \\ & 1 & \hat{\theta}_2^{(1)} & & \hat{\alpha}_2^{(m)} & \hat{\theta}_2^{(m)} \\ & & 1 & & \vdots & \vdots \\ & & & \ddots & \hat{\alpha}_{m-1}^{(m)} & \hat{\theta}_{m-1}^{(m)} \\ & & & & 1 & \hat{\theta}_m^{(m)} \\ & & & & & 1 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} \frac{1}{J_f^{(0)}} & & & & & \\ & \frac{1}{J_b^{(0)}} & & & & \\ & & \frac{1}{J_f^{(1)}} & & & \\ & & & \ddots & & \\ & & & & \frac{1}{J_b^{(m-1)}} & \\ & & & & & \frac{1}{J_f^{(m)}} \end{bmatrix}$$

where $\hat{\theta}^{(k)}$ are the forward model LS estimates and $\hat{\alpha}^{(k)}$ are the reverse model LS estimates with corresponding loss functions $J_f^{(k)}$ and $J_b^{(k)}$ for the set $(0 \leq k \leq m)$ of models defined by:

$$\mathbf{U}^T(t)\phi(t) = \epsilon(t) \quad (1.29)$$

where $\epsilon(t)$ are the LS residuals for each model. It is not difficult to show that eqn.(1.29) is equivalent to writing the following $(2m + 1)$ equations:

$$\begin{bmatrix} 0 \\ -y(t-m) \\ -y(t-m), u(t-m) \\ \vdots \\ -y(t-m), u(t-m), -y(t-m+1), \dots, -y(t-1) \\ -y(t-m), u(t-m), -y(t-m+1), \dots, -y(t-1), u(t-1) \end{bmatrix} \Rightarrow \begin{matrix} y(t-m) \\ u(t-m) \\ y(t-m+1) \\ \vdots \\ u(t-1) \\ y(t) \end{matrix}$$

where “ $\Rightarrow$ ” denotes the best (in the least-squares sense) linear combination of the variables on the left hand side to fit/predict the variable on the right hand side.

The proof of the $\mathcal{AUDI}$ method, as derived by Niu, is given in Appendix C: it employs a ‘brute-force’ approach and hence is rather tedious and does not give much insight into the

AUDZ technique. We shall see later in section 2.1 that a more elegant approach is to use a Lagrange multiplier.

The loss functions obtained from the AUDZ method are very useful and can be used to select appropriate model order, thereby avoiding problems associated with over- and under-parameterisation. Indeed, unlike classical LS where the covariance matrix loses full rank if the system is over-parameterised, only the over-parameterised models of AUDZ are affected and *not* the lower order models (Niu and Ljung, 1995). The Levinson and LSL type algorithms also share this robustness to over-parameterisation, however the AUDZ method is superior to Levinson and LSL methods due to the ability one has to utilise the loss functions to test for the *identifiability* of each model (Niu and Fisher, 1994), and also to select models with appropriate order.

1.5 Organisation of the thesis

The structure of the thesis is outlined in the following paragraphs.

Chapter 2 presents an approach to Niu's AUDZ algorithm based on classical LS theory in which the problem is posed as an equality constrained optimisation. The resulting multiple model LS algorithm is then compared with existing multiple model algorithms in terms of speed, convergence and stability. The recursive version is then used with appropriate jacketing to obtain an on-line robust estimator for linear, time-varying processes that display variations in dynamics, model order and dead-time. Simulations are given to demonstrate the utility of the multiple model RLS estimator.

Chapter 3 deals with the parameter tracking problem. A brief taxonomy of key parameter tracking concepts is presented in the continuous-time context using the basic differential equations for LMS and RLS. The continuous-time formulation was chosen as it is considered to give the user greater insight into the behaviour of each technique. A comparison between the various RLS-based methods and the LMS estimator is also performed.

Chapter 4 gives a derivation of the multiple model LS algorithm in the δ -domain. The derivation also employs a Lagrange multiplier technique and is developed for both batch and recursive versions of the method. Simulations are performed to show how this multiple model technique can be used to obtain models for δ -domain based GPC methods and, in the limit where the sampling time tends to zero, continuous-time based GPC. The model order is selected according to the model with the lowest loss function value.

Chapter 5 uses the multiple model LS technique with the appropriate estimator jacketing to provide an adaptive, multiple model based prediction control scheme with input and output constraints. This multiple model based predictive controller is then tested on a coupled tanks rig where the levels of two tanks are controlled using two pumps whilst constraining the level of the third tank to remain within some specified range. The effects of under/over-parameterisation of the plant model are then explored by comparing the performance of a multiple model based predictive controller to that of fixed model predictive control.

Chapter 6 draws conclusions from the research and discusses avenues for future work.

Chapter 2

Simultaneous estimation of multiple order models on-line

Clarke (1995) developed a simple derivation of Niu's $\mathcal{AUDI}$ algorithm using classical LS theory in which the problem is posed as an equality constrained optimisation. In this chapter the resulting Multiple Model Least-Squares (MMLS) algorithm is then compared with existing multiple model algorithms in terms of speed, convergence and stability. The recursive version is then used with appropriate jacketing to obtain an on-line robust estimator for linear, time-varying processes which display variations in dynamics, model order and dead-time.

2.1 Multiple Model Least-Squares

The key to MMLS (Clarke, 1995; Bowyer and Clarke, 1996) involves moving the observation $y(t)$ in the standard LS representation over to the right hand side giving

$$0 = \theta_1 x_1(t) + \theta_2 x_2(t) + \dots + \theta_n x_n(t) \underline{-y(t)} + e(t),$$

or in vector form:

$$0 = \boldsymbol{\theta}_{n+1}^T \boldsymbol{\phi}(t) + e(t),$$

where $e(t)$ is the measurement error and the augmented vectors are:

$$\boldsymbol{\theta}_{n+1} = \begin{bmatrix} \boldsymbol{\theta}_n \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1} \quad \text{and} \quad \boldsymbol{\phi}(t) = \begin{bmatrix} \mathbf{x}(t) \\ -y(t) \end{bmatrix} \in \mathbb{R}^{n+1}.$$

Note that the coefficient associated with the current output $y(t)$ is simply *one*. The $\mathcal{AUDI}$ structure requires that the past input/output data are *interleaved* and for example in control applications employing an ARX model, $\phi^T(t) = [-y(t-m) u(t-m) \cdots -y(t-1) u(t-1) - y(t)]$, where m is the maximum order. The stacked equations obtained from N successive observations are then:

$$\mathbf{0} = [\mathbf{X}_N \quad -\mathbf{y}_N] \begin{bmatrix} \hat{\boldsymbol{\theta}}_n \\ 1 \end{bmatrix} + \boldsymbol{\epsilon}_N = \boldsymbol{\Phi}_N \hat{\boldsymbol{\theta}}_{n+1} + \boldsymbol{\epsilon}_N,$$

where $\hat{\boldsymbol{\theta}}_n$ is the LS estimate of $\boldsymbol{\theta}_n$ and $\boldsymbol{\epsilon}_N$ are the LS residuals.

Since we know that the $(n+1)^{th}$ parameter must be equal to one, a Lagrange multiplier can be used to solve for $\hat{\boldsymbol{\theta}}_{n+1}$ subject to the constraint $\mathbf{q}_{n+1}^T \hat{\boldsymbol{\theta}}_{n+1} = 1$, where $\mathbf{q}_{n+1}^T = [\underbrace{0 \ 0 \ \dots \ 0}_n \ 1]$. Minimising the Lagrange function:

$$\mathcal{L}(\hat{\boldsymbol{\theta}}_{n+1}, \mu) = \frac{1}{2} \boldsymbol{\epsilon}_N^T \boldsymbol{\epsilon}_N + \mu (\mathbf{q}_{n+1}^T \hat{\boldsymbol{\theta}}_{n+1} - 1),$$

via $\nabla_{\hat{\boldsymbol{\theta}}_{n+1}} \mathcal{L} = \mathbf{0}$ gives:

$$(\boldsymbol{\Phi}_N^T \boldsymbol{\Phi}_N) \hat{\boldsymbol{\theta}}_{n+1} + \mu \mathbf{q}_{n+1} = \mathbf{0}.$$

This stationarity condition can be written as:

$$\mathbf{S}_{n+1} \hat{\boldsymbol{\theta}}_{n+1} = \begin{bmatrix} \mathbf{0} \\ -\mu \end{bmatrix},$$

where $\mathbf{S}_{n+1} = \boldsymbol{\Phi}_N^T \boldsymbol{\Phi}_N$ is the AIM. Partitioning the AIM shows its *augmented* nature:

$$\begin{bmatrix} \mathbf{S}_n & -\mathbf{X}_N^T \mathbf{y}_N \\ -\mathbf{y}_N^T \mathbf{X}_N & \mathbf{y}_N^T \mathbf{y}_N \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\theta}}_n \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mu \end{bmatrix}, \quad (2.1)$$

where $\mathbf{S}_n = \mathbf{X}_N^T \mathbf{X}_N$. Note that the first n equations of (2.1) give the standard LS solution for an n -parameter relation and that the last equation gives the value of the Lagrange multiplier:

$$-\mu = -\mathbf{y}_N^T \mathbf{X}_N \hat{\boldsymbol{\theta}}_n + \mathbf{y}_N^T \mathbf{y}_N = \mathbf{y}_N^T \boldsymbol{\epsilon}_N = (\boldsymbol{\theta}_n^T \mathbf{X}_N^T + \mathbf{e}_N^T) \boldsymbol{\epsilon}_N$$

and, since the residuals are orthogonal to the data (see eqn.(1.15)), $\mathbf{X}_N^T \boldsymbol{\epsilon}_N = \mathbf{0}$ and thus

$$-\mu = \mathbf{e}_N^T \boldsymbol{\epsilon}_N = \mathbf{e}_N^T [\mathbf{I} - \mathbf{X}_N \mathbf{S}_n^{-1} \mathbf{X}_N^T] \boldsymbol{\epsilon}_N = \mathbf{e}_N^T \mathbf{B} \boldsymbol{\epsilon}_N$$

where the matrix $B = [I - X_N S_n^{-1} X_N^T]$ is idempotent, i.e. $B = B^T = B^2$. Hence

$$-\mu = e_N^T B^2 e_N = \epsilon_N^T \epsilon_N = J_n$$

which is the LS loss function for the n -parameter relation.

Using the block matrix inversion lemma¹, we see that eqn.(2.1) can be re-written as:

$$\begin{aligned} \begin{bmatrix} \hat{\theta}_n \\ 1 \end{bmatrix} &= \begin{bmatrix} S_n^{-1} + \hat{\theta}_n \hat{\theta}_n^T / J_n & \hat{\theta}_n / J_n \\ \hat{\theta}_n^T / J_n & 1 / J_n \end{bmatrix} \begin{bmatrix} 0 \\ J_n \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} I & \hat{\theta}_n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S_n^{-1} & 0 \\ 0 & 1 / J_n \end{bmatrix} \begin{bmatrix} I & \hat{\theta}_n \\ 0 & 1 \end{bmatrix}^T}_{S_{n+1}^{-1} = U D U^T} \begin{bmatrix} 0 \\ J_n \end{bmatrix} \end{aligned} \quad (2.2)$$

where the block UDU^T factorisation of S_{n+1}^{-1} (highlighted by the underbrace) contains *both* the LS parameter estimates in the last column of U *and* the (inverse) LS loss function as the last entry of D . Thus by defining the ACM as $P_{n+1} = S_{n+1}^{-1}$ we see from eqn.(2.2) that P_{n+1} possesses a *nested structure* since P_n appears in P_{n+1} :

$$P_{n+1} = \begin{bmatrix} I & \hat{\theta}_n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_n & 0 \\ 0 & \frac{1}{J_n} \end{bmatrix} \begin{bmatrix} I & \hat{\theta}_n \\ 0 & 1 \end{bmatrix}^T, \quad (2.3)$$

and by recursively decomposing all the nested ACM's we get the following theorem:

Theorem 1 (The MMLS method) *By decomposing the augmented covariance matrix P_{n+1} into UDU^T form, i.e.*

$$P_{n+1} = U D U^T, \quad (2.4)$$

¹Block matrix inversion lemma:

$$\begin{bmatrix} A & v \\ v^T & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1}(I + v v^T A^{-1} / \beta) & -A^{-1} v / \beta \\ -v^T A^{-1} / \beta & 1 / \beta \end{bmatrix}, \quad \text{where } \beta = \alpha - v^T A^{-1} v.$$

the upper triangular $\mathcal{U}$ contains the estimated multiple model parameters and the diagonal matrix $\mathcal{D}$ contains the inverses of the associated LS loss function values, i.e.

$$\mathcal{U} = \begin{bmatrix} 1 & \hat{\theta}_{11} & \dots & \hat{\theta}_{1n} \\ & 1 & & \vdots \\ & & \ddots & \hat{\theta}_{nn} \\ & & & 1 \end{bmatrix} \quad \text{and} \quad \mathcal{D} = \begin{bmatrix} \frac{1}{J_0} & & & \\ & \frac{1}{J_1} & & \\ & & \ddots & \\ & & & \frac{1}{J_n} \end{bmatrix}$$

where $\hat{\theta}_i$ correspond to the i -parameter relation:

$$0 = \hat{\theta}_{1i}x_1 + \hat{\theta}_{2i}x_2 + \dots + \hat{\theta}_{ii}x_i + x_{i+1} + \epsilon$$

for $i = 1, 2, \dots, n$.

Proof 1 (The MMLS method) Given the parameters and the losses of LS models up to the $(k-1)^{th}$ order according to eqn.(2.4), i.e.

$$\mathbf{P}_k = \begin{bmatrix} 1 & \hat{\theta}_{1,1} & \dots & \hat{\theta}_{1,k-1} \\ & 1 & & \vdots \\ & & \ddots & \hat{\theta}_{k-1,k-1} \\ & & & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{J_0} & & & \\ & \frac{1}{J_1} & & \\ & & \ddots & \\ & & & \frac{1}{J_k} \end{bmatrix} \begin{bmatrix} 1 & & & \\ \hat{\theta}_{1,1} & 1 & & \\ \vdots & & \ddots & \\ \hat{\theta}_{1,k-1} & \dots & \hat{\theta}_{k-1,k-1} & 1 \end{bmatrix}.$$

This can be rewritten as:

$$\mathbf{P}_k = 1/J_0 + \sum_{i=1}^{k-1} \left\{ \begin{bmatrix} \hat{\theta}_i \\ 1 \end{bmatrix} \begin{bmatrix} \hat{\theta}_i \\ 1 \end{bmatrix}^T / J_i \right\}. \quad (2.5)$$

Now employing the nested structure of eqn.(2.3):

$$\begin{aligned} \mathbf{P}_{k+1} &= \begin{bmatrix} \mathbf{I} & \hat{\theta}_k \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_k & \mathbf{0} \\ \mathbf{0} & \frac{1}{J_k} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \hat{\theta}_k \\ \mathbf{0} & 1 \end{bmatrix}^T \\ &= \mathbf{P}_k + \begin{bmatrix} \hat{\theta}_k \\ 1 \end{bmatrix} \begin{bmatrix} \hat{\theta}_k \\ 1 \end{bmatrix}^T / J_k \end{aligned}$$

and substituting for $\mathbf{P}_k$ using eqn.(2.5) we get that

$$\mathbf{P}_{k+1} = 1/J_0 + \sum_{i=1}^k \left\{ \begin{bmatrix} \hat{\theta}_i \\ 1 \end{bmatrix} \begin{bmatrix} \hat{\theta}_i \\ 1 \end{bmatrix}^T / J_i \right\}$$

which gives the parameters and losses of all the models up to the next order (k) as required.

Clearly both eqn.(2.4) and eqn.(2.3) give the same values for the 0^{th} order model, i.e.

$$\mathbf{P}_1 = 1/J_0$$

and thus, by the principal of induction, the proof is complete. ■

2.1.1 The batch algorithm

Since the AIM is symmetric and positive semi-definite it can be factorised into $\mathbf{S}_{n+1} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ form. Taking the inverse, we get:

$$\mathbf{S}_{n+1}^{-1} = (\mathbf{L}^{-1})^T \mathbf{D}^{-1} \mathbf{L}^{-1} = \mathbf{U}\mathbf{D}\mathbf{U}^T$$

which is exactly what we require. Hence the MMLS algorithm can be implemented using a $\mathbf{L}\mathbf{D}\mathbf{L}^T$ factorisation as in Table 2.1.

Table 2.1: A batch implementation of MMLS

Decompose $\mathbf{S}_{n+1}$	$\longrightarrow$	$\mathbf{L}\mathbf{D}\mathbf{L}^T$
then,		
Estimates $\hat{\boldsymbol{\theta}}_i$	$\longrightarrow$	$(i+1)^{\text{th}}$ row of $\mathbf{L}^{-1}$
Loss function J_i	$\longrightarrow$	$(i+1)^{\text{th}}$ entry of $\mathbf{D}$

Remark 2 This is only appropriate for off-line applications because of the need to find the inverse of the matrix $\mathbf{L}$.

Remark 3 Other decomposition techniques such as Cholesky, LU and QR can be used to obtain the upper triangular parameter matrix $\mathbf{U}$ and the diagonal loss function matrix $\mathbf{D}$. The conversion procedures required for such decompositions can be found in (Niu et al., 1995b).

Remark 4 Since the MMLS technique is essentially order recursive LS all the properties of classical LS apply.

2.1.2 The on-line algorithm

As new measurements are made, the AIM can be updated *recursively* using:

$$\mathbf{S}_{n+1}(t) = \beta \mathbf{S}_{n+1}(t-1) + \phi(t)\phi^T(t),$$

where β is the exponential forgetting-factor (Clarke, 1981a). The inverse of $\mathbf{S}_{n+1}$ can be computed recursively using the UD filter of Bierman (1976) which updates the $\mathbf{U}$ and $\mathbf{D}$ factors of $\mathbf{S}_{n+1}^{-1}$ directly for improved numerical performance. Thus the Multiple Model Recursive Least-Squares (MMRLS) algorithm is as shown in Table 2.2.

Table 2.2: The MMRLS algorithm

$\mathbf{f} = \bar{\mathbf{U}}^T \phi$	Prediction errors (negated)
$\mathbf{v} = \bar{\mathbf{D}} \mathbf{f}$	
$s_0 = \beta$	
for $j = 1$ to $n + 1$	From 0 to n parameter model
$s_j = s_{j-1} + \mathbf{f}_j \mathbf{v}_j$	
$\mathbf{g}^{(j-1)} = [v_1 \ v_2 \ \dots \ v_{j-1} \ \underbrace{0 \ \dots \ 0}_{n-j+2}]^T$	(first $j - 1$ elements of $\mathbf{v}$)
$\mathbf{k}_j = \bar{\mathbf{U}} \mathbf{g}^{(j-1)} / s_{j-1}$	Kalman gain
$\hat{\mathbf{u}}^{(j)} = \bar{\mathbf{u}}^{(j)} - \mathbf{k}_j \mathbf{f}_j$	Parameter update
$\hat{\mathbf{d}}^{(j)} = \bar{\mathbf{d}}^{(j)} (s_{j-1} / s_j) / \beta$	Loss function update
end	

where matrices $\bar{\mathbf{U}} = \mathbf{U}(t-1)$, $\hat{\mathbf{U}} = \mathbf{U}(t)$, $\bar{\mathbf{D}} = \mathbf{D}(t-1)$, $\hat{\mathbf{D}} = \mathbf{D}(t)$, and where the vectors $\bar{\mathbf{u}}^{(j)}$, $\hat{\mathbf{u}}^{(j)}$, $\bar{\mathbf{d}}^{(j)}$ and $\hat{\mathbf{d}}^{(j)}$ represent the j^{th} column of each matrix respectively.

Remark 5 The algorithm in Table 2.2 is in fact identical to Bierman's original algorithm, and is presented here in a form which highlights the order recursive nature of MMLS (compared with the fixed order RLS equations 1.16–1.18).

2.2 Rapprochement with existing order recursive algorithms

In this section the MMLS technique is compared to the LSL method and the LDA for AR processes.

2.2.1 Least-Squares lattice algorithms

Theorem 7.6.2 of Goodwin and Sin (1984) gives the properties of the lattice filter for a scalar full-rank wide-sense stationary random AR process $\{y(t)\}$. Here we are interested in the orthogonality properties of the forward and backward residuals as depicted in Figure 2.1. It is also worth observing that the forward and reverse reflection coefficients K_i^f and K_i^b , and the transformation matrices L_f and L_b , are equal under such conditions.

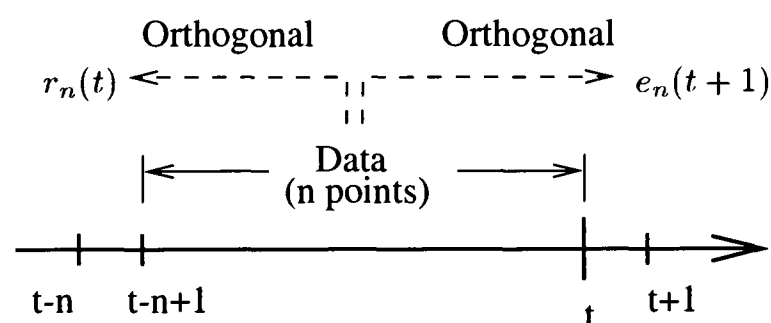


Figure 2.1: Diagram of the forward and backward predictions

The significance of the forward prediction errors e_i being order independent is that the transformed information matrix (see Appendix A),

$$\tilde{S}(t) = \sum_{k=1}^t \tilde{\mathbf{x}}(k) \tilde{\mathbf{x}}^T(k) = \begin{pmatrix} \sum e_0^2(t-n) & & & \\ & \sum e_1^2(t-n+1) & & \\ & & \ddots & \\ & & & \sum e_n^2(t) \end{pmatrix} = \mathbf{D}(t),$$

is diagonal and contains the sum of squared errors for each order model. Using eqn.(A.1) we get that:

$$\mathbf{S}(t) = \mathbf{L}_f^{-1}(t) \mathbf{D}(t) \mathbf{L}_f^{-T}(t),$$

where $\mathbf{L}_f(t)$ is a lower triangular transformation matrix containing the parameter estimates defined by eqn.(1.24). In terms of the inverse we get that

$$\mathbf{P}(t) = \mathbf{S}^{-1}(t) = \mathbf{L}_f^T(t) \mathbf{D}^{-1}(t) \mathbf{L}_f(t) = \mathbf{U}(t) \mathbf{D}(t) \mathbf{U}^T(t) \quad (2.6)$$

which is exactly the MMLS algorithm for AR model structures. Thus the LSL reflection coefficients for AR signal modelling can be obtained directly from MMLS by using eqn.(1.25).

2.2.2 Levinson type algorithms

In order to see the relationship between the LDA and MMLS, rewrite the original set of equations (1.28) as:

$$\begin{pmatrix} \mathbf{R}_k & \mathbf{r}_k^R \\ \mathbf{r}_k^{R^T} & r_0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\theta}_k^R \\ 1 \end{pmatrix} = \mathbf{R}_{k+1} \begin{pmatrix} \boldsymbol{\theta}_k^R \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \sigma_k^2 \end{pmatrix} \quad k = 1, \dots, n.$$

These can be expressed as:

$$\underbrace{\begin{pmatrix} 1 & & & \\ a_{11} & 1 & & \\ \vdots & & \ddots & \\ a_{nn} & \cdots & a_{n1} & 1 \end{pmatrix}}_{\mathbf{U}^T} \underbrace{\begin{pmatrix} r_0 & r_1 & \cdots & r_n \\ r_1 & r_0 & & r_{n-1} \\ \vdots & & \ddots & \\ r_n & r_{n-1} & & r_0 \end{pmatrix}}_{\mathbf{R}_{n+1}} \underbrace{\begin{pmatrix} 1 & a_{11} & \cdots & a_{nn} \\ & 1 & & \vdots \\ & & \ddots & a_{n1} \\ & & & 1 \end{pmatrix}}_{\mathbf{U}} = \underbrace{\begin{pmatrix} r_0 & & & \\ & \sigma_1^2 & & \\ & & \ddots & \\ & & & \sigma_n^2 \end{pmatrix}}_{\mathbf{D}_{n+1}}$$

due to the fact that all the lower order $\{\mathbf{R}_i\}$ matrices are contained in $\mathbf{R}_{n+1}$ (as before). Clearly the upper triangular matrix $\mathbf{U}$ contains the same parameters as $\mathbf{U}_{n+1}$, with the difference that the parameters for each order model are contained in the columns of $\mathbf{U}$ as opposed to the rows of $\mathbf{U}_{n+1}$. Thus we have that

$$\mathbf{R}_{n+1} = \mathbf{U}^{-T} \mathbf{D}_{n+1} \mathbf{U}^{-1}$$

or, in terms of the inverse:

$$\mathbf{R}_{n+1}^{-1} = \mathbf{U} \mathbf{D}_{n+1}^{-1} \mathbf{U}^T,$$

and, since

$$\mathbf{R}_{n+1} = \lim_{t \rightarrow \infty} \mathbf{S}(t)/t \quad \text{i.e.} \quad \mathbf{R}_{n+1}(t) \approx \mathbf{S}(t)/t \quad (\text{for large } t),$$

where $\mathbf{S}(t)$ is defined in eqn.(A.1) for $\mathbf{x}(t) = (y(t-n) \cdots y(t))^T$, we get that

$$\mathbf{S}^{-1}(t) = \mathbf{U} \left(\mathbf{D}_{n+1}^{-1}/t \right) \mathbf{U}^T = \mathbf{U} \mathbf{D} \mathbf{U}^T. \quad (2.7)$$

Hence by looking at the original equations from a reversed parameter vector stand point, and approximating the correlation matrix $\mathbf{R}_{n+1}$ by a finite-time average, we get the parameters of all the models $\boldsymbol{\theta}_k^R$ in the columns of $\mathbf{U}$ and their inverse loss functions $1/(t\sigma_k^2)$ in the diagonal matrix $\mathbf{D}$ by decomposing $\mathbf{S}^{-1}(t)$ into $\mathbf{UDU}^T$ form – which is the MMLS algorithm for AR modelling.

Notice that this is very similar to the direct LDU decomposition method described earlier in section 1.4.2, the difference being that MMLS requires a reversed parameter vector $\boldsymbol{\theta}_k^R$ (i.e. the higher order parameters are found at the front of parameter vector) whereas the LDA gives the parameters $\boldsymbol{\theta}_k$ in increasing order. Thus for MMLS a $\mathbf{UDU}^T$ factorisation is performed on the inverse of the sample covariance matrix, $\mathbf{R}_{n+1}^{-1}$ instead of a $\mathbf{LDL}^T$ factorisation.

Also notice that eqn.(2.7) is exactly the factorisation we achieved in eqn.(2.6) by changing the basis of an AR model. By comparison one sees that the transformation matrix $\mathbf{T}(t) = \mathbf{U}^T(t)$. Thus the transformed regressors are given by:

$$\tilde{\mathbf{x}}(t) = \mathbf{U}^T(t)\mathbf{x}(t) = \begin{pmatrix} e_0(t-n) \\ e_1(t-n+1) \\ \vdots \\ e_n(t) \end{pmatrix}$$

where $e_k(t-n+k) = y(t-n+k) - \hat{y}(t-n+k)$ is the *forward prediction error* for the k^{th} order model based on past data. Hence the set $\{e_0(t-n), \dots, e_n(t)\}$ forms an orthogonal basis for $\{y(t-n), \dots, y(t)\}$ as with the LSL algorithm. Thus the reflection coefficients can be obtained as follows:

$$\tilde{\boldsymbol{\theta}}_n = \begin{pmatrix} K_0^f \\ \vdots \\ K_{n-1}^f \end{pmatrix} = \mathbf{L}_f^{-T} \boldsymbol{\theta}_n = \mathbf{U}(1:n, 1:n)^{-1} \mathbf{U}(n:-1:1, n+1)$$

where MATLAB indexing is used, i.e. $\mathbf{A}(i_1:i_2, j_1:j_2)$ denotes the submatrix of $\mathbf{A}$ specified by rows i_1 to i_2 and columns j_1 to j_2 , and $\mathbf{A}(i_1:\Delta i:i_2, j)$ refers to elements in rows i_1 to i_2

in steps of Δi of column j . Now, since the k^{th} order reflection coefficient is $\rho_k = a_{k+1,k+1}$, the reflection coefficients can be easily obtained directly from the parameter matrix:

$$\tilde{\theta}_n^T = (a_{1,1} \ \dots \ a_{n+1,n+1}) = \mathcal{U}(1, 2 : n+1),$$

i.e. the top row of the $\mathcal{U}$ matrix, starting from the second element in that row.

2.2.3 Comparison results

Speed

The algorithm efficiency can be measured by the number of floating point operations (flops) performed to obtain the multiple order parameters for a batch of 1000 samples of a 3^{rd} order AR model:

$$(1 - 2.3q^{-1} + 1.9q^{-2} - 0.56q^{-3}) y(t) = e(t)$$

where $e(t)$ is a Gaussian distributed random variable with unit variance and zero mean. Figure 2.2 shows the results obtained for (a) the batch case using the LDA, direct LDU decomposition, and MMLS methods, and (b) for the on-line case using the LSL, RLS and MMRLS algorithms. The best fit polynomials obtained for the curves above are shown in Table 2.3 where k is the order of the AR model.

Table 2.3: Best fit polynomials of algorithm speed (measured in MATLAB flops) versus AR model order

Batch method	Speed [nflops]	On-line method	Speed [nflops/update]
LDA	$3.5k^2 + 15.5k - 2$	LSL	$4k^2 + 32k + 2$
LDU	$5k^3 + 17k^2 + 21k + 9$	RLS	$2.4k^3 + 4.5k^2 + 19.5k - 13$
MMLS	$1.67k^3 + 8.5k^2 + 13.83k + 7$	MMRLS	$2k^3 + 9k^2 + 17k + 7$

Consider the batch algorithms first. Since the LDA is $O(k^2)$ and the others are $O(k^3)$, only the LDA is a “fast” algorithm. This is because the LDA makes explicit use of the Toeplitz structure of the correlation matrix. Thus the LDA is the preferred method in terms of

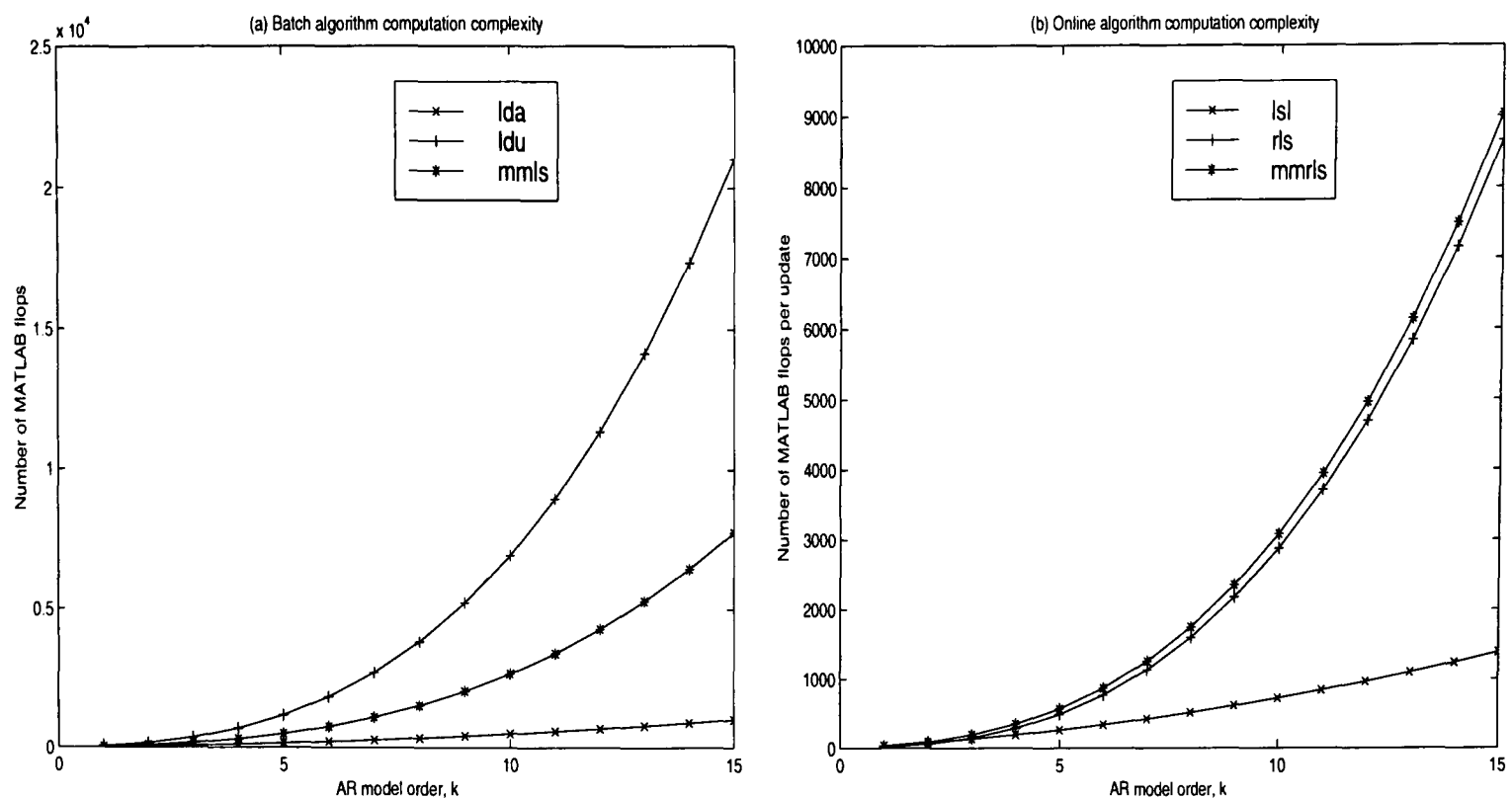


Figure 2.2: Number of flops (as measured in MATLAB) versus AR model order for (a) Batch methods and (b) On-line methods

computational speed. For on-line applications, Table 2.3 shows that LSL is also a “fast” algorithm being $O(k^2)$ per update and that there is very little difference in the RLS and MMRLS algorithms in terms of execution speed.

Numerical properties

The numerical properties of square-root factorisation based LS algorithms have been studied in detail by Graupe et al. (1980). Since MMLS is based on a U-D factorisation, which is a type of square-root factorisation, the results given in (Graupe et al., 1980) can be used to infer the properties of MMLS versus Levinson type algorithms for AR models. Indeed Graupe et al. (1980) show that square-root factorisation based algorithms (such as MMLS) are numerically superior to Levinson and direct LS methods in terms of *convergence* (i.e. using restricted word length) and *stability* (when using large data sets). In addition Ljung and Ljung (1985) show that the conventional RLS based algorithms and the LSL based algorithms are both exponentially stable to round-off errors and other numerical disturbances, of which the U-D factorisation algorithms are numerically favourable due to better conditioning of the covariance matrix. These results together with those of Verhaegen

Table 2.4: Summary of algorithm properties for AR modelling

	Batch		On-line	
	MMLS	LDA	MMRLS	LSL
Speed	$O(k^3)$	$O(k^2)$	$O(k^3)$	$O(k^2)$
Convergence	Good	Poor	Good	Good
Conditioning & Stability	Excellent	Poor	Excellent	Good

(1989) and Graupe et al. (1980) are summarised in Table 2.4. Thus the choice of method depends on whether computational speed is of prime importance or not: if speed is critical then use the LDA for batch cases and LSL algorithms for on-line applications. However these “fast” methods have a penalty in terms of numerical performance, an area which the U-D factorisation based MMLS and MMRLS methods are particularly strong. It should also be noted that MMRLS has an added advantage over LSL algorithms in that the user is presented with the estimates directly (in contrast to LSL algorithms which give the reflection coefficients).

2.3 Reliable estimator jacketing

The results of section 2.2.3 indicate that MMRLS is the preferred method for on-line estimation of multiple order models. This being the case, the user must also consider a number of practical aspects to achieve a reliable estimation algorithm, and the following conditions for exponential convergence of an RLS based estimator (see Parkum et al., 1992) proved helpful in developing a robust on-line multiple order model estimation algorithm:

1. No measurement noise: $e(t) = 0$ for all t .
2. Bounded input data: $\|\phi(t)\| < M$ for all t .
3. Bounded (augmented) covariance matrix: $\lambda_{\min} \mathbf{I} \leq \mathbf{P}_{n+1}(t) \leq \lambda_{\max} \mathbf{I}$ for all t .
4. Persistently exciting data: $\sum_{k=t+1}^{t+N} \phi(k)\phi^T(k) \geq \rho \mathbf{I}$ for all t and some $N \geq \dim \phi$.

The resulting estimator “jacketing” techniques employed in the final MMRLS estimator are now addressed.

2.3.1 Data pre-filtering

Obviously the first condition on page 39 cannot be achieved in practice. However, if we assume some knowledge about the noise characteristics then, by using a data pre-filter (or “whitening” filter) the signal to noise ratio can be significantly improved over the frequencies of interest, thereby yielding better estimates as we shall now discuss.

Assume that the plant behaviour can be described accurately by an ARX model:

$$A_0(q^{-1})y(t) = B_0(q^{-1})u(t). \quad (2.8)$$

This is very seldom true in practice because there are always disturbances acting on real systems. A more realistic model should take these unwanted effects into account.

If we simply append a term $x_d(t)$ to represent these unwanted disturbance effects we get an Equation Error (EE) model:

$$A_0(q^{-1})y(t) = B_0(q^{-1})u(t) + x_d(t) \quad (2.9)$$

and, if we then assume that $x_d(t)$ can be represented accurately by filtering white noise, we get the ARMAX model (Ljung, 1987):

$$A(q^{-1})y(t) = B(q^{-1})u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t).$$

Remarks

1. Since the polynomial C is very difficult to estimate in practice (Mohtadi, 1987), we usually approximate it as T which is then used as a design parameter. Guidelines on how to choose it will be discussed later.
2. Similarly D is also difficult to estimate on-line and we usually assume it to be 1 or Δ , depending on whether the data has steady state offsets or not.
3. If we assume $C = T$ and $D = \Delta$, we get the CARIMA model of GPC (Clarke et al., 1987) which has been used successfully in a wide range of applications (Clarke, 1994).

Thus assuming $C(q^{-1}) = T(q^{-1})$, and multiplying through by D and dividing by T the ARMAX model can be written

$$A(q^{-1})Dy^f(t) = B(q^{-1})Du^f(t) + e(t) \quad \text{where } D\{\cdot\}^f \equiv \frac{D(q^{-1})}{T(q^{-1})}\{\cdot\} \quad (2.10)$$

which is now in the form of the simple linear regression model. Thus, given D and T , we can estimate A and B in the presence of equation error disturbances using the LS method.

On the other hand, rather than appending $x_d(t)$ as an equation error, a better reflection of reality might be to treat $x_d(t)$ as an output error, thus giving an Output Error (OE) model:

$$y(t) = \frac{B_0(q^{-1})}{A_0(q^{-1})}u(t) + x_d(t). \quad (2.11)$$

Assuming again that the disturbance can be generated from a white noise source we get a BJ model (Ljung, 1987):

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t)$$

which can easily be re-arranged into a linear regression form:

$$A(q^{-1})Dy^f(t) = B(q^{-1})Du^f(t) + e(t) \quad \text{where } D\{\cdot\}^f \equiv \frac{D(q^{-1})}{AT(q^{-1})}\{\cdot\}, \quad (2.12)$$

and thus by appropriate choices of T and D , we can estimate A and B in the presence of output disturbances $x_d(t)$ using LS as before.

For the sake of analysis, suppose we have an estimated model $\hat{B}/\hat{A}$ which attempts to describe the true plant B_0/A_0 over the critical frequency range. The equation error model prediction error is then:

$$e^*(t) = \frac{D}{T} \left[(B_0 - \hat{B})u(t) - (A_0 - \hat{A})y(t) + x_d(t) \right] \quad (2.13)$$

We aim to reduce the effects of the disturbance term $x_d(t)$ without loss of estimator performance. Since the prediction error drives the LS estimator, we use D/T to emphasise over which frequencies we wish to have a good model. Thus all disturbances which have frequency components outside the pass-band of the D/T filter are attenuated. Hopefully the disturbances will not be dominant in the pass-band of the D/T filter. If the disturbance dominates in the pass-band of D/T , we can either alter the frequency range of interest (by

tuning D or T) or switch off the estimator at times when this occurs. Typically though, we have DC disturbance (due to offsets) and high frequency disturbances (noise) and hence D/T is usually a band-pass filter.

Now consider the prediction error for an output error model:

$$e^*(t) = \frac{\hat{A}D}{T} \left[\left(\frac{B_0}{A_0} - \frac{\hat{B}}{\hat{A}} \right) u(t) + x_d(t) \right] \quad (2.14)$$

which is similar to eqn.(2.13) except that now we use $\hat{A}D/T$ to define the frequency range of interest. Thus the T-filter for an OE model of a process must include an extra $\hat{A}$ factor to achieve the same frequency of interest for estimation of an EE model of the process.

In summary LS can be used to estimate EE and OE models using the ARX model structure:

$$\hat{A}(q^{-1})Dy^f(t) = \hat{B}(q^{-1})Du^f(t) + e(t) \quad \text{where } D\{\cdot\}^f \equiv \frac{D(q^{-1})}{T_e(q^{-1})}.$$

Guidelines for the choice of the data pre-filter D/T_e are given below:

1. $D(q^{-1}) = \Delta = (1 - q^{-1})$ if there is a DC offset else $D(q^{-1}) = 1$, and
2. the estimation filter

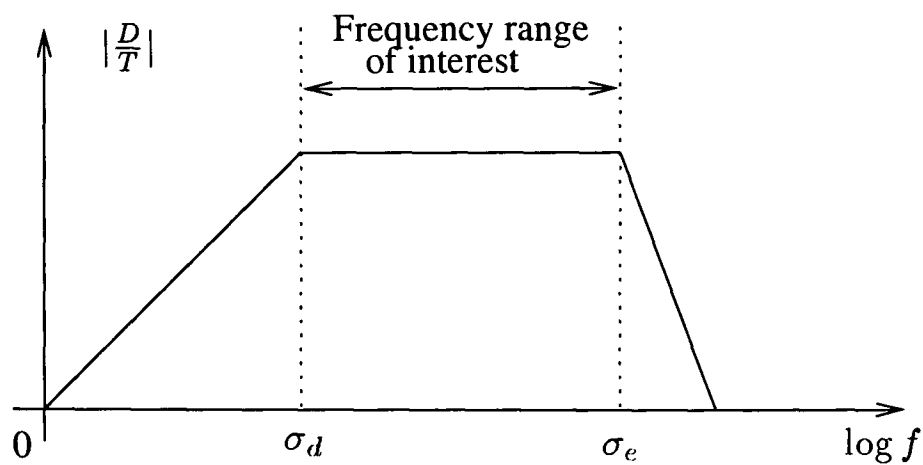
$$T_e(q^{-1}) = \begin{cases} T(q^{-1}) & \text{for equation error models, and} \\ \hat{A}T(q^{-1}) & \text{for output error models,} \end{cases}$$

with

$$T(q^{-1}) = \begin{cases} (1 - \sigma_e q^{-1})^\nu & \text{for } D = 1 \text{ and,} \\ (1 - \sigma_d q^{-1})(1 - \sigma_e q^{-1})^\nu & \text{for } D = \Delta \end{cases}$$

where the dominant pole is assumed to lie on a segment of the real axis inside a disc of radius σ_d in the z -plane, and all the unmodelled poles are assumed to lie on a segment of the real axis inside a disc of radius σ_e in the z -plane. The value of ν determines the high-frequency gain roll-off rate. Figure 2.3 shows the frequency response of the bandpass filter D/T for the case where $D = \Delta$ and $\nu = 3$.

Thus, by assuming some properties of the plant disturbance, the T_e filter can be chosen to only estimate the dynamics of the plant in the frequency range of interest using a simple linear regression and to reject the effects of the assumed disturbance.

Figure 2.3: Desired frequency response of the term D/T

2.3.2 Data normalisation

Data normalisation was first proposed by Egardt (1979) and is required to obtain $\|\phi(t)\|$ -independent properties of the estimation algorithm and to ensure boundedness of the prediction error in the presence of plant-model mismatch which is a key requirement for proving stability of adaptive systems (Ortega et al., 1985; Cluett et al., 1988; Yoon, 1994).

Consider the ‘tuned’ output prediction error given in eqn.(2.14) where it has been assumed again that $\hat{B}/\hat{A}$ is an estimate of the actual plant transfer function B_0/A_0 . Since $e^*(t)$ includes the modelling error term $(B_0/A_0 - \hat{B}/\hat{A})u(t)$, it is not guaranteed to be bounded, making convergence analysis impossible. However, there exists an over-bounding function $m(t)$ such that:

$$|e^*(t)| \leq \epsilon m(t) + |x_d(t)| \quad (2.15)$$

where ϵ is a finite constant representing the size of the unmodelled dynamics. Assuming no disturbances ($x_d(t) = 0$), a candidate overbounding function of the modelling error is:

$$m(t) = \max(1, \|\phi(t)\|^f),$$

where the filtered data vector size is given by

$$\|\phi(t)\|^f = \frac{1 - \sigma_e}{(1 - \sigma_e q^{-1})} \|\phi(t)\|$$

and it is assumed that the poles of the process lie on a segment of the real axis outside a disc with radius σ_e in the z -plane. This guarantees that the magnitude of the elements of $\phi_n(t) = \phi(t)/m(t)$ are ≤ 1 thus bounding $|e^*(t)|$ and satisfying condition 2 on page 39.

2.3.3 Covariance matrix anti-windup mechanism

An exponential forgetting-factor β is used to provide parameter tracking ability where past data is weighted using an exponentially shaped windowing function. By doing so, the ACM eigenvalues are effectively bounded from below. Intuitively, the behaviour of the forgetting mechanism should be such that:

$$\begin{aligned}\epsilon(t) \text{ small} &\longrightarrow \text{slow update (hardly any forgetting, } \beta \approx 1); \\ \epsilon(t) \text{ large} &\longrightarrow \text{fast update (forget almost everything, } \beta \rightarrow 0),\end{aligned}$$

which can be achieved using the variable forgetting strategy of Fortescue et al. (1981):

$$\beta(t) = J_{opt} / [J_{opt} + \epsilon_{n+1}^2(t)]$$

where $\epsilon_{n+1}(t)$ is the *a posteriori* prediction error for the highest order (n -parameter) model, J_{opt} is the optimal LS loss function $J_{opt} = N_0 \sigma_0^2$, and N_0 is the asymptotic memory length given perfect estimation and a Gaussian white measurement noise sequence $\{e(t)\}$ with variance σ_0^2 .

To prevent the covariance wind-up involved in forgetting past data, matrix regularisation is employed where the $\mathcal{D}$ entries are clipped to a maximum of $\mathcal{D}_{max}$ which guarantees that the ACM eigenvalues are bounded from above (see Ljung and Söderström, 1983, chapter 6.5 for details). Thus condition 3 on page 39 has been satisfied.

2.3.4 Estimation on/off criterion

The final condition on page 39 requires the user to test that the (augmented) information matrix $\mathbf{S}$ is of full rank which involves a computation which is too cumbersome in practice, and measures based on the condition number of the (augmented) covariance matrix $\mathbf{P}$ may be used instead as in the Improved Least Squares algorithm of Sripada and Fisher (1987).

Obtaining an input sequence that excites all the relevant modes of the process is usually difficult to guarantee on-line; in most adaptive systems there are often long periods of insufficient excitation where the estimated parameters tend to drift due to disturbances.

Adding external excitation is one solution to this problem, but is not always practical. An alternative approach is to stop estimation when the data is not *persistently exciting* (see e.g. Åström and Wittenmark, 1989).

Now, since the loss function update for the i -parameter model in the MMRLS algorithm is readily available: $J_i(t) = J_i(t-1)(s_{i+1}/s_i)$, the value of $s_{i+1} - s_i = f_{i+1}v_{i+1}$ determines the increase in the loss. Ideally, the estimator should be turned off when the loss function is no longer increasing (perfect modelling), and turned on if the loss function starts to grow. Thus the sum $\sum_i f_i v_i = \phi^T(t) \mathbf{P}_{n+1}(t-1) \phi(t)$ provides an excellent estimation on/off criterion:

if $\mathbf{f}^T \mathbf{v} < \delta$ then no update, i.e. $\mathbf{U}(t) = \mathbf{U}(t-1)$ and $\mathbf{D}(t) = \mathbf{D}(t-1)$.
 else perform the $\mathbf{U}$ and $\mathbf{D}$ update of Table 2.2.

where δ is a user-chosen value specifying the minimum increase in loss function per sample, below which the estimator is turned off.

2.3.5 Robustness to outliers

Up to now it is assumed that each observation, y_i , has a measurement error that is independently random and distributed as a normal (Gaussian) distribution around the “true” model $y(u)$ where u is the input sequence. The standard deviations, σ , of these normal distributions have been assumed to be the same for all points. Thus the probability of the data set occurring within some Δy on each point is the product of the probabilities of each point (Press et al., 1992),

$$P \propto \prod_{i=1}^N \left\{ e^{-\frac{1}{2} \left(\frac{y_i - y(u)}{\sigma} \right)^2} \Delta y \right\}. \quad (2.16)$$

Maximising eqn.(2.16) is equivalent to minimising the negative of its logarithm, namely

$$\left[\sum_{i=1}^N \frac{(y_i - y(u))^2}{\sigma^2} \right] - N \log \Delta y.$$

Since N , σ and Δy are all constants, minimising this equation is equivalent to minimising

$$\sum_{i=1}^N \left(y_i - y(u) \right)^2,$$

which is the familiar LS cost function. Hence LS is a *maximum likelihood* (ML) estimation of the fitted parameters *if* the measurement errors are independent and normally distributed with constant deviation.

If the measurement errors are not normally distributed, for example if there are a number of *outliers* in the data set, then instead of eqn.(2.16) we should write

$$P = \prod_{i=1}^N \left\{ e^{-\rho(y_i, y\{u_i; \hat{\theta}\})} \Delta y \right\}$$

where the function ρ is the negative logarithm of the probability density and Δy is the bound associated with each point. Taking the logarithm as before results in minimising the expression:

$$\sum_{i=1}^N \rho(y_i, y\{u_i; \hat{\theta}\}).$$

Usually the function ρ depends not on each argument (measured y_i and predicted output $y(x_i)$) independently, but only on their difference, at least if scaled by some weighting factors σ_i which we are able to assign to each data point. In this case the M -estimate is said to be *local*. Thus the estimation problem is now:

$$\text{minimise over } \hat{\theta} \quad \sum_{i=1}^N \rho \left(\frac{y_i - y(u_i; \hat{\theta})}{\sigma_i} \right), \quad (2.17)$$

where the function $\rho(z)$ is a function of a single variable $z \equiv [y_i - y(u_i; \hat{\theta})]/\sigma_i = \epsilon_i/\sigma_i$. If we define the derivative of $\rho(z)$ to be the function $\psi(z)$,

$$\psi(z) = \frac{d\rho(z)}{dz}$$

then the estimates can be found by taking the derivative of eqn.(2.17) and setting it to zero:

$$0 = \sum_{i=1}^N \frac{1}{\sigma_i} \psi(z) \left(\frac{\partial y(u_i; \hat{\theta})}{\partial \hat{\theta}_k} \right).$$

For the specialisation of *normally* distributed errors as in eqn.(2.16) we get that

$$\rho(z) = \frac{1}{2}z^2 \quad \psi(z) = z \quad (\text{normal}),$$

where the deviations are minimised using the familiar 2-norm. From this one can see clearly that outliers have a large affect on the cost which is obviously very undesirable. The next two distributions try to take into account these outliers.

If the errors are distributed as a *Laplacian* or *two-sided exponential*,

$$P\{y_i - y(u_i)\} = P(\epsilon_i) \sim \exp\left(-\left|\frac{\epsilon_i}{\sigma_i}\right|\right)$$

then

$$\rho(z) = |z| \quad \psi(z) = \text{sgn}(z) \quad (\text{Laplacian}).$$

In this case the ML estimator is obtained by minimising the *mean absolute deviation* (1-norm) rather than the mean square deviation (2-norm). Here the ‘tails’ of the Laplacian distribution are asymptotically much larger than any corresponding Gaussian and thus the 1-norm based ML estimator is more robust to outliers than the 2-norm based LS estimator. McMichael (1988) discusses the benefits of robust estimation using different (L_p) norms where $1 \leq p \leq 2$. A distribution with even more extensive ‘tails’ is the *Cauchy* or *Lorentzian* distribution,

$$P(\epsilon_i) \sim \frac{1}{1 + \frac{1}{2} \left(\frac{\epsilon_i}{\sigma_i}\right)^2}$$

which implies

$$\rho(z) = \log\left(1 + \frac{1}{2}z^2\right) \quad \psi(z) = \frac{z}{1 + \frac{1}{2}z^2} \quad (\text{Lorentzian}).$$

Figure 2.4 shows how the estimation errors are weighted for normal (dotted line), Laplacian (solid line) and Lorentzian (dot-dashed line) distributions.

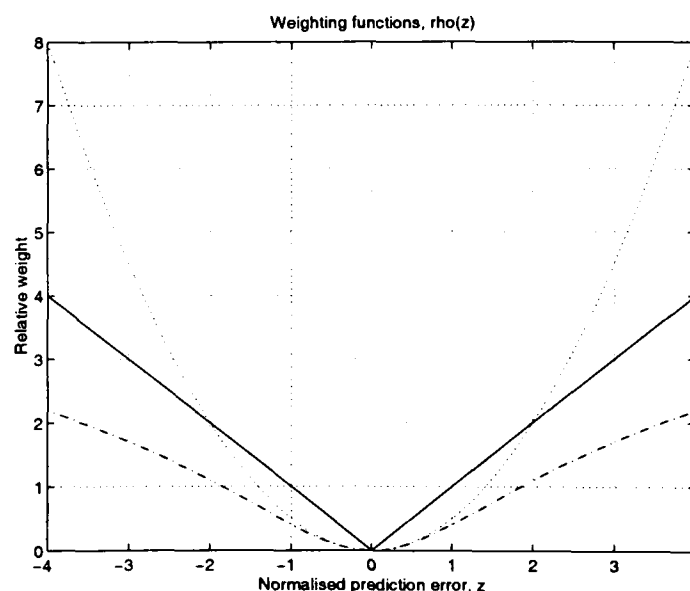


Figure 2.4: Weighting functions $\rho(z)$ (dotted line = normal, solid line = Laplacian, and dot-dashed line = Lorentzian)

The Gauss-Newton parameter update equations for the ML estimator are (Ljung and Söderström, 1983):

$$\begin{aligned}\hat{\boldsymbol{\theta}}(t) &= \hat{\boldsymbol{\theta}}(t-1) + \mathbf{P}(t)\mathbf{x}(t)\psi(z) \\ \mathbf{P}(t) &= \left(\mathbf{I} - \frac{\mathbf{P}(t-1)\mathbf{x}(t)\mathbf{x}^T(t)}{\lambda\alpha(t) + \mathbf{x}^T(t)\mathbf{P}(t-1)\mathbf{x}(t)} \right) \frac{\mathbf{P}(t-1)}{\lambda}\end{aligned}$$

where $\alpha(t) = \frac{\partial\psi(z)}{\partial z}$.

For normal distributions, $\psi_{normal}(z) = z = \epsilon(t)$ (assuming that the unknown noise variance $\sigma = 1$) and thus $\alpha = 1$; we get the standard RLS equations as expected. Now, observing that

$$\psi_{Lorentzian}(z) = \frac{\psi_{normal}(z)}{1 + 0.5\psi_{normal}^2(z)},$$

the Gauss-Newton equations can be obtained using the simple RLS update by employing the following simple modification of the prediction error:

$$\epsilon_{robust}(t) = \frac{\epsilon(t)}{1 + 0.5\epsilon^2(t)} \quad (2.18)$$

and by including the factor:

$$\alpha(t) = \frac{1 - 0.5\epsilon^2(t)}{(1 + 0.5\epsilon^2(t))^2}$$

to the denominator of the covariance update. Figure 2.5 shows the weighting of the prediction errors relative to normally distributed errors as per eqn.(2.18). Thus the Lorentzian behaves in a similar way to ordinary LS for small prediction errors, but then the weighting function ρ starts decreasing for large deviations, so that very deviant points (the true outliers) are not counted at all.

2.3.6 The data vector structure

Pairing the past input and output observations in the least squares regression vector gives rise to a *shift structure* (Ljung and Söderström, 1983) and is one of the key elements to achieving estimates of the multiple order ARX models, as in the original *AUDI* algorithm of Niu et al. (1992). Thus the augmented regression and parameter vectors for the ARX plant model:

$$A(q^{-1})y(t) = B(q^{-1})u(t - \underline{r})$$

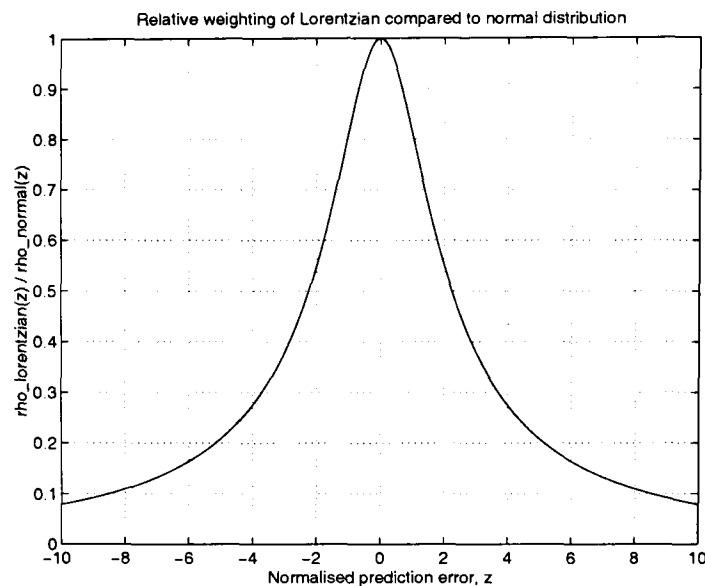


Figure 2.5: Mapping function from normally distributed prediction errors to Lorentzian

where $\underline{\tau}$ is the assumed minimum dead-time, is given by:

$$\phi = [u_{t-\underline{\tau}-n_b}, u_{t-\underline{\tau}-n_b+1}, \dots, \underline{-y_{t-n_a}}, \underline{u_{t-\underline{\tau}-n_a}}, \underline{-y_{t-n_a+1}}, \underline{u_{t-\underline{\tau}-n_a+1}}, \dots, u_{t-\underline{\tau}-1}, \underline{-y_{t-1}}, \underline{u_{t-\underline{\tau}}}, -y_t]^T$$

$$\theta = [b_{n_b}, b_{n_b-1}, \dots, b_{n_a+1}, \underline{a_{n_a}}, \underline{b_{n_a}}, \underline{a_{n_a-1}}, \underline{b_{n_a-1}}, \dots, b_2, \underline{a_1}, \underline{b_1}, 1]^T.$$

where n_a and n_b are the maximum number of plant $A(q^{-1})$ and $B(q^{-1})$ parameters to be estimated respectively. Presenting the data to the MMLS algorithm in this fashion, the parameters in the columns of the $\mathcal{U}$ matrix correspond to the following models:

$$\left[\begin{array}{l} 0 \\ u_{t-\underline{\tau}-n_b} \\ \vdots \\ u_{t-\underline{\tau}-n_b}, u_{t-\underline{\tau}-n_b+1}, \dots, u_{t-\underline{\tau}-n_a-1} \\ u_{t-\underline{\tau}-n_b}, u_{t-\underline{\tau}-n_b+1}, \dots, u_{t-\underline{\tau}-n_a-1}, -y_{t-n_a} \\ \vdots \\ u_{t-\underline{\tau}-n_b}, \dots, u_{t-\underline{\tau}-n_a-1}, -y_{t-n_a}, u_{t-\underline{\tau}-n_a}, \dots, u_{t-\underline{\tau}-1}, -y_{t-1} \\ u_{t-\underline{\tau}-n_b}, \dots, u_{t-\underline{\tau}-n_a-1}, -y_{t-n_a}, u_{t-\underline{\tau}-n_a}, \dots, u_{t-\underline{\tau}-1}, -y_{t-1}, u_{t-\underline{\tau}} \end{array} \right] \Rightarrow \left[\begin{array}{l} u_{t-\underline{\tau}-n_b} \\ u_{t-\underline{\tau}-n_b+1} \\ \vdots \\ y_{t-n_a} \\ u_{t-\underline{\tau}-n_a} \\ \vdots \\ u_{t-\underline{\tau}} \\ y_t \end{array} \right]$$

where $\Rightarrow$ indicates the result of a linear combination of the variables on the left-hand side to fit the variable on the right-hand side.

Notice that n_b is not constrained to be equal to n_a as in the case of $\mathcal{AUDI}$. Thus by adding an extra $\Delta\tau = \bar{\tau} - \underline{\tau}$ past inputs to the regression vector, plants with dead-time variations of up to $\Delta\tau$ samples can be estimated. This does require more careful interpretation of the $\mathcal{U}$

matrix columns and $\mathcal{D}$ entries since the first $n_b - n_a$ columns of $\mathcal{U}$ (and elements of $\mathcal{D}$) are not particularly useful as they give the coefficients of a (low order) input–input mapping. If $n_b - n_a < 0$ then the estimates in the first $n_a - n_b$ columns correspond to output–output mappings. The remaining columns of $\mathcal{U}$ give the multiple order ARX models, alternating between the plant models (where the result of the linear combination is an *output*) and the feedback models (where the result of the linear combination is an *input*).

Example 1 (The MMLS estimator structure) Consider the following first order ARX model of a system

$$(1 - 0.9q^{-1})y(t) = 0.1u(t - \tau)$$

where the expected dead-time $\tau = 5 \pm 1$ samples giving $\underline{\tau} = 4$ and $\Delta\tau = 2$. The MMLS structure is determined by $n_a = 1$, $n_b = 1 + \Delta\tau = 3$ and $\underline{\tau} = 4$ resulting in the following augmented regression vector:

$$\phi = [u_{t-6}, u_{t-5}, -y_{t-1}, u_{t-4}, -y_t]^T.$$

Using a 100 sample input sequence $u(t)$ of Gaussian white noise with unit variance, the final parameter matrix obtained using the Batch MMLS algorithm is:

$$\mathcal{U} = \begin{bmatrix} 1 & 0.0510 & 0.0946 & -0.0512 & 0.0000 \\ & 1 & 0.0003 & 0.0747 & 0.1000 \\ & & 1 & -0.1056 & -0.9000 \\ & & & 1 & 0.0000 \\ & & & & 1 \end{bmatrix}$$

where the columns of the matrix $\mathcal{U}$ contain the parameters for the set of linear models given by the set of equations $\mathcal{U}^T \phi = 0$. The associated loss functions are given by inverting the diagonal elements of $\mathcal{D}$ i.e. $J = \text{diag}(\mathcal{D}^{-1})$:

$$J = [100.6987, 112.9235, 3.8153, 113.3096, 0.0000]$$

from which we see that the model with coefficients in the last column of $\mathcal{U}$, i.e.

$$0u_{t-6} + 0.1u_{t-5} - 0.9(-y_{t-1}) + 0u_{t-4} + 1(-y_t) = 0.$$

has a near perfect fit since $J^{(5)} = 0.0000$, matching the actual ARX model above, as expected.

■

2.3.7 Model order selection

The problem of deciding a suitable order model for the observed data has previously been treated as a hypothesis test or from an information-theoretic point of view (Gustafsson and Hjalmarsson, 1995). Perhaps the most well-known methods are: Final Prediction Error (FPE) (Akaike, 1969), Information Criterion A (AIC) (Akaike, 1972) and Information Criterion B (BIC) (Akaike, 1977; Schwartz, 1978), Rissanen's Minimum Description Length (MDL) (Rissanen, 1978) and the Error Variance Norm (EVN) of Young et al. (1980). The underlying idea is the *parsimonious principle* which states that there should be a trade-off between model fit and model complexity. Thus all criteria have one term measuring model fit and another term penalising the complexity. Freeman (1985) shows that there is no clear winner as to which of these methods is the best, and so the simple BIC is used to select the model order:

$$\dim \hat{\theta} = \arg \min_i \left\{ \left(1 + \frac{2i \log N}{N} \right) \frac{J_i}{N} \right\}$$

where $J_i = 1/\mathcal{D}_{i+1,i+1}$ is the LS loss function of the i -parameter regression model and $N = 1/(1 - \beta)$ is the *asymptotic memory length* which indicates the window of past data most useful in providing the estimates.

In a typical configuration, $n_a = m$, the maximum ARX order to be estimated, and $n_b = n_a + \Delta\tau$. Then the ARX model parameters are contained in every second column of the $\mathcal{U}$ matrix starting from the last column (order m) and counting inwards (until column $\Delta\tau + 1$ is reached = the zero order model). In example 1, the highest order ARX model is of order 1 with the parameter estimates in column 5, and the lowest order ARX model is of order 0 with the parameter estimates in column 3. Thus the ARX model order estimate is given by:

$$\hat{k} = (\dim \hat{\theta} - \Delta\tau - 1)/2. \quad (2.19)$$

2.4 Test simulation and results

The MMRLS estimator given in Table 2.2 was tested to see if it could track the following time varying process which includes step changes in pole and zero positions, steady-state gain, dead-time and also order:

$$\frac{B(q^{-1})}{A(q^{-1})} = \begin{cases} \frac{0.5q^{-1}-0.3q^{-2}}{1-1.5q^{-1}+0.7q^{-2}} & (0 \leq t < 1000) & \text{Nominal plant} \\ \frac{0.9q^{-1}-0.7q^{-2}}{1-1.55q^{-1}+0.65q^{-2}} & (1000 \leq t < 2000) & \text{Pole/Zero change} \\ \frac{0.1q^{-1}}{1-0.9q^{-1}} & (2000 \leq t < 3000) & \text{Order decrease} \\ \frac{0.25q^{-4}-0.33q^{-5}+0.11q^{-6}}{1-2.5q^{-1}+2.33q^{-2}-0.8q^{-3}} & (3000 \leq t < 4000) & \text{3 sample delay, 3}^{rd} \text{ order} \\ \frac{0.5q^{-1}-0.3q^{-2}}{1-1.5q^{-1}+0.7q^{-2}} & (4000 \leq t < 5000) & \text{Return to nominal} \end{cases}$$

It should be noted that the “data pre-filtering” and “Lorentzian data-weighting” jacketing techniques described in section 2.3 were not used in this simulation since the data was “clean” (i.e. containing no outliers and using zero mean, normally distributed measurement errors) and where the main aim of the simulation was to demonstrate the model order tracking ability of the MMRLS algorithm with fixed and variable forgetting. In practice, however, all of the jacketing of section 2.3 should be incorporated into the final estimation routine in order to obtain reliable and meaningful parameters. The process was excited using a 1000 sample, 15-bit Pseudo Random Binary Sequence (PRBS) sequence which was repeated for each of the 5 models above. See the input signal graph in Figure 2.6. This input sequence included a long period of no excitation ($250 < t \leq 650$) followed by a period of low excitation ($650 < t \leq 900$) where the PRBS sequence had an amplitude of just 0.25 to test the estimation on/off mechanism. The measurement noise was a zero-mean white noise sequence with a variance of $\sigma_0^2 = 0.0004$.

The estimator was initialised with $\mathcal{U}_0 = \mathbf{I}$ and $\mathcal{D}_0 = \mathcal{D}_{max}\mathbf{I}$. The values $n_a = 3$ (the maximum order model to be estimated) and $n_b = n_a + \Delta\tau = 6$ (since $\tau = 1$ and the dead-time variation $\Delta\tau = 3$) were used to determine the estimator configuration. The estimator tuning parameters were: $\mathcal{D}_{max} = 10^4$, $\sigma_e = 0.3$, $N_0 = 250$, $\sigma_0^2 = 0.0004$ and $\delta = 0.05$. The value of the regularisation constant $\mathcal{D}_{max}$ was chosen as a trade-off between parameter noise sensitivity and ultimate parameter tracking performance. The value of the normalisation

filter pole σ_e was chosen assuming all the unmodelled poles lie on a segment of the real axis inside disc of radius 0.3 in the z -plane. The value of the measurement noise variance σ_0^2 was obtained from prior knowledge. The “ideal” asymptotic memory length N_0 was chosen such that the optimal loss function $J_{opt} = N_0\sigma_0^2 = 0.1$, a compromise which gave smooth estimates for constant plant models and yet fast adaption when the plant changed. The on/off limit δ was chosen to be a fraction of J_{opt} so that the estimator automatically switches off as it approaches the optimal state.

Figure 2.6 shows the input/output signals together with the estimated parameter trajectories during the pole/zero plant change at $t = 1000$. Clearly the estimator successfully tracks the change in the 2nd order ARX model parameters variations via exponential forgetting. The estimation on/off mechanism prevents estimator wind-up by switching estimation off during periods of low excitation yielding smooth parameter variations, even after periods of poor excitation. The next figure shows the results with variable forgetting. Comparing the parameter trajectories of Figure 2.6 with those in Figure 2.7 we see that the variable forgetting-factor improves the speed of convergence of the estimates with less “jitter” on the parameter estimates during periods of poor excitation. To see this quantitatively, a measure of the estimated model quality is sought. Consider the model *explanation error* which is defined as:

$$Q = \sqrt{\frac{E\hat{\epsilon}^2(t)}{Ey^2(t)}} = \sqrt{\frac{J_i}{Ey^2}}$$

i.e. the fraction of Root Mean-Square (RMS) output *not* generated by the predicted output, where J_i is the MMLS loss function for the i^{th} order ARX model and E is the expectation operator. Ideally this is zero for perfect modelling and unity at worst, and is a good measure of the quality of the estimated model. Tables 2.5 and 2.6 give the explanation errors for each order model at times just before the plant changes were made for both the fixed and variable forgetting cases. Comparing the values in Table 2.5 with those in Table 2.6 we see that variable forgetting scheme yields lower explanation errors indicating better model quality. It can be seen that without order penalisation, the highest order model (i.e. 3rd order) has (on average) the best performance (i.e. the lowest explanation error), hence showing the need for the order penalisation when selecting the model order.

Table 2.5: Model quality (fixed forgetting)

ARX order	Explanation error, Q				
	t=1000	t=2000	t=3000	t=4000	t=5000
1	0.4073	0.2736	0.3205	0.4739	0.4694
2	0.1904	0.1220	0.3202	0.2080	0.1751
3	0.1869	0.1188	0.3098	0.1149	0.1759
Best	3	3	3	3	2
True	2	2	1	3	2

Table 2.6: Model quality (variable forgetting)

ARX order	Explanation error, Q				
	t=1000	t=2000	t=3000	t=4000	t=5000
1	0.2670	0.1798	0.2081	0.4651	0.2625
2	0.0909	0.0570	0.2056	0.1462	0.0814
3	0.0842	0.0575	0.2029	0.0684	0.0795
Best	3	2	3	3	3
True	2	2	1	3	2

The estimated plant order trajectories are shown in Figures 2.8 and 2.9. Clearly the order selection criterion of eqn.(2.19) manages to track changes in plant order in both the fixed and variable forgetting cases. The performance of the order estimator with variable forgetting is seen to be better than for the fixed forgetting case in that it adapts to plant order changes more quickly.

Measuring model quality based on the model’s explanation error Q does not necessarily give the best model for predictive control. As we shall see this is due to the fact that Q depends on the expected value of the *one-step-ahead* prediction $\epsilon(t)$ and does not take into account the accuracy of long range predictions which are required by many predictive control schemes (such as the GPC algorithm of Clarke et al. (1987) and DMC of Cutler and Ramaker (1980)). In GPC and DMC these long range predictions are based on the estimated model’s step responses. Thus a better measure of model quality for these predictive controllers would be obtained by considering the estimated step responses of the 1st (dot-dash line), 2nd (dotted

line) and 3^{rd} (dashed line) order plant models with the true model (solid line) are given in Figure 2.10 where the parameters used were obtained just before each plant change. Looking at the estimated step responses in Figure 2.10 one observes that the step response of the best-fit (with order given in Figure 2.9) model captures the dynamics of the true model step response very well indeed, as is also evidenced by comparing the Bode magnitude plots of the best-fit model versus true model in Figure 2.11. The RMS errors in each model's step responses for a prediction horizon $N_2 = 50$ are given in the Table 2.7. Apart from the

Table 2.7: Selection of the best-fit model using Step responses

ARX order	Step response RMS errors: $\sqrt{1/50 \sum_{50} (y - \hat{y})^2}$				
	t=1000	t=2000	t=3000	t=4000	t=5000
1	0.5497	0.8657	0.0035	0.2941	0.5757
2	0.0112	0.0912	0.0153	0.2988	0.0251
3	0.0512	0.0307	0.0196	0.2534	0.0396
Best	2	3	1	3	2
True	2	2	1	3	2

second plant change $2000 \leq t < 3000$, one can see from the table above that the best-fit is achieved by the model whose order is the same as that of the plant, thus demonstrating the benefit to be gained from on-line model order selection in self-tuning control.

The simulation results show that an on-line MMRLS strategy with appropriate jacketing derived from the parameter convergence considerations of Parkum et al. (1992) can be used to provide the user with the LS estimates of an ARX model of a time-varying system which displays changes in model order and plant dead-time. The results also show that the step response coefficients obtained for the selected model are in general better than those obtained by a fixed (highest) order model and thus would afford better performance in an explicit self-tuning control scheme. Another important feature of the final algorithm is that since most existing recursive parameter estimators in self-tuning and adaptive control are based on the UD filter, very little change is required to existing code.

2.5 Conclusions

The MMLS estimation technique which is based on Niu's *AUDZ* method is seen to be appealing since it uses the data contained in the (augmented) information matrix more *efficiently* compared to classical LS in that it provides the user with the parameter estimates for each model from order 0 to m *simultaneously* together with the associated loss function values, and is inherently numerically stable due to the UD factorisation involved. Thus MMLS can be seen as a compact method for obtaining multiple order models. Indeed the time-recursive version is of particular interest for real-time applications.

The *AUDZ* method gives time-recursive MMLS via Bierman's UD estimator which is the preferred algorithm in self-tuning applications due to its robust numerical properties. Thus multiple models can be obtained for "free" simply by making a small change to existing UD estimator code in most self-tuning systems. It has also been shown that the MMLS technique is similar to the LSL algorithm for wide-sense stationary AR modelling, and has the following advantages over Levinson's algorithm:

- **Numerically superior** due to the UD factorisation involved;
- **More general model structure:** MMLS is not restricted to the AR case but can be applied to ARX models with possibly variable time delay;
- **Accurate for small data sets** ($N \geq n$) as it is an extension of classical LS and hence the sample covariances are not required;
- **Amenable to variable forgetting** techniques in the recursive version.

On the other hand, since the Toeplitz structure is no longer available for exploitation, MMLS requires $O(n^3)$ rather than $O(n^2)$ computations in the batch case.

The simulation results have shown the utility of MMRLS with appropriate jacketing for tracking the changes of the dynamics and model order of a time-varying process on-line. Thus this estimator can be used to provide a time-varying model with a model quality indicator for self-tuning systems such as adaptive GPC.

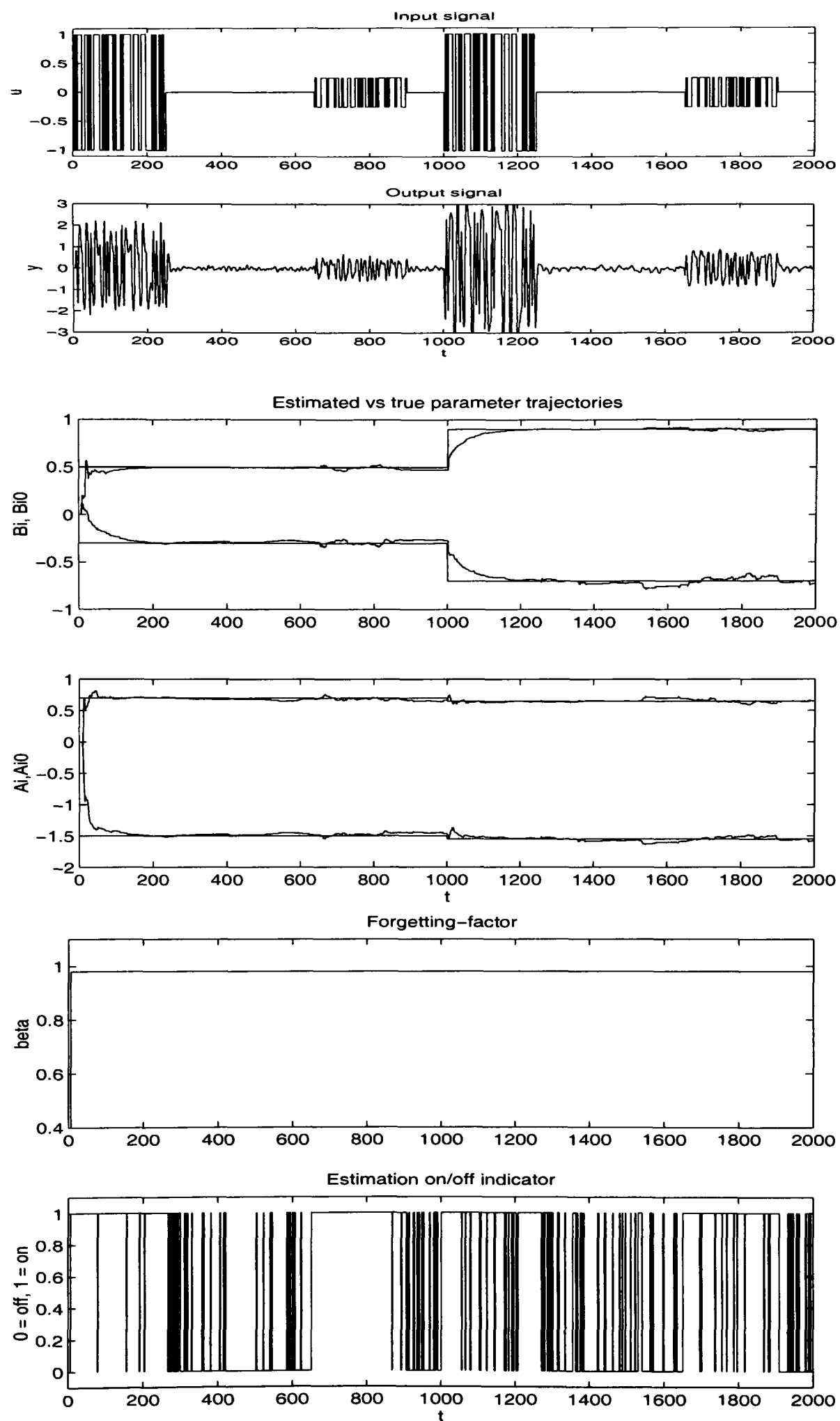


Figure 2.6: Parameter tracking (2nd order model, $0 \leq t < 2000$) with fixed forgetting ($\beta = 0.98$)

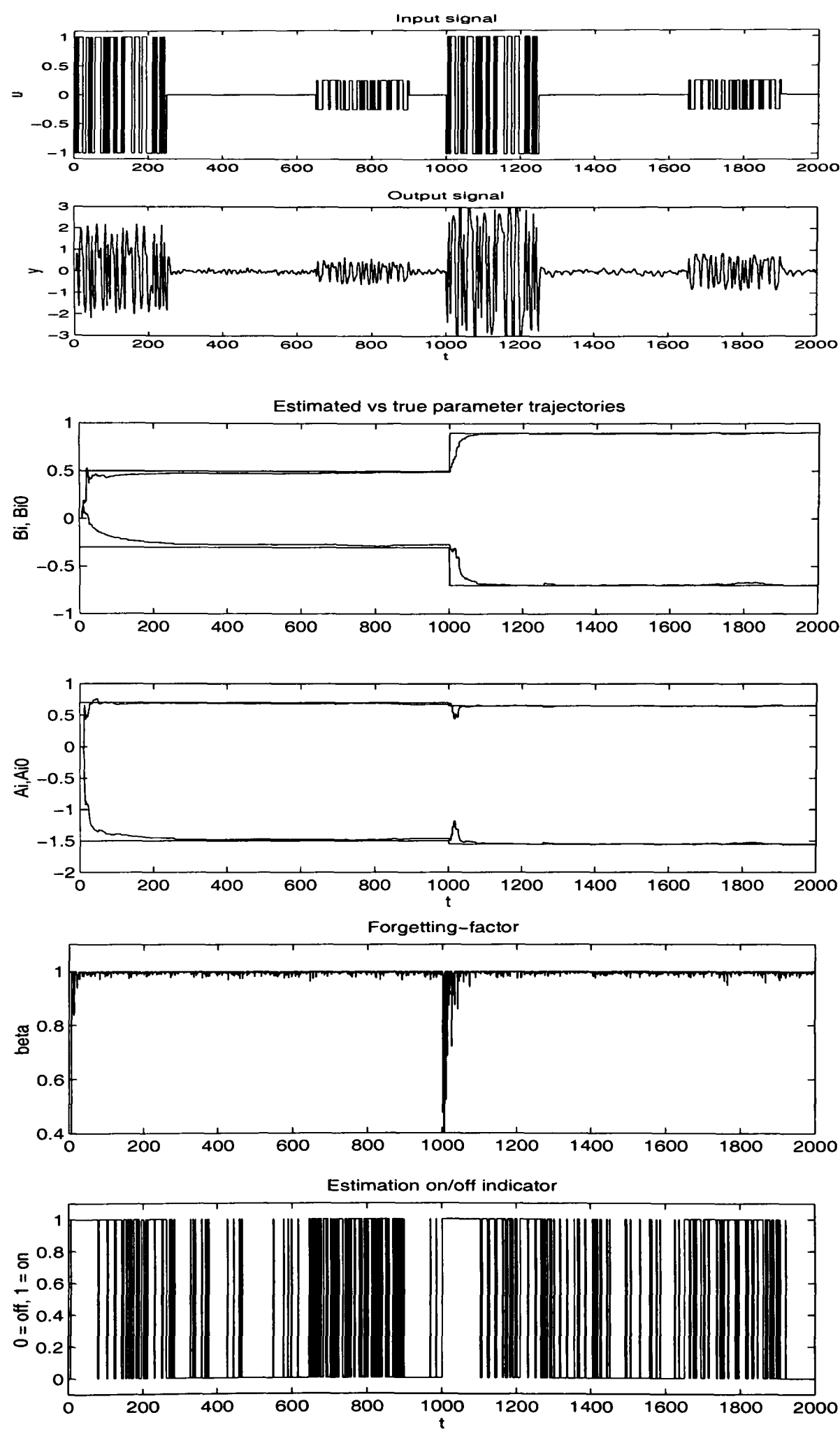


Figure 2.7: Parameter tracking (2^{nd} order model, $0 \leq t < 2000$) with variable forgetting

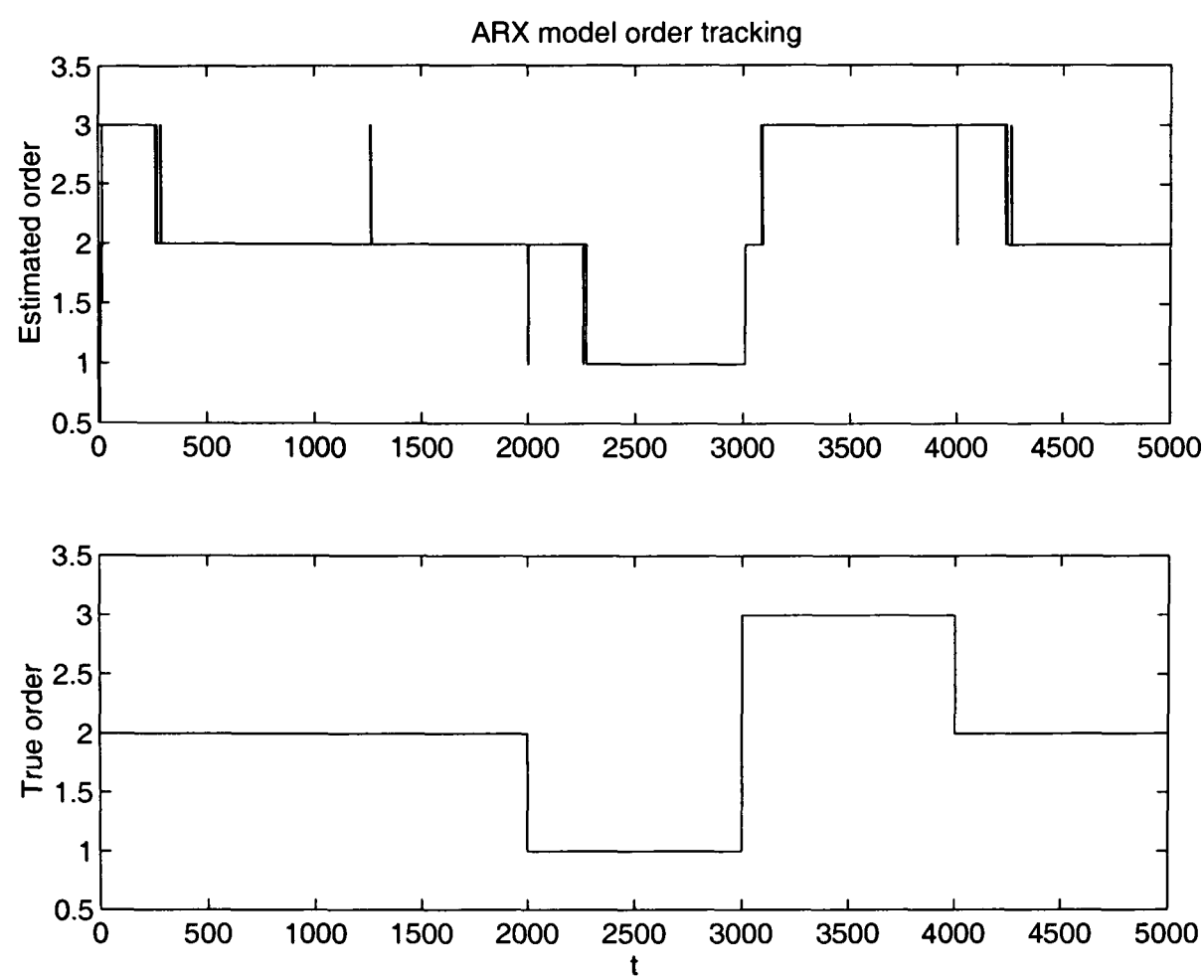


Figure 2.8: Estimated model order (fixed forgetting)

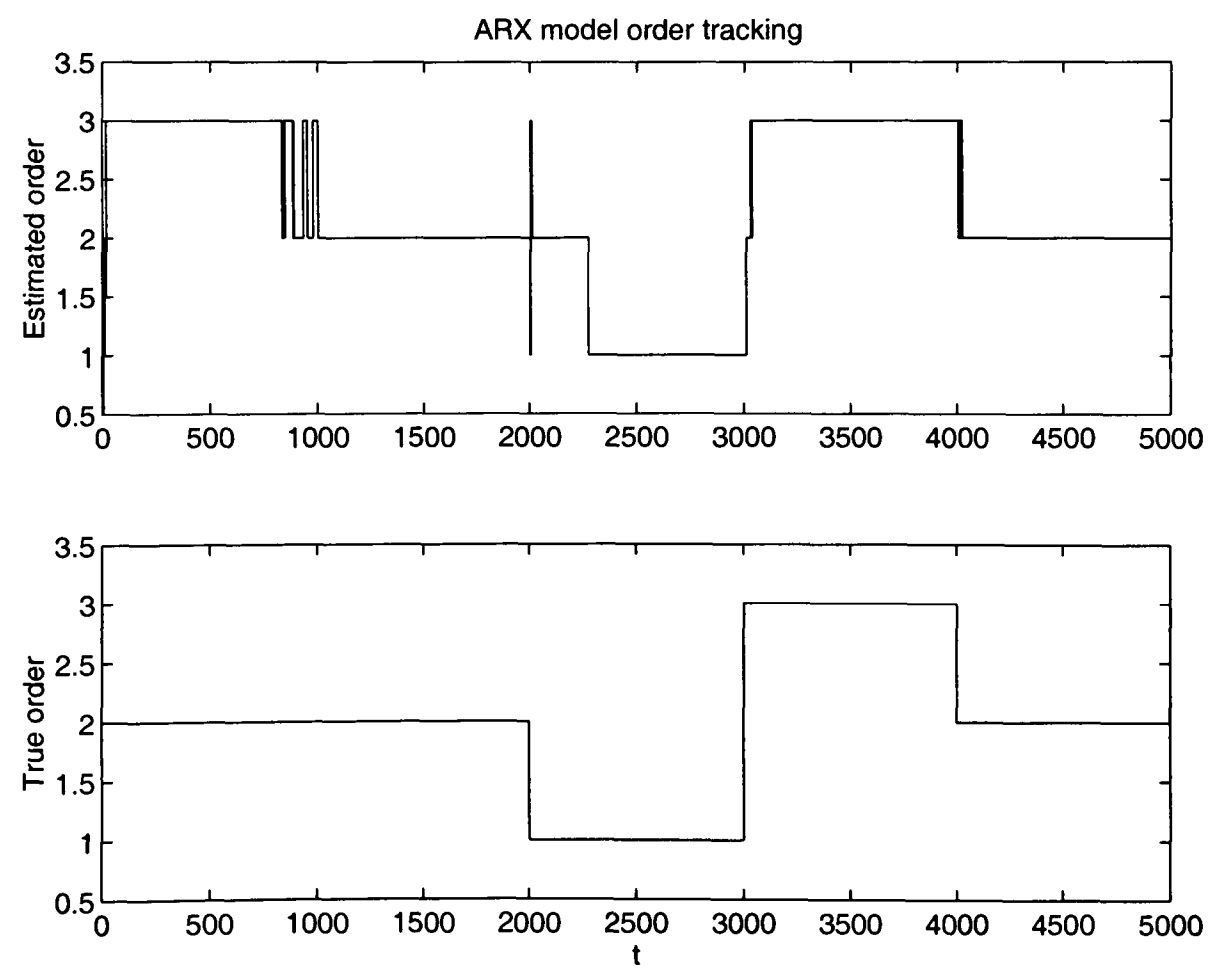


Figure 2.9: Estimated model order (variable forgetting)

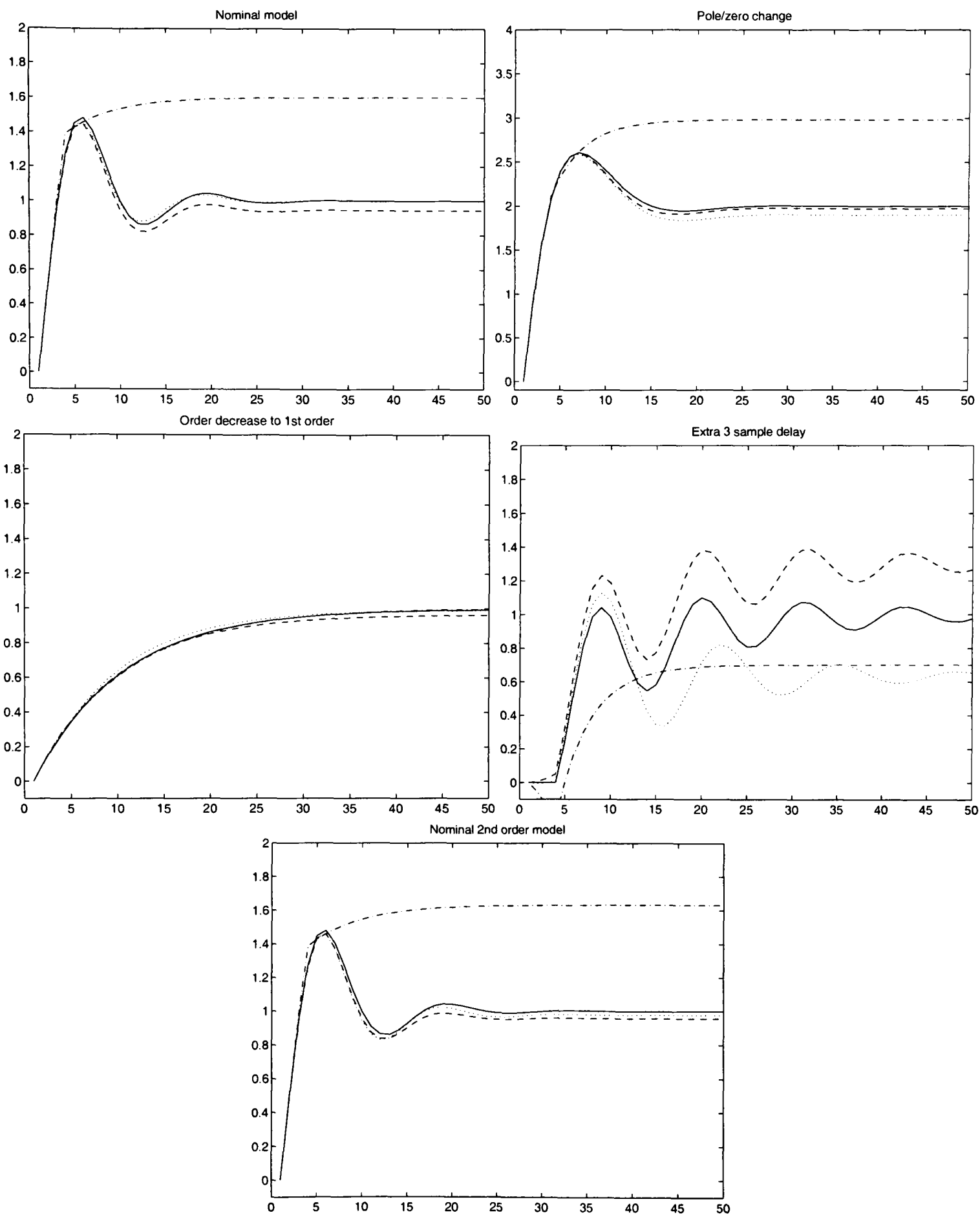


Figure 2.10: Estimated vs true step responses (dot-dash line → 1st order, dotted line → 2nd order, and dashed line → 3rd order plant models with the true model response plotted as a solid line)

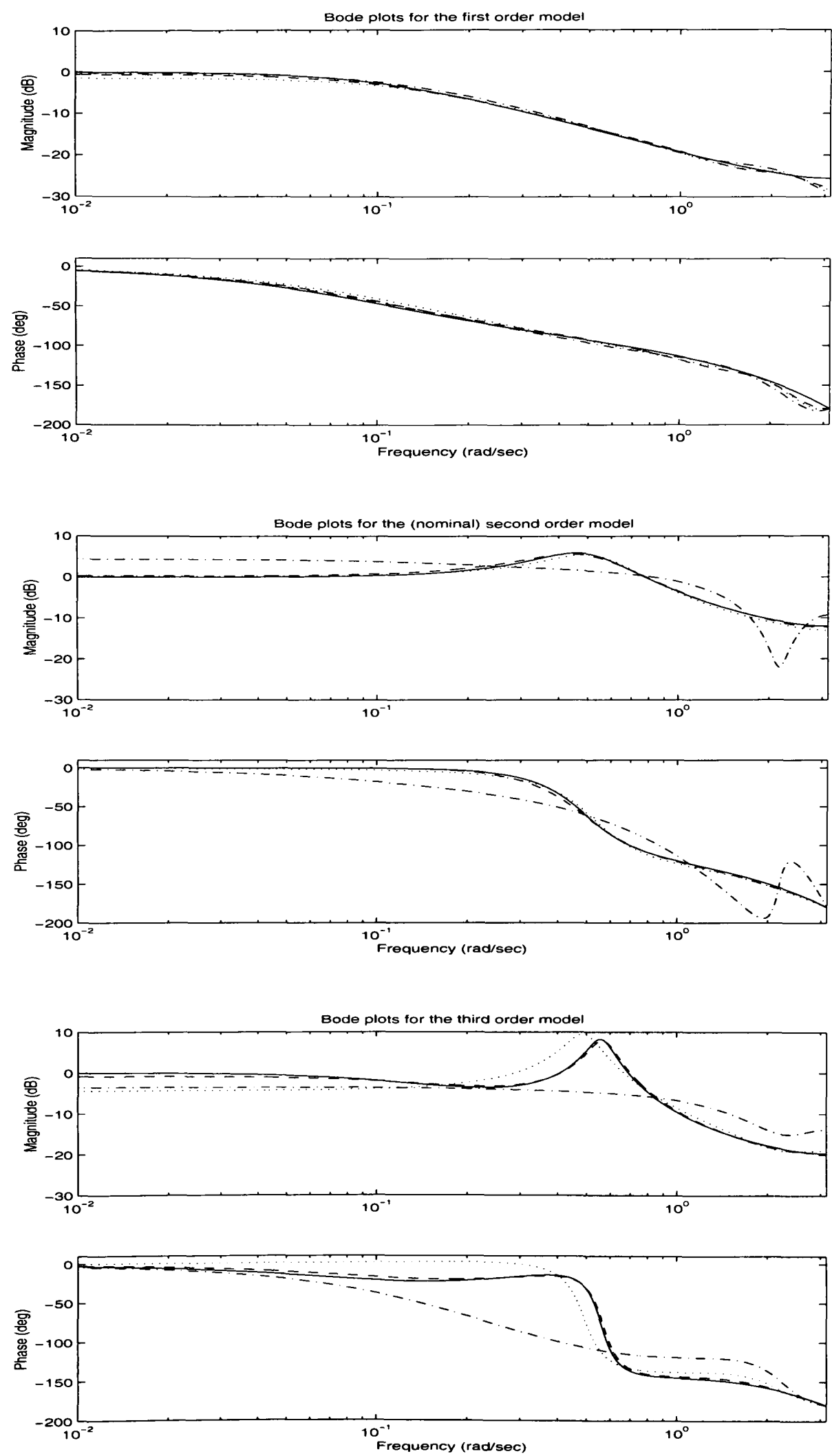


Figure 2.11: Estimated vs true model frequency responses (solid line → true model, dot-dash line → 1st order model, dotted line → 2nd order model, and dashed line → 3rd order model).

Chapter 3

The parameter tracking problem: a continuous-time approach

Numerous methods exist in the literature for continuous-time parameter estimation of time-varying systems (Young, 1981). Most of these are based on exponentially weighted RLS (see for example Gawthrop, 1987) or the Least Mean Squares (LMS) algorithm of Widrow and Hoff Jr. (1960), both of which have been studied in great detail with respect to tracking time-varying systems (see Ljung and Gunnarsson, 1990, for a survey). Though the fundamental ideas of continuous-time LMS and RLS estimation are well known, there has been no systematic study of the continuous-time versions of the various approaches to ‘forgetting’ that have been proposed in the discrete-time context. It is considered that this study is a useful guide to the intuition of the designer of a practical parameter estimation scheme.

3.1 Continuous-time recursive exponentially weighted Least-Squares

In this section various forgetting methods based on the differential equations and properties of exponentially weighted LS (see for example Gawthrop, 1987) are presented. The implications of exponential forgetting are observed by analysing the behaviour of the resulting differential equations and concepts such as regularisation (Levenberg, 1944; Marquardt, 1963), directional (Kulhavý and Kárný, 1984), restricted exponential (Hägglund, 1983) and variable forgetting (Fortescue et al., 1981) are introduced.

3.1.1 The off-line estimator

Consider the linear-in-the-parameters model with measurement noise:

$$y(t) = \boldsymbol{\theta}^T(t)\mathbf{x}(t) + e(t). \quad (\text{regression model}) \quad (3.1)$$

Given an estimate $\hat{\boldsymbol{\theta}}(t)$ the model can also be written:

$$y(t) = \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(t) + \epsilon(t),$$

where $\epsilon(t)$ is the model's prediction error. The estimate using time-weighted least squares is given by:

$$\hat{\boldsymbol{\theta}}(t) = \arg \min_{\hat{\boldsymbol{\theta}}} J(t, \hat{\boldsymbol{\theta}}),$$

where the weighted LS loss function is defined as

$$J(t, \hat{\boldsymbol{\theta}}) = \int_0^t w(t, \tau) \left(y(\tau) - \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(\tau) \right)^2 d\tau$$

and where $w(t, \tau)$ is a user-chosen weighting function. Traditionally this is taken to be just 1 for LS where all the data in the window $0 \leq \tau \leq t$ are weighted equally. However, if a time series model is being used (as is the case for estimating models for predictive controllers) and the process changes slowly with time, it would make more sense to emphasise the recent data when calculating the parameter estimates. The exponential windowing function $w(t, \tau) = e^{-\alpha(t-\tau)}$ has proven to be a popular choice in adaptive control; the properties of which will now be briefly discussed.

The exponential windowing function. Consider the integral:

$$I(t) = \int_0^t e^{-\alpha(t-\tau)} f(\tau) d\tau.$$

where the time-weighting is shown in Figure 3.1. Note that the time constant $T_c = 1/\alpha$ gives an indication of the window of past data contributing most to the final integral.

Then from first principles,

$$\frac{dI(t)}{dt} = f(t) - \alpha I(t)$$

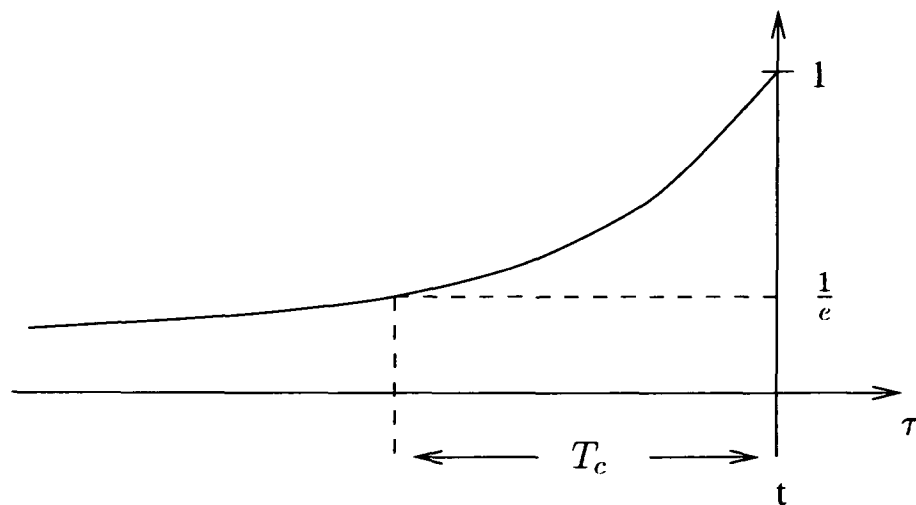


Figure 3.1: The exponential time-weighting function

which is the differential equation for the update of an exponentially time-weighted integral. Notice that $I(t)$ is a convolution integral, i.e.

$$I(t) = h * f$$

where h is the exponential window and $*$ is the convolution operator. Thus taking the Laplace transform of both sides we get

$$I(s) = \frac{1}{s + \alpha} F(s)$$

i.e. forgetting past data using an exponential time-weighting function can be interpreted as low-pass filtering the input where α is the first-order filter constant.

Computing the estimate. The Exponentially Weighted Least-Squares (EWLS) loss function is thus

$$J(t, \hat{\theta}) = \int_0^t e^{-\alpha(t-\tau)} \left(y(\tau) - \hat{\theta}^T(t) x(\tau) \right)^2 d\tau = \{\epsilon^2(t)\}_\alpha$$

where $\{\cdot\}_\alpha$ denotes the exponentially time-weighted convolution with time-constant $1/\alpha$.

Minimising via $\nabla_{\hat{\theta}} J(t, \hat{\theta}) = \mathbf{0}$ gives:

$$\left\{ \int_0^t e^{-\alpha(t-\tau)} x(\tau) x^T(\tau) d\tau \right\} \hat{\theta}(t) = \int_0^t e^{-\alpha(t-\tau)} x(\tau) y(\tau) d\tau$$

i.e.

$$\hat{\theta}(t) = S_{xx}^{-1}(t) s_{xy}(t), \quad (\text{EWLS estimate}) \quad (3.2)$$

where

$$\mathbf{S}_{xx}(t) = \int_0^t e^{-\alpha(t-\tau)} \mathbf{x}(\tau) \mathbf{x}^T(\tau) d\tau \quad (3.3)$$

and

$$\mathbf{s}_{xy}(t) = \int_0^t e^{-\alpha(t-\tau)} \mathbf{x}(\tau) y(\tau) d\tau. \quad (3.4)$$

3.1.2 Statistical properties

Using the true model of eqn.(3.1) the solution can also be written:

$$\mathbf{S}_{xx}(t) \hat{\boldsymbol{\theta}}(t) = \mathbf{S}_{xx}(t) \boldsymbol{\theta}(t) + \mathbf{s}_{xe}(t),$$

i.e.

$$\hat{\boldsymbol{\theta}}(t) = \boldsymbol{\theta}(t) + \mathbf{S}_{xx}^{-1}(t) \mathbf{s}_{xe}(t), \quad (3.5)$$

where

$$\mathbf{s}_{xe}(t) = \int_0^t e^{-\alpha(t-\tau)} \mathbf{x}(\tau) e(\tau) d\tau,$$

from which we get that if the measurement noise sequence $\{e(\tau)\}$ is statistically independent of the input data sequences $\{x_i(\tau)\}$ for $(0 \leq \tau \leq t)$, which is true for ‘white’ noise sequences with zero mean, and the true model coefficients are constant ($\boldsymbol{\theta}(t) = \boldsymbol{\theta}$ for all t) then

$$E \{ \hat{\boldsymbol{\theta}}(t) \} = \boldsymbol{\theta}$$

i.e. the estimate is said to be *unbiased*.

The covariance matrix for this fixed true model case is given by:

$$\begin{aligned} \text{cov } \hat{\boldsymbol{\theta}}(t) &= E \left\{ \left(\hat{\boldsymbol{\theta}}(t) - E\{\hat{\boldsymbol{\theta}}(t)\} \right) \left(\hat{\boldsymbol{\theta}}(t) - E\{\hat{\boldsymbol{\theta}}(t)\} \right)^T \right\} = E \left\{ \tilde{\boldsymbol{\theta}}(t) \tilde{\boldsymbol{\theta}}^T(t) \right\} \\ &= \mathbf{S}_{xx}^{-1}(t) \mathbf{s}_{xe}(t) \mathbf{s}_{xe}^T(t) \mathbf{S}_{xx}^{-1T}(t), \end{aligned}$$

where $\tilde{\boldsymbol{\theta}}(t) = \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(t)$ is the parameter error vector. The estimated parameter covariance reduces to:

$$\text{cov } \hat{\boldsymbol{\theta}}(t) = \sigma_0^2 \mathbf{S}_{xx}^{-1}(t), \quad (3.6)$$

if the measurement noise $\{e(t)\}$ is a sequence of independent random variables with zero mean and variance σ_0^2 .

Now given that $\hat{y}(\tau) = \mathbf{x}^T(\tau)\hat{\boldsymbol{\theta}}(t)$ is the model's prediction the least squares residual sequence can be obtained as follows:

$$\begin{aligned}\tilde{y}(\tau) &= y(\tau) - \hat{y}(\tau) \\ &= \mathbf{x}^T(\tau)\boldsymbol{\theta} + e(\tau) - \mathbf{x}^T(\tau)\hat{\boldsymbol{\theta}}(t) = \mathbf{x}^T(\tau)\tilde{\boldsymbol{\theta}}(t) + e(\tau) \\ &= \mathbf{x}^T(\tau)\boldsymbol{\theta} + e(\tau) - \mathbf{x}^T(\tau)(\boldsymbol{\theta} + \mathbf{S}_{xx}^{-1}(t)\mathbf{s}_{xe}(t)) \\ &= e(\tau) - \mathbf{x}^T(\tau)\mathbf{S}_{xx}^{-1}(t)\mathbf{s}_{xe}(t),\end{aligned}$$

and forming the integral:

$$\begin{aligned}\int_0^t e^{-\alpha(t-\tau)} \mathbf{x}(\tau) \tilde{y}(\tau) d\tau &= \int_0^t e^{-\alpha(t-\tau)} \mathbf{x}(\tau) (e(\tau) - \mathbf{x}^T(\tau)\mathbf{S}_{xx}^{-1}(t)\mathbf{s}_{xe}(t)) d\tau \\ &= \int_0^t e^{-\alpha(t-\tau)} \mathbf{x}(\tau) e(\tau) d\tau - \left\{ \int_0^t e^{-\alpha(t-\tau)} \mathbf{x}(\tau) \mathbf{x}^T(\tau) d\tau \right\} \mathbf{S}_{xx}^{-1}(t) \mathbf{s}_{xe}(t) \\ &= \mathbf{s}_{xe}(t) - \mathbf{S}_{xx}(t) \mathbf{S}_{xx}^{-1}(t) \mathbf{s}_{xe}(t) \\ &= \mathbf{0},\end{aligned}$$

which shows that $\{\mathbf{x}(\tau)\} \perp \{\tilde{y}(\tau)\}$ i.e. the data in each 'channel' x_i are orthogonal to the residuals and thus the EWLS estimate of eqn.(3.2) is a Best Linear Unbiased Estimate (BLUE) (Ljung, 1987) given the data sequence $\{\mathbf{x}(\tau)\}$ for $(0 \leq \tau \leq t)$.

Now since the predicted outputs $\hat{y}(\tau) = \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(\tau)$ they must also be orthogonal to the residuals $\tilde{y}(\tau)$ and thus the integral of squared observations:

$$\int_0^t e^{-\alpha(t-\tau)} y^2(\tau) d\tau = \int_0^t e^{-\alpha(t-\tau)} \hat{y}^2(\tau) d\tau + \int_0^t e^{-\alpha(t-\tau)} \tilde{y}^2(\tau) d\tau$$

and thus the ratio:

$$R_y^2(t) = \frac{\int_0^t e^{-\alpha(t-\tau)} \hat{y}^2(\tau) d\tau}{\int_0^t e^{-\alpha(t-\tau)} y^2(\tau) d\tau} = 1 - \frac{J(t, \hat{\boldsymbol{\theta}})}{\{y^2(t)\}_\alpha}$$

measures the proportion of total variation of the output y that is explained by the estimated model predictions $\hat{y}$. R_y is known as the Multiple Correlation Coefficient (MCC) (Ljung, 1987) and is often expressed as a percentage.

Remark 6 *The term*

$$Q^2(t) = \frac{J(t, \hat{\boldsymbol{\theta}})}{\{y^2(t)\}_\alpha} \quad (\text{Explanation error squared})$$

gives the percentage of the output variance not explained by the model predictions and is thus it also provides a useful measure of model quality (as in section 2.4).

3.1.3 Conversion to real-time form

Defining $\mathbf{S}(t) = \mathbf{S}_{xx}(t)$ and $\mathbf{r}(t) = \mathbf{s}_{xy}(t)$ and differentiating the off-line estimator,

$$\mathbf{S}(t)\hat{\boldsymbol{\theta}}(t) = \mathbf{r}(t),$$

with respect to t gives (omitting the argument t for clarity):

$$\mathbf{S} \frac{d\hat{\boldsymbol{\theta}}}{dt} + \frac{d\mathbf{S}}{dt} \hat{\boldsymbol{\theta}} = \frac{d\mathbf{r}}{dt}. \quad (3.7)$$

From eqn.(3.3) and eqn.(3.4), $\frac{d\mathbf{S}}{dt} = \mathbf{x}\mathbf{x}^T - \alpha\mathbf{S}$ and $\frac{d\mathbf{r}}{dt} = \mathbf{x}y - \alpha\mathbf{r}$, and thus eqn.(3.7) can be written

$$\mathbf{S} \frac{d\hat{\boldsymbol{\theta}}}{dt} + (\mathbf{x}\mathbf{x}^T)\hat{\boldsymbol{\theta}} = \mathbf{x}y + \alpha(\mathbf{S}\hat{\boldsymbol{\theta}} - \mathbf{r})$$

which can be simplified by noting that $\mathbf{S}\hat{\boldsymbol{\theta}} - \mathbf{r} = \mathbf{0}$ and that the current prediction error,

$$\epsilon(t) = y(t) - \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(t) = y(t) - \mathbf{x}^T(t)\hat{\boldsymbol{\theta}}(t)$$

and thus the differential equations of continuous-time RLS are:

$$\frac{d\mathbf{S}}{dt} + \alpha\mathbf{S} = \mathbf{x}\mathbf{x}^T, \quad (\text{Information update}) \quad (3.8)$$

$$\frac{d\hat{\boldsymbol{\theta}}}{dt} = \mathbf{S}^{-1}\mathbf{x}\epsilon. \quad (\text{Parameter update}) \quad (3.9)$$

In RLS it is conventional to propagate the inverse $\mathbf{P}(t) = \mathbf{S}^{-1}(t)$ in the calculations to eliminate the need for computing this inverse at each time step. Thus $\mathbf{P}(t)\mathbf{S}(t) = \mathbf{I}$ and hence

$$\frac{d}{dt}(\mathbf{P}(t)\mathbf{S}(t)) = \mathbf{0}$$

i.e.

$$\frac{d\mathbf{P}}{dt}\mathbf{S} + \mathbf{P}\frac{d\mathbf{S}}{dt} = \mathbf{0}$$

giving

$$\frac{d\mathbf{P}}{dt} = -\mathbf{P}\frac{d\mathbf{S}}{dt}\mathbf{P} \quad (3.10)$$

where $\mathbf{P}(t)$ is commonly loosely referred to as the *covariance matrix* since it is simply a scaled version of eqn.(3.6) in the ideal case. Substituting the information update eqn.(3.8) in eqn.(3.10) we get:

$$\frac{d\mathbf{P}}{dt} = -\mathbf{P}(\mathbf{x}\mathbf{x}^T - \alpha\mathbf{S})\mathbf{P}$$

$$\frac{d\mathbf{P}}{dt} + (\mathbf{P}\mathbf{x}\mathbf{x}^T - \alpha\mathbf{I})\mathbf{P} = \mathbf{0}. \quad (3.11)$$

The effect of the exponential forgetting-factor α when there is no excitation ($\mathbf{x} = \mathbf{0}$) is clearly shown in this formulation since the covariance update of eqn.(3.11) reduces to

$$\frac{d\mathbf{P}}{dt} - \alpha\mathbf{P} = \mathbf{0}$$

and thus the elements of $\mathbf{P}$ increase exponentially at a rate determined by α (i.e. the covariance matrix tends to ‘blow up’).

Regularised exponential forgetting

One method for avoiding these ‘blow ups’ is to add a small quantity $\mu\mathbf{I}$ to $\mathbf{x}\mathbf{x}^T$ in the RLS information update equation which effectively limits the covariance matrix to $\alpha/\mu\mathbf{I}$ since in the steady-state the covariance update of eqn.(3.11) with this modification becomes $\mathbf{P}(\mathbf{x}\mathbf{x}^T + \mu\mathbf{I}) - \alpha\mathbf{I} = \mathbf{0}$. Ljung and Söderström (1983) refer to this method as Levenberg-Marquardt regularisation after techniques suggested by Levenberg (1944) and Marquardt (1963) in relation to non-linear least squares minimisation.

Thus the continuous-time RLS estimator equations with exponential forgetting and regularisation are:

$$\epsilon(t) = y(t) - \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(t) \quad (\text{Prediction error})$$

$$\frac{d\mathbf{P}(t)}{dt} = \left(\alpha\mathbf{I} - \mathbf{P}(t) [\mathbf{x}(t)\mathbf{x}^T(t) + \mu\mathbf{I}] \right) \mathbf{P}(t) \quad (\text{Covariance update})$$

$$\frac{d\hat{\boldsymbol{\theta}}(t)}{dt} = \mathbf{P}(t)\mathbf{x}(t)\epsilon(t). \quad (\text{Parameter update})$$

Remark 7 *The Exponential Forgetting and Resetting Algorithm (EFRA) of Salgado et al. (1988) with recent modification by Han and Frank (1997) is essentially a discrete-time approximation of these continuous-time differential equations.*

Directional exponential forgetting

Alternatively, instead of using the covariance update

$$\frac{d\mathbf{S}(t)}{dt} = \mathbf{x}(t)\mathbf{x}^T(t) - \alpha\mathbf{S}(t), \quad (3.12)$$

where a portion $\alpha\mathbf{S}$ is removed from the information matrix $\mathbf{S}$ regardless of the information contained in the data vector $\mathbf{x}$ (i.e. forgets data indiscriminately), we could use the following *directional* information update (Hägglund, 1983):

$$\frac{d\mathbf{S}(t)}{dt} = \mathbf{x}(t)\mathbf{x}^T(t) - w(t)\mathbf{x}(t)\mathbf{x}^T(t), \quad (3.13)$$

which only forgets data “in the direction of” the new data $\mathbf{x}$. The advantage of using this form of update is that it prevents the covariance wind-up phenomenon observed with exponential forgetting when the data is not exciting in all directions of $\mathbb{R}^n$ (Kulhavý and Kárný, 1984), as is demonstrated later in Example 2 and in the simulation results of section 3.3.

The directional exponential forgetting strategy selects the value of the directional forgetting-factor $w(t)$ such that the resulting algorithm yields similar parameter tracking properties as RLS with exponential forgetting whilst avoiding covariance ‘blow up’ during periods of poor excitation. One might try equating the exponential and directional forgetting information updates of eqn.(3.12) and eqn.(3.13) as follows:

$$\alpha\mathbf{S}(t) = w(t)\mathbf{x}(t)\mathbf{x}^T(t)$$

however, this cannot be solved for $w(t)$ in the multiparameter case since $\mathbf{x}\mathbf{x}^T$ has only unit rank. The best we can hope to achieve is to obtain the same parameter update gain in the direction of the Kalman gain vector $\mathbf{k}$ and thus multiplying both sides by $\mathbf{k} = \mathbf{P}\mathbf{x}$:

$$\alpha\mathbf{x} = w(t)\mathbf{x}(\mathbf{x}^T\mathbf{P}\mathbf{x})$$

giving the directional forgetting-factor

$$w(t) = \begin{cases} \frac{\alpha}{L(t)} & L(t) > 0 \\ 0 & L(t) = 0 \end{cases}$$

where $L(t) = \mathbf{x}^T(t)\mathbf{P}(t)\mathbf{x}(t)$.

The continuous-time RLS estimator equations with directional exponential forgetting are thus

$$\epsilon(t) = y(t) - \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(t) \quad (\text{Prediction error})$$

$$\frac{d\mathbf{P}(t)}{dt} = (1 - w(t))\mathbf{P}(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{P}(t) \quad (\text{Covariance update})$$

$$\frac{d\hat{\boldsymbol{\theta}}(t)}{dt} = \mathbf{P}(t)\mathbf{x}(t)\epsilon(t). \quad (\text{Parameter update})$$

Remark 8 *This is the continuous-time version of the directional and restricted exponential forgetting algorithms of Kulhavý and Kárný (1984) and Hägglund (1983) respectively.*

In the following example the directional exponential forgetting update of eqn.(3.13) is compared to the standard exponential forgetting update of eqn.(3.12).

Example 2 (Directional exponential forgetting) *Suppose that at time t , the information matrix $\mathbf{S}(t) = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}$ and thus the covariance matrix $\mathbf{P}(t) = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$, and that the new data vector $\mathbf{x}(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ with an exponential forgetting-factor $\alpha = 1$.*

Then the exponential forgetting update is given by

$$\begin{aligned} \frac{d\mathbf{S}(t)}{dt} &= \mathbf{x}(t)\mathbf{x}^T(t) - \alpha\mathbf{S}(t) \\ &= \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} - \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 2 & -1 \end{pmatrix} \end{aligned}$$

and, using simple Euler integration¹ with unit step-length, the new information matrix is

$$\mathbf{S}(t+1) = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 2 \\ 2 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

¹Analytical integration can be performed here. However when these algorithms are implemented a Euler approximation is usually used to generate the corresponding discrete-time difference equations and hence the choice of Euler integration.

which is only of rank one with eigenvalues $\lambda = \{0, 2\}$. Thus the inverse of $\mathbf{S}^{-1}(t+1) = \mathbf{P}(t+1)$ does not exist and the parameter update of eqn.(3.9) fails.

On the other hand, using the equivalent directional forgetting-factor

$$w(t) = \frac{\alpha}{\mathbf{x}^T(t)\mathbf{P}(t)\mathbf{x}(t)} = \frac{1}{5}$$

we get the following information matrix update

$$\begin{aligned} \frac{d\mathbf{S}(t)}{dt} &= \mathbf{x}(t)\mathbf{x}^T(t) - w(t)\mathbf{x}(t)\mathbf{x}^T(t) \\ &= \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} - \frac{1}{5} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \frac{4}{5} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \end{aligned}$$

and thus

$$\mathbf{S}(t+1) = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} + \frac{4}{5} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 9 & -1 \\ -1 & 14 \end{pmatrix}$$

which is of full rank with eigenvalues $\lambda = \{1.7615, 2.8385\}$. Thus in this case the inverse $\mathbf{S}^{-1}(t+1) = \mathbf{P}(t+1)$ exists and the parameter update can be successfully computed using eqn.(3.9). ■

Thus this example has demonstrated that covariance blow-up experienced with standard exponential forgetting can be avoided by using directional exponential forgetting.

3.1.4 Convergence behaviour

An expression for the modelling error $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}$ can be obtained by differentiating with respect to t , i.e.

$$\begin{aligned} \frac{d\tilde{\boldsymbol{\theta}}}{dt} &= \frac{d\boldsymbol{\theta}}{dt} - \frac{d\hat{\boldsymbol{\theta}}}{dt} \\ &= \frac{d\boldsymbol{\theta}}{dt} - \left(\mathbf{P}\mathbf{x}y - \mathbf{P}\mathbf{x}\mathbf{x}^T\hat{\boldsymbol{\theta}} \right) \end{aligned}$$

and substituting $y = \mathbf{x}^T\boldsymbol{\theta} + e$ to give

$$\frac{d\tilde{\boldsymbol{\theta}}}{dt} + \mathbf{P}(\mathbf{x}\mathbf{x}^T)\tilde{\boldsymbol{\theta}} = \frac{d\boldsymbol{\theta}}{dt} - \mathbf{P}\mathbf{x}e. \quad (\text{RLS convergence}) \quad (3.14)$$

A solution can be found under the assumptions that the parameter vector θ is *constant* and that there is no measurement noise $e(t) = 0$. We start by considering the following:

$$\begin{aligned} \frac{d}{dt} \left(S(t) \tilde{\theta}(t) \right) &= S(t) \frac{d\tilde{\theta}(t)}{dt} + \frac{dS(t)}{dt} \tilde{\theta}(t) \\ &= S \frac{d\tilde{\theta}}{dt} + (xx^T - \alpha S) \tilde{\theta} \end{aligned}$$

but

$$S \frac{d\tilde{\theta}}{dt} + xx^T \tilde{\theta} = 0$$

and so

$$\frac{d}{dt} (S\tilde{\theta}) + \alpha (S\tilde{\theta}) = 0$$

giving

$$q(t) = q(0) e^{-\alpha t}$$

where $q(t) = S(t) \tilde{\theta}(t)$. The parameter errors are thus

$$\tilde{\theta}(t) = (S^{-1}(t) S(0)) \tilde{\theta}(0) e^{-\alpha t}$$

or, given that $S(0) = \gamma I$,

$$\tilde{\theta}(t) = \gamma P(t) \tilde{\theta}(0) e^{-\alpha t} \quad (\text{RLS parameter error}) \quad (3.15)$$

which shows that, if $P(t)$ is bounded, then RLS achieves exponential parameter convergence (determined by the exponential forgetting-factor α).

3.1.5 The variable forgetting-factor

The LS prediction error $\epsilon(t) = y(t) - \hat{y}(t)$ can be expressed as

$$\epsilon(t) = x^T(t) \tilde{\theta}(t) + e(t). \quad (3.16)$$

which shows that the prediction error is made up of two components: the model error and the unknown disturbance. Since the LS ‘loss function’ $J(t)$ can be written in terms of the prediction error ϵ :

$$J(t) = \int_0^t e^{-\alpha(t-\tau)} \epsilon^2(\tau) d\tau = \{\epsilon^2(t)\}_\alpha,$$

we see that the exponential forgetting-factor α simply filters the squared prediction errors. The choice of α depends on whether we want to identify a time-invariant system (where we

use a small value of α to average over a lot of data to eliminate the effects of noise on the cost and hence the estimate) or whether we want to track a time-varying plant (which requires a value of α of the order of the bandwidth of the plant variation so that the cost is able to reflect this variation). This highlights the problem with using a fixed forgetting-factor: no attempt is made to distinguish between a prediction error caused by plant-model mismatch $\tilde{\theta}$ and the measurement noise e . Thus we need to build some prior knowledge about the noise into the forgetting mechanism.

To begin with, consider a time-invariant plant, and suppose the noise is Gaussian with zero mean and variance σ_0^2 and that estimation is perfect for all t giving $\epsilon(t) = e(t)$ always. Then the expected value of the irreducible or ‘ideal loss’ is given by:

$$\begin{aligned}\mathcal{E}\{J_{opt}\} &= \int_0^t e^{-\alpha(t-\tau)} \sigma_0^2 d\tau \\ &= \sigma_0^2 e^{-\alpha t} \int_0^t e^{\alpha\tau} d\tau \\ &= \sigma_0^2 e^{-\alpha t} \frac{1}{\alpha} (e^{\alpha t} - 1) \\ &= \frac{\sigma_0^2}{\alpha} (1 - e^{-\alpha t}).\end{aligned}$$

For large t this approaches $\sigma_0^2/\alpha = T_c \sigma_0^2$ where T_c is the ‘asymptotic averaging time’ i.e. the past time-window containing data most influential in providing the estimate (see Figure 3.1). Thus we have a value for the optimal loss function given a perfect model and disturbances which are assumed Gaussian distributed with zero mean and known variance.

Now consider the case where the plant is time-varying and thus the estimation is no longer perfect. The corresponding differential equation for the loss function J is given by:

$$\frac{dJ}{dt} + \alpha J = \epsilon^2(t).$$

We would expect the average value of ϵ^2 to be greater than σ_0^2 since $\epsilon(t) = \mathbf{x}^T(t)\tilde{\theta}(t) + e(t)$. In particular when the estimate is far from the true value of $\theta(t)$ we expect that $\epsilon^2(t) \gg \sigma_0^2$. One way to proceed is to assert that the value of J provides a measure of the amount of ‘information’ used to obtain the parameter estimate and should be kept *constant* in the steady-state, i.e.

$$\alpha J_{ss} = \epsilon^2.$$

Thus a good value for the forgetting-factor α is obtained by setting J_{ss} equal to the optimal loss function:

$$\alpha(t)J_{opt} = \epsilon^2(t),$$

giving:

$$\alpha(t) = \frac{1}{T_c} \left(\frac{\epsilon(t)}{\sigma_0} \right)^2. \quad (\text{Variable forgetting-factor}) \quad (3.17)$$

Remark 9 *This is the idea behind the much acclaimed variable forgetting factor of Fortescue et al. (1981).*

3.2 Comparison with the Least Mean Squares algorithm

Consider the simple cost function

$$J(t) = \frac{1}{2} \epsilon^2(t) \quad (3.18)$$

which (in contrast to LS) only depends on the current prediction error $\epsilon(t) = y(t) - \hat{y}(t) = y(t) - \mathbf{x}^T(t)\hat{\boldsymbol{\theta}}(t)$. Differentiating with respect to the parameter vectors gives:

$$\begin{aligned} \frac{\partial J}{\partial \hat{\boldsymbol{\theta}}} &= \frac{\partial J}{\partial \epsilon} \frac{\partial \epsilon}{\partial \hat{\boldsymbol{\theta}}} \\ &= -\epsilon \mathbf{x}. \end{aligned}$$

The LMS algorithm proposed by Widrow and Hoff Jr. (1960) minimises eqn.(3.18) by adjusting the parameter estimates along the line of steepest descent, i.e.

$$\frac{d\hat{\boldsymbol{\theta}}}{dt} = -\mu \frac{\partial J}{\partial \hat{\boldsymbol{\theta}}} = \mu \mathbf{x} \epsilon \quad (3.19)$$

where μ is the estimator gain. Now the parameter error $\tilde{\boldsymbol{\theta}}(t) = \boldsymbol{\theta}(t) - \hat{\boldsymbol{\theta}}(t)$ and thus

$$\frac{d\tilde{\boldsymbol{\theta}}}{dt} = \frac{d\boldsymbol{\theta}}{dt} - \frac{d\hat{\boldsymbol{\theta}}}{dt}$$

hence the convergence of the LMS estimator is governed by the following d.e.

$$\frac{d\tilde{\boldsymbol{\theta}}}{dt} = \frac{d\boldsymbol{\theta}}{dt} - \mu \mathbf{x} (\mathbf{x}^T \tilde{\boldsymbol{\theta}} + e)$$

i.e.
$$\frac{d\tilde{\theta}}{dt} + \mu \mathbf{x} \mathbf{x}^T \tilde{\theta} = \frac{d\theta}{dt} - \mu \mathbf{x} e \quad (\text{LMS convergence}) \quad (3.20)$$

which is the same as RLS with $\mathbf{P} = \mu \mathbf{I}$.

In order to perform a simple comparison between LMS and RLS consider the LMS d.e. for estimating a constant gain K using a step of size U as the input:

$$\frac{d\tilde{\theta}(t)}{dt} + \mu U^2 \tilde{\theta}(t) = 0$$

where the parameter error $\tilde{\theta}(t) = K - \hat{\theta}(t)$.

The solution to this d.e. is given by:

$$\tilde{\theta}(t) = \tilde{\theta}(0) e^{-\mu U^2 t} \quad (3.21)$$

where the exponential convergence rate: $\beta_{lms}(t) = \mu U^2$ which is *constant* but depends on the size of the input step.

The solution given in eqn.(3.21) is monotonic and thus the Integral of Absolute Error (IAE) for the estimated parameter is given by:

$$\begin{aligned} \text{IAE}_{lms} &= \int_0^\infty \tilde{\theta}(t) dt = \tilde{\theta}(0) \int_0^\infty e^{-\mu U^2 t} dt \\ &= \frac{\tilde{\theta}(0)}{\mu U^2} \quad (\mu > 0) \end{aligned} \quad (3.22)$$

Now the corresponding d.e.'s for RLS are given by:

$$\begin{aligned} \frac{dS(t)}{dt} + \alpha S(t) &= U^2 \\ \frac{d\tilde{\theta}(t)}{dt} + \frac{U^2}{S(t)} \tilde{\theta}(t) &= 0 \end{aligned}$$

If $S(0) = \gamma U^2$ then we get the closed-form solutions:

$$\tilde{\theta}(t) = \frac{\gamma}{t + \gamma} \tilde{\theta}(0) \quad (\alpha = 0)$$

$$\tilde{\theta}(t) = \frac{\gamma}{(\gamma - 1/\alpha) + e^{\alpha t}/\alpha} \tilde{\theta}(0) \quad (\alpha > 0)$$

i.e. as $\gamma \rightarrow 0$, $\tilde{\theta}(t) \rightarrow 0$ for $t > 0$. The instantaneous exponential convergence rate of the parameter error is given by $\beta_{rls}(t) = U^2/S(t)$ for $\alpha \neq 0$, and since $\lim_{t \rightarrow \infty} S(t) = U^2/\alpha$,

the convergence rate of RLS tends towards a constant value of $\beta_{rls} \rightarrow \alpha$. In particular if $\alpha = 1/\gamma = U^2/S(0)$ then

$$\tilde{\theta}(t) = \tilde{\theta}(0)e^{-\alpha t}$$

and we get a constant convergence rate of $\beta_{rls} = \alpha$ and thus the RLS estimator behaves identically to LMS with $\mu U^2 = \alpha$.

The total parameter error for a single parameter model using RLS with a step input is given by

$$\begin{aligned} \text{IAE}_{rls} &= \int_0^\infty \tilde{\theta}(t) dt = \tilde{\theta}(0) \int_0^\infty \frac{\gamma}{(\gamma - 1/\alpha) + e^{\alpha t}/\alpha} dt \\ &= \frac{\gamma}{1 - \alpha\gamma} \tilde{\theta}(0) \log_e \left(\frac{1}{\alpha\gamma} \right) \quad (\alpha > 0) \end{aligned} \quad (3.23)$$

Figure 3.3 shows the results of a comparison between RLS and LMS for the single parameter model with a step input. The comparison was based on the IAE of eqn.(3.23) and eqn.(3.22) respectively using forgetting-factor values $\alpha = \mu$ and excitation values U^2 ranging from 10^{-6} to 10^3 , and where the RLS estimator was initialised with (a) $S(0) = 10^{-6}$ and (b) $S(0) = 10^6$.

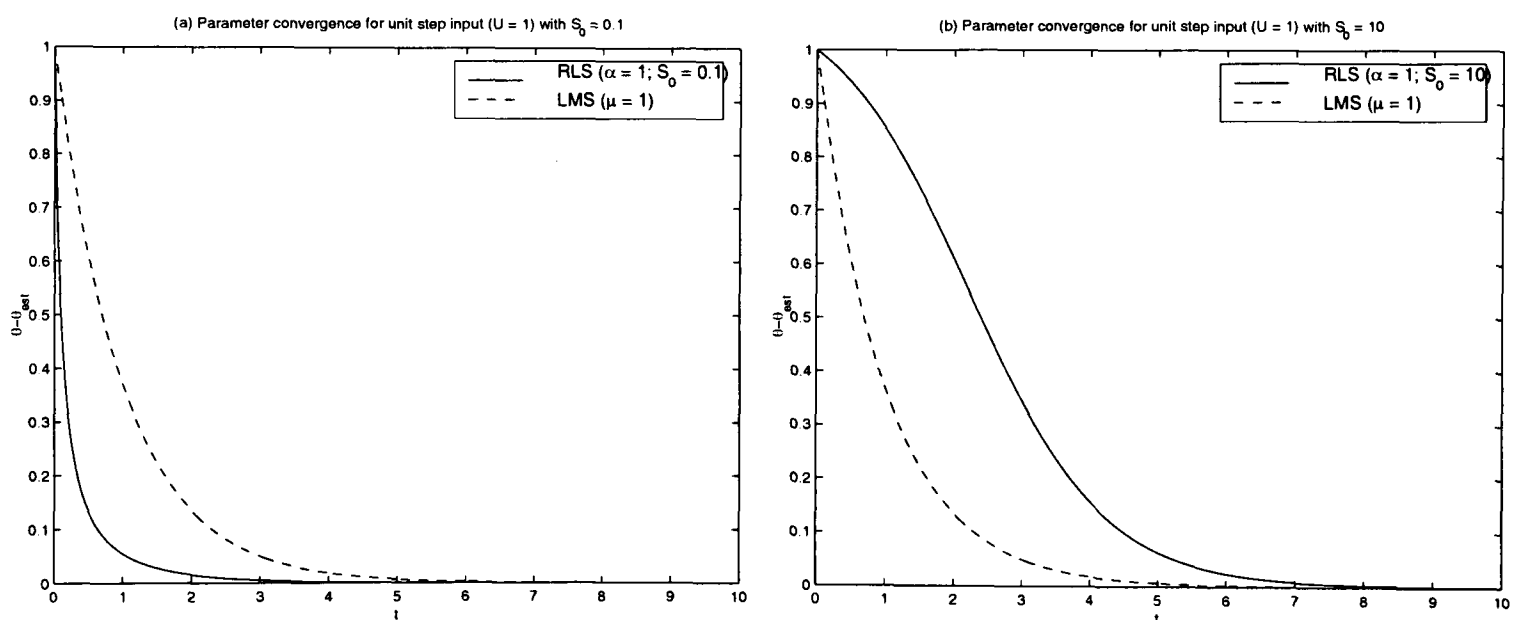


Figure 3.2: Convergence of RLS vs LMS for a unit step input

Figures 3.2 and 3.3 show that the performance of RLS is better than that of LMS for small values of $S(0)$ since in Figure 3.3(a) the RLS surface lies beneath that of LMS for *all* forgetting and excitation values. However it should be noted that the tracking ability of

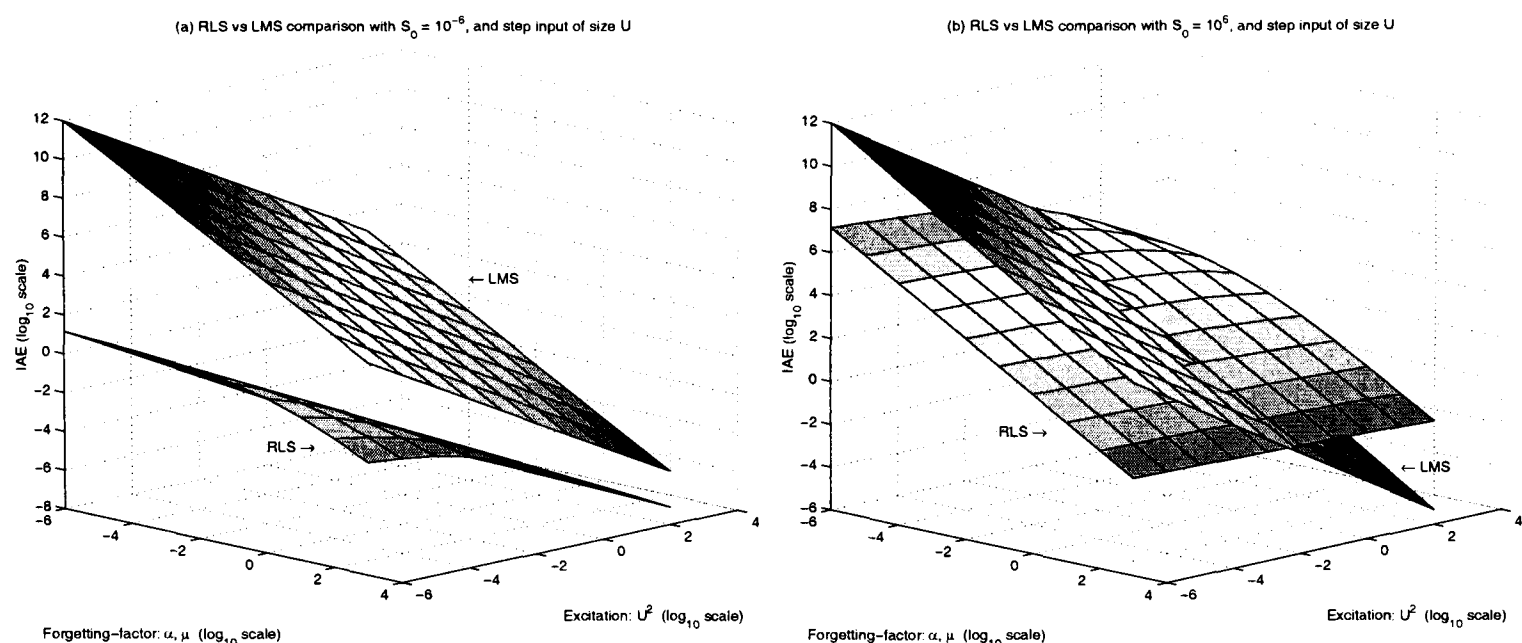


Figure 3.3: IAE comparison between RLS vs LMS for a unit step input

RLS is very much dependent on the initial value of the information matrix S and, if badly chosen, RLS approaches the solution *slower* than LMS as shown in Figure 3.2(b).

3.3 Simulation results

Figure 3.4 shows the SIMULINK worksheet which was used to simulate a 2-parameter continuous-time RLS estimator of a straight line model, i.e. $y = \theta_1 x_1 + \theta_2$, where x_1 was a sinusoidal input which was switched off during the period ($175s < t < 350s$).

The SIMULINK Dormand-Prince variable-step integration algorithm was employed to solve the RLS differential equations with a minimum step size of $0.01s$, and with relative and absolute tolerances of 10^{-6} . The estimator was initialised with a covariance matrix $P(0) = 10^9 I$, asymptotic memory length $T_c = 100s$ and expected noise variance $\sigma_0 = 0.01$. Looking at the results shown in Figure 3.5 one observes that the covariance element P_{11} blows up during the period $175s < t < 350s$ when there is no longer any sinusoidal data in x_1 . Thus the covariance matrix P becomes very ill-conditioned; rendering poor parameter estimates which are no longer able to track the system changes and ‘burst’ when new data becomes available at time $t = 350s$ due to their inflated sensitivity to the prediction errors.

In order to overcome this estimator wind-up phenomenon, a regularisation coefficient $\mu =$

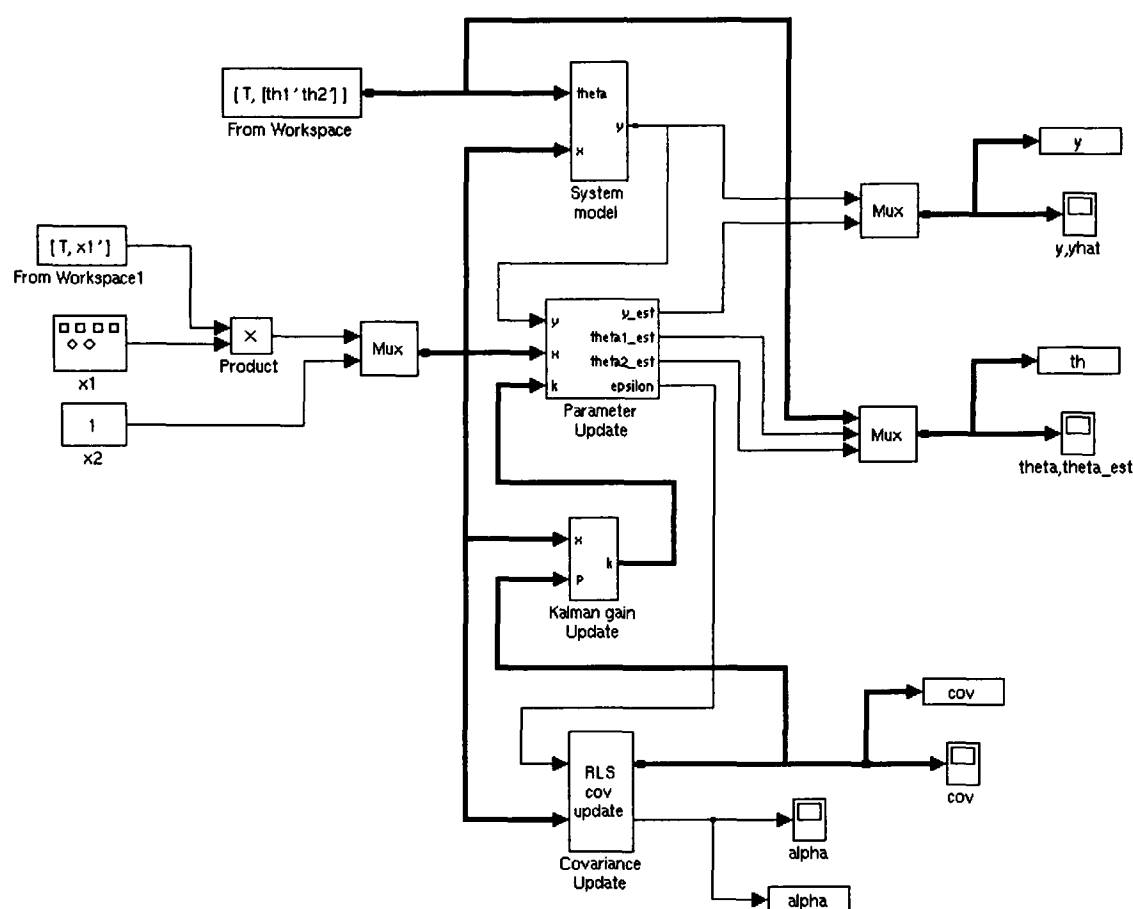


Figure 3.4: SIMULINK worksheet for a 2-parameter continuous-time RLS estimator

0.1 was included in the RLS equations, and starting this time with an initial covariance matrix $P(0) = I/\mu$. The results are shown in Figure 3.6. Notice how the variable forgetting mechanism achieves good parameter tracking performance and that ‘blow ups’ due to periods of poor excitation are prevented by the addition of μI to the information update equation which limits P to $\alpha_{max}/\mu I = 100$.

Figure 3.7 shows the results of the continuous-time RLS estimator with variable forgetting and regularisation in the presence of measurement noise. The estimator tuning parameters were the same as for the noise-free case except that the expected noise variance σ_0 was increased to 0.1 to match the actual measurement noise variance. From the plots the estimator is seen to successfully track the parameter changes in the presence of measurement noise albeit rather slower than in the noise-free case due the extra filtering (i.e. smaller α) required to obtain smoother estimates.

The results of the continuous-time RLS estimator using directional exponential variable forgetting are shown in Figure 3.8. Notice that the covariance element P_{11} (solid line in

subplot 3 of Figure 3.8) does not ‘blow up’ when the sinusoidal excitation x_1 is removed for $175s < t < 350s$ whereas in Figure 3.7 $\mathbf{P}_{11}$ increases exponentially up until the regularisation limit. Thus the directional forgetting scheme maintains good conditioning of the covariance matrix $\mathbf{P}$ and hence yields good parameter estimates when the data in each ‘channel’ is not sufficiently exciting. However the parameter tracking performance of this scheme is seen to be inferior to that of exponential forgetting with regularisation (compare the wild oscillations of $\hat{\theta}_2$ at times $t \approx 150s$ and $t \approx 475s$ in Figure 3.8 to the smooth estimate in Figure 3.7). Thus using a directional forgetting anti-wind-up mechanism instead of regularisation has a trade-off of slower parameter convergence, particularly when more exciting data becomes available.

The results of the continuous-time LMS estimator with and without measurement noise are shown in Figure 3.10 and Figure 3.9 respectively. Comparing the parameter trajectories of Figures 3.6 and 3.9 it is evident that the parameter tracking performance of the LMS estimator is not as good as RLS. Indeed, even if the LMS estimator gain μ is increased to a value greater than one, in an attempt to achieve better performance, the parameter oscillations observed in the constant offset estimate $\hat{\theta}_2$ get steadily worse. The trend is similar for the case with measurement noise: RLS with variable forgetting and regularisation clearly outperforms the LMS estimator in terms of tracking speed as shown in Table 3.1 where IAE_i denotes the integral of absolute error and ISE_i denotes the integral of squared error in parameter estimate $\hat{\theta}_i$. Again, if the LMS estimator gain μ is increased to a value greater than 0.1, in an attempt to achieve better performance, the parameter estimate variation increases due to the measurement noise.

Table 3.1: Results of different parameter tracking methods: with noise

Tracking method	IAE_1	IAE_2	ISE_1	ISE_2
RLS with variable forgetting and regularisation	231.4	44.11	259.5	16.35
RLS with directional variable forgetting	223.7	54.10	164.4	19.62
LMS	257.9	77.76	199.8	42.62

3.4 Conclusions

In this chapter the continuous-time domain was used to compare the performance of a number of parameter tracking techniques based on RLS and LMS methods. The continuous-time differential equations are well-known and appear in the literature, having been used to prove convergence behaviour of RLS and LMS methods. The taxonomy presented in this chapter is aimed at giving the user insight into the behaviour of various forgetting strategies such as the exponential forgetting and resetting algorithm of Salgado et al. (1988), variable forgetting of Fortescue et al. (1981) and restricted exponential/directional forgetting of Hägglund (1983) and Kulhavý and Kárný (1984), all of which were formulated in the context of discrete-time applications.

Results of simulations performed using these various continuous-time tracking algorithms demonstrate that RLS with variable forgetting and regularisation is the method of choice for efficient parameter tracking. On the basis of these results, the variable forgetting and regularisation forgetting method was implemented as the parameter tracking algorithm in the discrete-time adaptive control scheme described in chapter 5.

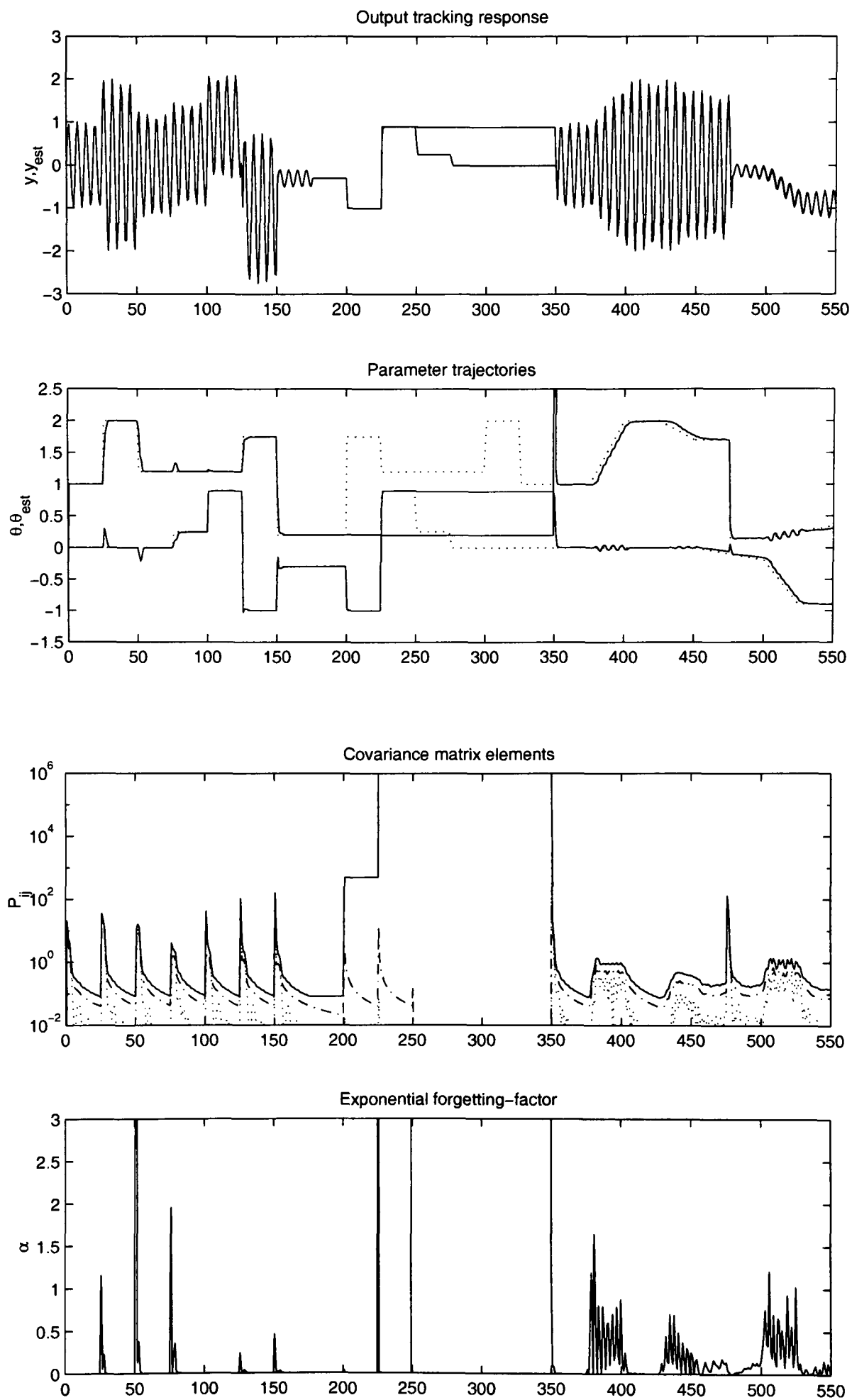


Figure 3.5: Continuous-time RLS with variable forgetting: noise-free case ($\sigma_0 = 0.01$, $T_c = 100s$)

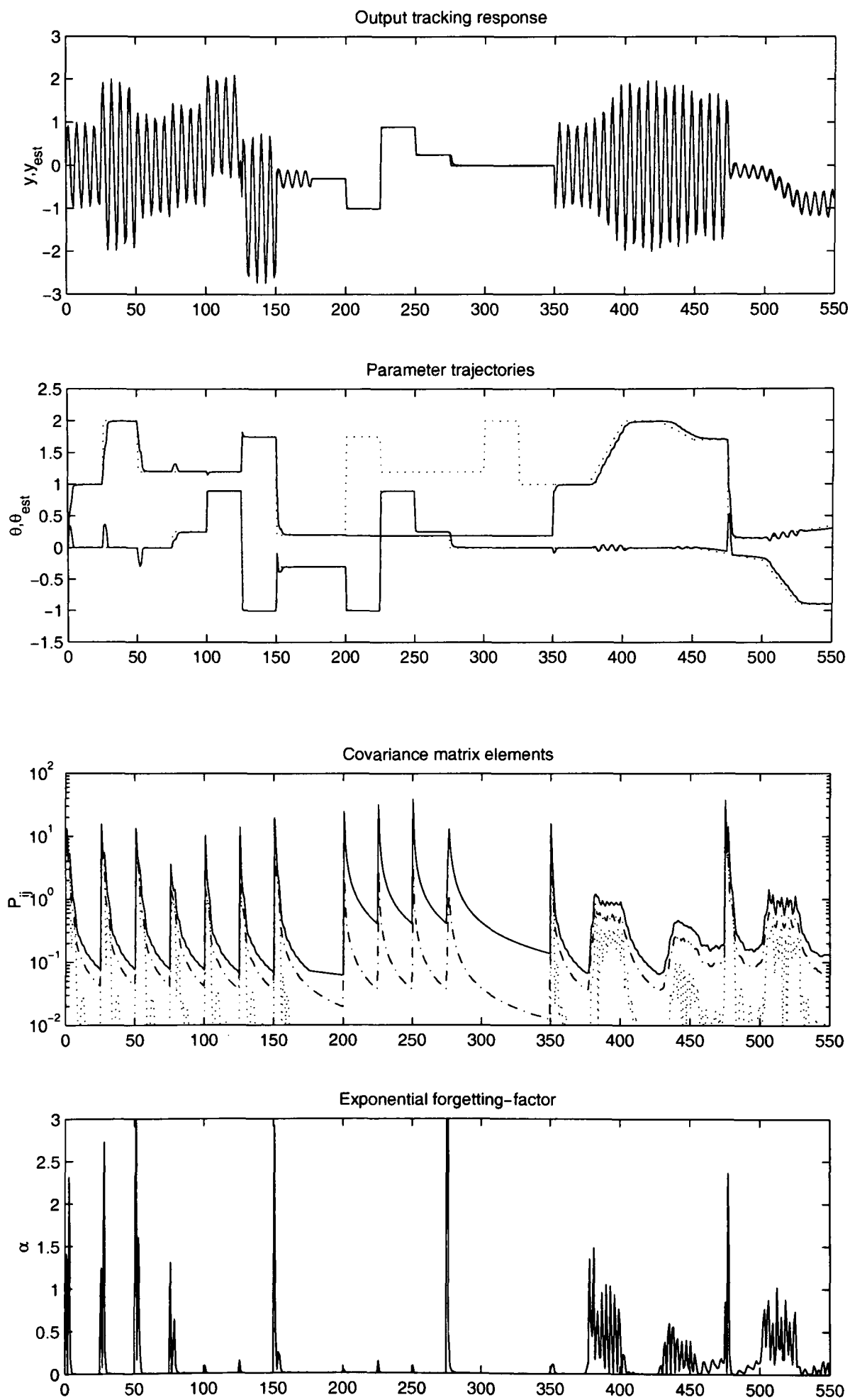


Figure 3.6: Continuous-time RLS with variable forgetting and regularisation: noise-free case ($\sigma_0 = 0.01$, $T_c = 100s$, $\mu = 0.1$)

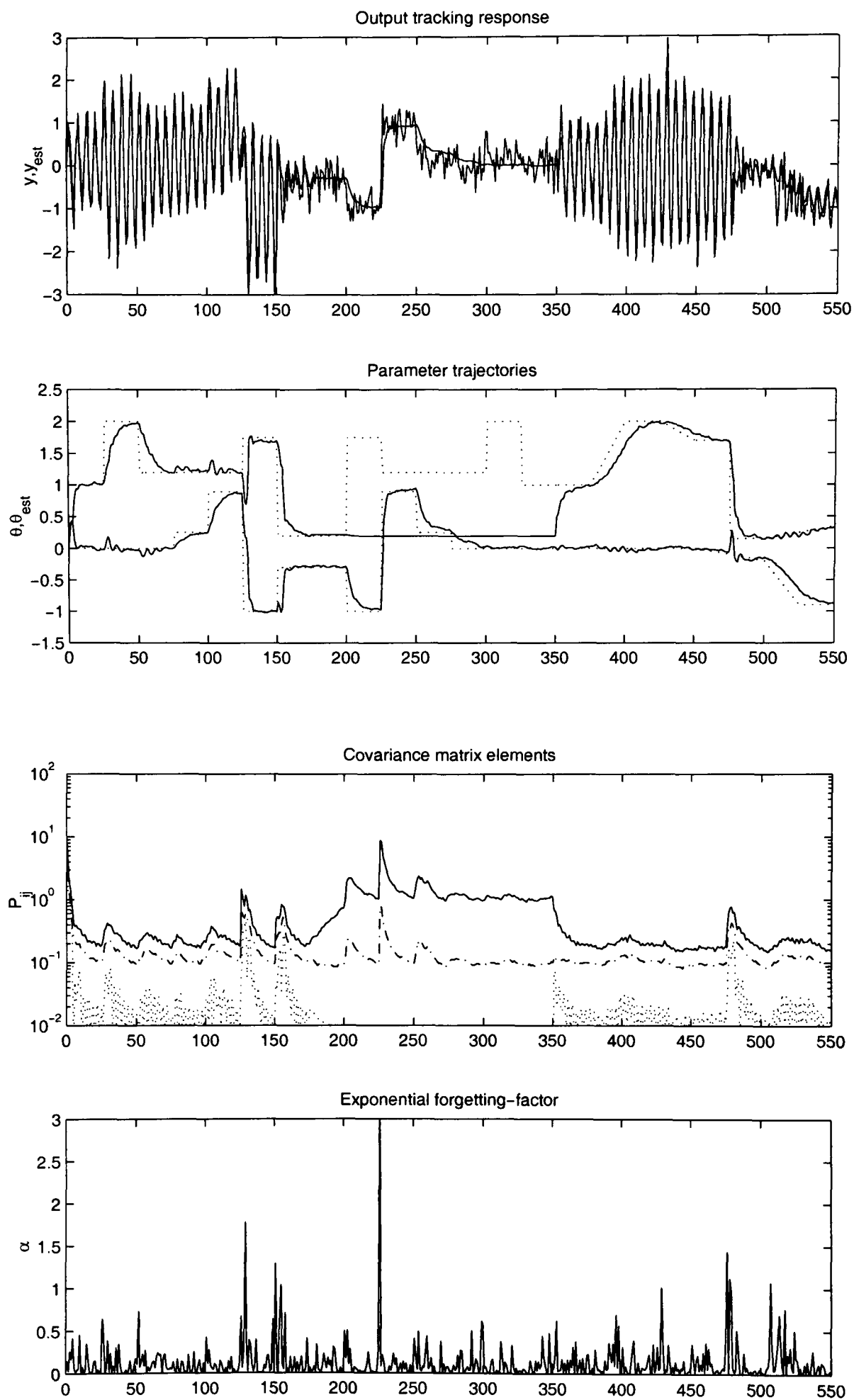


Figure 3.7: Continuous-time RLS with variable forgetting and regularisation: with noise ($\sigma_0 = 0.1$, $T_c = 100s$, $\mu = 0.1$)

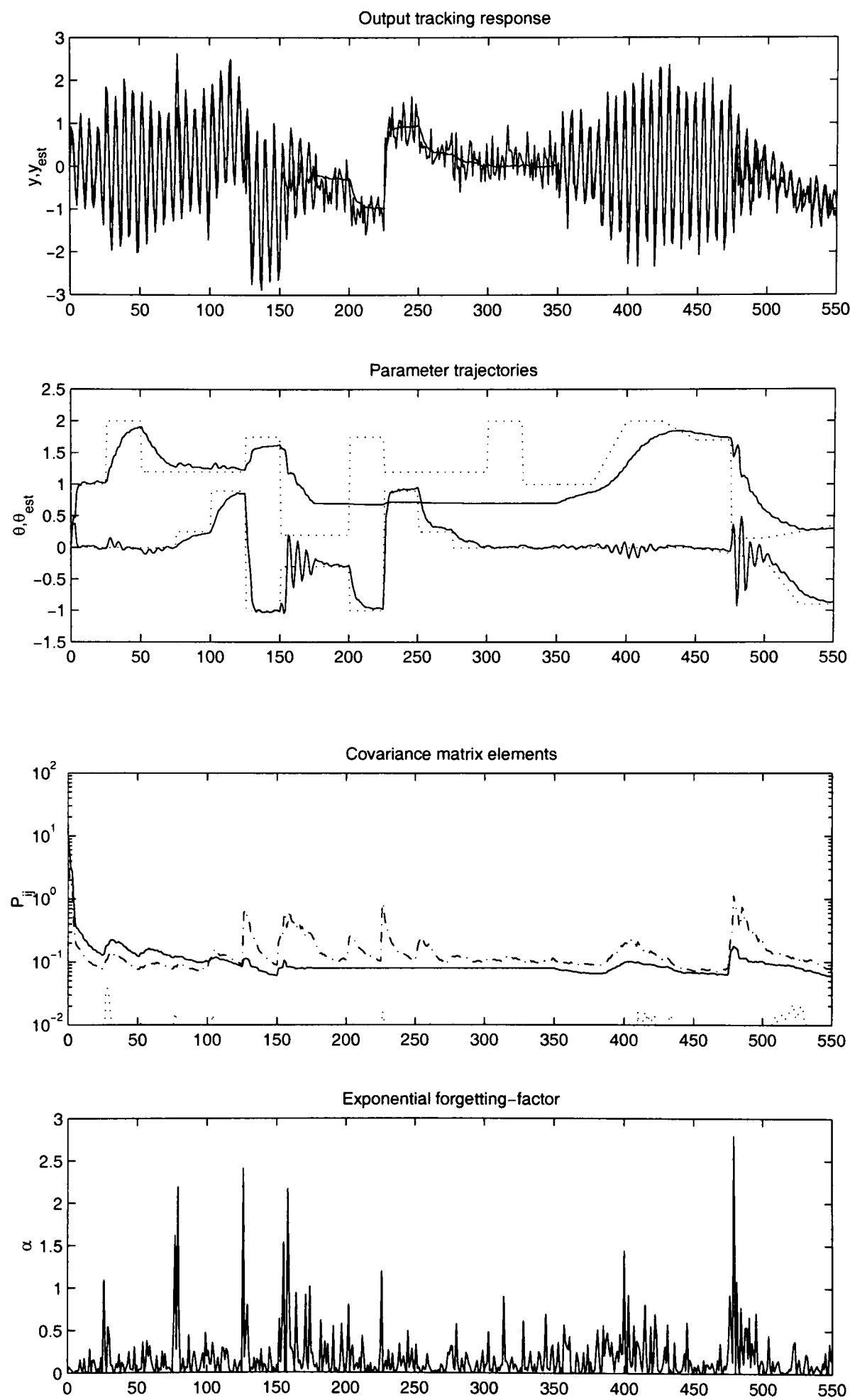


Figure 3.8: Continuous-time RLS with directional variable forgetting: with noise ($\sigma_0 = 0.1$, $T_c = 100s$)

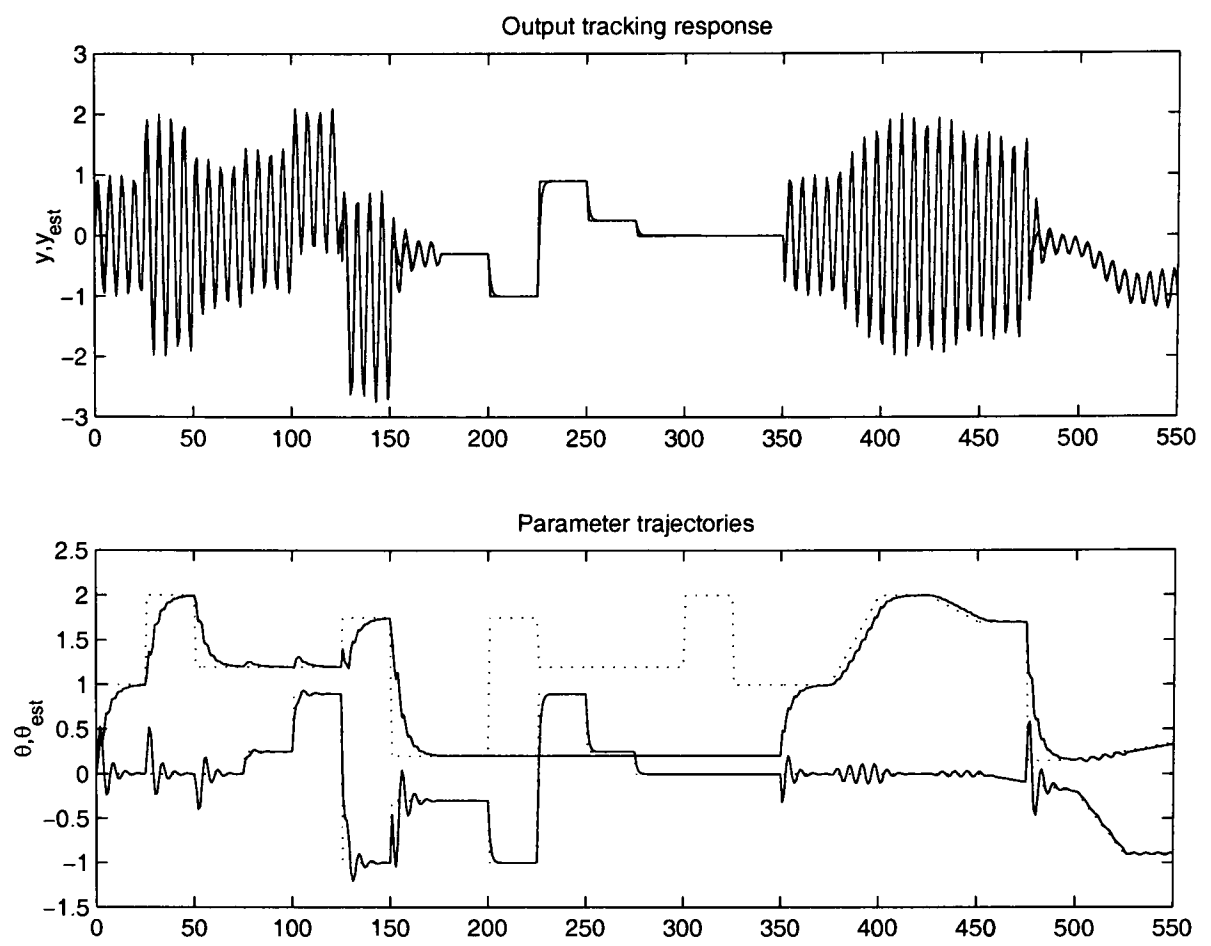


Figure 3.9: Continuous-time LMS ($\mu = 1$): without noise

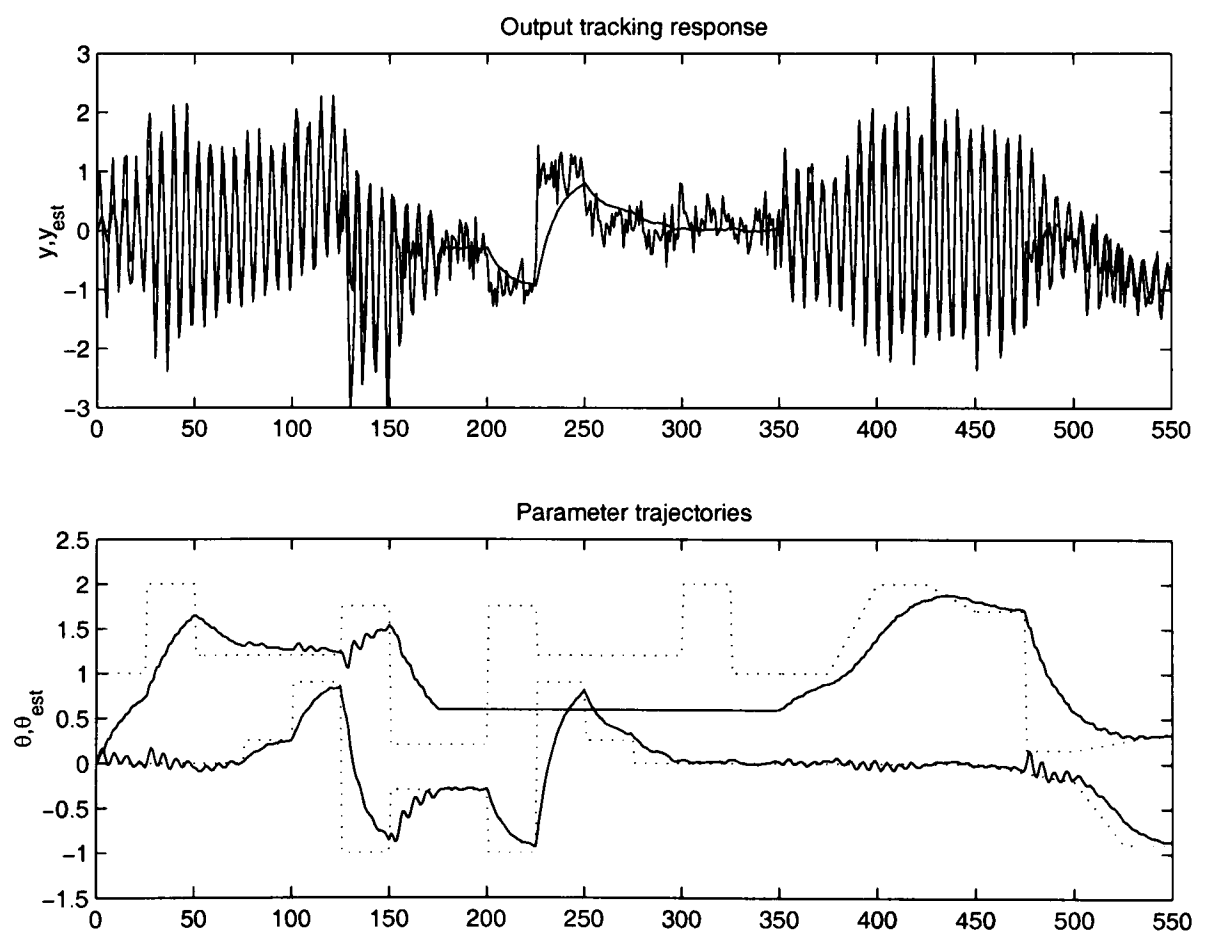


Figure 3.10: Continuous-time LMS ($\mu = 0.1$): with noise

Chapter 4

Multiple Model Least-Squares in the δ -domain

In a physically derived continuous-time model, one usually has some knowledge of values of system parameters or at least ranges of the parameters. This knowledge is normally destroyed in a sampled description. However, the fact that δ -domain descriptions converge to the underlying continuous-time descriptions as the sampling time tends to zero makes it possible to utilise such *a priori* knowledge in δ -domain design and estimation algorithms.

This chapter presents a derivation of MMLS in the δ -domain. The derivation uses Clarke's Lagrange multiplier technique and has been developed for both batch and recursive versions of the method (Kuznetsov et al., 1999). The δ -MMLS technique can then be used to obtain models for δ -domain MBPC algorithms (such as DGPC (Neumann et al., 1992, 1993), WGPC of Thygesen (1993), and DCGPC and DQGPC of Lauritsen et al. (1997)) directly and, in the limit where the sampling time tends to zero, CGPC (Demircioğlu and Gawthrop, 1991). The best-fit model is then be selected by using the set of loss function values.

4.1 Introduction

In recent years it has become widely accepted that the shift operator is not well suited to describe sampled data systems at fast sampling rates (Lauritsen et al., 1997). It has been suggested by Middleton and co-workers that using system models parameterised by

the *delta operator* (Goodwin et al., 1986; Middleton and Goodwin, 1990):

$$\delta = \frac{q - 1}{h}$$

where q is the forward shift operator and h is the sampling interval, gives close relationship between the sampled system and the underlying continuous-time system at fast sample rates. This is due to the fact that, if $x(t)$ is a differentiable signal, then

$$\lim_{h \rightarrow 0} \delta x(t) = \frac{d}{dt} x(t)$$

i.e. the discrete-time δ -operator tends towards differentiation in continuous-time in the limit where the sampling interval goes to zero. Middleton and Goodwin (1990) show that this operator facilitates the unification of modelling and control design in discrete and continuous time. To aid development of the multiple model δ -LS algorithm the following notation is used where the *generalised derivative* is written

$$\rho \text{ which denotes } \begin{cases} d/dt & \text{in continuous-time} \\ \delta & \text{in discrete-time} \end{cases}$$

and the *generalised integral* which is represented by

$$\mathcal{S}_{t_1}^{t_2} x(\tau) d\tau \text{ and denotes } \begin{cases} \int_{t_1}^{t_2} x(\tau) d\tau & \text{in continuous-time} \\ h \sum_{k=t_1/h}^{t_2/h-1} x(kh) & \text{in discrete-time.} \end{cases}$$

Middleton and Goodwin (1990) also introduce $E(\alpha, t)$ which is the *generalised exponential* denoting $e^{\alpha t}$ in continuous-time and its multistep Euler approximation $(1 + \alpha h)^{t/h}$ in discrete-time.

Using this notation Middleton and Goodwin (1990) derived the δ -domain batch LS and on-line RLS equations (see Appendix D). Clearly the LS equations only give a fixed order model which can lead to an under/over-parameterisation of the model, with associated penalty on performance of predictive controllers such as DGPC. Thus it would be useful to estimate a set of models and to choose the best one for the controller.

In the past Levinson-type methods have been used to provide the user with a set of reduced order models from which an appropriate model can be selected. However since the Toeplitz structure in the Levinson method is only obtained for infinite horizons it means that firstly,

large data sets are required to approximate the correlation matrix and secondly, variable forgetting cannot be applied. Moreover, as is shown in (Vijayan et al., 1991), the Levinson method breaks down when sampling at a rapid rate due to the fact that all the poles tend to one with the parameters a_i ($0 \leq i \leq n$) converging to the binomial coefficients $(-1)^i \binom{n}{i}$ independently of the underlying process. Thus Vijayan et al. (1991) proposed a Levinson-type algorithm for the δ -domain; again restricting it to AR modelling only. The δ -domain MMLS technique developed in this sequel provides another method for estimating reduced order models and exhibits better properties than the Levinson-type methods.

The on-line algorithm is seen to have further appeal as it is inherently based on Bierman's UD filter (Bierman, 1977) which is currently the state-of-the-art parameter estimation routine in the adaptive control field due to its computational efficiency and good numerical properties. Thus by making a few changes to existing parameter estimation code, the multiple order models and their losses can be obtained for "free". As with the batch case, the loss function values can be used to select the model order *on-line* for use in adaptive control schemes requiring δ -domain ARX plant models.

4.2 Batch Multiple Model Least-Squares in the δ -domain

Let us move the scalar output of the system $y(t)$ in the standard LS model to the right-hand side of the equation to obtain:

$$0 = \boldsymbol{\theta}_n^T(t) \mathbf{x}(t) - y(t) + e(t) = \boldsymbol{\theta}_{n+1}^T(t) \boldsymbol{\phi}(t) + e(t),$$

where

$$\boldsymbol{\theta}_{n+1}(t) = \begin{pmatrix} \boldsymbol{\theta}_n(t) \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1} \quad \text{and} \quad \boldsymbol{\phi}(t) = \begin{pmatrix} \mathbf{x}(t) \\ -y(t) \end{pmatrix} \in \mathbb{R}^{n+1}$$

are the *augmented vectors*, and the subscript n is used to denote an n -parameter model. Note that the coefficient associated with the output $-y(t)$ is simply *one*. Hence, the parameter estimate vector $\hat{\boldsymbol{\theta}}_{n+1}(t)$ is found by minimising the loss function

$$J(\boldsymbol{\theta}_{n+1}, t) = \mathcal{S}_0^t w(t, \tau) (\boldsymbol{\theta}_{n+1}^T(t) \boldsymbol{\phi}(\tau))^2 d\tau$$

subject to the constraint $\mathbf{q}_{n+1}^T \hat{\boldsymbol{\theta}}_{n+1}(t) = 1$, where $\mathbf{q}_{n+1} = (\underbrace{0 \dots 0}_n 1)^T \in \mathbb{R}^{n+1}$; $w(t, \tau)$ is some weighting function which is usually $E(-\alpha, t - \tau)$ in adaptive systems.

The Lagrange function is thus

$$L(\boldsymbol{\theta}_{n+1}, \mu, t) = J(\boldsymbol{\theta}_{n+1}, t) + \mu(t) \left(\mathbf{q}_{n+1}^T \hat{\boldsymbol{\theta}}_{n+1}(t) - 1 \right),$$

and minimising via $\nabla_{\boldsymbol{\theta}_{n+1}} L(\boldsymbol{\theta}_{n+1}, \mu, t) = \mathbf{0}$ gives:

$$\mathbf{S}_{\phi\phi}(t) \hat{\boldsymbol{\theta}}_{n+1}(t) + \mu(t) \mathbf{q}_{n+1} = \mathbf{0},$$

where

$$\mathbf{S}_{\phi\phi}(t) = \mathbf{S}_0^t E(-\alpha, t - \tau) \phi(\tau) \phi^T(\tau) d\tau \quad (\text{AIM}). \quad (4.1)$$

The AIM can be partitioned as follows ('t' is omitted for brevity):

$$\mathbf{S}_{\phi\phi} \hat{\boldsymbol{\theta}}_{n+1} = \begin{pmatrix} \mathbf{S}_{xx} & -s_{xy} \\ -s_{xy}^T & Y^2 \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\theta}}_n \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mu \end{pmatrix}. \quad (4.2)$$

The first n equations of (4.2) give the standard LS solution of eqn.(D.1), i.e.

$$\hat{\boldsymbol{\theta}}_n = \mathbf{S}_{xx}^{-1} s_{xy},$$

and the last gives the value of the Lagrange multiplier:

$$-\mu = -s_{xy}^T \hat{\boldsymbol{\theta}}_n + Y^2 = J^* = J_n,$$

i.e. $-\mu(t)$ is equal to the value of the LS loss function of the estimated n -parameter model.

Now defining the ACM as the inverse of the AIM: $\mathbf{P}_{n+1}(t) = \mathbf{S}_{\phi\phi}^{-1}(t)$, and using the block matrix inversion lemma¹, we see that eqn.(4.2) can be rewritten as:

$$\begin{pmatrix} \hat{\boldsymbol{\theta}}_n \\ 1 \end{pmatrix} = \mathbf{P}_{n+1} \begin{pmatrix} \mathbf{0} \\ -\mu \end{pmatrix} = \begin{pmatrix} \mathbf{S}_{xx}^{-1} + \hat{\boldsymbol{\theta}}_n \hat{\boldsymbol{\theta}}_n^T / J_n & \hat{\boldsymbol{\theta}}_n / J_n \\ \hat{\boldsymbol{\theta}}_n^T / J_n & 1 / J_n \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ J_n \end{pmatrix}$$

¹Block matrix inversion lemma:

$$\begin{pmatrix} \mathbf{A} & \mathbf{v} \\ \mathbf{v}^T & \alpha \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1}(\mathbf{I} + \mathbf{v}\mathbf{v}^T \mathbf{A}^{-1} / \beta) & -\mathbf{A}^{-1} \mathbf{v} / \beta \\ -\mathbf{v}^T \mathbf{A}^{-1} / \beta & 1 / \beta \end{pmatrix} \quad \text{where} \quad \beta = \alpha - \mathbf{v}^T \mathbf{A}^{-1} \mathbf{v}.$$

or

$$\begin{pmatrix} \hat{\theta}_n \\ 1 \end{pmatrix} = \left\{ \begin{pmatrix} \mathbf{I} & \hat{\theta}_n \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{S}_{xx}^{-1} & \mathbf{0} \\ \mathbf{0} & 1/J_n \end{pmatrix} \begin{pmatrix} \mathbf{I} & \hat{\theta}_n \\ \mathbf{0} & 1 \end{pmatrix}^T \right\} \begin{pmatrix} \mathbf{0} \\ J_n \end{pmatrix}. \quad (4.3)$$

Hence by decomposing the ACM into block $\mathbf{UDU}^T$ form we get *both* the LS parameter estimates in the last column of $\mathbf{U}$ *and* the (inverse) LS loss function as the last entry of $\mathbf{D}$.

Now since $\mathbf{S}_{xx}^{-1} = \mathbf{P}_n$, the reduced order covariance matrix, we see from eqn.(4.3) that $\mathbf{P}_{n+1}$ possesses a *nested structure* because $\mathbf{P}_n$ appears in $\mathbf{P}_{n+1}$:

$$\mathbf{P}_{n+1} = \begin{pmatrix} \mathbf{I} & \hat{\theta}_n \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{P}_n & \mathbf{0} \\ \mathbf{0} & \frac{1}{J_n} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \hat{\theta}_n \\ \mathbf{0} & 1 \end{pmatrix}^T,$$

and by recursively decomposing all the nested ACM's we get the following theorem:

Theorem 2 (The δ -MMLS method) *If the ACM is factorised into $\mathbf{UDU}^T$ form, i.e. $\mathbf{P}_{n+1} = \mathbf{UDU}^T$, then the LS parameter estimates $\hat{\theta}_i$ corresponding to the i -parameter model:*

$$0 = \hat{\theta}_{1i}x_1 + \hat{\theta}_{2i}x_2 + \dots + \hat{\theta}_{ii}x_i - x_{i+1} + \epsilon,$$

for $i = 1, 2, \dots, n$, are contained in the $(i+1)^{th}$ column of the parameter matrix:

$$\mathbf{U} = \begin{pmatrix} 1 & \hat{\theta}_{11} & \dots & \hat{\theta}_{1n} \\ & 1 & & \vdots \\ & & \ddots & \hat{\theta}_{nn} \\ & & & 1 \end{pmatrix} \quad (\text{Parameter matrix}),$$

and the inverse of the associated LS loss function $1/J_i$ is the $(i+1)^{th}$ diagonal entry of the loss function matrix:

$$\mathbf{D} = \begin{pmatrix} \frac{1}{J_0} & & & \\ & \frac{1}{J_1} & & \\ & & \ddots & \\ & & & \frac{1}{J_n} \end{pmatrix} \quad (\text{Loss function matrix}).$$

Proof 2 *By induction as in Proof 1.* ■

Table 4.1: A batch implementation of δ -MMLS

Decompose $S_{\phi\phi}$	$\longrightarrow$	LDL^T
then,		
Estimates $\hat{\theta}_i$	$\longrightarrow$	$(i+1)^{th}$ row of L^{-1}
Loss function J_i	$\longrightarrow$	$(i+1)^{th}$ entry of D

Since the AIM is symmetric and positive semi-definite, it can be factorised into LDL^T form.

Taking the inverse we get:

$$S_{\phi\phi}^{-1} = (L^{-1})^T D^{-1} L^{-1} = \mathcal{U} \mathcal{D} \mathcal{U}^T$$

which gives the parameter matrix and loss function matrix of Theorem 2. Thus the batch δ -MMLS algorithm can be implemented as in Table 4.1.

Remark 10 Table 4.1 is only appropriate for off-line applications because of the need to find the inverse of the matrix L .

Remark 11 Other decomposition techniques such as Cholesky, LU and QR can be used to obtain the upper triangular parameter matrix $\mathcal{U}$ and the diagonal loss function matrix $\mathcal{D}^{-1}$. The conversion procedures required for such decompositions can be found in (Niu et al., 1995b).

Remark 12 Since MMLS technique is essentially order-recursive LS all the properties of classical LS apply.

4.3 Recursive Multiple Model Least-Squares in the δ -domain

As new measurements are made, the ACM is updated recursively using the following covariance update (see Appendix D):

$$\rho P_{n+1} = \frac{1}{1 - h\alpha} \left(\alpha P_{n+1} - \frac{P_{n+1} \phi \phi^T P_{n+1}}{1 + h \phi^T P_{n+1} \phi} \right). \quad (4.4)$$

However, this update is numerically ill-conditioned, and implementation using this equation directly can lead to the ACM $\mathbf{P}_{n+1}$ losing positive definiteness and then to inaccurate results. Since $\mathbf{P}_{n+1}$ must be positive definite, the common solution is to factorise the ACM and to update the factors rather than the full matrix. One approach is to perform the $\mathbf{UDU}^T$ factorisation (Bierman, 1977):

$$\mathbf{P}_{n+1} = \mathbf{U}\mathbf{D}\mathbf{U}^T,$$

where $\mathbf{D}$ is a diagonal matrix of inverse LS losses and $\mathbf{U}$ is a unit upper-triangular matrix containing the LS parameters (as in the batch case). Then the ACM update of eqn.(4.4) can be rewritten as follows:

$$\rho(\mathbf{U}\mathbf{D}\mathbf{U}^T) = \frac{1}{1-h\alpha} \mathbf{U} \left(\alpha \mathbf{D} - \frac{\mathbf{g}\mathbf{g}^T}{1+h\mathbf{f}^T\mathbf{g}} \right) \mathbf{U}^T, \quad (4.5)$$

where $\mathbf{f} = \mathbf{U}^T\boldsymbol{\phi}$ and $\mathbf{g} = \mathbf{D}\mathbf{f}$.

Now, if the term in brackets can be decomposed in the $\mathbf{UDU}^T$ form, then the ACM update is simply

$$\rho(\mathbf{U}\bar{\mathbf{D}}\mathbf{U}^T) = (\mathbf{U}\mathbf{U}) \left(\frac{\alpha}{1-h\alpha} \mathbf{D} \right) (\mathbf{U}\mathbf{U})^T.$$

where $\bar{\mathbf{D}} = \mathbf{D}/\alpha$. Thus the main problem is to factorise

$$\bar{\mathbf{D}} - \frac{\mathbf{g}\mathbf{g}^T}{1+h\alpha\mathbf{f}^T\mathbf{g}} \text{ into } \mathbf{U}\mathbf{D}\mathbf{U}^T,$$

which is a problem of *rank one modification of the Cholesky factorisation*. There are a variety of methods that can be applied, and selecting an appropriate one depends on whether the matrices involved are positive definite, singular or near singular (Gill et al., 1974). Below we consider three suitable techniques.

Agee-Turner factorisation update. The factors $\mathbf{U}$ and $\mathbf{D}$ can be calculated as suggested by Agee and Turner (1976) (see also (Bierman, 1977)). This method is numerically stable if the matrix to be decomposed is positive definite and not near singular. However the elements of $\mathbf{D}$ are computed as differences, and therefore are susceptible to rounding errors, leading to loss of accuracy. It is pointed out in (Gill et al., 1974) that these roundoff errors may cause the diagonal elements d_i to become negative so that the update is no

longer positive definite. For such problems more stable algorithms have been derived, such as those based on orthogonal transformations (c.f. the Householder update below).

Householder matrices update. In this method, the factorisation is performed using Householder matrices (Gill et al., 1974). The resulting matrix is positive definite regardless of any numerical errors made. The most important difference from the Agee-Turner method is the alternative expression for calculating the diagonal matrix update which guarantees that the computed d_i 's can never become negative. The algorithm is more computationally demanding as more multiplications and square roots are needed, but it always yields a positive definite factorisation even under extreme circumstances, such as the case involving decomposition of a near singular matrix.

Bierman's recursive UD filter. This is a special numerically stable variant of the Agee-Turner algorithm developed originally for the discrete-time RLS algorithm (Bierman, 1976, 1977). We now extend this UD filter for implementation of MMRLS in the δ -domain due to its good numerical properties and efficient computation. Converting eqn.(4.4) to discrete-time we get the following recursive ACM update

$$\mathbf{P}_{n+1}(t+h) = \frac{1}{\beta} \left(\mathbf{P}_{n+1}(t) - \frac{\mathbf{P}_{n+1}(t)\phi(t)\phi^T(t)\mathbf{P}_{n+1}(t)}{1/h + \phi^T(t)\mathbf{P}_{n+1}(t)\phi(t)} \right), \quad (4.6)$$

where $\beta = (1 - h\alpha)$ is the δ -domain forgetting-factor. Thus, to obtain the multiple order model parameter estimates and the associated loss functions, the UD filter of Bierman (1977) is used to update the $\mathbf{U}$ and $\mathbf{D}$ factors of $\mathbf{P}_{n+1}$ directly, i.e.

$$\mathbf{U}(t+h)\mathbf{D}(t+h)\mathbf{U}^T(t+h) = \frac{1}{\beta} \mathbf{U}(t) \left(\mathbf{D}(t) - \frac{\mathbf{g}\mathbf{g}^T}{1/h + \mathbf{f}^T\mathbf{g}} \right) \mathbf{U}^T(t),$$

where $\mathbf{f} = \mathbf{U}^T(t)\phi(t)$ and $\mathbf{g} = \mathbf{D}(t)\mathbf{f}(t)$.

The resulting δ -MMRLS update is as shown in Table 4.2, where matrices $\bar{\mathbf{U}} = \mathbf{U}(t)$, $\hat{\mathbf{U}} = \mathbf{U}(t+h)$, $\bar{\mathbf{D}} = \mathbf{D}(t)$, $\hat{\mathbf{D}} = \mathbf{D}(t+h)$, and where the vectors $\bar{\mathbf{u}}^{(j)}$, $\hat{\mathbf{u}}^{(j)}$, $\bar{\mathbf{d}}^{(j)}$ and $\hat{\mathbf{d}}^{(j)}$ represent the j^{th} column of each matrix respectively.

Table 4.2: The δ -MMRLS time update

$\mathbf{f} = \bar{\mathbf{U}}^T \phi(t)$	Prediction errors (negated)
$\mathbf{v} = \bar{\mathbf{D}} \mathbf{f}$	
$\beta = 1 - h\alpha$	Forgetting-factor
$s_0 = 1/h$	
for $j = 1$ to $n + 1$	From 0 to n parameter model
$s_j = s_{j-1} + f_j v_j$	
$\mathbf{g} = [v_1 \ v_2 \ \dots \ v_{j-1} \ \underbrace{0 \ \dots \ 0}_{n-j+2}]^T$	(first $j - 1$ elements of $\mathbf{v}$)
$\mathbf{k} = \bar{\mathbf{U}} \mathbf{g} / s_{j-1}$	Kalman gain
$\hat{\mathbf{u}}^{(j)} = \bar{\mathbf{u}}^{(j)} - \mathbf{k} f_j$	Parameter update
$\hat{\mathbf{d}}^{(j)} = \bar{\mathbf{d}}^{(j)}(s_{j-1}/s_j)/\beta$	Loss function update
end	

Remark 13 Notice the appealing similarity between the parameter update for order j of this δ -MMRLS algorithm and the classical (discrete-time) RLS parameter update.

Remark 14 The UD factorisation used in the δ -MMRLS algorithm guarantees that the ACM remains positive definite and thus this implementation is numerically superior to the δ -RLS equations given in Table D.1.

Remark 15 This is a discrete-time algorithm and is only valid for $h > 0$ although h can be made arbitrarily small to approximate continuous-time results. For $h = 0$ the ACM update of eqn.(4.6) should be replaced by the eqn.(4.5) to compute the $\mathbf{U}$ and $\mathbf{D}$ factors in continuous-time, i.e.

$$\frac{d}{dt}(\mathbf{U} \mathbf{D} \mathbf{U}^T) = \alpha \mathbf{U} (\mathbf{D} - \mathbf{g} \mathbf{g}^T) \mathbf{U}^T.$$

Remark 16 The UD factorisation is closely related to the Levinson algorithm (Söderström and Stoica, 1989) for the simultaneous estimation of multiple order AR models. However,

the MMLS technique is not restricted to the AR case. Besides, the Levinson method breaks down when the continuous-time process is sampled at fairly rapid rates (Vijayan et al., 1991).

4.4 Estimation of multiple order δ -domain ARX models

Let us approximate a Constant-Coefficient Linear Differential Equation model:

$$(s^{n_a} + a_{n_a-1}s^{n_a-1} + \dots + a_1s + a_0)y(t) = (b_{n_b}s^{n_b} + b_{n_b-1}s^{n_b-1} + \dots + b_1s + b_0)u(t),$$

by the ARX δ -operator model

$$(\delta^{n_a} + a_{n_a-1}\delta^{n_a-1} + \dots + a_1\delta + a_0)y(t) = (b_{n_b}\delta^{n_b} + b_{n_b-1}\delta^{n_b-1} + \dots + b_1\delta + b_0)u(t),$$

where $n_a \geq n_b$ is the model order, $u(t)$ and $y(t)$ are the system input and output respectively. The conventional way of using LS for parameter estimation is to simply move the terms $a_{n_a-1}\delta^{n_a-1}y(t), \dots, a_0y(t)$ to the right-hand side in order to get the standard linear regression model:

$$\bar{y}(t) = \delta^{n_a}y(t) = \boldsymbol{\theta}^T \mathbf{x}(t),$$

where

$$\begin{aligned} \mathbf{x}(t) &= (-\delta^{n_a-1}y(t), \dots, -\delta y(t), -y(t), \delta^{n_b}u(t), \dots, \delta u(t), u(t))^T, \\ \boldsymbol{\theta} &= (a_{n_a-1}, \dots, a_1, a_0, b_{n_b}, \dots, b_1, b_0)^T. \end{aligned}$$

The main idea of the AUDI approach (Niu et al., 1992; Niu and Fisher, 1993) is to move *all* the terms in y to the right-hand side, and to *interleave* the data and the parameters so that taking the case where $n_a = n_b + 1 = m$ (the maximum order) one gets:

$$0 = \boldsymbol{\theta}_{n+1}^T \boldsymbol{\phi}(t),$$

where

$$\begin{aligned} \boldsymbol{\phi}(t) &= (-y(t), u(t), -\delta y(t), \delta u(t), \dots, -\delta^{m-1}y(t), \delta^{m-1}u(t), -\delta^m y(t))^T, \\ \boldsymbol{\theta}_{n+1} &= (a_0, b_0, a_1, b_1, \dots, a_{m-1}, b_{m-1}, 1)^T. \end{aligned}$$

Presenting the input–output data to the MMLS algorithm in this order, the parameters in the columns of the $\mathcal{U}$ matrix correspond to the following relations:

$$\begin{aligned}
 \{\} & \Rightarrow y \\
 \{-y\} & \Rightarrow -u \\
 \{-y, u\} & \Rightarrow \delta y \\
 \{-y, u, -\delta y\} & \Rightarrow -\delta u \\
 & \vdots \\
 \{-y, u, -\delta y, \dots, \delta^{m-2}u\} & \Rightarrow \delta^{m-1}y \\
 \{-y, u, -\delta y, \dots, \delta^{m-2}u, -\delta^{m-1}y\} & \Rightarrow -\delta^{m-1}u \\
 \{-y, u, -\delta y, \dots, \delta^{m-2}u, -\delta^{m-1}y, \delta^{m-1}u\} & \Rightarrow \delta^m y
 \end{aligned}$$

where ‘ $\Rightarrow$ ’ indicates the result of a linear combination of the set of variables on the left-hand side to fit the variable given on the right-hand side. Thus the forward models (cause $\rightarrow$ effect, i.e. the plant) are given in every odd column of the $\mathcal{U}$ matrix (where the variable on the right is a derivative of the output y), and the even columns give the reverse models (cause $\leftarrow$ effect, i.e. any feedback present).

4.5 The forgetting strategy

Since MMRLS gives a BLUE estimate for finite data windows then, unlike the Levinson algorithm, it can be naturally combined with the variable forgetting techniques of chapter 3. Let us write the LS loss function $J(\hat{\theta}(t), t)$ in terms of the prediction errors ϵ :

$$J(t) = \mathcal{S}_0^t E(-\alpha, t - \tau) \epsilon^2(\tau) d\tau. \quad (4.7)$$

Suppose now that estimation is perfect for all t so that $\epsilon(t) = e(t)$ always and the variance of the noise is σ_0^2 . Then the expected value of the irreducible or *ideal loss* is given by:

$$J_{opt}(t) = \mathcal{S}_0^t E(-\alpha, t - \tau) \sigma_0^2 d\tau = \frac{\sigma_0^2}{\alpha} \left(1 - E(-\alpha, t)\right). \quad (4.8)$$

For large t this approaches $\sigma_0^2/\alpha = T_c \sigma_0^2$ where $T_c = 1/\alpha$ corresponds to the *asymptotic averaging time*. The value T_c is important as it gives an indication of the length of the past time window containing data most influential in providing the estimate (see Figure 3.1).

Now differentiating eqn.(4.7) with respect to time, we obtain the following:

$$\rho J(t) + \alpha J(t) = (1 - h\alpha)\epsilon^2(t),$$

which shows that the loss tends to zero at a rate determined by the exponential forgetting-factor α , and is driven away from zero by the current prediction error $\epsilon(t)$. We would expect the average value of ϵ^2 to be greater than σ_0^2 since $\epsilon(t) = \tilde{\theta}^T(t)x(t) + e(t)$ where $\tilde{\theta}(t) = \theta(t) - \hat{\theta}(t)$ is the modelling error. In particular when the estimate is far from the true value of $\theta(t)$ we expect that $\epsilon^2(t) \gg \sigma_0^2$. One way to proceed is to assert that the value of J contains ‘information’ on the parameter estimate and should be kept constant in the steady-state (i.e. $\rho J = 0$) and thus $J_{ss} = (1 - h\alpha)\epsilon^2/\alpha$. Hence a good value for the exponential forgetting-factor α is obtained by setting J_{ss} equal to the steady-state value of the ideal loss function, i.e.

$$J_{ss} = (1 - h\alpha(t))\epsilon^2(t)/\alpha(t) = T_c\sigma_0^2,$$

giving

$$\alpha(t) = \frac{\epsilon^2(t)}{T_c\sigma_0^2 + h\epsilon^2(t)}. \quad (\text{Variable forgetting-factor})$$

The multiple correlation coefficients can be easily obtained from the δ -MMRLS algorithm by computing:

$$R_y(i, t) = \sqrt{1 - J_i(t)/J_0(t)} = \sqrt{1 - d_0(t)/d_i(t)}$$

where $J_0(t) = \mathcal{S}_0^t e^{-\alpha(t-\tau)} y^2(\tau) d\tau$. The best-fit model has the largest $\mathcal{D}$ entry and thus a good measure of model quality is given by:

$$Q(t) = \sqrt{1 - d_0(t)/d_{max}(t)}, \quad (\text{Model quality})$$

where $Q = 1$ indicates perfect modelling and $Q \rightarrow 0$ indicates decreasing confidence in the model. Typically $Q > 0.9$ which implies that the estimated model explains at least 90% of the output signal.

Now, if the data vector $\phi(t)$ is further augmented with a 1 in the first position, i.e. $\phi^T(t) = (1 \ \phi^T(t))^T$, then the 0th order model is $0 = \hat{\theta}_0\phi_1^T(t) + \epsilon_0(t)$. Since the Lagrange approach

constrains $\hat{\theta}_0 = 1$ for the 0^{th} order model, we have that $0 = 1 \times 1 + \epsilon_0$, and hence the first loss function entry obtained from the δ -MMRLS algorithm is given by:

$$J_0(t) = 1/d_0(t) = \mathcal{S}_0^t E(-\alpha, t - \tau) (-1)^2 d\tau = \frac{1}{\alpha} (1 - E(-\alpha, t)), \quad (4.9)$$

where $d_0(t)$ is the first element of the $\mathcal{D}$ matrix. Therefore $\lim_{t \rightarrow \infty} J_0(t) = 1/\alpha = T_c$, which is the asymptotic averaging time given a time-varying forgetting-factor α :

$$\hat{T}_c(t) = 1/d_0(t). \quad (\text{Asymptotic averaging time})$$

Note: if the data vector is not augmented with a 1 as the first entry, then the following expression can be used as an estimate of T_c

$$\hat{T}_c(t+h) = (1 - h\alpha(t))\hat{T}_c(t) + h.$$

Also, by dividing eqn.(4.8) by eqn.(4.9), the measurement noise variance can be obtained:

$$\sigma_0^2(t) = J_{opt}(t)/J_0(t),$$

and if we assume that the smallest loss (i.e. of the best fit model) $J_{min}(t)$ is a good estimate of $J_{opt}(t)$ then the best fit model's average residual variance is given by:

$$\bar{\epsilon}_{min}^2(t) = J_{min}(t)/J_0(t) = d_0(t)/d_{max}(t),$$

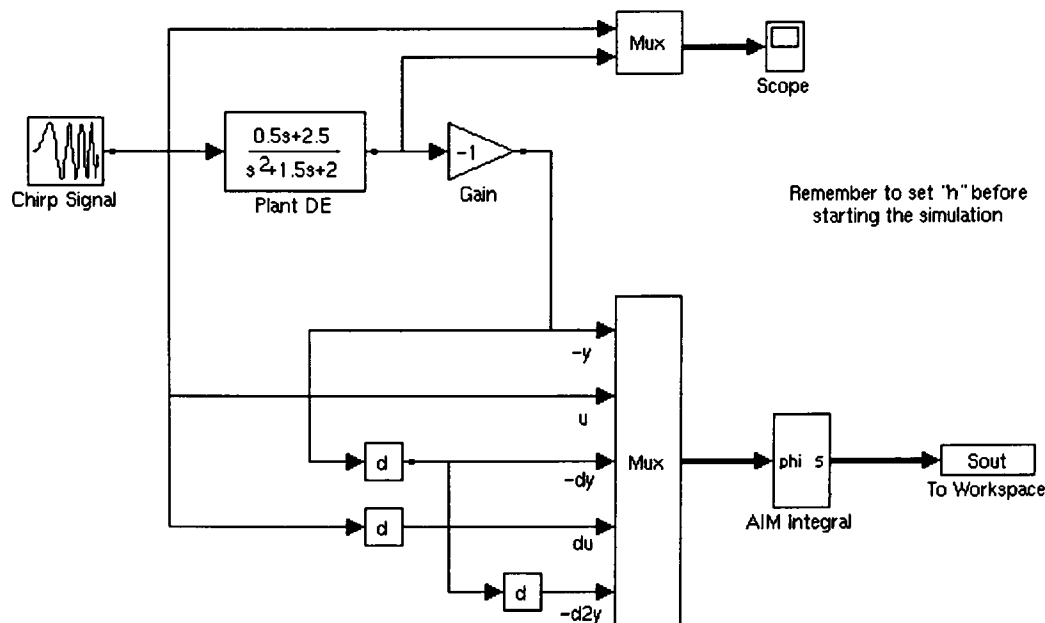
where $d_{max}(t)$ is the largest $\mathcal{D}$ entry. This should tend towards the noise variance σ_0^2 if the model fit is good and thus $\bar{\epsilon}_{min}^2(t)$ can be used to estimate the level of measurement noise present.

4.6 Simulations

4.6.1 The batch case

The batch δ -MMLS algorithm was used to estimate a system described by the following 2^{nd} order linear differential equation:

$$\frac{dy^2}{dt^2} + 1.5\frac{dy}{dt} + 2y = 0.5\frac{du}{dt} + 2.5u,$$

Figure 4.1: SIMULINK block diagram of the δ -MMLS worksheet

where u is the input to the system and y is the output. The system was excited by a chirp signal (which is differentiable up to any order) where the frequency was swept from 0.001 Hz to 0.1 Hz in 100 s (see Figure 4.1). The final AIM $S_{\phi\phi}(100)$ was decomposed into LDL^T form, giving the parameter matrix $\mathcal{U} = (\mathbf{L}^{-1})^T$ and the loss functions $J = \text{diag}(\mathbf{D})$ as shown in Tables 4.3 and 4.4 for sampling times h equal to 0.1 s and 0.01 s respectively. In the tables, ' $\Longleftrightarrow$ ' denotes 'parameter associated with'. The estimated parameters and the loss function for the 1st order model are given in column 3, and those for the 2nd order are contained in column 5. Clearly the 2nd order model fits the data the best due to the small loss function values. Also, since the loss function J_i for any particular model order i decreases with sampling time h , the quality of the estimates improves with faster sampling rates. Indeed, in the continuous-time case (approximated here by $h = 0.0001$ s), the 2nd order estimates approach the true system differential equation coefficients – see the last column of Table 4.5.

Figure 4.2 shows the estimated step responses for the 1st and 2nd order models versus the true model for $h = 0.1$ s. Clearly the 2nd order model fits the true model better than the 1st order model, as indicated by the loss functions of Table 4.3: $J_5 = 9.7 \times 10^{-7}$ and $J_3 = 0.4916$ respectively, which illustrates the ease of model order selection using δ -MMLS.

Table 4.3: Parameter matrix and loss functions for δ -MMLS ($h = 0.1$ s, $\alpha = 0$)

$\mathcal{U} =$	1.0000	0.7575	1.5555	-1.1415	2.1546	$\iff$	y
	0	1.0000	2.0692	-1.4280	2.6934	$\iff$	u
	0	0	1.0000	0.0867	1.3736	$\iff$	δy
	0	0	0	1.0000	0.2355	$\iff$	δu
	0	0	0	0	1.0000	$\iff$	$\delta^2 y$
$J =$	76.5069	2.3477	0.4916	0.0133	0.0000		
					(9.7e-7)		

Table 4.4: Parameter matrix and loss functions for δ -MMLS ($h = 0.01$ s, $\alpha = 0$)

$\mathcal{U} =$	1.0000	0.7577	1.5695	-1.2145	2.0153	$\iff$	y
	0	1.0000	2.0790	-1.5188	2.5191	$\iff$	u
	0	0	1.0000	0.0410	1.4881	$\iff$	δy
	0	0	0	1.0000	0.4747	$\iff$	δu
	0	0	0	0	1.0000	$\iff$	$\delta^2 y$
$J =$	76.4731	2.3418	0.4322	0.0152	0.0000		
					(3.3e-8)		

Table 4.5: Parameter matrix and loss functions for continuous-time MMLS ($\alpha = 0$)

$\mathcal{U} =$	1.0000	0.7577	1.5710	-1.2231	2.0002	$\iff$	y
	0	1.0000	2.0800	-1.5295	2.5002	$\iff$	u
	0	0	1.0000	0.0356	1.4999	$\iff$	δy
	0	0	0	1.0000	0.4997	$\iff$	δu
	0	0	0	0	1.0000	$\iff$	$\delta^2 y$
$J =$	76.4693	2.3412	0.4259	0.0154	0.0000		
					(2.9e-10)		

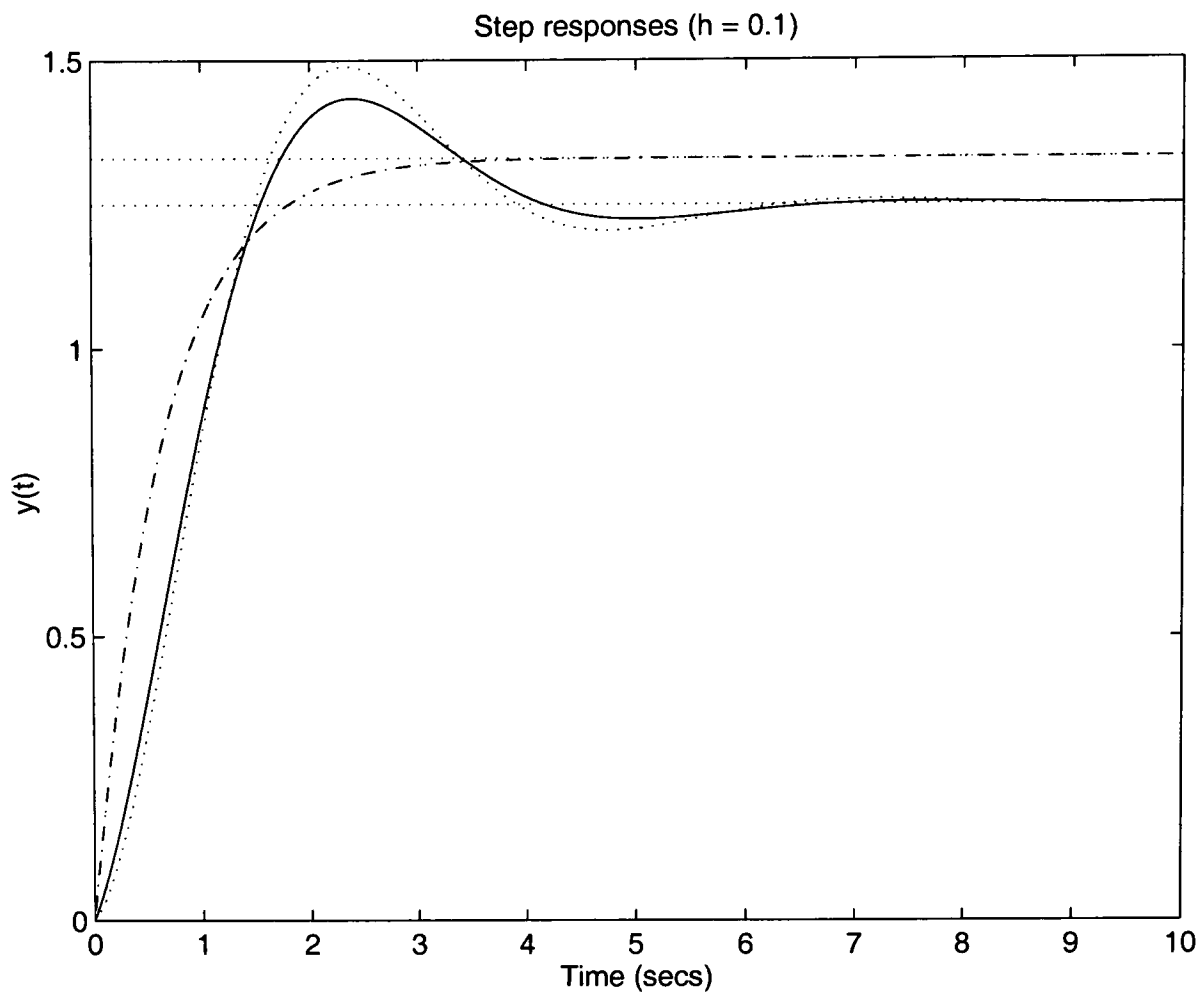


Figure 4.2: Step responses for 1st (dot-dash line) and 2nd order (dotted line) δ -MMLS estimated models versus the true model (solid line)

4.6.2 The recursive case

The δ -MMRLS algorithm was tested using SIMULINK with worksheets shown in Figures 4.3 and 4.4. The d-MMRLS block in Figure 4.3 was implemented as a C code MEX-file S-function due to the matrix manipulations involved and was tuned with ideal steady-state loss function $J_{ss} = T_c \sigma_0^2 = 0.1$. The 4th order Butterworth low-pass filters were used to ensure that the data was smooth and differentiable up to order 3 (the maximum plant order), as well as to weight the LS fit to low frequencies. The cutoff frequency of 2 rad/s was chosen in accordance to the expected bandwidth of the plant (for the 2nd order nominal model, $\omega_b \approx \omega_n = \sqrt{2}$ rad/s).

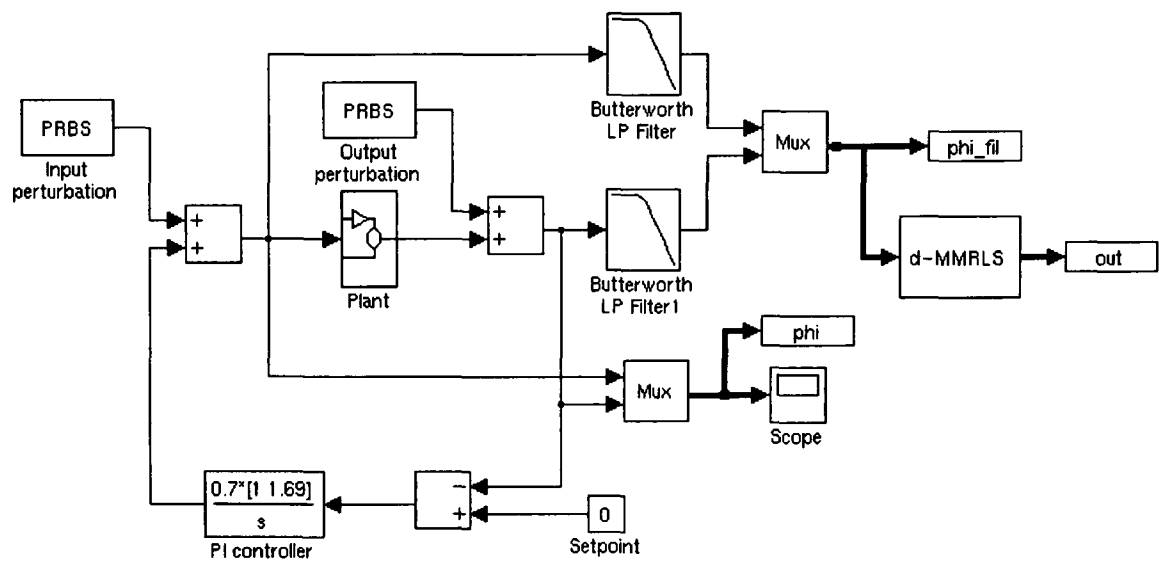


Figure 4.3: SIMULINK block diagram of the δ -MMRLS worksheet

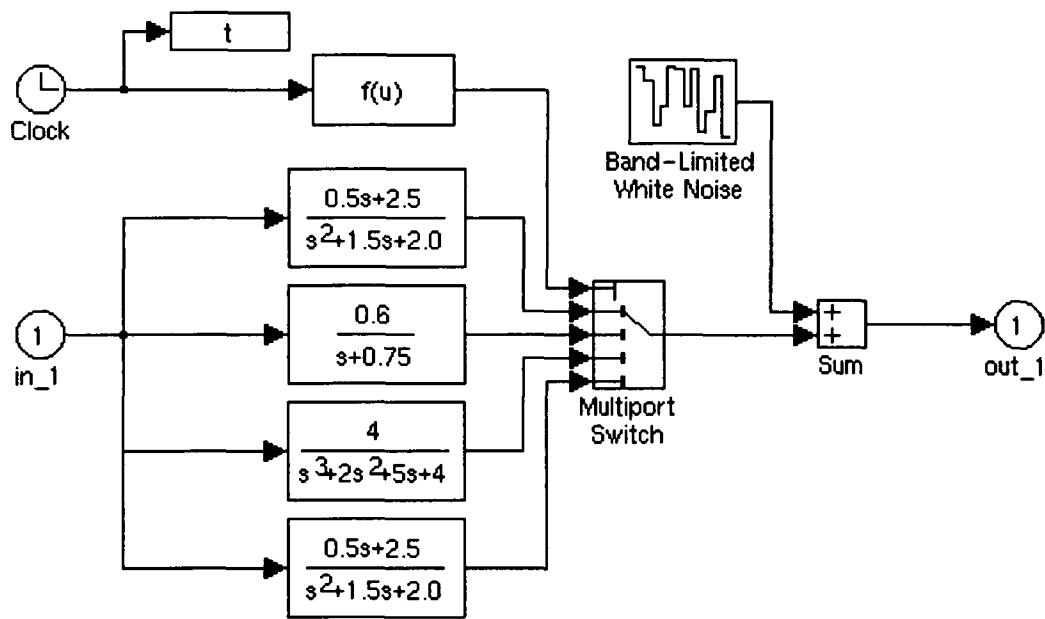


Figure 4.4: SIMULINK block diagram of the time-varying, continuous-time plant

Open-loop simulation tests

In these two simulation runs, the system was in open loop using a 15-bit PRBS sequence of unit amplitude to excite the time-varying plant. The first run had no output noise and the second included a zero mean white noise sequence with variance $\sigma_0^2 = 0.01$ added to the plant output.

The estimated parameters and their associated losses after 99 seconds are given in Table 4.6 (no noise) and Table 4.7 (with noise). From these two tables, one sees that the parameters in column 5 correspond to the lowest loss function value of $J_5(99) = 0.000314$ in the noise free case, and $J_5(99) = 0.0445$ with noise, and thus the best fit models at $t = 99$ s are given by:

$$(\delta^2 + 1.49\delta + 2.02)y(t) = (0.475\delta + 2.53)u(t) \quad (\text{no noise})$$

$$(\delta^2 + 1.49\delta + 2.02)y(t) = (0.474\delta + 2.52)u(t) \quad (\text{with noise}),$$

both of which compare favourably with the batch δ -MMLS estimate (noise free, with no forgetting) of

$$(\delta^2 + 1.49\delta + 2.02)y(t) = (0.475\delta + 2.52)u(t),$$

and as to be expected, the estimated δ -domain coefficients are very close to the continuous-time plant differential equation:

$$(s^2 + 1.5s + 2)y(t) = (0.5s + 2.5)u(t).$$

At time $t = [100, 200, 300]$ s, the plant changes and Tables 4.8 and 4.9 give the parameters and the loss functions for the estimated models just before each plant change, with and without noise respectively. Clearly the variable forgetting strategy enables the algorithm to track changes in the plant for both order and parameter changes. The parameter coefficients are also seen to approximate the continuous-time plant quite closely for a sampling rate of $h = 0.01$ s.

Table 4.6: Parameter matrix and loss functions for at $t = 99$ s (Open-loop without noise, $h = 0.01$ s)

$\mathcal{U} =$	1	0.516	0.746	-1.52	2.02	-807	-245	$\iff$	y
	0	1	1.46	-1.9	2.53	-1.01e3	-306	$\iff$	u
	0	0	1	-0.774	1.49	-597	-179	$\iff$	δy
	0	0	0	1	0.475	-192	-55.7	$\iff$	δu
	0	0	0	0	1	-400	-120	$\iff$	$\delta^2 y$
	0	0	0	0	0	1	0.779	$\iff$	$\delta^2 u$
	0	0	0	0	0	0	1	$\iff$	$\delta^3 y$
$J =$	87.1	32.2	17	21.4	0.000314	0.00279	0.00099		

Table 4.7: Parameter matrix and loss functions at $t = 99$ s (Open-loop with noise, $h = 0.01$ s)

$\mathcal{U} =$	1	0.513	0.743	-1.51	2.02	2.66	-0.0691	$\iff$	y
	0	1	1.45	-1.9	2.52	4.2	-0.0862	$\iff$	u
	0	0	1	-0.773	1.49	0.745	1.97	$\iff$	δy
	0	0	0	1	0.474	-1.6	2.51	$\iff$	δu
	0	0	0	0	1	-0.0914	1.46	$\iff$	$\delta^2 y$
	0	0	0	0	0	1	0.472	$\iff$	$\delta^2 u$
	0	0	0	0	0	0	1	$\iff$	$\delta^3 y$
$J =$	72.8	27.1	14.3	17.9	0.0445	42.1	0.325		

Table 4.10: Final parameter matrix and loss functions (Closed-loop with input excitation, $h = 0.01$ s)

$\mathcal{U} =$	1	-0.00533	-0.00457	-1.94e-4	0.00299	-0.00238	5.35e-4	3.14e-5	$\iff$	1
	0	1	0.144	0.195	-1.72	2.00	-0.675	-0.0299	$\iff$	y
	0	0	1	1.46	-2.16	2.50	2.03	0.0269	$\iff$	u
	0	0	0	1	-1.33	1.47	-0.592	2.00	$\iff$	δy
	0	0	0	0	1	0.482	-2.46	2.47	$\iff$	δu
	0	0	0	0	0	1	-1.91	1.44	$\iff$	$\delta^2 y$
	0	0	0	0	0	0	1	0.495	$\iff$	$\delta^2 u$
	0	0	0	0	0	0	0	1	$\iff$	$\delta^3 y$
$J =$	121	44.5	44.1	12.4	25.3	0.295	52.3	1.26		

Closed-loop test with input perturbation

Figure 4.5 shows the trajectories of the estimated parameters, loss functions and estimated model order for the system in closed-loop with a Proportional plus Integral (PI) controller and a 15-bit PRBS input perturbation signal. The first sub-figure shows the parameter trajectories of the best-fit model of order shown in the sub-figure immediately below where the dotted lines mark the true model order. Clearly the δ -MMRLS algorithm manages to track the plant changes successfully using a variable forgetting strategy which tries to keep the amount information, i.e. the LS loss function, constant and equal to $J_{ss} = 0.1$ in this case – as can be seen in the graph of J_{min} which tends to 0.1, as required. The on-line estimate of the model quality Q gives a measure of the goodness-of-fit of the best identified model at each time instant. As is to be expected, the value of Q drops sharply at each plant change (corresponding to a small T_c and large forgetting-factor β) and then works its way back towards unity as more and more information is accumulated for providing the estimate (which is indicated by increasing T_c and $\beta \rightarrow 1$).

Table 4.10 shows the parameter matrix and loss functions at the end of the input perturbation closed-loop simulation run. At the end of the simulation run, the minimum loss function $J_{min}(499) = 0.2946$ and the estimated asymptotic sample length $\hat{T}_c(499) = 120.9$ s,

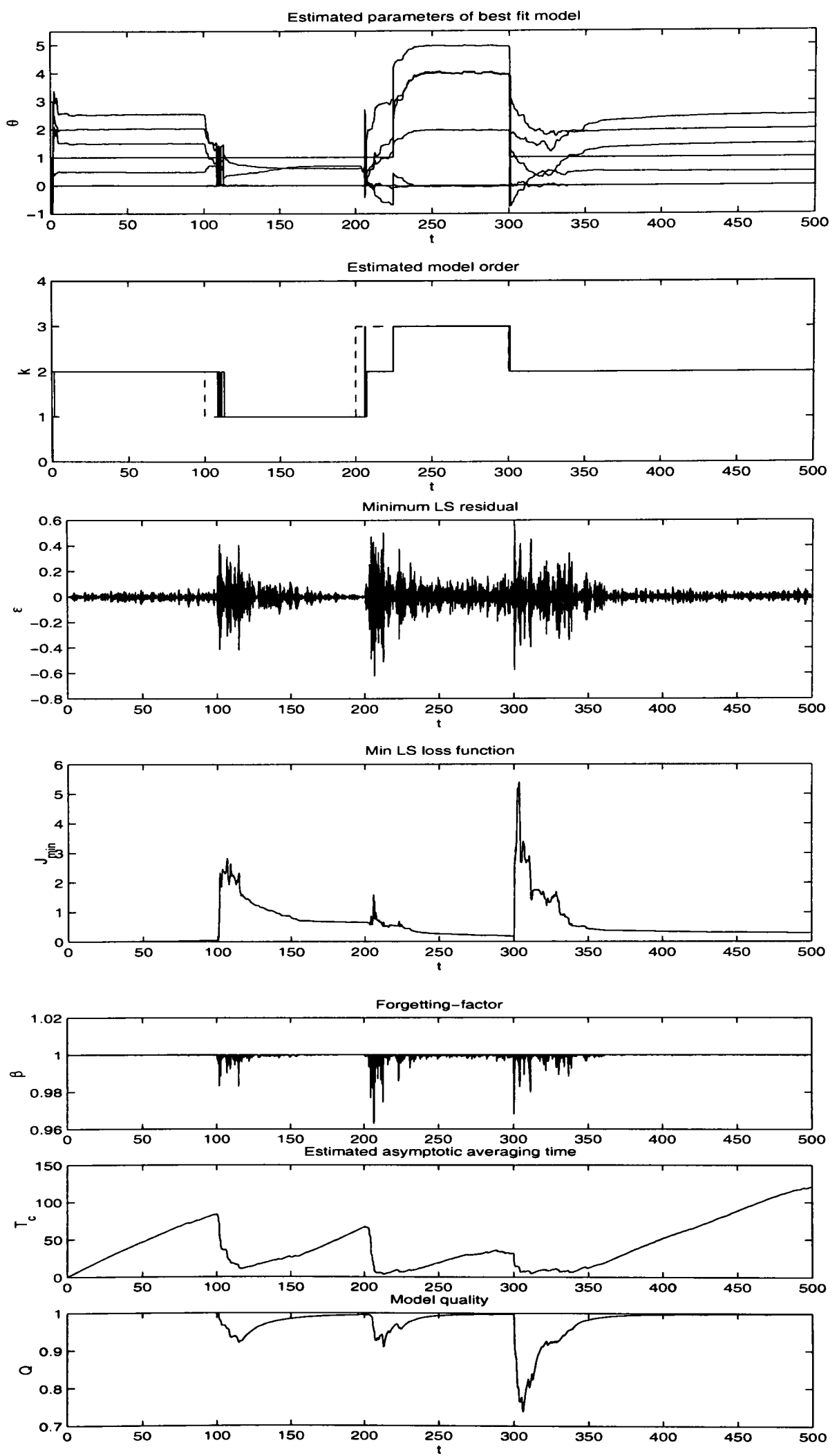


Figure 4.5: Simulation results for δ -MMRLS in closed-loop with input excitation

giving $Q(499) = 0.9988$ implying that the final model:

$$(\delta^2 + 1.47\delta + 2.00)y(t) = (0.482\delta + 2.50)u(t) - 0.00238$$

is very good indeed. In addition the estimated δ -domain model coefficients closely match the continuous-time differential equation coefficients at $t = 499$ s

$$(s^2 + 1.5s + 2)y(t) = (0.5s + 2.5)u(t)$$

and thus it is easy to estimate physical parameters such as the damping factor and natural frequency of the second order system using δ -MMRLS.

4.7 Conclusions

This chapter has presented the MMLS technique in the δ -domain for both batch and on-line applications. The δ -MMLS method is seen to be appealing for parameter estimation as it:

- is order recursive, providing the highest order LS model estimates together with all the low order model estimates as a by-product;
- is not restricted by AR models as is the case for the Levinson-type algorithms of (Vijayan et al., 1991; Goodwin et al., 1992);
- automatically provides loss functions for each order model, allowing model order selection;
- is inherently numerically stable due to the UD factorisation involved;
- is based on the δ -domain LS and RLS algorithms of Middleton and Goodwin (1990) which have good numerical and convergence properties when implemented on restricted word length platforms employing fast sampling (Middleton and Goodwin, 1990; Gevers and Li, 1993).

Indeed, the simulation results have shown that the best-fit model coefficients (i.e. the model with the lowest loss function value) using the δ -MMLS technique approach those of the true

continuous-time system as the sample time tends to zero. Thus the δ -MMLS method can be used to obtain the order and the coefficients of ARX plant models for any continuous-time and/or δ -operator based MBPC as well as enabling the user to monitor and bound those coefficients in the estimated model that have some physical significance.

Chapter 5

Application of multiple model based predictive control

In this chapter the MMLS technique is coupled with GPC to provide an adaptive, multiple model based prediction control scheme which is used to control the levels of two coupled tanks using two pumps whilst constraining the level of the third tank to remain within some specified range of levels. The main objective is to investigate the performance of the multiple model GPC (MMGPC) compared to fixed model GPC. Since MMGPC is adaptive by nature the comparison is two-fold: firstly the benefit of adaptation for overcoming plant non-linearity is explored, and secondly the benefits of choosing the model order on-line is examined with respect to overcoming the problems associated with under/over-parameterisation of the plant model. It should be noted that the choice of coupled tanks is not strictly suited to effect the latter comparison because there is no exact linear parameterisation due to the non-linearities inherent in the coupled tank system. However almost all processes in the real world display non-linearities and thus it is still interesting to investigate the behaviour of on-line model order selection in an adaptive predictive control scheme.

5.1 The coupled tanks experimental rig

The coupled tanks system is shown in Figure 5.1. The three tanks will be referred to as Tank 1 (left), Tank 2 (right) and Tank 3 (centre). The voltage to the two pumps can be used to control the flow of water into the left and right tanks, and there are three differential

pressure sensors to measure the water levels in each tank. The outlet valves at the bottom of each tank (labelled Valve 1, Valve 2 and Valve 3 respectively) can be used to change the dynamics of each tank, and the interconnecting valves (labelled Valve 13 and Valve 23) can be used to alter the amount of coupling between tanks 1 and 3, and tanks 2 and 3 respectively.

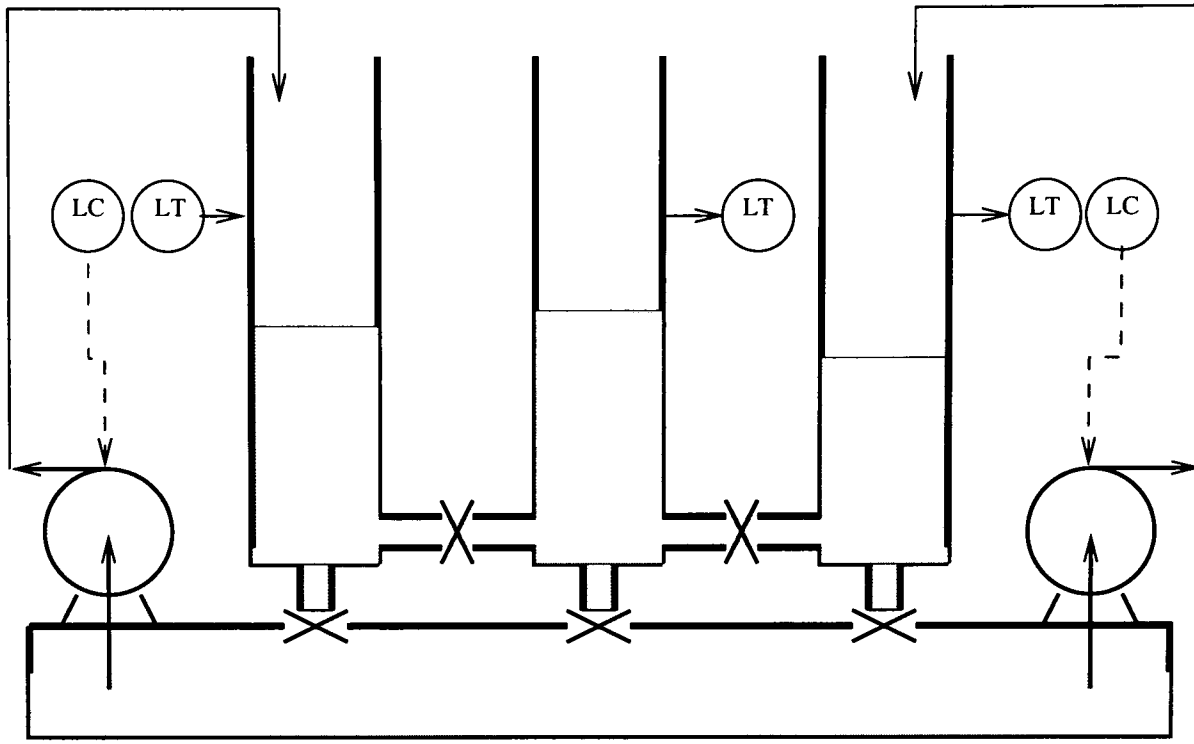


Figure 5.1: The coupled tanks apparatus

Using the derivation of Appendix E and ignoring the fluid resistances of the pipes, the behaviour of the coupled tank system relating the pump input mass flow rates to the tank levels is described by the following set of coupled, non-linear differential equations:

$$\dot{h}_1 = \frac{1}{A_1} \left(F_1 - a_1 \sqrt{2gh_1} - \text{sign}(h_1 - h_3) a_{13} \sqrt{2g|h_1 - h_3|} \right) \quad (5.1)$$

$$\dot{h}_2 = \frac{1}{A_2} \left(F_2 - a_2 \sqrt{2gh_2} - \text{sign}(h_2 - h_3) a_{23} \sqrt{2g|h_2 - h_3|} \right) \quad (5.2)$$

$$\begin{aligned} \dot{h}_3 = \frac{1}{A_3} \left(-a_3 \sqrt{2gh_3} - \text{sign}(h_3 - h_1) a_{13} \sqrt{2g|h_3 - h_1|} \right. \\ \left. - \text{sign}(h_3 - h_2) a_{23} \sqrt{2g|h_3 - h_2|} \right) \end{aligned} \quad (5.3)$$

where g is the acceleration due to gravity in m/s^2 ; F_1 and F_2 are the input volume flow rates in m^3/s ; h_1 , h_2 and h_3 are the tank levels in m ; a_1 , a_2 and a_3 are the drain valve cross-sectional areas in m^2 with a_{13} and a_{23} the interconnecting valve areas in m^2 ; and $A_1 = A_2 = A_3 = A$ is the cross-sectional area of each tank in m^2 .

The dSPACE Real-Time Interface (RTI) was used to implement the control strategy on the tank system. A schematic of the dSPACE RTI is shown in Figure 5.2. The controllers were designed and tested using MATLAB/SIMULINK which provided a user-friendly design environment. The final controller (see Figure 5.3) was then compiled and loaded onto the Texas Instruments (TI) TMS320C31 Digital Signal Processing (DSP) chip using the dSPACE RTI which first converts the SIMULINK worksheet into C source code and then compiles the C source using the TI C compiler. Data visualisation and logging was achieved using the dSPACE real-time TRACE tool. The dSPACE COCKPIT tool was then used as a console for the user to modify various parameters corresponding to the SIMULINK worksheet on-line.

The COCKPIT window in Figure 5.4 shows the console developed for the coupled tanks experiment where different control methods can be selected by clicking on the appropriate radio buttons. For example, simple PI control can be achieved by selecting the “GPC+PI” Control button (i.e. cascaded GPC and PI) with the GPC outer loop running in “O/L manual control” Controller mode. The setpoints can either be generated manually by selecting the “Man” Input button and dragging the “SP1” and “SP2” slider bars, or automatically by selecting “Auto” which generates random size step changes between “Amin” and “Amax” at random times as shown in the Trace Plots window. The levels of each tank are shown by the bar instruments “PV1”, “PV2” and “PV3”.

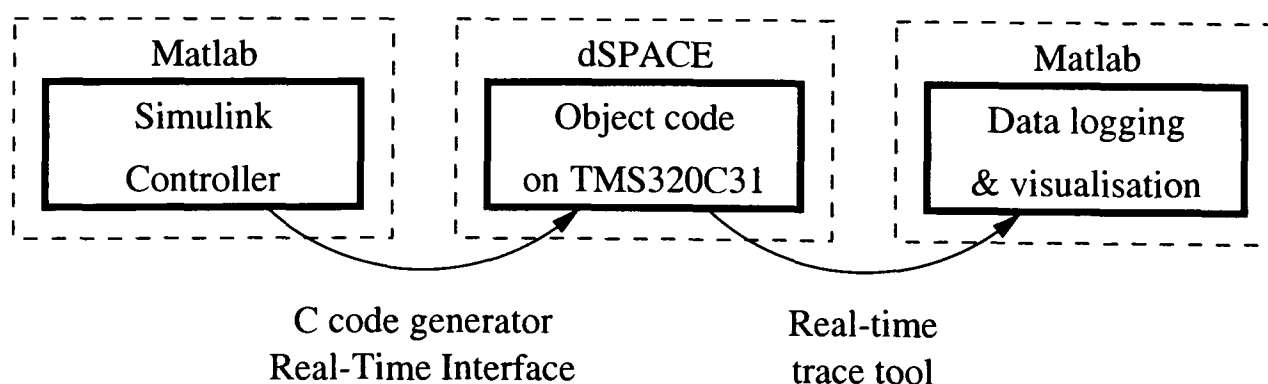


Figure 5.2: The dSPACE Real-Time Interface (RTI)

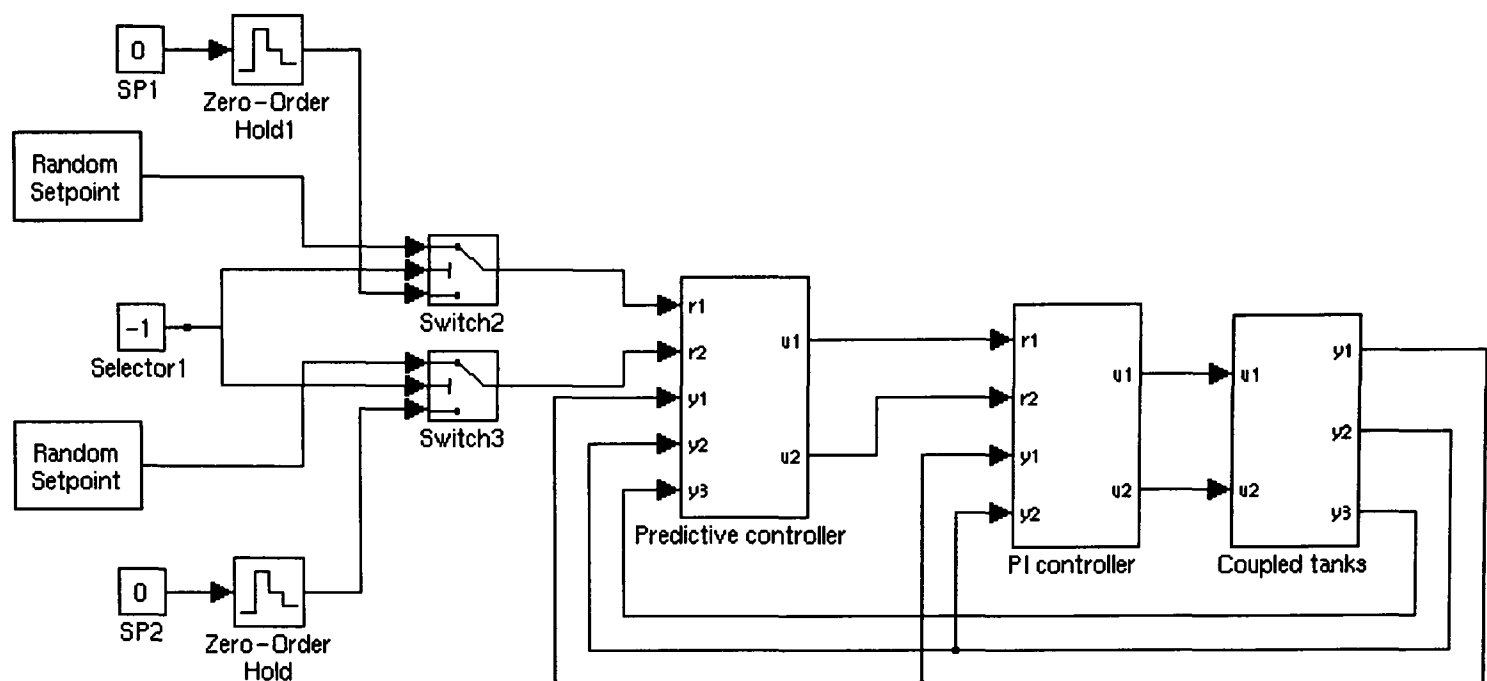


Figure 5.3: The SIMULINK worksheet for Real-Time control of the coupled tank

5.2 The control law

In this section an adaptive multivariable constrained predictive control scheme is developed. The predictive control scheme minimises a multivariable GPC cost function (Mohtadi et al., 1986, 1987) where the predictions involved are generated by a set of Multi-Input, Single-Output (MISO) CARIMA models, one for each output. The model coefficients are then obtained by converting the MISO CARIMA models to ARX form (by filtering the input and output sequences by a suitable band-pass filter Δ/T) and then employing the MMRLS algorithm of chapter 2 where Akaike's information theoretic criterion (Akaike, 1977) is used to select a parsimonious model from the set of multiple order models. The resulting constrained optimisation problem is then solved using the Mixed Weights Least-Squares (MWLS) algorithm of Lawson and Hanson (1974) using a formulation first proposed in (Rossiter and Kouvaritakis, 1993).

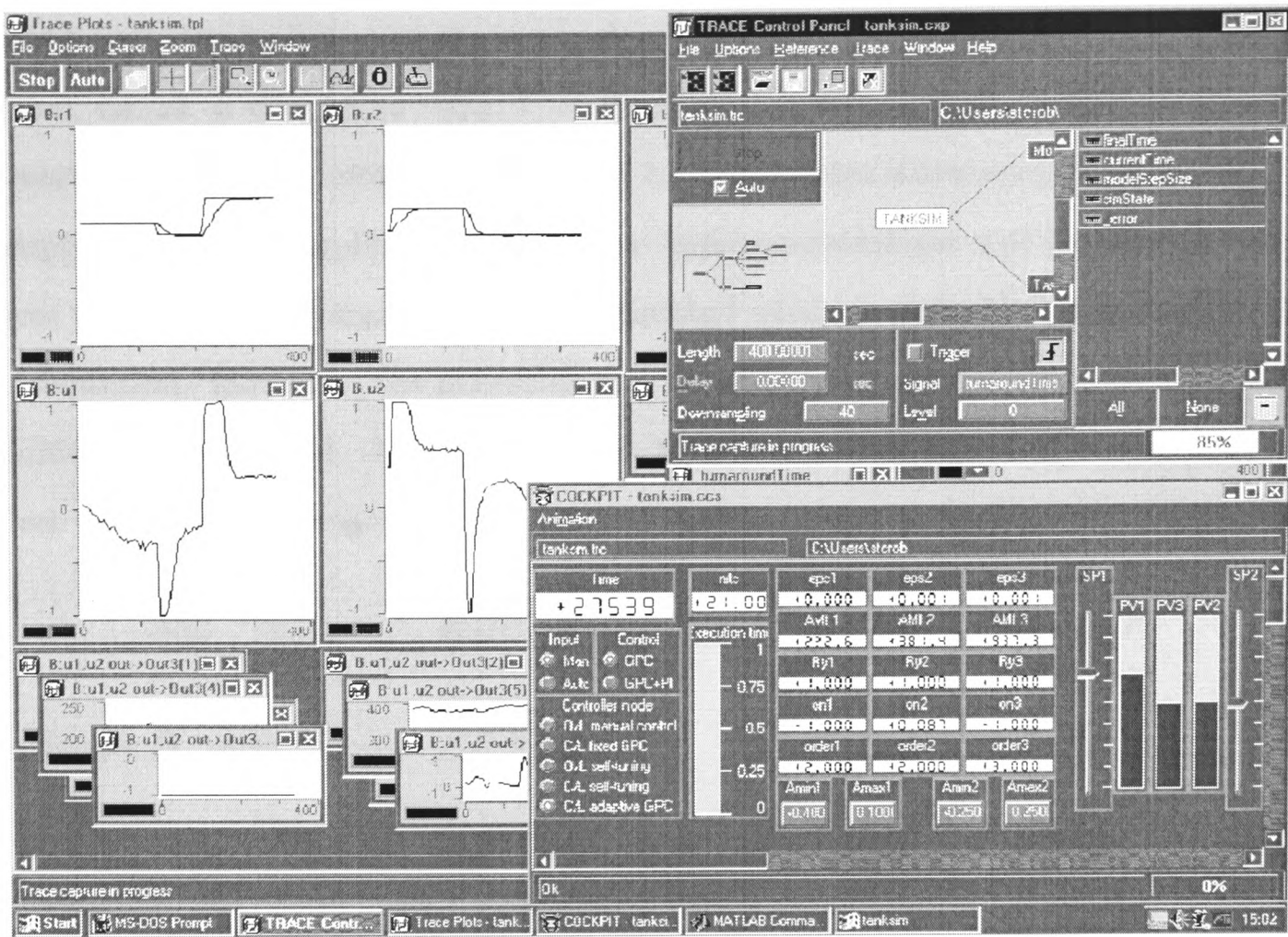


Figure 5.4: The COCKPIT console and TRACE plots for Real-Time control of the coupled tank

5.2.1 Multivariable modelling and predictions

Consider the Multi-Input, Multi-Output (MIMO) CARIMA model of a square $(p \times p)$ multivariable plant,

$$\mathcal{A}(q^{-1}) \mathbf{y}(t) = \mathcal{B}(q^{-1}) \mathbf{u}(t-1) + \frac{\mathcal{T}(q^{-1})}{\Delta} \boldsymbol{\xi}(t) \quad (5.4)$$

where $\mathcal{A}(q^{-1}) = (A_{ij}(q^{-1}))$, $\mathcal{B}(q^{-1}) = (B_{ij}(q^{-1}))$ and $\mathcal{T}(q^{-1}) = (T_{ij}(q^{-1}))$ are polynomial matrices of order p satisfying $\mathcal{A}(0) = \mathbf{I}$ and $\det(\mathcal{A}^{-1}(1)\mathcal{B}(1)) \neq 0$;

$$\mathbf{y}(t) = [y_1(t) \ y_2(t) \ \cdots \ y_p(t)]^T,$$

$$\mathbf{u}(t) = [u_1(t) \ u_2(t) \ \cdots \ u_p(t)]^T,$$

$$\boldsymbol{\xi}(t) = [\xi_1(t) \ \xi_2(t) \ \cdots \ \xi_p(t)]^T,$$

are the output vector, input vector and white noise vector respectively, and $\Delta = (1 - q^{-1})$ is the difference operator. The subscripts denote the *channel* of the corresponding quantities. Process time-delays are represented as leading zeros in $\mathcal{B}$.

In the sequel, the polynomial matrices $\mathcal{A}(q^{-1})$ and $\mathcal{T}(q^{-1})$ are further assumed to be diagonal, i.e. $\mathcal{A} = \text{diag}(A_i(q^{-1}))$; $\mathcal{T} = \text{diag}(T_i(q^{-1})) \forall i \in [1, p]$. Although this is not the most general form, most processes can be represented adequately in this way. The assumption not only simplifies the controller implementation (as will be shown), but also reduces the number of parameters to be estimated which is desirable when employing on-line adaptation. Moreover, the potential gain of using full polynomials matrices is usually insignificant (Tham et al., 1991). Hence the MIMO model of eqn.(5.4) can be considered as a set of p MISO processes:

$$A_i(q^{-1})y_i(t) = \sum_{j=1}^p B_{ij}(q^{-1})u_j(t-1) + \frac{T_i(q^{-1})}{\Delta}\xi_i(t) \quad \forall i \in [1, p] \quad (5.5)$$

If we denote $\mathbf{B}_i$ as the i^{th} row of $\mathbf{B}$ then

$$A_i(q^{-1})y_i(t) = \mathbf{B}_i(q^{-1})\mathbf{u}(t-1) + \frac{T_i(q^{-1})}{\Delta}\xi_i(t) \quad \forall i \in [1, p] \quad (5.6)$$

from which one can obtain the output predictions using the following set of MISO ARX models:

$$A_i(q^{-1}) dy_i^f(t) = \mathbf{B}_i(q^{-1}) d\mathbf{u}^f(t-1) \quad \forall i \in [1, p] \quad (5.7)$$

where $d\{\cdot\}^f$ denotes filtering by $\Delta/T_i(q^{-1})$.

5.2.2 The unconstrained multivariable GPC control law

The multivariable GPC cost function is defined as

$$J(t) = \sum_{i=1}^p \left\{ \sum_{j=1}^N \mu_i(j) [w_i(t+j|t) - \hat{y}_i(t+j|t)]^2 + \bar{\lambda}_i \sum_{j=0}^{N_u} [\Delta u_i(t+j|t)]^2 \right\} \quad (5.8)$$

where N is the prediction horizon, N_u the control horizon, $\hat{y}_i(t+j|t)$ the future output predictions, $w_i(t+j|t)$ the future set-point sequence, $\bar{\lambda}_i$ the constant weighting on control moves and

$$\mu_i(j) = \begin{cases} \bar{\mu}_i & \forall j \in [N_{1i}, N_{2i}], \\ 0 & \text{otherwise.} \end{cases}$$

the weights on the tracking errors with $\bar{\mu}_i$ being constants. The choice of $\mu_i(j)$ allows tailoring of the relative importance of the outputs and also enables the use of different prediction horizons for each. The prediction horizon N is chosen according to $N = \max\{N_{21}, N_{22}, \dots, N_{2p}\}$. The subscript i denotes the i^{th} channel of the system. Note that $\mathcal{T}(q^{-1})$ is a design polynomial matrix and is not estimated. It forms an important part in the design of GPC.

Define the following quantities:

$$\mathcal{G} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{G}_2 & \mathbf{G}_1 & \cdots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{G}_{N_u} & \cdots & \cdots & \mathbf{G}_1 \\ \vdots & & & \vdots \\ \mathbf{G}_N & \mathbf{G}_{N-1} & \cdots & \mathbf{G}_{N-N_u+1} \end{bmatrix} \in \mathbb{R}^{(pN \times pN_u)},$$

$$\mathbf{G}_k = \begin{bmatrix} g_{11}^k & g_{12}^k & \cdots & g_{1p}^k \\ g_{21}^k & g_{22}^k & \cdots & g_{2p}^k \\ \vdots & & \cdots & \vdots \\ g_{p1}^k & g_{p2}^k & \cdots & g_{pp}^k \end{bmatrix} \in \mathbb{R}^{(p \times p)},$$

$$\Delta \mathbf{u} = [\Delta u_1(t|t) \cdots \Delta u_p(t|t) \Delta u_1(t+1|t) \cdots \Delta u_p(t+N_u-1|t)]^T \in \mathbb{R}^{(p \times N_u)},$$

$$\hat{\mathbf{y}} = [\hat{y}_1(t+1|t) \cdots \hat{y}_p(t+1|t) \hat{y}_1(t+2|t) \cdots \hat{y}_p(t+N|t)]^T \in \mathbb{R}^{(p \times N)},$$

$$\mathbf{w} = [w_1(t+1|t) \cdots w_p(t+1|t) w_1(t+2|t) \cdots w_p(t+N|t)]^T \in \mathbb{R}^{(p \times N)},$$

$$\mathbf{f} = [f_1(t+1|t) \cdots f_p(t+1|t) f_1(t+2|t) \cdots f_p(t+N|t)]^T \in \mathbb{R}^{(p \times N)},$$

$$\mathbf{\Lambda} = \text{diag}\{\bar{\lambda}_1 \cdots \bar{\lambda}_p \bar{\lambda}_1 \cdots \bar{\lambda}_p \cdots \bar{\lambda}_1 \cdots \bar{\lambda}_p\} \in \mathbb{R}^{(pN_u \times pN_u)},$$

$$\mathbf{M} = \text{diag}\{\mu_1(1) \cdots \mu_p(1) \mu_1(2) \cdots \mu_p(2) \cdots \mu_1(N) \cdots \mu_p(N)\} \in \mathbb{R}^{(pN \times pN)}$$

where g_{ij}^k is the k^{th} step response coefficient of output i given a step in input j , $\mathbf{0}$ is a null matrix of dimension p and $f_i(t+j|t)$ are the free response predictions for output i which can be obtained by iterating

$$\mathcal{T}^{-1} \mathcal{A} \Delta \mathbf{y}(t) = \mathcal{T}^{-1} \mathcal{B} \Delta \mathbf{u}(t), \quad (5.9)$$

forward in time assuming that the future deviation of inputs is zero. Alternately the predictions can be acquired by solving Diophantine equations. However, this is not favoured in practice due to the added complexity, especially in an adaptive context.

The output prediction vector can now be written in a compact matrix form

$$\hat{\mathbf{y}} = \mathcal{G}\Delta\mathbf{u} + \mathbf{f},$$

and the controller tracking errors are

$$\mathcal{E} = \mathbf{w} - \mathbf{f}.$$

Thus the cost function becomes

$$\begin{aligned} J(t) &= \mathcal{E}^T \mathbf{M} \mathcal{E} + \Delta\mathbf{u}^T \mathbf{\Lambda} \Delta\mathbf{u} \\ &= \Delta\mathbf{u}^T (\mathcal{G}^T \mathbf{M} \mathcal{G} + \mathbf{\Lambda}) \Delta\mathbf{u} + 2(\mathbf{f} - \mathbf{w})^T \mathcal{G} \Delta\mathbf{u} + \text{const.} \\ &= \|\mathbf{R} \Delta\mathbf{u} - \mathbf{V}\|_2^2 \end{aligned} \tag{5.10}$$

where

$$\mathbf{R} = \begin{bmatrix} \mathbf{M}^{1/2} \mathcal{G} \\ \mathbf{\Lambda}^{1/2} \end{bmatrix} \in \mathbb{R}^{(2pN \times pN_u)} \quad \mathbf{V} = \begin{bmatrix} \mathbf{M}^{1/2} \mathcal{E} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(2pN \times pN_u)}$$

Minimisation of the cost gives

$$\begin{aligned} \Delta\mathbf{u} &= (\mathbf{R}^T \mathbf{R})^{-1} \mathbf{R}^T \mathbf{V} \\ &= (\mathcal{G}^T \mathbf{M} \mathcal{G} + \mathbf{\Lambda})^{-1} \mathcal{G}^T \mathbf{M} (\mathbf{w} - \mathbf{f}), \end{aligned} \tag{5.11}$$

where we adopt the standard *receding horizon* policy by applying only the first postulated control action (i.e. the first p elements of $\Delta\mathbf{u}$) at each time instant. Eqn.(5.11) is the unconstrained control law where the step response coefficients of $\mathcal{G}$ need to be estimated from the plant data.

One can follow some simple guidelines to obtain appropriate controller parameters for good performance in many cases. The upper control horizons N_{2i} usually reflect the desired closed-loop settling times. N_u is usually chosen according to the complexity of the plant.

The more unstable open-loop poles (or highly under-damped poles) the higher N_u is recommended. Control weighting is used to restrict control activities. The relative importance of the outputs can be adjusted using the tracking error weights. It has been found that the rôle of $\mathcal{T}(q^{-1})$ is of prime importance in practice (Lambert, 1987; Mohtadi, 1988b; Robinson and Clarke, 1991; Kouvaritakis and Rossiter, 1992). In short, it enhances the robustness and performance of the closed-loop in the face of unmodelled dynamics. Through $\mathcal{T}(q^{-1})$ disturbance rejection properties can also be tailored.

5.2.3 Addition of hard constraints

The following four types of hard constraints are included in the control law:

- Input magnitude constraints: $\underline{u}_i \leq u_i(t) \leq \bar{u}_i \quad \forall i \in [1, p]$
- Input rate constraints: $|\Delta u_i(t)| \leq \overline{\Delta u}_i \quad \forall i \in [1, p]$
- Output magnitude constraints: $\underline{y}_i \leq y_i(t) \leq \bar{y}_i \quad \forall i \in [1, p]$
- Auxiliary output magnitude constraints: $\underline{y}_i \leq y_i(t) \leq \bar{y}_i \quad \forall i \in [p+1, q]$

where $u_i(t)$ and $y_i(t)$ are the i^{th} process input and output respectively. The *auxiliary constraints* are applicable for oversquare ($p \times q$, $q > p$) systems where only p outputs can be controlled using the p available inputs. The extra $q - p$ outputs cannot be controlled but can be constrained to lie within a specified range. The setup is shown in Figure 5.5 where the square ($p \times p$) part of the oversquare ($p \times q$) system is defined as the *main process* and the remaining $p \times (q - p)$ subsystem is called the *auxiliary process*; whose outputs are denoted by $v(t) = y_i(t) \forall i \in [p+1, q]$.

Notice that we have assumed that the input rate constraints are symmetrical and can be expressed as a single inequality. Using the following transformations (Rossiter and Kouvaritakis, 1993):

$$\begin{aligned} u_i^c &= \frac{\bar{u}_i + \underline{u}_i}{2} & u_i^r &= \frac{\bar{u}_i - \underline{u}_i}{2} & \forall i \in [1, p] \\ y_i^c &= \frac{\bar{y}_i + \underline{y}_i}{2} & y_i^r &= \frac{\bar{y}_i - \underline{y}_i}{2} & \forall i \in [1, q] \end{aligned}$$

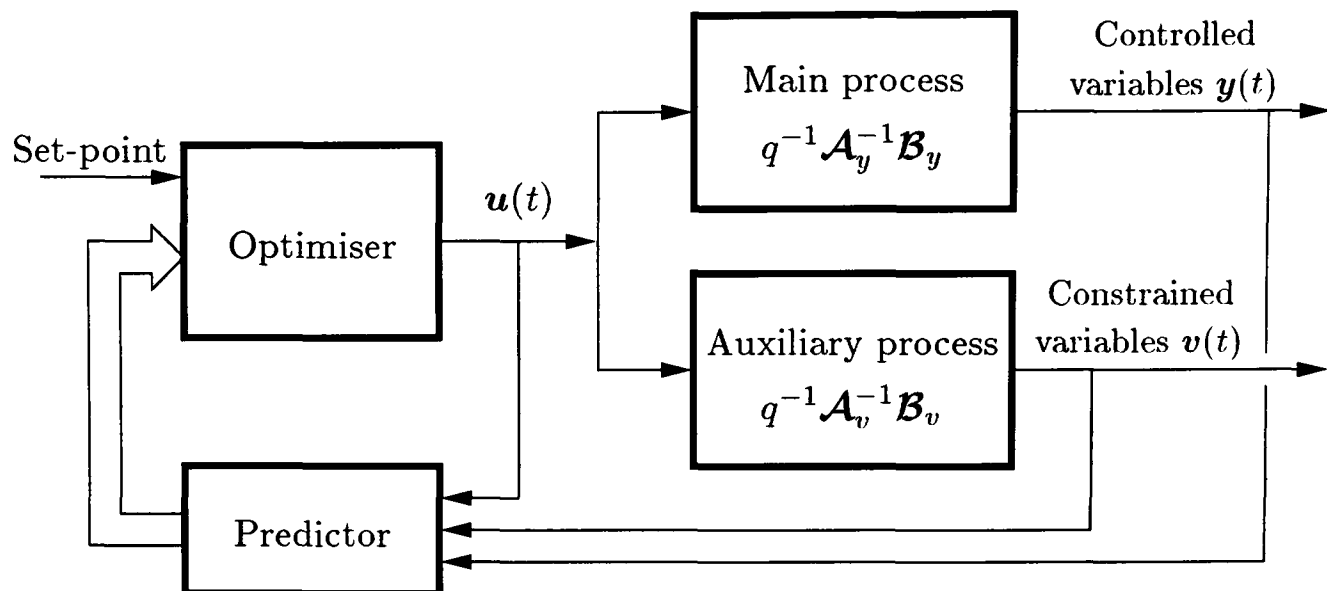


Figure 5.5: The multivariable constrained predictive control scheme

it is easy to see that the three inequalities

$$|u_i(t) - u_i^c|/u_i^r \leq 1 \quad \forall i \in [1, p] \quad (5.12)$$

$$|\Delta u_i(t)|/\overline{\Delta u_i} \leq 1 \quad \forall i \in [1, p] \quad (5.13)$$

$$|y_i(t) - y_i^c|/y_i^r \leq 1 \quad \forall i \in [1, q] \quad (5.14)$$

are equivalent to the original set of constraints. The control objective now consists of minimising the performance index of eqn.(5.8) such that the constraints eqn.(5.12)–eqn.(5.14) are satisfied. The solution to this optimisation problem is usually obtained by solving a Quadratic Program (QP). Suppose $\Delta \mathbf{u}^{QP}(t)$ is the QP solution. The difficulty in computing $\Delta \mathbf{u}^{QP}(t)$ stems from the fact that the shape of the set of feasible solutions is an irregular shaped manifold limited by hyperplanes defined by the constraints. Furthermore, if the problem has no feasible solution, i.e. there is no solution that can simultaneously satisfy all the constraints, a separate procedure must ensure that a reasonable compromise is achieved between the unsatisfied constraints (Rossiter et al., 1995; Scokaert and Rawlings, 1996; Chow, 1996).

We will use an alternative solution proposed by Rossiter and Kouvaritakis (1993) which uses the MWLS algorithm of Lawson and Hanson (1974) to solve the constrained optimisation problem. This solution has the major advantage that it automatically provides a compromise solution when the problem has no feasible solution. Thus in a similar fashion to

that of Constrained Stable Generalised Predictive Control (CSGPC) (Rossiter and Kouvaritakis, 1993) and the Model Weighting Adaptive Control (MWAC) scheme of (Gendron and Perrier, 1994) we rewrite the input constraints in the convenient form:

$$\|L_u \Delta u - C_u\|_\infty \leq 1$$

where

$$L_u = \begin{bmatrix} \tilde{I}_u & & & \\ \tilde{I}_u & \tilde{I}_u & & \\ \vdots & & \ddots & \\ \tilde{I}_u & \tilde{I}_u & \dots & \tilde{I}_u \end{bmatrix} \quad \tilde{I}_u = \begin{bmatrix} 1/u_1^r & & & \\ & 1/u_2^r & & \\ & & \ddots & \\ & & & 1/u_p^r \end{bmatrix}$$

and

$$C_u = \begin{bmatrix} \tilde{u}^c \\ \tilde{u}^c \\ \vdots \\ \tilde{u}^c \end{bmatrix} \quad \tilde{u}^c = \begin{bmatrix} (u_1^c - u(t-1)) / u_1^r \\ (u_2^c - u(t-1)) / u_2^r \\ \vdots \\ (u_p^c - u(t-1)) / u_p^r \end{bmatrix}$$

Similarly, the output constraints can be written

$$\|L_y \Delta u - C_y\|_\infty \leq 1$$

where

$$L_y = \begin{bmatrix} \tilde{G}_1 & & & \\ \tilde{G}_2 & \tilde{G}_1 & & \\ \vdots & & \ddots & \\ \tilde{G}_{N_u} & \dots & \dots & \tilde{G}_1 \\ \vdots & & \dots & \vdots \\ \tilde{G}_N & \tilde{G}_{N-1} & \dots & \tilde{G}_{N-N_u+1} \end{bmatrix} \quad \tilde{G}_k = \begin{bmatrix} g_{11}^k/y_1^r & g_{12}^k/y_1^r & \dots & g_{1p}^k/y_1^r \\ g_{21}^k/y_2^r & g_{22}^k/y_2^r & \dots & g_{2p}^k/y_2^r \\ \vdots & & & \vdots \\ g_{q1}^k/y_q^r & g_{q2}^k/y_q^r & \dots & g_{qp}^k/y_q^r \end{bmatrix}$$

and

$$C_y = \begin{bmatrix} \tilde{y}^c \\ \tilde{y}^c \\ \vdots \\ \tilde{y}^c \end{bmatrix} \quad \tilde{y}^c = \begin{bmatrix} (y_1^c - f_1(t+1|t)) / y_1^r \\ (y_2^c - f_2(t+1|t)) / y_2^r \\ \vdots \\ (y_q^c - f_q(t+1|t)) / y_q^r \end{bmatrix}$$

Finally, the input rate constraints can be expressed

$$\|L_{du}\Delta u\|_{\infty} \leq 1$$

where

$$L_{du} = \begin{bmatrix} \tilde{I}_{du} \\ \tilde{I}_{du} & \tilde{I}_{du} \\ \vdots & & \ddots \\ \tilde{I}_{du} & \tilde{I}_{du} & \dots & \tilde{I}_{du} \end{bmatrix} \quad \tilde{I}_{du} = \begin{bmatrix} 1/\overline{\Delta u_1} & & & \\ & 1/\overline{\Delta u_2} & & \\ & & \ddots & \\ & & & 1/\overline{\Delta u_p} \end{bmatrix}$$

Grouping all the constraints under a single norm we get

$$\|L\Delta u - C\|_{\infty} \leq 1 \quad (5.15)$$

where

$$L = \begin{bmatrix} L_u \\ L_y \\ L_{du} \end{bmatrix} \in \mathbb{R}^{(N_c \times pN_u)} \quad C = \begin{bmatrix} C_u \\ C_y \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(N_c \times pN_u)}$$

where the total number of constraints $N_c = 2pN_u + qN$.

5.2.4 The Mixed Weights Least-Squares solution

The multivariable constrained predictive control law is obtained by combining the GPC cost of eqn.(5.8) (l_2 -norm) with the constraints given by eqn.(5.15) (l_{∞} -norm). The future controls can then be obtained by solving the following MWLS problem:

$$\Delta u^{(i)}(t) = \arg \min_{\Delta u} \left\| \begin{bmatrix} w^{(i)}I & \mathbf{0} \\ \mathbf{0} & W^{(i)} \end{bmatrix}^{1/2} \left(\begin{bmatrix} R \\ L \end{bmatrix} \Delta u(t) - \begin{bmatrix} V \\ C \end{bmatrix} \right) \right\|_2^2 \quad (5.16)$$

i.e. starting with the scalar $w^{(0)} = 1$ and the diagonal matrix $W^{(0)} = I$, iterate

$$\Delta u^{(i)}(t) = \left(\begin{bmatrix} R \\ L \end{bmatrix}^T \begin{bmatrix} w^{(i)}I & \mathbf{0} \\ \mathbf{0} & W^{(i)} \end{bmatrix} \begin{bmatrix} R \\ L \end{bmatrix} \right)^{-1} \begin{bmatrix} R \\ L \end{bmatrix}^T \begin{bmatrix} w^{(i)}I & \mathbf{0} \\ \mathbf{0} & W^{(i)} \end{bmatrix} \begin{bmatrix} V \\ C \end{bmatrix}$$

using the following update for the scalar $w^{(i)}$:

$$w^{(i+1)} = \frac{w^{(i)}}{\sum_{k=1}^{N_c} w_{kk}^{(i)} |e_k^{(i)}|}$$

with the entries of the diagonal matrix $\mathbf{W}^{(i)}$ given by

$$w_{jj}^{(i+1)} = \frac{w_{jj}^{(i)} |e_j^{(i)}|}{\sum_{k=1}^{N_c} w_{kk}^{(i)} |e_k^{(i)}|}$$

where the optimisation error vector for iteration i is

$$\mathbf{e}^{(i)} = \mathbf{L}\Delta\mathbf{u}^{(i)}(t) - \mathbf{C}$$

and repeat the procedure until $\Delta\mathbf{u}^{(i)}(t)$ converges. Rossiter and Kouvaritakis (1993) show that if there exists a feasible solution to the constraint problem then

$$\lim_{i \rightarrow \infty} \Delta\mathbf{u}^{(i)}(t) = \Delta\mathbf{u}^{QP}(t) \quad (\text{If feasible})$$

and if there is no feasible solution then $\lim_{i \rightarrow \infty} \Delta\mathbf{u}^{(i)}(t)$ still exists and it solves

$$\lim_{i \rightarrow \infty} \Delta\mathbf{u}^{(i)}(t) = \min \|\mathbf{L}\Delta\mathbf{u}(t) - \mathbf{C}\|_{\infty} \quad (\text{If infeasible})$$

i.e. it minimises the *maximum constraint violation*.

In practice, the iteration are stopped when the solution of two consecutive iterations do not vary by more than a sufficiently small value, i.e.

$$|\Delta\mathbf{u}^{(i+1)}(t) - \Delta\mathbf{u}^{(i)}(t)| < \varepsilon. \quad (\text{Stopping rule})$$

5.2.5 The multivariable MMLS on-line estimator

Maniar et al. (1997) shows that a full $p \times q$ MIMO system can be decomposed into q MISO subsystems as in eqn.(5.5) which can be easily converted to the ARX form of eqn.(5.7). Thus q MMRLS estimators can be used to identify the order and the model parameters corresponding to each output on-line. Thus, for the i^{th} output, we have the following augmented regression model:

$$0 = \boldsymbol{\theta}_i^T(t) \boldsymbol{\phi}_i(t), \quad \forall i \in [1, q]$$

with augmented data vector and parameter vector structure

$$\begin{aligned}\phi_i(t) &= \left(du_1^f(t - \bar{\tau}_i - m) \dots du_p^f(t - \bar{\tau}_i - m), \dots, du_1^f(t - \underline{\tau}_i - m + 1) \dots du_p^f(t - \underline{\tau}_i - m + 1), \right. \\ &\quad du_1^f(t - \underline{\tau}_i - m) \dots du_p^f(t - \underline{\tau}_i - m) - dy_i^f(t - m), \dots, \\ &\quad \left. du_1^f(t - \underline{\tau}_i - 1) \dots du_p^f(t - \underline{\tau}_i - 1) - dy_i^f(t - 1), du_1^f(t - \underline{\tau}_i) \dots du_p^f(t - \underline{\tau}_i) - dy_i^f(t) \right)^T, \\ \theta_i(t) &= \left(b_{i1}^{\bar{\tau}_i+m} \dots b_{ip}^{\bar{\tau}_i+m}, \dots, b_{i1}^{\underline{\tau}_i+m-1} \dots b_{ip}^{\underline{\tau}_i+m-1}, b_{i1}^{\underline{\tau}_i+m} \dots b_{ip}^{\underline{\tau}_i+m} a_i^m, \dots, \right. \\ &\quad \left. b_{i1}^{\underline{\tau}_i+1} \dots b_{ip}^{\underline{\tau}_i+1} a_i^1, b_{i1}^{\underline{\tau}_i} \dots b_{ip}^{\underline{\tau}_i} \underline{1} \right)^T,\end{aligned}$$

where $d\{\cdot\}^f$ denotes filtering by Δ/T_i , b_{ij}^k and a_i^k are the k^{th} coefficients of the MISO polynomials $B_{ij}(q^{-1})$ and $A_i(q^{-1})$ respectively, m is the maximum order of the model and $\underline{\tau}_i, \bar{\tau}_i$ define the minimum and maximum expected dead-time of the loop; the coefficients b_{ij}^k for $0 \leq k < \underline{\tau}_i$ are assumed to be equal to zero.

Example 3 (The coupled tanks) *If we assume that the maximum order for each of the 3 MISO tank models $m = 3$ and that the expected minimum and maximum dead-times are $\underline{\tau}_1 = \underline{\tau}_2 = 1, \underline{\tau}_3 = 2, \bar{\tau}_1 = \bar{\tau}_2 = 3, \bar{\tau}_3 = 5$ samples, then the MMRLS augmented data and parameter vectors for the coupled tank system are given by:*

Tank 1 (left)

$$\begin{aligned}\phi_1(t) &= \left(du_1^f(t - 6) du_2^f(t - 6), du_1^f(t - 5) du_2^f(t - 5), \right. \\ &\quad du_1^f(t - 4) du_2^f(t - 4) - dy_1^f(t - 3), du_1^f(t - 3) du_2^f(t - 3) - dy_1^f(t - 2), \\ &\quad \left. du_1^f(t - 2) du_2^f(t - 2) - dy_1^f(t - 1), du_1^f(t - 1) du_2^f(t - 1) - dy_1^f(t) \right)^T, \\ \theta_1(t) &= \left(b_{11}^6 b_{12}^6, b_{11}^5 b_{12}^5, b_{11}^4 b_{12}^4 a_1^3, b_{11}^3 b_{12}^3 a_1^2, b_{11}^2 b_{12}^2 a_1^1, b_{11}^1 b_{12}^1 \underline{1} \right)^T.\end{aligned}$$

Tank 2 (right)

$$\begin{aligned}\phi_2(t) &= \left(du_1^f(t - 6) du_2^f(t - 6), du_1^f(t - 5) du_2^f(t - 5), \right. \\ &\quad du_1^f(t - 4) du_2^f(t - 4) - dy_2^f(t - 3), du_1^f(t - 3) du_2^f(t - 3) - dy_2^f(t - 2), \\ &\quad \left. du_1^f(t - 2) du_2^f(t - 2) - dy_2^f(t - 1), du_1^f(t - 1) du_2^f(t - 1) - dy_2^f(t) \right)^T, \\ \theta_2(t) &= \left(b_{21}^6 b_{22}^6, b_{21}^5 b_{22}^5, b_{21}^4 b_{22}^4 a_2^3, b_{21}^3 b_{22}^3 a_2^2, b_{21}^2 b_{22}^2 a_2^1, b_{21}^1 b_{22}^1 \underline{1} \right)^T.\end{aligned}$$

Tank 3 (centre)

$$\begin{aligned}\phi_3(t) &= \left(du_1^f(t - 8) du_2^f(t - 8), du_1^f(t - 7) du_2^f(t - 7), du_1^f(t - 6) du_2^f(t - 6), \right. \\ &\quad \left. du_1^f(t - 5) du_2^f(t - 5) - dy_3^f(t - 3), du_1^f(t - 4) du_2^f(t - 4) - dy_3^f(t - 2), \right.\end{aligned}$$

$$\begin{aligned} & du_1^f(t-3) \ du_2^f(t-3) \ -dy_3^f(t-1), \ du_1^f(t-2) \ du_2^f(t-2) \ \underline{-dy_3^f(t)}^T, \\ \theta_3(t) = & \left(b_{31}^8 \ b_{32}^8, b_{31}^7 \ b_{32}^7, \ b_{31}^6 \ b_{32}^6, \ b_{31}^5 \ b_{32}^5 \ a_3^4, \ b_{31}^4 \ b_{32}^4 \ a_3^2, \ b_{31}^3 \ b_{32}^3 \ a_3^1, \ b_{31}^2 \ b_{32}^2 \ 1 \right)^T. \end{aligned}$$

Thus we can model the coupled tank system on-line by using the 2×3 MISO model structure of Figure 5.6 and employing 3 MMRLS estimators, i.e. one for each tank. Note that we can only control two levels given that we only have two pumps. However, the extra output (the level of, say, the middle tank) can be constrained to lie within specified bounds by including it as an auxiliary output constraint in the controller optimisation function of eqn.(5.16). ■

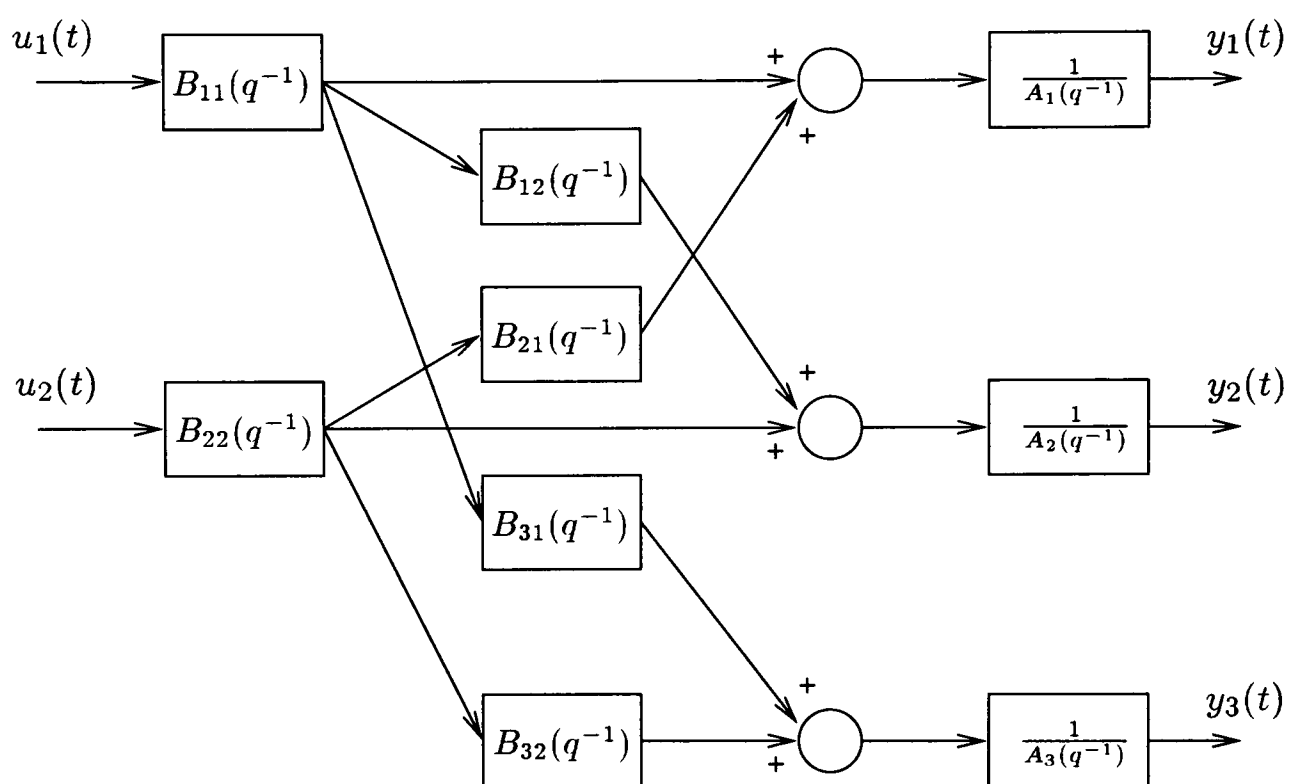


Figure 5.6: The 2×3 MISO model structure for the coupled tanks

The adaptive multiple model GPC (MMGPC) control law is obtained by combining the MMRLS estimator described in the coupled tanks example in section 5.2.5 with the multivariable constrained GPC control law of section 5.2.4. However, in order to improve the performance of the control law the following additional checks based on *a priori* knowledge of the system were imposed before the estimated plant parameters were used in the computation of the next control moves:

- The estimated models were checked for unstable poles (i.e. do any roots of $A_i(q^{-1})$ lie outside the unit circle in the z -plane ?) so that the future (open-loop) predictions

remained stable and hence bounded.

- The gain of the primary input–output models were also be checked to make sure that it was not negative (i.e. $B_{ii}(1)/A_i(1) > 0$) on the assumption that a positive change in pump i flowrate always causes a positive change in the level of corresponding tank i .

If any of these were not satisfied, then the estimated model coefficients used in the computing the next controls were not updated, and the previous model of the process was used again for computing the next control move.

5.3 Simulation-based comparison with an optimal non-linear predictive controller

In this section the coupled tank system is simulated by integrating the non-linear differential equations of eqn.(5.1)–eqn.(5.3) using the SIMULINK Dormand-Prince variable step algorithm with the following values for the system constants:

$$A = 0.02; \quad a_1 = 10^{-4}; \quad a_{12} = 7 \times 10^{-5}; \quad a_2 = 7 \times 10^{-5}; \quad a_{23} = 10^{-4}; \quad a_3 = 3 \times 10^{-5}$$

where all have units m^2 and are based on the actual rig dimensions which gives the model similar behaviour to the actual coupled tanks rig for a particular (but arbitrary) configuration of the five valves. This simulated coupled tanks model was then used to compare the performance of the adaptive multiple model GPC scheme with a non-linear GPC control scheme.

The non-linear predictive controller used the predictions generated from the non-linear model and a non-linear constrained optimisation routine (in this case the MATLAB Optimisation Toolbox's "constr" function) to minimise the same multivariable GPC cost function as eqn.(5.8). The output predictions were obtained by integrating the differential equations (5.1)–(5.3) (with identical system constants to those of the simulated coupled tanks model) using the Runge–Kutta 4 step method with a fixed integration step size of $t_s/1000$ where t_s is the controller sampling interval. The SIMULINK worksheet for the non-linear predictive controller and coupled tanks model is shown in Figure 5.7.

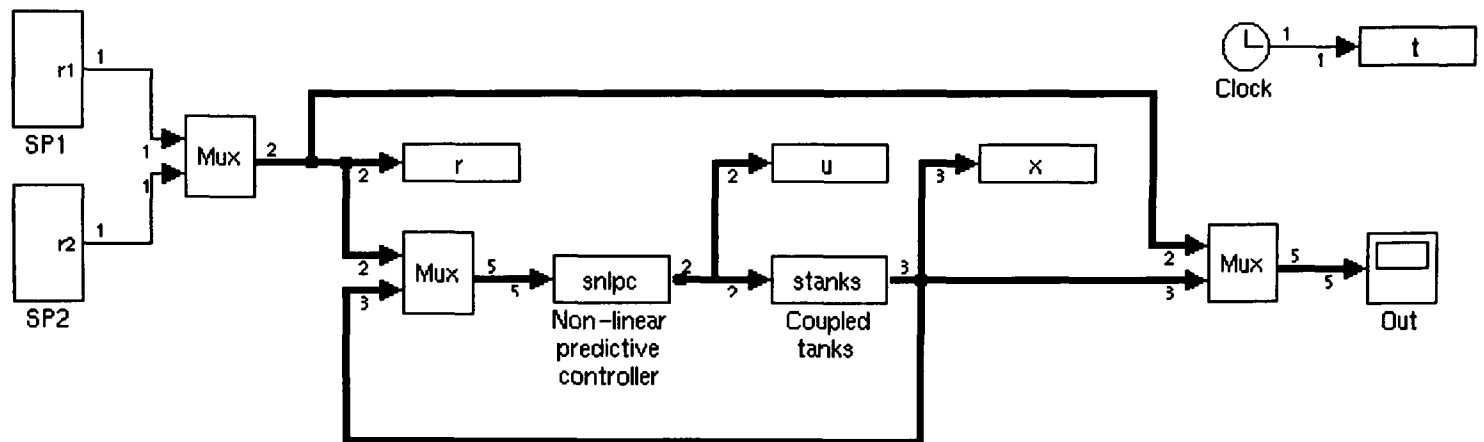


Figure 5.7: SIMULINK worksheet of the constrained non-linear predictive control scheme

5.3.1 Controller tuning

A lower prediction horizon $N_1 = 1$ sample was chosen since theoretically there should be no transport delay in changing the flow rate into each tank. The findings were that short prediction horizons gave faster closed-loop performance (compare the aggressive minimum variance ($N_2 = 1$) response of Figure 5.8 to the slow mean-level like ($N_2 = 10$) response of Figure 5.9). Thus an upper prediction horizon of $N_2 = 4$ samples was chosen as the smallest horizon to encompass the maximum expected dead-time which we assume to be 3 samples (see also the next paragraph on tuning the MMRLS estimator). Another reason for this relatively short prediction horizon ($N_2 = 4$ corresponds to an output horizon of just 16 seconds which is a small fraction of the system rise-time of $\tau \sim 100$ seconds) is due to the assumed linear model structure of MMGPC which is inherently incapable to providing accurate long-range predictions due to the non-linear nature of the couple tanks system. Since there were no unstable or oscillatory modes, the control horizon was chosen as $N_u = 1$. A relatively quick sampling interval of $t_s = 4s$ ($t_s/\tau \sim 1/25$) was employed so as to react quickly to any disturbances on the system. Thus the final GPC tuning parameters were:

$$N_1 = 1; \quad N_{2i} = 4; \quad N_u = 1; \quad \bar{\mu}_i = 1; \quad \bar{\lambda}_i = 10^{-6} \quad \forall i \in [1, 2]$$

for each loop. The following input constraints were placed on the predictive controller:

$$0 \leq u_i \leq 1; \quad -1 \leq \Delta u_i \leq 1 \quad \forall i \in [1, 2]$$

with the following output constraints

$$0.125 \leq y_i \leq 0.875 \quad \forall i \in [1, 3]$$

where the inputs and outputs were normalised such that $0 \leq u_i \leq 1$ correspond to flow rates of $0 - 0.5$ [l/s] and $0 \leq y_i \leq 1$ correspond to levels of $0 - 1$ [m] respectively.

Theoretically there should be no time delay between voltage to pump and flow of water into the tank (assuming an ideal pump and that the pipes do not expand) and thus the minimum dead-time $\underline{\tau} = 1$ sample with no dead-time variation should be used. However, since the sample time t_s was relatively fast compared to the process dynamics, a dead-time variation $\Delta\tau = 2$ samples (i.e. $\bar{\tau} = 3$) was employed in the MMRLS data vector as a “sleep-easy” factor to handle unforeseen changes such as non-ideal pump characteristics or brief blockages in the pipes. The regularisation constant $\mathcal{D}_{max} = 1/J_{min}$ where J_{min} is the minimum possible loss function value which was taken to be 10^{-6} . The variable forgetting mechanism which attempts to keep the minimum loss function value constant and equal to the ideal loss $J_{opt} = AML \cdot \sigma^2 = 10^{-3}$ was tuned to give an Asymptotic Memory Length of approximately 1000 samples. Thus the final MMRLS estimator tuning parameters were

$$\underline{\tau} = 1; \quad \bar{\tau} = 3; \quad m = 3; \quad \mathcal{D}_{max} = 10^6; \quad J_{opt} = 10^{-3} \quad \forall i \in [1, 3]$$

where m is the maximum order of the ARX model. The CARIMA model design polynomial was chosen to be

$$T_i(q^{-1}) = (1 - 0.8q^{-1})^2 \quad \forall i \in [1, 3]$$

for both the GPC controller and the MMRLS estimator. The frequency response of the resulting bandpass data filter Δ/T in Figure 5.10. The shape of this filter was chosen to eliminate DC offsets (via the Δ term) in estimation and control, and to improve the controller robustness by limiting the bandwidth of the closed-loop system (via the $1/T$ low-pass filter). Thus Δ/T reduces the ill-effects of plant-model mismatch (due to the assumed linear model structure and, for example, sudden changes in valve positions) and other disturbances (such as measurement noise due to the pump disturbing the water surface as it pumps water into the tank) on the adaptive control law by preventing signals outside the desired closed-loop bandwidth from affecting the controller.

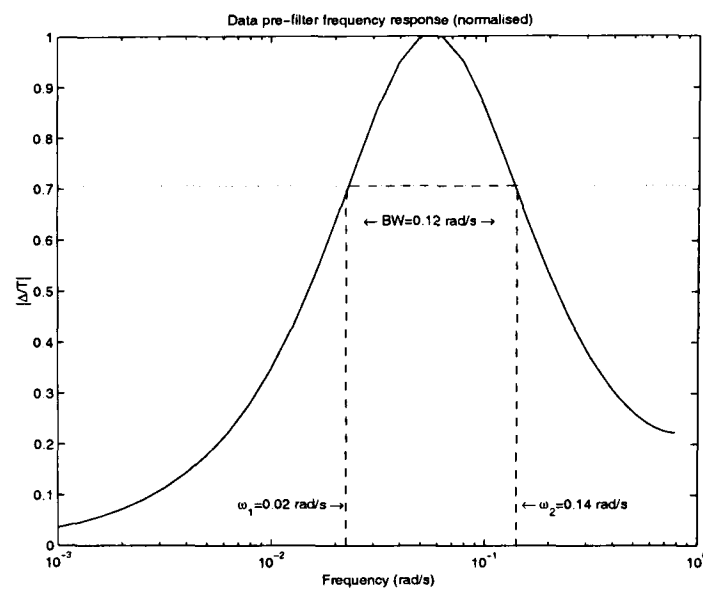


Figure 5.10: The data pre-filter frequency response for a sampling time $t_s = 4s$

5.3.2 Simulation results

Figures 5.11 and 5.12 show the responses of non-linear predictive control versus that of adaptive MMGPC (which was trained up using random setpoint changes for the first 1000 seconds of the simulation). Clearly the performances of both schemes are very similar in this noise-free simulation which indicates that adaptation can be used to compensate for the non-linear behaviour displayed by coupled tanks. The big advantage of the adaptive algorithm is that it is much faster in terms of computational speed: it only took just over a minute to run the MMGPC simulation (on a Sun SPARCstation 5) whereas the non-linear algorithm took the order of 10 hours ! However the adaptive MMGPC algorithm cannot be used to control all classes of non-linear systems, and in fact it would appear that adaptive algorithms such as MMGPC can only be used for systems which display smooth non-linearities (i.e. differentiable up to any order) such that the adaptive mechanism has an opportunity to use the data in the vicinity of the current operating point to provide an adequate linearised model of the system about this point.

5.4 Experimental results

In this section experiments are performed on the actual coupled tanks rig described in section 5.1 in order to determine the benefits of the multiple model strategy with respect to

under/over-parameterisation of the true plant. The predictive controller was tuned using identical settings as those employed in section 5.3.1, apart from the constraints, which were modified to lie within the dSPACE signal ranges of -1 to 1 , i.e.

$$-1 \leq u_i \leq 1; \quad -1 \leq \Delta u_i \leq 1 \quad \forall i \in [1, 2]$$

and

$$-0.75 \leq y_i \leq 0.75 \quad \forall i \in [1, 3].$$

The following test procedure given in Table 5.1 was used to simulate the behaviour of a simple batch process where the product is removed from particular tanks as well as being moved from one tank to the other according to a specific schedule. The length of the run

Table 5.1: The coupled tanks test sequence ($\checkmark$ denotes valve “fully open” and $\times$ denotes valve “closed”)

t (secs)	Valve 1	Valve 3	Valve 2	Valve 13	Valve 23	Notes
0	30%	$\checkmark$	50%	50%	$\checkmark$	O/L control
20						Start logging data
40						O/L self-tuning
400						C/L adaptive
2500		$\times$				Start of disturbances
3000		$\checkmark$				
3500		$\times$			$\times$	
4000		$\checkmark$				Infeasible constraints
4500		$\times$			$\checkmark$	
5000		$\checkmark$				End of disturbances
10000	30%		50%	50%		End data logging

was $10000s \approx 3$ hours which was divided into 3 main phases: the initialisation and system identification phase $0 < t < 2500s$, the disturbance testing phase $2500 < t < 5000s$ (where the valve positions where adjusted according to Table 5.1) and the fixed configuration (i.e. no changes in valve settings) phase $5000 < t < 10000s$. Various control strategies were then tested using this test sequence including fixed model versus multiple model and fixed mode versus adaptive mode.

Figures 5.13–5.15 show the full record of results of adaptive multiple model GPC (MMGPC) with constraints for the test sequence described in Table 5.1. The first observation in Figure 5.13 is the poor setpoint tracking and gross violation of constraints at time $t \approx 4000s$. This is due to a physical limitation of the maximum flow provided by pump 1 and thus rendering the output constraint on the auxiliary variable y_3 infeasible. The MWLS solution for the control then tries to minimise the largest constraint violation and thus completely ignores setpoint tracking, and hence the observed response. The solution to this problem is to either increase the pump capacity, or to relax the lower constraint of the auxiliary variable, say, $y_3 \geq -1$.

The estimated model parameters for the 2×3 MISO model of the coupled tanks are given in Figure 5.15 with corresponding measures of model fit in Figure 5.14. Clearly the estimated model quality deteriorates at time $t \approx 4000$ due to the input saturation observed in both control signals ($u_1 > 1$ and $u_2 > 1$), as indicated by the increased LS residuals ϵ_i , the forgetting of more data (smaller AML's) and the negative jump in multiple correlation coefficient Ry_2 . As expected, once control has been regained ($t > 4500$), these measures improve again as more “good” data becomes available and the parameters converge to their new values (look, for example, at the exponential ‘funnel’ shapes of the Moving Average (MA) parameter trajectories B_{31} in Figure 5.15).

The estimated model orders are also given in Figure 5.15 and show that, under normal operating conditions, a second order model is sufficient for modelling the first and second tank levels; when the disturbances started (which only altered the second and third tank configurations) the estimated model order for the second tank increased to three which was the maximum order allowed. One also notices that, even though the model order is changing violently at times (see the more detailed trace of “Estimated model order, loop 2” in Figure 5.20), the MMGPC controller operates smoothly (see “Control, loop 2” and “Response, loop 2” of Figure 5.18) i.e. completely bumpless transfer. This is expected since all the models are generated by MMRLS *simultaneously* using an identical record of past data and only the model with the “best” (in the sense of Akaike) past performance is selected.

Table 5.2: Performance measures

Name	Norm	Definition
Sum of Absolute Errors (SAE)	l_1	$\sum_t e(t) $
Sum of Squared Errors (SSE)	l_2	$\sum_t e^2(t)$
Maximum Error (ME)	l_∞	$\max_t e(t)$

For the purposes of analysis, the full run was split into two sections: the first window running from time $t = 7500s$ to $t = 9500s$ in which there were no changes to the tank configuration, and the second which included system changes (as described in Table 5.1) which ran from $t = 2000s$ up until the time of infeasibility $t = 4000s$. The analysis used three measures, summarised in Table 5.2, to compare performances of the various strategies quantitatively. Tables 5.3 and 5.4 summarise the performance of self-tuned and adaptive multiple model GPC (MMGPC) compared to 1st order GPC (1GPC) and 3rd order GPC (3GPC). Tables 5.3 and 5.4 show that adaptive 3GPC gives the best results (on average)

Table 5.3: Summary of results: no disturbances ($7500s < t < 9500s$)

	SAE ₁	SAE ₂	SSE ₁	SSE ₂	ME ₁	ME ₂
Fixed 1GPC	9.80	4.06	1.12	0.378	0.201	0.200
Fixed 3GPC	9.71	3.88	1.16	0.397	0.201	0.200
Fixed MMSGPC	10.8	4.41	1.12	0.392	0.204	0.201
Adaptive 1GPC	9.87	4.06	1.12	0.375	0.202	0.200
Adaptive 3GPC	9.17	3.74	1.10	0.394	0.201	0.200
Adaptive MMSGPC	9.51	3.67	1.14	0.385	0.200	0.200

closely followed by adaptive MMSGPC. The interesting point here is that adaptation is seen to give the predictive control schemes improved performance compared to their fixed model counterparts (which have higher performance indices in Table 5.3) when the chosen model structure used for generating the future predictions is incorrect (i.e. linear instead of non-linear in this case) and when the data is exciting (i.e. plenty of setpoint changes). For example, look at the slow responses of self-tuned (fixed) MMSGPC in Figure 5.17 compared to the fast responses of adaptive MMSGPC in Figure 5.21.

Table 5.4: Summary of results during disturbances ($2000s < t < 4000s$)

	SAE ₁	SAE ₂	SSE ₁	SSE ₂	ME ₁	ME ₂
Fixed 1GPC	9.04	3.85	0.785	0.166	0.215	0.140
Fixed 3GPC	7.29	2.46	0.711	0.121	0.202	0.139
Fixed MMGPC	9.41	3.11	0.793	0.125	0.214	0.140
Adaptive 1GPC	9.70	4.93	0.832	0.206	0.211	0.140
Adaptive 3GPC	6.87	2.23	0.708	0.115	0.202	0.141
Adaptive MMGPC	6.88	2.30	0.698	0.119	0.199	0.139

Table 5.4 shows that when the system is perturbed by a sequence of disturbances the adaptive control schemes again manage to outperform their fixed model counterparts. This is to be expected since fixed control schemes cannot adapt to the change in plant behaviour. In addition one observes that the fixed and adaptive controllers based on a 1st order model (1GPC) performed very badly indicating that a first order model structure was insufficient to cope with the system configuration changes. This can be verified by observing the large peaks in the “LS residual” sequence of Figure 5.25 with the smaller, more “white” residuals of Figure 5.19.

5.5 Discussion

5.5.1 The effects of under/over-parameterisation

The experimental results of Tables 5.5 and 5.6 confirm that the controllers based on a fixed order 1st order model (1GPC) performed badly compared to those using higher order models (3GPC and MMGPC) in the face of disturbances. This suggests that, although the first order model performed well for the nominal coupled tank configuration, it is insufficient to cope with changes in the configuration (i.e. disturbances). The use of adaptation did not improve matters and thus there is a definite penalty in using a low order model (i.e. under-parameterising).

It was hoped to show that, if a high order model is used for control, it would be over-

Table 5.5: Summary of results relative to MMGPC: no disturbances ($7500s < t < 9500s$)

	SAE ₁	SAE ₂	SSE ₁	SSE ₂	ME ₁	ME ₂
Adaptive MMGPC	1	1	1	1	1	1
Adaptive 1GPC	1.04	1.11	0.98	0.97	1.01	1.00
Adaptive 3GPC	0.96	1.02	0.96	1.02	1.01	1.00

Table 5.6: Summary of results during disturbances ($2000s < t < 4000s$) relative to MMGPC

	SAE ₁	SAE ₂	SSE ₁	SSE ₂	ME ₁	ME ₂
Adaptive MMGPC	1	1	1	1	1	1
Adaptive 1GPC	1.41	2.14	1.19	1.73	1.06	1.01
Adaptive 3GPC	1.00	0.97	1.01	0.97	1.02	1.01

parameterised with a consequent penalty in performance. However the results in Tables 5.5 and 5.6 are inconclusive for the over-parameterisation case. This is most probably due to the fact that a third order model is not actually an over-parameterisation since there exist three energy storage elements (potential energy of the water in each tank) thus indicating that the model should indeed be of order 3. Thus one would expect the results for MMGPC (which also selected a 3rd order model for the system using Akaike's criterion) to be very similar to those of 3GPC, suggesting that higher order models should be tested. Unfortunately the small 128K memory of the dSPACE board made it impossible to try models with order greater than three. Another consideration is that the actuators had very limited authority resulting in relatively slow closed-loop responses even for "mean-level" settings and thus fast closed-loop responses, which should accentuate the adverse effects of over-parameterisation, could not be achieved.

5.5.2 Adaptation policies

The simulation and experimental results have demonstrated that adaptation is able to improve the performance of a predictive controller based on a *linear* model of a *non-linear*

system (i.e. the coupled tanks). Thus adaptation should be considered if one wishes to control a smoothly non-linear system using simple (i.e. relatively inexpensive) linear predictive control law. However the design of the adaptation structure and policy is critical to the success of the final control strategy. The following paragraphs give a brief discussion of the benefits and disadvantages of various adaptation policies:

Never adapt

With this policy, the plant model is obtained off-line using either open or closed loop data, and once commissioned this model is built into the controller and remains fixed for the controller life-time. Most industrial applications to date use this policy as it is a very robust in that control is predictable and the performance of the controller is unaffected by its past behaviour. However all physical processes age with time and thus one would expect the closed-loop performance of a fixed model controller to degrade with time. In addition, if the controller tuning parameters are changed, the model may no longer be applicable for the new settings. For example, suppose that a predictive controller was commissioned with mean-level settings (giving a low closed-loop bandwidth) with a corresponding model which has a good fit for low frequencies. If the settings were changed to more aggressive control, the model fit might not be good enough for these higher frequencies and thus may give poor performance.

Adapt always

As the results have shown, if adaptation is used, much better performance can be obtained for systems where the nominal model structure is insufficient. However, for the adaptive mechanism to work efficiently, the data has to be sufficiently exciting to provide information about the disturbances or system changes. If the data is not sufficiently exciting, the estimated model quality slowly deteriorates as the last portion of “good” data is forgotten, giving violent and unpredictable transient behaviour which then leads to “bursting” as the model parameters explode and converge again. Thus some form of jacketing is required to prevent this from happening.

Adapt on request

This is also known as self-tuning where the model is continuously updated on-line but is only used for control once the user is satisfied with the quality of the estimated model. This “intelligent adaptation” is the most pragmatic approach as it combines the robustness of the fixed model case (when the estimated model quality is poor) with the performance benefits of adaptation (when the estimated model is good). However determining the “goodness” of a model requires independent observations or a priori information (e.g. the plant must be stable with positive gain) which are difficult to obtain and problem specific. In addition, since the estimator is purposely designed with the control structure in mind, the performance of the self-tuned predictive controller (with a “matched” estimator–control design) should be better than that of fixed control where the parameters are obtained from a system identification stand-point.

5.6 Conclusions

In this chapter the MMRLS algorithm has been coupled with multivariable GPC to give an adaptive predictive control scheme which adapts not only to the plant parameters, but also tries to select a parsimonious model order. The simulations results comparing adaptive MMGPC with a non-linear model based GPC control scheme show that the adaptation mechanism using MMRLS gives the sub-optimal linear controller comparable performance to the optimal non-linear control scheme, and at much lower computational burden. However, this is a special case where the non-linearities displayed by the coupled tanks system are smooth and “non-dominant” in a sense, and thus in general one would not expect adaptive MMGPC to perform as well as the full non-linear model based GPC scheme.

The experimental results show that this multiple model scheme gives better performance than a fixed low-order adaptive controller. Unfortunately, due to memory limitations of the dSPACE card and also choice of rig, the results comparing the multiple model scheme with fixed high-order model adaptive controller were inconclusive. Still, given that MMGPC always gives as good performance as the high order controller or better, and since it has one

less design parameter to choose (i.e. the model order) without adding much computational burden to the controller, there is a strong case for using MMGPC instead of fixed model order GPC.

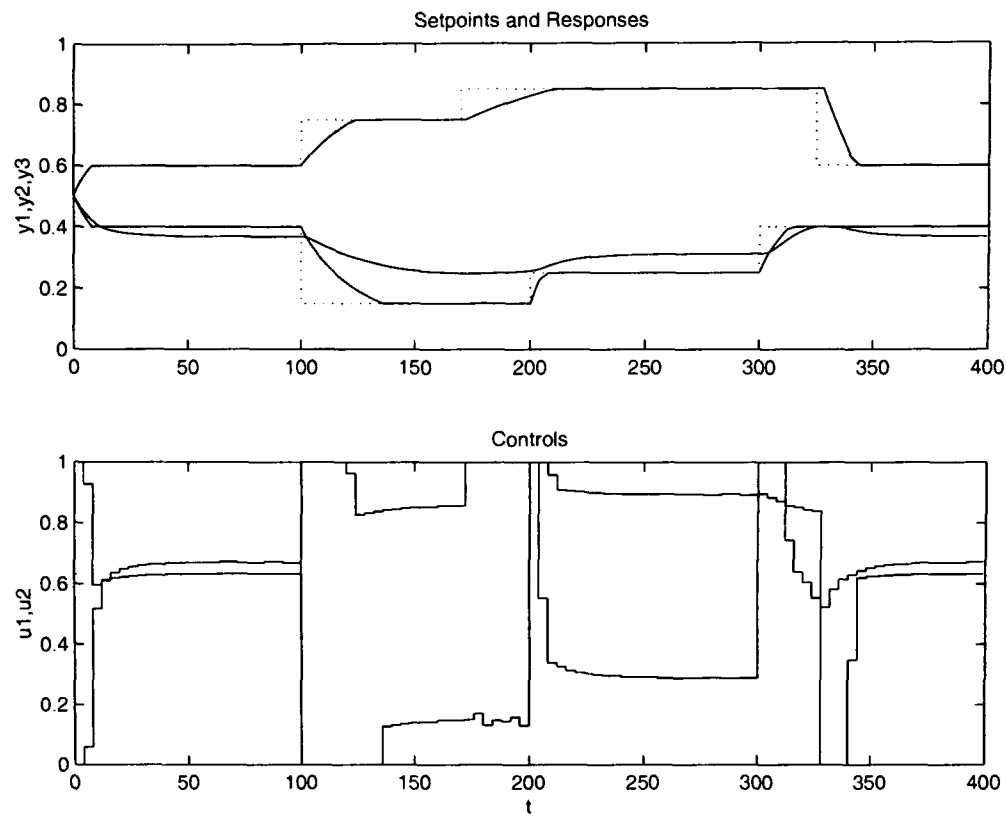


Figure 5.8: “Minimum variance” non-linear predictive control of coupled tanks ($N_1 = 1$, $N_2 = 1$, $N_u = 1$, $\bar{\mu} = 1$, $\bar{\lambda} = 10^{-6}$ and $T(q^{-1}) = 1$)

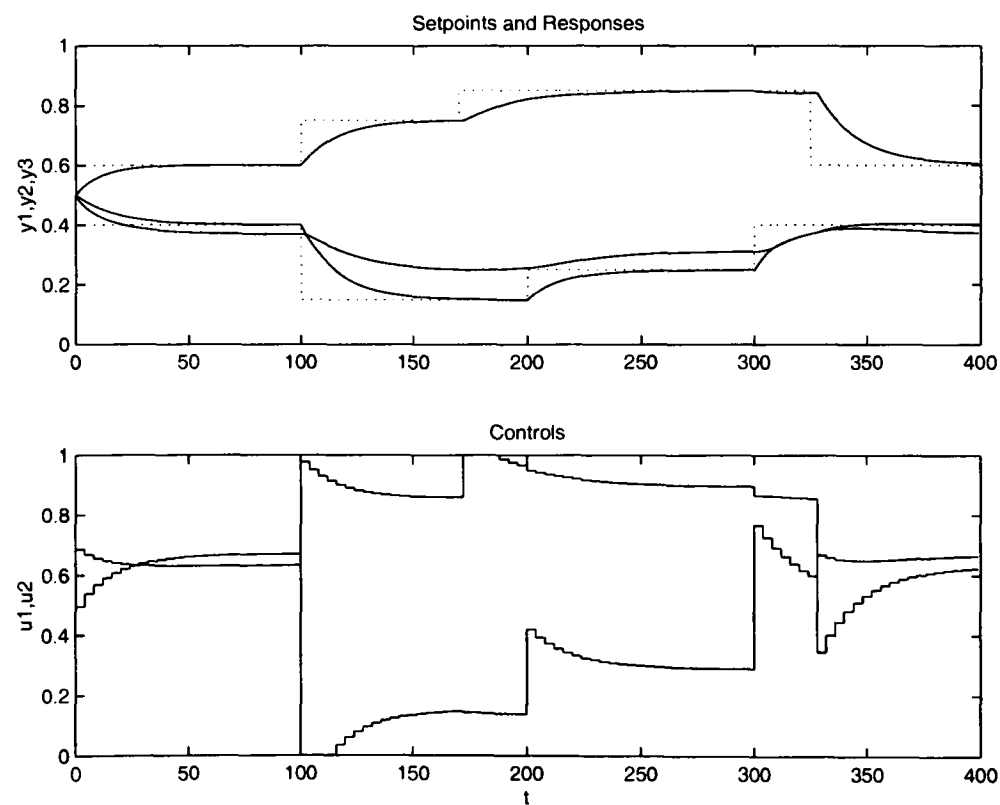


Figure 5.9: “Mean-level” non-linear predictive control of coupled tanks ($N_1 = 1$, $N_2 = 10$, $N_u = 1$, $\bar{\mu} = 1$, $\bar{\lambda} = 10^{-6}$ and $T(q^{-1}) = 1$)

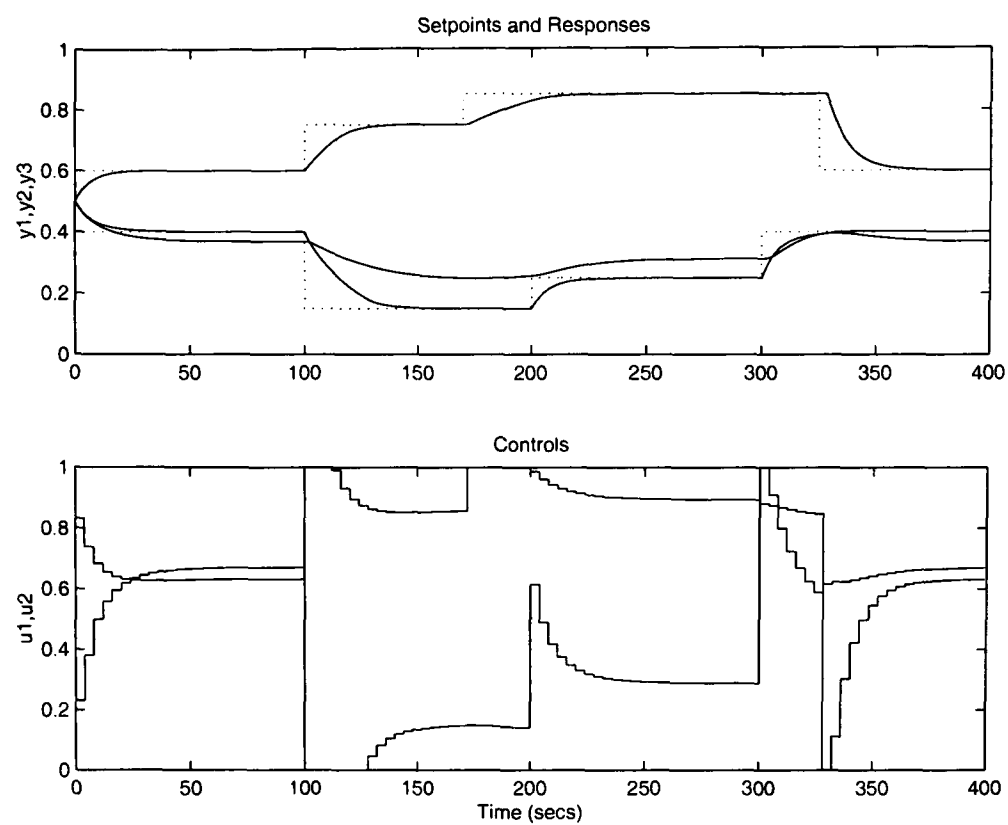


Figure 5.11: Constrained non-linear GPC responses with final tuning parameters of $N_1 = 1$, $N_2 = 4$, $N_u = 1$, $\bar{\mu} = 1$, $\bar{\lambda} = 10^{-6}$ and $T(q^{-1}) = 1$ (perfect modelling)

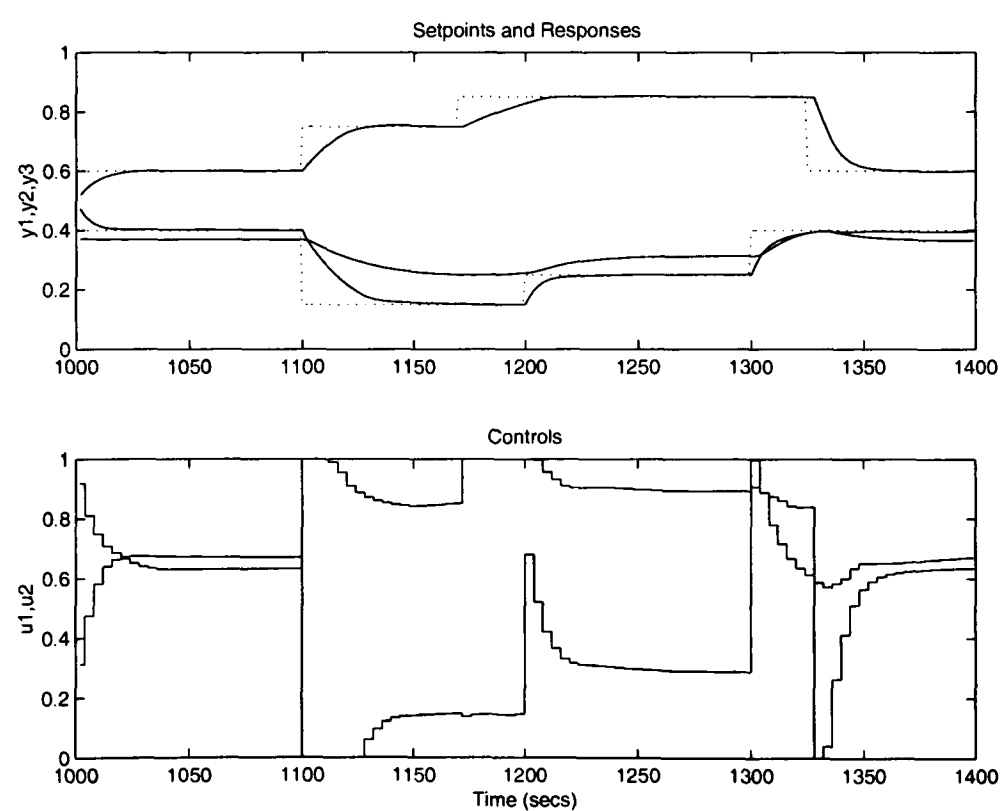


Figure 5.12: Constrained adaptive MMGPC responses with final tuning parameters of $N_1 = 1$, $N_2 = 4$, $N_u = 1$, $\bar{\mu} = 1$, $\bar{\lambda} = 10^{-6}$ and $T(q^{-1}) = (1 - 0.8)^2$

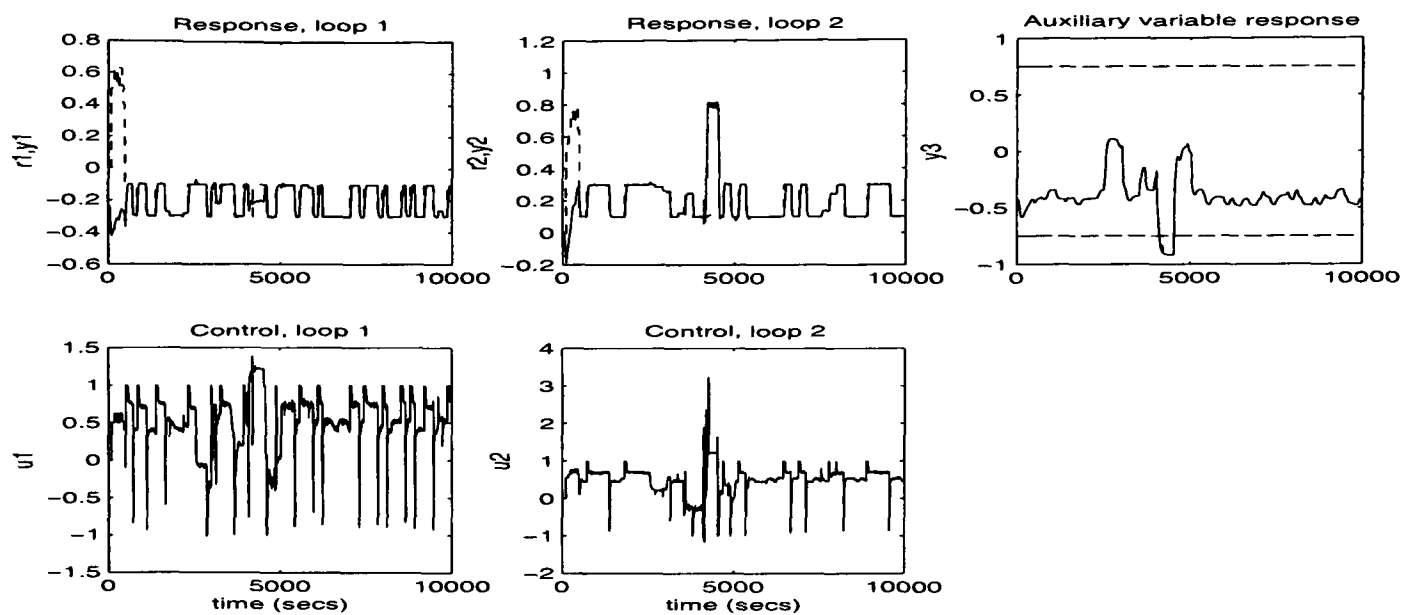


Figure 5.13: Adaptive MMGPC responses - full record

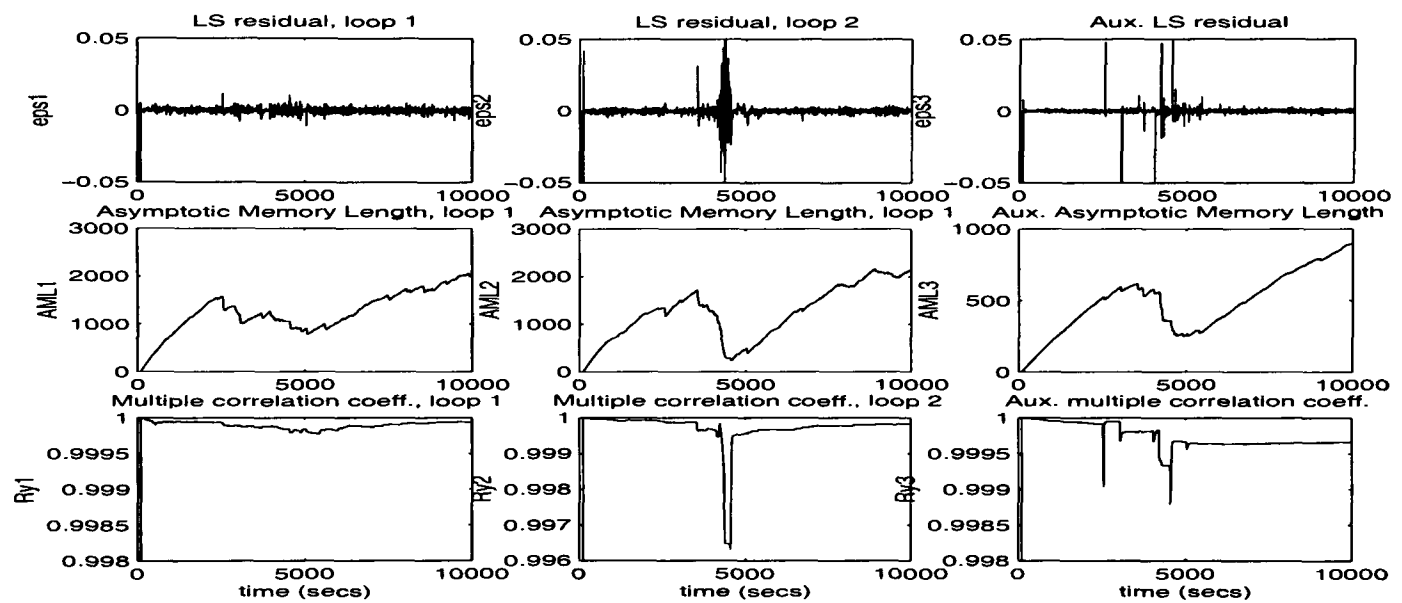


Figure 5.14: MMRLS estimator model fit indicators - full record

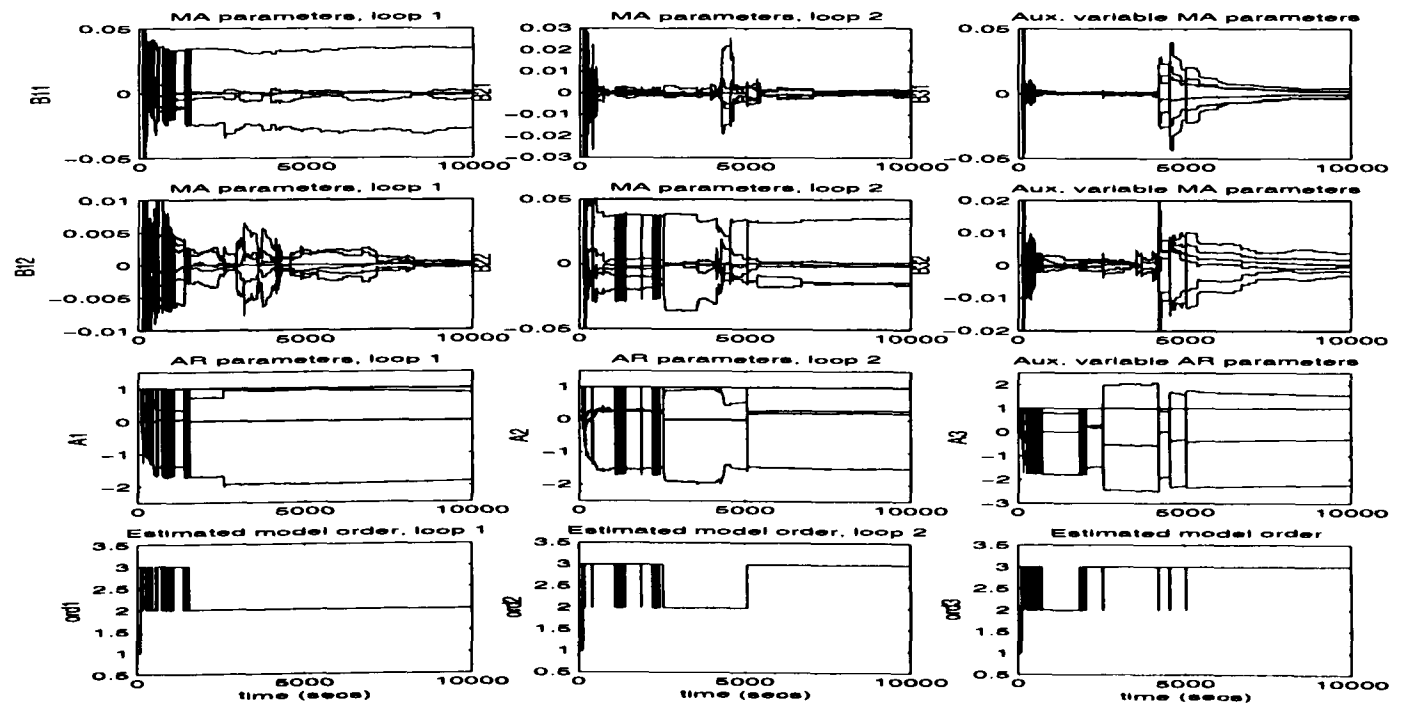


Figure 5.15: MMRLS parameter trajectories - full record

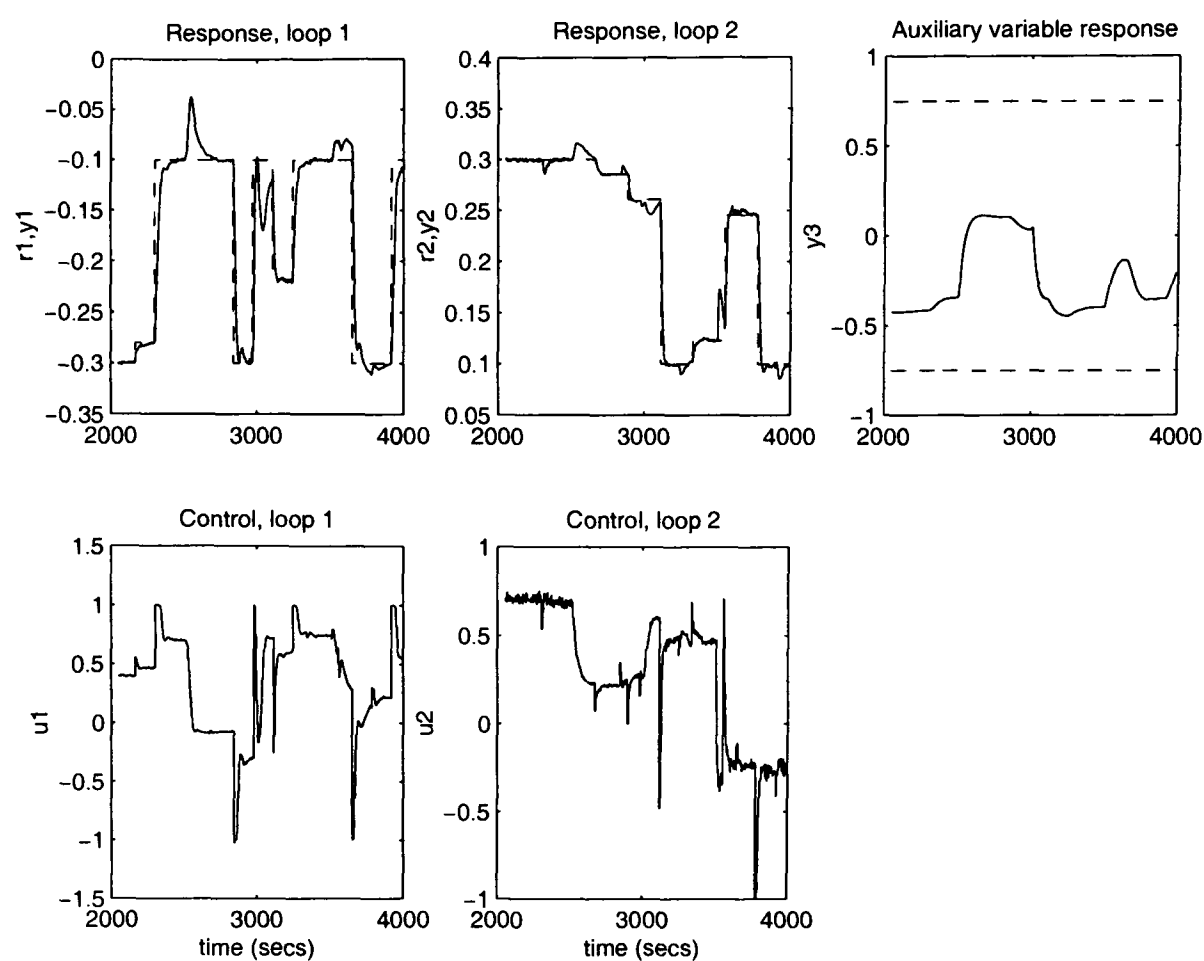


Figure 5.16: Self-tuned MMGPC response during disturbances

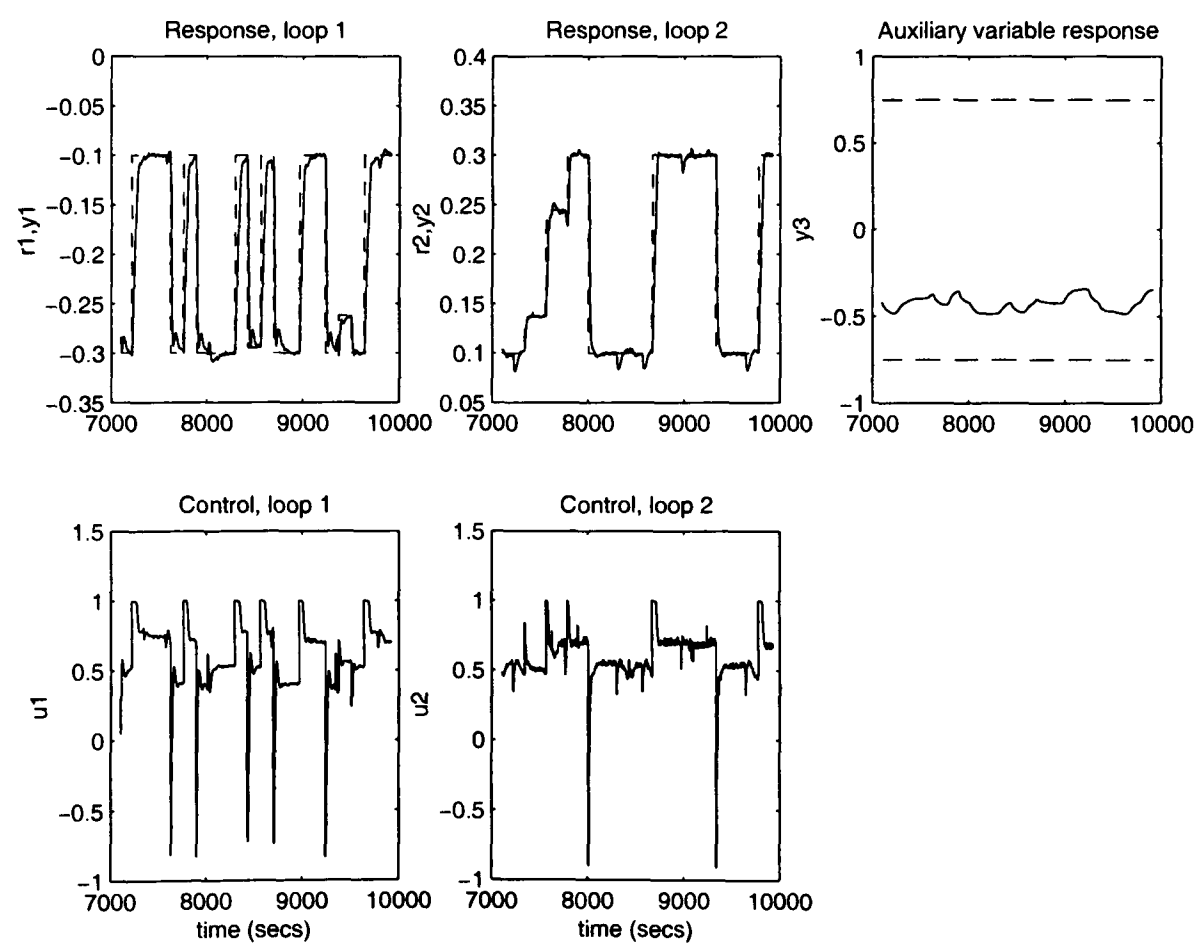


Figure 5.17: Self-tuned MMGPC response after disturbances

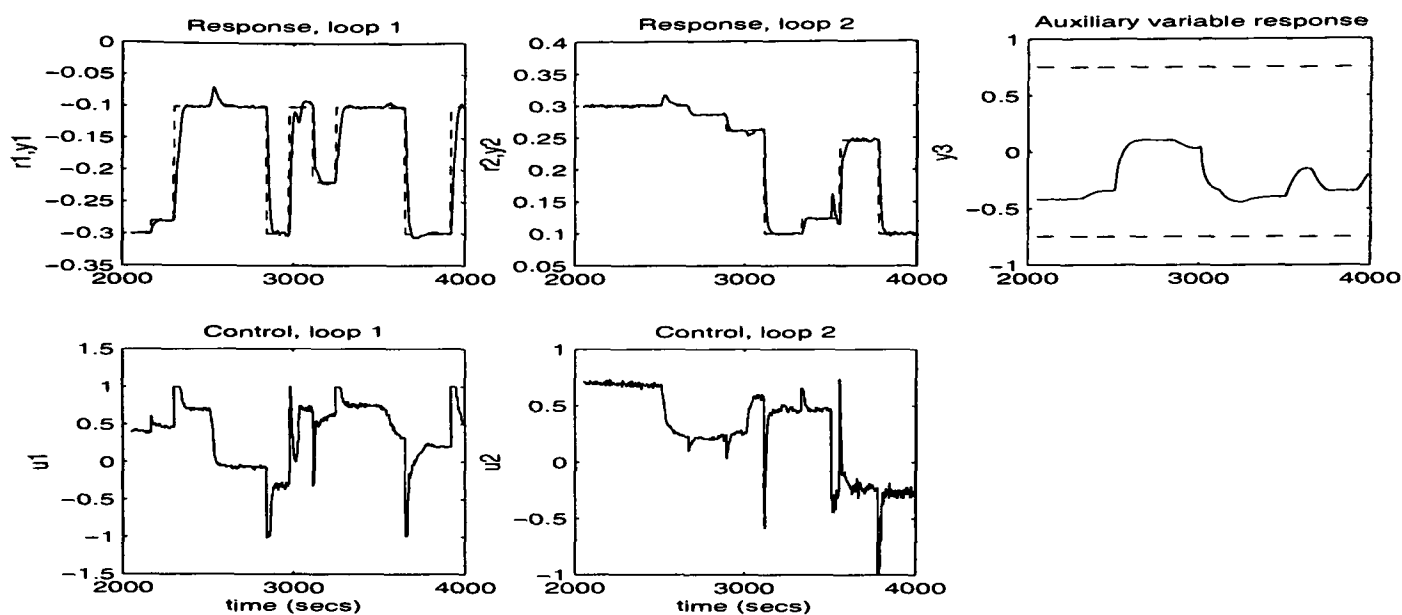


Figure 5.18: Adaptive MMGPC response during disturbances

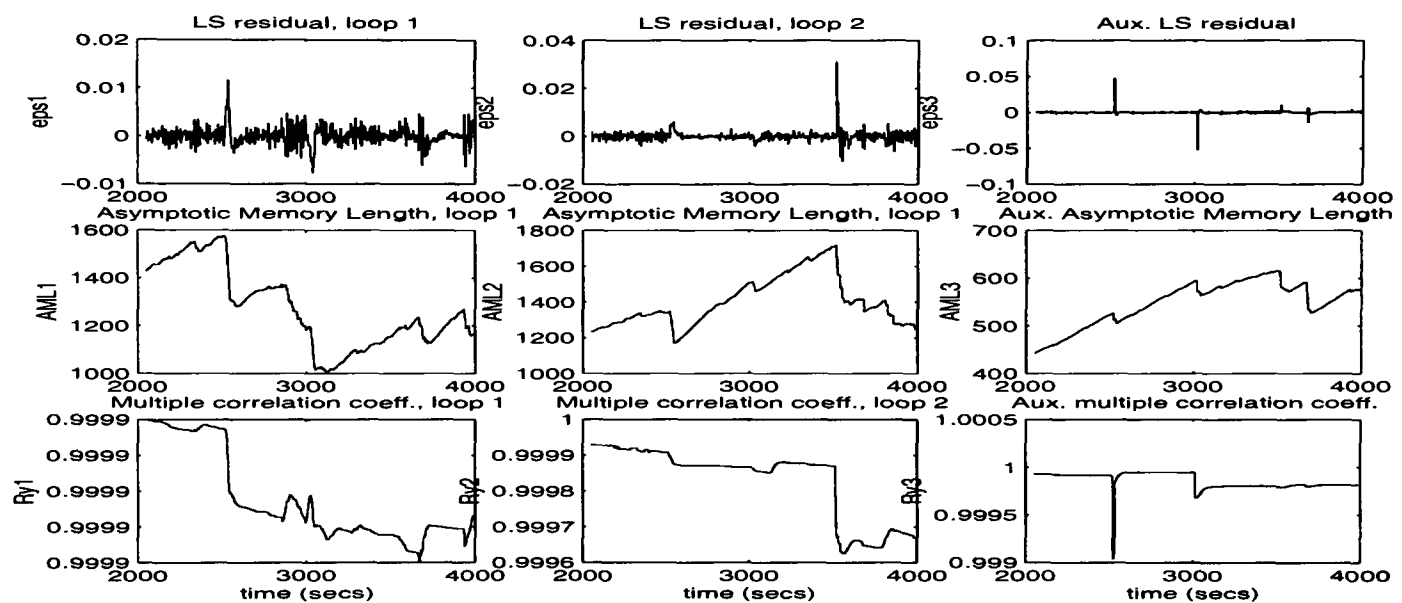


Figure 5.19: MMRLS model fit indicators during disturbances

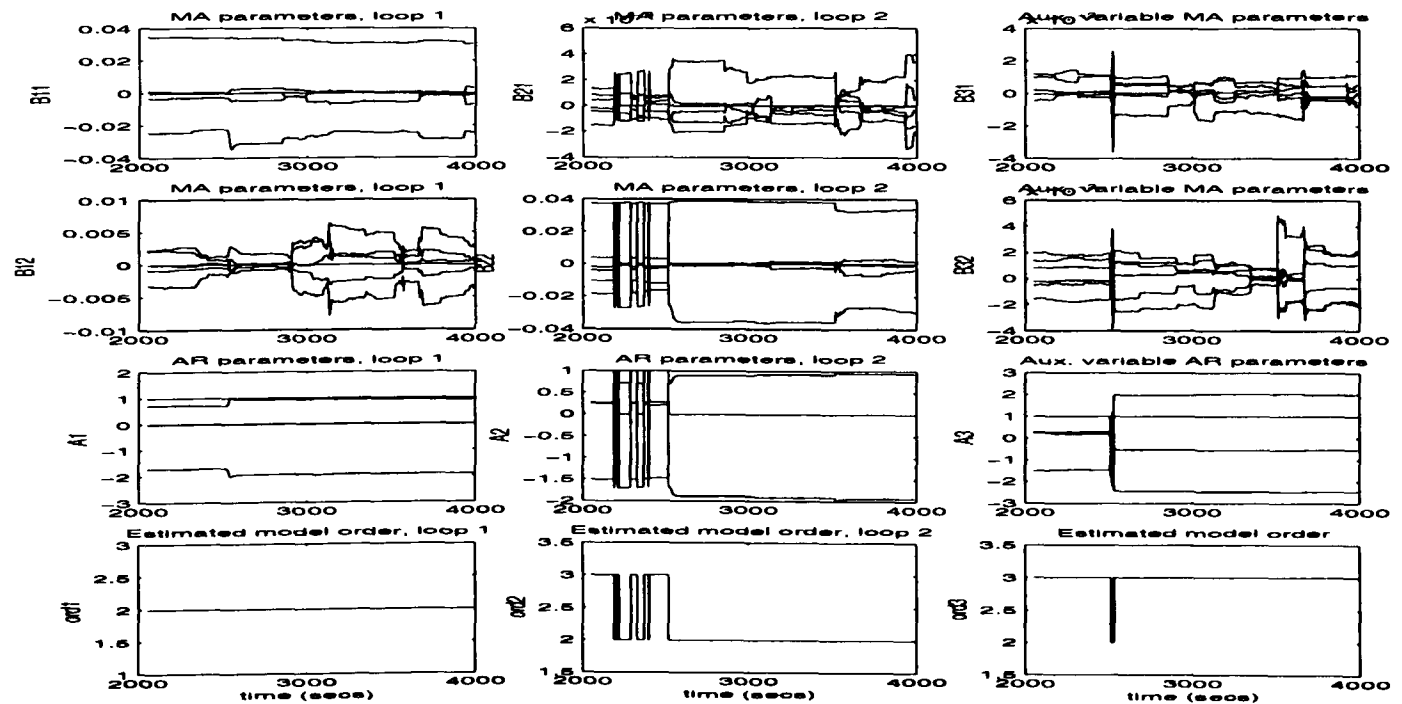


Figure 5.20: MMRLS parameter trajectories during disturbances

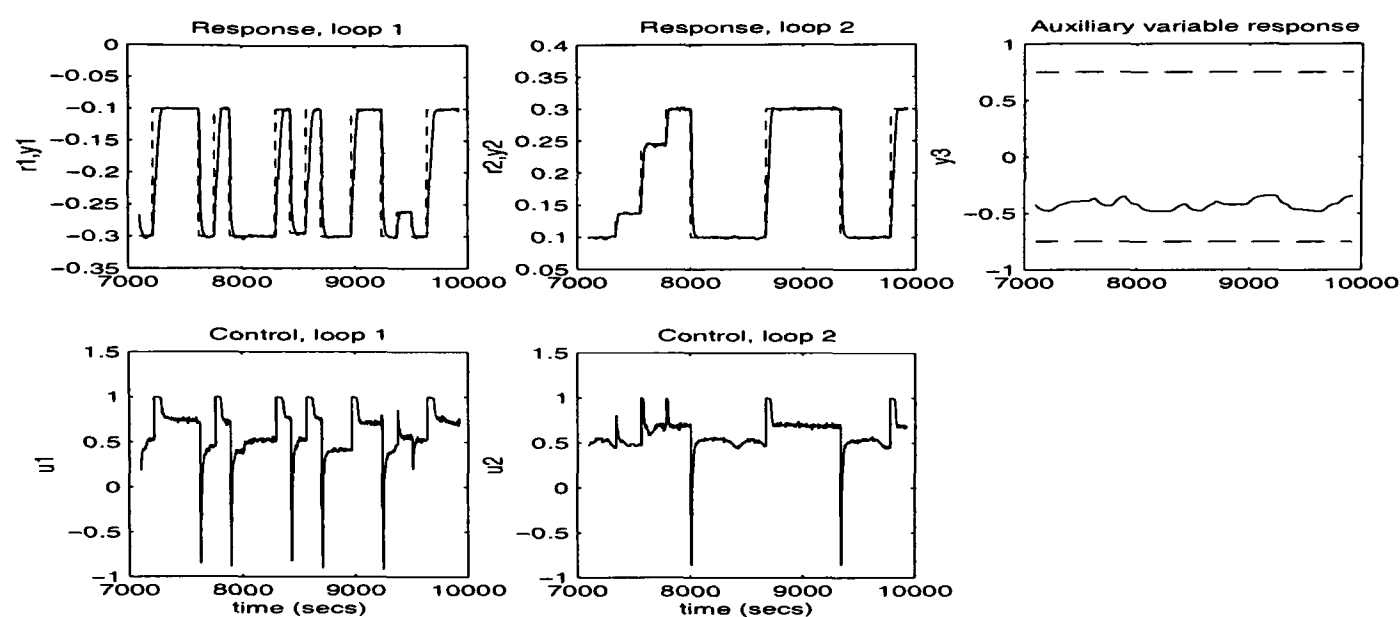


Figure 5.21: Adaptive MMGPC response after disturbances

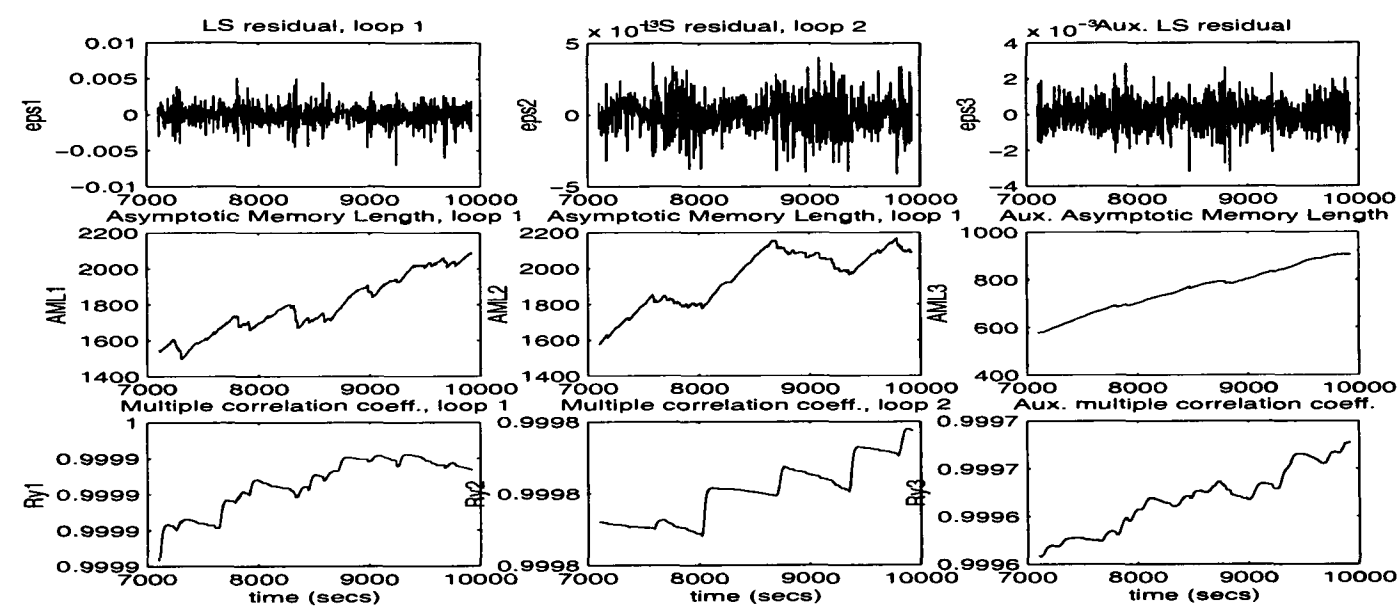


Figure 5.22: MMRLS model fit indicators after disturbances

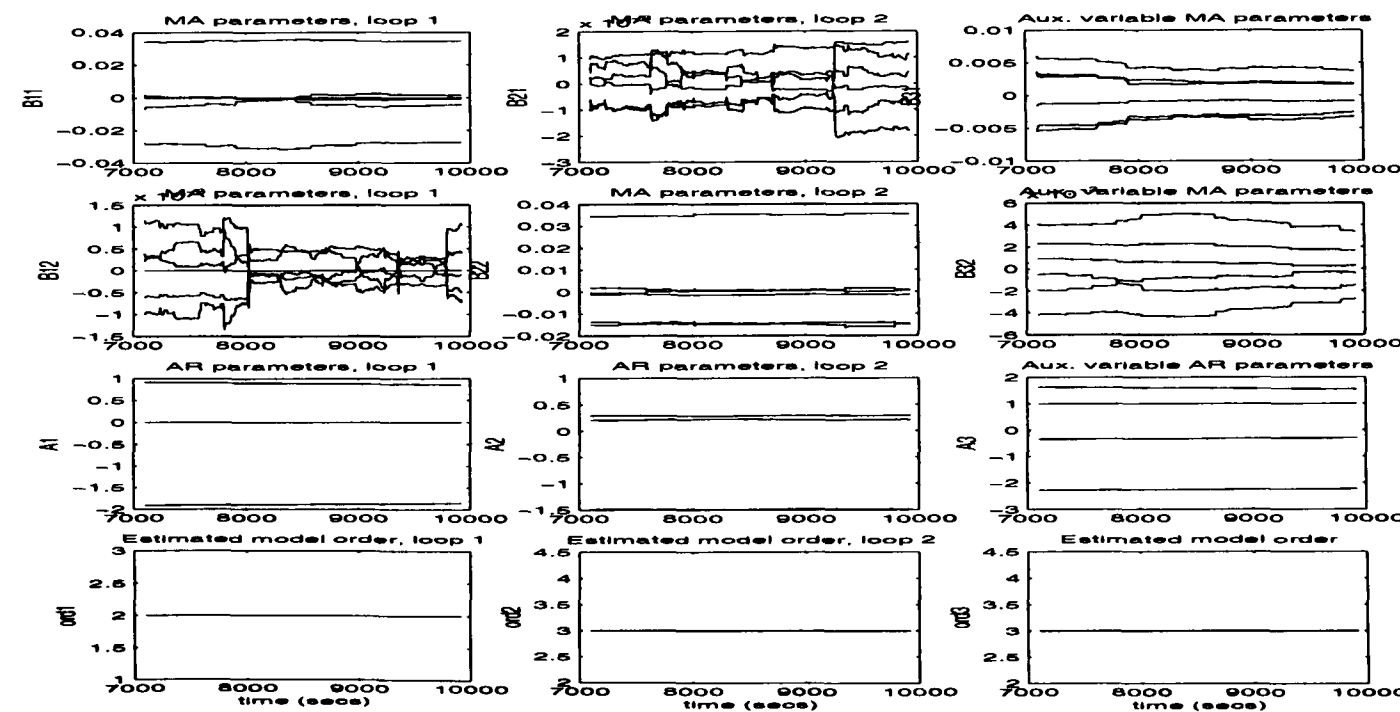


Figure 5.23: MMRLS parameter trajectories after disturbances

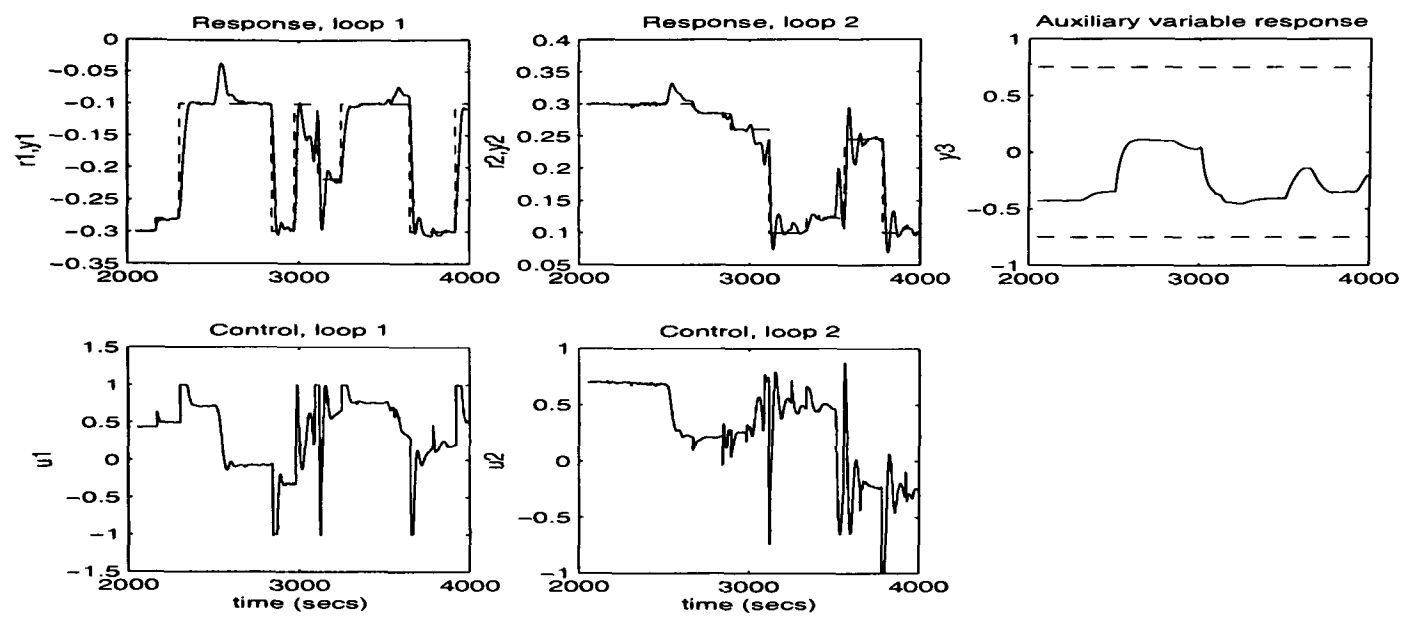


Figure 5.24: Adaptive 1GPC response during disturbances

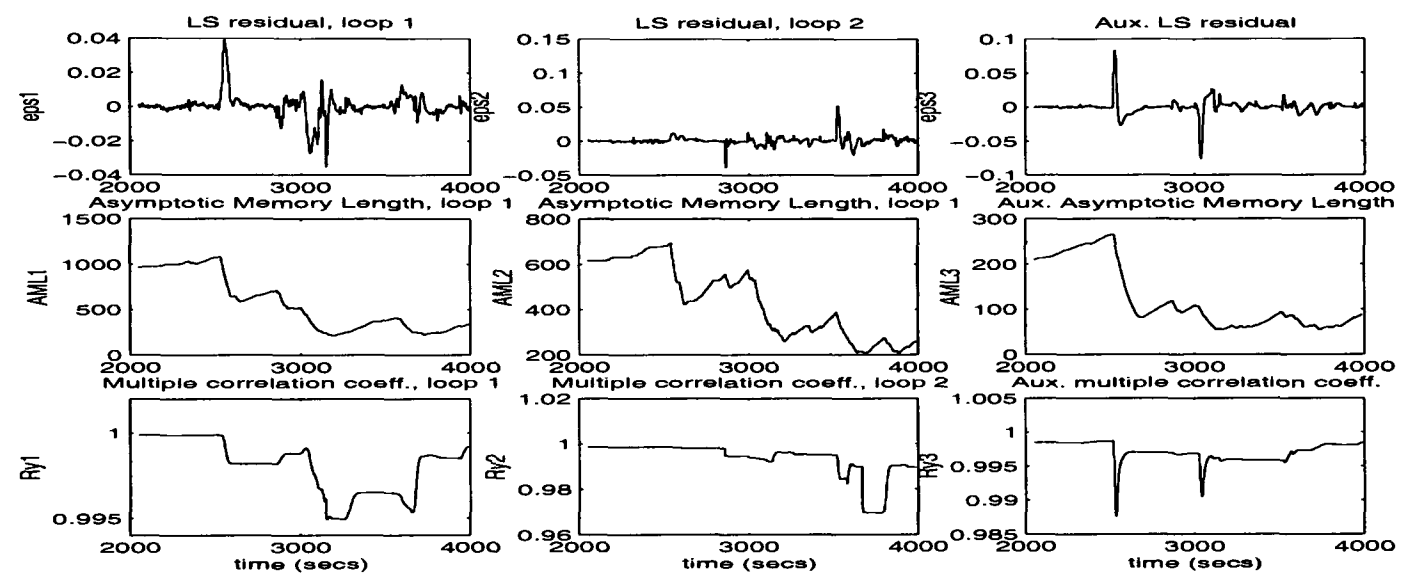


Figure 5.25: 1RLS model fit indicators during disturbances

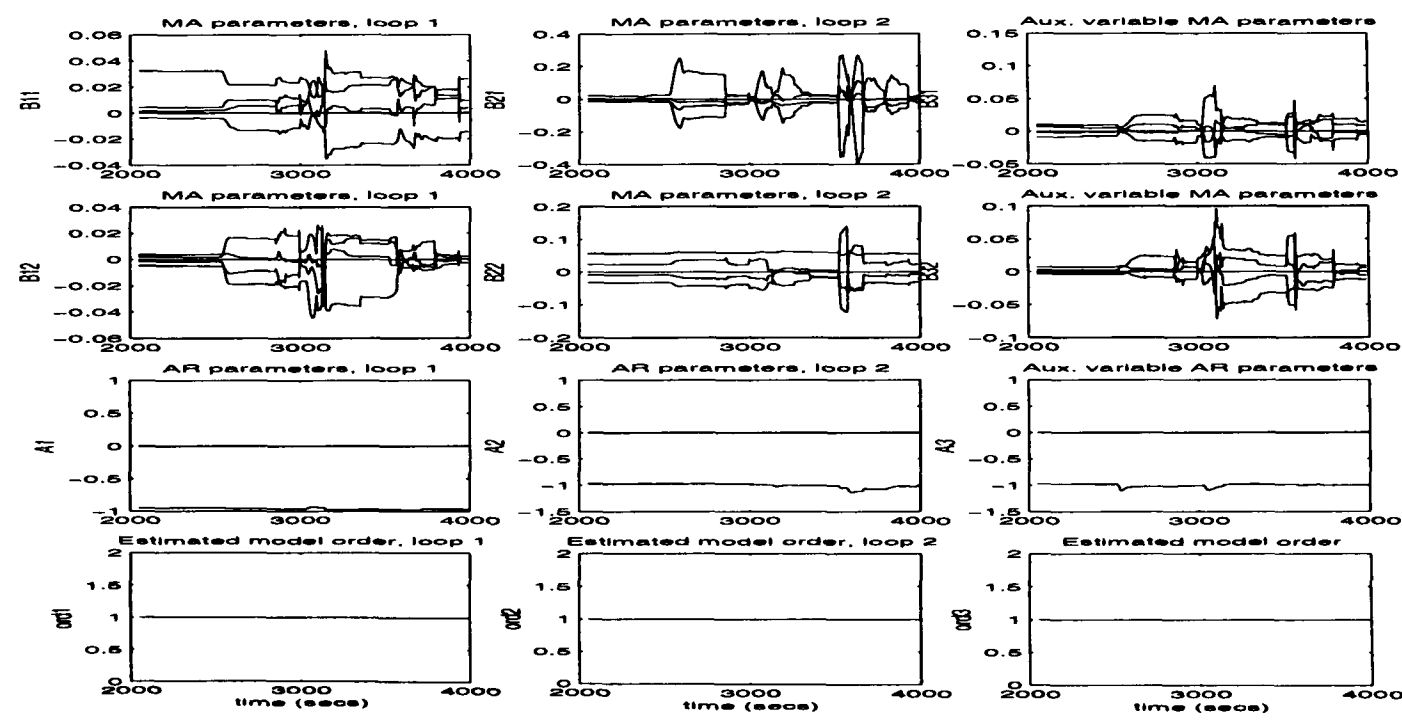


Figure 5.26: 1RLS parameter trajectories during disturbances

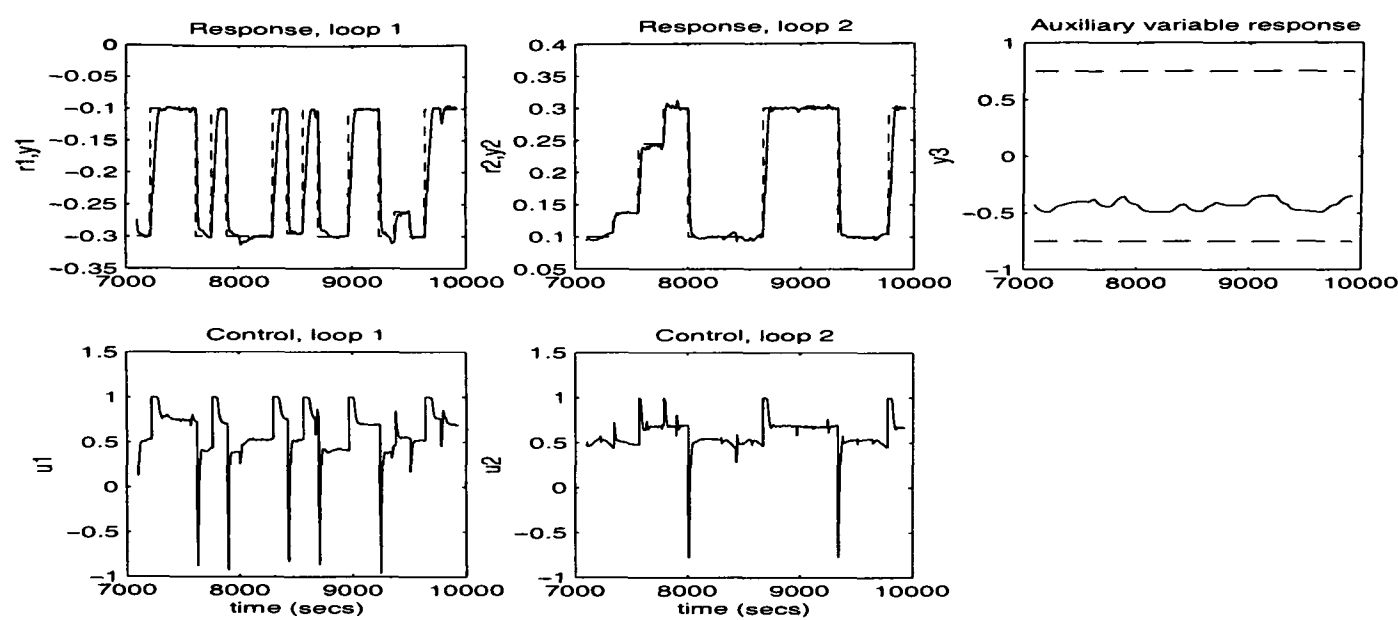


Figure 5.27: Adaptive 1GPC response after disturbances

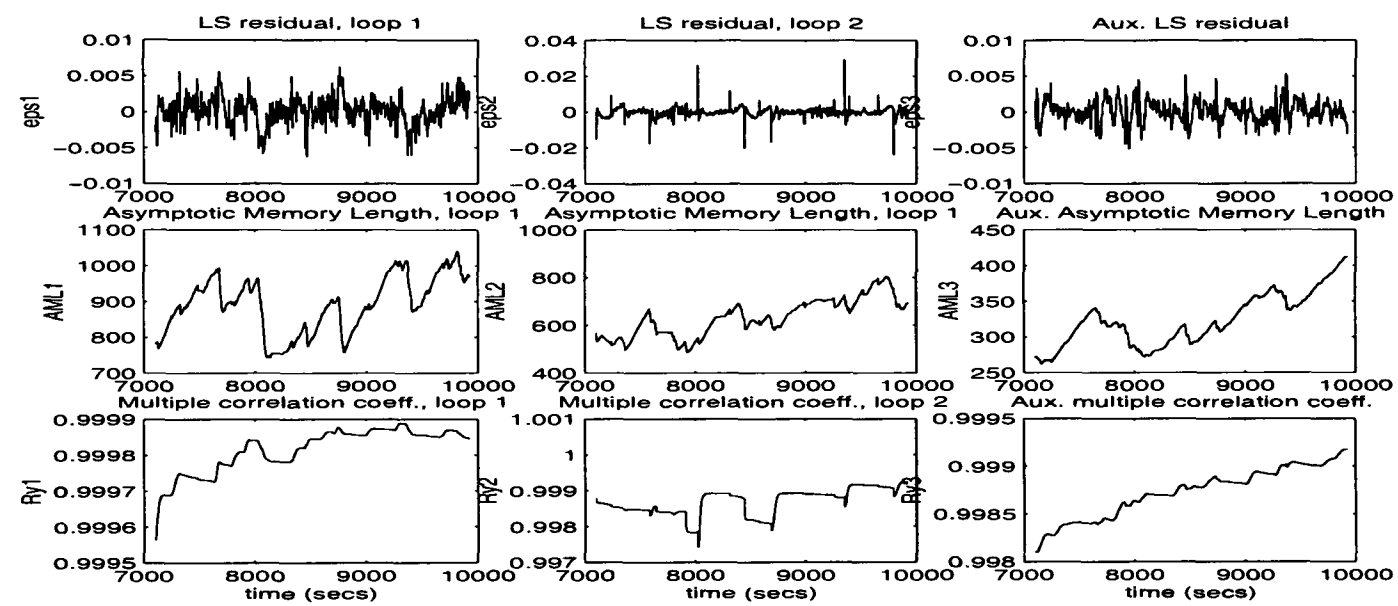


Figure 5.28: 1RLS model fit indicators after disturbances

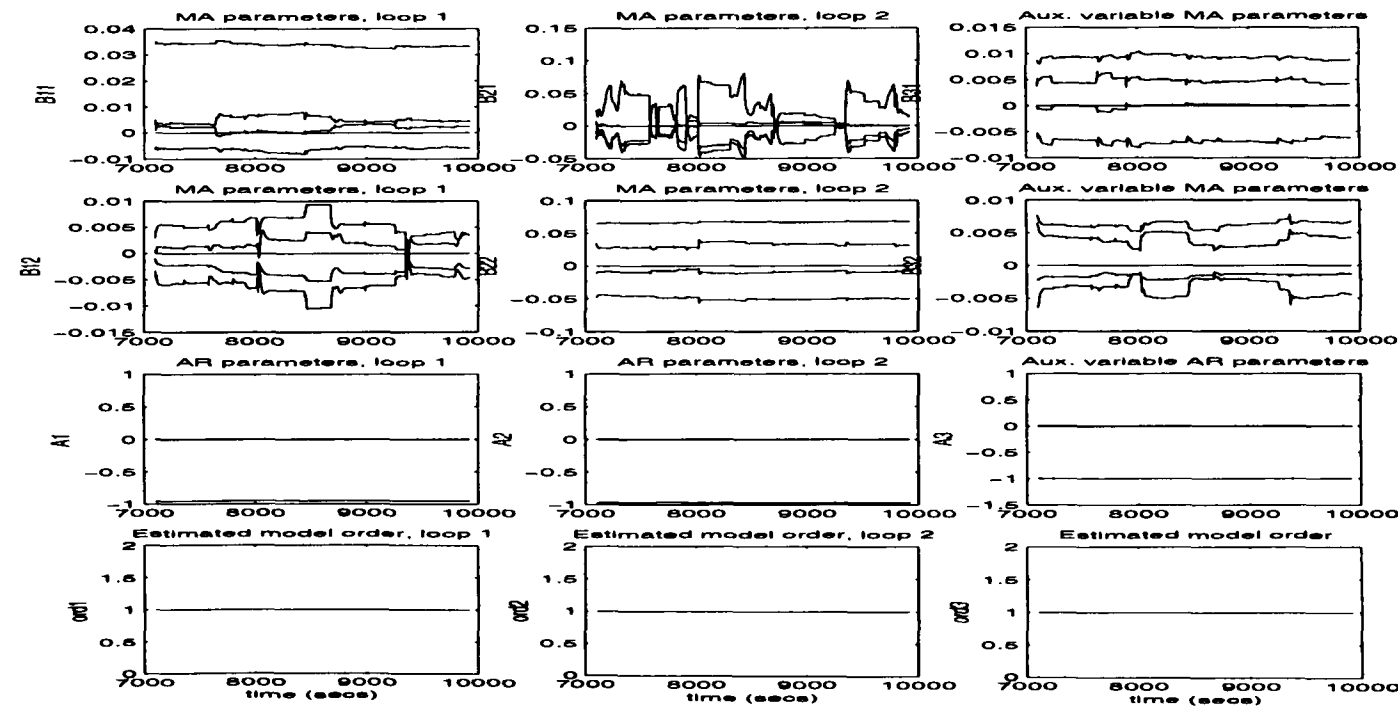


Figure 5.29: 1RLS parameter trajectories after disturbances

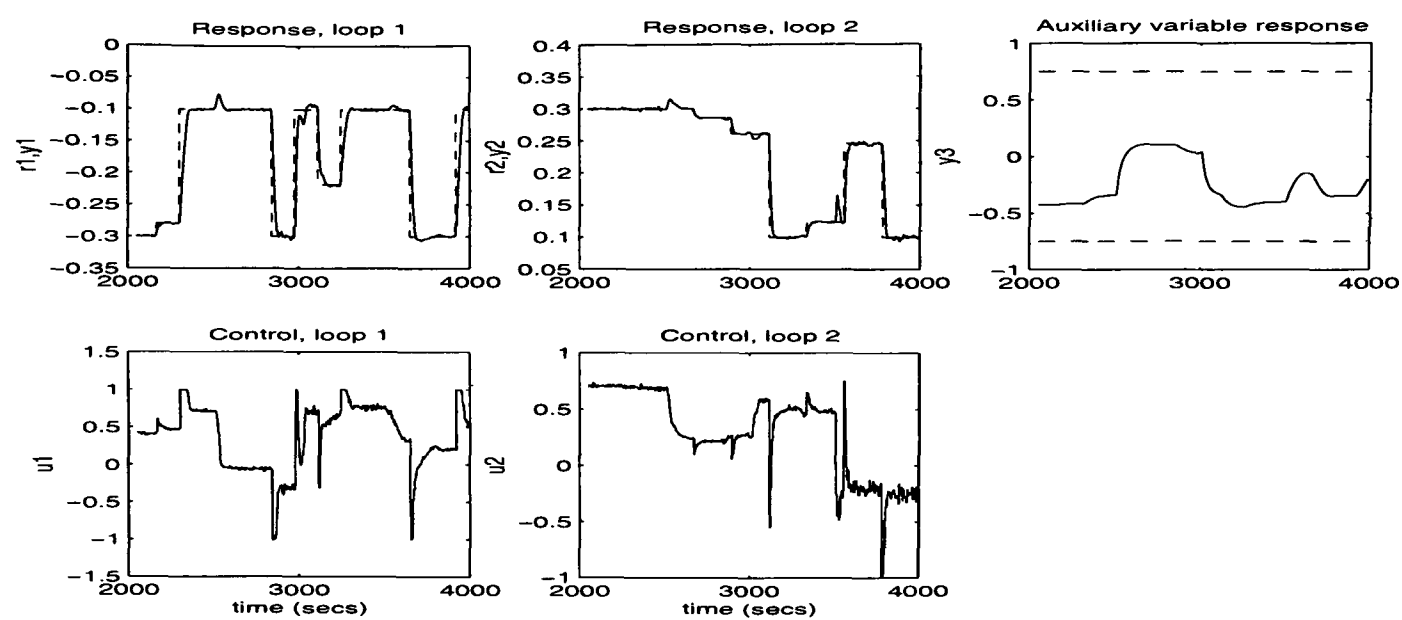


Figure 5.30: Adaptive 3GPC response during disturbances

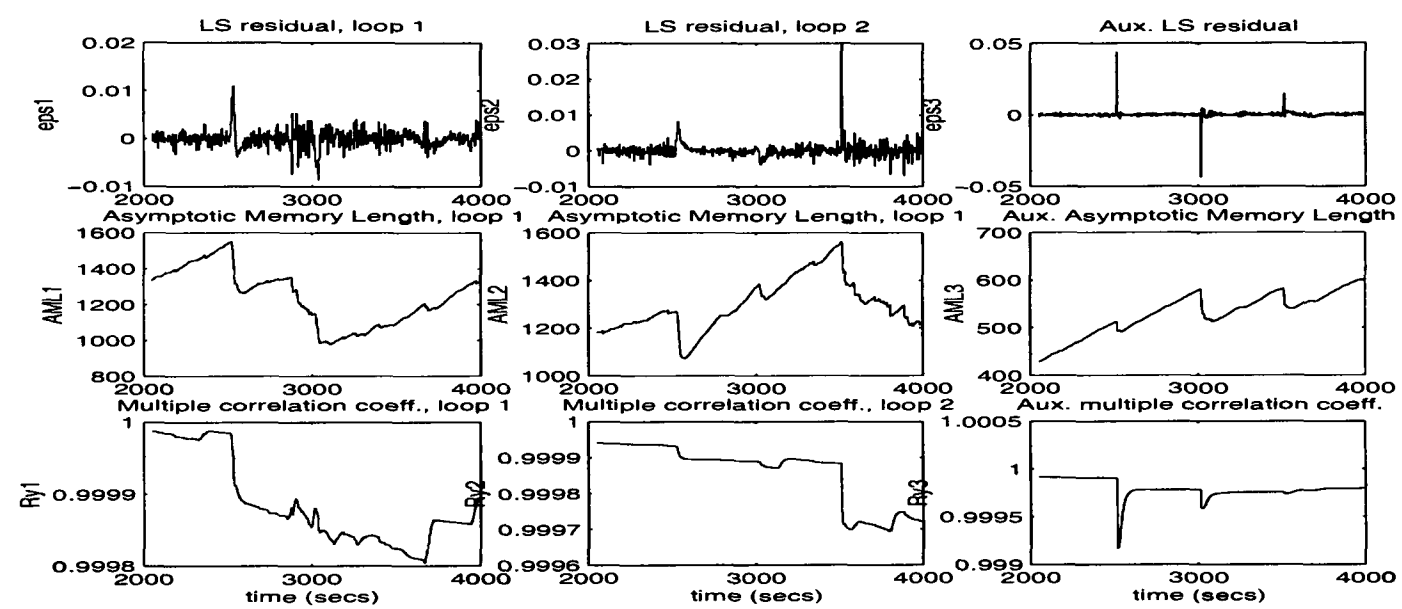


Figure 5.31: 3RLS model fit indicators during disturbances

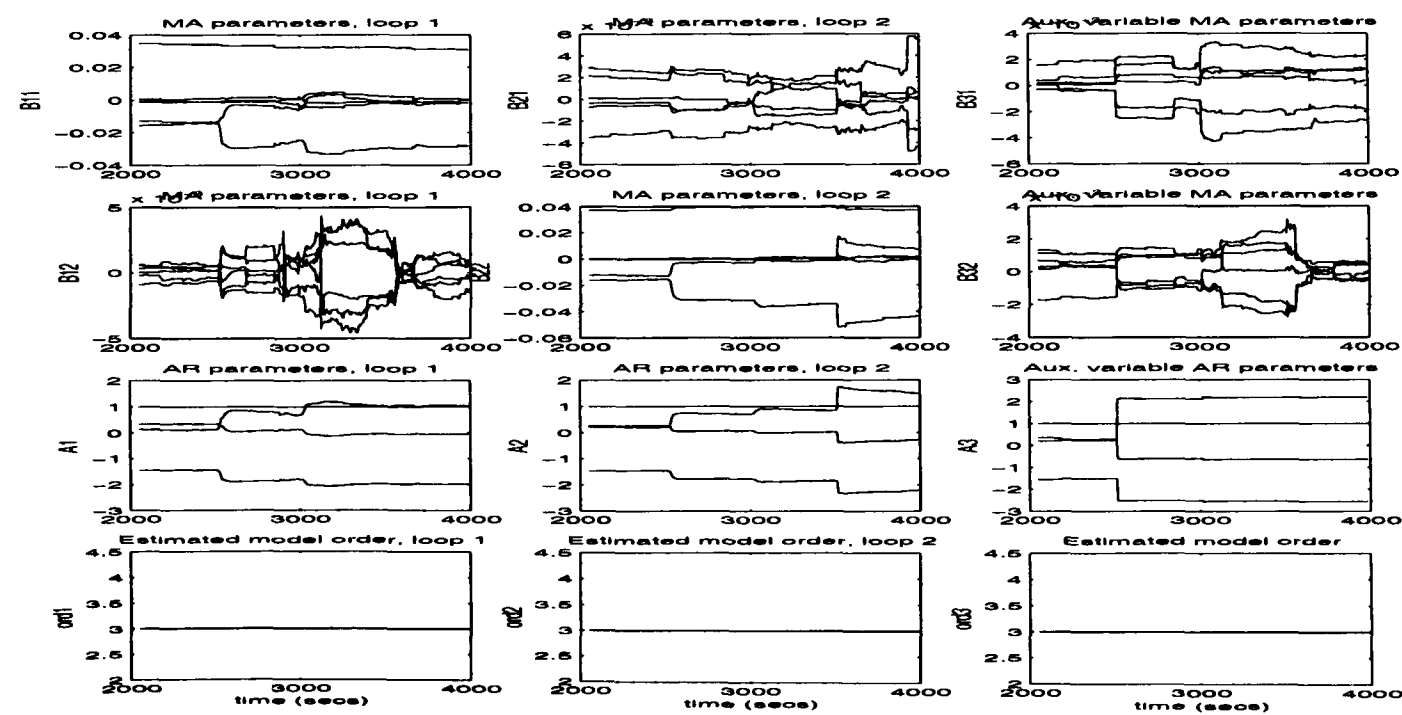


Figure 5.32: 3RLS parameter trajectories during disturbances

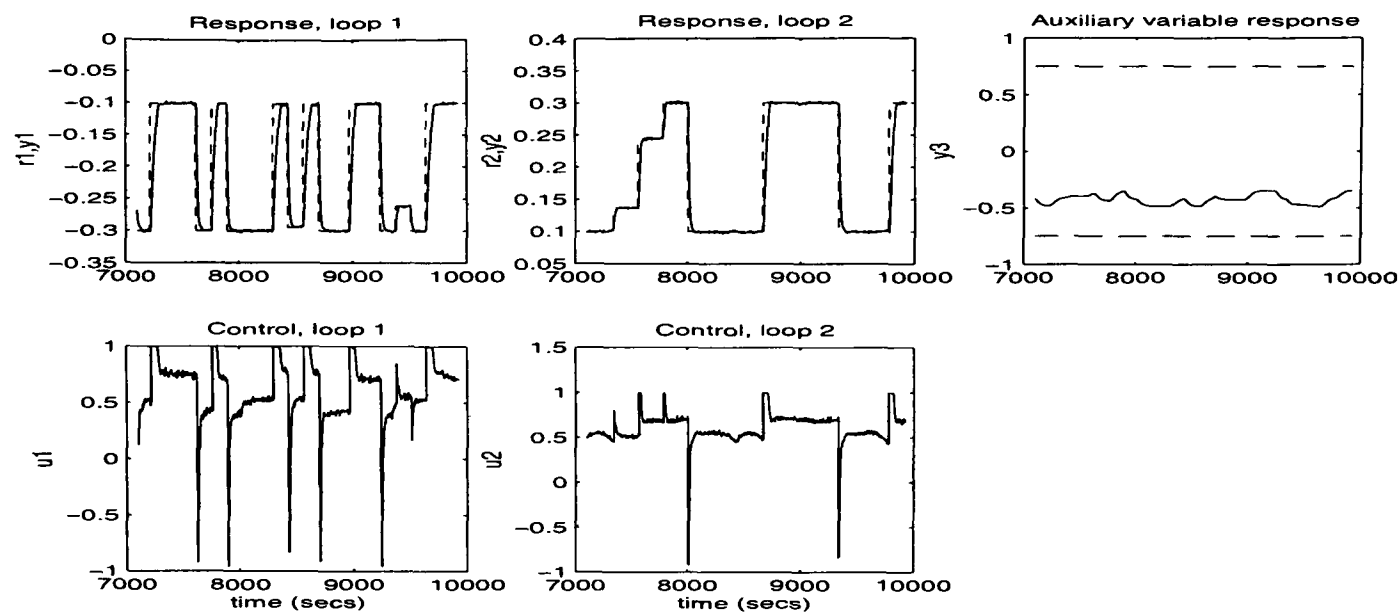


Figure 5.33: Adaptive 3GPC response after disturbances

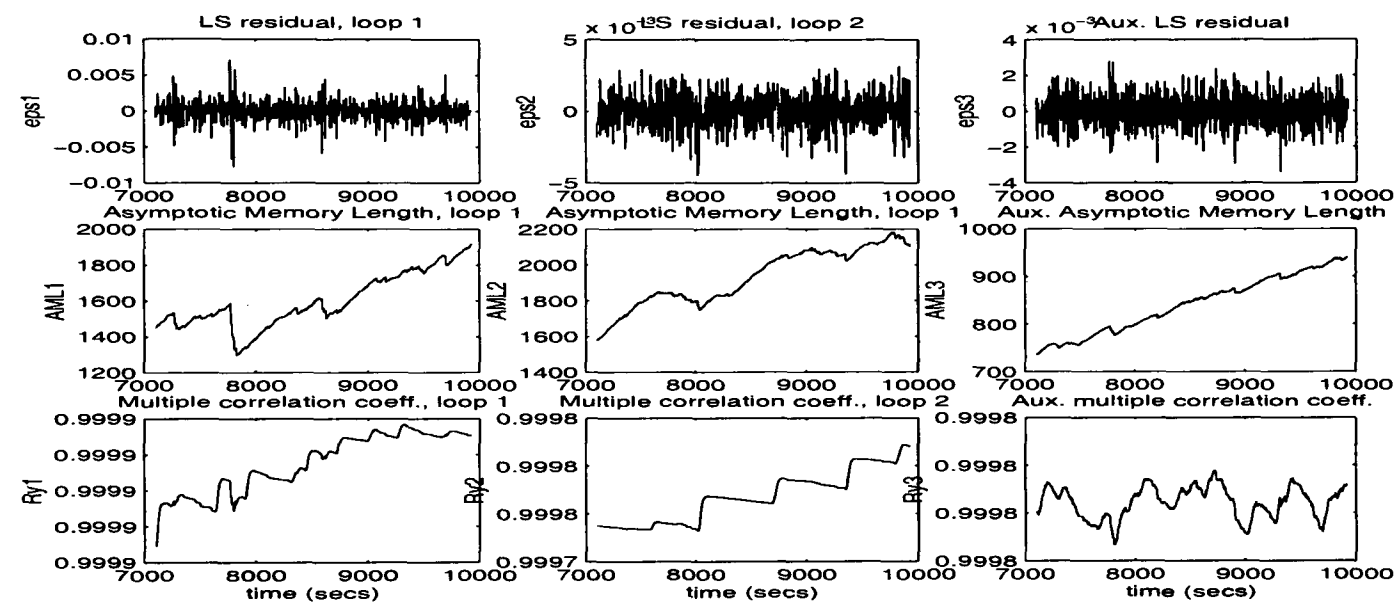


Figure 5.34: 3RLS model fit indicators after disturbances

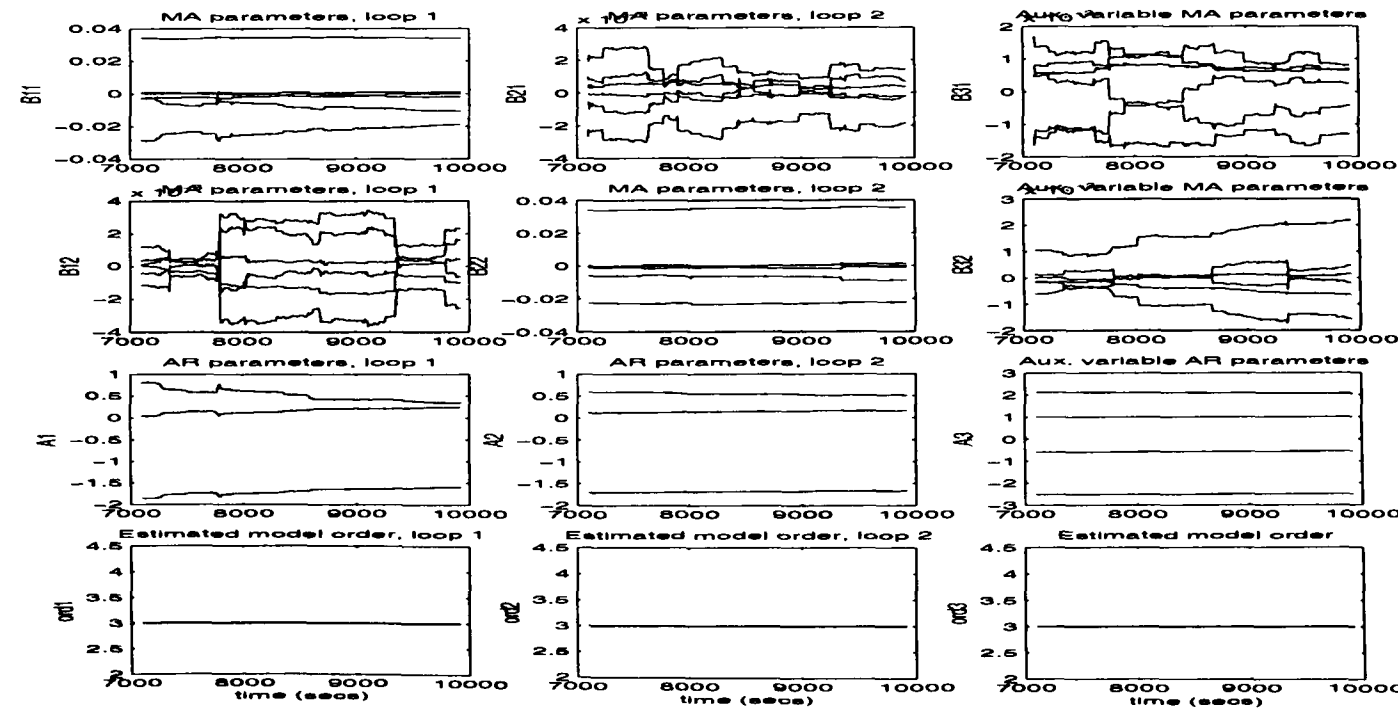


Figure 5.35: 3RLS parameter trajectories after disturbances

Chapter 6

Conclusions and future work

6.1 Conclusions

Most the of work in self-tuning control to date has assumed that the model of the plant is of fixed order. Since the order of the plant is not always known this often leads to problems associated with under/over-parameterisation of the assumed plant model with associated degradation of the model predictions and hence control. The multiple order model strategy presented in this thesis estimates both the parameters and the order of a linear model of the time-varying plant *on-line*. This was facilitated by the recent work of Niu and co-workers who have extended the method of UD updating so that, with only a small change to the UD update code of existing self-tuners controllers, a wealth of additional information can be obtained directly from the U and D matrices including estimates of all the lower order models and their loss functions. However, the multiple model method has been derived in this thesis using Clarke's Lagrange multiplier approach which differs from Niu's derivation, leading to possibly a more direct understanding of the *AUDZ* method. The *AUDZ* method has also been extended slightly to handle systems with unknown time delay.

This MMLS method is indeed very appealing as it uses the data contained in the (augmented) information matrix more *efficiently* compared to classical LS in that it provides the user with the parameter estimates for each model from order 0 to m *simultaneously*, with the associated loss functions, and is inherently numerically stable due to the square root factorisation involved. It has also be shown that the MMLS technique is the method of choice for obtaining reduced order models as it is similar to LSL algorithms for wide-sense

stationary AR modelling, and displays a number of advantages over Levinson's algorithm:

- **Numerically superior** due to the UD factorisation involved;
- **More general model structure:** MMLS is not restricted to the AR case but can be applied to ARX models with possibly variable time delay;
- **Accurate for small data sets** ($N \geq n$) as it is an extension of classical LS and hence the sample covariances are not required;
- **Amenable to variable forgetting** techniques in the recursive version.

On the other hand, since the Toeplitz structure is no longer available for exploitation, MMLS requires more computations in the batch (off-line) case.

In order to analyse and compare the behaviour of various discrete-time parameter tracking techniques based on RLS and LMS methods the continuous-time domain was adopted due to the simplicity of the resulting differential equations. It was shown that RLS with variable forgetting and regularisation gives robust and very efficient parameter tracking and hence it was selected as the preferred forgetting mechanism for application to self-tuning control.

The MMLS technique was then ported to the δ -domain for both batch and on-line applications. The δ -MMLS method is appealing as it is based on Middleton's δ -domain LS algorithm which has been shown to display good numerical and convergence properties when implemented on restricted word length platforms (e.g. embedded systems) using small sample times relative to the system's rise time. The simulation results have shown that the parameter estimates approach the continuous-time system differential equation coefficients as the sampling time tends to zero, enabling the user to monitor and bound those parameters that have some physical significance. In addition, it has been shown how the δ -MMLS method can be used to obtain the order and coefficients of ARX plant models for any continuous-time and/or δ -domain based MBPC control scheme.

Finally the MMRLS algorithm was coupled with multivariable GPC to give an adaptive predictive control scheme which adapts not only to the plant parameters, but also tries to select a parsimonious model order. The simulation results indicate that this multiple

model scheme gives similar performance to an optimal GPC scheme due to the adaptive mechanism. The experimental results show that better performance can be obtained using MMGPC than with a fixed low-order adaptive controller. However the results comparing the multiple model scheme with fixed high-order model adaptive controller were inconclusive due to the limitations of the rig. Nevertheless, given that MMGPC always gives as good performance as the high order controller or better, and since it has one less design parameter to choose (i.e. the model order) without adding much computational burden to the controller, there is a strong case for using MMGPC instead of fixed model order GPC.

6.2 Future work

The following highlights areas which still need to be considered:

MMLS initialisation. The work to date on AUDZ and MMLS has assumed that the initial parameters are all zero (apart from a_0 which is constrained to equal one). There have been no studies into how the $\mathbf{U}$ and $\mathbf{D}$ matrices should be initialised for known good starting model parameters. Thus given good parameters for a particular model order, what is the best way to choose the initial values of the higher and lower order models, and indeed the feedback controller models in such a case ?

Increasing the maximum order in the coupled tank experiment. As discussed in chapter 5, the maximum model order was limited to 3 by the dSPACE hardware. There might be a compiler option to optimise the code with respect to memory space (as opposed execution speed) so that higher order models can be tried out in an attempt to show inferior performance due to process over-parameterisation.

Estimation of time-delay for δ -MMLS. The problem with sampling a process fast with respect to the process bandwidth is that estimating the time-delay becomes increasingly more complex and time-consuming. For the δ -domain approach to be an attractive domain for system modelling, a method must be developed to give a good estimate of the system's time-delay. The δ -MMLS technique can then be used to provide multiple order δ -domain models. The bootstrap method of Elnaggar et al.

(1989) would be a good starting point as it estimates the time delay explicitly whilst using a parameter vector of the same length as for the undelayed system.

Experimental comparison between MMLS and δ -MMLS. Since the dynamics of the coupled tanks system are relatively slow (of the order minutes) compared to the maximum sampling time of the dSPACE card (milliseconds) the coupled tanks rig could be used to compare the performance of modelling in the z -domain versus modelling in the δ -domain with respect to ease of model order selection and ability to estimate the physical system parameters, e.g. the cross-sectional area of the pipe joining two tanks.

Automatic prediction horizon selection. Ideally the prediction horizon should be chosen according to our confidence in the estimated model such that:

- As the model quality $Q(t) \rightarrow 1$ then $N_2(t) \rightarrow \infty$, which is the GPC $^\infty$ of Sokaert (1997).
- As the model quality $Q(t) \rightarrow 0$ then $N_2(t) \rightarrow N_1(t) = \hat{\tau}(t)$, which is Minimum Variance (MV) control (Åström, 1970).

One way of achieving this would be to use a control dual of Fortescue's variable forgetting-factor for parameter estimation, i.e.

$$N_2(t) = \frac{N_0 \sigma_0^2}{1 - Q(t)}$$

where N_0 is the prediction horizon for perfect modelling and when the noise is Gaussian distributed with variance σ_0^2 and zero mean. Since $J_{opt} = T_c \sigma_0^2$ (from the estimator variable forgetting routine) we can further write:

$$N_2(t) = \frac{\kappa J_{opt}}{1 - Q(t)}$$

where $0 < \kappa < 1$ determines the ratio of prediction horizon (future) to asymptotic memory length (past).

Appendix A

Change of basis in the regressor space

Given the standard linear regression model:

$$y(t) = \boldsymbol{\theta}^T \mathbf{x}(t),$$

we can change the basis of the regressor space (i.e. the space spanned by the data $\mathbf{x}(t)$) using an invertible transformation matrix $\mathbf{T}$ giving a new regression vector:

$$\tilde{\mathbf{x}}(t) = \mathbf{T}\mathbf{x}(t)$$

with associated parameters:

$$\tilde{\boldsymbol{\theta}} = \mathbf{T}^{-T} \boldsymbol{\theta}$$

and thus the model can be written as:

$$y(t) = \boldsymbol{\theta}^T \mathbf{x}(t) = (\mathbf{T}^T \tilde{\boldsymbol{\theta}})^T \mathbf{T}^{-1} \tilde{\mathbf{x}}(t) = \tilde{\boldsymbol{\theta}}^T \tilde{\mathbf{x}}(t).$$

The LS estimate of $\tilde{\boldsymbol{\theta}}$ is thus given by:

$$\hat{\tilde{\boldsymbol{\theta}}}(t) = \tilde{\mathbf{S}}^{-1}(t) \tilde{\mathbf{s}}_{xy}(t)$$

where

$$\tilde{\mathbf{S}}(t) = \mathbf{T}\mathbf{S}(t)\mathbf{T}^T \quad \text{with} \quad \mathbf{S}(t) = \sum_{k=1}^t \mathbf{x}(k)\mathbf{x}^T(k) \quad (\text{A.1})$$

and

$$\tilde{\mathbf{s}}_{xy}(t) = \mathbf{T}\mathbf{s}_{xy} \quad \text{with} \quad \mathbf{s}_{xy} = \sum_{k=1}^t \mathbf{x}(k)y(k),$$

i.e.

$$\hat{\hat{\boldsymbol{\theta}}}(t) = \mathbf{T}^{-T} \hat{\boldsymbol{\theta}}(t) = \mathbf{T}^{-T} \mathbf{S}^{-1}(t) \mathbf{s}_{xy}(t).$$

Thus a change of basis in the regressor space has some straight-forward consequences for the estimates.

Appendix B

Bierman's recursive U-D estimator

The standard RLS algorithm is numerically ill-conditioned, and implementation using these equations directly often leads to the covariance matrix $\mathbf{P}$ losing positive definiteness and then to inaccurate results. Since $\mathbf{P}$ *must* be positive definite, the common solution is to use a factored form of $\mathbf{P}$ and update the factors rather than the full matrix. One approach is the $\mathbf{UDU}^T$ form:

$$\mathbf{P}(t) = \mathbf{U}(t)\mathbf{D}(t)\mathbf{U}^T(t),$$

where $\mathbf{D}$ is a diagonal matrix and $\mathbf{U}$ is an upper-triangular matrix with unit entries along the diagonal. Defining $\hat{\mathbf{U}} = \mathbf{U}(t)$, $\bar{\mathbf{U}} = \mathbf{U}(t-1)$, $\hat{\mathbf{D}} = \mathbf{D}(t)$ and $\bar{\mathbf{D}} = \mathbf{D}(t-1)$ the covariance update equation can be written:

$$\hat{\mathbf{U}}\hat{\mathbf{D}}\hat{\mathbf{U}}^T = \frac{1}{\beta}\bar{\mathbf{U}}\left[\bar{\mathbf{D}} - \frac{\bar{\mathbf{D}}\bar{\mathbf{U}}^T\mathbf{x}\mathbf{x}^T\bar{\mathbf{U}}\bar{\mathbf{D}}}{\beta + \mathbf{x}^T\bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{U}}^T\mathbf{x}}\right]\bar{\mathbf{U}}^T,$$

and defining the vectors $\mathbf{f} = \bar{\mathbf{U}}^T\mathbf{x}$, $\mathbf{g} = \bar{\mathbf{D}}\mathbf{f}$ we get the RLS equations using U-D factors:

$$\epsilon(t) = \frac{y(t) - \hat{\boldsymbol{\theta}}^T(t-1)\mathbf{x}(t)}{1 + \mathbf{f}^T\mathbf{g}} \quad (\text{Prediction error})$$

$$\mathbf{k}(t) = \frac{\bar{\mathbf{U}}\mathbf{g}}{\beta + \mathbf{f}^T\mathbf{g}} \quad (\text{Kalman gain})$$

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \mathbf{k}(t)\epsilon(t) \quad (\text{Parameter update})$$

$$\hat{\mathbf{U}}\hat{\mathbf{D}}\hat{\mathbf{U}}^T = \frac{1}{\beta}\bar{\mathbf{U}}\left[\bar{\mathbf{D}} - \frac{\mathbf{g}\mathbf{g}^T}{\beta + \mathbf{f}^T\mathbf{g}}\right]\bar{\mathbf{U}}^T \quad (\text{Covariance update})$$

B.1 Derivation of the recursive U-D filter

If the term within the brackets can be decomposed as $\mathbf{U}_n \mathbf{D}_n \mathbf{U}_n^T$ then the covariance matrix update is simply:

$$\hat{\mathbf{U}} \hat{\mathbf{D}} \hat{\mathbf{U}}^T = (\bar{\mathbf{U}} \mathbf{U}_n) \frac{\mathbf{D}_n}{\beta} (\mathbf{U}_n^T \bar{\mathbf{U}}^T).$$

Thus the basic problem reduces to factorising:

$$\bar{\mathbf{D}} - \frac{\mathbf{g}\mathbf{g}^T}{\beta + \mathbf{f}^T \mathbf{g}} \quad \text{into:} \quad \mathbf{U}_n \mathbf{D}_n \mathbf{U}_n^T$$

i.e. we need to find d_i and $\mathbf{u}^{(i)}$ such that

$$\sum_{i=1}^n d_i \mathbf{u}^{(i)} \mathbf{u}^{(i)T} - \sum_{i=1}^n \bar{d}_i \mathbf{e}_i \mathbf{e}_i^T + \mathbf{g}\mathbf{g}^T/s_n = \mathbf{0}, \quad (\text{B.1})$$

where

$$\begin{aligned} d_i &= i^{\text{th}} \text{ diagonal element of } \mathbf{D}_n \\ \mathbf{g}^{(i)} &= [g_1 \ g_2 \ \dots \ g_i \ 0 \ \dots \ 0]^T \quad \text{i.e. } \mathbf{g} = \mathbf{g}^{(n)} \\ \mathbf{e}_i &= i^{\text{th}} \text{ unit-vector} \\ \mathbf{U}_n &= [\mathbf{u}^{(1)} \ \mathbf{u}^{(2)} \ \dots \ \mathbf{u}^{(n)}], \text{ with } u_i^{(i)} = 1 \\ s_j &= \beta + \mathbf{f}^{(j)T} \mathbf{g}^{(j)} = \beta + f_1 g_1 + \dots + f_j g_j \quad (g_i = \bar{d}_i f_i). \end{aligned}$$

Starting with $i = n$ we get that:

$$\mathbf{M}^{(n)} = d_n \mathbf{u}^{(n)} \mathbf{u}^{(n)T} - \bar{d}_n \mathbf{e}_n \mathbf{e}_n^T + \mathbf{g}\mathbf{g}^T/s_n,$$

which must have zero entries along the bottom row and right-hand column since $\mathbf{u}^{(i)} \mathbf{u}^{(i)T}$ has zero elements past row/column i . Using the fact that $u_n^{(n)} = 1$ the bottom-right most entry gives:

$$\begin{aligned} 0 &= d_n - \bar{d}_n + g_n^2/s_n, \\ &= s_n(d_n - \bar{d}_n) + g_n^2 = s_n(d_n - \bar{d}_n) + g_n f_n \bar{d}_n, \end{aligned}$$

i.e.

$$s_n d_n = s_{n-1} \bar{d}_n.$$

The rest of the bottom row entries ($i < n$) must satisfy:

$$d_n u_i^{(n)} + g_i g_n / s_n = 0$$

which gives

$$u_i^{(n)} = -\frac{g_i g_n}{d_n s_n} = -\frac{g_i f_n \bar{d}_n}{s_{n-1} \bar{d}_n} = -\frac{g_i f_n}{s_{n-1}}.$$

Thus by choosing:

$$d_n = \frac{s_{n-1}}{s_n} \bar{d}_n,$$

and

$$\mathbf{u}^{(n)} = \mathbf{e}_n - \frac{f_n}{s_{n-1}} \mathbf{g}^{(n-1)},$$

the matrix $\mathbf{M}^{(n)}$ has zeros along the bottom row and right-hand column and thus

$$\begin{aligned} \mathbf{M}^{(n)} &= d_n \left(\frac{-f_n}{s_{n-1}} \mathbf{g}^{(n-1)} \right) \left(\frac{-f_n}{s_{n-1}} \mathbf{g}^{(n-1)T} \right) + \mathbf{g}^{(n-1)} \mathbf{g}^{(n-1)T} / s_n \\ &= \left(\frac{d_n f_n^2}{s_{n-1}^2} + \frac{1}{s_n} \right) \mathbf{g}^{(n-1)} \mathbf{g}^{(n-1)T} = \left(\frac{g_n f_n + s_{n-1}}{s_n s_{n-1}} \right) \mathbf{g}^{(n-1)} \mathbf{g}^{(n-1)T} \\ &= \mathbf{g}^{(n-1)} \mathbf{g}^{(n-1)T} / s_{n-1}. \end{aligned}$$

Hence the problem in eqn.(B.1) has been reduced to:

$$\sum_{i=1}^{n-1} d_i \mathbf{u}^{(i)} \mathbf{u}^{(i)T} - \sum_{i=1}^{n-1} \bar{d}_i \mathbf{e}_i \mathbf{e}_i^T + \mathbf{g}^{(n-1)} \mathbf{g}^{(n-1)T} / s_{n-1} = \mathbf{0},$$

which is exactly the same as the original problem with n replaced by $n-1$. Thus the $\mathbf{U}_n \mathbf{D}_n \mathbf{U}_n^T$ factorisation algorithm is the *forward* iteration of:

$$\begin{aligned} s_j &= s_{j-1} + f_j g_j \quad (s_0 = \beta) \\ d_j &= \frac{s_{j-1}}{s_j} \bar{d}_j \\ u_j^{(j)} &= 1, \\ u_i^{(j)} &= \frac{-f_j}{s_{j-1}} g_j \quad (i = 1 \dots j-1, j > 1). \end{aligned}$$

B.2 Regularisation of U-D filter

The Levenberg-Marquardt modification adds a small identity matrix $\mu \mathbf{I}$ to the information matrix $\mathbf{S}(t)$ constantly, whether required or not. It would seem more natural to add such a modification only to an emerging null space, so as to prevent any eigenvalue of $\mathbf{S}(t)$ from tending to zero resulting in covariance 'blow-up'. In the U-D factorisation algorithm, by introducing limits on the elements of $\mathbf{U}(t)$ and $\mathbf{D}(t)$ we can ensure that the covariance matrix $\mathbf{P}(t) = \mathbf{U}(t)\mathbf{D}(t)\mathbf{U}^T(t)$ remains bounded by ensuring that:

$$\mathbf{S}(t) = \mathbf{P}^{-1}(t) = (\mathbf{U}^{-1}(t))^T \mathbf{D}^{-1}(t) \mathbf{U}^{-1}(t) \geq \mu \mathbf{I}.$$

Assuming that the data vectors $\mathbf{x}(t)$ do not tend to infinity then the elements of $\mathbf{U}^{-1}(t)$ must remain bounded, and so must the elements of $\mathbf{U}(t)$ (since $\det[\mathbf{U}(t)] = 1$). The boundedness of $\mathbf{U}(t)$ will take care of itself, and thus it is sufficient to introduce a limit for the elements of $\mathbf{D}(t)$, i.e.

$$\mathbf{D}(t) \leq \mathbf{I}/\mu.$$

Appendix C

Proof of the $\mathcal{AUDI}$ method

The following derivation is taken from an appendix in Niu (1993).

Consider the following theorem

Theorem 3 (Decomposition of Partitioned Matrix) *Given a non-negative definite symmetrical matrix $\mathbf{S}$ of dimension $d \times d$, then the LDL^T -decomposition of its partitioned matrix is given by*

$$\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix}^T \quad (\text{C.1})$$

where $\mathbf{A}$ is the $k \times k$ submatrix of $\mathbf{S}$ with $0 < k < d$, and is assumed to be non-singular;

$$\mathbf{\Delta} = \mathbf{D} - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B}$$

This theorem can be easily proven by direct multiplication of the right-hand side of eqn.(C.1).

The derivation begins by observing that the augmented data vector $\phi(t)$ possesses the following “shift structure” Ljung et al. (1978); Ljung and Söderström (1983) From which it can be shown that

$$\begin{aligned} \phi^{(n)}(t) &= \begin{bmatrix} \mathbf{h}^{(n)}(t) \\ -y(t) \end{bmatrix} & \mathbf{h}^{(n)}(t) &= \begin{bmatrix} \phi^{(n-1)}(t) \\ u(t-1) \end{bmatrix} \\ \phi^{(n-1)}(t) &= \begin{bmatrix} \mathbf{h}^{(n-1)}(t) \\ -y(t-1) \end{bmatrix} & \mathbf{h}^{(n-1)}(t) &= \begin{bmatrix} \phi^{(n-2)}(t) \\ u(t-2) \end{bmatrix} \\ &\vdots & &\vdots \\ \phi^{(0)}(t) &= [-y(t-n)] \end{aligned}$$

$$\phi^{(n)}(t) = \begin{bmatrix} -y(t-n) & u(t-n) & \dots & -y(t-2) & u(t-2) & -y(t-1) & u(t-1) & -y(t) \end{bmatrix}^T$$

Define the AIM for different model orders as follows

$$\begin{aligned} \mathbf{S}^{(n)}(t) &= \left[\sum_{j=1}^t \phi^{(n)}(j) \phi^{(n)T}(j) \right]_{(2n+1) \times (2n+1)} \\ \mathbf{R}^{(n)}(t) &= \left[\sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j) \right]_{2n \times 2n} \\ \mathbf{S}^{(n-1)}(t) &= \left[\sum_{j=1}^t \phi^{(n-1)}(j) \phi^{(n-1)T}(j) \right]_{(2n-1) \times (2n-1)} \\ \mathbf{S}^{(n-1)}(t) &= \left[\sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j) \right]_{(2n-2) \times (2n-2)} \\ &\vdots \\ \mathbf{S}^{(0)}(t) &= \left[\sum_{j=1}^t \phi^{(0)}(j) \phi^{(0)T}(j) \right]_{1 \times 1} \end{aligned}$$

Obviously $\mathbf{S}^{(n)}(t)$ can be partitioned and decomposed via Theorem 3

$$\begin{aligned} \mathbf{S}^{(n)}(t) &= \sum_{j=1}^t \phi^{(n)}(j) \phi^{(n)T}(j) \\ &= \left[\begin{array}{c|c} \sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j) & \sum_{j=1}^t \mathbf{h}^{(n)}(j) z(j) \\ \hline -\sum_{j=1}^t z(j) \mathbf{h}^{(n)T}(j) & \sum_{j=1}^t z^2(j) \end{array} \right] \\ &= \left[\begin{array}{c|c} \mathbf{I}_{2n} & \mathbf{0} \\ \hline -\left(\sum_{j=1}^t z(j) \mathbf{h}^{(n)T}(j)\right) \left(\sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j)\right)^{-1} & \mathbf{I}_1 \end{array} \right] \\ &\times \left[\begin{array}{c|c} \sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j) & \mathbf{0} \\ \hline \mathbf{0} & \Delta \end{array} \right] \\ &\times \left[\begin{array}{c|c} \mathbf{I}_{2n} & \mathbf{0} \\ \hline -\left(\sum_{j=1}^t z(j) \mathbf{h}^{(n)T}(j)\right) \left(\sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j)\right)^{-1} & \mathbf{I}_1 \end{array} \right]^T \end{aligned}$$

$$= \left[\begin{array}{c|c} \mathbf{I}_{2n} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\theta}}^{(n)T}(t) & \mathbf{I}_1 \end{array} \right] \left[\begin{array}{c|c} \mathbf{R}^{(n)}(t) & \mathbf{0} \\ \hline \mathbf{0} & J_f^{(n)}(t) \end{array} \right] \left[\begin{array}{c|c} \mathbf{I}_{2n} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\theta}}^{(n)T}(t) & \mathbf{I}_1 \end{array} \right]^T \quad (\text{C.2})$$

where

$$\hat{\boldsymbol{\theta}}^{(n)}(t) = \left(\sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j) \right)^{-1} \left(\sum_{j=1}^t \mathbf{h}^{(n)}(j) z(j) \right)$$

is the parameter estimate of the n^{th} order model, and

$$\begin{aligned} J_f^{(n)}(t) &= \Delta \\ &= \sum_{j=1}^t z^2(j) - \left(\sum_{j=1}^t z(j) \mathbf{h}^{(n)T}(j) \right) \left(\sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j) \right)^{-1} \left(\sum_{j=1}^t \mathbf{h}^{(n)}(j) z(j) \right) \\ &= \sum_{j=1}^t \epsilon^2(j) \end{aligned}$$

is the corresponding loss function for the n^{th} order estimates.

This is called a “nested structure” Söderström and Stoica (1989) since $\mathbf{R}^{(n)}(t)$ is nested in the $\mathbf{S}^{(n)}(t)$ matrix. Now $\mathbf{R}^{(n)}(t)$ can be decomposed in a similar manner to eqn.(C.2) as follows:

$$\begin{aligned} \mathbf{R}^{(n)}(t) &= \sum_{j=1}^t \mathbf{h}^{(n)}(j) \mathbf{h}^{(n)T}(j) \\ &= \left[\begin{array}{c|c} \sum_{j=1}^t \boldsymbol{\phi}^{(n-1)}(j) \boldsymbol{\phi}^{(n-1)T}(j) & \sum_{j=1}^t \boldsymbol{\phi}^{(n-1)}(j) u(j) \\ \hline -\sum_{j=1}^t u(j) \boldsymbol{\phi}^{(n-1)T}(j) & \sum_{j=1}^t u^2(j) \end{array} \right] \\ &= \left[\begin{array}{c|c} \mathbf{I}_{2n-1} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\alpha}}^{(n)T}(t) & \mathbf{I}_1 \end{array} \right] \left[\begin{array}{c|c} \mathbf{S}^{(n-1)}(t) & \mathbf{0} \\ \hline \mathbf{0} & J_b^{(n)}(t) \end{array} \right] \left[\begin{array}{c|c} \mathbf{I}_{2n-1} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\alpha}}^{(n)T}(t) & \mathbf{I}_1 \end{array} \right]^T \end{aligned}$$

Recursively decomposing all the nested matrices in the same manner leads to:

$$\begin{aligned} \mathbf{S}^{(n)}(t) &= \left[\begin{array}{c|c} \mathbf{I}_{2n} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\theta}}^{(n)T}(t) & \mathbf{I}_1 \end{array} \right] \left[\begin{array}{cc|c} \mathbf{I}_{2n-1} & \mathbf{0} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\alpha}}^{(n)T}(t) & \mathbf{I}_1 & \\ \hline \mathbf{0} & & \mathbf{I}_1 \end{array} \right] \left[\begin{array}{cc|c} \mathbf{I}_{2n-2} & \mathbf{0} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\theta}}^{(n-1)T}(t) & \mathbf{I}_1 & \\ \hline \mathbf{0} & & \mathbf{I}_2 \end{array} \right] \cdots \\ &\quad \times \left[\begin{array}{cc|c} \mathbf{I}_2 & \mathbf{0} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\theta}}^{(1)T}(t) & \mathbf{I}_1 & \\ \hline \mathbf{0} & & \mathbf{I}_{2n-2} \end{array} \right] \left[\begin{array}{cc|c} \mathbf{I}_2 & \mathbf{0} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\alpha}}^{(0)T}(t) & \mathbf{I}_1 & \\ \hline \mathbf{0} & & \mathbf{I}_{2n-1} \end{array} \right] \end{aligned}$$

$$\begin{aligned}
 & \times \begin{bmatrix} J_f^{(0)}(t) & & & & \\ & J_b^{(1)}(t) & & & \\ & & J_f^{(1)}(t) & & \\ & & & \ddots & \\ & & & & J_b^{(n)}(t) \\ & & & & & J_f^{(n)}(t) \end{bmatrix} \\
 & \times \left[\begin{array}{cc|c} I_2 & \mathbf{0} & \mathbf{0} \\ -\hat{\boldsymbol{\alpha}}^{(0)T}(t) & I_1 & \\ \hline \mathbf{0} & & I_{2n-1} \end{array} \right]^T \left[\begin{array}{cc|c} I_2 & \mathbf{0} & \mathbf{0} \\ -\hat{\boldsymbol{\theta}}^{(1)T}(t) & I_1 & \\ \hline \mathbf{0} & & I_{2n-2} \end{array} \right]^T \dots \\
 & \times \left[\begin{array}{cc|c} I_{2n-2} & \mathbf{0} & \mathbf{0} \\ -\hat{\boldsymbol{\theta}}^{(n-1)T}(t) & I_1 & \\ \hline \mathbf{0} & & I_2 \end{array} \right]^T \left[\begin{array}{cc|c} I_{2n-1} & \mathbf{0} & \mathbf{0} \\ -\hat{\boldsymbol{\alpha}}^{(n)T}(t) & I_1 & \\ \hline \mathbf{0} & & I_1 \end{array} \right]^T \left[\begin{array}{c|c} I_{2n} & \mathbf{0} \\ \hline -\hat{\boldsymbol{\theta}}^{(n)T}(t) & I_1 \end{array} \right]^T \\
 & = [\mathbf{L}_{2n+1} \mathbf{L}_{2n} \cdots \mathbf{L}_2 \mathbf{L}_1] \mathbf{D} [\mathbf{L}_1^T \mathbf{L}_2^T \cdots \mathbf{L}_{2n}^T \mathbf{L}_{2n+1}^T]
 \end{aligned}$$

i.e.

$$\mathbf{S}(t) = \mathbf{L}(t) \mathbf{D}(t) \mathbf{L}^T(t).$$

It is easy to verify that $\mathbf{L}_i^{-1} = -\mathbf{L}_i$ for $(0 < i \leq 2n + 1)$ and thus

$$\mathbf{u} = [\mathbf{L}^{-1}]^T$$

is the AUDI parameter matrix.

Appendix D

Least-Squares in the δ -domain

Consider the linear-in-the-parameters model

$$y(t) = \boldsymbol{\theta}^T(t)\mathbf{x}(t) + e(t),$$

where $\boldsymbol{\theta}(t) \in \mathbb{R}^n$ is a vector of unknown parameters, $\mathbf{x}(t) \in \mathbb{R}^n$ is a vector of known data, $y(t)$ is a scalar output of the system and $e(t)$ is the measurement noise. Given an estimate $\hat{\boldsymbol{\theta}}(t)$, the model's prediction is:

$$\hat{y}(t) = \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(t)$$

with corresponding prediction error

$$\epsilon(t) = y(t) - \hat{y}(t).$$

The LS estimate using exponential data weighting is given by

$$\hat{\boldsymbol{\theta}}(t) = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, t),$$

where

$$J(\hat{\boldsymbol{\theta}}, t) = \mathbf{S}_0^t E(-\alpha, t - \tau) \left(y(\tau) - \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(\tau) \right)^2 d\tau$$

is the LS *loss function* and $0 < \alpha \leq 1/h$ is the exponential forgetting-factor.

Minimising via $\nabla_{\hat{\boldsymbol{\theta}}} J(\hat{\boldsymbol{\theta}}, t) = \mathbf{0}$ gives:

$$\{\mathbf{S}_0^t E(-\alpha, t - \tau) \mathbf{x}(\tau) \mathbf{x}^T(\tau) d\tau\} \hat{\boldsymbol{\theta}}(t) = \mathbf{S}_0^t E(-\alpha, t - \tau) \mathbf{x}(\tau) y(\tau) d\tau,$$

i.e.

$$\hat{\boldsymbol{\theta}}(t) = \mathbf{S}_{xx}^{-1}(t) \mathbf{s}_{xy}(t) \quad (\text{LS estimate}) \quad (\text{D.1})$$

with a corresponding LS loss function value:

$$\begin{aligned} J^*(t) &= J(\hat{\boldsymbol{\theta}}(t), t) \\ &= \mathbf{S}_0^t E(-\alpha, t - \tau) \left(y^2(\tau) - 2\hat{\boldsymbol{\theta}}^T(t) \mathbf{x}(\tau) y(\tau) + \hat{\boldsymbol{\theta}}^T(t) \mathbf{x}(\tau) \mathbf{x}^T(\tau) \hat{\boldsymbol{\theta}}(t) \right) d\tau \\ &= Y^2(t) - 2\mathbf{s}_{xy}^T(t) \mathbf{S}_{xx}^{-1}(t) \mathbf{S}_0^t E(-\alpha, t - \tau) \mathbf{x}(\tau) y(\tau) d\tau + \hat{\boldsymbol{\theta}}^T(t) \mathbf{S}_{xx}(t) \hat{\boldsymbol{\theta}}(t) \\ &= Y^2(t) - \mathbf{s}_{xy}^T(t) \mathbf{S}_{xx}^{-1}(t) \mathbf{s}_{xy}(t), \quad (\text{LS loss}) \end{aligned}$$

where

$$\mathbf{S}_{xx}(t) = \mathbf{S}_0^t E(-\alpha, t - \tau) \mathbf{x}(\tau) \mathbf{x}^T(\tau) d\tau, \quad (\text{Information matrix}) \quad (\text{D.2})$$

$$\mathbf{s}_{xy}(t) = \mathbf{S}_0^t E(-\alpha, t - \tau) \mathbf{x}(\tau) y(\tau) d\tau, \quad (\text{D.3})$$

$$Y^2(t) = \mathbf{S}_0^t E(-\alpha, t - \tau) y^2(\tau) d\tau.$$

Defining the covariance matrix $\mathbf{P}(t)$ as the inverse of the information matrix $\mathbf{S}_{xx}(t)$, the off-line estimator of eqn.(D.1) can be rewritten as

$$\hat{\boldsymbol{\theta}}(t) = \mathbf{P}(t) \mathbf{s}_{xy}(t)$$

and differentiating with respect to t gives (using the generalised product rule):

$$\rho \hat{\boldsymbol{\theta}}(t) = \{\rho \mathbf{P}(t)\} \mathbf{s}_{xy}(t) + \mathbf{P}(t) \{\rho \mathbf{s}_{xy}(t)\} + h \{\rho \mathbf{P}(t)\} \{\rho \mathbf{s}_{xy}(t)\}. \quad (\text{D.4})$$

Now $\rho\{\mathbf{P}(t) \mathbf{P}^{-1}(t)\} = 0$ and thus (using the generalised product rule again)

$$\{\rho \mathbf{P}(t)\} \mathbf{P}^{-1}(t) + \mathbf{P}(t) \{\rho \mathbf{P}^{-1}(t)\} + h \{\rho \mathbf{P}(t)\} \{\rho \mathbf{P}^{-1}(t)\} = 0$$

giving

$$\{\rho \mathbf{P}(t)\} = -\mathbf{P}(t) \{\rho \mathbf{P}^{-1}(t)\} (\mathbf{P}^{-1}(t) + h \{\rho \mathbf{P}^{-1}(t)\})^{-1}. \quad (\text{D.5})$$

Differentiating eqn.(D.2) leads to

$$\rho \mathbf{P}^{-1}(t) = (1 - \alpha h) \mathbf{x}(t) \mathbf{x}^T(t) - \alpha \mathbf{P}^{-1}(t)$$

Table D.1: The δ -RLS estimator equations

$\epsilon(t)$	$= y(t) - \hat{\boldsymbol{\theta}}^T(t)\mathbf{x}(t)$	(Prediction error)
$\rho\mathbf{P}(t)$	$= \frac{1}{1 - \alpha h} \left(\alpha\mathbf{P}(t) - \frac{\mathbf{P}(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{P}(t)}{1 + h\mathbf{x}^T(t)\mathbf{P}(t)\mathbf{x}(t)} \right)$	(Covariance update)
$\rho\hat{\boldsymbol{\theta}}(t)$	$= \frac{\mathbf{P}(t)\mathbf{x}(t)\epsilon(t)}{1 + h\mathbf{x}^T(t)\mathbf{P}(t)\mathbf{x}(t)}$	(Parameter update)

and substituting into eqn.(D.5) we get (using the matrix inversion lemma¹):

$$\rho\mathbf{P}(t) = \left(-\mathbf{P}(t)\mathbf{x}(t)\mathbf{x}^T(t) + \frac{\alpha\mathbf{I}}{1 - \alpha h} \right) \left(\mathbf{P}(t) - \frac{\mathbf{P}(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{P}(t)}{1/h + \mathbf{x}^T(t)\mathbf{P}(t)\mathbf{x}(t)} \right),$$

which can be rearranged to give the covariance update formula in Table D.1. Differentiating eqn.(D.3) yields

$$\rho s_{xy}(t) = (1 - \alpha h) x(t)y(t) - \alpha s_{xy}(t)$$

and substituting into eqn.(D.4) gives the desired parameter update differential equation as shown in Table D.1.

¹Matrix inversion lemma: $(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1}$.

Appendix E

Derivation of the coupled tanks model

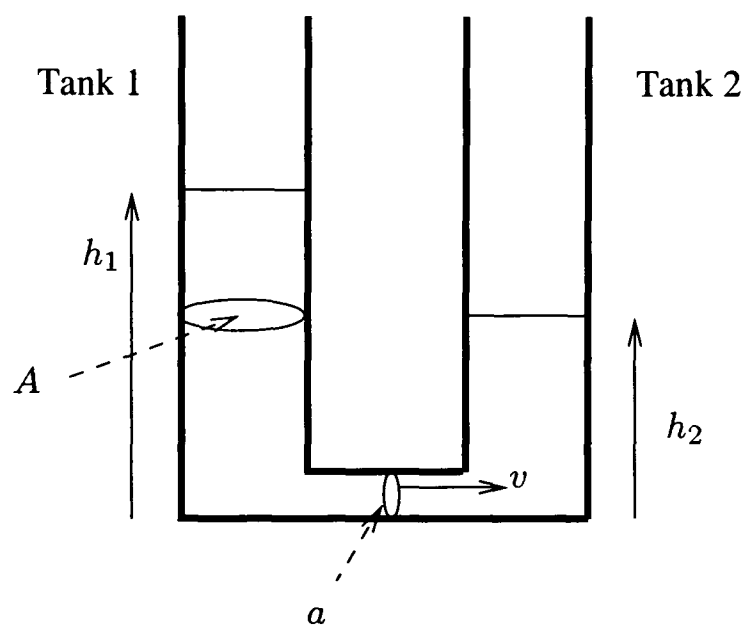


Figure E.1: Coupled tanks

Start with an energy balance for an element of mass δm in tank 1 shown in Figure E.1

$$\delta m g \Delta h = \frac{1}{2} \delta m v^2$$

and thus

$$v^2 = 2g|\Delta h|.$$

where Δh is the difference in tank levels $h_1 - h_2$ and v is the velocity of the liquid in the pipe joining the two tanks. Note that if the pipe does not connect to any other tank, h_2 should be taken to be the height of the pipe outlet (usually zero).

Now suppose that the mass element occupies a volume corresponding to a height δh in tank 1 which has a cross-sectional area A , i.e.

$$\delta m = \rho \Delta V = \rho A \delta h$$

where ρ is the density of the liquid. Assuming that the liquid is incompressible, if the element δm moves δh meters down the tank in δt seconds, a mass element of equal size must move a distance δl meters along the pipe of cross-sectional area a in the same time, i.e.

$$\frac{\delta m}{\delta t} = \rho A \frac{\delta h}{\delta t} = \rho a \frac{\delta l}{\delta t}$$

and thus in the limit as $\delta t \rightarrow 0$ we obtain an expression for the mass flow rate in the pipe in terms of the difference in tank levels:

$$\frac{dm}{dt} = \rho a v = \rho a \operatorname{sign}(\Delta h) \sqrt{2g|\Delta h|}$$

which is positive for flows out of tank 1 into tank 2, and negative vice versa. The corresponding volumetric outflow rate from tank 1 is

$$F_{out} = a \operatorname{sign}(\Delta h) \sqrt{2g|\Delta h|}$$

Thus, for the simple tank configuration in Figure E.1, the levels of each tank are given by the coupled, non-linear differential equations

$$\begin{aligned} \frac{dh_1}{dt} &= \frac{1}{A} \left(\sum F_{in} - \sum F_{out} \right) \\ &= \frac{1}{A} \left(F_1 - a \operatorname{sign}(h_1 - h_2) \sqrt{2g|h_1 - h_2|} \right) \\ \frac{dh_2}{dt} &= \frac{1}{A} \left(\sum F_{in} - \sum F_{out} \right) \\ &= \frac{1}{A} \left(F_2 - a \operatorname{sign}(h_2 - h_1) \sqrt{2g|h_2 - h_1|} \right) \end{aligned}$$

where F_1 and F_2 are the inflows in m^3/s for each tank respectively.

References

- Agee, W. S. and Turner, R. H. (1976). Triangular decomposition of a positive definite matrix plus a symmetric dyad with application to Kalman filtering. Technical Report 38, White Sands Missile Range.
- Akaike, H. (1969). Fitting autoregressive models for prediction. *Annals of Institute of Mathematical Statistics*, **21**, 243–247.
- Akaike, H. (1972). Use of an information theoretic quantity for statistical model identification. In *Proceedings of the 5th Hawaii International Conference on Systems Science*, pp. 249–250.
- Akaike, H. (1977). On entropy maximisation principle. In *Proceedings of a Symposium on Applications of Statistics*.
- Åström, K. J. (1970). *Introduction to Stochastic Control Theory*. Academic Press.
- Åström, K. J. and Wittenmark, B. (1989). *Adaptive control*. Addison Wesley.
- Bierman, G. J. (1976). Measurement updating using the U-D factorisation. *Automatica*, **12**, 375–382.
- Bierman, G. J. (1977). *Factorisation methods for discrete sequential estimation*. Academic Press.
- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press Oxford.
- Bowyer, R. O. and Clarke, D. W. (1996). Estimation of multiple order models. In Friswell, M. I. and Mottershead, J. E. (Eds), *Identification in Engineering Systems*, pp. 205–214, Swansea, Wales. The Cromwell Press.
- Bowyer, R. O. and Clarke, D. W. (1998). Multiple model least-squares using a Lagrange multiplier approach. *Electronics Letters*, **34**(3), 311–312.

- Camacho, E. F. and Bordons, C. (1995). *Model Predictive Control in the Process Industry*. Springer.
- Chen, S. and Billings, S. A. (1989). Representations of non-linear systems: the NARMAX model. *International Journal of Control*, **49**(3), 1013–1032.
- Chow, C. M. (1996). *Predictive control with constraints*. D.Phil thesis, Department of Engineering Science, University of Oxford.
- Clarke, D. W. (1981a). Implementation of self-tuning controllers. In Harris, C. J. and Billings, S. A. (Eds), *Self-tuning and adaptive control: Theory and applications*, pp. 144–165. Peter Peregrinus.
- Clarke, D. W. (1981b). Introduction to self-tuning controllers. In Harris, C. J. and Billings, S. A. (Eds), *Self-tuning and adaptive control: Theory and applications*, pp. 36–71. Peter Peregrinus.
- Clarke, D. W. (1988). Introduction to self-tuning control. In Warwick, K. (Ed.), *Implementation of self-tuning controllers*, pp. 1–22. Peter Peregrinus.
- Clarke, D. W. (Ed.) (1994). *Advances in Model-Based Predictive Control*. Oxford University Press.
- Clarke, D. W. (1995). Simultaneous estimation of models with different order. In *Proceedings of the COPERNICUS summer school on identification and optimisation*, pp. 29–36, Prague, The Czech Republic.
- Clarke, D. W. and Hinton, C. (1997). Adaptive control of materials-testing machines. *Automatica*, **33**(6), 1119–1131.
- Clarke, D. W., Mohtadi, C., and Tuffs, P. S. (1987). Generalised predictive control. Part I : the basic algorithm and Part II: extensions and interpretations. *Automatica*, **23**(2), 137–160.
- Clarke, D. W. and Scattolini, R. (1991). Constrained receding-horizon predictive control. *IEE Proceedings Part D*, **138**(4), 347–354.

- Cluett, W. R., Shah, S. L., and Fisher, D. G. (1988). Robustness analysis of discrete-time adaptive control systems using input-output stability theory: a tutorial. *IEEE Proceedings Part D*, **135**(2), 133–141.
- Cutler, C. R. and Ramaker, B. L. (1980). Dynamic matrix control: a computer control algorithm. In *Joint Automatic Control Conference*, San Francisco, U.S.A.
- Demircioğlu, H. and Gawthrop, P. J. (1991). Continuous-time generalised predictive control. *Automatica*, **27**(1), 55–94.
- DMC, C. (1994). *DMC: Technology overview*. Product literature from DMC, Corp.
- Dollar, R., Melton, L. L., Morshedi, A. M., and Glasgow, D. T. (1993). Consider adaptive multivariable predictive controllers. In *Hydrocarbon Process.*, pp. 109–112.
- Durbin, J. (1960). The fitting of time-series models. *Review of Institute of International Statistics*, **28**(4), 233–243.
- Egardt, B. (1979). *Stability of Adaptive Controllers (Lecture notes in Control and Information Sciences, No. 20)*. Springer-Verlag.
- Elnaggar, A., Dumont, G. A., and Elshafei, A. (1989). Recursive estimation for system of unknown delay. In *Proceedings of the 28th Conference on Decision and Control*, Tampa, Florida.
- Eykhoff, P. (1974). *System Identification*. John Wiley & Sons.
- Fortescue, T. R., Kershenbaum, L. S., and Ydstie, B. E. (1981). Implementation of self-tuning regulators with variable forgetting factor. *Automatica*, **17**(6), 831–835.
- Freeman, T. G. (1985). Selecting the best linear transfer function model. *Automatica*, **21**(4), 361–370.
- Gauss, K. F. (1809). *Theoria motus corporum coelestium. English translation in: Theory of the motion of the heavenly bodies*. Dover, New York (1963).
- Gawthrop, P. J. (1987). *Continuous-time self-tuning control, Vol. 1: Design*. Research Studies Press.

- Gendron, S. and Perrier, M. (1994). Constrained model predictive control of low order plants based on model weighting adaption. *Journal A*, **35**(3), 18–25.
- Gevers, M. and Li, G. (1993). *Parametrizations in Control, Estimation and Filtering Problems*. Springer-Verlag.
- Gill, P. E., Golub, G. H., Murray, W., and Saunders, M. A. (1974). Methods for modifying matrix factorisations. *Mathematics of Computation*, **28**(126), 505–535.
- Goodwin, G. C., Leal, R. L., Mayne, D. Q., and Middleton, R. H. (1986). Rapprochement between continuous and discrete model reference adaptive control. *Automatica*, **22**, 199–207.
- Goodwin, G. C., Middleton, R. H., and Poor, H. V. (1992). High-speed digital signal processing and control. *Proceedings of the IEEE*, **80**(2), 240–259.
- Goodwin, G. C. and Sin, K. S. (1984). *Adaptive filtering, prediction and control*. Prentice Hall.
- Graupe, D., Jain, V. K., and Salah, J. (1980). A comparative analysis of various least-squares identification algorithms. *Automatica*, **16**, 663–681.
- Gustafsson, F. and Hjalmarsson, H. (1995). Twenty-one ML estimators for model selection. *Automatica*, **31**(10), 1377–1392.
- Hägglund, T. (1983). Recursive least squares identification with forgetting of old data. Technical Report LUTFD2/(TFRT-1025)/1-20/1983, Department of Automatic Control, Lund Institute of Technology.
- Han, Z. and Frank, P. M. (1997). Comment on ‘modified least squares algorithm incorporating exponential resetting and forgetting’. *International Journal of Control*, **68**(4), 947–953.
- Hinton, C. J. (1992). *Control of servohydraulic materials-testing machines*. D.Phil thesis, Department of Engineering Science, University of Oxford.
- Honeywell, I. (1995). *RMPCT concepts reference*. Product literature from Honeywell, Inc.

- Kouvaritakis, B. and Rossiter, J. A. (1992). Stable generalised predictive control. *IEE Proceedings Part D*, **139**(4), 349–362.
- Kulhavý, R. and Kárný, M. (1984). Tracking of slowly varying parameters by directional forgetting. In *Proceedings of IFAC 9th World Congress*, Budapest, Hungary.
- Kuznetsov, A. G., Bowyer, R. O., and Clarke, D. W. (1999). Estimation of multiple order models in the δ -domain. *International Journal of Control*, **72**(7/8), 629–642.
- Lambert, E. P. (1987). *Process control applications of long-range prediction*. D.Phil thesis, Department of Engineering Science, University of Oxford.
- Lauritsen, M. B., Rostgaard, M., and Poulsen, N. K. (1997). GPC using a delta-domain emulator-based approach. *International Journal of Control*, **68**(1), 219–232.
- Lawson, C. L. and Hanson, R. J. (1974). *Solving least-squares problems*. Prentice-Hall.
- Levenberg, K. (1944). A method for the solution of certain nonlinear problems in least squares. *Quarterly of Applied Mathematics*, **2**, 164–168.
- Levinson, N. (1947). The Weiner rms (root mean square) error criterion – filter design and prediction. *Journal of Mathematical Physics*, **25**, 261–278.
- Ljung, L. (1987). *System identification: Theory for the user*. Prentice-Hall.
- Ljung, L. and Gunnarsson, S. (1990). Adaption and tracking in system identification – a survey. *Automatica*, **26**(1), 7–21.
- Ljung, L. and Söderström, T. (1983). *Theory and Practice of Recursive Identification*. MIT Press.
- Ljung, S. and Ljung, L. (1985). Error propagation properties of recursive least-squares adaption algorithms. *Automatica*, **21**(2), 157–167.
- Ljung, S., Morf, M., and Falconer, D. (1978). Fast calculation of gain matrices for recursive estimation schemes. *International Journal of Control*, **27**(1), 1–19.

- Maniar, V. M., Shah, S. L., Fisher, D. G., and Mutha, R. K. (1997). Multivariable constrained adaptive GPC: theory and experimental evaluation. *International Journal of Adaptive Control and Signal Processing*, **11**(4), 343–365.
- Marquardt, D. W. (1963). An algorithm estimation of non-linear parameters. *Journal of the Society for Industrial and Applied Mathematics*, **11**, 431–441.
- McMichael, D. W. (1988). *On-line fault detection, a system-nonspecific approach*. D.Phil thesis, Department of Engineering Science, Oxford University.
- Middleton, R. H. and Goodwin, G. C. (1990). *Digital control and estimation*. Prentice Hall.
- Mohtadi, C. (1987). *Advanced self-tuning algorithms*. D.Phil thesis, Department of Engineering Science, Oxford University.
- Mohtadi, C. (1988a). Numerical algorithms in self-tuning control. In Warwick, K. (Ed.), *Implementation of self-tuning controllers*, pp. 67–95. Peter Peregrinus.
- Mohtadi, C. (1988b). On the role of prefiltering in parameter estimation and control. In *Workshop on adaptive control strategies for industrial use*, Lodge Kananaskis, Alberta, Canada.
- Mohtadi, C., Shah, S. L., and Clarke, D. W. (1986). Generalized predictive control of multivariable systems. Technical Report OUEL 1640/86, Department of Engineering Science, Oxford University.
- Mohtadi, C., Shah, S. L., and Clarke, D. W. (1987). Multivariable adaptive control without a prior knowledge of the delay matrix. *Systems and Control Letters*, **9**, 295–306.
- Neumann, R. D., Dumur, D., and Boucher, P. (1992). Delta-operator generalised predictive control (DGPC). In *Proceedings of the 31st Conference on Decision and Control*, pp. 2224–2225, Tuscon, Arizona.
- Neumann, R. D., Dumur, D., and Boucher, P. (1993). Application of delta-operator generalised predictive control (DGPC). In *Proceedings of the 32nd Conference on Decision and Control*, pp. 2499–2504, San Antonio, Texas.

- Niu, S. (1993). *Augmented UD Identification for process control*. D.Phil thesis, Department of Chemical Engineering, University of Alberta.
- Niu, S. and Fisher, D. G. (1993). Information forgetting using the augmented UD identification algorithm. In *Proceedings of the American Control Conference*, pp. 796–800, San Francisco, CA.
- Niu, S. and Fisher, D. G. (1994). Monitoring parameter identifiability during on-line identification. In *Proceedings of 3rd IEEE Control Application Conference*, pp. 1717–1722, Glasgow, Scotland.
- Niu, S., Fisher, D. G., Ljung, L., and Shah, S. L. (1995a). A tutorial on multiple model least-squares and augmented UD identification. Technical Report LiTH-ISY-R-1710, Department of Electrical Engineering, Linköping University, Linköping.
- Niu, S., Fisher, D. G., and Xiao, D. (1992). An augmented UD identification algorithm. *International Journal of Control*, **56**(1), 193–211.
- Niu, S. and Ljung, L. (1995). Multiple model parameter estimation. Technical Report LiTH-ISY-R-1711, Department of Electrical Engineering, Linköping University, Linköping.
- Niu, S., Ljung, L., and Björk, Å. (1995b). Decomposition methods for least-squares parameter estimation. Technical Report LiTH-ISY-R-1712, Department of Electrical Engineering, Linköping University, Linköping.
- Ortega, R., Praly, L., and Landau, I. D. (1985). Robustness of discrete-time adaptive controllers. *IEEE Transactions on Automatic Control*, **30**(12), 1179–1179.
- Parkum, J. E., Poulsen, N. K., and Holst, J. (1992). Recursive forgetting algorithms. *International Journal of Control*, **55**(1), 109–128.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical recipes in C*. Cambridge University Press.
- Qin, S. J. and Badgwell, T. A. (1996). An overview of industrial model predictive control technology. *AIChE Symposium Series*, **93**(316), 232–256.

- Rawlings, J. B. and Muske, K. R. (1993). Stability of constrained receding horizon control. *IEEE Transactions of Automatic Control*, **38**(10), 1512–1516.
- Richalet, J., Rault, A., Testud, J. L., and Papon, J. (1978). Model predictive heuristic control: applications to industrial processes. *Automatica*, **14**(5), 413–428.
- Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, **14**(5), 465–471.
- Robinson, B. D. and Clarke, D. W. (1991). Robustness effects of a prefilter in generalised predictive control. *IEE Proceedings Part D*, **138**(1), 2–8.
- Rossiter, J. A., Gossner, J. R., and Kouvaritakis, B. (1996). Infinite horizon stable predictive control. *IEEE Transactions on Automatic Control*, **41**(10), 1522–1527.
- Rossiter, J. A. and Kouvaritakis, B. (1993). Constrained stable generalised predictive control. *IEE Proceedings Part D*, **140**(4), 243–254.
- Rossiter, J. A., Kouvaritakis, B., and Gossner, J. R. (1995). Feasibility and stability results for constrained stable generalized predictive control. *Automatica*, **31**(6), 863–877.
- Salgado, M. E., Goodwin, G. C., and Middleton, R. H. (1988). Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control*, **47**(2), 478–491.
- Sandoz, D. J. (1996). The capability of model predictive control. *Measurement and Control*, **29**(4), 106–111.
- Schwartz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461–464.
- Scokaert, P. O. M. (1994). *Constrained predictive control*. D.Phil thesis, Department of Engineering Science, Oxford University.
- Scokaert, P. O. M. (1997). Infinite horizon generalized predictive control. *International Journal of Control*, **66**(1), 161–75.
- Scokaert, P. O. M. and Rawlings, J. B. (1996). On infeasibilities in model predictive control. In *Proceedings of Chemical Process Control – V. Assessment and new directions for research*, Tahoe City, California.

- Setpoint, I. (1993). *SMC-Idcom: A state-of-the-art multivariable predictive controller*. Product literature from Setpoint, Inc.
- Söderström, T. and Stoica, P. (1989). *System Identification*. Prentice Hall.
- Sripada, R. and Fisher, D. G. (1987). Improved least squares identification. *International Journal of Control*, **46**(6), 1889–1913.
- Tham, M. T., Vagi, F., and Morris, A. J. (1991). Multivariable and multirate self-tuning control: a distillation column case study. *IEE Proceedings Part D*, **138**(1), 9–24.
- Thornton, C. L. and Bierman, G. J. (1978). Filtering and error analysis via the UDU covariance factorisation. *IEEE Transactions of Automatic Control*, **AC-23**(5), 901–907.
- Thygesen, B. G. (1993). *Modelling and Control in the Delta domain*. Ph.D thesis, No.70, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark.
- Verhaegen, M. H. (1989). Round-off error propagation in four generally-applicable, recursive, least-squares estimation schemes. *Automatica*, **25**(3), 437–444.
- Vijayan, R., Poor, H. V., Moore, J. B., and Goodwin, G. C. (1991). A Levinson-type algorithm for modeling fast-sampled data. *IEEE Transactions of Automatic Control*, **AC-36**(3), 314–321.
- Widrow, B. and Hoff Jr., M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, **4**, 96–104.
- Yoon, T.-W. (1994). *Robust adaptive predictive control*. D.Phil thesis, Department of Engineering Science, Oxford University.
- Young, P. C. (1981). Parameter estimation for continuous-time models – a survey. *Automatica*, **17**(1), 23–39.
- Young, P. C., Jakeman, A. J., and McMurtrie, R. E. (1980). An instrumental variable method for model order identification. *Automatica*, **16**(3), 281–294.
- Zadeh, L. A. (1994). The role of fuzzy logic in modeling, identification and control. *Modeling, Identification and Control*, **15**(3), 191–203.

