ISSN 0280-5316
ISRN LUTFD2/TFRT--5581--SE

An Adaptive Local Model Networks Strategy for Nonlinear Control

Ola Nockhammar

Department of Automatic Control
Lund Institute of Technology
June 1997

Department of Automatic Control **Lund Institute of Technology** **Box 118** **S-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* June 1997 *Document Number* ISRN LUTFD2/TFRT--5581--SE
Author(s) Ola Nockhammar	*Supervisor* Prof. Dr. Ken Hunt Dr. Karl-Erik Årzén *Sponsoring organisation*

Title and subtitle

An Adaptive Local Model Networks Strategy for Nonlinear Control

Abstract

The paper describes an adaptive local model networks strategy intended for control of a class of nonlinear, time-varying systems. Some recent progresses in multiple model identification, local model networks and switching are combined to a very powerful and flexible strategy. The method does not need much more computational effort than a conventional adaptive control scheme. More memory are however needed. The paper describes all aspects of the method: robust real-time controller design, robust real-time multiple model identification, local model networks and switching. A simulation example of the high performance vehicle speed control problem is also presented. The work has done as part of an ESPRIT III project called *Neural Adaptive Control Technology.*

Key words

Adaptive Control, Nonlinear Control, Local Model Networks, Model Reduction, Multiple Model Least Squares

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280–5316		*ISBN*
Language English	*Number of pages* 90	*Recipient's notes*
Security classification		

Contents

Chapter 1

Introduction

Many real processes are nonlinear and time-varying, with different system complexity in different operating points. The adaptive local model networks strategy for nonlinear control, described in the paper, is intended for such systems. Despite the powerfulness, it does not need much more computational effort than conventional adaptive control, only more memory.

The major parts of the strategy are:

- robust real-time identification of locally valid linear models of order 1 to n, distributed over the whole space of operation;
- interpolation of the locally valid linear models of order 1 to n to globally valid local model networks of order 1 to n;
- online model validation to choose the local model network with the correct model order in the entire operating space;
- robust real-time model reduction and controller design.

Robust real-time identification of linear models of order 1 to n are achieved through a new least squares identification method called the Multiple Model Least Squares method (MMLS), given by Niu *et al* [17], and an extended conditional updating rule. The MMLS method simultaneously gives least squares estimates of order 1 to n and the corresponding loss functions, to the same computational effort that the conventional recursive least squares (RLS) algorithm gives the n:th order estimates.

The locally valid linear models are then interpolated to globally valid local model networks of order 1 to n. The interpolation can be viewed as a mechanism to get smooth switching between the locally valid linear models.

Since we have one local model network of each order (order 1 to n), there is always one local model network with the correct model order in all operating points. To get the correct model order in all operating points, we use online model validation. An Akaike Information Criterion (AIC) could, for example, be used as a validation criteria. The validation tells how good the different local model networks are in different regimes.

The local model network estimates have, however, the drawback that they have many parameters to tune up. It means that it might take long time to re-tune all parameters in the local model networks when the plant's dynamics has changed.

To get faster adaptation, we take the locally valid models of order 1 to n in the present regime of operation as candidates to the 'best' model. Finally, we also take fixed local model networks of order 1 to n as candidates for the 'best' model. The fixed models are there for the cause of safety or lower bound performance. With a suitable switching criteria, the performance of the adaptive strategy will never be worse than the performance for the fixed local model network.
Each model has an associated loss-function in all regimes to validate the local performance of that model. The switching rule should be of the type: even though there is a better model, keep the present model if it is good enough.

The local model networks and the switching between local model networks, can be viewed as two different kind of switching. The local model networks can be seen as a kind of structural switching, to capture the known system nonlinearities. The purpose of the switching between local model networks, on the other hand, is to automatically choose the correct model order in all operating points, and to achieve fast adaptation to changes in the plant dynamics, and to imply robustness.

The model produced by the adaptive local model networks strategy is now used for online controller design. Since the controller design is made online, it is very important that the design algorithm never breaks down, is fast to compute and has good numerical properties. To ensure this, we use a model reduction algorithm that gives a controller solution with condition number less than a specified number. The algorithm exploits the very special structure of the problem, and is therefore fast and has good numerical properties. A two degrees of freedom controller is then computed based on the reduced model, to give the closed loop system the desired response to the command signal.

The resulting strategy is shown to be very powerful and able to cope with many different situations. It is easy to adjust the strategy to take the behavior of the specific plant into account. In spite of the powerfulness, the strategy does not need much more computational effort than a conventional adaptive control scheme. This is because only one local model is updated at each time instant, and one controller design problem is solved at each time instant, i.e. the same as for conventional adaptive control. The main extra computations are due to the interpolation and the online model validation. These operations are, however, relatively cheap.

Outline of the paper

In the second chapter, different aspects of online controller design are discussed. A two degrees of freedom controller design algorithm and a model reduction algorithm, both suitable for real-time application, are given. Some analysis and an experiment of the sensitivity to modeling errors are also given. The author's main contribution in this chapter is the model reduction algorithm based on the Sylvester matrix. The method has , however, strong similarities with an algorithm given by Barnett [3].

The third chapter treats the Multiple Model Least Squares (MMLS) method. The method given here, differs from the one given by Niu *et al* by the representation of the regressor vector. Our regressor vector is mirrored compared to the one used in Niu's method, leading to a slightly different method. This modification is due to the author. A modified updating formula and an extended conditional updating formula that use the multiple model structure, are also given. Both these formulas are to our knowledge, new approaches.

Chapter 4 treats the local model networks/local controller networks. Original contributions to these topics are Johansen and Foss 1992, 1993, Johansen 1994, Hunt and Johansen 1997 [8, 9, 10, 6].

In Chapter 5 the MMLS algorithm is used to estimate the local models in the local model networks. Online model validation and switching are also discussed. The idea to switch between a set of fixed and estimated models has been used before by, for example, Narendra *et al* [15]. Narendra is switching between a set of fixed and estimated linear models based on online model validation. We are instead switching between a set of fixed and estimated local model networks of order 1 to n, based on online model validation. This is to our knowledge a new approach. Adaptive networks/online controller design are discussed by Wang [18]. The difference of our approach to the one used in [18] is that we use the MMLS algorithm to estimate the local models of order 1 to n in the local model networks, while Wang uses one conventional RLS algorithm to estimate the entire network parameters of order $N \cdot n$.

In Chapter 6, we compare the control performance of the adaptive local model networks strategy for control with the control performance of conventional adaptive control, on a highly nonlinear and unstable plant. It is shown that the adaptive local model networks strategy can give significantly improved control performance on such plants.

In Chapter 7, we apply the strategy on the vehicle speed control problem, and shows that it can give excellent control performance over the complete operational range.

The Work has been done as part of an ESPRIT III project called *Neural Adaptive Control Technology.*

Acknowledgments

The author is grateful to many helpful people at Daimler-Benz Research (Intelligente Systeme, Berlin) and Lund Institute of Technology (Department of Automatic Control). Special thanks to the supervisors Ken Hunt and Karl-Erik Årzén, and to Thomas Schauer for many good discussions, and to Karl-Johan Åström for the chance to do this.

Chapter 2

Linear Control: Synthesis, Analysis and Algorithms

2.1 Introduction

In an adaptive control setting the controller is designed online. It is therefore very important that the design algorithms are numerically robust and fast. In this chapter we discuss such control methods and other aspects of real-time control. An analysis of modelling error sensitivity is also given. These aspects are covered because the adaptive implementations later uses these algorithms. Pole-placement design algorithms and the polynomial reduction algorithms based on the Sylvester matrix can be found in Barnett [3]. The algorithms presented here have, however, been derived independently of [3].

2.2 Pole Placement Design

Assume that the process being investigated is given by the following difference equation model:

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_1 u(t-1) + \cdots + b_n u(t-n), \tag{2.1}$$

where $y(t)$ is the process output and $u(t)$ is the process input. This equation can be described by the input-output model[1]

$$A(q^{-1})y(t) = B(q^{-1})u(t) \tag{2.2}$$

where $A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$ and $B(q^{-1}) = 0 + b_1 q^{-1} + \cdots + b_n q^{-n}$.
A general two-degrees-of-freedom controller with output feedback is given by

$$R(q^{-1})u(t) = T(q^{-1})u_c(t) - S(q^{-1})y(t) \tag{2.3}$$

where $u_c(t)$ is the command signal. $R(q^{-1})$, $S(q^{-1})$ and $T(q^{-1})$ are the controller polynomials. $R(q^{-1})$ is monic.
Eliminating $u(t)$ between the process model eq. (2.2) and the controller eq. (2.3) gives the closed-loop system

$$y(t) = \frac{B(q^{-1})T(q^{-1})}{A(q^{-1})R(q^{-1}) + B(q^{-1})S(q^{-1})}u_c(t). \tag{2.4}$$

[1] q^{-1} denotes the delay operator

Pole-placement design is done mainly in two steps [2]:

1. solve the algebraic problem of finding polynomials $R(q^{-1})$ and $S(q^{-1})$ which give the closed-loop system the desired dynamics $A_{cl}(q^{-1})$, i.e. solve the Diophantine equation $A(q^{-1})R(q^{-1})+B(q^{-1})S(q^{-1}) = A_{cl}(q^{-1})$ with respect to $R(q^{-1})$ and $S(q^{-1})$;
2. determine the polynomial $T(q^{-1})$ which gives the system the desired response to command signals, i.e. solve the equation $B(q^{-1})T(q^{-1}) = B_{cl}(q^{-1})$ with respect to $T(q^{-1})$.

Normally, $A_{cl}(q^{-1})$ can be factorised into a 'controller' polynomial and an 'observer' polynomial

$$A_{cl}(q^{-1}) = A_c(q^{-1})A_o(q^{-1}). \tag{2.5}$$

$B_{cl}(q^{-1})$ is then constrained as

$$B_{cl}(q^{-1}) \quad = \quad b_{cl}B(q^{-1})A_o(q^{-1}). \tag{2.6}$$

The solution to step 2 is then simply $T(q^{-1}) = b_{cl}A_o(q^{-1})$, where b_{cl} is chosen to give the desired static gain. With this solution no process zeros are cancelled. The response to command signals is given by

$$y(t) = \frac{b_{cl}B(q^{-1})}{A_c(q^{-1})}u_c(t) \tag{2.7}$$

The main computational step in pole placement design using polynomials is thus to solve the Diophantine equation.

2.3 The Diophantine Equation

The discussion in the previous section showed that the Diophantine equation plays the central role in pole placement with polynomials. We will now analyse this equation. The fundamental mathematical problem is to understand the properties of the polynomial equation

$$AX + BY = C, \tag{2.8}$$

where A, B and C are known and X and Y are unknown. This is a well known problem in algebra, and is called the Diophantine equation. The problem is usually solved with a Euclidean-like method. We will here solve it using linear matrix equations. The main reason for using an Euclidean like method instead of linear matrix equations is that it is said to exploit the structure of the problem better, and therefor is faster. We will solve it here with a linear matrix equation method that exploits the structure as well as the Euclidean algorithm. With this method it is easy to detect badly conditioned solutions, which will be the case if there are near common factors between A and B.

We will first set up the equations for polynomials of second order and then generalise to arbitrary polynomials.

Suppose A, B and C are given by

$$\begin{aligned} A(q^{-1}) &= 1 + a_1q^{-1} + a_2q^{-2} \\ B(q^{-1}) &= 0 + b_1q^{-1} + b_2q^{-2} \\ C(q^{-1}) &= 1 + c_1q^{-1} + c_2q^{-2} + c_3q^{-3} + c_4q^{-4} \end{aligned}$$

and suppose we want to find two third order polynomials

$$\begin{aligned} X(q^{-1}) &= 1 + x_1 q^{-1} + x_2 q^{-2} + x_3 q^{-3} \\ Y(q^{-1}) &= y_0 + y_1 q^{-1} + y_2 q^{-2} + y_3 q^{-3} \end{aligned}$$

that satisfy the Diophantine equation (2.8). By performing multiplications and sorting by equal order, equation (2.8) becomes

$$\begin{aligned} 1 + (a_1 + 1x_1 + b_1 y_0)q^{-1} + (a_2 + a_1 x_1 + 1x_2 + b_2 y_0 + b_1 y_1)q^{-2} + \\ (a_2 x_1 + a_1 x_2 + 1x_3 + b_2 y_1 + b_1 y_2)q^{-3} + (a_2 x_2 + a_1 x_3 + b_2 y_2 + b_1 y_3)q^{-4} + \\ (a_2 x_3 + b_2 y_3)q^{-5} \\ = 1 + c_1 q^{-1} + c_2 q^{-2} + c_3 q^{-3} + c_4 q^{-4} \end{aligned}$$

This equation can be rewritten as the equation system

$$\begin{aligned} 1 &= 1 \\ 1x_1 + b_1 y_0 &= c_1 - a_1 \\ a_1 x_1 + 1x_2 + b_2 y_0 + b_1 y_1 &= c_2 - a_2 \\ a_2 x_1 + a_1 x_2 + 1x_3 + b_2 y_1 + b_1 y_2 &= c_3 \\ a_2 x_2 + a_1 x_3 + b_2 y_2 + b_1 y_3 &= c_4 \\ a_2 x_3 + b_2 y_3 &= 0 \end{aligned}$$

or using matrices

$$\begin{pmatrix} 1 & 0 & 0 & b_1 & 0 & 0 & 0 \\ a_1 & 1 & 0 & b_2 & b_1 & 0 & 0 \\ a_2 & a_1 & 1 & 0 & b_2 & b_1 & 0 \\ 0 & a_2 & a_1 & 0 & 0 & b_2 & b_1 \\ 0 & 0 & a_2 & 0 & 0 & 0 & b_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} c_1 - a_1 \\ c_2 - a_2 \\ c_3 \\ c_4 \\ 0 \end{pmatrix} \tag{2.9}$$

The matrix on the left-hand side is called the Sylvester matrix. Equation (2.9) has 7 unknown variables. But since we only have 5 equations we can only determine 5 or less variables uniquely. A unique minimum degree solution is found by setting $x_3 = 0$, $y_2 = 0$ and $y_3 = 0$. By imposing these conditions equation (2.9) reduces to

$$\begin{pmatrix} 1 & 0 & b_1 & 0 \\ a_1 & 1 & b_2 & b_1 \\ a_2 & a_1 & 0 & b_2 \\ 0 & a_2 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} c_1 - a_1 \\ c_2 - a_2 \\ c_3 \\ c_4 \end{pmatrix} \tag{2.10}$$

The matrix on the left has a very special structure, which will be used to solve the problem efficiently.

We will now generalise the problem to arbitrary polynomials. Suppose A, B and C are given by

$$\begin{aligned} A(x^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_n q^{-n} \\ B(x^{-1}) &= 0 + b_1 q^{-1} + \cdots + b_n q^{-n} \\ C(x^{-1}) &= 1 + c_1 q^{-1} + \cdots + c_{2n-1} q^{-(2n-1)} + c_{2n} q^{-2n} \end{aligned}$$

Again we want to find a minimum degree solution to the Diophantine equation (2.8) with X monic and Y having $y_0 \neq 0$ (this corresponds to a minimum degree controller without time delay). The X and Y polynomials are then given by

$$\begin{aligned} X(q^{-1}) &= 1 + x_1 q^{-1} + \cdots + x_n q^{-n} \\ Y(q^{-1}) &= y_0 + y_1 q^{-1} + \cdots + y_{n-1} q^{-(n-1)} \end{aligned}$$

Performing the multiplications in the Diophantine equation (2.8) and sorting by equal order give the linear system of equations,

$$\left(\begin{array}{ccccc|ccccc} 1 & 0 & \cdots & 0 & 0 & b_1 & 0 & \cdots & 0 & 0 \\ a_1 & 1 & \cdots & 0 & 0 & b_2 & b_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & \cdots & 1 & 0 & b_{n-1} & b_{n-2} & \cdots & b_1 & 0 \\ a_{n-1} & a_{n-2} & \cdots & a_1 & 1 & b_n & b_{n-1} & \cdots & b_2 & b_1 \\ \hline a_n & a_{n-1} & \cdots & a_2 & a_1 & 0 & b_n & \cdots & b_3 & b_2 \\ 0 & a_n & \cdots & a_3 & a_2 & 0 & 0 & \cdots & b_4 & b_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_n & a_{n-1} & 0 & 0 & \cdots & 0 & b_n \\ 0 & 0 & \cdots & 0 & a_n & 0 & 0 & \cdots & 0 & 0 \end{array}\right) \left(\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \\ \hline y_0 \\ y_1 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{array}\right) = \left(\begin{array}{c} c_1 - a_1 \\ c_2 - a_2 \\ \vdots \\ c_{n-1} - a_{n-1} \\ c_n - a_n \\ \hline c_{n+1} \\ c_{n+2} \\ \vdots \\ c_{2n-1} \\ c_{2n} \end{array}\right) \quad (2.11)$$

The matrix to the left can be divided into four sub matrices with very special structure. The upper left submatrix is a lower triangular toeplitz[2] matrix. Since it has unity diagonal, it has always full rank. The lower left sub matrix is an upper triangular toeplitz matrix. This matrix has full rank if $a_n \neq 0$. The upper right sub matrix is a lower triangular toeplitz matrix. It has full rank if $b_1 \neq 0$. The lower right sub matrix is an upper triangular toeplitz matrix. Equation (2.11) can thus be rewritten as the following matrix equations

$$\begin{pmatrix} L_{11} & L_{12} \\ R_{21} & R_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_x \\ b_y \end{pmatrix} \tag{2.12}$$

where L_{11}, L_{12}, R_{21} and R_{22} are the sub matrices defined in (2.11) and

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, y = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix}, b_x = \begin{pmatrix} c_1 - a_1 \\ \vdots \\ c_n - a_n \end{pmatrix}, b_y = \begin{pmatrix} c_{n+1} \\ \vdots \\ c_{2n} \end{pmatrix}$$

The problem can be written as

$$L_{11}x + L_{12}y = b_x \tag{2.13}$$
$$R_{21}x + R_{22}y = b_y \tag{2.13'}$$

Subtracting $R_{21}L_{11}^{-1}$ times (2.13) from (2.13') results in

$$L_{11}x + L_{12}y = b_x \tag{2.14}$$
$$(R_{22} - R_{21}L_{11}^{-1}L_{12})y = b_y - R_{21}L_{11}^{-1}b_x \tag{2.14'}$$

This can always be done since L_{11} is always invertible and R_{21} has full rank (if $a_n \neq 0$). We have now separated the problem into two subproblems of half the dimension of the original problem. The solution is obtained by first solving (2.14') for y, and then insert this into (2.14) and solving for x:

$$y = (R_{22} - R_{21}L_{11}^{-1}L_{12})^{-1}(b_y - R_{21}L_{11}^{-1}b_x) \tag{2.15}$$
$$x = L_{11}^{-1}(b_x - L_{12}y) \tag{2.15'}$$

[2]A band matrix with constant values on all diagonals.

It is necessary to compute the inverse of L_{11}. But since L_{11} is a lower triangular toeplitz matrix and the inverse of a lower triangular toeplitz matrix also is a lower triangular toeplitz matrix, the computation of this can be done very efficiently; only the first column in the inverse has to be computed. The product of two lower triangular toeplitz matrices is also lower triangular toeplitz, i.e. only the first column in the product has to be computed.

The result is an algorithm which utilises the structure of the problem and has good numerical properties. The matrix $R_{22}-R_{21}L_{11}^{-1}L_{12}$ can however be singular or badly conditioned due to common factors between polynomials A and B. Therefore common factors between A and B should be cancelled.

2.4 A Model Reduction Algorithm: Sylvester Matrix Approach

Common factors are a serious problem in control design. The Diophantine equation (2.8) has a solution if and only if the greatest common factor of A and B divides C [2]. In practice the solution will become numerically sensitive whenever the roots become close. The analysis/experiment given in Sections 2.5 and 2.6 shows that model order reduction can give improved robustness to modelling errors.
In an adaptive control setting, the Diophantine equation is solved online using estimates of the plant. It is then of highest importance to have a control design algorithm that is numerically robust and fast to compute.
In this section we present a polynomial reduction algorithm based on the Sylvester matrix. The method has good numerical properties and exploit the structure of the problem well. The algorithm gives useful information on how difficult the plant is to control. This information could be used in the contoller specification.
A proof/derivation of the method is given in Appendix A and is summarised here as an algorithm.

Algorithm 2.1 (gcd(A,B))
Given two polynomials

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

and

$$B(q^{-1}) = 0 + b_1 q^{-1} + \cdots + b_n q^{-n}$$

such that $a_0 = 1$, $a_n \neq 0$, $b_0 = 0$ *and* $b_1 \neq 0$.

1. *Define matrices* $\mathbf{L}_{11}$, $\mathbf{L}_{12}$, $\mathbf{R}_{21}$ *and* $\mathbf{R}_{22}$ *according to*

$$\mathbf{L}_{11} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ a_1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & \cdots & 1 & 0 \\ a_{n-1} & a_{n-2} & \cdots & a_1 & 1 \end{pmatrix} \quad \mathbf{L}_{12} = \begin{pmatrix} b_1 & 0 & \cdots & 0 & 0 \\ b_2 & b_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{n-1} & b_{n-2} & \cdots & b_1 & 0 \\ b_n & b_{n-1} & \cdots & b_2 & b_1 \end{pmatrix}$$

$$\mathbf{R}_{21} = \begin{pmatrix} a_n & a_{n-1} & \cdots & a_2 & a_1 \\ 0 & a_n & \cdots & a_3 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_n & a_{n-1} \\ 0 & 0 & \cdots & 0 & a_n \end{pmatrix} \quad \mathbf{R}_{22} = \begin{pmatrix} 0 & b_n & \cdots & b_3 & b_2 \\ 0 & 0 & \cdots & b_4 & b_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & b_n \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

2. *LU-factorise as* $\mathbf{LU} = \mathbf{P}(\mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12})^T$
   $\mathbf{U}^T$ *will then have the structure*

$$\mathbf{U}^T = \left(\begin{array}{ccccc|ccc} u_{1,1} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ u_{2,1} & u_{2,2} & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{r-1,1} & u_{r-1,2} & \cdots & u_{r-1,r-1} & 0 & 0 & \cdots & 0 \\ u_{r,1} & u_{r,2} & \cdots & u_{r,r-1} & \mathbf{u_{r,r}} & 0 & \cdots & 0 \\ \hline u_{r+1,1} & u_{r+1,2} & \cdots & u_{r+1,r-1} & \mathbf{u_{r+1,r}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{n,1} & u_{n,2} & \cdots & u_{n,r-1} & \mathbf{u_{n,r}} & 0 & \cdots & 0 \end{array}\right) \tag{2.16}$$

   where $r = 2n - k$.

3. *Calculate a norm on the columns of* $\mathbf{U}^T$

4. *The common factor is found by normalisation of the last column in* $\mathbf{U}^T$ *with nonzero norm (in practice this should be replaced with: the last column of* $\mathbf{U}^T$ *with norm > epsilon). That is the common factor is given by*

$$K = \gcd(A, B) = \frac{1}{\mathbf{u_{r,r}}} \begin{pmatrix} \mathbf{u_{r,r}} \\ \vdots \\ \mathbf{u_{n,r}} \end{pmatrix}$$

   where r is the last column of $\mathbf{U}^T$ *with norm* $> \epsilon$.
   The polynomial is read as

$$K(q^{-1}) = 1 + \frac{u_{r+1,r}}{u_{r,r}} q^{-1} + \cdots + \frac{u_{n,r}}{u_{r,r}} q^{-(n-r)}$$

□

Remark 1

The main work of the algorithm is to

- *compute the inverse of* $\mathbf{L}_{11}$*;*
- *compute the product* $\mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12}$*;*
- *compute the LU-factorisation.*

Since all matrices have a very special toeplitz structure all these operations can be done very fast and accurate. The inversion of $\mathbf{L}_{11}$ *can be done very fast, since the lower-triangular toeplitz structure is preserved through inversion. It mean that we only have to compute the first column in the inverse. The product of two lower-triangular toeplitz matrices is also lower-triangular toeplitz. It means that we only have to compute the first column in the product. Many elements are zeros as well. The LU-factorisation can be made fast since we only need the* $\mathbf{U}$ *matrix.*

Remark 2

The common factor could be calculated by a LU-factorisation of the transpose of the full Sylvester matrix. But this does not use the structure of the problem.

Remark 3

Step 2 can be exchanged with:
QR-factorise as $QR = (\mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12})^T$
and replace U *with* R *in the steps that follows.*

The polynomial reduction is now completed with a simple polynomial division. Divide polynomials, $A(q^{-1})$ and $B(q^{-1})$ with $K(q^{-1}) = gcd(A(q^{-1}), B(q^{-1}))$.

We demonstrate the algorithm with an example.

Example 2.1 (Find greatest common devisor)
Let $A(q^{-1}) = (1+2q^{-1})(1+2q^{-1}+3q^{-2}) = 1+4q^{-1}+7q^{-2}+6q^{-3}$ *and* $B(q^{-1}) = (1+2q^{-1})(0+q^{-1}-q^{-2}) = 1q^{-1}+q^{-2}-2q^{-3}$. $A(q^{-1})$ *and* $B(q^{-1})$ *have a common factor* $K(q^{-1}) = 1+2q^{-1}$. *The Sylvester matrix is then defined by*

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix} = \left(\begin{array}{ccc|ccc} 1.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 4.00 & 1.00 & 0.00 & 1.00 & 1.00 & 0.00 \\ 7.00 & 4.00 & 1.00 & -2.00 & 1.00 & 1.00 \\ \hline 6.00 & 7.00 & 4.00 & 0.00 & -2.00 & 1.00 \\ 0.00 & 6.00 & 7.00 & 0.00 & 0.00 & -2.00 \\ 0.00 & 0.00 & 6.00 & 0.00 & 0.00 & 0.00 \end{array}\right). \tag{2.17}$$

Algorithm (2.1) now gives

$$\left(\begin{array}{ccc|ccc} 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 4.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 7.00 & 4.00 & 1.00 & -0.00 & 0.00 & 0.00 \\ \hline 6.00 & 7.00 & 4.00 & 3.00 & 0.00 & 0.00 \\ 0.00 & 6.00 & 7.00 & -3.00 & \mathbf{18.00} & 0.00 \\ 0.00 & 0.00 & 6.00 & -18.00 & \mathbf{36.00} & 0.00 \end{array}\right) \left(\begin{array}{ccc|ccc} 1.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & -3.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 3.00 & -3.00 & 1.00 \\ \hline 0.00 & 0.00 & 0.00 & 1.00 & 1.00 & -1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & -0.67 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{array}\right) \tag{2.18}$$

The last column in the left matrix is zero. By normalisation of the fifth column we get

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{2} \end{pmatrix}, \tag{2.19}$$

which is equal to the common factor.
□

2.5 Sensitivity to Modelling Errors

When designing a controller we have to have in mind that the design is based on a model of the system, not the true system. There will always be some modelling-errors. A study on the sensitivity to modelling errors can be found in [2]. In this section we will give an alternative study, based on the Sylvester matrix. It will be shown that the algorithm for solving the Diophantine equation (section 2.3) and polynomial reduction algorithm (section 2.4) gives useful information on how sensitive the system is to modelling errors. Often it is desireable to approximate the model by a lower-order model.

Suppose the process is described by $H^0 = \frac{B^0}{A^0}$, and that the control design is based on the model $H = \frac{B}{A}$. We do a control-design by solving the Diophantine equation $AR + BS = A_c$ for R and S. But the closed-loop will have the dynamics $A^0R + B^0S = A_c^{'}$, and this is not equal to A_c. The roots of $A_c^{'}$ characterise the closed-loop system. If any root is outside the unit-circle the system will be unstable. To investigate this further we will convert the problem to the matrix domain.

The Diophantine equation in matrix form is $\mathbf{A}x = b$, where $\mathbf{A}$ is the Sylvester matrix (2.11),

$$x = \begin{pmatrix} r_1 \\ \cdots \\ r_n \\ s_0 \\ \vdots \\ s_{n-1} \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} a_{c_1} - a_1 \\ \vdots \\ a_{c_n} - a_n \\ a_{c_{n+1}} \\ \vdots \\ a_{c_{2n}} \end{pmatrix} \tag{2.20}$$

The Diophantine equation is solved by $x = \mathbf{A}^{-1}b$. With this controller the closed-loop system will have the dynamics $b^{'} = \mathbf{A}^0 x = \mathbf{A}^0 \cdot \mathbf{A}^{-1}b$. If $\mathbf{A} \neq \mathbf{A}^0$ then $b^{'} \neq b$. Assuming that the modelling errors are given by $\delta\mathbf{A}$, we have $\mathbf{A} = \mathbf{A}^0 + \delta\mathbf{A}$ or that $\mathbf{A}^0 = \mathbf{A} - \delta\mathbf{A}$. The difference in closed-loop dynamics is then

$$\mathbf{b} - \mathbf{b}^{'} = \mathbf{A}\mathbf{x} - \mathbf{A}^0\mathbf{x} = \mathbf{A}\mathbf{x} - (\mathbf{A} - \delta\mathbf{A})\mathbf{x} = \delta\mathbf{A} \cdot \mathbf{x} = \delta\mathbf{A} \cdot \mathbf{A}^{-1}\mathbf{b}. \tag{2.21}$$

The norm of this is

$$||\mathbf{b} - \mathbf{b}^{'}|| \leq ||\delta\mathbf{A}|| \cdot ||\mathbf{A}^{-1}|| \cdot ||\mathbf{b}||. \tag{2.22}$$

A reasonable assumption is that the norm of the modelling error is proportional to the norm of the Sylvester matrix,

$$||\delta\mathbf{A}|| =\propto ||\mathbf{A}||. \tag{2.23}$$

With this assumption, (2.22) can be written as

$$\frac{||\mathbf{b} - \mathbf{b}^{'}||}{||\mathbf{b}||} \propto ||\mathbf{A}|| \cdot ||\mathbf{A}^{-1}|| = \kappa(\mathbf{A}). \tag{2.24}$$

The relative error in the closed-loop polynomial is thus proportional to the conditional number of the Sylvester matrix $\mathbf{A}$. The difference between the controllers based on $\mathbf{A}$ or $\mathbf{A}^0$ is

$$\mathbf{x} - \mathbf{x}^0 = \mathbf{A}^{-1}\mathbf{b} - \mathbf{A}^{0^{-1}}\mathbf{b} = (\mathbf{A}^{-1} - \mathbf{A}^{0^{-1}})\mathbf{b}. \tag{2.25}$$

Taking the norm of this leads to

$$\frac{||\mathbf{x} - \mathbf{x}^0||}{||\mathbf{x}^0||} \leq ||\mathbf{A}^{-1} - \mathbf{A}^{0^{-1}}|| \cdot ||\mathbf{A}^{0^{-1}}||. \tag{2.26}$$

2.6 Sensitivity to Modelling Errors Experiment

In the previous section we found that the condition-number is an important measure for how sensitive the system is to modelling errors. We will now exploit this further by a numerical experiment. We have done the following : the true plant, H^0, is of 3rd order. We have a model of this, H, such that the rise-times of the model differ by random numbers up to 10 percent from the rise-times of H^0 (by rise-time we mean the rise-time of a specific pole or zero).
The following has been done:

1. • compute a controller for the model, i.e. solve

$$AR + BS = A_{cl} \tag{2.27}$$

with respect to R and S.

• compute condition number for the Sylvester-matrix.

• compute a controller for the true plant, i.e. solve

$$A^0 R^0 + B^0 S^0 = A_{cl} \tag{2.28}$$

with respect to R^0 and S^0. A_{cl} is the same as for the model.

• compare the controllers.

• compute the characteristic equation for the closed-loop system by applying the controller based on the model on the true plant, i.e. compute

$$A^0 R + B^0 S = A'_{cl}. \tag{2.29}$$

• compute the roots of A'_{cl}.

2. • approximate the 3rd order model with a 2nd order model.

• approximate the 3rd order true system with a 2nd order system.

• repeat point 1 for these systems. The closed-loop system is computed as

$$A^0 R_{2nd} + B^0 S_{2nd} = A'_{cl}. \tag{2.30}$$

3. • approximate the 3rd order model with a 1st order model.

• approximate the 3rd order true system with a 1st order system.

• repeat point 1 for these systems. The closed-loop system is computed as

$$A^0 R_{1st} + B^0 S_{1st} = A'_{cl}. \tag{2.31}$$

The polynomial approximations have been done using the polynomial-reduction algorithm described in section 2.4.

Summary of experiment

The experiment described above was done on a number of models with different features. The results of these experiments are given in tables B.1- B.7 in appendix B, and can be summarised as:

- The condition number gives a useful indication on how sensitive the system is to modelling errors;
- Control design based on reduced models give in most cases improved robustness to modelling errors;
- High condition numbers are due to common factors or very fast poles (stable or unstable).

We propose that model reduction should be done when the condition number is greater than $\approx 10^3$. The control design should be based on the first model having a conditional number less than $\approx 10^3$.

2.7 Youla-Kučera Parameterisation

The Diophantine equation algorithm discussed in Section 2.3 gave a minimum degree solution to the Diophantine equation. Sometimes we want to have a controller that contains a specific factor, for example $(1-q^{-1})$ in $R(q^{-1})$ for integral action. This could be achieved with help of the Youla-Kučera parameterisation.

Assume that we have a solution R^0 and S^0 to the equation

$$AR^0 + BS^0 = A_{cl}^0 \tag{2.32}$$

The Youla-Kučera parametrisation then gives all rational stabilising controllers

$$\begin{aligned} S &= XS^0 + YA && (2.33) \\ R &= XR^0 - YB && (2.34) \end{aligned}$$

where X is chosen as a stable polynomial. The Y polynomial can be chosen completely free. This will be used to design stabilising controllers with constraints on the controller polynomials. The closed-loop characteristic polynomial with the controller (2.33) and (2.34) is then

$$\begin{aligned} AR + BS &= A(XR^0 - YB) + B(XS^0 + YA) \\ &= X(AR^0 + BS^0) + (BYA - AYB) \\ &= XA_{cl}^0 && (2.35) \end{aligned}$$

Example 2.2 (Integral action and Nyquist cut-off)
Suppose we want a controller with integral action and zero feedback-gain at the Nyquist frequency. This is achieved by constraint the controller polynomials to include the factors $R_d = 1 - q^{-1}$ in $R(q^{-1})$ and $S_d = 1 + q^{-1}$ in $S(q^{-1})$. These conditions can be written as $R(1) = R'(1)R_d(1) = 0$ and $S(-1) = S'(-1)R_d(-1) = 0$. We choose X as a stable polynomial given by $X(q^{-1}) = 1 + x_1q^{-1}$ and Y as $Y(q^{-1}) = y_0 + y_1q^{-1}$ where y_0 and y_1 can be chosen arbitrarily. The Youla-Kučera parametrisation (2.33) and (2.34) then becomes

$$\begin{aligned} X(-1)S^0(-1) + Y(-1)A(-1) &= S(-1) = 0 \\ X(1)R^0(1) - Y(1)B(1) &= R(1) = 0 \end{aligned}$$

which lead to

$$\begin{aligned} A(-1)(y_0 - y_1) &= -X(-1)S^0(-1) \\ B(1)(y_0 + y_1) &= X(1)R^0(1). \end{aligned}$$

We have the linear system of equations

$$\begin{pmatrix} A(-1) & -A(-1) \\ B(1) & B(1) \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} -X(-1)S^0(-1) \\ X(1)R^0(1) \end{pmatrix} \tag{2.36}$$

It is easily seen that this is solvable if and only if $A(-1) \neq 0$ and $B(1) \neq 0$. Solving for y_0 and y_1 gives $Y(q^{-1})$, and a controller with integral action and Nyquist cut-off is given by

$$\begin{aligned} S &= XS^0 + YA && (2.37) \\ R &= XR^0 - YB && (2.38) \end{aligned}$$

Remarks

If $B(1)$ is close to zero the solution to (2.36) is badly conditioned. But in this case the integral action is not needed, because the steady state error will be small anyway. Equation (2.36) can then be modified to

$$A(-1)y_0 = -X(-1)S^0(-1) \tag{2.39}$$

$$y_1 = 0. \tag{2.40}$$

The solution to (2.36) is also badly conditioned if $A(-1)$ is close to zero. But in this case the Nyquist cut-off is not needed, because the feedback gain at the Nyquist frequency is small anyway. Equation (2.36) can then be modified to

$$B(1)y_0 = X(1)R^0(1) \tag{2.41}$$

$$y_1 = 0. \tag{2.42}$$

Finally, if both $B(1)$ and $A(-1)$ are close to zero, either integral action or Nyquist cut-off is needed.
□

Another way to achieve constraints in the controller polynomials is to include the constraints in the Diophantine equation. Suppose we want the R-polynomial to contain a factor R_d and the S-polynomial a factor S_d. This can be done by solving the following Diophantine equation with respect to R' and S':

$$(AR_d)R' + (BS_d)S' = A_{cl}. \tag{2.43}$$

The controller polynomials are then

$$R = R'R_d \tag{2.44}$$

$$S = S'S_d. \tag{2.45}$$

The solution is, however, badly conditioned if there are close to common factors between R_d and B or between S_d and A. In the adaptive control case the control design is repeated every time-step using estimates of the plant polynomials. Thus a test of common factors should be a part of a numerically robust algorithm for real-time applications.

Chapter 3

Recursive Multiple Model Least Squares

3.1 Introduction

Very recently, new improvements has been achieved in system identification. A Least Squares method that simultaneously produces models of all orders up to a maximum order n, for the same amount of computation as the conventional least squares method, has been presented by Niu *et al* [17]. Problems associated with over- and under-parametrisation can be avoided with this method.
The chapter begins with a general discussion of recursive identification, whereafter the Multiple Model Least Squares method (MMLS) is described. An extended conditional updating rule, and a modification of the information matrix updating formula are also presented. In an appendix a proof/derivation of the MMLS method and the recursive LDL^T updating algorithm are given.
Our main contributions to the chapter are the modification of the MMLS method to work on regressors like (3.4), the distributed conditional updating rule, the modification of the information matrix updating formula (3.13), and the proof/derivation of the MMLS algorithm for regressors like (3.4) in Appendix C.

3.2 Recursive Identification

In recursive estimation methods the estimates are modified at each time step to take the new information into account. The standard method for this is the recursive Least Squares method with forgetting factor, defined by

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \tag{3.1}$$

$$K(t) = P(t)\varphi(t) \tag{3.2}$$

$$P(t) = \frac{1}{\lambda}\left(P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)}\right) \tag{3.3}$$

In the above algorithm the 'forgetting factor' λ ($0 \leq \lambda \leq 1$) weights the measurements, such that a measurement received n samples ago has a weighting proportional to λ^n.
This is a quadratic method and is known to give very fast convergence. But it is also associated with some potential implementation difficulties:

1. If the algorithm is to remain capable of tracking sudden parameter changes, the updating gain $K(t)$ has to be prevented from becoming too small as the

parameter estimates improve. Moreover, if good data is arriving and $K(t)$ becomes small, Equation (3.3) implies that P is near singular and roundoff error over many updating steps may cause the computed P matrix to become indefinite and the algorithm to break down [5];

2. On the other hand, when λ is less than 1 and little new information on θ is brought in by the observations, Equation (3.3) shows that P may increase as λ^{-t}. This is known as the 'burst phenomenon' or as 'estimator wind-up'. If P becomes large in this way then observation noise, or a sudden increase in information, may induce large spurious variations in θ [5];

3. θ can become severely 'biased' towards areas of the input-space most recently visited. Biasing results in that the estimates converge to a local region in the input space, thereby forgetting other regions which have been learnt. This problem is particularly problematic when there is miss-match between plant and model and can lead to severe problems in a nonlinear setting [16].

4. Usually θ are parameters of a linear transfer function. The order of the transfer function has to be chosen beforehand. An order higher than the true plant will lead to parameter divergence. This is known as over-parameterisation. An order lower than the true plant will, on the other hand, lead to time-varying parameters and bad convergence.

Many methods have been proposed to tackle points 1 and 2. One is to adjust the forgetting factor λ as a function of the quality of the present observation in combination with detection of parameter changes. Another method is to reset the P matrix in a periodic way or when a change in dynamics is detected. The numerical difficulties are normally dealt with a factorisation of P that maintain the symmetric and positive definite properties. Suitable factorisation are $P = UDU^T$, $P = LDL^T$ and $P = QDQ^T$. The updating formula (3.3) is then modified to update the factors (U,D), (L,D) or (Q,D) separately. When the covariance matrix P is factorised in one of these ways, it is easy to supervise and regularise the positive definite property by manipulation on the diagonal matrix, D. The (U,D) updating method is known as the Bierman method. The (Q,D) updating method was given by Hägglund.

A way to tackle point 3 is to use directional forgetting; only forget in the direction of P parallel to the new information vector; do not forget information of P orthogonal to the new information vector. Algorithms that utilise this idea have been given by Kulhavy and Hägglund. A simple alternative to these methods is to use conditional updating; only update when the present observation contains large enough new information about θ.

Point 4 have traditionally been dealt with trial and error or by estimating multiple models with different order through multiple estimation algorithms. Very recently Niu *et al* [17] has presented a method that gives estimates of all model orders up to a maximum model order and the corresponding loss functions. This to approximately the same computational effort as for estimating one model with the conventional recursive Least Squares method. One of many nice properties of this method is that estimates of lower order models are independent of estimates of higher order models. The user chooses a maximum possible model order. The most appropriate model order can then be easily decided based on the loss functions.

3.3 Multiple Model Least Squares

The Multiple Model Least Squares method (MMLS) is a Least Squares method (LS) that simultaneously estimates models of all orders up to a maximum order. This is done by a single matrix operation, and is therefore no more computational expensive than the conventional LS method. The method uses factorisation structures, which gives it improved numerical performance. The main advantage of the MMLS method is that it is robust against overparametrisation. Singularity problems occur in the higher order model estimates first. Lower order model estimates are independent of these singularities.

The fundamental difference between the MMLS and the LS methods is the arrangement of the data-vector. The MMLS includes the last output measurement $y(t)$ in the data vector. Another difference is that the data elements have to be sorted after time. The reason for this will be explained later.
The augmented data vector is defined [1] as

$$\varphi(t) = [-y(t)\ u(t-1)\ \ -y(t-1)\ \ldots\ u(t-n)\ \ -y(t-n)]^T \tag{3.4}$$

The highest order model , $\hat{\theta}^{(n)}$, then satisfies

$$\varphi^T(t) \begin{pmatrix} 1 \\ \hat{\theta}^{(n)} \end{pmatrix} = 0$$

Which can be rewritten in the usual form

$$y(t) = \varphi'^T(t)\hat{\theta}^{(n)}$$

The idea of the MMLS method is to interpret sub-elements of the data vector (3.4) as regressor vectors for lower order models. The sub-vector composed of element 3 to $2n+1$ serves as regressor vector for the model of order $(n-1)$,

$$\begin{pmatrix} -y(t-1) & u(t-2) & -y(t-2) & \cdots & u(t-n) & -y(t-n) \end{pmatrix} \begin{pmatrix} 1 \\ \hat{\theta}^{(n-1)} \end{pmatrix} = 0$$

A sub-vector of this vector will in the same way serve as regressor vector for a model of order $n-2$. This can be continued down to the regressor for the first order model,

$$\begin{pmatrix} -y(t-(n-1)) & u(t-n) & -y(t-n) \end{pmatrix} \begin{pmatrix} 1 \\ \hat{\theta}^{(1)} \end{pmatrix} = 0$$

The sub-vectors should be regressor-vectors of meaningful causal lower order models. This is the reason for the special structure of (3.4). With a data vector defined as (3.4), the k th order parameter vector is defined as

$$\hat{\theta}^{(k)} = \begin{pmatrix} b_1 & a_1 & \cdots & b_k & a_k \end{pmatrix}^T . \tag{3.5}$$

Define the augmented data matrix as

$$\Phi(t) = \begin{pmatrix} \varphi^T(1) \\ \varphi^T(2) \\ \vdots \\ \varphi^T(t) \end{pmatrix} \tag{3.6}$$

[1] The MMLS method given by Niu uses the augmented data vector

$$\varphi(t) = [-y(t-n)\ u(t-n)\ \ldots\ -y(t-1)\ u(t-1)\ \ -y(t)]^T.$$

The regressor-vector (3.4) will lead to a slightly different algorithm. The properties of the methods are however the same.

and then the augmented information matrix (AIM) as

$$S(t) = \Phi^T(t)\Phi(t) = \sum_{j=1}^{t} \varphi(j)\varphi^T(j). \tag{3.7}$$

Now decompose the AIM into a UDU^T decomposition

$$S(t) = U(t)D(t)U^T(t) \tag{3.8}$$

where $U(t)$ is a upper triangular matrix with unit diagonal, and $D(t)$ is a diagonal matrix. Least Squares estimates of models of all orders (up to n) are now found in

$$\Theta = U^{-T} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & 1 & 0 & \cdots & 0 & 0 & 0 \\ \hat{\theta}_{ff}^{(n)} & \vdots & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \hat{\theta}_{fb}^{(n-1)} & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \hat{\theta}_{ff}^{(n-1)} & \cdots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \hat{\theta}_{ff}^{(1)} & 1 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \hat{\theta}_{fb}^{(0)} & 1 \end{pmatrix} \tag{3.9}$$

The diagonal matrix, $D(t)$, is further equal to the loss functions for the different estimates,

$$D(t) = diag\left(\mathbf{J}_{ff}^{(n)} \quad \mathbf{J}_{fb}^{(n-1)} \quad \mathbf{J}_{ff}^{(n-1)} \quad \cdots \quad \mathbf{J}_{ff}^{(1)} \quad \mathbf{J}_{fb}^{(0)} \quad \mathbf{J}_{ff}^{(0)} \right) \tag{3.10}$$

Proofs/Derivations of these results are given in Appendix C.

The subscript $X_{ff}^{(k)}$ represents the k th order forward model, and $X_{fb}^{(k)}$ represents the k th order backward model.

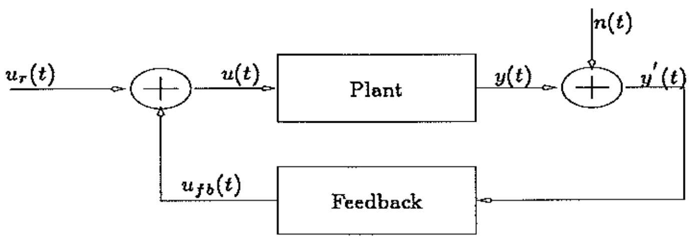


Figure 3.1: Forward-Backward models.

The forward models are models of the dynamics from $u(t)$ to $y'(t)$, i.e. the plant,

$$y'(t) = \frac{B_{ff}^{(k)}(q^{-1})}{A_{ff}^{(k)}(q^{-1})}u(t) + n(t),$$

where $n(t)$ is a disturbance (see Figure 3.1). The backward models are models of the dynamics from $y'(t)$ to $u(t)$,

$$u(t) = \frac{B_{fb}^{(k)}(q^{-1})}{A_{fb}^{(k)}(q^{-1})}y'(t) + u_r(t)$$

i.e. the feedback, thereof the name. The reference signal, $u_r(t)$, can be viewed as a disturbance for these backward models.

Over parameterisation or insufficient excitation are serious problems in standard least squares algorithms like (3.1) - (3.3). The problem has it's origin in that the information matrix, S(t), becomes rank deficit in these situations, and is therefore not invertible. The MMLS method avoids these problems, since inversion of $S(t)$ never take place. $S(t)$ is first factorised into UDU^T, whereafter U^T is inverted. Since $U(t)$ has unitary diagonal, it is always invertible. The singularities emerge as zeros in $D(t)$. But $D(t)$ never has to be inverted and is easy to regularise.

3.4 Recursive MMLS

The augmented information matrix is given by

$$S(t) = \sum_{j=1}^{t} \varphi(j)\varphi^T(j). \tag{3.11}$$

This can be separated into old and new information as

$$S(t) = S(t-1) + \varphi(t)\varphi^T(t) \tag{3.12}$$

where

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

Usually a forgetting[2] mechanism is incorporated into (3.12). The MMLS method gives some extended possibilities for this. It is for example possible to use

$$S(t) = \lambda S(t-1) + (1-\lambda)\varphi(t)\varphi^T(t), \tag{3.13}$$

which has the good property that $S(t)$ will converge to the mean value of $\varphi(t)\varphi^T(t)$,

$$\lim_{t\to\infty} S(t) = E\{\varphi(t)\varphi^T(t)\}, \tag{3.14}$$

which implies that the loss-functions will, independently of the forgetting factor λ, converge to the mean squared prediction error,

$$\lim_{t\to\infty} \mathrm{J}_{xx}^{((k))}(t) = E\{e_{xx}^{(k)^2}(t)\}, \tag{3.15}$$

$1 \le k \le n$. This makes interpretation of the loss-functions simple, which is very important in a multiple algorithm setting (see Section 5.2).
Further insight is obtained by writing (3.13) in the form[3]

$$S(t) = \lambda^t S(0) + (1-\lambda)\sum_{j=1}^{t} \lambda^{(t-j)}\varphi(j)\varphi^T(j). \tag{3.16}$$

[2] The standard recursive Least Squares method with forgetting factor uses

$$S(t) = \lambda S(t-1) + \varphi(t)\varphi^T(t).$$

[3] Conventional recursive Least Squares with forgetting factor leads to

$$S(t) = \lambda^t S(0) + \sum_{j=1}^{t} \lambda^{(t-j)}\varphi(j)\varphi^T(j).$$

From Equation (3.16) it can be seen that the initial value of the information matrix, $S(0)$, decays as λ^t. The information which arrived j steps ago are weighted as $(1-\lambda)\lambda^j$.

Define the augmented covariance matrix (ACM) as the inverse of the augmented information matrix,

$$C(t) = S^{-1}(t). \tag{3.17}$$

Now insert (3.8) into (3.17) and we have

$$\begin{aligned} C(t) &= U^{-T}D^{-1}U^{-1} && (3.18)\\ &= \Theta(t)D^{-1}\Theta^T(t) && (3.19) \end{aligned}$$

The MMLS estimates, Θ, can thus be obtained through a $LD'L^T$ transformation of $C(t)$. The L matrix is then equal to Θ. The loss-functions are calculated as the inverse of D'.

The goal is to find a recursive updating formula for the MMLS estimates. A theorem called the Matrix Inversion Lemma (MIL) takes us one step in that direction. The matrix inversion lemma states that

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}, \tag{3.20}$$

and is valid when A, C and $C^{-1}+DA^{-1}B$ are nonsingular square matrices (a proof can be found in [2]). Applying the Matrix Inversion Lemma to eq. (3.13) leads to

$$\begin{aligned} C(t) &= (\lambda S(t-1) + (1-\lambda)\varphi^T(t)\varphi(t))^{-1}\\ &= \frac{1}{\lambda}\left(S(t-1) + \varphi^T(t)\frac{1-\lambda}{\lambda}\varphi(t)\right)^{-1}\\ &= \frac{1}{\lambda}\left(S(t-1)^{-1} - \frac{S(t-1)^{-1}\varphi^T(t)\varphi(t)S(t-1)^{-1}}{\frac{\lambda}{1-\lambda}+\varphi^T(t)S(t-1)^{-1}\varphi(t)}\right)\\ &= \frac{1}{\lambda}\left(C(t-1) - \frac{C(t-1)\varphi^T(t)\varphi(t)C(t-1)}{\frac{\lambda}{1-\lambda}+\varphi^T(t)C(t-1)\varphi(t)}\right) && (3.21) \end{aligned}$$

where MIL is used to obtain the third equality.

It now remains to LDL^T factorise $C(t)$. This should be done in a way that takes advantage of the fact that we already have a LDL^T factorisation of $C(t-1)$. Thus $L(t)$ and $D(t)$ are computed as

$$\begin{aligned} L(t) &= f(L(t-1), D(t-1), \varphi(t)); && (3.22)\\ D(t) &= g(L(t-1), D(t-1), \varphi(t)). && (3.23) \end{aligned}$$

A derivation of such an algorithm is given in Appendix D.

3.5 An Extended Conditional Updating Rule

As mentioned in Section 3.2 it is necessary to modify the recursive Least Squares method to prevent 'windup' and 'forgetting'. A simple way to do this is to use conditional updating. The idea is to only use data that is informative enough.
Conditional updating is usually implemented as a hard decision rule. Either the data $\varphi(t)$ is used or it is not used. The MMLS approach offers some extended possibilities to soften up the decision. Even though $\varphi(t)$ is of no use for the model

of highest order, $\hat{\theta}^{(n)}$, it might carry information valuable for the lower order estimates, $\hat{\theta}^{(n-1)} \ldots \hat{\theta}^{(1)}$. A way to do this is to look at the one step ahead prediction error for all models.

Small prediction error can be the result of that

- the model is good;
- the excitation is poor.

Clearly we have no use of a data vector having small prediction error. The conditional updating rule we propose is to only update estimates to models with squared prediction error larger than a value ϵ.

One step ahead prediction error for all models are calculated by a single matrix multiplication,

$$\underline{e}(t) = \Theta^T(t-1)\varphi(t) \tag{3.24}$$

The extended conditional updating rule is then:

Update $\hat{\theta}^{(k)}$ and $J^{(k)}$ if and only if

$$e^{(k)^2}(t) > \epsilon, \tag{3.25}$$

where $e^{(k)}$ is the one step ahead prediction error for the k'th order model.

Notice that the data vector can be used for the i'th order model estimates even though it is not used for the j'th order model estimates.
The rule described above can be viewed as a kind of directional updating.

3.6 Initialisation

The MMLS method makes it easy to incorporate prior knowledge of the process into the initialisation. The standard LS method is usually initialised as

$$\begin{aligned} \theta(0) &= \theta_0 \\ P(0) &= c \cdot I \end{aligned}$$

where c is a constant and I is the unity matrix. In the MMLS method we can initialise the whole covariance matrix as

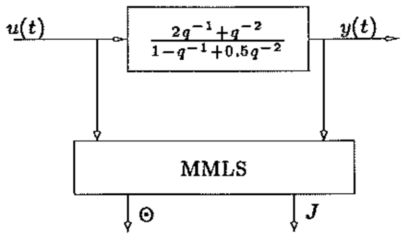
$$L(0) = \Theta_0 \tag{3.26}$$

$$D(0) = J_0^{-1}. \tag{3.27}$$

J_0 should be chosen as the expected mean squared prediction error of the Θ_0. A large value of J_0 will give Θ_0 big influence on the future estimates (see Equation 3.16). A small value of J_0 will in the same way reduce Θ_0's influence on future estimates.
The enhanced possibilities of initialisation can improve the initial convergence speed, which could be very useful in a covariance resetting scheme.

3.7 Example: Robustness to Over-Parametrisation

The following simulation example illustrates the robustness properties of the recursive MMLS method.
Assume that the process is described by the linear transfer function,



and that we do not have knowledge of the true order of the system, only that the order is less than or equal to three. Thus we set n to 3. The augmented data vector has then the form

$$\varphi(t) = \begin{pmatrix} -y(t) & u(t-1) & -y(t-1) & u(t-2) & -y(t-2) & u(t-3) & -y(t-3) \end{pmatrix}^T$$

The input and output are shown in Figure 3.2.

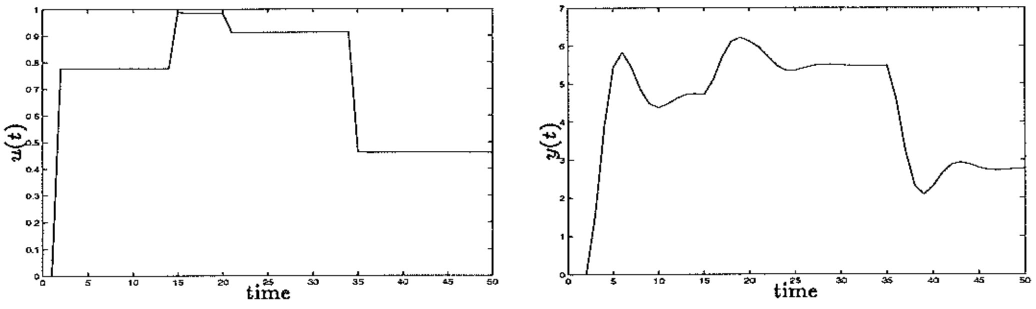


Figure 3.2: Input and output signals.

The system excitation is insufficient large parts of the time. The recursive MMLS uses forgetting factor $\lambda = 0.95$, conditional update value $\epsilon = 1 \cdot 10^{-4}$, initial models

$$\begin{aligned}
\frac{B_1(q^{-1})}{A_1(q^{-1})} &= \frac{q^{-1}}{1-0.9q^{-1}}, \\
\frac{B_2(q^{-1})}{A_2(q^{-1})} &= \frac{q^{-1}}{1-1.8q^{-1}+0.81q^{-2}}, \\
\frac{B_3(q^{-1})}{A_3(q^{-1})} &= \frac{q^{-1}}{1-2.7q^{-1}+2.43q^{-2}-0.72q^{-3}},
\end{aligned}$$

and initial loss functions $J(0) = 5 \cdot 10^{-5}$. We use sample-time $h = 1$s.

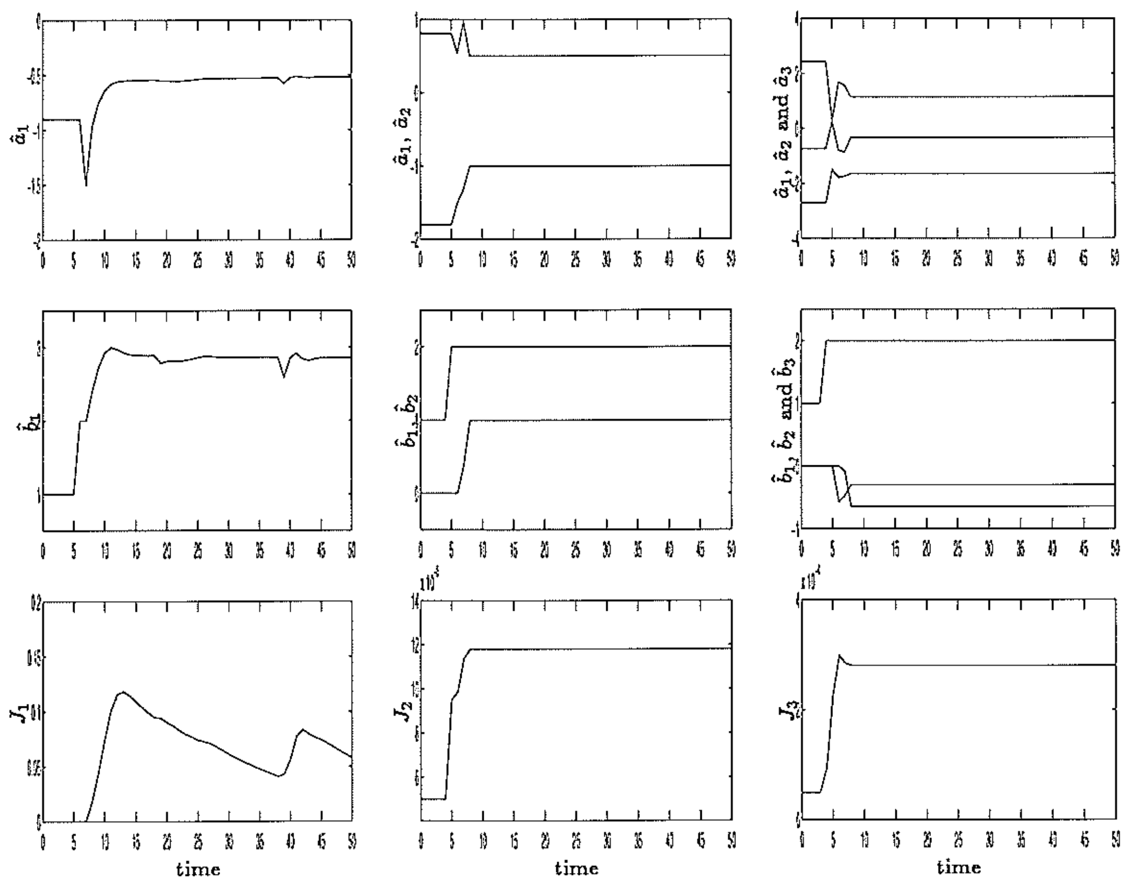


Figure 3.3: Estimates and corresponding loss functions for first, second and third order models.

The first order model estimates, shown to the left in Figure 3.3, never converge to any constant values. The loss function is relatively large, indicating that the estimates are bad. The second order model estimates, shown in the middle of Figure 3.3, on the other hand converges very fast to the correct values. The loss function is very small, and converges to a value close to the conditional updating value ϵ. The third order model estimates, shown to the right in Figure 3.3, also converges. The 3rd order loss function is however larger then the 2nd order lossfunction. This is due to the extended conditional updating rule described in Section 3.5. Since we don't gain anything by the third order estimates compared to the second order estimates, the second order estimates are a reasonable choice. If we want a model with higher accuracy we have to chose a lower value of ϵ.

Chapter 4

Nonlinear Modelling and Control using Local Model/Controller Interpolation

4.1 Introduction

In this chapter we will describe a nonlinear modelling and control structure consisting of interpolation of local linear models/controllers to a globally valid model/controller. The interpolation is done on the basis of 'scheduling' on a number of physical variables which are known to capture the system nonlinearities. We will first discuss the idea of interpolation.
The basic ideas of interpolating locally valid models/controllers can be found in Johansen and Foss 1992, 1993, Johansen 1994, Hunt and Johansen 1997, Murray-Smith and Johansen 1994 [9, 10, 8, 6, 14].

4.2 The Principle of Interpolation

Consider a system of the form

$$y \quad = \quad f(\mathbf{x}).$$

First order Taylor series expansion about a set of points, $\mathbf{x} = \mathbf{x}_i \ 1 \leq i \leq N$, gives locally valid approximations,

$$\hat{f}_i(\mathbf{x}) = f(\mathbf{x}_i) + \nabla^T f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_i}(\mathbf{x} - \mathbf{x}_i),$$

where $\mathbf{x}$, $\mathbf{x}_i \in R^n$ and $\nabla = \left(\frac{\partial}{\partial x_1}, \ \frac{\partial}{\partial x_2}, \ \cdots, \ \frac{\partial}{\partial x_n} \right)^T$. The linearised models $\hat{f}_i(\mathbf{x})$ are good approximations of $f(\mathbf{x})$ close to the linearisation points $\mathbf{x}_i$. Further away the local functions $\hat{f}_i(\mathbf{x})$ may give poor approximation of $f(\mathbf{x})$. One way to improve the global approximation properties is to interpolate the locally valid models to a globally valid model. The new interpolated model is then

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \rho_i(\mathbf{x}) \hat{f}_i(\mathbf{x}), \qquad (4.1)$$

where $\rho_i(\mathbf{x})$ is a interpolation function (or validity function) associated with the local model $\hat{f}_i(\mathbf{x})$.
The interpolation functions should be chosen with care, and be dependent on the nature of $f(\mathbf{x})$. The interpolation functions should have the properties:

- $\rho_i(\mathbf{x})$ should be close to 1 for all $\mathbf{x}$ near $\mathbf{x}_i$ and tend in a smooth fashion to 0 as the 'distance' to $\mathbf{x}_i$ increase;
- The sum of all validity functions should be equal to 1 for all values of $\mathbf{x}$, i.e. $\sum_{i=1}^{N} \rho_i(\mathbf{x}) = 1 \quad \forall\, \mathbf{x}$.

The following functions are examples of interpolation functions fulfilling the above properties:

- linear splines;
- normalised Gaussian functions:

$$\varrho_i(\mathbf{x}) = e^{-\frac{(\mathbf{x}-\mathbf{x}_i)^2}{2\sigma_i^2}},$$

$$\rho_i(\mathbf{x}) = \frac{\varrho_i(\mathbf{x})}{\sum_{j=1}^{N} \varrho_j(\mathbf{x})};$$

- cubic splines.

We illustrate the idea of interpolating local models with an example.

Example 4.1 (Interpolation of Local Models)
Consider the system described by $y = 2 - e^{-x}$. Linearisation about the points $x_1 = 0.5$ and $x_2 = 1.5$ gives the locally valid approximations

$$\begin{aligned} \hat{f}_1(x) &= (2 - e^{-0.5}) + e^{-0.5}(x - 0.5) \\ \hat{f}_2(x) &= (2 - e^{-1.5}) + e^{-1.5}(x - 1.5). \end{aligned}$$

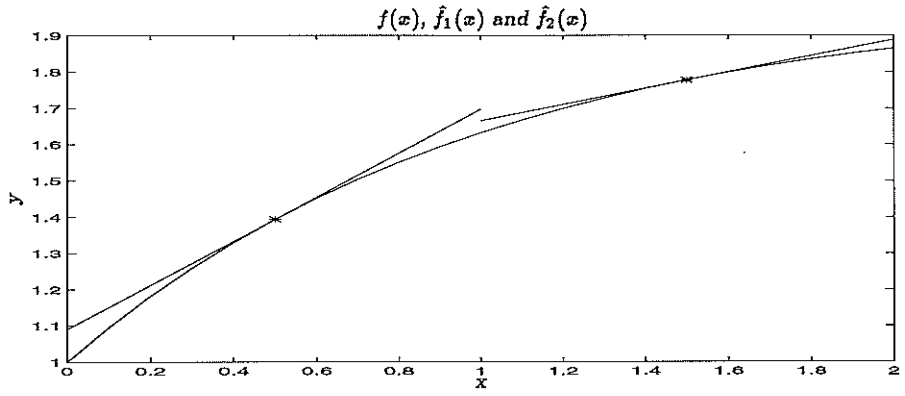


Figure 4.1: Exact model $f(x)$ and two local valid approximations, $\hat{f}_1(x)$ and $\hat{f}_2(x)$.

It is seen in Figure 4.1 that the linearised functions are close to $f(x)$ near the points of linearisation. We will now use interpolation between these two locally *valid linearisations to obtain a good approximation for all x. This interpolation can be written as*

$$\hat{f}(x) = \rho_1(x)\hat{f}_1(x) + \rho_2(x)\hat{f}_2(x).$$

We will employ the following functions for the interpolation,

linear interpolation

$$\rho_1(x) = \frac{x_2 - x}{x_2 - x_1}$$

$$\rho_2(x) = \frac{x_1 - x}{x_1 - x_2}$$

normalised Gaussian

$$\rho_1(x) = \frac{e^{-\frac{(x-x_1)^2}{2\sigma_1{}^2}}}{e^{-\frac{(x-x_1)^2}{2\sigma_1{}^2}} + e^{-\frac{(x-x_2)^2}{2\sigma_2{}^2}}}$$

$$\rho_2(x) = \frac{e^{-\frac{(x-x_2)^2}{2\sigma_2{}^2}}}{e^{-\frac{(x-x_1)^2}{2\sigma_1{}^2}} + e^{-\frac{(x-x_2)^2}{2\sigma_2{}^2}}}$$

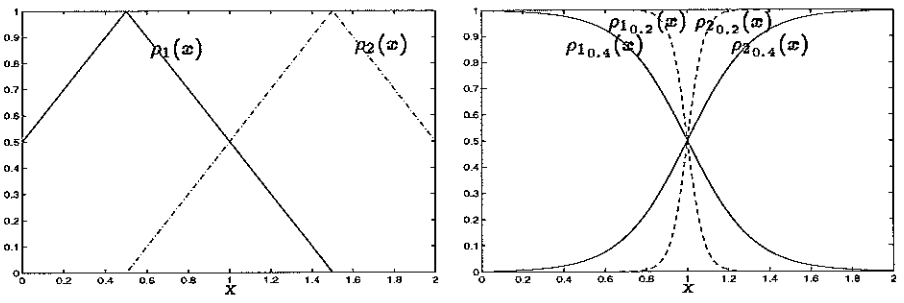


Figure 4.2: Examples of validity functions. Piecewise linear interpolation functions in the left figure, and normalised Gaussian interpolation functions in the right figure. The normalised gaussian functions have standard deviations $\sigma = 0.4$ or $\sigma = 0.2$.

In Figure 4.2 it can be seen that both these basis-functions have the properties that they are close to 1 near the points of linearisation, and that they decrease smoothly to zero as the distance to the point of linearisation increases.

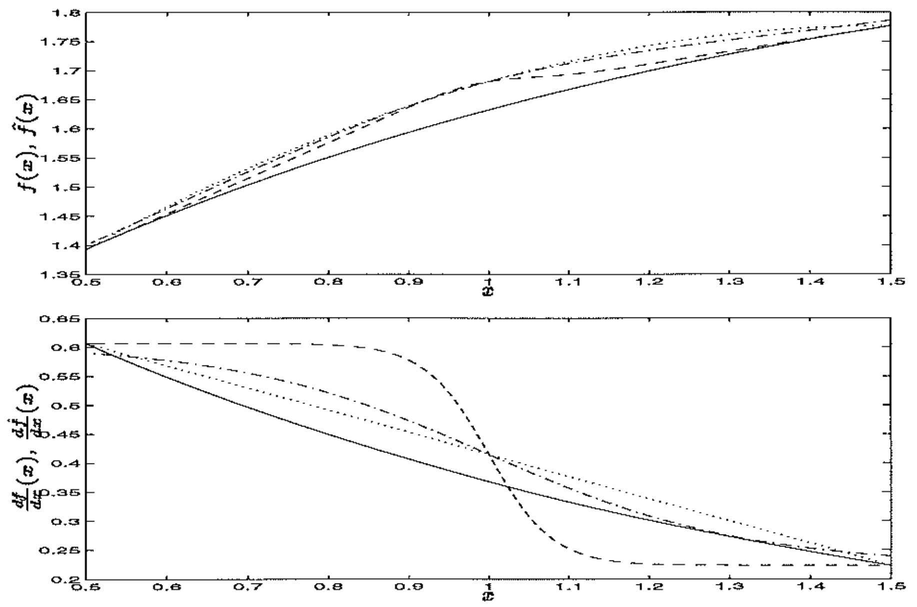


Figure 4.3: Approximation capacities of $f(x)$ and $\frac{df}{dx}(x)$. Solid : $f(x)$ and $\frac{df}{dx}(x)$, dotted : $\hat{f}_{\text{linear}}(x)$ and $\frac{d\hat{f}_{\text{linear}}}{dx}(x)$, dash-dotted : $\hat{f}_{\text{gauss}_{0.4}}(x)$ and $\frac{d\hat{f}_{\text{gauss}_{0.4}}}{dx}(x)$, dashed : $\hat{f}_{\text{gauss}_{0.2}}(x)$ and $\frac{d\hat{f}_{\text{gauss}_{0.2}}}{dx}(x)$.

The upper plot in Figure 4.1 shows that the interpolation-function with hard switching, $\hat{f}_{\text{gauss}_{0.2}}(x)$, gives the best approximation of $f(x)$ for most values of x. The lower plot in Figure 4.1 shows that $\frac{d}{dx}\hat{f}_{\text{gauss}_{0.2}}(x)$ is the worst approximation of $\frac{df}{dx}(x)$, while $\frac{d\hat{f}_{\text{linear}}}{dx}(x)$ and $\frac{d\hat{f}_{\text{gauss}_{0.4}}}{dx}(x)$ give relatively small approximation errors of $\frac{df}{dx}(x)$.
□

From this simple example we can see that the choice of basis-functions for (4.1) is different depending on the approximation criterion. The criteria to chose the set of basis functions which has the best capacity of approximating $f(x)$ could be questioned. A good criteria should take the approximation of $\frac{d}{dx}f(x)$ into account. See Kalkkuhl *et al* [12] for discussion.

4.3 Local Model Networks

Consider a nonlinear dynamical system having the form,

$$y(t) = f(y(t-1), \dots, y(t-n), u(t-1), \dots, u(t-n)) + e(t), \tag{4.2}$$

where u is input, y is output and e is noise. The system (4.2) is known as a NARX model[1] (Nonlinear AutoRegressive with eXogenous input). To simplify the notation we define the information vector ψ to be

$$\psi(t-1) = [y(t-1), \dots, y(t-n), u(t-1), \dots, u(t-n)]^T \tag{4.3}$$

whereafter (4.2) becomes

$$y(t) = f(\psi(t-1)) + e(t). \tag{4.4}$$

[1]The system (4.2) can be extended to more general model structures.

Assume that the main nonlinearities of the process can be captured by an operating vector $\tilde{\phi} \in \mathbf{R}^{n_{\tilde{\phi}}}$. This is a vector which can often be defined on a lower dimensional subspace of ψ, $\tilde{\phi} = H(\psi)$. We divide this operating space into different regimes, $R_i \subset \mathbf{R}^{n_{\tilde{\phi}}}$, $1 \leq i \leq N$. The regimes are chosen in such a way that the dynamics do not vary too much within a regime. An operating point,

$$\psi_i^* = [y_{i,1}^*, \cdots, y_{i,n}^*, u_{i,1}^*, \cdots, u_{i,n}^*]^T, \tag{4.5}$$

is associated with each regime. We now linearise the nonlinear system (4.4) around these operating points,

$$\begin{aligned} \hat{f}_i(\psi) &= f(\psi_i^*) + \nabla_\psi^T f(\psi)|_{\psi=\psi_i^*}(\psi - \psi_i^*) \\ &= f(\psi_i^*) + \theta_i^T(\psi - \psi_i^*) \\ &= \theta_i^T \psi + d_i \end{aligned} \tag{4.6}$$

where $\theta_i = [-a_{i,1}, \cdots, -a_{i,n}, b_{i,1}, \cdots, b_{i,n}]^T$ and $d_i = f(\psi_i^*) - \theta_i^T \psi_i^*$. The principle is illustrated in Figure 4.4.

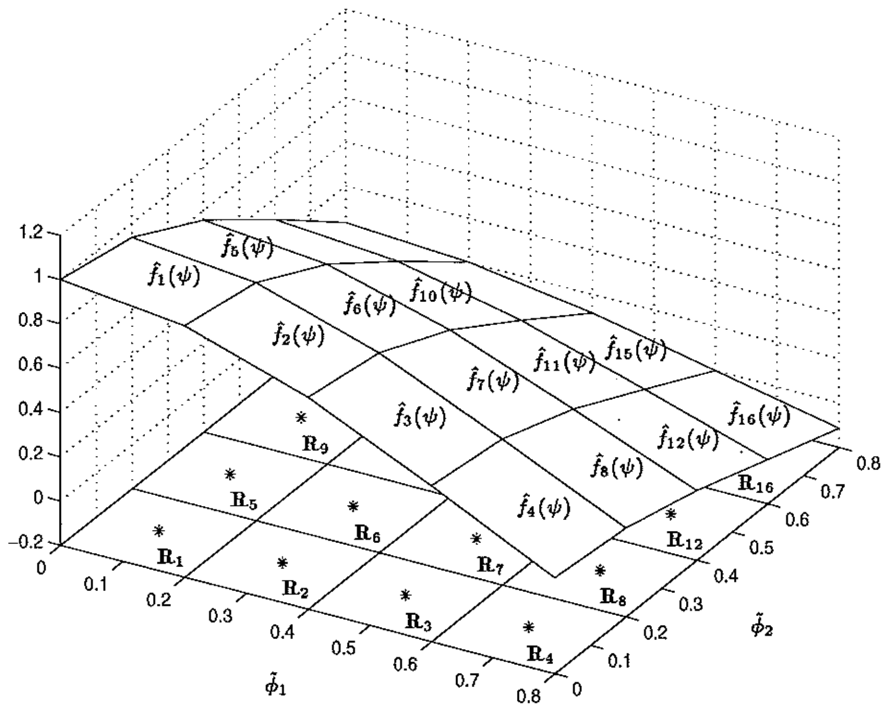


Figure 4.4: Local Model Network structure.

Locally, in the neighbourhood of $\tilde{\phi}^* = H(\psi_i^*)$, the system (4.4) behaves like a linear system with offset,

$$A_i(q^{-1})y(t) = B_i(q^{-1})u(t) + d_i + e(t). \tag{4.7}$$

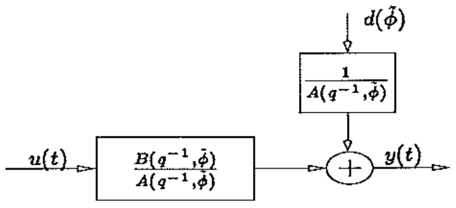
The idea is now to interpolate the local valid models to a model valid in the whole operating space. This can formally be written as

$$y(t) = \sum_{i=1}^{N} \rho_i(\tilde{\phi}(t-1))\hat{f}_i(\psi(t-1)) \tag{4.8}$$

and is known as a Local Model Network (LMN). The locally valid models in (4.8) do not have their own local state. Rather they share a common 'global' state defined by the information vector ψ.
When the local models are linear the local model network (4.8) becomes

$$y(t) = \psi^T(t-1)\left(\sum_{i=1}^{N} \rho_i(\tilde{\phi}(t-1))\theta_i\right) + \left(\sum_{i=1}^{N} \rho_i(\tilde{\phi}(t-1))d_i\right) + e(t). \tag{4.9}$$

The LMN (4.9) can be interpreted as a linear system with operating point dependent parameters and disturbance (see Figure 4.5),



$d(\tilde{\phi})$
$\frac{1}{A(q^{-1},\tilde{\phi})}$
$u(t)$
$\frac{B(q^{-1},\tilde{\phi})}{A(q^{-1},\tilde{\phi})}$
$+$
$y(t)$

Figure 4.5: Block diagram of an LMN. The LMN can be interpreted as a linear system with operating point dependent parameters and disturbance.

where

$$\begin{aligned} A(q^{-1},\tilde{\phi}) &= \sum_{i=1}^{N} \rho_i(\tilde{\phi})A_i(q^{-1}), && (4.10)\\ B(q^{-1},\tilde{\phi}) &= \sum_{i=1}^{N} \rho_i(\tilde{\phi})B_i(q^{-1}), && (4.11)\\ d(\tilde{\phi}) &= \sum_{i=1}^{N} \rho_i(\tilde{\phi})d_i. && (4.12) \end{aligned}$$

This linear structure will be used in the control design in the next section.

4.4 Nonlinear Control based on Local Model Networks

Assume that the process to be controlled is described by the NARX model (4.4). In most cases we do not have exact knowledge of this nonlinear model. Here, we will describe two control strategies based on the Local Model Network (4.9).

As mentioned in Section 4.3, the LMN (4.9) can be wieved as a time-varying linear system with operating point dependent parameters and an operating point dependent disturbance. This enables us to use standard linear control design methods, which are well understood. The controller parameters should, however, be made operating point dependent.

Online Control Design

An obvious solution is to recalculate the controller parameters at each time step.

At time t the process behaviour is characterised by $\tilde{\phi} = \tilde{\phi}(t)$. This means that the system at time instant t behaves like the linear system

$$y(t) = \frac{B(q^{-1}, \tilde{\phi})}{A(q^{-1}, \tilde{\phi})} u(t) + \frac{1}{A(q^{-1}, \tilde{\phi})} d(\tilde{\phi}). \tag{4.13}$$

Thus we simply do a linear control design based on this model (see Figure 4.6).

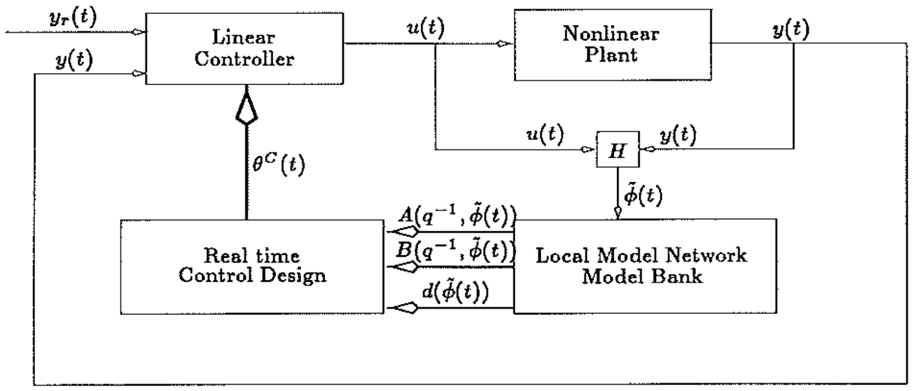


Figure 4.6: Online Control Design

This online control design strategy gives complete operating point based design. Model uncertainty and disturbances in specific operating points can thus be taken into account. We can give the system different rise times in different operating points. The transparent structure makes it easy to tune the controller online.
Online design has to be augmented with safeguards to make sure that the control design algorithm does not break down. There has to be a mechanism to find and cancel common factors. Since the control design can take some time, it is in many cases reasonable to first compute the control action, $u(t)$, based on the controller parameters at the previous time instant $\theta^C(t-1)$, and then compute the new controller parameters $\theta^C(t)$.
The online control design approach is favourable in an adaptive/selftuning setting. Here, recalculation of the controller parameters are necessary anyway.

Local Controller Network

An alternative to real time control design is to interpolate locally valid controllers. For each of the locally valid models in the LMN (4.9), $1 \leq i \leq N$, a controller θ_i^C is designed. These locally valid controllers are then interpolated to a globally valid controller,

$$\theta^C = \sum_{i=1}^{N} \rho_i(\tilde{\phi}) \theta_i^C, \tag{4.14}$$

where $\tilde{\phi}$ is the same operating point vector as in the LMN (see Figure 4.7).

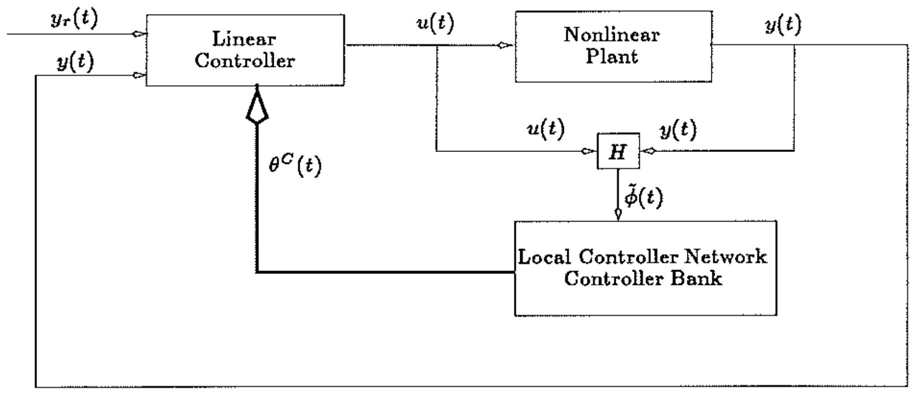


Figure 4.7: Local Controller Network.

The local controller network is an approximation of thė online control design strategy. This is clear since control design, $C(\cdot)$, is a nonlinear operation,

$$\alpha_1 \cdot C(\theta_1) + \alpha_2 \cdot C(\theta_2) \neq C(\alpha_1 \cdot \theta_1 + \alpha_2 \cdot \theta_2). \tag{4.15}$$

The resulting system is by construction stable close to points of linearisation. Standard tools, such as Bode and Nyquist plots, can be used to analyse stability ,performance and robustness properties for each fixed operating points. It should be noticed that stability of the local controllers not necessarily leads to stablity in between two regimes. Analysis and guidelines for the discrete-time local controller networks are presented in [6]. Often the global stability properties have to be verified through extensive simulations.
Local controller networks have strong links to gain scheduling control. It can be viewed as a systematic method for the scheduling procedure. One difference to gain scheduling is that the linearisation points do not necessarily need to be at equilibrium points in the LCN formulation.

Chapter 5

Recursive Identification of Local Model Networks

5.1 Introduction

In this chapter we combine the MMLS algorithm with local model networks and switching between local model networks with different properties. The MMLS algorithm identifies the locally valid models in the local model networks. Different ordered local model networks are thus produced simultanously. To cope with time variations in the process and to automatically get correct ordered models in all operating points, we switch between fixed and adaptive local model networks with different properties based on online model validation.
The author's main contribution to this chapter are the ideas to use the MMLS in the local model networks setting, and to switch between fixed and adaptive local model networks based on online model validation. The regressor filter difficulty, pointed out in section 5.4, is also due to the author.

5.2 Recursive Identification of multi-order Local Model Networks

Our goal is to identify the nonlinear dynamical system

$$y(t) = f(\psi(t-1)) + e(t) \tag{5.1}$$

in real time (see Section 4.3 for definitions). The function could also vary with time. In Chapter 4 we showed that the NARX system (5.1) can be modelled by a Local Model Network (4.9). We will now extend the Local Model Network approach to provide online tuning of the locally valid models in the local model networks.
Again we assume that the process behaviour at time t is characterised by an operating vector $\tilde{\phi} = \tilde{\phi}(t) \in \mathbf{R}^{n_{\tilde{\phi}}}$. We divide this operating space into different regimes, $R_i \subset \mathbf{R}^{n_{\tilde{\phi}}}$, $1 \leq i \leq N$. The regimes are chosen in such a way that the dynamics is approximately the same within a regime. In each of these regimes we now employ a recursive Multiple Model Least Squares algorithm (rMMLS). Thus we identify locally valid multi-order models Θ_i, $1 \leq i \leq N$, where Θ is defined by Equation 3.9.

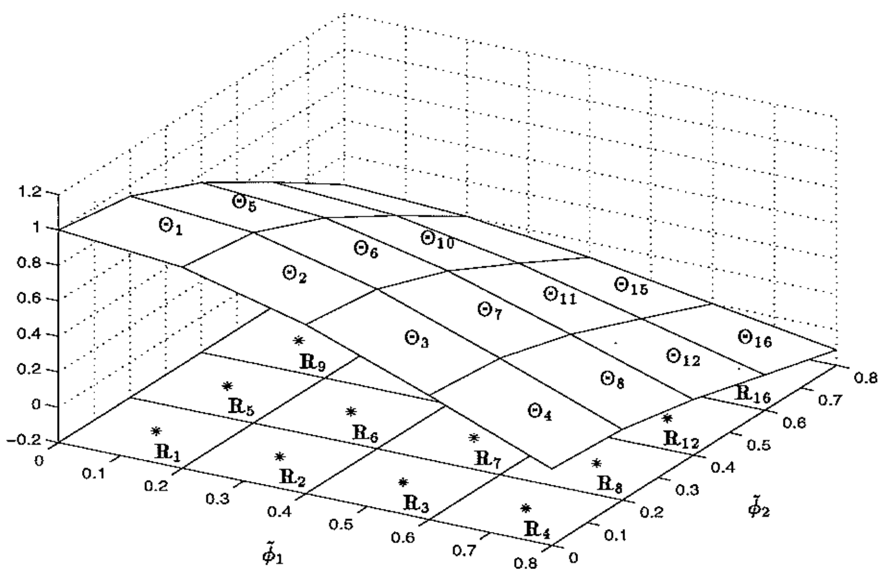


Figure 5.1: Regime based recursive Multiple Model Least Squares. One recursive MMLS algorithm is employed in each regime.

The locally valid multi-order models are then interpolated to a globally valid multi-order model, Θ, through

$$\Theta = \sum_{i=1}^{N} \rho_i(\tilde{\phi})\Theta_i. \tag{5.2}$$

Equation (5.2) can be interpreted as multi-order Local Model Networks. We have one LMN of each order,

$$\hat{\theta}^{(1)} = \sum_{i=1}^{N} \rho_i(\tilde{\phi})\hat{\theta}_i^{(1)} \tag{5.3}$$

$$\vdots$$

$$\hat{\theta}^{(n)} = \sum_{i=1}^{N} \rho_i(\tilde{\phi})\hat{\theta}_i^{(n)}. \tag{5.4}$$

This can be very useful, since the plant model order could be different in and in-between different regimes. With the multiple order structure there is always one LMN with the correct model order. The LMN with the correct model order could be chosen by online model validation (see Section 5.3).
The MMLS method has the nice property that it simultaneously gives loss-functions associated with the models. Thus, we can directly see how good the Local Model Networks are in specific operating points. If the loss-functions are large in a specific operating regime, we can directly carry out more experiments here.
When so many identifiers are working at the same time, it is clearly of greatest importance to have robust estimation algorithms. We have to deal with all problems associated with linear estimation, plus some new difficulties associated with the nonlinear nature of the plant.

The local rMMLS algorithms can use different forgetting factors, conditional updating values and initial parameters in different regimes. This allows specific operating conditions in a regime to be taken into accont. The small modification of the information matrix updating formula, described in Section 3.4, can be of great practical use in these situations. With this modification the loss-functions will be equal to the mean squared prediction error regardless of the forgetting factor λ. This makes the system very flexible and the lossfunctions easy to interprete.

Notice that only one of the multi-order models is identified at a time. It means that no more computational effort is needed than for a single recursive MMLS algorithm.

5.3 Switch Between Fixed and Adaptive Local Model Networks

The purpose of the adaptive local model networks strategy is to control a timevarying nonlinear plant. Many difficult situations can occour in such an environment, which have to be taken into accout for successful application.

- The plant complexity can be different in different operating points. For example, the model order can be different in different operating points.

The adaptive local model networks strategy adresses this problem by estimating local model networks of multiple orders. Provided we chose the maximum model order high enough, there is always one local model network with correct order in the entire operating space.

- The recursively estimated local model networks have, however, the drawback that it can take some time to tune up the entire network estimates. This can be a problem when there is a rapid change in the plant dynamics.

The locally valid model estimates can adapt fast to these changes. They are not penalised by neighbouring models that are poorly tuned.
The idea we have investigated is to build up a bank of local model networks with different orders and properties, and switch between these local model networks on the basis of on-line model validation. See Figure 5.2.

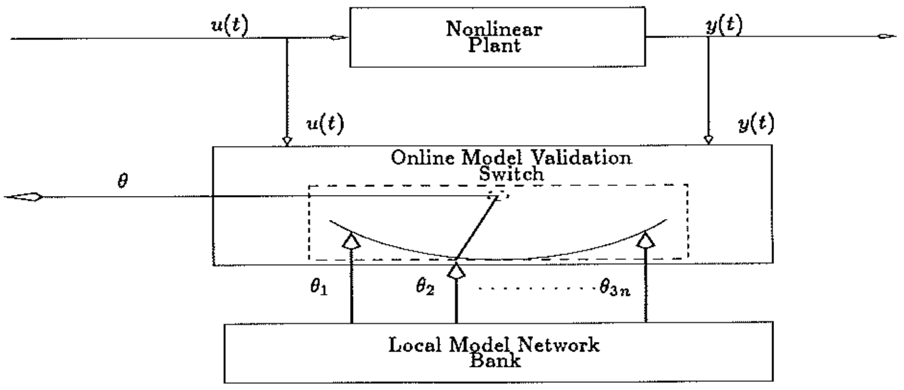


Figure 5.2: Switching Structure. Choose the lowest order model that is good enough. If no model is good enough, choose the 'best' one. n is the maximum model order.

To cope with different situations we have used the following types of models:

1. Θ_{LMN} : local model network estimates of order 1 to n;
2. Θ_{Local} : local estimates of order 1 to n from the present regime of operation;
3. Θ_{Fix} : fixed local model networks of order 1 to n.

These different types of models complement each other. Θ_{LMN} will become the best model when the system has tuned up. Θ_{Local} can respond quickly to changed plant dynamics. Θ_{Fix} is a backup to provide security. The system will never be worse than for a fixed local model network.

There are a many possible switching rules that could be used. The criteria should, however, be such that switching only takes place when necessary. Many models will be good when the system is poorly excited. This motivates the following switching rule:

Choose the model with lowest degree that is good enough.
If no model is good enough, choose the 'best' one.

An Akaike information criterion test (AIC) can, for example, be used as a criteria. The AIC-test has the form [11, 13]

$$AIC_i(p) = log(\hat{\sigma}^2(\theta_i^{(p)})) + \frac{2p}{M} \tag{5.5}$$

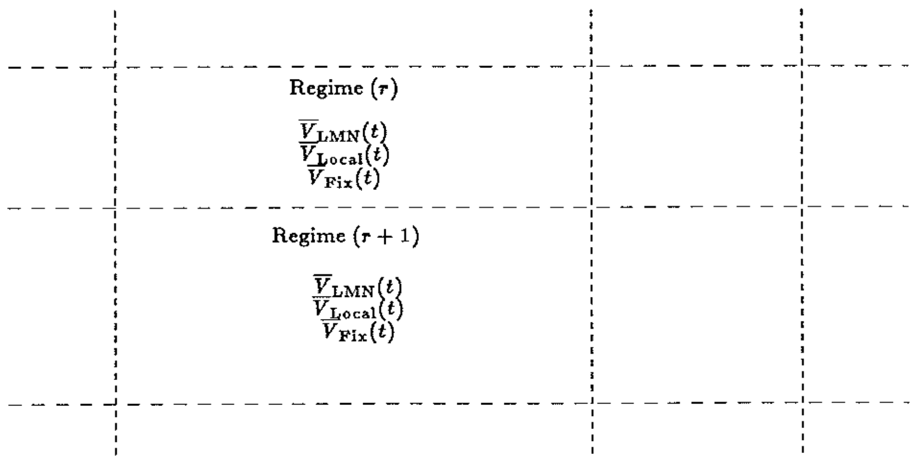
where p is model order, M is the measurement-memory length and $\hat{\sigma}^2$ is the variance for that estimate. The estimated variance, $\hat{\sigma}^2$, can be calculated as:

$$\hat{\sigma}^{(p)^2}_{\text{LMN}} = \frac{1}{M}\sum_{i=t-M}^{t} \epsilon_i^2(\hat{\theta}^{(p)}_{\text{LMN}}(i)) \tag{5.6}$$

$$\hat{\sigma}^{(p)^2}_{\text{Local}} = \frac{1}{M}\sum_{i=t-M}^{t} \epsilon_i^2(\hat{\theta}^{(p)}_{\text{Local}}(i)) \tag{5.7}$$

$$\hat{\sigma}^{(p)^2}_{\text{Fix}} = \frac{1}{M}\sum_{i=t-M}^{t} \epsilon_i^2(\hat{\theta}^{(p)}_{\text{Fix}}(i)) \tag{5.8}$$

where ϵ_i is the prediction error for the i'th order model. All models have associated loss functions in all regimes (see Figure 5.3).



Regime (r)
$\overline{V}_{\text{LMN}}(t)$
$\overline{V}_{\text{Local}}(t)$
$\overline{V}_{\text{Fix}}(t)$
Regime ($r+1$)
$\overline{V}_{\text{LMN}}(t)$
$\overline{V}_{\text{Local}}(t)$
$\overline{V}_{\text{Fix}}(t)$

Figure 5.3: Separate lossfunctions in different regimes. $\overline{V}_X = [V_X^{(n)} \cdots V_X^{(1)}]$. X denotes LMN/Local/Fix model.

The model's performances in a specific regime are remembered from the last time the system were operating in that regime.

5.4 A Local Model Network Estimation Difficulty

Some minor modifications of the estimation algorithm are needed to make the recursive estimation of local model networks work well in a nonlinear setting. The problem is best illustrated with an example.
Assume that the plant is described by

$$y(t) = \arctan(y(t-1)) + u(t-1)$$

The plant can locally be viewed as a linear system with an operating point dependent pole.
We divide the system into three different regimes, according to

$$\begin{aligned} R_1 &= \{-\infty \leq y(t-1) \leq -0.86\} \\ R_2 &= \{-0.86 \leq y(t-1) \leq 0.86\} \\ R_3 &= \{0.86 \leq y(t-1) \leq \infty\} \end{aligned}$$

See Figure 5.4.

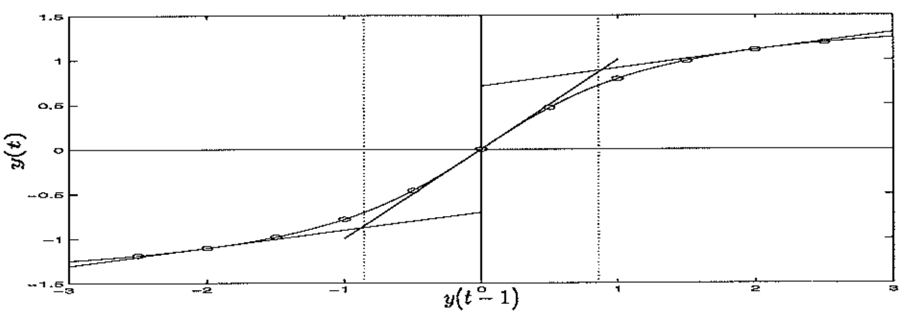
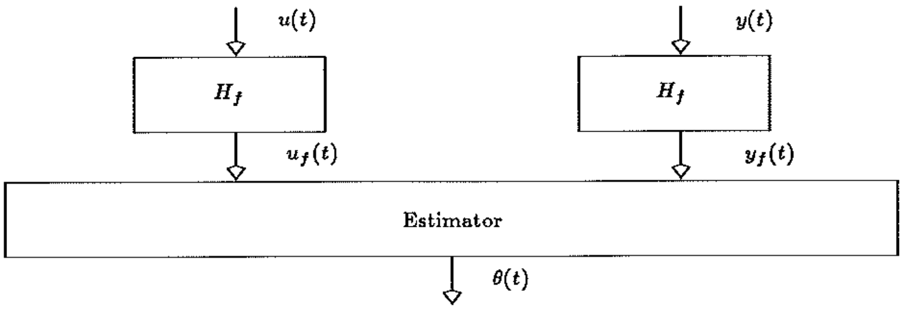


Figure 5.4: NARX model and locally valid linearisations.

In each regime we attempt to estime a locally valid linear approximation of the system. It is then desired that all elements in the local regressors are generated in that specific regime. Otherwise, the estimator will be confused because some of the elements in the regressor are generated from a system with different parameters. A way to tackle the problem is to:
suspend the parameter updating until the system has been in the same regime for at least the order of the model time-steps.
This is easily implemented by keeping track of the regime of operation over time.

5.5 Regressor Filter

It is common practice to filter the data before they are used for parameter estimation [1].



The purpose of the regressor filters is to reduce the relative influence of noise and unmodelled dynamics in the estimation. For control applications it is not necessary to have perfect knowledge of the process. It is usually enough to have an accurate model for frequencies around the crossover frequency.
Assume that the process can be described by

$$y(t) = G(q)u(t) + v(t) \tag{5.9}$$

where $v(t)$ is a disturbance, typically with low and high frequency components. This will cause difficulties in the parameter estimation; the Least Squares method works best when the disturbance is white noise. Thus the filter H_f should be of bandpass type. Application of the filter H_f to Eq. (5.9) gives

$$y_f(t) = G(q)u_f(t) + v_f(t) \tag{5.10}$$

where $y_f(t) = H_f(q)y(t)$, $u_f(t) = H_f(q)u(t)$ and $v_f(t) = H_f(q)v(t)$.
The filter will however also have the negative effect that it will spread out the meassurement over a longer timeperiod, and therfore make it more difficult to decide in which regime a measurement is generated; a filter is a weighted sum of the past measurements. To avoid interactions between regimes it is then necessary to wait until the transients of the filter have decayed before the data can be used for estimation. Of course it is desirable to make this time as short as possible.
FIR-filter (Finite Impulse Response filter)
An FIR-filter is a filter of the type

$$u_f(t) = b_0u(t) + b_1u(t-1) + \cdots + b_nu(t-n) \tag{5.11}$$

The filtered signal is computed as a weighted sum of the n last measurements $u(t)$. It means that we know that a measurement $n+1$ time-units ago will not have any influence on the filtered signal. A possible scheme is thus to suspend the the parameter updating $n+1$ steps after a regime shift. If, however, a filter with high attenuation is needed, the filter length has to be very long and much data will be thrown away.
IIR-filter (Infinite Impulse Response filter)
A IIR-filter has the form

$$u_f(t) = \frac{B(q)}{A(q)}u(t) \tag{5.12}$$

With this filter constellation it is possible to obtain high attenuation with moderate filter length. But since the filtered signal in a recursive way is function of all past measurement, it might take long time before the influence of a measurement has decayed. One possibility is to reset the filter when a change of regime is done, and wait until the filter transients have decayed before using the data for estimation.

Estimate Offset
An alternative to filtering is to have a model of the disturbance or parts of it in the estimation model. Since we know that the major part of the disturbance $v(t)$ is of low frequency, it could be favourable to have a model of the offset in the regressor. This is easily obtained by extending the regressor vector with a '1',

$$y(t) = \varphi^T\theta + d = \begin{pmatrix} \varphi^T & 1 \end{pmatrix} \begin{pmatrix} \theta \\ d \end{pmatrix} = \varphi_e^T\theta_e. \tag{5.13}$$

This removes the necessity for the filter to attenuate the low frequencies. The filter is then reduced (if necessary at all) to a lower order low-pass filter to attenuate the high frequency components of $v(t)$.
The offset d could be used in a feed-forward manner to compensate for the offset disturbance. But since the estimate could be inaccurate it might be better to use a controller with integral action.

5.6 Simulation Example: Recursive Identification of a NARX System

Assume that the process is described by the first order NARX model,

$$y(t) = \arctan(y(t-1)) + u(t-1).$$

which locally can be viewed as a linear system with an operating point dependent pole (see Figure 5.4). We chose to schedule on the variable $y(t-1)$, using the validity function set shown in figure 5.5.

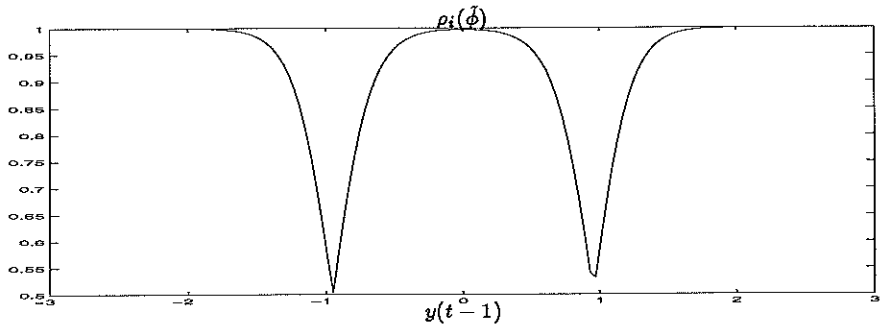


Figure 5.5: Validity function set.

The input and resulting output are plotted below.

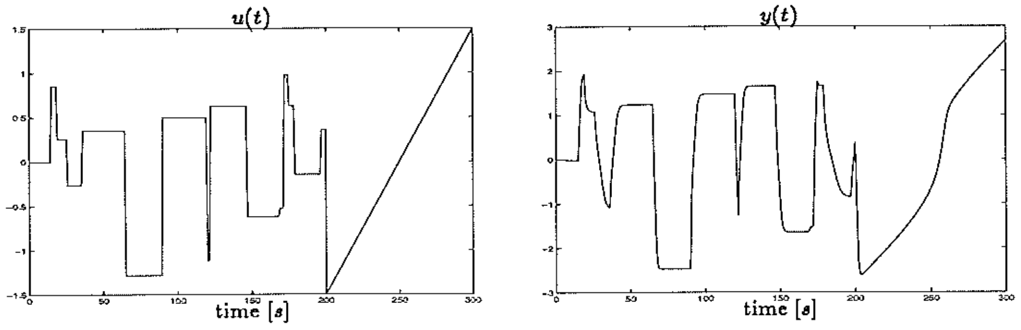


Figure 5.6: Input and output signals.

The first 200 seconds the input is a step signal with random amplitude. The last 100 seconds the input is a ramp-signal. Recursive MMLS algorithms are used to estimate locally valid first order ARX models with offsets. The locally valid models are initiated as:

$$y(t) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}u(t) + \hat{d}(t) = \frac{q^{-1}}{1 - 0.9q^{-1}}u(t) + 0.$$

The loss-functions are initiated to $J(0) = 1 \cdot 10^{-6}$. The MMLS algorithms uses forgetting factor $\lambda = 0.95$, conditional update value $\epsilon = 1 \cdot 10^{-4}$ and 1 step suspended updating after regime-shift.

The true and estimated plant parameters are shown in Figure 5.7 and Figure 5.8.

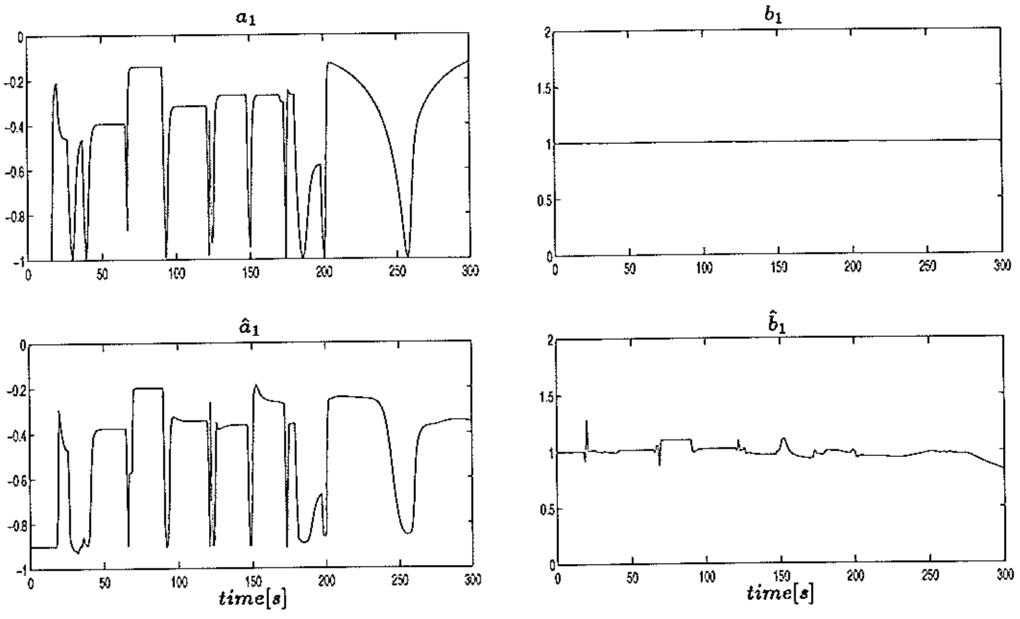


Figure 5.7: True and estimated plant parameters versus time. The upper plots shows the true plant parameters, a_1 to the left and b_1 to the right. The lower plots shows the parameter estimates $\hat{a}_1$ to the left and $\hat{b}_1$ to the right.

After some learning time, it can be seen that the estimates can follow the very fast changes of the parameters. This would have been very difficult to achieve with a single estimator. The estimates could have been improved even more with a better validity function set. The validity functions used here are slightly too steep. Another way to improve the estimates is to increase the number of models.

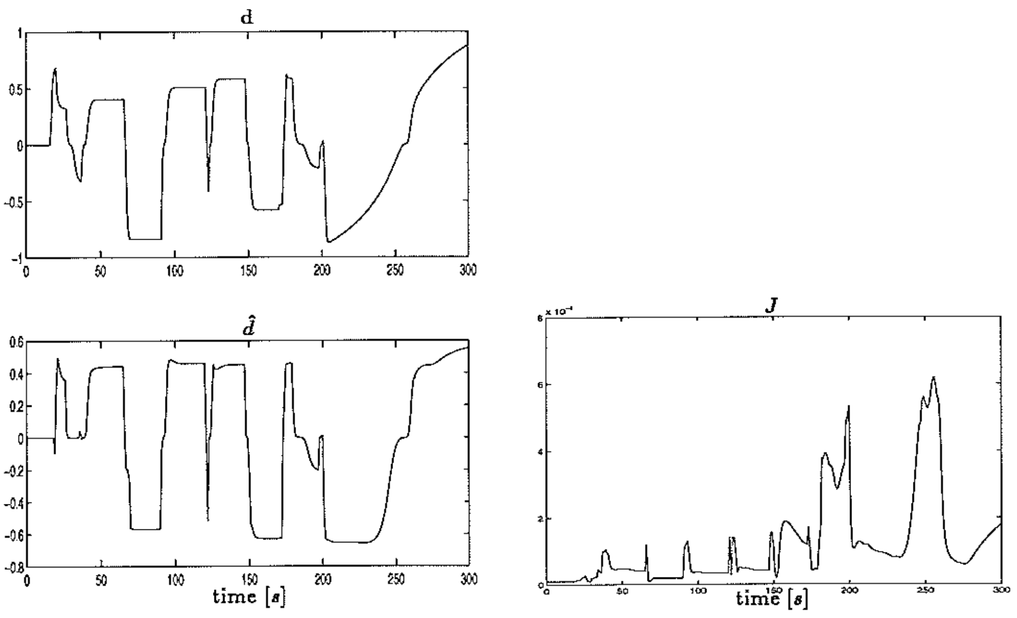


Figure 5.8: True plant offset and corresponding estimate versus time. The upper left figure shows the true offset from the plant. The lower left figure shows the estimate. The lower right figure shows the loss function.

The offset estimate is also very good most of the time. It follows the very fast changes of the plant offset. If we had not had the offset in the estimation model, this offset would have acted on the system as a disturbance, spanning the whole frequency range. Thus, offset estimation is a good idea. The loss function is shown in the right figure. The loss function is initialised to a low value. The large fluctuations with respect to time are due to that the plant is operating in different regimes at different times. From the loss function we can actually see how good the estimates are as a function of operating point. The loss function is equal to the mean squared prediction error (see Section 3.4).

Chapter 6

Adaptive Local Model Networks Strategy for Nonlinear Control

6.1 Introduction

Many real systems are difficult and time consuming to model in advance. The process is often influenced by external actions. An adaptive system can be very useful in these situations. An adaptive system is a system that can adapt its behavior to changes in the environment. Here we will present a so-called indirect adaptive control system or self-tuning control system. It is called indirect because a model is first estimated, whereafter a controller is designed based on the identified model [1].

Selftuning Local Model Networks

In Section 4.3 we showed that many nonlinear processes can be modeled with Local Model Networks (LMN). It can, however, be very time consuming to estimate this LMN in advance. Often it is difficult to collect enough good data which exploit all process dynamics in different operating points. We are convinced that Adaptive Local Model Networks can be of great practical use in these situations. An Adaptive Local Model Network is an Adaptive Multi-model Control strategy where a Local Model Network is estimated in real-time, and a controller is designed online based on the Local Model Network. Since all modeling and control design are done online we can directly see the resulting control performance in different operating points. If the performance is bad in some operating points we can directly do more experiments here. When we are satisfied with the control performance we can turn off the identification. A fixed Local Controller Network can then be used to approximate the controller parameters.

Adaptive Local Model Networks Control Strategy

Many processes are not only nonlinear but also time-varying. The process might be influenced by the surrounding environment. An Adaptive Local Model Network combined with switching could be used even in this case. The idea is to build up a bank of fixed and estimated local model networks, and switch between these networks based on online model validation.

6.2 An Adaptive Local Model Networks Strategy For Nonlinear Control

The full Adaptive Local Model Network Strategy is a combination of recursive multiple model identification, Local Model Networks, switching and online controller design. Full treatments of these issues are given in Chapters 2-5. Here, a short summary is given:
The operating space is divided into disjoint regimes. In each regime a recursive Multiple Model Least Squares method is employed to identify locally valid linear models of all orders up to a maximum order n. Only the multiple model that is in the present regime of operation is updated. The locally valid multiple-order models are now interpolated to globally valid models of multiple order, valid around the present point of operation. Online validation can be used to determine the 'best' of these models. Candidates for the 'best' model are not only the interpolated multiple order models but also the estimated and initial multiple-order models in the present regime of operation. An Akaike Information Criterion test (AIC) or Minimum Description Length test (MDL) gives the model-type and model order that fit the system 'best'. A lower order model is chosen if it is good enough, even though there are better models of higher orders. The 'winning' model is now the basis for online controller design. Model reduction simplifies the model, if necessary, before control design takes place to give the closed-loop system the desired behavior.

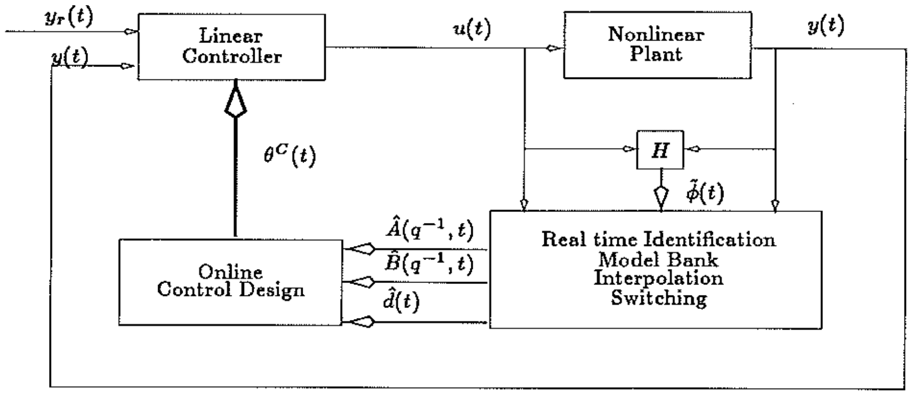


Figure 6.1: Adaptive Local Model Networks Strategy For Nonlinear Control.

The Adaptive Local Model Networks Strategy is a kind of Multiple Model Adaptive Control strategy (MMAC). The main difference between our approach and other MMAC strategies is that we are interpolating the model estimates and design one controller based on the interpolated model instead of designing multiple controllers and interpolate the controllers. Another difference is that we only identify one model at a time. The Adaptive Local Model Network Strategy does not need much more computation effort than a conventional indirect adaptive control. More memory is however needed.

6.3 Simulation Example

In this simulation example we will compare the adaptive local model networks strategy with conventional adaptive control, and show that it can give significantly improved control performance of highly nonlinear processes.

Assume that the plant is described by

$$y(t) = 2\tanh(y(t-1)) + 3u(t-1) \tag{6.1}$$

which is a unstable, highly nonlinear process. See Figure 6.2.

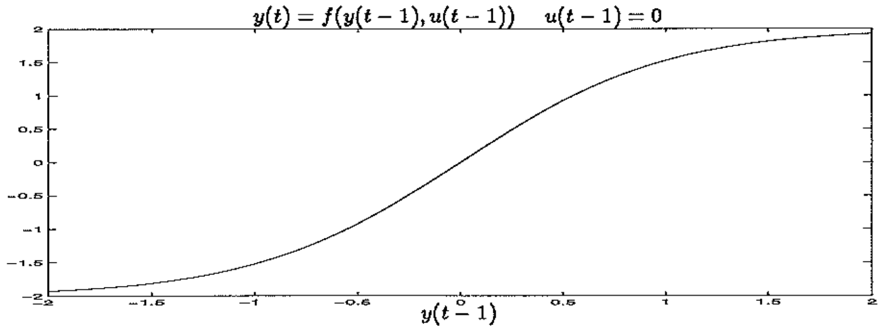


Figure 6.2: Plant dynamics.

Local linearisation of the nonlinear system (6.1) gives that the pole is close to 2 for $y(t-1)$ near zero and goes to 0 for $|y(t-1)| \to \infty$. We identify first order linear ARX models with offsets,

$$y(t) = \frac{b_1 q^{-1}}{1 + a_1 q^{-1}} u(t) + d. \tag{6.2}$$

The identification algorithms are initialized with parameters $\hat{b}_1(0) = 1$, $\hat{a}_1(0) = -0.9$, $\hat{d}(0) = 0$ and $J(0) = 1 \cdot 10^{-5}$, and use a forgetting factor equal to $\lambda = 0.95$. We use online designed two-degrees-of-freedom controllers of second order (see section 2.2). The model reduction algorithm, described in section 2.4, is used to cancel common factors. The controller is implemented with integral action and Nyquist cut-off (see Section 2.7). For simplicity, the controller is specified to give a closed loop system with rise-time $t^r = 10s$. We use an observer rise-time $t^{obs} = 3s$. The damping factor is equal to 1 in both cases. The controller is implemented with anti-windup. The online model validation (see Section 5.3) use 5 sample times memory AIC test. No disturbances are used in the simulations.

The simulations are performed in Matlab/Simulink (See Figure 6.3).

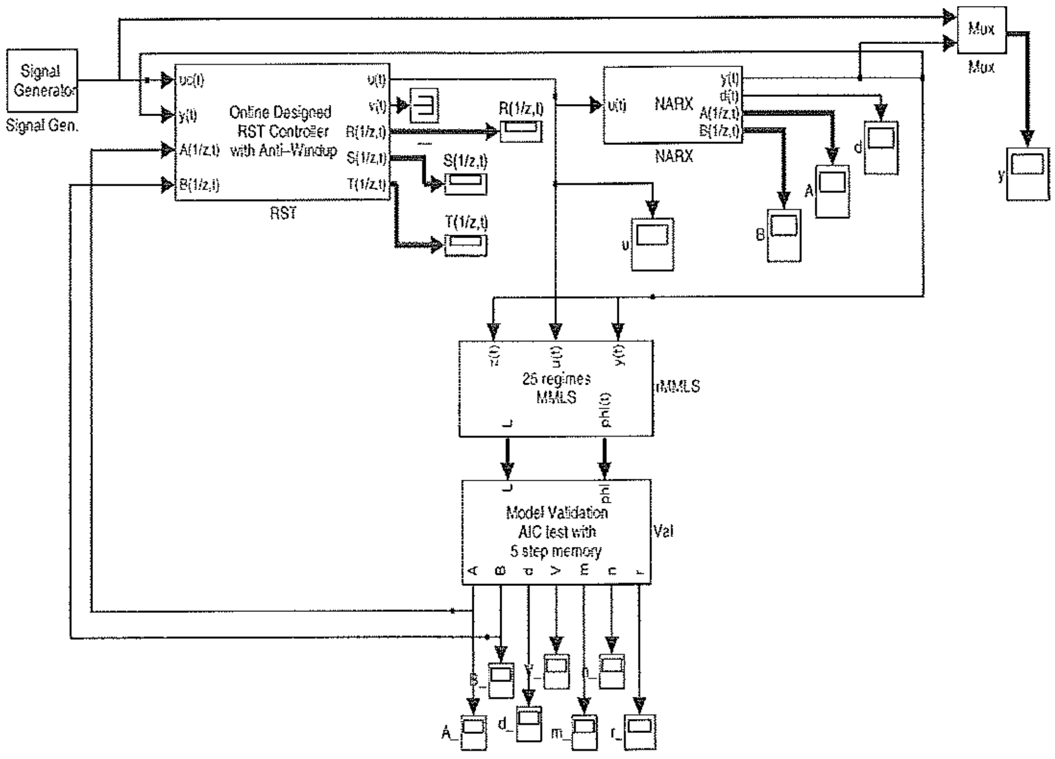


Figure 6.3: Simulink simulation environment.

Conventional Adaptive Control[1]

First, we will attempt to control the plant with conventional adaptive control. The output and control signal are shown in Figure 6.4.

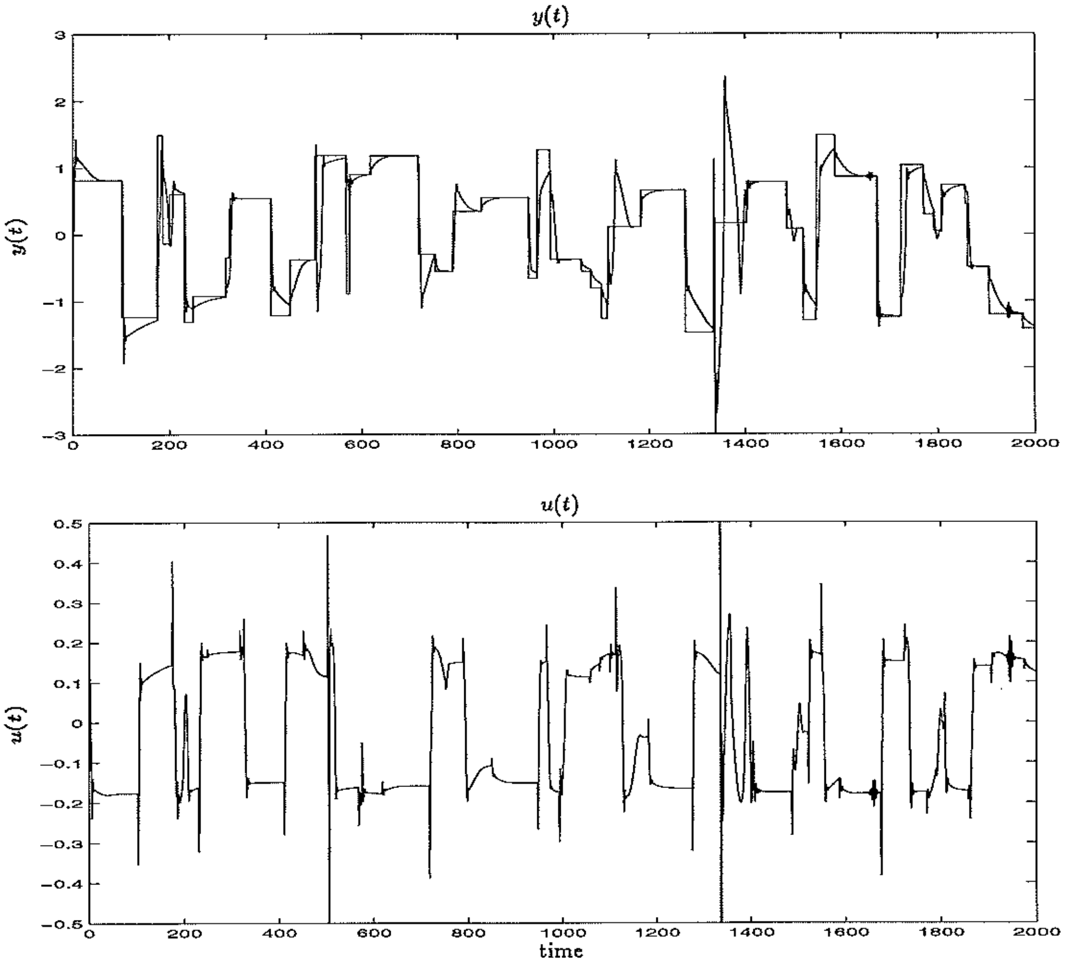


Figure 6.4: Output and control signal.

It can be seen that conventional adaptive control has big difficulties to control the nonlinear and unstable system (6.1). The output has large overshoots when the reference step is large. This is because the plant dynamics are extremely different in different operating points. The model estimates has no chance to follow these fast changes. The steady state error is, however, zero.

[1] Real-time identification of one linear model, and online controller design based on the identified model.

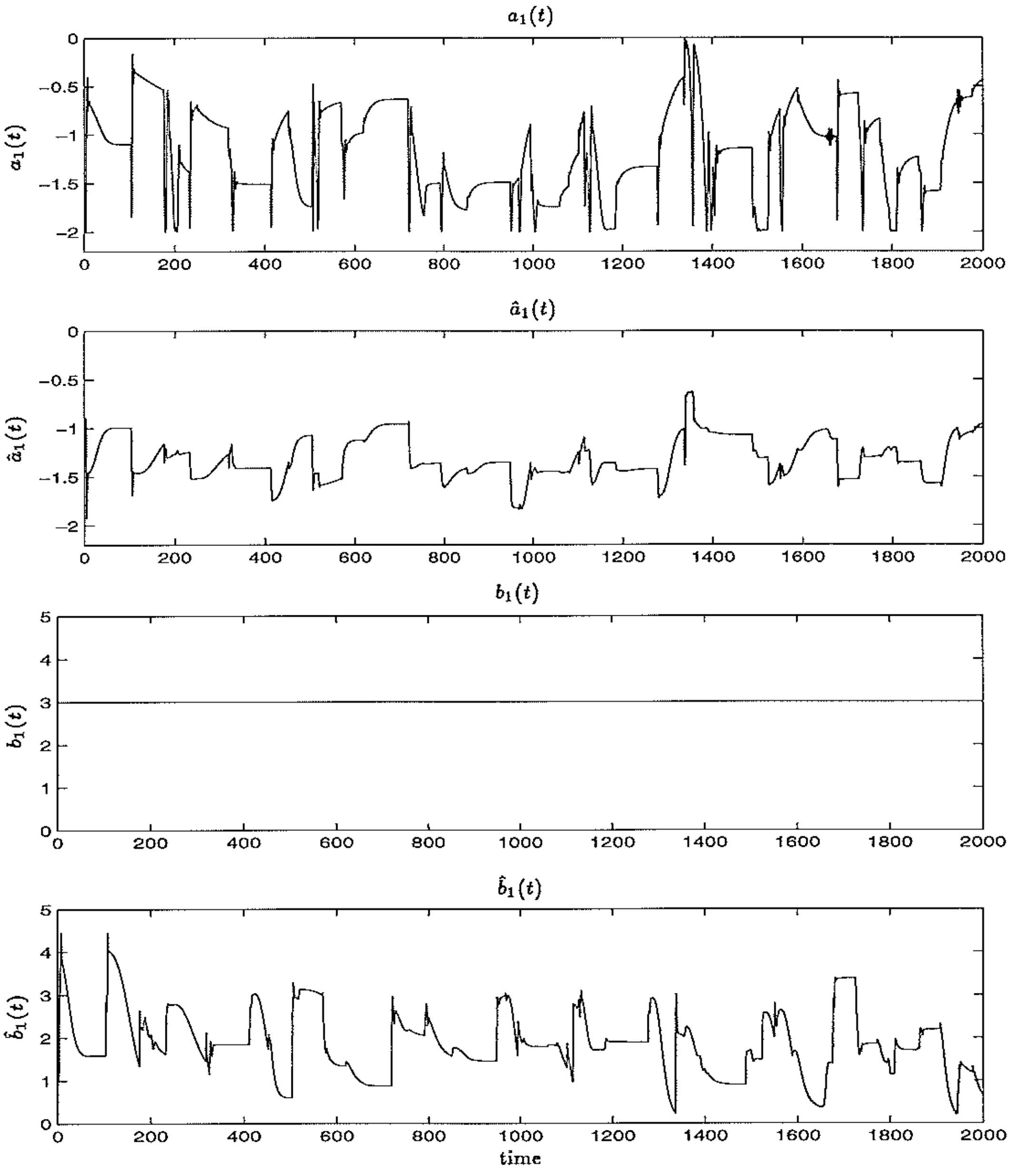


Figure 6.5: True plant parameters, $a_1(t)$, $b_1(t)$, and corresponding estimates, $\hat{a}_1(t)$ and $\hat{b}_1(t)$ (Conventional adaptive control).

It can be seen in Figure 6.3 that the estimates have serious difficulties in following the very fast changes in true parameters. We have not used any noise in this simulation. The noise would have detoriated the parameter tracking capability even more, and by that the control performance.

Adaptive Local Model Networks Strategy

We will now control the same system with the Adaptive Local Model Networks Strategy.
We divide the dynamics into 25 disjoint regimes according to

$$\begin{aligned}
R_1 &= \{-3 \le y(t-1) \le -2.75\} \\
&\vdots \\
R_k &= \{-3 + 0.25(k-1) \le y(t-1) \le -3 + 0.25k\} \\
&\vdots \\
R_{25} &= \{2.75 \le y(t-1) \le 3\}
\end{aligned}$$

In all these regimes we identify models of structure (7.1). The identification algorithms are modified to suspend parameter updating until the plant has been in a specific regime for 2 time-steps (see Section 5.4). The output and control signals are shown in Figure 6.6.

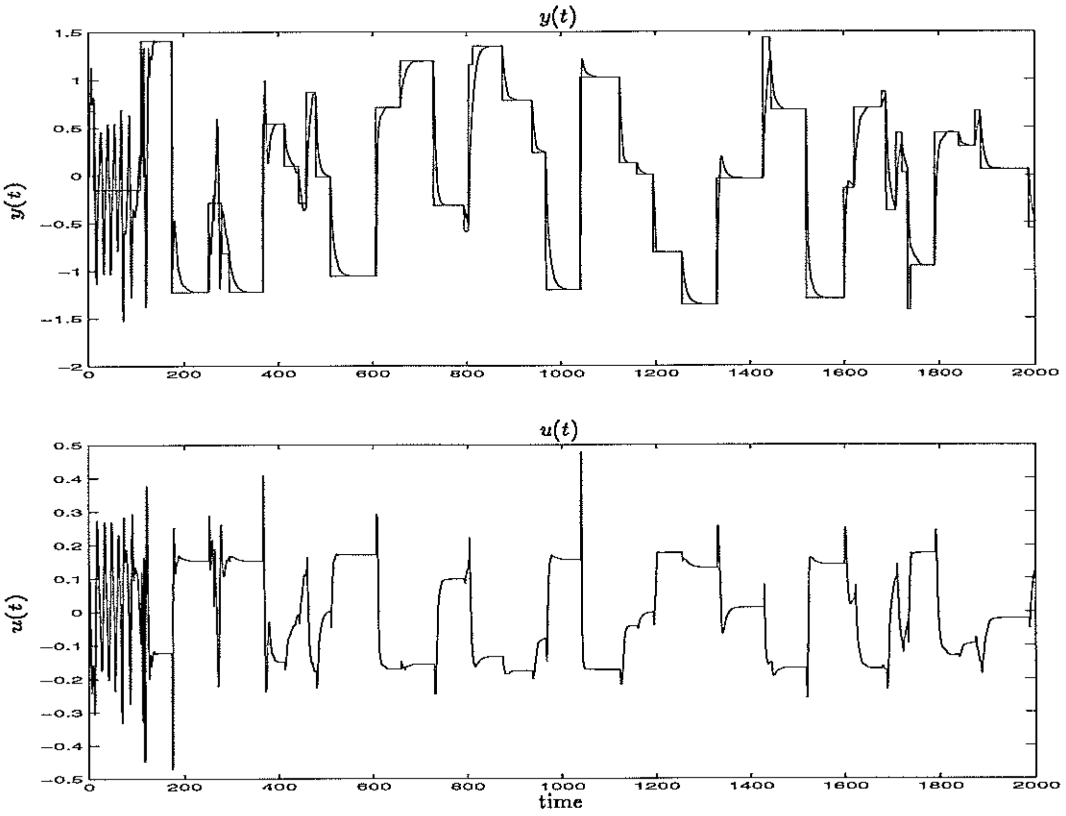


Figure 6.6: Output and control signal (Adaptive Local Model Networks Strategy).

In the beginning of the experiment, the output fluctuations are huge. This is due to many parameters to tune up and badly initialized model estimates. After some time the output follows the reference nearly perfect in the whole operating range. Compared to conventional adaptive control in Figure 6.4, this is a significant improvement. The performance can be expected to improve even more when all model estimates have tuned up.
The reason for this great control performance is that the model in a specific op-

erating point is remembered from the last visit in that regime. The conventional adaptive controller forgets this information.

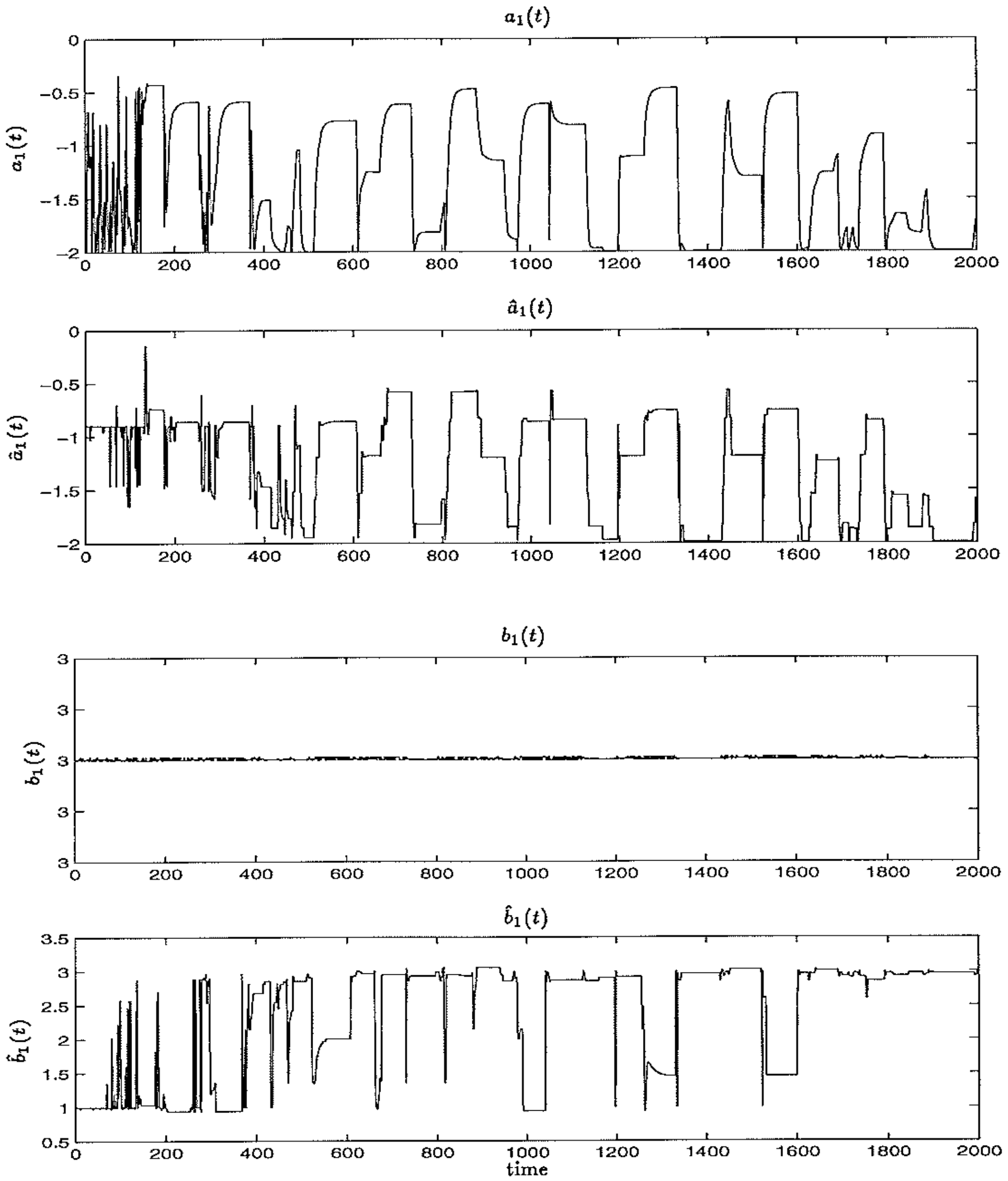


Figure 6.7: True plant parameter, $a_1(t)$, $b_1(t)$, and corresponding estimates, $\hat{a}_1(t)$ and $\hat{b}_1(t)$ (Adaptive local model networks strategy).

In Figure 6.7 it can be seen that it takes some time to achieve good estimates in all regimes. But after this learning time, the estimates are able to follow the very fast parameter changes. Compared to the conventional adaptive control estimates, shown in Figure 6.3, this is a great improvement.

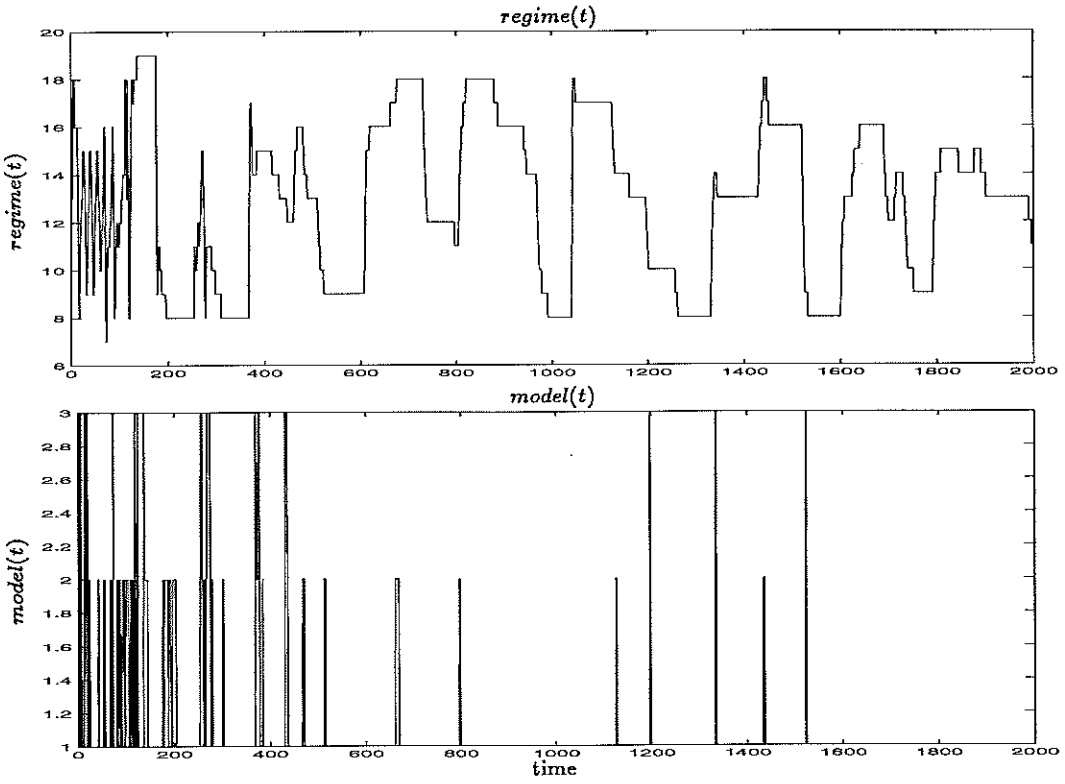


Figure 6.8: Regime of operation and used model type versus time. local model networks : 1, local models : 2, fixed models : 3.

The regime of operation and used model type are plotted in Figure 6.8. The regime of operation plot shows which set of models are updated at different times. It can be seen that some regimes have very few samples, and that some regimes have no samples at all. The model type plot shows that the switching frequency is very high at the beginning. The fixed and local sets of models are used most of the time. After some time, when the entire networks have tuned up, the estimated local model networks are used more and more. The different types of models are complementing each other. The fixed models are securities, used when the estimated models are bad. The local model estimates can adapt fast to changed process dynamics and give acceptable initial performance. The local model networks estimates gives excellent control performance when all parameters have tuned up.

Chapter 7

Adaptive Vehicle Speed Control Simulations

7.1 Introduction

Automatic inter-vehicle distance control systems are currently of great strategic importance in the automotive industry. The specific problem tackled here is the speed control problem, since a distance controller can be built upon a high-precision speed control loop. The vehicle dynamics is nonlinear and time-varying; the dynamics is dependent on the vehicle mass, road incline, rolling resistance, and wind resistance. It might be necessary to adapt the controller to some of these effects to achieve good enough control performance. In this section we have investigated the use of the Adaptive Local Model Networks Controller Strategy for this. The vehicle model used in the simulations were identified using measured data from an experimental vehicle. Fritz 1995, Kalkkuhl *et al* 1997, Hunt *et al* 1995 [4, 12, 7].

7.2 Vehicle Dynamics

A vehicle consists of a number of complex subsystems. The system can however be greatly simplified for the purpose of longitudinal vehicle control.
The system inputs are the throttle position and the brake pressure[1]. The engine produces a torque which is dependent upon the throttle angle and engine speed; this relationship is described by the engine characteristics and is nonlinear. The vehicle nonlinearities can to a large extent be characterized by a number of measurable quantities :

1. vehicle speed
2. throttle angle
3. gear

This will be used to construct reasonable model structures for the local model networks.

[1]In the work described here, the brakes are not used. However, since the throttle and brakes are not simultaneously applied, the throttle-brake combination can be considered as a split-range input signal. The methods described here are therefore immediately applicable when braking is included in the control problem.

7.3 Vehicle Model

The vehicle model used in the simulations were identified using measured data from an experimental vehicle. The model (denoted as M42) is a local model network. A model of the automatic gearbox is also available for the simulations.
The sample time used for system identification and simulation was 480ms.

7.4 Performance Specifications

The vehicle control specification is a trade-off between fast speed tracking response and passenger comfort. A summary is given here:

1. The closed loop system should give suitably fast disturbance rejection.
2. The design should be insensitive to un-modeled dynamics and measurement inaccuracies.
3. The closed loop system should have a pre-specified command response.
4. The closed-loop properties should be consistent over a wide operational envelope despite the system nonlinearities.

7.5 Adaptive Vehicle Speed Control Simulations

One recursive MMLS identification algorithm is employed in each regime to identify first order linear ARX models with offsets,

$$y(t) = \frac{b_1 q^{-1}}{1 + a_1 q^{-1}} u(t) + d \tag{7.1}$$

The MMLS algorithms uses forgetting factor $\lambda = 0.99$, conditional updating value $\epsilon = 1 \cdot 10^{-5}$, initial loss-function $J(0) = 1 \cdot 10^{-7}$, initial model $y(t) = \frac{q^{-1}}{1-0.9q^{-1}} u(t) + 0$. We use online designed two-degrees-of-freedom controllers of second order (see section 2.2). The model reduction algorithm, described in section 2.4, is used to cancel common factors. The controller is implemented with integral action and Nyquist cut-off (see Section 2.7). For simplicity, the controller is specified to give a closed loop system with rise-time[2] $t^r = 10s$. We use an observer rise-time $t^{obs} = 3s$. The damping factor is equal to 1 in both cases. The controller is implemented with anti-windup. The online model validation (see Section 5.3) use a 5 sample times memory AIC test. No disturbances are used in the simulations. The simulation environment is shown in Figure 7.1.

[2]The specification should be faster for low gears.

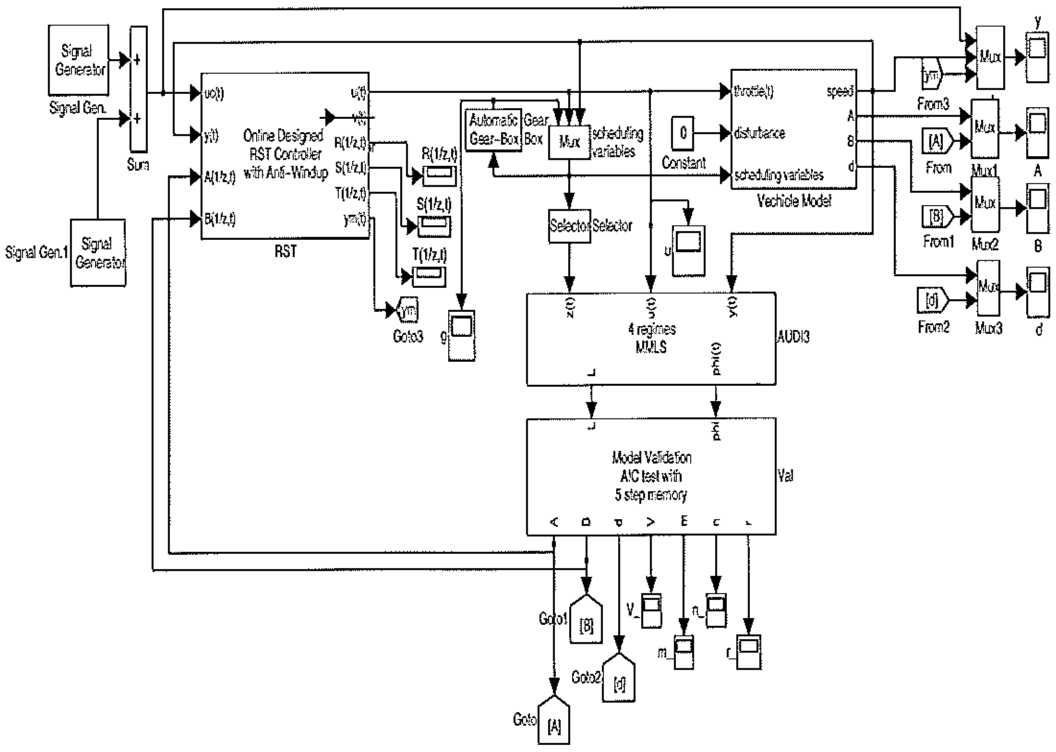


Figure 7.1: Simulink simulation environment.

One Model/Gear

Our first experiment is to only schedule on gear, $g(t) = gear$.

$$\begin{aligned} R_1 &= \{g(t) = 1\} \\ R_2 &= \{g(t) = 2\} \\ R_3 &= \{g(t) = 3\} \\ R_4 &= \{g(t) = 4\} \end{aligned}$$

Speed and parameter tracking response to reference-speed profiles are shown in Figures 7.2-7.10. In Figure 7.2-7.3 the initial performances are plotted. Figure 7.4-7.5 shows the tracking performance when the networks have tuned up. In Figure 7.6-7.9, we have zoomed in the characteristic behaviors for tuned networks. Figure 7.10 shows the response to a large reference-speed step.

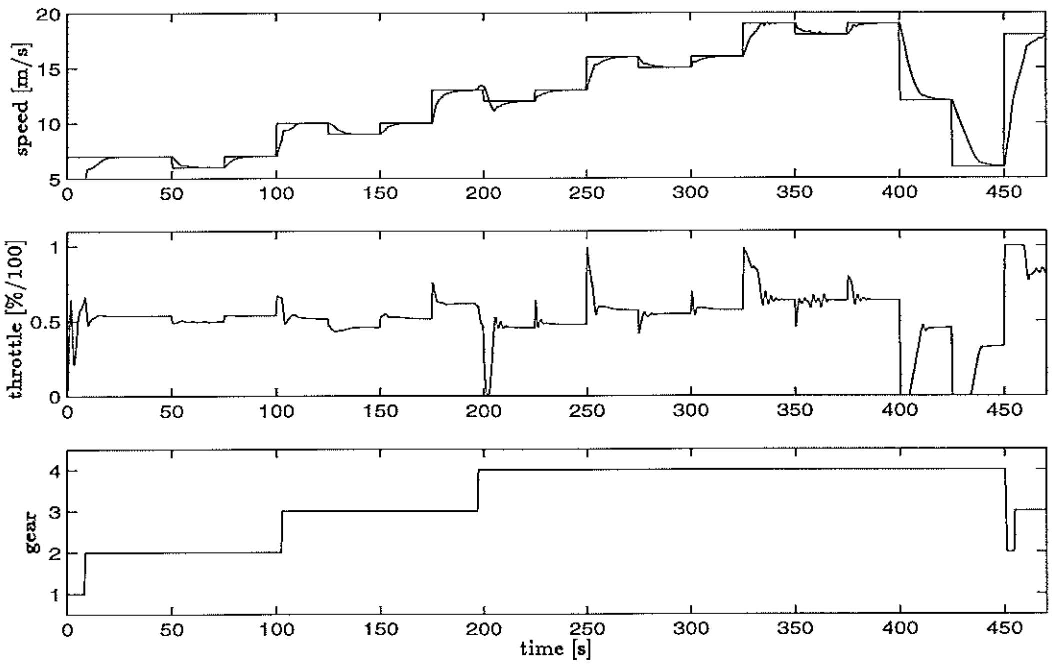


Figure 7.2: Initial Speed tracking response, starting with a bad model.

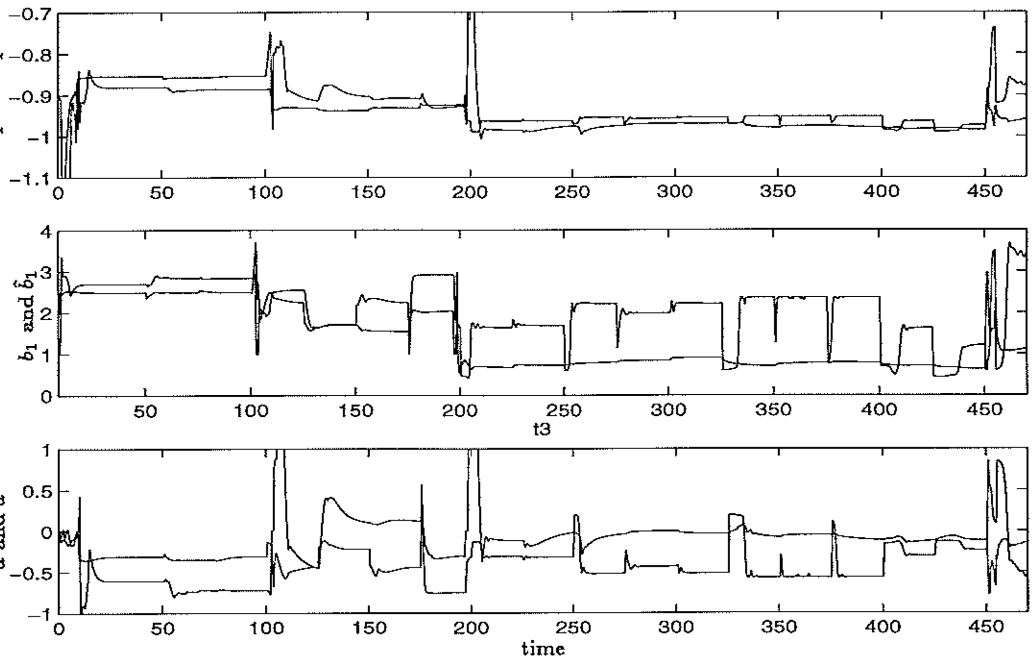


Figure 7.3: Initial parameter tracking response. True and estimated plant parameters versus time.

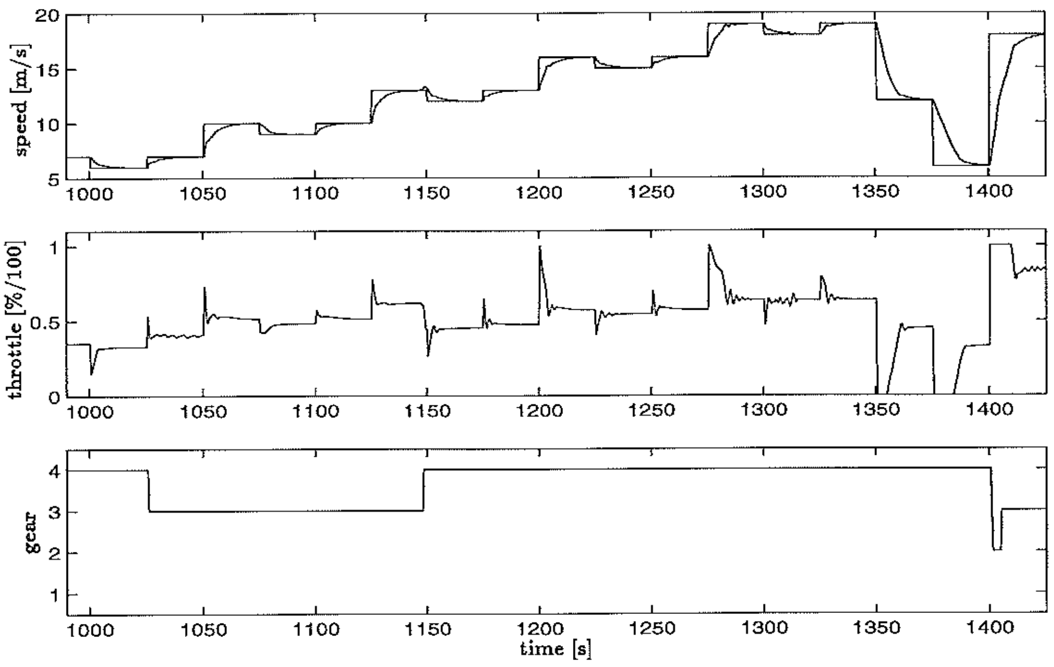


Figure 7.4: Speed tracking response when the models have tuned up.

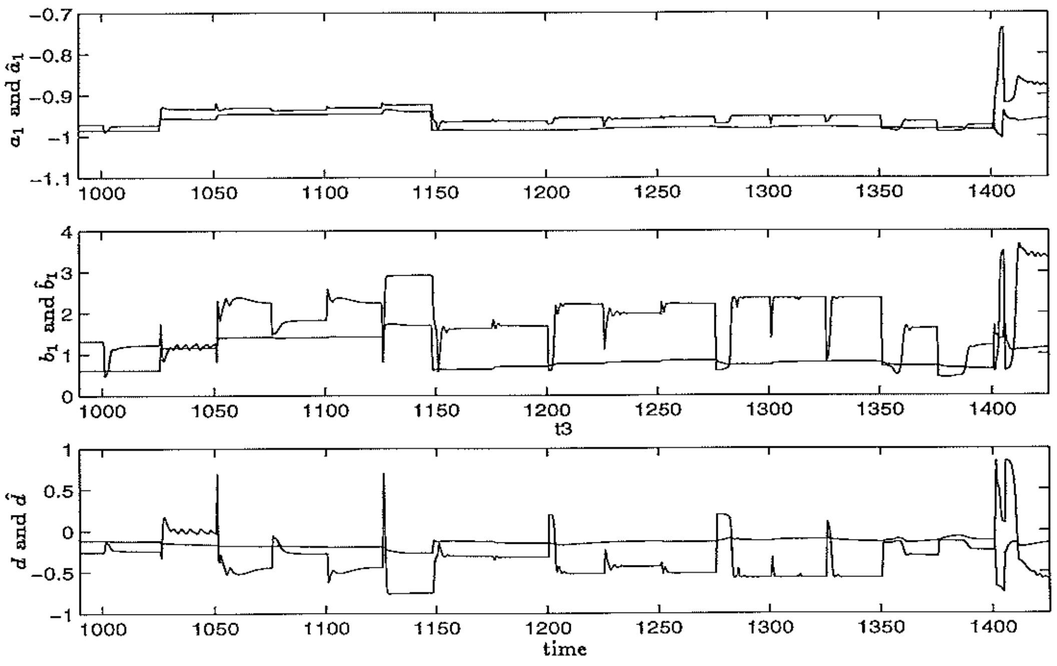


Figure 7.5: True and estimated plant parameters versus time for tuned networks.

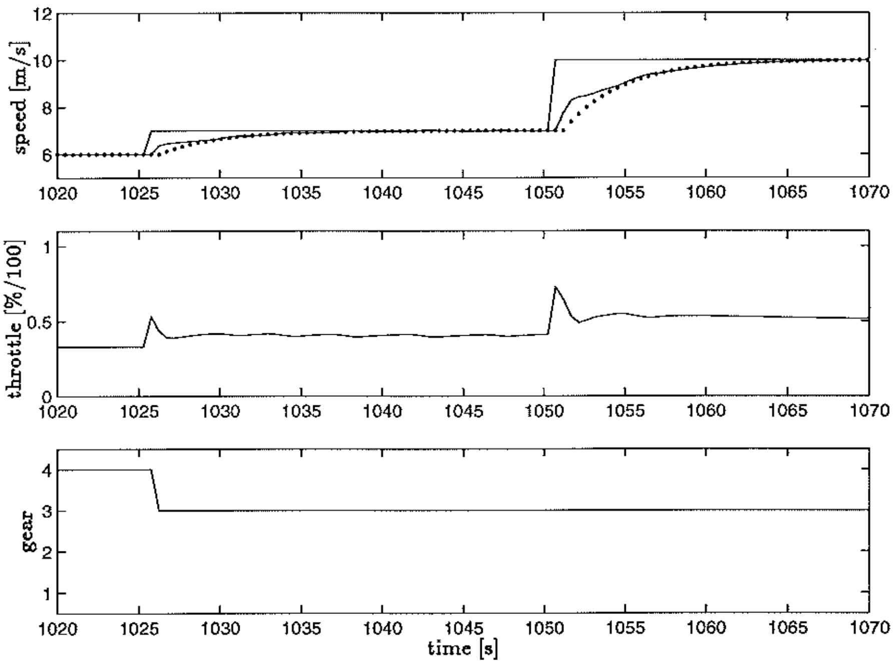


Figure 7.6: Speed tracking test 1 with tuned models. wanted response : dotted, simulated response : solid.

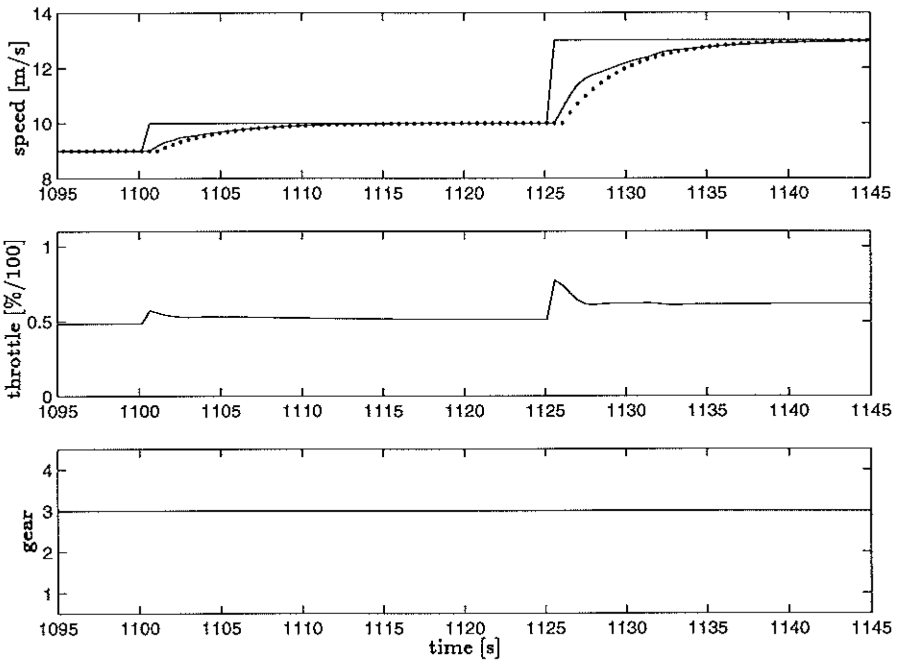


Figure 7.7: Speed tracking test 2. wanted response : dotted, simulated response : solid.

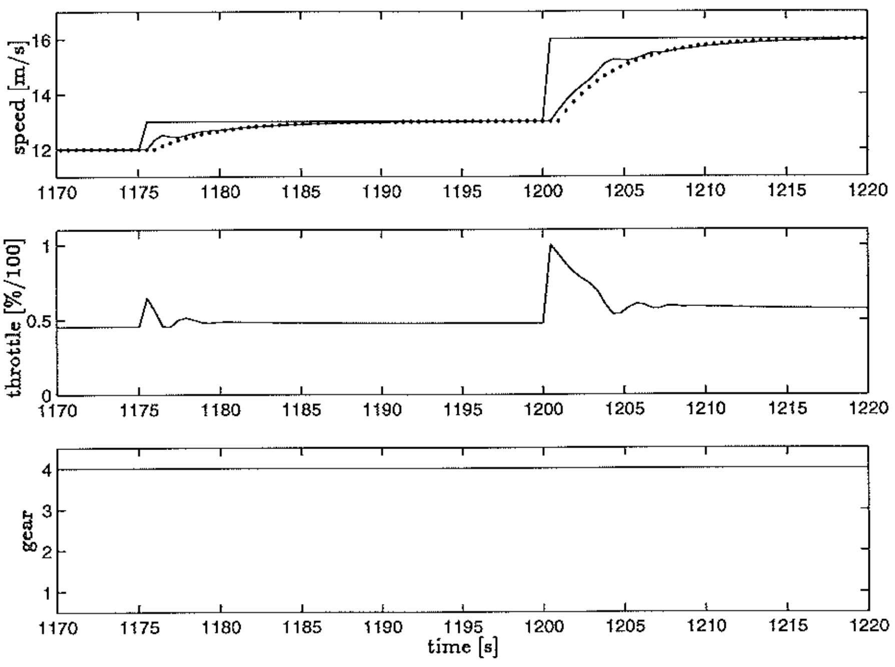


Figure 7.8: Speed tracking test 3. wanted response : dotted, simulated response : solid.

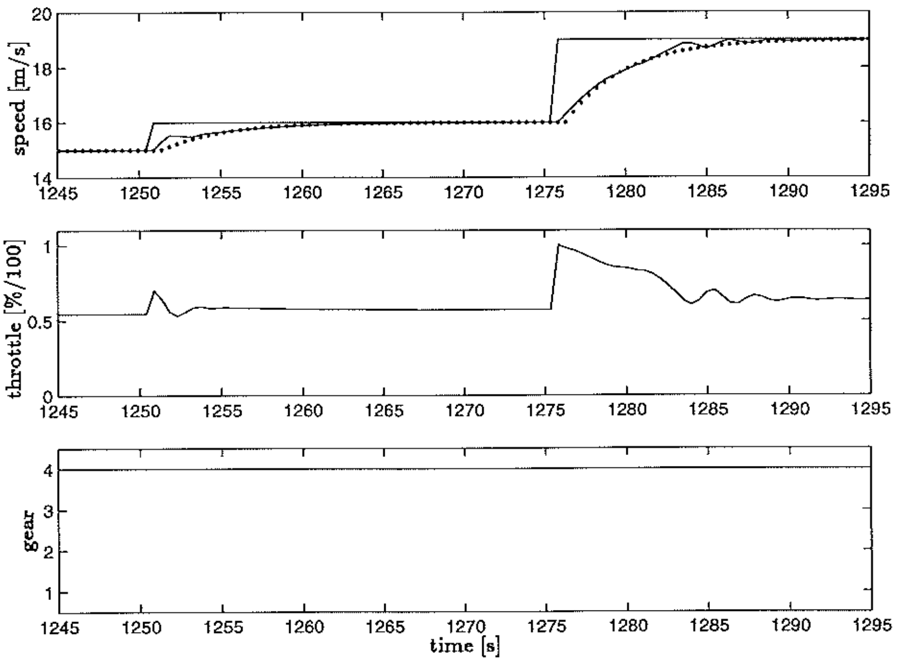


Figure 7.9: Speed tracking test 4. wanted response : dotted, simulated response : solid.

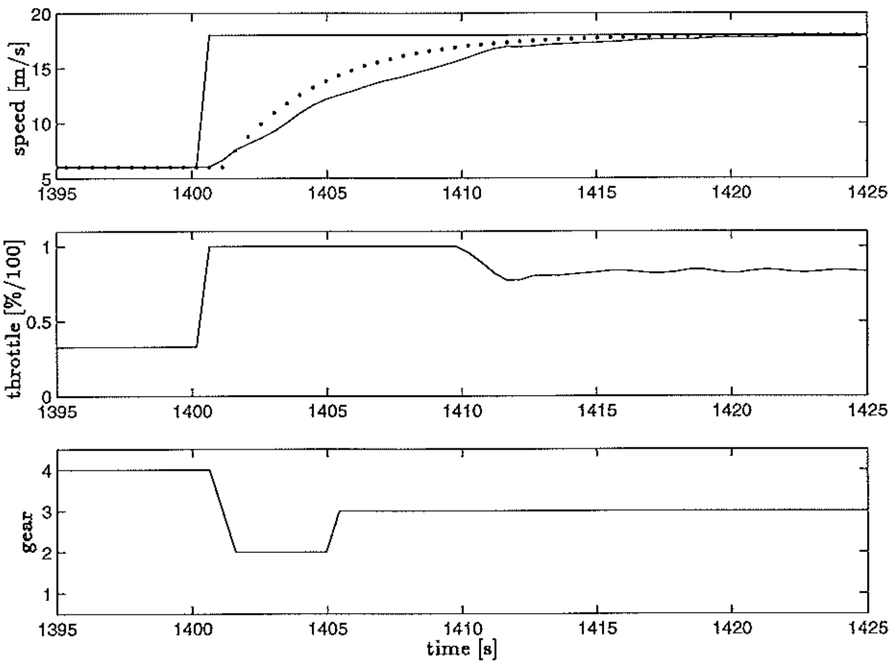


Figure 7.10: Large step speed tracking response; wanted response : dotted; simulated response : solid.

In spite structural model errors (see Figure 7.5), the speed tracking response is close to the specified most of the time, i.e. 10*s* rise-time and no overshot or static errors. Some minor irregularities, due to parameter variations within the regimes or gear shifts, can be seen in the plots. The 10*s* rise-time is a bit to fast for the 4'th gear, while a faster response could be used for the lower gears. When using a faster specification, the structural miss-matches become more obvious.

Nine Models/Gear

In the previous section, we saw the one model/gear was not enough if we want fast, high performance control. We will here investigate the use of nine models/gear, where we schedule on throttle and speed according to Figure 7.11, $\tilde{\phi} = \begin{pmatrix} throttle \\ speed \\ gear \end{pmatrix}$. All together we have 36 regimes.

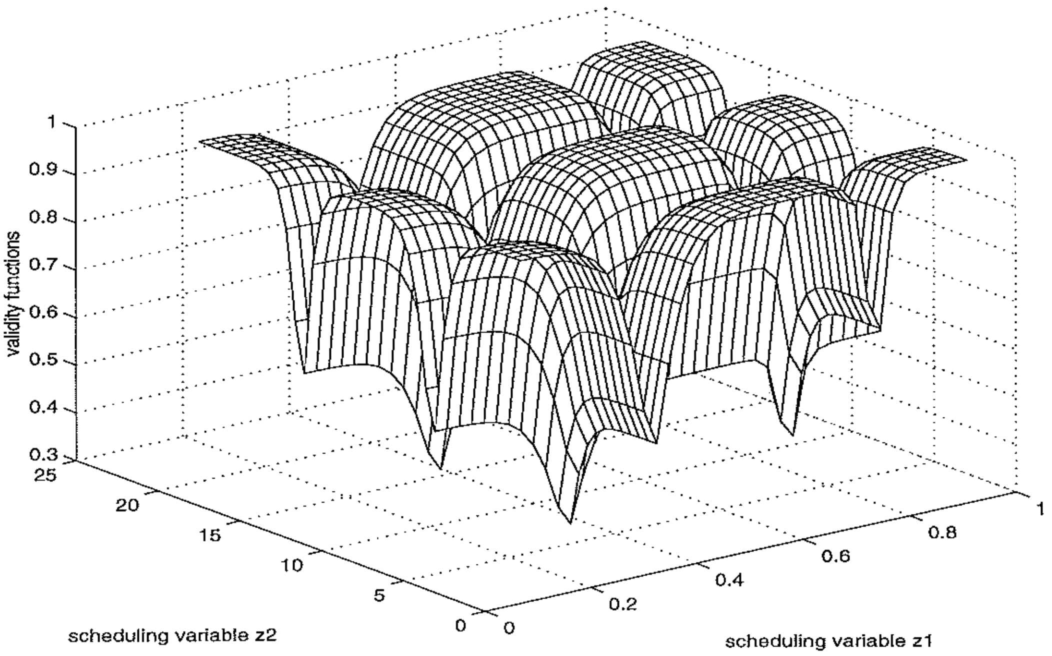


Figure 7.11: Validity functions. Scheduling variable z_1 is throttle, and z_2 is speed.

Speed and parameter tracking response to reference-speed profiles are shown in Figures 7.12-7.20. In Figure 7.12-7.13 the initial performances are plotted. Figure 7.14-7.15 shows the tracking performance when the networks have tuned up. In Figure 7.16-7.19, we have zoomed in the characteristic behaviors for tuned networks. Figure 7.20 shows the response to a large reference-speed step.

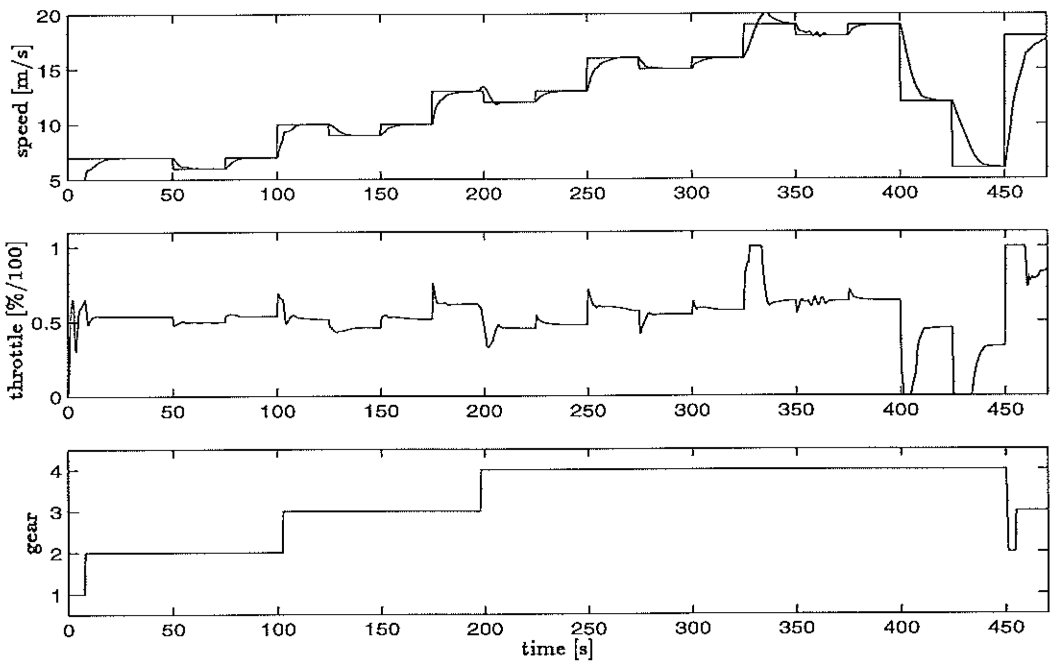


Figure 7.12: Initial Speed tracking response, starting with a bad model.

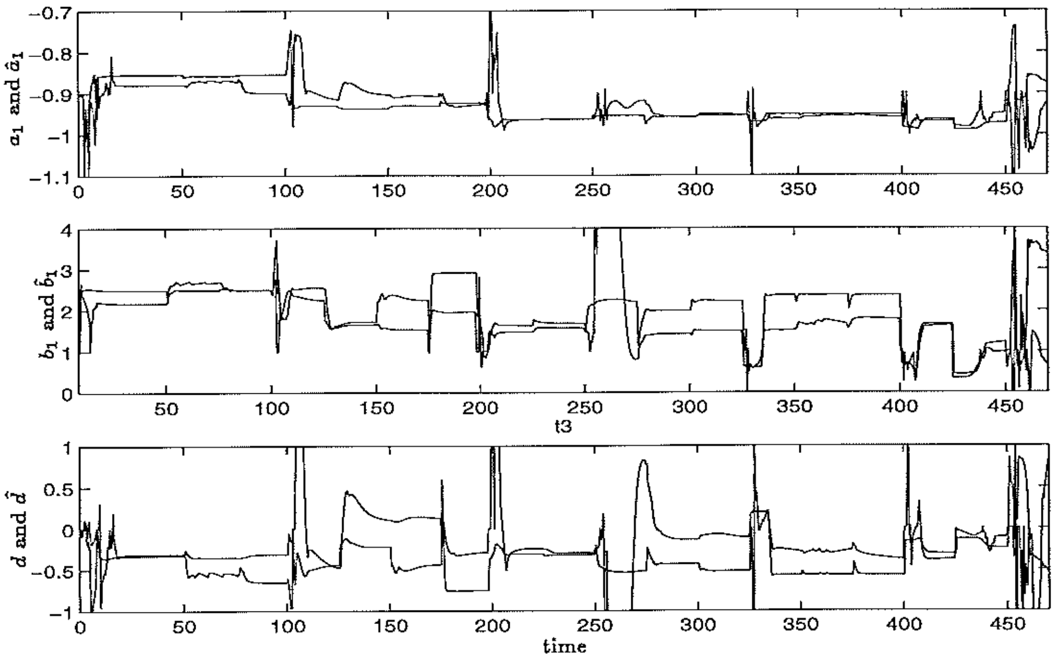


Figure 7.13: Initial parameter tracking response. True and estimated plant parameters versus time.

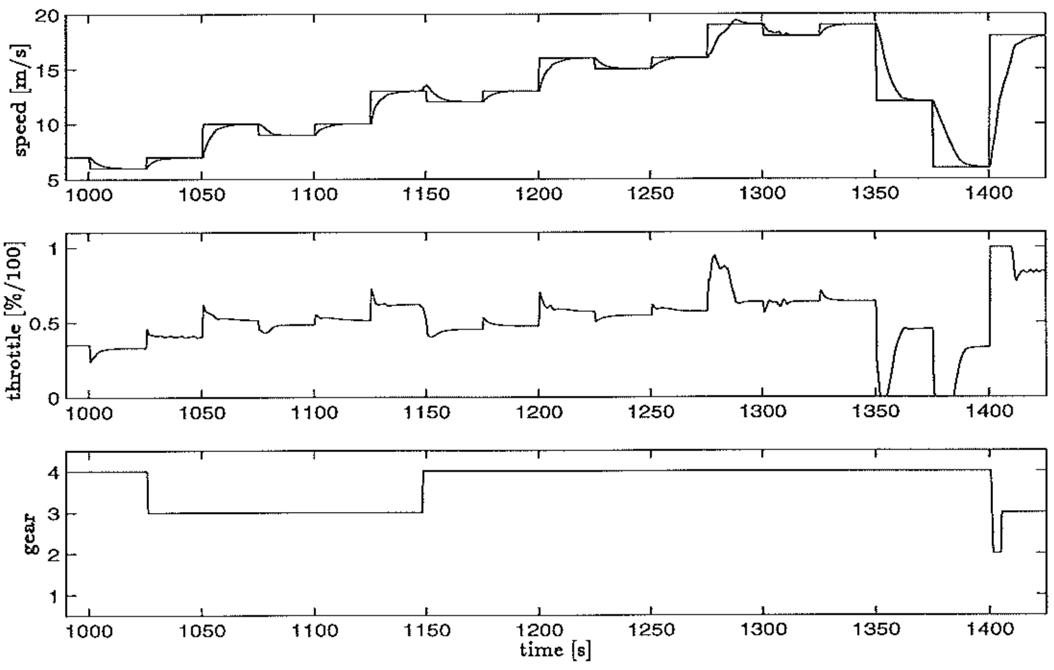


Figure 7.14: Speed tracking response when the models have tuned up.

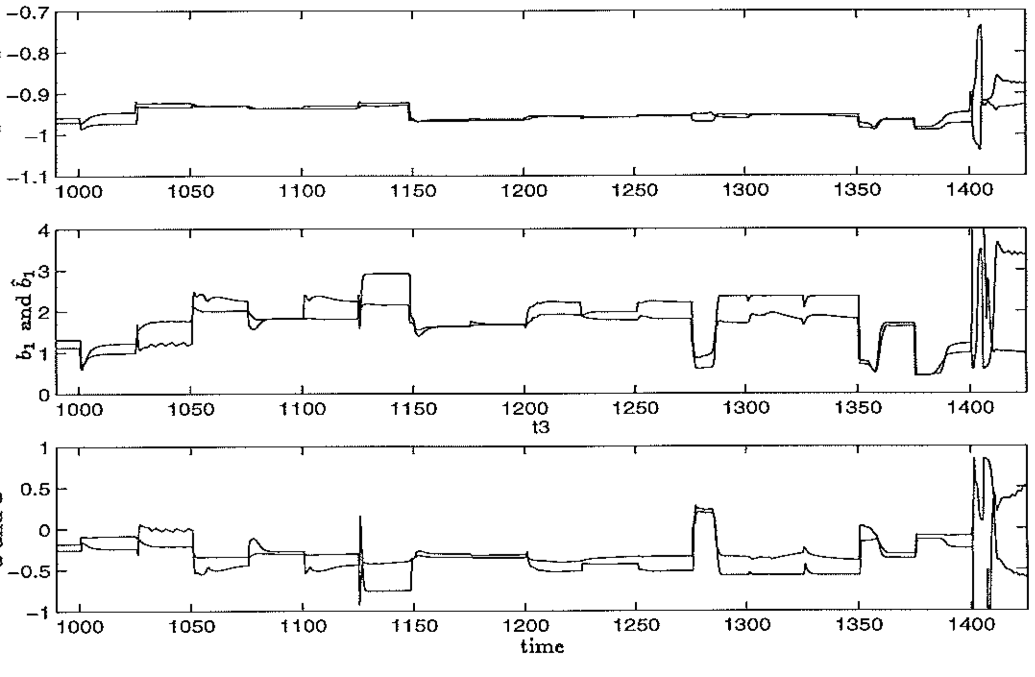


Figure 7.15: True and estimated plant parameters versus time.

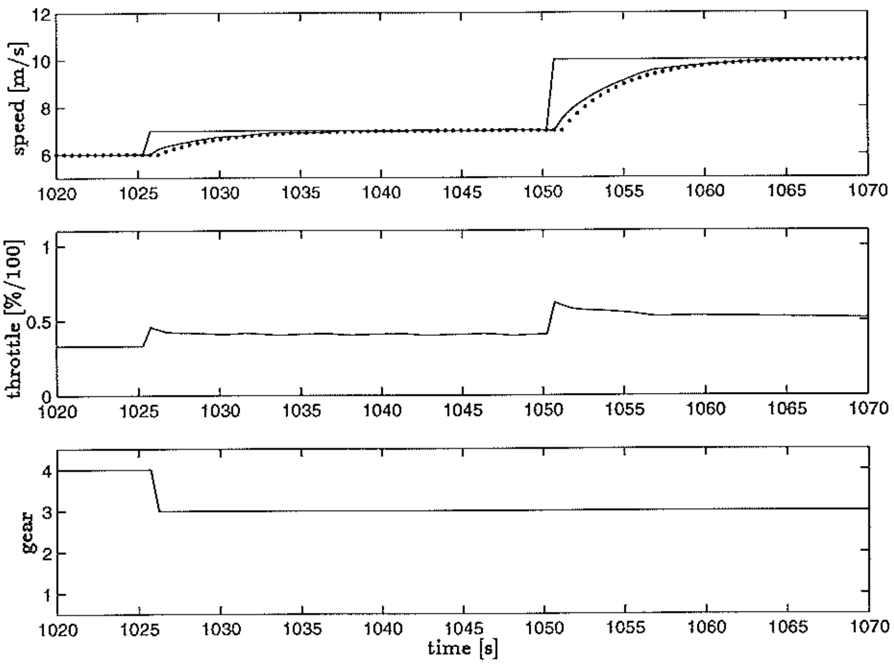


Figure 7.16: Speed tracking test 1 with tuned models. wanted response : dotted, simulated response : solid.

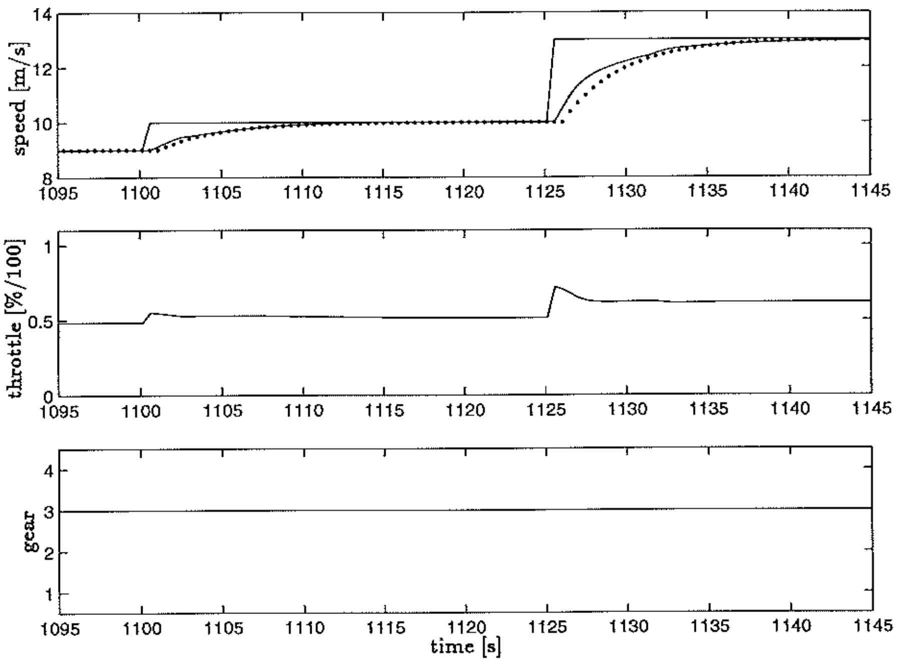


Figure 7.17: Speed tracking test 2. wanted response : dotted, simulated response : solid.

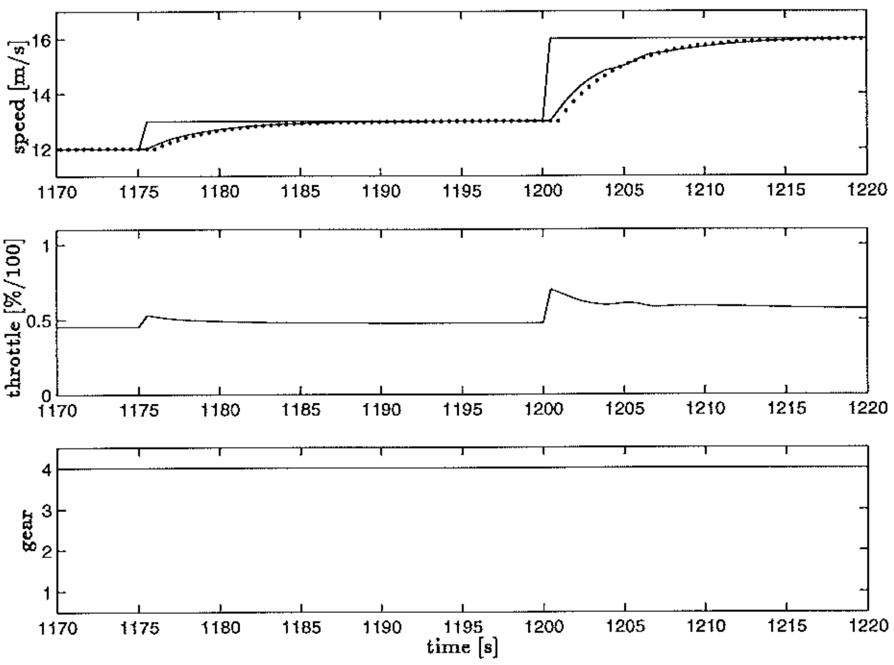


Figure 7.18: Speed tracking test 3. wanted response : dotted, simulated response : solid.

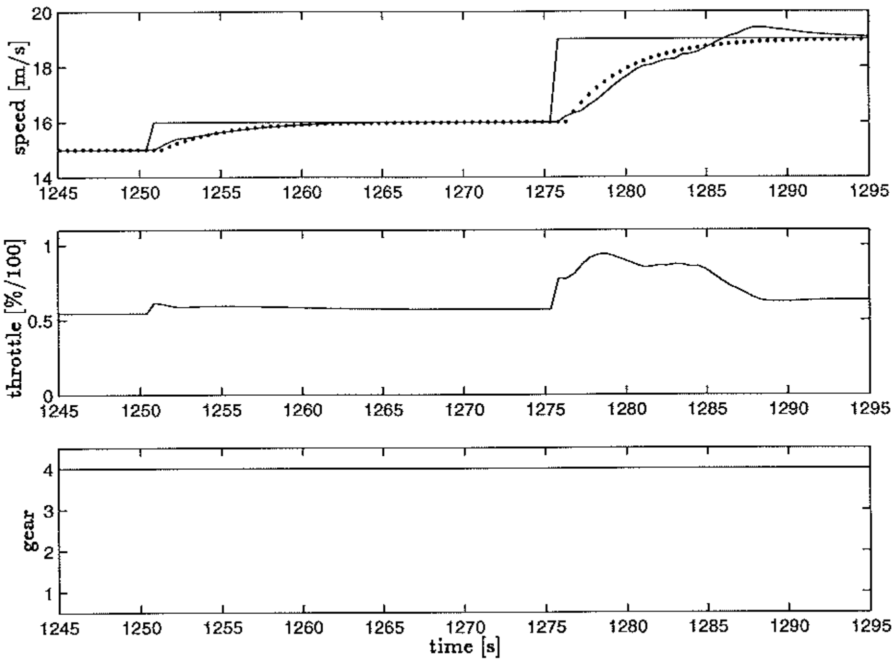


Figure 7.19: Speed tracking test 4. wanted response : dotted, simulated response : solid.

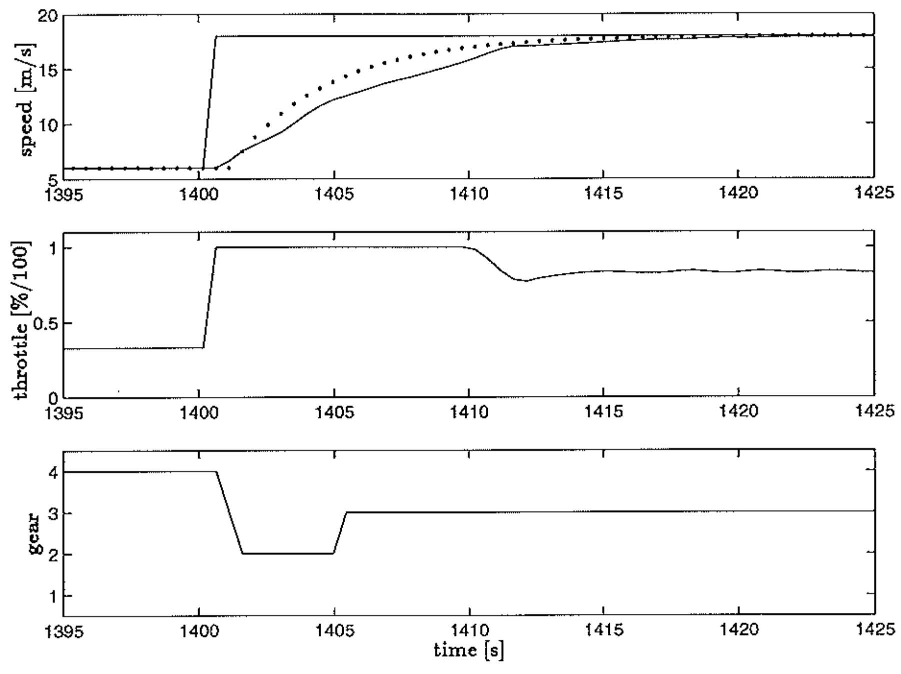


Figure 7.20: Large step speed tracking response. wanted response : dotted, simulated response : solid.

It can be seen that it takes some time to tune up all models. After this initial learning time, the speed-tracking rise-time is approximately the same in the entire space of operation. A comparison of Figure 7.15 with Figure 7.5 shows that the parameter-estimates are much closer to true parameters in the nine models/gear setting than in the one model/gear setting. Control with the nine models/gear overcomes many of the irregularities in the one-model/gear control response. The response is smooth and nearly exactly the same as the specified response in all operating points. The model mismatches are, as expected, much less here than in the one model/gear case (compare Figure 7.5 with Figure 7.15). But since the local model networks are build up by more local models now, it takes longer time to tune up the nine models/gear networks.

7.6 Conclusions

We have investigated the use of the adaptive local model networks strategy to the high performance vehicle speed control problem. The purpose of the adaptive strategy is to adapt the controller to changed dynamics (change of vehicle mass, uphill/downhill, wind-resistance or road incline). The one model per gear networks gives acceptable response when the controller specification is not too fast. If faster speed tracking responses are wanted, more models are needed. Nine models per gear, scheduled by speed and throttle position, gives significantly improved response for fast tracking specifications. The irregularities in the one model per gear response are reduced in the nine models per gear response. It takes, however, much longer time to tune up all models in the nine models per gear networks than in the one model per gear networks. Hence, we should use the minimum amount of models that gives the system an acceptable response at a specified speed tracking response.

Regimes with few data samples could use fixed models exclusively; no adaptive models.

Appendix A

Proof/Derivation of a Model Reduction Algorithm based on the Sylvester Matrix

The basic idea of the algorithm is to convert the problem into a standard system of linear equations, and then investigate the column space of the corresponding matrices. The following ideas from linear algebra are useful.

Definition A.1 (Column Space)
The Column space (alt. span or vector-space) of $\mathbf{A}$ *is defined by*
$\mathcal{R}(\mathbf{A}) \overset{\text{def}}{=} \{y \in I\!R^m \ : \ y = \mathbf{A}x \ \ \textit{for some} \ \ x \in I\!R^n\}$
$\mathcal{R}(\mathbf{A})$ *is the space that can be reached by any combination of the columns in* $\mathbf{A}$
□

Definition A.2 (dimension of column-space)
The dimension of the column space of A is written as
$dim(\mathcal{R}(\mathbf{A}))$
□

Definition A.3 (full rank)
A matrix $\mathbf{A} \in \mathbf{R}^{m \times n}$*,* $m \geq n$*, is said to have* **full rank** *if and only if* $dim(\mathcal{R}(\mathbf{A})) = n$
□

Consider the multiplication of two polynomials $A(q^{-1})$ and $X(q^{-1})$, where $X(q^{-1})$ is monic,

$$A(q^{-1})X(q^{-1}) \tag{A.1}$$

This product can be written as

$$A(q^{-1}) + q^{-1}A(q^{-1})x_1 + q^{-2}A(q^{-1})x_2 + \cdots + q^{-n}A(q^{-1})x_n \tag{A.2}$$

or with matrix notation

$$\begin{pmatrix} A(q^{-1}) & q^{-1}A(q^{-1}) & q^{-2}A(q^{-1}) & \cdots & q^{-n}A(q^{-1}) \end{pmatrix}^T \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{A.3}$$

Now think about the polynomials as vectors, where the power of q^{-1} determines the position in the vector.

Suppose the polynomials A and B are given by $A(q^{-1})$[1] $= 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$, $B(q^{-1}) = 0 + b_1 q^{-1} + \cdots + b_n q^{-n}$. We are interested in which modes that can be reached by multiplying these polynomials with two other polynomials X and Y, given by $X(q^{-1}) = 1 + x_1 q^{-1} + \cdots + x_n q^{-n}$ and $Y(q^{-1}) = y_0 + y_1 q^{-1} + \cdots + y_{n-1} q^{-(n-1)}$. These polynomial-multiplications can be written as matrix-multiplications with matrices

$$\mathbf{A} = \left(\begin{array}{ccccc} 1 & 0 & \cdots & 0 & 0 \\ a_1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & \cdots & 1 & 0 \\ a_{n-1} & a_{n-2} & \cdots & a_1 & 1 \\ a_n & a_{n-1} & \cdots & a_2 & a_1 \\ 0 & a_n & \cdots & a_3 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_n & a_{n-1} \\ 0 & 0 & \cdots & 0 & a_n \end{array}\right) \quad \begin{array}{|c} q^{-1} \\ q^{-2} \\ \vdots \\ q^{-(n-1)} \\ q^{-n} \\ q^{-(n+1)} \\ q^{-(n+2)} \\ \vdots \\ q^{-(2n-1)} \\ q^{-2n} \end{array} \tag{A.4}$$

and

$$\mathbf{B} = \left(\begin{array}{ccccc} b_1 & 0 & \cdots & 0 & 0 \\ b_2 & b_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{n-1} & b_{n-2} & \cdots & b_1 & 0 \\ b_n & b_{n-1} & \cdots & b_2 & b_1 \\ 0 & b_n & \cdots & b_3 & b_2 \\ 0 & 0 & \cdots & b_4 & b_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & b_n \\ 0 & 0 & \cdots & 0 & 0 \end{array}\right) \quad \begin{array}{|c} q^{-1} \\ q^{-2} \\ \vdots \\ q^{-(n-1)} \\ q^{-n} \\ q^{-(n+1)} \\ q^{-(n+2)} \\ \vdots \\ q^{-(2n-1)} \\ q^{-2n} \end{array} \tag{A.5}$$

The rows in the matrices correspond to a specific polynomial power in the product. The first and the last row for example corresponds to q^{-1} respectively q^{-2n}. The last row in $\mathbf{B}$ is all zeros since the highest power in $B(q^{-1})Y(q^{-1})$ is $q^{-(2n-1)}$ and not q^{-2n}. The matrix-ranks are in both cases equal to n (if $B(q^{-1}) \not\equiv 0$). That is the there are n independent columns in the matrices, $\mathcal{R}(\mathbf{A}) = \mathbf{A}$ and $\mathcal{R}(\mathbf{B}) = \mathbf{B}$. But what is $dim(\mathcal{R}(\mathbf{A}) \bigcup \mathcal{R}(\mathbf{B}))$? This corresponds to the reachable space when the columns of both $\mathbf{A}$ and $\mathbf{B}$ are used.

A common factor between the polynomials is equivalent to a common column space in $\mathcal{R}(\mathbf{A})$ and $\mathcal{R}(\mathbf{B})$. A common space gives only rise to one column in $\mathcal{R}(\mathbf{A}) \bigcup \mathcal{R}(\mathbf{B})$. Thus we have the following:

Theorem A.1 (Dimension of Joint Space)
Two polynomial of order n given by

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_n q^{-n} \\ B(q^{-1}) &= 0 + b_1 q^{-1} + \cdots + b_n q^{-n}, \end{aligned}$$

[1] The polynomials are here given in backward decomposition to be compatible with the control design. It could equally well be given in forward decomposition. The operator is just seen as an indeterminate.

having a common factor of order k,

$$K(q^{-1}) = 1 + k_1 q^{-1} + \cdots + k_k q^{-k},$$

have dimension of joint space equal to

$$dim(\mathcal{R}(\mathbf{A}) \bigcup \mathcal{R}(\mathbf{B})) = 2n - k.$$

Proof.
$dim(\mathcal{R}(\mathbf{A}) \bigcup \mathcal{R}(\mathbf{B})) = dim(\mathcal{R}(\mathbf{A})) + dim(\mathcal{R}(\mathbf{B})) - dim(\mathcal{R}(\mathbf{A}) \bigcap \mathcal{R}(\mathbf{B})) = n + n - k$
□

Our goal is to find the common column space, **K**. We know that this is a part of

$$\mathcal{R}(\mathbf{A}) \bigcup \mathcal{R}(\mathbf{B})$$

The problem is 'just' to sort it out.
The following theorem gives a way this can be done.

Theorem A.2 (Space-Invariant to Matrix Multiplication to the right)
Right-multiplication of a full rank matrix does not change the column space. That is

$$\mathcal{R}(\mathbf{AB}) = \mathcal{R}(\mathbf{A})$$

where **B** *has full rank.*

Proof.
Matrix multiplication to the right is equivalent to a change of basis. Since the right matrix has full rank, all columns present in **A** *must also be present in* **AB**.
□

From Theorem A.1 and A.2 we know that the union of **A** and **B** has space dimension $2n - k$ and that a matrix multiplication to the right does not change the column space. As mentioned before, the goal is to find the common space. We will present a way to do this by basis transformations. We want to find a basis where the column space is obvious. That is, to find a full rank matrix that transforms the space to a basis where the k rightmost columns are equal to zero. The following theorem solves this problem.

Theorem A.3 (Factorisation)
The matrix

$$[\ \mathbf{A} \quad \mathbf{B}\ ] = \left(\begin{array}{ccccc|ccccc} 1 & 0 & \cdots & 0 & 0 & b_1 & 0 & \cdots & 0 & 0 \\ a_1 & 1 & \cdots & 0 & 0 & b_2 & b_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & \cdots & 1 & 0 & b_{n-1} & b_{n-2} & \cdots & b_1 & 0 \\ a_{n-1} & a_{n-2} & \cdots & a_1 & 1 & b_n & b_{n-1} & \cdots & b_2 & b_1 \\ \hline a_n & a_{n-1} & \cdots & a_2 & a_1 & 0 & b_n & \cdots & b_3 & b_2 \\ 0 & a_n & \cdots & a_3 & a_2 & 0 & 0 & \cdots & b_4 & b_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_n & a_{n-1} & 0 & 0 & \cdots & 0 & b_n \\ 0 & 0 & \cdots & 0 & a_n & 0 & 0 & \cdots & 0 & 0 \end{array}\right) = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix} \tag{A.6}$$

can be factorised as

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & 0 \\ \mathbf{R}_{21} & \mathbf{U}^T \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L}_{11}^{-1}\mathbf{L}_{12} \\ \mathbf{0} & \mathbf{L}^T\mathbf{P} \end{pmatrix} \tag{A.7}$$

where $\mathbf{L}$*,* $\mathbf{U}$ *and* $\mathbf{P}$ *comes from a LU-factorisation of* $(\mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12})^T$*.* $\mathbf{U}^T$ *have rank equal to* $r = n - k$*, where* n *is the dimension of the matrix and* k *the dimension of the common space.* $\mathbf{U}^T$ *is given by*

$$\mathbf{U}^T = \left(\begin{array}{ccccc|ccc} u_{1,1} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ u_{2,1} & u_{2,2} & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{r-1,1} & u_{r-1,2} & \cdots & u_{r-1,r-1} & 0 & 0 & \cdots & 0 \\ u_{r,1} & u_{r,2} & \cdots & u_{r,r-1} & \mathbf{u_{r,r}} & 0 & \cdots & 0 \\ \hline u_{r+1,1} & u_{r+1,2} & \cdots & u_{r+1,r-1} & \mathbf{u_{r+1,r}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{n,1} & u_{n,2} & \cdots & u_{n,r-1} & \mathbf{u_{n,r}} & 0 & \cdots & 0 \end{array}\right) \tag{A.8}$$

Proof.
We will prove this in two steps.

1. *Find a full rank coordinate-transformation that makes the upper-right submatrix zero while maintaining the left submatrices.*
2. *Find a full rank coordinate-transformation that makes the* k *rightmost columns in the lower-right submatrix produced in step 1 zero.*

Step 1

Find a full rank transformation-matrix X *that solves the problem*

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{X}_{21} & \mathbf{X}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & 0 \\ \mathbf{R}_{21} & \mathbf{Y} \end{pmatrix} \tag{A.9}$$

One solution to this problem is $\mathbf{X}_{11} = \mathbf{X}_{22} = I$*,* $\mathbf{X}_{21} = 0$ *and* $\mathbf{X}_{12} = -\mathbf{L}_{11}^{-1}\mathbf{L}_{12}$*.* $\mathbf{L}_{11}$ *is always invertible.* $\mathbf{Y}$ *will then become* $\mathbf{Y} = \mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12}$*. The inverse of* $\mathbf{X}$ *is easily seen to be given by*

$$\mathbf{X}^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{L}_{11}^{-1}\mathbf{L}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{A.10}$$

This means that the Sylvester-matrix can be factorised as

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & 0 \\ \mathbf{R}_{21} & \mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L}_{11}^{-1}\mathbf{L}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{A.11}$$

Step 2

Find a full rank matrix that makes the k *rightmost columns to zeros. This problem can be solved by an LU factorisation of the submatrix* $(\mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12})^T$*. It means that we make a factorisation such that* $\mathbf{LU} = \mathbf{P}(\mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12})^T$ *, where* $\mathbf{L}$ *is a unitary lower left matrix,* $\mathbf{U}$ *is a upper right matrix and* $\mathbf{P}$ *is a permutation matrix. By transposing this expression and multiply by* $\mathbf{P}$ *to the right we have that* $\mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12} = \mathbf{U}^T\mathbf{L}^T\mathbf{P}$*. The matrix*

$$\begin{pmatrix} \mathbf{L}_{11} & 0 \\ \mathbf{R}_{21} & \mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12} \end{pmatrix} \tag{A.12}$$

can thus be factorised as

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{R}_{21} & \mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{L}_{11}^{-1}\mathbf{L}_{12} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{R}_{21} & \mathbf{U}^T \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^T\mathbf{P} \end{pmatrix} \tag{A.13}$$

Combine these two transformations to

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{R}_{21} & \mathbf{U}^T \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^T\mathbf{P} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L}_{11}^{-1}\mathbf{L}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{A.14}$$

$$= \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{R}_{21} & \mathbf{U}^T \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L}_{11}^{-1}\mathbf{L}_{12} \\ \mathbf{0} & \mathbf{L}^T\mathbf{P} \end{pmatrix} \tag{A.15}$$

Remark

The QR decomposition could have been used instead of the LU decomposition.

□

We now apply Theorem (A.3) on the matrix

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix} \tag{A.16}$$

and find that (A.16) can be factorised as

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{R}_{21} & \mathbf{U}^T \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L}_{11}^{-1}\mathbf{L}_{12} \\ \mathbf{0} & \mathbf{L}^T\mathbf{P} \end{pmatrix} \tag{A.17}$$

Theorem (A.2) gives that the space of $(\mathbf{AB})$ is equal to

$$\begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{R}_{21} & \mathbf{U}^T \end{pmatrix} \tag{A.18}$$

The common space can now simply be found as the last nonzero column of $\mathbf{U}^T$.

Appendix B

Sensitivity to Modelling Errors Experiment

In section 2.6 we made a sensitivity to modelling errors experiment, were we compared the sensitivity of closed loop systems to modelling errors when the controller design was based on the unreduced model versus first and second order approximations. Here, the results of these experiments are presented in the form of tables. The systems are sorted into tables with equal stability properties.

	Full System			2nd order approximation			1st order approximation		
	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,3)\|}$	$max\|\lambda_i'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,2)\|}$	$max\|\lambda_i'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\|U^T(:,1)\|$	$max\|\lambda_i'\|$
1	$\frac{1q^{-1}(1-0.99q^{-1})(1-0.88q^{-1})}{(1-0.99q^{-1})(1-0.97q^{-1})(1-0.64q^{-1})}$	10^8	12.425	$\frac{1q^{-1}(1-0.88q^{-1})}{(1-0.97q^{-1})(1-0.64q^{-1})}$	10^2	0.809	$\frac{1q^{-1}}{(1-0.77q^{-1})}$	10^0	0.805
2	$\frac{1q^{-1}(1-1.02q^{-1})(1-1.01q^{-1})}{(1-0.99q^{-1})(1-0.97q^{-1})(1-0.97q^{-1})}$	10^7	3.023	$\frac{1q^{-1}(1-1.03q^{-1})}{(1-0.97q^{-1})(1-0.97q^{-1})}$	10^2	0.823	$\frac{1q^{-1}}{(1-0.90q^{-1})}$	10^0	0.807
3	$\frac{1q^{-1}(1-1.04q^{-1})(1-0.97q^{-1})}{(1-0.99q^{-1})(1-0.95q^{-1})(1-0.89q^{-1})}$	10^6	1.257	$\frac{1q^{-1}(1-1.04q^{-1})}{(1-0.97q^{-1})(1-0.89q^{-1})}$	10^2	0.781	$\frac{1q^{-1}}{(1-0.81q^{-1})}$	10^0	0.806
4	$\frac{1q^{-1}(1-0.97q^{-1})(1-0.08q^{-1})}{(1-0.97q^{-1})(1-0.95q^{-1})(1-0.64q^{-1})}$	10^6	1.551	$\frac{1q^{-1}(1-0.08q^{-1})}{(1-0.95q^{-1})(1-0.64q^{-1})}$	10^1	0.745	$\frac{1q^{-1}}{(1-1.03q^{-1})}$	10^1	0.818
5	$\frac{1q^{-1}(1-1.13q^{-1})(1-0.91q^{-1})}{(1-0.89q^{-1})(1-0.89q^{-1})(1-0.75q^{-1})}$	10^6	1.020	$\frac{1q^{-1}(1-1.13q^{-1})}{(1-0.87q^{-1})(1-0.76q^{-1})}$	10^1	0.752	$\frac{1q^{-1}}{(1-0.32q^{-1})}$	10^0	0.766
6	$\frac{1q^{-1}(1-1.02q^{-1})(1-0.01q^{-1})}{(1-0.99q^{-1})(1-0.75q^{-1})(1-0.01q^{-1})}$	10^5	7.595	$\frac{1q^{-1}(1-1.02q^{-1})}{(1-0.99q^{-1})(1-0.75q^{-1})}$	10^2	0.870	$\frac{1q^{-1}}{(1-0.72q^{-1})}$	10^0	0.808
7	$\frac{1q^{-1}(1-0.95q^{-1})(1-0.00q^{-1})}{(1-0.99q^{-1})(1-0.95q^{-1})(1-0.33q^{-1})}$	10^5	1.127	$\frac{1q^{-1}(1-0.01q^{-1})}{(1-1.00q^{-1})(1-0.33q^{-1})}$	10^1	0.773	$\frac{1q^{-1}}{(1-1.00q^{-1})}$	10^0	0.810
8	$\frac{1q^{-1}(1-0.99q^{-1})(1-0.11q^{-1})}{(1-0.89q^{-1})(1-0.75q^{-1})(1-0.11q^{-1})}$	10^5	3.181	$\frac{1q^{-1}(1-0.99q^{-1})}{(1-0.89q^{-1})(1-0.76q^{-1})}$	10^1	0.901	$\frac{1q^{-1}}{(1-0.62q^{-1})}$	10^0	0.800
9	$\frac{1q^{-1}(1-0.99q^{-1})(1-0.97q^{-1})}{(1-0.97q^{-1})(1-0.11q^{-1})(1-0.11q^{-1})}$	10^4	1.131	$\frac{1q^{-1}(1-0.98q^{-1})}{(1-0.12q^{-1})(1-0.12q^{-1})}$	10^2	0.963	$\frac{1q^{-1}}{(1-0.23q^{-1})}$	10^0	0.754
10	$\frac{1q^{-1}(1-1.00q^{-1})(1-0.66q^{-1})}{(1-0.99q^{-1})(1-0.89q^{-1})(1-0.33q^{-1})}$	10^4	1.127	$\frac{1q^{-1}(1-0.65q^{-1})}{(1-0.87q^{-1})(1-0.33q^{-1})}$	10^1	0.731	$\frac{1q^{-1}}{(1-0.69q^{-1})}$	10^0	0.809
11	$\frac{1q^{-1}(1-1.00q^{-1})(1-0.95q^{-1})}{(1-0.99q^{-1})(1-0.64q^{-1})(1-0.11q^{-1})}$	10^4	1.132	$\frac{1q^{-1}(1-0.96q^{-1})}{(1-0.64q^{-1})(1-0.11q^{-1})}$	10^0	0.838	$\frac{1q^{-1}}{(1-0.69q^{-1})}$	10^{-0}	0.808
12	$\frac{1q^{-1}(1-3.61q^{-1})(1-1.27q^{-1})}{(1-0.89q^{-1})(1-0.33q^{-1})(1-0.01q^{-1})}$	10^3	1.311	$\frac{1q^{-1}(1-4.88q^{-1})}{(1-0.88q^{-1})(1-0.35q^{-1})}$	10^1	0.769	$\frac{1q^{-1}}{(1-0.06q^{-1})}$	10^1	0.902
13	$\frac{1q^{-1}(1-10.4q^{-1})(1-0.94q^{-1})}{(1-0.80q^{-1})(1-0.64q^{-1})(1-0.01q^{-1})}$	10^3	1.006	$\frac{1q^{-1}(1-11.3q^{-1})}{(1-0.88q^{-1})(1-0.52q^{-1})}$	10^1	0.820	$\frac{1q^{-1}}{(1-0.51q^{-1})}$	10^1	0.738
14	$\frac{1q^{-1}(1-127.q^{-1})(1-1.00q^{-1})}{(1-0.89q^{-1})(1-0.75q^{-1})(1-0.01q^{-1})}$	10^2	1.214	$\frac{1q^{-1}(1-128.q^{-1})}{(1-1.06q^{-1})(1-0.48q^{-1})}$	10^1	0.945	$\frac{1q^{-1}}{(1-0.66q^{-1})}$	10^2	0.806

Table B.1: In all these examples control design based on the non-reduced models resulted in unstable closed loop systems. Control design based on the first and second order approximations gave stable closed loop systems. It can be seen that all systems have in common that they have very high condition numbers. Models 1, 2, 3 and 6 have close to common factors. Models 13 and 14 have one pole that is much faster than the others.

	Full System			2nd order approximation			1st order approximation		
	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,3)\|}$	$max\|\lambda_i{}'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,2)\|}$	$max\|\lambda_i{}'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\|U^T(:,1)\|$	$max\|\lambda_i{}'\|$
1	$\frac{1q^{-1}(1-1.05q^{-1})(1-0.00q^{-1})}{(1-0.11q^{-1})(1-0.01q^{-1})(1-0.01q^{-1})}$	10^8	179.897	$\frac{1q^{-1}(1-1.05q^{-1})}{(1-0.11q^{-1})(1-0.01q^{-1})}$	10^3	1.215	$\frac{1q^{-1}}{(1-0.12q^{-1})}$	10^{-0}	0.739
2	$\frac{1q^{-1}(1-8.93q^{-1})(1-0.82q^{-1})}{(1-0.11q^{-1})(1-0.01q^{-1})(1-0.01q^{-1})}$	10^5	18.750	$\frac{1q^{-1}(1-9.75q^{-1})}{(1-0.10q^{-1})(1-0.02q^{-1})}$	10^0	2.037	$\frac{1q^{-1}}{(1+0.63q^{-1})}$	10^1	0.920

Table B.2: These models gave unstable closed loop systems when the control design was based on the full order system and the second order approximations, but stable closed loop systems when the control design was based on the first order approximation. The models have in common that all poles are very fast.

	Full System			2nd order approximation			1st order approximation		
	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,3)\|}$	$max\|\lambda_i^{'}\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,2)\|}$	$max\|\lambda_i^{'}\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\|U^T(:,1)\|$	$max\|\lambda_i^{'}\|$
1	$\frac{1q^{-1}(1-1.46q^{-1})(1-1.23q^{-1})}{(1-0.33q^{-1})(1-0.33q^{-1})(1-0.01q^{-1})}$	10^3	1.592	$\frac{1q^{-1}(1-2.68q^{-1})}{(1-0.34q^{-1})(1-0.34q^{-1})}$	10^1	0.891	$\frac{1q^{-1}}{(1+0.15q^{-1})}$	10^0	1.141

Table B.3: This model gave an unstable closed loop system when the control design was based on the non-reduced system and the first order approximation, but a stable closed loop system when the control design was based on the second order approximation. It can be seen that the condition number of the second order model is small, indicating that it should not be further reduced.

	Full System			2nd order approximation			1st order approximation		
	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,3)\|}$	$max\|\lambda_i'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,2)\|}$	$max\|\lambda_i'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\|U^T(:,1)\|$	$max\|\lambda_i'\|$
1	$\frac{1q^{-1}(1-1.13q^{-1})(1-1.02q^{-1})}{(1-0.97q^{-1})(1-0.11q^{-1})(1-0.11q^{-1})}$	10^3	0.913	$\frac{1q^{-1}(1-1.43q^{-1})}{(1-0.65q^{-1})(1+0.17q^{-1})}$	10^2	1.175	$\frac{1q^{-1}}{(1-0.22q^{-1})}$	10^0	0.776
2	$\frac{1q^{-1}(1-90.8q^{-1})(1-0.35q^{-1})}{(1-0.97q^{-1})(1-0.95q^{-1})(1-0.75q^{-1})}$	10^2	0.881	$\frac{1q^{-1}(1-90.7q^{-1})}{(1-1.16q^{-1})(1-1.16q^{-1})}$	10^1	1.011	$\frac{1q^{-1}}{(1-1.43q^{-1})}$	10^3	0.825
3	$\frac{1q^{-1}(1-1.33q^{-1})(1-1.33q^{-1})}{(1-0.95q^{-1})(1-0.80q^{-1})(1-0.01q^{-1})}$	10^2	0.994	$\frac{1q^{-1}(1-2.57q^{-1})}{(1-1.11q^{-1})(1-0.54q^{-1})}$	10^1	1.003	$\frac{1q^{-1}}{(1-0.66q^{-1})}$	10^0	0.808
4	$\frac{1q^{-1}(1-1.12q^{-1})(1-0.68q^{-1})}{(1-0.99q^{-1})(1-0.97q^{-1})(1-0.33q^{-1})}$	10^2	0.861	$\frac{1q^{-1}(1+0.51q^{-1})}{(1+1.29q^{-1})(1+1.29q^{-1})}$	10^2	1.289	$\frac{1q^{-1}}{(1-0.58q^{-1})}$	10^0	0.795

Table B.4: These models gave unstable closed loop systems when the control design was based on the second order approximation, but stable closed loop systems when the control design was based on the non-reduced system and the first order approximation. All models have a relatively low condition number. None of the models have close to common factors.

	Full System			2nd order approximation			1st order approximation		
	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\lVert U^T(:,1)\rVert}{\lVert U^T(:,3)\rVert}$	$max\lvert\lambda_i'\rvert$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\lVert U^T(:,1)\rVert}{\lVert U^T(:,2)\rVert}$	$max\lvert\lambda_i'\rvert$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\lVert U^T(:,1)\rVert$	$max\lvert\lambda_i'\rvert$
1	$\frac{1q^{-1}(1-1.09q^{-1})(1-0.90q^{-1})}{(1-1.02q^{-1})(1-0.89q^{-1})(1-0.33q^{-1})}$	10^4	0.969	$\frac{1q^{-1}(1-1.09q^{-1})}{(1-1.01q^{-1})(1-0.33q^{-1})}$	10^1	0.823	$\frac{1q^{-1}}{(1-0.01q^{-1})}$	10^{-0}	1.956

Table B.5: This model gave a stable closed loop system when control design was based on the non-reduced model and the second order approximation, but unstable closed loop system when the control design was based on the first order approximation. It can be seen that there is a near common factor in the non-reduced system. This factor is cancelled in the second order approximation. The second order approximation has poles and zeros far from each other, and a low condition number, and should not be reduced.

	Full System			2nd order approximation			1st order approximation		
	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\lVert U^T(:,1)\rVert}{\lVert U^T(:,3)\rVert}$	$max\lvert\lambda_i'\rvert$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\lVert U^T(:,1)\rVert}{\lVert U^T(:,2)\rVert}$	$max\lvert\lambda_i'\rvert$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\lVert U^T(:,1)\rVert$	$max\lvert\lambda_i'\rvert$
1	$\frac{1q^{-1}(1-3.00q^{-1})(1-1.51q^{-1})}{(1-3q^{-1})(1-3q^{-1})(1-1.31q^{-1})}$	10^8	2754.230	$\frac{1q^{-1}(1-1.51q^{-1})}{(1-2.99q^{-1})(1-1.31q^{-1})}$	10^2	1.324	$\frac{1q^{-1}}{(1-2.98q^{-1})}$	10^2	0.684
2	$\frac{1q^{-1}(1-1.28q^{-1})(1-0.30q^{-1})}{(1-81q^{-1})(1-1.11q^{-1})(1-1.02q^{-1})}$	10^7	2.044	$\frac{1q^{-1}(1-0.12q^{-1})}{(1-80.9q^{-1})(1-0.67q^{-1})}$	10^6	1.459	$\frac{1q^{-1}}{(1-81q^{-1})}$	10^6	0.711
3	$\frac{1q^{-1}(1-1.01q^{-1})(1-0.12q^{-1})}{(1-9q^{-1})(1-1.04q^{-1})(1-0.01q^{-1})}$	10^6	1.964	$\frac{1q^{-1}(1-1.18q^{-1})}{(1-8.99q^{-1})(1-1.11q^{-1})}$	10^5	1.108	$\frac{1q^{-1}}{(1-9.00q^{-1})}$	10^3	0.696
4	$\frac{1q^{-1}(1-1.00q^{-1})(1-0.11q^{-1})}{(1-0.97q^{-1})(1-0.95q^{-1})(1-0.11q^{-1})}$	10^5	6.323	$\frac{1q^{-1}(1-1.00q^{-1})}{(1-0.97q^{-1})(1-0.97q^{-1})}$	10^3	1.593	$\frac{1q^{-1}}{(1-0.93q^{-1})}$	10^0	0.801
5	$\frac{1q^{-1}(1-6.47q^{-1})(1-1.00q^{-1})}{(1-1.04q^{-1})(1-1.00q^{-1})(1-0.01q^{-1})}$	10^5	2.882	$\frac{1q^{-1}(1-6.47q^{-1})}{(1-1.04q^{-1})(1-0.01q^{-1})}$	10^2	1.254	$\frac{1q^{-1}}{(1-1.04q^{-1})}$	10^1	0.799
6	$\frac{1q^{-1}(1-1.10q^{-1})(1-0.97q^{-1})}{(1-3q^{-1})(1-1.11q^{-1})(1-0.01q^{-1})}$	10^5	1.311	$\frac{1q^{-1}(1-0.89q^{-1})}{(1-2.97q^{-1})(1+0.03q^{-1})}$	10^3	1.161	$\frac{1q^{-1}}{(1-3.00q^{-1})}$	10^1	0.692
7	$\frac{1q^{-1}(1-0.79q^{-1})(1-0.56q^{-1})}{(1-9q^{-1})(1-1.55q^{-1})(1-1.24q^{-1})}$	10^5	1.262	$\frac{1q^{-1}(1-0.23q^{-1})}{(1-8.99q^{-1})(1-1.67q^{-1})}$	10^3	1.500	$\frac{1q^{-1}}{(1-9.00q^{-1})}$	10^3	0.673
8	$\frac{1q^{-1}(1-1.31q^{-1})(1-0.25q^{-1})}{(1-9q^{-1})(1-0.80q^{-1})(1-0.64q^{-1})}$	10^4	1.221	$\frac{1q^{-1}(1-1.17q^{-1})}{(1-8.98q^{-1})(1-1.06q^{-1})}$	10^3	1.116	$\frac{1q^{-1}}{(1-8.99q^{-1})}$	10^3	0.670
9	$\frac{1q^{-1}(1-0.77q^{-1})(1-0.11q^{-1})}{(1-9q^{-1})(1-3q^{-1})(1-1.31q^{-1})}$	10^4	1.676	$\frac{1q^{-1}(1+0.38q^{-1})}{(1-8.98q^{-1})(1-3.05q^{-1})}$	10^2	2.253	$\frac{1q^{-1}}{(1-9.08q^{-1})}$	10^4	0.707
10	$\frac{1q^{-1}(1-8.41q^{-1})(1-0.97q^{-1})}{(1-3q^{-1})(1-1.02q^{-1})(1-0.89q^{-1})}$	10^4	1.029	$\frac{1q^{-1}(1-8.41q^{-1})}{(1-3.00q^{-1})(1-0.94q^{-1})}$	10^1	1.048	$\frac{1q^{-1}}{(1-3.09q^{-1})}$	10^2	0.698
11	$\frac{1q^{-1}(1-1.55q^{-1})(1-1.09q^{-1})}{(1-9q^{-1})(1-0.33q^{-1})(1-0.11q^{-1})}$	10^4	1.207	$\frac{1q^{-1}(1-2.76q^{-1})}{(1-8.98q^{-1})(1-0.57q^{-1})}$	10^3	1.924	$\frac{1q^{-1}}{(1-9.00q^{-1})}$	10^3	0.673
12	$\frac{1q^{-1}(1-0.97q^{-1})(1-0.64q^{-1})}{(1-9.00q^{-1})(1-8.99q^{-1})(1-8.99q^{-1})}$	10^4	8.505	$\frac{1q^{-1}(1+3.95q^{-1})}{(1-11.1q^{-1})(1-11.1q^{-1})}$	10^2	5.359	$\frac{1q^{-1}}{(1-13.3q^{-1})}$	10^6	0.682
13	$\frac{1q^{-1}(1-1.54q^{-1})(1-0.00q^{-1})}{(1-9q^{-1})(1-1.31q^{-1})(1-0.64q^{-1})}$	10^4	1.231	$\frac{1q^{-1}(1-1.56q^{-1})}{(1-8.87q^{-1})(1-2.10q^{-1})}$	10^4	1.444	$\frac{1q^{-1}}{(1-8.99q^{-1})}$	10^3	0.683
14	$\frac{1q^{-1}(1-3.74q^{-1})(1-1.04q^{-1})}{(1-3q^{-1})(1-1.02q^{-1})(1-0.75q^{-1})}$	10^3	1.232	$\frac{1q^{-1}(1-3.74q^{-1})}{(1-2.99q^{-1})(1-0.73q^{-1})}$	10^1	1.185	$\frac{1q^{-1}}{(1-3.14q^{-1})}$	10^1	0.698

Table B.6: These models resulted in unstable closed loop systems when control design was based on the non-reduced models or the second order approximation. Control design based on the first order approximations gave stable closed loop systems. It can be seen that the non-reduced system, and in some cases also the second order approximation, have very high condition numbers.

	Full System			2nd order approximation			1st order approximation		
	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,3)\|}$	$max\|\lambda_i{}'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\frac{\|U^T(:,1)\|}{\|U^T(:,2)\|}$	$max\|\lambda_i{}'\|$	$\frac{B(q^{-1})}{A(q^{-1})}$	$\|U^T(:,1)\|$	$max\|\lambda_i{}'\|$
1	$\frac{1q^{-1}(1-0.00q^{-1})(1-0.00q^{-1})}{(1-81q^{-1})(1-1.55q^{-1})(1-0.01q^{-1})}$	10^{13}	257.890	$\frac{1q^{-1}(1-0.00q^{-1})}{(1-81q^{-1})(1-1.55q^{-1})}$	10^5	16.385	$\frac{1q^{-1}}{(1-81.0q^{-1})}$	10^6	15.648
2	$\frac{1q^{-1}(1-0.97q^{-1})(1-0.79q^{-1})}{(1-81q^{-1})(1-1.11q^{-1})(1-0.80q^{-1})}$	10^{10}	26.381	$\frac{1q^{-1}(1-0.97q^{-1})}{(1-81.0q^{-1})(1-1.11q^{-1})}$	10^7	26.222	$\frac{1q^{-1}}{(1-81q^{-1})}$	10^6	25.136
3	$\frac{1q^{-1}(1-0.76q^{-1})(1-0.64q^{-1})}{(1-81q^{-1})(1-0.64q^{-1})(1-0.11q^{-1})}$	10^{10}	16.720	$\frac{1q^{-1}(1-0.76q^{-1})}{(1-80.9q^{-1})(1-0.10q^{-1})}$	10^7	16.874	$\frac{1q^{-1}}{(1-81q^{-1})}$	10^6	16.990
4	$\frac{1q^{-1}(1-3.23q^{-1})(1-0.89q^{-1})}{(1-81.0q^{-1})(1-80.9q^{-1})(1-0.95q^{-1})}$	10^9	62.240	$\frac{1q^{-1}(1-3.17q^{-1})}{(1-81.0q^{-1})(1-80.9q^{-1})}$	10^3	61.947	$\frac{1q^{-1}}{(1-101.q^{-1})}$	10^8	14.941
5	$\frac{1q^{-1}(1-1.10q^{-1})(1-0.29q^{-1})}{(1-81q^{-1})(1-1.11q^{-1})(1-0.75q^{-1})}$	10^9	19.971	$\frac{1q^{-1}(1-0.30q^{-1})}{(1-81.0q^{-1})(1-0.77q^{-1})}$	10^6	19.269	$\frac{1q^{-1}}{(1-81q^{-1})}$	10^6	19.060
6	$\frac{1q^{-1}(1-0.90q^{-1})(1-0.30q^{-1})}{(1-81q^{-1})(1-1.31q^{-1})(1-0.01q^{-1})}$	10^8	7.941	$\frac{1q^{-1}(1-1.19q^{-1})}{(1-80.9q^{-1})(1-1.31q^{-1})}$	10^6	9.125	$\frac{1q^{-1}}{(1-81.0q^{-1})}$	10^6	7.364
7	$\frac{1q^{-1}(1-2.53q^{-1})(1-1.01q^{-1})}{(1-81q^{-1})(1-0.95q^{-1})(1-0.75q^{-1})}$	10^8	15.893	$\frac{1q^{-1}(1-2.52q^{-1})}{(1-80.9q^{-1})(1-0.68q^{-1})}$	10^6	15.325	$\frac{1q^{-1}}{(1-80.9q^{-1})}$	10^6	13.397
8	$\frac{1q^{-1}(1-109.q^{-1})(1-3.15q^{-1})}{(1-81.0q^{-1})(1-80.9q^{-1})(1-0.97q^{-1})}$	10^5	191.473	$\frac{1q^{-1}(1-111.q^{-1})}{(1-81.1q^{-1})(1-80.8q^{-1})}$	10^1	177.986	$\frac{1q^{-1}}{(1-595.q^{-1})}$	10^7	519.762
9	$\frac{1q^{-1}(1-247.q^{-1})(1-5.07q^{-1})}{(1-81q^{-1})(1-1.24q^{-1})(1-1.00q^{-1})}$	10^4	50.278	$\frac{1q^{-1}(1-252.q^{-1})}{(1-80.9q^{-1})(1-1.79q^{-1})}$	10^3	45.905	$\frac{1q^{-1}}{(1-80.9q^{-1})}$	10^6	15.264
10	$\frac{1q^{-1}(1-8.98q^{-1})(1-0.01q^{-1})}{(1-9q^{-1})(1-1.00q^{-1})(1-0.95q^{-1})}$	10^1	65.568	$\frac{1q^{-1}(1-8.63q^{-1})}{(1-8.94q^{-1})(1-1.64q^{-1})}$	10^1	15.884	$\frac{1q^{-1}}{(1-3.45q^{-1})}$	10^2	6.054
11	$\frac{1q^{-1}(1-1.33q^{-1})(1-1.31q^{-1})}{(1-1.55q^{-1})(1-0.99q^{-1})(1-0.11q^{-1})}$	10^{-0}	1.052	$\frac{1q^{-1}(1-2.60q^{-1})}{(1-1.30q^{-1})(1-1.30q^{-1})}$	10^{-0}	1.371	$\frac{1q^{-1}}{(1+0.48q^{-1})}$	10^{-1}	5.732

Table B.7: These examples resulted in unstable closed loop system for all three cases. The condition number is extremely large in some cases. The models are extremely unstable.

Appendix C

Proof/Derivation of the Multiple Model Least Squares Identification Method

In this section we will give a derivation of the Multiple Model Least Squares method. The method, presented here, differs from the one given by Steve Niu in the representation of the augmented data vector. The data vector we use is mirrored compared to the one used by Niu *et al* [17]. This means that the algorithm has to be mirrored as well. Niu uses regressor vectors like

$$\varphi(t) = [-y(t-n)\ u(t-n)\ \ldots\ -y(t-1)\ u(t-1)\ \ -y(t)]^T.$$

leading to the algorithm,

$$\begin{aligned} S(t) &= L(t)D(t)L^T(t) \\ \Theta(t) &= L^{-T}(t) \\ J(t) &= D(t). \end{aligned}$$

With our definition of regressor vector,

$$\varphi(t) = [-y(t)\ u(t-1)\ \ -y(t-1)\ \ldots\ u(t-n)\ \ -y(t-n)]^T, \tag{C.1}$$

the algorithm will become

$$\begin{aligned} S(t) &= U(t)D(t)U^T(t) \\ \Theta(t) &= U^{-T}(t) \\ J(t) &= D(t). \end{aligned}$$

The algorithms have, however, the same properties.

Proof of the Multiple Model Least Squares Method

Represent the augmented data-vector $\underline{\varphi}(t)$ in the shift-structure,

$$\begin{array}{ccccccccccccc}
\underline{\varphi}(t) & = & [& -y_t & u_{t-1} & -y_{t-1} & u_{t-2} & \cdots & -y_{t-(n-1)} & u_{t-n} & & -y_{t-n} &]^T \\
 & & & & & & & & & & \lfloor & \varphi_{fb}^{(0)^T} & \rfloor \\
 & & & & & & & & & \lfloor & \varphi_{ff}^{(1)^T} & & \rfloor \\
 & & & & & & & & \lfloor & & \varphi_{fb}^{(1)^T} & & \rfloor \\
 & & & & & & & & & \vdots & & & \\
 & & & & & & \lfloor & & \varphi_{ff}^{(n-1)^T} & & & & \rfloor \\
 & & & & & \lfloor & & & \varphi_{fb}^{(n-1)^T} & & & & \rfloor \\
 & & & & \lfloor & & & & \varphi_{ff}^{(n)^T} & & & & \rfloor
\end{array}$$

$\underline{\varphi}(t)$ has the property that the time index decreases from left to right. The reason for this special structure is that we want to give the subspace of $\underline{\varphi}(t)$ the meaning of causal lower order regressors for lower order models. 'ff' stands for feed-forward model and 'fb' for feed-back model.
The subvectors of $\underline{\varphi}(t)$ fulfil that

$$\begin{aligned}
0 &= \underline{\varphi}^T(t)\hat{\underline{\theta}}(t) & \qquad y(t) &= \varphi_{ff}^{(n)^T}\hat{\theta}_{ff}^{(n)} \\
u(t-1) &= \varphi_{fb}^{(n-1)^T}\hat{\theta}_{fb}^{(n-1)} & \qquad y(t-1) &= \varphi_{ff}^{(n-1)^T}\hat{\theta}_{ff}^{(n-1)} \\
&\vdots & &\vdots \\
u(t-(n-1)) &= \varphi_{fb}^{(1)^T}\hat{\theta}_{fb}^{(1)} & \qquad y(t-(n-1)) &= \varphi_{ff}^{(1)^T}\hat{\theta}_{ff}^{(1)} \\
u(t-n) &= \varphi_{fb}^{(0)^T}\hat{\theta}_{fb}^{(0)} & &
\end{aligned}$$

and can be interpreted as regressor vectors for models of lower orders.

The augmented data-vector can be written in the following order recursive fashion,

$$\begin{aligned}
\underline{\varphi}(t) &= \begin{pmatrix} -y(t) \\ \varphi_{ff}^{(n)} \end{pmatrix} & \qquad \varphi_{ff}^{(n)} &= \begin{pmatrix} u(t-1) \\ \varphi_{fb}^{(n-1)} \end{pmatrix} \\
\varphi_{fb}^{(n-1)} &= \begin{pmatrix} -y(t-1) \\ \varphi_{ff}^{(n-1)} \end{pmatrix} & \qquad \varphi_{ff}^{(n-1)} &= \begin{pmatrix} u(t-2) \\ \varphi_{fb}^{(n-2)} \end{pmatrix} \\
&\vdots & &\vdots \\
\varphi_{fb}^{(1)} &= \begin{pmatrix} -y(t-(n-1)) \\ \varphi_{ff}^{(1)} \end{pmatrix} & \qquad \varphi_{ff}^{(1)} &= \begin{pmatrix} u(t-n) \\ \varphi_{fb}^{(0)} \end{pmatrix} \\
\varphi_{fb}^{(0)} &= -y(t-n) & &
\end{aligned}$$

Define the augmented information matrix (AIM) as

$$\underline{\mathbf{S}}(t) = \sum_{k=1}^{t} \underline{\varphi}(k)\underline{\varphi}^T(k). \tag{C.2}$$

Now partitionate $\underline{\mathbf{S}}(t)$ into lower dimensional sub-matrices,

$$\begin{aligned}
\underline{\mathbf{S}}(t) &= \sum_{k=1}^{t} \underline{\varphi}(k)\underline{\varphi}^T(k) \\
&= \sum_{k=1}^{t} \begin{pmatrix} -y(k) \\ \varphi_{ff}^{(n)}(k) \end{pmatrix} \begin{pmatrix} -y(k) & {\varphi_{ff}^{(n)}}^T(k) \end{pmatrix} \\
&= \begin{pmatrix} \sum_{k=1}^{t} y(k)^2 & -\sum_{k=1}^{t} y(k){\varphi_{ff}^{(n)}}^T(k) \\ -\sum_{k=1}^{t} \varphi_{ff}^{(n)}(k)y(k) & \sum_{k=1}^{t} \varphi_{ff}^{(n)}(k){\varphi_{ff}^{(n)}}^T(k) \end{pmatrix} \\
&= \begin{pmatrix} \sum_{k=1}^{t} y(k)^2 & -\sum_{k=1}^{t} y(k){\varphi_{ff}^{(n)}}^T(k) \\ -\sum_{k=1}^{t} \varphi_{ff}^{(n)}(k)y(k) & \mathbf{S}_{ff}^{(n)}(t) \end{pmatrix}
\end{aligned}$$

The lower right sub matrix is then the information matrix for the n:th order feed-forward model, $\mathbf{S}_{ff}^{(n)}(k) = \sum_{k=1}^{t} \varphi_{ff}^{(n)}(k){\varphi_{ff}^{(n)}}^T(k)$. The following theorem is the key to the solution of the MMLS problem.

Theorem C.1 (Decomposition of Partitionated Matrix)
[1] *A non-negative definite symmetrical matrix* $\mathbf{S}$ *can be factorised as*

$$\begin{pmatrix} D & B^T \\ B & A \end{pmatrix} = \begin{pmatrix} 1 & B^TA^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \Delta & 0 \\ 0 & A \end{pmatrix} \begin{pmatrix} 1 & B^TA^{-1} \\ 0 & I \end{pmatrix}^T, \tag{C.3}$$

where A is a sub matrix of S, and is assumed to be non-singular, and

$$\Delta = D - B^TA^{-1}B \tag{C.4}$$

Proof. Direct multiplication of the right-hand side of C.3 gives the left hand side.

Now we apply this theorem on $\mathbf{S}(t)$. Matrices A, B and D are then

$$A = \sum_{k=1}^{t} \varphi_{ff}^{(n)}(k)\varphi_{ff}^{(n)}(k)^T \qquad B = -\sum_{k=1}^{t} \varphi_{ff}^{(n)}(k)y(k)$$

$$D = \sum_{k=1}^{t} y(k)^2$$

[1] Niu uses the decomposition:

$$\begin{pmatrix} A & B \\ B^T & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ B^TA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & \Delta \end{pmatrix} \begin{pmatrix} I & 0 \\ B^TA^{-1} & I \end{pmatrix}^T$$

where A is a sub matrix of S, and is assumed to be non-singular, and

$$\Delta = D - B^TA^{-1}B.$$

$B^T A^{-1}$ becomes then

$$\begin{aligned}
B^T A^{-1} &= \left(-\sum_{k=1}^{t} \varphi_{ff}^{(n)^T}(k) y(k)\right) \left(\sum_{k=1}^{t} \varphi_{ff}^{(n)}(k) \varphi_{ff}^{(n)^T}(k)\right)^{-1} \\
&= -\left(\left(\sum_{k=1}^{t} \varphi_{ff}^{(n)}(k) \varphi_{ff}^{(n)^T}(k)\right)^{-1} \left(\sum_{k=1}^{t} \varphi_{ff}^{(n)}(k) y(k)\right)\right)^T \\
&= -\hat{\theta}_{ff}^{(n)}(t)^T.
\end{aligned}$$

We see that $B^T A^{-1}$ is equal to the transpose of the negative n:th order feed-forward parameter-vector, $-\hat{\theta}_{ff}^{(n)}(t)^T$. Δ becomes,

$$\begin{aligned}
\Delta &= D - B^T A^{-1} B \\
&= \sum_{k=1}^{t} y^2(k) - \left(-\sum_{k=1}^{t} y(k) \varphi_{ff}^{(n)^T}(k)\right)(-\hat{\theta}_{ff}^{(n)}) \\
&= \sum_{k=1}^{t} y(k) \left(y(k) - \varphi_{ff}^{(n)^T}(k) \hat{\theta}_{ff}^{(n)}\right) \\
&\uparrow \\
\{\textstyle\sum_{k=1}^{t} & \varphi_{ff}^{(n)^T}(k) \hat{\theta}_{ff}^{(n)} \left(y(k) - \varphi_{ff}^{(n)^T}(k) \hat{\theta}_{ff}^{(n)}\right) = 0\} \\
&= \sum_{k=1}^{t} \left(y(k) - \varphi_{ff}^{(n)^T}(k) \hat{\theta}_{ff}^{(n)}\right)^2 \\
&= \sum_{k=1}^{t} e_{ff}^{(n)^2}(k) \\
&= J_{ff}^{(n)}(t)
\end{aligned}$$

Thus Δ is equal to the loss function for the n:th order feed-forward model, $\Delta = J_{ff}^{(n)}(t)$. Substitution of $B^T A^{-1}$ and Δ into C.3 gives that the augmented information matrix can be factorised as

$$\underline{\mathbf{S}}(t) = \begin{pmatrix} 1 & -\hat{\theta}_{ff}^{(n)}(t)^T \\ 0 & I_{2n} \end{pmatrix} \begin{pmatrix} \mathbf{J}_{ff}^{(n)} & 0 \\ 0 & \mathbf{S}_{ff}^{(n)} \end{pmatrix} \begin{pmatrix} 1 & -\hat{\theta}_{ff}^{(n)}(t)^T \\ 0 & I_{2n} \end{pmatrix}^T$$

The lower right sub matrix, $\mathbf{S}_{ff}^{(n)}(t)$, can be written as

$$\mathbf{S}_{ff}^{(n)}(t) = \begin{pmatrix} \sum_{k=1}^{t} y(k)^2 & -\sum_{k=1}^{t} y(k) \varphi_{fb}^{(n-1)^T}(k) \\ -\sum_{k=1}^{t} \varphi_{fb}^{(n-1)}(k) y(k) & \mathbf{S}_{fb}^{(n-1)}(t) \end{pmatrix}$$

Application of Theorem C.3 on $\mathbf{S}_{ff}^{(n)}(t)$ gives

$$\mathbf{S}_{ff}^{(n)} = \begin{pmatrix} 1 & -\hat{\theta}_{fb}^{(n-1)}(t)^T \\ 0 & I_{2n-1} \end{pmatrix} \begin{pmatrix} \mathbf{J}_{fb}^{(n-1)} & 0 \\ 0 & \mathbf{S}_{fb}^{(n-1)} \end{pmatrix} \begin{pmatrix} 1 & -\hat{\theta}_{fb}^{(n-1)}(t)^T \\ 0 & I_{2n-1} \end{pmatrix}^T$$

This procedure can be repeated until the lower right submatix is a scalar.

$$\begin{aligned}
\mathbf{S}_{fb}^{(n-1)} &= \begin{pmatrix} 1 & -\hat{\theta}_{ff}^{(n-1)}(t)^T \\ 0 & I_{2n-2} \end{pmatrix} \begin{pmatrix} \mathbf{J}_{ff}^{(n-1)} & 0 \\ 0 & \mathbf{S}_{ff}^{(n-1)} \end{pmatrix} \begin{pmatrix} 1 & -\hat{\theta}_{ff}^{(n-1)}(t)^T \\ 0 & I_{2n-2} \end{pmatrix}^T \\
&\vdots \\
\mathbf{S}_{fb}^{(1)} &= \begin{pmatrix} 1 & -\hat{\theta}_{ff}^{(1)}(t)^T \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} \mathbf{J}_{ff}^{(1)} & 0 \\ 0 & \mathbf{S}_{ff}^{(1)} \end{pmatrix} \begin{pmatrix} 1 & -\hat{\theta}_{ff}^{(1)}(t)^T \\ 0 & I_2 \end{pmatrix}^T \\
\mathbf{S}_{ff}^{(1)} &= \begin{pmatrix} 1 & -\hat{\theta}_{fb}^{(0)}(t)^T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{J}_{fb}^{(0)} & 0 \\ 0 & \mathbf{S}_{fb}^{(0)} \end{pmatrix} \begin{pmatrix} 1 & -\hat{\theta}_{fb}^{(0)}(t)^T \\ 0 & 1 \end{pmatrix}^T \\
\mathbf{S}_{fb}^{(0)} &= \sum_{k=1}^{t} y^2(k)
\end{aligned}$$

Together, these transformations define a UDU^T transformation of $\underline{\mathbf{S}}(t)$ according to

$$\begin{aligned}
\underline{\mathbf{S}}(t) &= \left(\mathbf{U}_{ff}^{(n)}\mathbf{U}_{fb}^{(n-1)}\cdots\mathbf{U}_{ff}^{(1)}\mathbf{U}_{fb}^{(0)}\right)\cdot D \cdot \left(\mathbf{U}_{fb}^{(0)^T}\mathbf{U}_{ff}^{(1)^T}\cdots\mathbf{U}_{fb}^{(n-1)^T}\mathbf{U}_{ff}^{(n)^T}\right) \\
&= UDU^T
\end{aligned}$$

where D is a diagonal matrix with diagonal elements

$$diag(D) = \begin{pmatrix} \mathbf{J}_{ff}^{(n)} & \mathbf{J}_{fb}^{(n-1)} & \mathbf{J}_{ff}^{(n-1)} & \mathbf{J}_{fb}^{(n-2)} & \cdots & \mathbf{J}_{ff}^{(1)} & \mathbf{J}_{fb}^{(0)} \end{pmatrix}$$

and U is a upper triangular matrix with unit diagonal. Let us take a closer look on the matrix U. U is a product of matrices of the form

$$\mathbf{U}_x^{(k)} = \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & -\hat{\theta}_x^{(k)} \\ 0 & 0 & I \end{pmatrix},$$

where x denotes ff or fb and k the model order. The product of such matrices is then

$$\begin{aligned}
U &= \mathbf{U}_{ff}^{(n)}\cdot\mathbf{U}_{fb}^{(n-1)}\cdots\mathbf{U}_{ff}^{(1)}\cdot\mathbf{U}_{fb}^{(0)} \\
&= \begin{pmatrix} 1 & -\hat{\theta}_{ff}^{(n)}(t)^T \\ 0 & I_{2n} \end{pmatrix}\cdot\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -\hat{\theta}_{fb}^{(n-1)}(t)^T \\ 0 & 0 & I_{2n-1} \end{pmatrix}\cdot\ \cdots\ \cdot\begin{pmatrix} I_{2n-1} & 0 & 0 \\ 0 & 1 & -\hat{\theta}_{fb}^{(0)}(t)^T \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

This can be written as

$$U = (I - e_1\underline{\vartheta}_1^T)\cdot(I - e_2\underline{\vartheta}_2^T)\cdots(I - e_{2n-1}\underline{\vartheta}_{2n-1}^T)\cdot(I - e_{2n}\underline{\vartheta}_{2n}^T)$$

where e_i is a vector which is one in position i, zero otherwise,

$$e_i = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}^T,$$

and $\underline{\vartheta}_i$ are augmented parameter-vector with the first i elements zero,

$$\underline{\vartheta}_i = \begin{pmatrix} 0 & \cdots & 0 & 0 & \hat{\theta}^{(\cdot)}_{ff/fb}(t)^T \end{pmatrix}^T.$$

Direct multiplication of the terms will mix up the parameter-vectors. We will instead investigate U^{-T}, and show that this problem is avoided then.
The following theorems are useful.

Theorem C.2 ()
$(ABC)^{-T} = A^{-T}B^{-T}C^{-T}$
Proof.
$(ABC)^{-T} = ((ABC)^{-1})^T = (C^{-1}B^{-1}A^{-1})^T = A^{-T}B^{-T}C^{-T}$.

Theorem C.3 ()
$(I - e_i\underline{\vartheta}_i^T)^{-1} = I + e_i\underline{\vartheta}_i^T$
Proof.
$(I - e_i\underline{\vartheta}_i^T)\cdot(I + e_i\underline{\vartheta}_i^T) = I + e_i\underline{\vartheta}_i^T - e_i\underline{\vartheta}_i^T - e_i\underline{\vartheta}_i^T \cdot e_i\underline{\vartheta}_i^T = I$
since $\underline{\vartheta}_i^T e_i = 0$.

By use of these two theorems, U^{-T} can be written as

$$\begin{aligned} U^{-T} &= (I - e_1\underline{\vartheta}_1^T)^{-T}\cdot(I - e_2\underline{\vartheta}_2^T)^{-T}\cdots(I - e_{2n-1}\underline{\vartheta}_{2n-1}^T)^{-T}\cdot(I - e_{2n}\underline{\vartheta}_{2n}^T)^{-T} \\ &= (I + \underline{\vartheta}_1 e_1^T)\cdot(I + \underline{\vartheta}_2 e_2^T)\cdots(I + \underline{\vartheta}_{2n-1}e_{2n-1}^T)\cdot(I + \underline{\vartheta}_{2n}e_{2n}^T). \end{aligned}$$

We now multiply the factors starting from the left

$$\begin{aligned} (I + \underline{\vartheta}_1 e_1^T)\cdot(I + \underline{\vartheta}_2 e_2^T) &= I + \underline{\vartheta}_1 e_1^T + \underline{\vartheta}_2 e_2^T + \underline{\vartheta}_1 e_1^T \underline{\vartheta}_2 e_2^T \\ &= I + \underline{\vartheta}_1 e_1^T + \underline{\vartheta}_2 e_2^T \end{aligned}$$

where the last equality follows from that $e_1^T\underline{\vartheta}_2 = 0$.
Continued multiplication leads to

$$U^{-T} = I + \underline{\vartheta}_1 e_1^T + \underline{\vartheta}_2 e_2^T + \cdots + \underline{\vartheta}_{2n-1}e_{2n-1}^T + \underline{\vartheta}_{2n}e_{2n}^T$$

since $e_i^T\underline{\vartheta}_j = 0$, $\forall j > i$.
Thus we have that U^{-T} is

$$\begin{aligned} U^{-T} &= I + \begin{pmatrix} \underline{\vartheta}_1 & \underline{\vartheta}_2 & \cdots & \underline{\vartheta}_{2n-1} & \underline{\vartheta}_{2n} & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & 1 & 0 & \cdots & 0 & 0 & 0 \\ \hat{\theta}^{(n)}_{ff} & \vdots & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \hat{\theta}^{(n-1)}_{fb} & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \hat{\theta}^{(n-1)}_{ff} & \cdots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \hat{\theta}^{(1)}_{ff} & 1 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \hat{\theta}^{(0)}_{fb} & 1 \end{pmatrix} \end{aligned}$$

The model parameter estimates for all model orders can now be read out of U^{-T} as the columns under the diagonal.

If the estimation is done in a recursive manner, it is practical to work with the inverse of the augmented information matrix,

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

$C(t)$ can be interpreted as the covariance matrix of the augmented parameter-vector. $C(t)$ can be factorised as

$$C(t) = S^{-1}(t) = (UDU^T)^{-1} = U^{-T}D^{-1}U^{-1} = LD^{'}L^T, \tag{C.5}$$

where $D^{'} = D^{-1}$ and $L = U^{-T}$. It means that all parameter estimates are found in L from a LDL^T factorisation of $C(t)$, and all loss function in the inverse of the diagonal matrix D. This is what is going to be used in the recursive version of MMLS. When the MMLS is used in an off-line manner, it is better to do an UDU^T factorisation of S and then inverse of U^T. U^T is always invertible which might not be the case with S.

Appendix D

Recursive LDL^T Updating Algorithm

The goal of this section is to derive an algorithm that update the LDL^T factorisation of the augmented covariance matrix, C,

$$\begin{aligned} L(t) &= f(L(t-1), D(t-1), \underline{\varphi}(t)), \\ D(t) &= g(L(t-1), D(t-1), \underline{\varphi}(t)), \end{aligned}$$

in a way that takes advantage of that we have the LDL^T factorisation of $C(t-1)$. Such an algorithm can be found in [1]. Here, we will present that algorithm in a bit different way.

The basic idea is to represent the extended covariance matrix in the following two ways:

1. A decomposition containing the old parameter matrix, $L(t-1)$, the old diagonal matrix, $D(t-1)$, and the new data vector, $\varphi(t)$.
2. A decomposition containing the new parameter matrix, $L(t)$, and the new diagonal matrix, $D(t)$.

The parameter updating is then achieved by a transformation from the first form to the second decomposition.

Let the augmented parameter-vector, θ, be Gaussian $N(\theta_0, C)$. Assume that a linear observation is made, and that this satisfies

$$z(t) = \varphi^T(t)\theta + e(t)$$

where e is normal $N(0, \sigma^2)$. Then joint covariance-matrix of the augmented variable, $z(t)$, and the old parameter vector, $\theta(t-1)$, is then

$$\begin{aligned} R(t) &= Cov \begin{pmatrix} z(t) \\ \theta(t-1) \end{pmatrix} = E\left\{ \begin{pmatrix} \tilde{z}(t) \\ \tilde{\theta}(t-1) \end{pmatrix} \begin{pmatrix} \tilde{z}(t) & \tilde{\theta}^T(t-1) \end{pmatrix} \right\} \\ &= \begin{pmatrix} E\left\{\tilde{z}(t)^2\right\} & E\left\{\tilde{z}(t)\tilde{\theta}^T(t-1)\right\} \\ E\left\{\tilde{\theta}(t-1)\tilde{z}(t)\right\} & E\left\{\tilde{\theta}(t-1)\tilde{\theta}^T(t-1)\right\} \end{pmatrix} \end{aligned} \tag{D.1}$$

where $\tilde{z}(t) = z(t) - E\{z(t)\}$ and $\tilde{\theta}(t-1) = \theta(t-1) - E\{\theta\}$. The covariance matrices are given by:

$$\begin{aligned}
E\left\{\tilde{z}(t)\tilde{z}^T(t)\right\} &= E\left\{\left(\varphi^T(t)\tilde{\theta}(t-1) + e(t)\right)\left(\varphi^T(t)\tilde{\theta}(t-1) + e(t)\right)^T\right\} \\
&= \varphi^T(t)E\left\{\tilde{\theta}(t-1)\tilde{\theta}^T(t-1)\right\}\varphi(t) + E\left\{e(t)e^T(t)\right\} \\
&= \varphi^T(t)C(t-1)\varphi(t) + \sigma^2(t) && \text{(D.2)} \\
E\left\{\tilde{z}(t)\tilde{\theta}^T(t-1)\right\} &= E\left\{\left(\varphi^T(t)\tilde{\theta}(t-1) + e(t)\right)\tilde{\theta}^T(t-1)\right\} \\
&= \varphi^T(t)E\left\{\tilde{\theta}(t-1)\tilde{\theta}^T(t-1)\right\} \\
&= \varphi^T(t)C(t-1) && \text{(D.3)} \\
E\left\{\tilde{\theta}(t-1)\tilde{z}^T(t)\right\} &= E\left\{\tilde{z}(t)\tilde{\theta}^T(t-1)\right\}^T \\
&= C(t-1)\varphi(t) && \text{(D.4)} \\
E\left\{\tilde{\theta}(t-1)\tilde{\theta}^T(t-1)\right\} &= C(t-1) && \text{(D.5)}
\end{aligned}$$

The results follow from the assumption that $\theta(t-1)$ and $e(t)$ are independent.

Decomposition 1

Insert (D.2)-(D.5) into (D.1) and the joint covariance matrix becomes

$$R(t) = \begin{pmatrix} \varphi^T(t)C(t-1)\varphi(t) + \sigma^2(t) & \varphi^T(t)C(t-1) \\ C(t-1)\varphi(t) & C(t-1) \end{pmatrix} \tag{D.6}$$

We have a factorisation of $C(t-1)$ given by

$$C(t-1) = L(t-1)D(t-1)L^T(t-1) \tag{D.7}$$

Insert (D.7) into (D.6) and we have

$$R(t) = \begin{pmatrix} \varphi^T(t)L(t-1)D(t-1)L^T(t-1)\varphi(t) + \sigma^2(t) & \varphi^T(t)L(t-1)D(t-1)L^T(t-1) \\ L(t-1)D(t-1)L^T(t-1)\varphi(t) & L(t-1)D(t-1)L^T(t-1) \end{pmatrix} \tag{D.8}$$

It is now straight forward to partition (D.8) into

$$R(t) = \begin{pmatrix} 1 & \varphi^T(t)L(t-1) \\ 0 & L(t-1) \end{pmatrix} \begin{pmatrix} \sigma^2(t) & 0 \\ 0 & D(t-1) \end{pmatrix} \begin{pmatrix} 1 & \varphi^T(t)L(t-1) \\ 0 & L(t-1) \end{pmatrix}^T \tag{D.9}$$

Decomposition 2

The updated augmented covariance matrix, $C(t)$, is equal to the conditional covariance of $\theta(t-1)$ given $z(t)$,

$$C(t) = cov\{\theta(t)\} = cov\{\theta(t-1)|z(t)\} \tag{D.10}$$

This can be seen as the new covariance of θ when all information from the latest augmented variable, $z(t)$, have been absorbed. The variable $\theta(t-1)|z(t)$ fulfils the property that it is independent of $z(t)$. This will now be used to achieve an equation for $C(t)$. It can be shown that $R(t)$ can be factorised as

$$R(t) = \begin{pmatrix} 1 & 0 \\ K(t) & L(t) \end{pmatrix} \begin{pmatrix} \sigma^2_{z(t)} & 0 \\ 0 & D(t) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ K(t) & L(t) \end{pmatrix}^T \tag{D.11}$$

Transformation

It now remains to find the appropriate transformation. A convenient method is the Dyadic decomposition.
The matrix decompositions defined above are very simular. The main difference between them are the first row and column of the left and right matrix factors. The first decomposition has nonzero first row and zero first column, while the second decomposition has the opposite.

Matrix multiplication of three matrices can be written as

$$ADA^T = \sum_{i=1}^{n} d_i A(:,i)A(:,i)^T. \tag{D.12}$$

This can be seen as the sum of n rank 1 matrices. Split the sum into the first rank 1 matrix and the sum of the other rank 1 matrices,

$$ADA^T = d_1 A(:,1)A(:,1)^T + \sum_{i=2}^{n} d_i A(:,i)A(:,i)^T. \tag{D.13}$$

But this is not a unique representation. The matrix product ADA^T can also be represented as a sum of some other rank 1 matrices,

$$ADA^T = \tilde{d}_1 \tilde{A}(:,1)\tilde{A}(:,1)^T + \sum_{i=2}^{n} \tilde{d}_i \tilde{A}(:,i)\tilde{A}(:,i)^T. \tag{D.14}$$

The idea is now to step by step transform ADA^T from representation (D.13) to representation (D.14). At each step two rank 1 matrices are transformed to two other rank 1 matrices, having the desired properties.

Given vectors

$$A(1,:) = \begin{pmatrix} 1 & A(2,1) & \cdots & A(n,1) \end{pmatrix}^T \tag{D.15}$$

and

$$A(k,:) = \begin{pmatrix} A(1,k) & A(2,k) & \cdots & A(n,k) \end{pmatrix}^T \tag{D.16}$$

and scalars d_1 and d_k. The Dyadic decomposition problem is then defined as the problem of finding new vectors,

$$\tilde{A}(1,:) = \begin{pmatrix} 1 & \tilde{A}(2,1) & \cdots & \tilde{A}(n,1) \end{pmatrix}^T \tag{D.17}$$

and

$$\tilde{A}(k,:) = \begin{pmatrix} 0 & \tilde{A}(2,k) & \cdots & \tilde{A}(n,k) \end{pmatrix}^T \tag{D.18}$$

and scalars $\tilde{d}_1$ and $\tilde{d}_k$ that satisfies

$$d_1 A(:,1)A(:,1)^T + d_k A(:,k)A(:,k)^T = \tilde{d}_1 \tilde{A}(:,1)\tilde{A}(:,1)^T + \tilde{d}_k \tilde{A}(:,k)\tilde{A}(:,k)^T \quad \text{(D.19)}$$

That is, to find two other rank 1 matrices that does not change the matrix product, and have the desired properties.

There are several solutions to the Dyadic decomposition problem. A simple one is

$$\begin{aligned} \tilde{d}_1 &= d_1 + d_k A(1,k)^2 && \text{(D.20)} \\ \tilde{d}_k &= \frac{d_1 d_k}{\tilde{d}_1} && \text{(D.21)} \\ \tilde{A}(:,k) &= A(:,k) - A(1,k)A(:,1) && \text{(D.22)} \\ \tilde{A}(:,1) &= A(:,1) + \frac{d_k A(1,k)}{\tilde{d}_1} && \text{(D.23)} \end{aligned}$$

The full transformation from representation (D.13) to (D.14) is obtained through repeated use of this Dyadic decomposition.

Summary

The L and D matrices can be updated as follows:

1. Represent the old parameter matrix and diagonal matrix, $L(t-1)$ and $D(t-1)$, and the new data vector $\varphi(t)$ at the form (D.9).

2. Transform this to the form of (D.11) by n Dyadic decompositions, starting from the end.

3. Extract the new parameter matrix, $L(t)$, and diagonal matrix, $D(t)$, from the transformed matrices according to form (D.11).

Bibliography

[1] K. J. Åström and B. Wittenmark. *Adaptive Control.* Addison-Wesley Publishing Company, 2 edition, 1995.

[2] K. J. Åström and B. Wittenmark. *Computer-Controlled Systems.* Prentice Hall Inc., 3 edition, 1997.

[3] Stephen Barnett. *Polynomials and linear control systems.* Marcel Dekker Inc., 1983.

[4] H. Fritz. Autonomous lateral road vehicle guidance using neural network controllers. In *Proc. 3rd European Control Conference, Rome, Italy*, pages 2748–2753, 1995.

[5] K. J. Hunt. *Stochastic Optimal Control Theory with Application in Self-tuning Control*, volume 117 of *Lecture Notes in Control and Information Sciences.* Springer-Verlag, Berlin, 1989.

[6] K. J. Hunt and T. A. Johansen. Design and analysis of gain-scheduled control using local controller networks. *International Journal of Control*, (5):619–651, 1997.

[7] K. J. Hunt, J. C. Kalkkuhl, H. Fritz, and T. A. Johansen. Constructive empirical modelling of longitudinal vehicle dynamics using local model networks. *Control Engineering Practice*, 4(2):167–178, February 1996.

[8] T. A. Johansen. *Operating regime based process modeling and identification.* PhD thesis, Department of Engineering Cymbernetics, Norwegian Institute of Technology, University of Trondheim, 1994.

[9] T. A. Johansen and B. A. Foss. A NARMAX model representation for adaptive control based on local models. *Modeling, Identification and Control*, (13):25–39, 1992.

[10] T. A. Johansen and B. A. Foss. Constructing NARMAX models using ARMAX models. *International Journal of Control*, (58):1125–1153, 1993.

[11] R Johansson. *System Modeling and Identification.* Prentice Hall Inc., 1993.

[12] J. Kalkkuhl, K. J. Hunt, and H. Fritz. FEM-based Neural Network Approach to Nonlinear Modelling with Application to Longitudinal Vehicle Dynamics. Technical report, Daimler-Benz Research and Technology, 1997.

[13] L. Ljung. *System Identification, theory for the user.* Prentice Hall Inc., 1987.

[14] R. Murray-Smith and T. A. Johansen. Local Learning in Local Model Networks. Technical report, Daimler-Benz Research and Technology, Berlin and SINTEF Automatic Control Trondheim, Norway, 1994.

[15] K. S. Narendra and J. Balakrishnan. Intelligent Control using Fixed and Adaptive Models. Technical report, Center for Systems Science, Dept. of Electrical Engineering, Yale University, 1994.

[16] Dietmar Neumerkel and Robert Shorten. Robust Learning Algorithms for Nonlinear Filtering. Technical report, Daimler-Benz research Berlin, 1996.

[17] S. Niu and L. Ljung. Multiple Model Parameter Estimation. Technical report, Department of Electrical Engineering, Linköping University, 1994.

[18] H. Wang, G.P. Liu, C.J. Harris, and M. Brown. *Advanced Adaptive Control.* Pergamon, 1995.