ISSN 0280-5316
ISRN LUTFD2/TFRT--5706--SE

Subspace Based Identification for Adaptive Control

Brad Schofield

Department of Automatic Control
Lund Institute of Technology
June 2003

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* June 2003 *Document Number* ISRN LUTFD2/TFRT--5706--SE
Author(s) Brad Schofield	*Supervisor* Rolf Johansson at LTH *Sponsoring organization*

Title and subtitle
Subspace Based Identification for Adaptive Control (Identifiering med underrumsmetoder för adaptive reglering)

Abstract
This thesis describes the results of an investigation into the use of Subspace-based System Identification techniques in the field of Adaptive Control. The project aims included adaption of subspace algorithms for
online use, proposal of new algorithms of adaptive control based on these algorithms, and analysis of properties of the proposed systems. The thesis includes reviews of various subspace algorithms used in the investigation, as well as a brief review of cur-rent adaptive control algorithms. A direct adaptive control algorithm based on a combination of subspace methods and model predictive control techniques, as well as an indirect algorithm combining an implemented online subspace identification algorithm and a model predictive controller, are presented.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 104	*Recipient's notes*
Security classification		

Subspace Based Identification for Adaptive Control

Brad Schofield

June 16, 2003

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

This project deals with two fields of systems and control theory which are intimately related, namely *Adaptive Control* and *System Identification*. Adaptive control is a term applied to any control system which has an ability to automatically adjust its parameters. This may be caused by the output of some identification element within the overall system, or by the variation of some performance measure. System identification is a term used for the process of obtaining a mathematical model from some form of measured data. This could be input-output data or frequency response information. There exists an extremely wide variety of techniques for performing system identification, all of which require differing levels of knowledge about the system in question.

Traditional approaches to adaptive control involve gradient search methods. Where an unknown system is to be identified, it is common to use a least-squares approach to identify parameters in an assumed model, which may then be used to design a controller. Such an approach works well in the case where a model structure can be specified, and for single-input, single-output (SISO) systems. However, when many inputs and outputs are present (MIMO systems), and the system is of a relatively high order, it becomes difficult not only to postulate a model structure, but also to solve the identification problem.

With regard to these problems it is clear that a method of system identification that requires a minimum of prior information on order and model structure, but can provide a useful mathematical model, is highly desirable. The class of system identification algorithms known as *Subspace Methods* are able to provide state-space representations of unknown systems from input-output data and a minimum of prior identification. They are block based methods, i.e. they operate on a given number of input-output samples, and they may be applied to MIMO systems as easily as SISO systems. By using modern linear algebra techniques such as the QR and singular value decompositions, their computational complexity is such that they are easily implementable.

1.2 Aims

In the light of the above it would clearly be desirable to devise some form of adaptive control strategy that utilizes subspace algorithms to perform system identification. This is the underlying aim of the project. Since the work constitutes in some way a new field of adaptive control it was decided not to place emphasis on solving a particular control task, but to propose algorithms and system structures which may subsequently be investigated further. Nevertheless it is always useful to investigate real world systems, and to this end such a system, the Furuta Inverted Pendulum, was used throughout the project.

The aims of the project may be specified more clearly as

- Investigation of online implementation of subspace algorithms
- Proposal of algorithms and system structures for an adaptive control system utilizing subspace methods of identification
- Investigation of the properties of such a system, including performance and computational complexity

These aims correspond at least approximately to the stages in which work on the project was completed.

1.3 Description of Work Undertaken

The work involved much theoretical research into the subspace identification algorithms, from sources such as [22]. Adaptive control was also reviewed, the main reference being [20]. Various other aspects of control and system identification were investigated throughout the project, including LQ control design, Model Predictive Control, and statistical significance testing. Practical work was carried out in the Matlab/Simulink environment, used for implementation of the algorithms and simulation of the systems. Experiments on real processes (namely the Furuta inverted pendulum) were also carried out in Matlab, via DAC/ADC interfaces to the process.

1.4 Structure of the Report

Since the particulars of subspace identification may not be known to the reader this report begins with a moderately in-depth overview of the subspace algorithms in Part I. They are by no means fully proved, but outlines of the derivations are provided. The sources are stated for those readers interested in gaining more knowledge on any particular algorithm. The part begins with an introduction to state-space identification, which is a starting point for the development of subspace methods. The presentation of subspace algorithms is split according to type, with one chapter describing the so-called ‘projection’ algorithms and another describing ‘intersection’ algorithms. For those who already have a knowledge of the subspace methods, or are more interested in control aspects, this section may be skipped.

In the same way, a very brief overview of the state of the art of adaptive control is provided in Part II. This serves as a useful reference for comparing proposed subspace-adaptive control algorithms with existing ones. It is also intended to give the reader who has not encountered adaptive control previously an insight into the structures and algorithms which currently exist, and perhaps to indicate some of their limitations. An overview of Model Predictive Control is presented in Chapter 6, since ideas and techniques from MPC will be called upon often in Part III. Like Part I, Part II may be skipped by those familiar with adaptive control.

Part III deals with the results obtained during the project, and is structured in accordance with the stated project aims. It begins with the methods of implementation and experimental results of online implementations of subspace algorithms, including tests on the Furuta Inverted Pendulum mentioned earlier. The next chapter deals with the proposed adaptive algorithms and system structures. Chapter 9 introduces a direct adaptive control strategy based on ideas drawn from subspace identification and model predictive control. This strategy was implemented in a Simulink controller block, and sucessfully tested in a simple example. In the conclusion, the proposed algorithms are summarised, and the areas deemed most important and most promising for future work are highlighted. Conclusions are drawn about the applicability of the proposed algorithms, and some qualitative comparisons are made between them and existing algorithms.

In the Appendices, some of the more important mathematical tools used in the thesis are briefly described, including projections and matrix factorizations. Also included in the appendices are descriptions and source code of the Matlab functions and Simulink m-files written to support the project.

1.5 Previous Work

The work presented in this thesis is based upon the work done on subspace identification by Verhaegen, De Moor, van Overshee, Moonen, Favoureel and many others. The sources include [26], [27], [3], [14] and [6]. The adaptive control background was provided by [20], by Åström and Wittenmark.

1.6 Acknowledgements

Technical help and direction was provided by Dr Rolf Johansson (project supervisor) and Johan Åkesson (PhD student) at the Department of Automatic Control, Lund Institute of Technology. The MPC tools utilized in Chapter 8 were also written by Johan Åkesson.

Thanks are also due to Dr Philippe De Wilde at the Department of Electrical and Electronic Engineering, Imperial College, and Christina Grossmann and colleagues at the International Office, LTH, for coordinating the exchange agreement.

Part I

Subspace Methods

Chapter 2

State Space Identification

Current system identification techniques used in adaptive control are of the gradient-search type, in which parameters of a predetermined model structure are estimated. For single-variable systems, and for systems in which the process structure is known, this approach works adequately. However, for multivariable systems, and those with structures that are unknown or complex, the tasks of both proposing a model structure and of finding a parameter set are complicated.

In the light of this, a method of identifying a mathematical model without the requirement of specifying a model order or structure is desirable. More specifically, since state-space models are a standard representation of dynamical systems, it is desirable to identify such a model from input-output data. Such a method, based on the multivariable transfer function of Markov parameters, was proposed by Ho and Kalman, and modified by Juang and Pappa. The latter algorithm is outlined in the following section.

2.1 Ho and Kalman Realization Algorithm

Consider the aforementioned multivariable transfer function:

$$H(z) = \sum_{k=0}^{\infty} H_k z^{-k} \tag{2.1}$$

where $\{H_k\}$ are the Markov parameters, i.e. matrices of impulse response coefficients. Consider also a general state-space system of the form:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.2}$$
$$y_k = Cx_k + Du_k + e_k \tag{2.3}$$

with $u_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}^l$, state $x_k \in \mathbb{R}^n$, dynamics noise sequence $w_k \in \mathbb{R}^n$ and output noise sequence $e_k \in \mathbb{R}^l$.

The aim is to identify the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$, $D \in \mathbb{R}^{l\times m}$, using the finite input-output data sequences $\{u_k\}_{k=1}^{N}$ and $\{y_k\}_{k=1}^{N}$.

The Markov parameters $\{H_k\}$ are given by:

$$H_k = CA^{k-1}B, \quad k = 1, 2, 3, \ldots, N \tag{2.4}$$

From the Markov parameters, form the Hankel matrix:

$$\mathcal{H}_{k,i,j} = \begin{pmatrix} H_{k+1} & H_{k+2} & \dots & H_{k+j} \\ H_{k+2} & H_{k+3} & \dots & H_{k+j+1} \\ \vdots & \vdots & \ddots & \vdots \\ H_{k+i} & H_{k+i+1} & \dots & H_{k+i+j-1} \end{pmatrix} \tag{2.5}$$

The modified algorithm by Juang and Pappa uses two such Hankel matrices, with $k = 0$ and $k = 1$:

$$\mathcal{H}_{0,i,j} = \begin{pmatrix} H_1 & H_2 & \dots & H_j \\ H_2 & H_3 & \dots & H_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ H_i & H_{i+1} & \dots & H_{i+j-1} \end{pmatrix} \tag{2.6}$$

$$\mathcal{H}_{1,i,j} = \begin{pmatrix} H_2 & H_3 & \dots & H_{j+1} \\ H_3 & H_4 & \dots & H_{j+2} \\ \vdots & \vdots & \ddots & \vdots \\ H_{i+1} & H_{i+2} & \dots & H_{i+j} \end{pmatrix} \tag{2.7}$$

The next step involves the calculation of the singular value decomposition of the Hankel matrix $\mathcal{H}_{0,i,j}$:

$$\mathcal{H}_{0,i,j} = U\Sigma V^T$$

A balanced realization of order n, input dimension m and output dimension l:

$$\begin{aligned} x(k+1) &= A_n x(k) + B_n u(k) \\ y(k) &= C_n x(k) + D_n u(k) \end{aligned}$$

is obtained, where the system matrices are obtained from:

$$\begin{aligned} A_n &= \Sigma_n^{-1/2} U_n^T \mathcal{H}_{1,i,j} V_n \Sigma_n^{-1/2} & \qquad B_n &= \Sigma_n^{-1/2} V_n^T E_u \\ C_n &= E_y^T U_n \Sigma_n^{-1/2} & \qquad D_n &= H_0 \end{aligned}$$

where the matrices E_u E_y are given by:

$$\begin{aligned} E_y^T &= [I_{l\times l} \; 0_{l\times(i-1)l}] \\ E_u^T &= [I_{m\times m} \; 0_{m\times(j-1)m}] \end{aligned}$$

The matrices U_n, V_n and Σ_n are given by:

$$\Sigma_n = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{pmatrix}$$

$$\begin{aligned} U_n &= U_{\text{first } n \text{ columns}} \\ V_n &= V_{\text{first } n \text{ columns}} \end{aligned}$$

This realization algorithm is based on the Markov parameters of the system, that is, the system's impulse response. While the practical significance is low, since the impulse response is unlikely to be available, the real importance of this algorithm is that it represents a special case of a subspace algorithm, namely one in which the input is an impulse.

Chapter 3

Projection Algorithms

3.1 Introduction

Subspace algorithms may be split into two classes, namely *projection algorithms* and *intersection algorithms*. These names derive from the methods used to obtain the system matrices from the input-output data organised in Hankel matrices. In this chapter the projection algorithms are considered. The following chapter deals with intersection algorithms.

The concept of the projection algorithms is fairly straightforward, stemming from the multiplication of the equation:

$$Y_f = \Gamma_i X_f + H_i U_f \tag{3.1}$$

with the projection operator $\Pi_{U_f^\perp}$. This yields the projection:

$$Y_f / U_f^\perp = \Gamma_i X_f / U_f^\perp \tag{3.2}$$

From Equation 3.2 an estimate of the column space of the extended observability matrix may be obtained, whose rank gives the system order.

The class of algorithms known as MIMO Output-Error State-sPace (MOESP) is important, since a number of variations of the basic algorithm exist which allow identification of several types of identification problems such as the noise-free case, white ouput noise, the output-error model and the innovations model. This chapter will concentrate on this class of algorithms, since they will be referred to and utilized in later chapters.

The first algorithm to be illustrated is the basic MOESP, used for noise-free identification. This derivation will also serve to outline the notation used in discussions of subspace methods throughout this thesis. The PIMOESP algorithm, for use with output-error noise model identification, and the POMOESP algorithm used for innovations model identification will also be described. The POMOESP algorithm is particularly important here since it is the one used in later chapters, for the implementation of adaptive controllers. It should be noted that the full proofs are not given here; they are in general rather involved and are not required within the scope of the thesis. Proofs may be found in [6], [26] and [22].

3.2 The Basic MOESP Algorithm

The basic MOESP formulation is derived for the noise-free identification problem, that is, it is purely a deterministic algorithm. While this is of course restrictive (any real-world process to be identified will be contaminated with noise), it is of value not only because it is required as a starting point for the other algorithms in its class, but also because it illustrates in a simple way the principles of projection algorithms.

Problem Formulation 3.2.1. *Given the input-output data sets $\{u_k\}_{k=1}^N$ with $u_k \in \mathbb{R}^m$ and $\{y_k\}_{k=1}^N$ with $y_k \in \mathbb{R}^l$ of an unknown system of order n given by:*

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{3.3}$$

Determine:

- *the order n of the system*
- *the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$, $D \in \mathbb{R}^{l\times m}$ up to a similarity transformation*

The first stage is the construction of the Hankel matrices from input-output data:

$$Y_{k,i,j} = \begin{pmatrix} y_k & y_{k+1} & \cdots & y_{k+j-1} \\ y_k & y_{k+2} & \cdots & y_{k+j} \\ \vdots & \vdots & \ddots & \vdots \\ y_{k+i-1} & y_{k+i} & \cdots & y_{k+j+i-2} \end{pmatrix} \tag{3.4}$$

$$U_{k,i,j} = \begin{pmatrix} u_k & u_{k+1} & \cdots & u_{k+j-1} \\ u_k & u_{k+2} & \cdots & u_{k+j} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k+i-1} & u_{k+i} & \cdots & u_{k+j+i-2} \end{pmatrix} \tag{3.5}$$

Construct the matrix $\Gamma_i \in \mathbb{R}^{li\times n}$:

$$\Gamma_i = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{i-1} \end{pmatrix}$$

If $i > n$, Γ_i has an interpretation as an extended observability matrix. If the matrix pair A, C is assumed to be observable, then it follows that $rank(\Gamma_i) = n$.

Construct also the lower block triangular Toeplitz matrix of Markov parameters H_i:

$$H_i = \begin{pmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ CA^{i-2}B & CA^{i-3} & CA^{i-4} & \dots & D \end{pmatrix}$$

Introduce the state vector:

$$X_{k,j} = \begin{pmatrix} x_k & x_{k+1} & \dots & x_{k+j-1} \end{pmatrix} \tag{3.6}$$

Using the notation defined above it is possible to express input-output data in the form:

$$Y_{k,i,j} = \Gamma_i X_{k,j} + H_i U_{k,i,j} \tag{3.7}$$

The method of operation of the MOESP algorithm is to obtain an estimate of the extended observability matrix Γ_i, and from this determine estimates of the system matrices A and C. The estimate of Γ_i is obtained by removing the $H_i U_{k,i,j}$ term from Equation 3.7, and determining the column space of the result.

This method is justified by considering that the future outputs of a system are given by the combination of the vector spaces of the state (which reflects the past inputs and outputs of the system) and the future inputs. By removing the $H_i U_{k,i,j}$ term from Equation 3.7, the algorithm determines the effect of the state on the output.

To remove the term $H_i U_{k,i,j}$ in Equation 3.7, consider multiplication by the projection operator onto the orthogonal complement of $U_{k,i,j}$, denoted $\Pi_{U^{\perp}_{k,i,j}}$ (see Appendix A.1.1). This yields:

$$Y_{k,i,j} \Pi_{U^{\perp}_{k,i,j}} = \Gamma_i X_{k,j} \Pi_{U^{\perp}_{k,i,j}}$$

This may be expressed as the orthogonal projection:

$$Y_{k,i,j} / U^{\perp}_{k,i,j} = \Gamma_i X_{k,j} / U^{\perp}_{k,i,j}$$

As shown in Appendix A.2, this projection may be solved by employing the RQ decomposition:

$$\begin{pmatrix} U_{0,i,j} \\ Y_{0,i,j} \end{pmatrix} = \begin{pmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} \tag{3.8}$$

From Equation 3.8, we obtain:

$$\begin{aligned} U_{0,i,j} &= R_{11} Q_1 \\ Y_{0,i,j} &= R_{21} Q_1 + R_{22} Q_2 \end{aligned} \tag{3.9}$$

and from Equation 3.7, we obtain:

$$Y_{0,i,j} = \Gamma_i X_{0,j} + H_i R_{11} Q_1 \tag{3.10}$$

Multiplying the expression for $Y_{0,i,j}$ in Equation 3.9 with Q_2^T, we get:

$$Y_{0,i,j} Q_2^T = R_{22}$$

Performing a similar multiplication from the right on Equation 3.10 yields:

$$Y_{0,i,j} Q_2^T = \Gamma_i X_{0,j} Q_2^T$$

This gives the result:

$$R_{22} = \Gamma_i X_{0,j} Q_2^T$$

It remains to obtain the column space of the extended observability matrix Γ_i. As illustrated in Appendix A.3, this may be found by taking the SVD of R_{22}:

$$R_{22} = U \Sigma V^T$$

In this noise-free case, the number of nonzero singular values gives the system order n. A basis in the column space of the extended observability matrix Γ_i is given by the first n columns of the matrix U. Estimates of the system matrices C and A (up to a similarity transformation) are obtained as:

$$\begin{aligned}\hat{C} &= U_{n(\text{first } l \text{ rows})} \\ \hat{A} &= U_1^{\dagger} U_2\end{aligned}$$

where:

$$\begin{aligned}U_1 &= U_{n(\text{first } (i-1) \times l \text{ rows})} \\ U_2 &= U_{n(\text{last } (i-1) \times l \text{ rows})}\end{aligned}$$

The methods for obtaining the B and D matrices are not so straightforward and will not be discussed here.

It can be shown that this basic algorithm also provides consistent estimates in the presence of white output noise (see for example [6]). In this case the singular values above the order of the system are nonzero, and the 'large' singular values must be chosen in order to obtain the system order and estimate the system matrices. This is in general done by inspection of the singular values.

3.3 The PIMOESP Algorithm

As seen in the previous section, the basic MOESP formulation gives consistent estimates in the presence of white output noise. In reality however, the noise encountered will seldom be white. In this section the output-error problem, this is, where coloured output noise is present.

Problem Formulation 3.3.1. *Given the input-output data sets $\{u_k\}_{k=1}^N$ with $u_k \in \mathbb{R}^m$ and $\{y_k\}_{k=1}^N$ with $y_k \in \mathbb{R}^l$ of an unknown system of order n given by:*

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k + v(k)\end{aligned} \tag{3.11}$$

where $v(k)$ is an additive zero mean noise signal, uncorrelated with the input, determine:

- *the order n of the system*
- *the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$, $D \in \mathbb{R}^{l\times m}$ up to a similarity transformation*

The Past-Input version of the MOESP algorithm, known as PIMOESP, solves this problem using an instrumental variable method. Forming Hankel matrices of appropriate dimensions as in Section 3.2, we obtain the representation:

$$Y_{k,i,j} = \Gamma_i X_{k,j} + H_i U_{k,i,j} + V_{k,i,j} \tag{3.12}$$

where $V_{k,i,j}$ is a Hankel matrix of the noise values.

As with the basic MOESP algorithm, the first step involves the removal of the $H_i U_{k,i,j}$ term through multiplication by the projection operator $\Pi_{U_{k,i,j}^{\perp}}$. To remove the remaining noise term, an instrument $\mathcal{Z}$, correlated with the state but uncorrelated with the noise, will be used. In other words, the matrix $\mathcal{Z}$ must satisfy the conditions:

$$\lim_{N\to\infty} \frac{1}{N} V_{k,i,j} \Pi_{\mathcal{Z}} = 0 \tag{3.13}$$

$$rank(\lim_{N\to\infty} \frac{1}{N} X_{k,j} \Pi_{\mathcal{Z}}) = n \tag{3.14}$$

In the PIMOESP the instrument is chosen to be a matrix of past input values. Since the noise is assumed to be uncorrelated with the input, it is clear that Equation 3.13 holds. Under conditions of persistent excitation, Equation 3.14 also holds.

It is now necessary to define two Hankel matrices for inputs and outputs, one of 'past' values, given by $U_{0,i,j}$, and another of 'future' values, given by $U_{i,2i,j}$ (note that the block row sizes need not be equal, it is merely chosen so in this case). From this the instrument $\mathcal{Z}$ may be defined as:

$$\mathcal{Z} = U_{0,i,j}$$

Multiplying Equation 3.12 (using the 'future' Hankel matrices) from the right with $\Pi_{U_{k,i,j}^{\perp}}$ and $\Pi_{\mathcal{Z}}$ and allowing the number of data points to tend to infinity gives:

$$\lim_{N\to\infty} \frac{1}{N} Y_{i,2i,j} \Pi_{U_{k,i,j}^{\perp}} \Pi_{U_{0,i,j}} = \lim_{N\to\infty} \frac{1}{N} \Gamma_i X_{i,j} \Pi_{U_{k,i,j}^{\perp}} \Pi_{U_{0,i,j}}$$

It is now desired to estimate the column space of Γ_i. This may be done in a similar way to Section 3.2 by using the RQ decomposition:

$$\begin{pmatrix} U_{i,2i,j} \\ \mathcal{Z} \\ Y_{i,2i,j} \end{pmatrix} = \begin{pmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \end{pmatrix}$$

As the number of data points tends to infinity we obtain:

$$\lim_{N\to\infty} \frac{1}{\sqrt{N}} R_{32} = \lim_{N\to\infty} \frac{1}{\sqrt{N}} \Gamma_i X_{i,j} Q_2^T$$

As before, we take the SVD of R_{32}:

$$R_{32} = U \Sigma V^T$$

The system matrices A and C are then calculated in the same manner as with the basic MOESP algorithm, giving:

$$\hat{C} = U_{n(\text{first } l \text{ rows})}$$
$$\hat{A} = U_1^{\dagger} U_2$$

where:

$$U_1 = U_{n(\text{first } (i-1) \times l \text{ rows})}$$
$$U_2 = U_{n(\text{last } (i-1) \times l \text{ rows})}$$

3.4 The POMOESP Algorithm

In this section the problem of identification of the innovations model will be considered. This model involves white output noise in addition to white noise on each of the states, the latter being known as process noise. The Past Output MOESP scheme involves both deterministic and stochastic parts, and is thus useful for control purposes, for example the construction of a one step ahead predictor.

Problem Formulation 3.4.1. *Given the input-output data sets $\{u_k\}_{k=1}^N$ with $u_k \in \mathbb{R}^m$ and $\{y_k\}_{k=1}^N$ with $y_k \in \mathbb{R}^l$ of an unknown system of order n given by:*

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w(k) \\ y_k &= Cx_k + Du_k + v(k) \end{aligned} \tag{3.15}$$

where $v(k)$ and $w(k)$ are additive zero mean white noise signals, uncorrelated with the input, and with covariance matrices:

$$\mathcal{E}\{\begin{pmatrix} w(k) \\ v(k) \end{pmatrix} \begin{pmatrix} w^T(k+\tau) & v^T(k+\tau) \end{pmatrix}\} = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \delta(\tau)$$

Let the system in Equation 3.15 be written on the equivalent innovations form:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ke(k) \\ y_k &= Cx_k + Du_k + e(k) \end{aligned} \tag{3.16}$$

Determine:

- *the order n of the system*
- *the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$, $D \in \mathbb{R}^{l\times m}$ up to a similarity transformation*
- *the Kalman gain K*

Consider the representation:

$$Y_{k,i,j} = \Gamma_i X_{k,j} + H_i U_{k,i,j} + G_i W_{k,i,j} + V_{k,i,j} \tag{3.17}$$

where:

$$G_i = \begin{pmatrix} 0 & 0 & \dots & 0 \\ C & 0 & \dots & 0 \\ CA & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{i-2} & CA^{i-3} & \dots & 0 \end{pmatrix}$$

Again the projection operator $\Pi_{U^\perp_{k,i,j}}$ will be used to remove the $H_i U_{k,i,j}$ term. An instrumental variable will also be used to remove the noise terms. The instrumental variable $U_{0,i,j}$ as used in the PIMOESP method is a possible candidate, since:

$$\lim_{N\to\infty} \frac{1}{N} W_{k,i,j} U^T_{0,i,j} = 0 \tag{3.18}$$

$$\lim_{N\to\infty} \frac{1}{N} V_{k,i,j} U^T_{0,i,j} = 0 \tag{3.19}$$

It can also be shown that:

$$\lim_{N\to\infty} \frac{1}{N} Y_{0,i,j} W_{k,i,j}^T = 0 \tag{3.20}$$

$$\lim_{N\to\infty} \frac{1}{N} Y_{0,i,j} V_{k,i,j}^T = 0 \tag{3.21}$$

The instrumental variable in the POMOESP method is thus chosen as a combination of past inputs and outputs:

$$\mathcal{Z} = \begin{pmatrix} U_{0,i,j} \\ Y_{0,i,j} \end{pmatrix}$$

Multiplying Equation 3.17 from the right by $\Pi_{U_{k,i,j}^{\perp}}$ and $\Pi_{\mathcal{Z}}$ gives:

$$\lim_{N\to\infty} \frac{1}{N} Y_{i,2i,j} \Pi_{U_{k,i,j}^{\perp}} \Pi_{\mathcal{Z}} = \lim_{N\to\infty} \frac{1}{N} \Gamma_i X_{i,j} \Pi_{U_{k,i,j}^{\perp}} \Pi_{\mathcal{Z}}$$

This is solved in the usual way using the RQ decomposition:

$$\begin{pmatrix} U_{i,2i,j} \\ \mathcal{Z} \\ Y_{i,2i,j} \end{pmatrix} = \begin{pmatrix} U_{i,2i,j} \\ U_{0,i,j} \\ Y_{0,i,j} \\ Y_{i,2i,j} \end{pmatrix} = \begin{pmatrix} R_{11} & 0 & 0 & 0 \\ R_{21} & R_{22} & 0 & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ R_{41} & R_{42} & R_{43} & R_{44} \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{pmatrix}$$

Letting the number of data points tend to infinity we obtain:

$$\lim_{N\to\infty} \frac{1}{\sqrt{N}} \begin{pmatrix} R_{42} & R_{43} \end{pmatrix} = \lim_{N\to\infty} \frac{1}{\sqrt{N}} \Gamma_i X_{i,j} \begin{pmatrix} Q_2 \\ Q3 \end{pmatrix}^T$$

To find an estimate of the column space of Γ_i, we take the SVD of $\begin{pmatrix} R_{42} & R_{43} \end{pmatrix}$:

$$\begin{pmatrix} R_{42} & R_{43} \end{pmatrix} = U \Sigma V^T$$

The system matrices A and C are then calculated in the same manner as with the basic MOESP algorithm, giving:

$$\hat{C} = U_{n(\text{first } l \text{ rows})}$$
$$\hat{A} = U_1^{\dagger} U_2$$

where:

$$U_1 = U_{n(\text{first } (i-1) \times l \text{ rows})}$$
$$U_2 = U_{n(\text{last } (i-1) \times l \text{ rows})}$$

The next step is stochastic stage in which the Kalman gain K is obtained. This begins by writing Equation 3.17 on innovations form:

$$Y_{k,i,j} = \Gamma_i \hat{X}_{k,j} + H_i U_{k,i,j} + M_i E_{k,i,j} \tag{3.22}$$

where:

$$M_i = \begin{pmatrix} I_l & 0 & \dots & 0 \\ CK & I_l & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{i-2}K & CA^{i-3}K & \dots & I_l \end{pmatrix}$$

In order to obtain estimates of the matrices Q, R and S we must first obtain an estimate of the state. Consider the observer equation:

$$\begin{aligned}\hat{x}(k+1) &= A\hat{x}(k) + Bu(k) + K(y(k) - \hat{y}(k)) \\ &= \tilde{A}\hat{x}(k) + \tilde{B}u(k) + Ky(k)\end{aligned}$$

where:

$$\tilde{A} = A - KC, \qquad \tilde{B} = B - KD$$

The state sequence may then be written as:

$$\hat{X}_{k,j} = \tilde{A}^i \hat{X}_{0,j} + \mathcal{C}_{u,i} U_{0,i,j} + \mathcal{C}_{y,i} Y_{0,i,j}$$

where $\hat{X}_{0,j}$ is the initial state and:

$$\begin{aligned}\mathcal{C}_{u,i} &= \begin{pmatrix} \tilde{A}^{i-1}\tilde{B} & \tilde{A}^{i-2}\tilde{B} & \dots & \tilde{B} \end{pmatrix} \\ \mathcal{C}_{y,i} &= \begin{pmatrix} \tilde{A}^{i-1}K & \tilde{A}^{i-2}K & \dots & K \end{pmatrix}\end{aligned}$$

Owing to the stability of $\tilde{A}$, the term $\tilde{A}^i \hat{X}_{0,j}$ tends to zero as i tends to infinity. Substituting the expression for the state sequence into Equation 3.22, and allowing i to tend to infinity, we obtain:

$$\lim_{i\to\infty} Y_{k,i,j} = \Gamma_i \lim_{i\to\infty} (\mathcal{C}_{u,i} U_{0,i,j} + \mathcal{C}_{y,i} Y_{0,i,j}) + \lim_{i\to\infty} H_i U_{k,i,j} + \lim_{i\to\infty} M_i E_{k,i,j}$$

Here, Γ_i and H_j may be computed from the system matrix estimates found in the deterministic identification stage. The state sequence may be obtained by using the instrumental variable:

$$\mathcal{Z} = \begin{pmatrix} U_{i,2i,j} \\ U_{0,i,j} \\ Y_{0,i,j} \end{pmatrix}$$

This gives:

$$\lim_{i\to\infty} Y_{k,i,j}\Pi_{\mathcal{Z}} = \lim_{i\to\infty} \Gamma_i \hat{X}_{k,j}\Pi_{\mathcal{Z}} + \lim_{i\to\infty} H_i U_{k,i,j}\Pi_{\mathcal{Z}} + \lim_{i\to\infty} M_i E_{k,i,j}\Pi_{\mathcal{Z}}$$

Since the row spaces of $\hat{X}_{k,j}$ and $U_{k,i,j}$ are completely spanned by the row space of $\mathcal{Z}$, the projection operator multiplying these terms may be omitted. This gives:

$$\hat{X}_{k,j} = \hat{\Gamma}_i^{\dagger}(Y_{k,i,j}\Pi_{\mathcal{Z}} - \hat{H}_i U_{k,i,j})$$

An estimate of $\hat{X}_{k+1,j}$ is given by:

$$\hat{X}_{k+1,j} = \hat{\Gamma}_{i-1}^{\dagger}(Y_{k+1,i-1,j}\Pi_{\mathcal{Z}^+} - \hat{H}_{i-1} U_{k+1,i-1,j})$$

where:

$$\mathcal{Z}^+ = \begin{pmatrix} U_{i+1,2i-1,j} \\ U_{0,i+1,j} \\ Y_{0,i+1,j} \end{pmatrix}$$

Estimates of $W_{k,i,j}$ and $V_{k,i,j}$ may be obtained by solving:

$$\begin{pmatrix} \hat{X}_{k+1,j} \\ Y_{k,1,j} \end{pmatrix} = \begin{pmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{pmatrix} \begin{pmatrix} \hat{X}_{k,j} \\ U_{k,1,j} \end{pmatrix} + \begin{pmatrix} \hat{W}_{k,1,j} \\ \hat{V}_{k,1,j} \end{pmatrix}$$

The covariance matrices may them be obtained as:

$$\begin{pmatrix} \hat{Q} & \hat{S} \\ \hat{S}^T & \hat{R} \end{pmatrix} = \frac{1}{N} \begin{pmatrix} \hat{W}_{k,1,j} \\ \hat{V}_{k,1,j} \end{pmatrix} \begin{pmatrix} \hat{W}_{k,1,j} \\ \hat{V}_{k,1,j} \end{pmatrix}^T$$

The Kalman gain K is finally given by:

$$\begin{aligned} \hat{K} &= (\hat{A}\hat{P}\hat{C}^T + \hat{S})(\hat{C}\hat{P}\hat{C}^T + \hat{R})^{-1} \\ \hat{P} &= \hat{A}\hat{P}\hat{A}^T + \hat{Q} + (\hat{A}\hat{P}\hat{C}^T + \hat{S})(\hat{C}\hat{P}\hat{C}^T + \hat{R})^{-1}(\hat{A}\hat{P}\hat{C}^T + \hat{S})^T \end{aligned}$$

The POMOESP algorithm is an attractive one from a control point of view due to its stochastic elements, which allow the calculation of the Kalman gain. This may be used in control applications for implementing for example a one step ahead predictor for the state. It should however be remembered that the subspace algorithms generate state-space models in an effectively arbitrary state-space basis, meaning that the states will almost certainly lack physical interpretation. This is important for the choice of control design methodology, and will be discussed in greater depth in later chapters.

A further algorithm exists for solving the errors-in-variables identification problem which incorporates noise on the input. However, in many identification tasks the input is specifically generated for the purposes of identification, and thus does not include noise. For this reason this variant of the MOESP algorithm will not be presented here.

Chapter 4

Intersection Algorithms

4.1 Introduction

The idea of the intersection algorithms is to obtain the future state sequence X_f as the intersection of the row space of the past inputs and outputs W_p and the future inputs and outputs U_f and Y_f:

$$\text{row space } X_f = \text{row space } W_p \cap \text{row space } \begin{pmatrix} U_f \\ Y_f \end{pmatrix} \tag{4.1}$$

Algorithms based on this idea are presented in [14]. Although these algorithms will not be implemented or directly utilized in this thesis, they are included here for completeness. Additionally, an online adaptive algorithm is presented in [14] which provides an interesting alternative to the more usual block based algorithms.

Unlike the MOESP class of algorithms presented in the previous chapter, the intersection algorithms presented here determine estimates of the system matrices via estimation of a state sequence. Since these algorithms are considered in less detail in this thesis than the MOESP class of algorithms, the presentation here will be very brief, consisting only of a statement of the algorithms. The full proofs may be found in [14].

4.2 Offline Algorithm

Define the Hankel matrices of input-output data (notice the different structure as compared to the MOESP algorithms):

$$H_1 = \begin{pmatrix} u(k) & u(k+1) & \dots & u(k+j-1) \\ y(k) & y(k+1) & \dots & y(k+j-1) \\ u(k+1) & u(k+2) & \dots & u(k+j) \\ y(k+1) & y(k+2) & \dots & y(k+j) \\ \vdots & \vdots & \ddots & \vdots \\ u(k+i-1) & u(k+i) & \dots & u(k+i+j-2) \\ y(k+i-1) & y(k+i) & \dots & y(k+i+j-2) \end{pmatrix}$$

$$H_2 = \begin{pmatrix} u(k+i) & u(k+1+1) & \dots & u(k+i+j-1) \\ y(k+i) & y(k+1+1) & \dots & y(k+i+j-1) \\ u(k+i+1) & u(k+i+2) & \dots & u(k+i+j) \\ y(k+i+1) & y(k+i+2) & \dots & y(k+i+j) \\ \vdots & \vdots & \ddots & \vdots \\ u(k+2i-1) & u(k+2i) & \dots & u(k+2i+j-2) \\ y(k+2i-1) & y(k+2i) & \dots & y(k+2i+j-2) \end{pmatrix}$$

Define H as the concatenation of the above matrices:

$$H = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}$$

Calculate and partition the SVD of H:

$$H = U\Sigma V^T = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \begin{pmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{pmatrix} V^T$$

Calculate the SVD of the matrix $U_{12}^T U_{11} \Sigma_{11}$:

$$U_{12}^T U_{11} \Sigma_{11} = \begin{pmatrix} U_q & U_q^\perp \end{pmatrix} \begin{pmatrix} \Sigma_q & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_q^T \\ V_q^T \end{pmatrix}$$

Estimates of the system matrices are obtained by solving the linear equations:

$$\begin{pmatrix} U_q^T U_{12}^T U(m+l+1:(i+1)(m+l),:)\Sigma \\ U(mi+li+m+1:(m+l)(i+1),:)\Sigma \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} U_q^T U_{12}^T U(1:mi+li,:)\Sigma \\ U(mi+li+1:mi+li+m,:)\Sigma \end{pmatrix}$$

where the matrix indices follow standard Matlab notation.

4.3 Online Algorithm

In [14] an online version of the algorithm presented in the previous section is derived. It involves updating the H matrix with a new column when new input-output samples are available. Old values may be discarded through the use of a forgetting factor. The scheme is thus analogous to an RLS algorithm with a forgetting factor, and is suited to situations where the process dynamics may be changing.

The algorithm is initialized by creating the matrices:

$$U_0 = I_{(2mi+2li)\times(2mi+2li)}, \qquad \Sigma_0 = 0_{(2mi+2li)\times(2mi+2li)}$$

where m and l are the input and output dimensions respectively, and i is the nominal block row size of the Hankel matrix H. At each sampling instant, construct a new column of the Hankel matrix H, H_k^+, and calculate the SVD:

$$U_k \Sigma_k V_k^T = \begin{pmatrix} \alpha U_{k-1}\Sigma_{k-1} & H_k^+ \end{pmatrix}$$

where $\alpha < 1$ is a forgetting factor, multiplying previous input-output information. Form the partition:

$$U_k \Sigma_k = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \begin{pmatrix} \Sigma_1 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Calculate and partition the SVD of $U_{12}^T U_{11} \Sigma_{11}$:

$$U_{12}^T U_{11} \Sigma_{11} = \begin{pmatrix} U_q & U_q^\perp \end{pmatrix} \begin{pmatrix} \Sigma_q & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_q^T \\ V_q^T \end{pmatrix}$$

Estimates of the system matrices are obtained by solving the linear equations:

$$\begin{pmatrix} U_q^T U_{12}^T U(m+l+1:(i+1)(m+l),:)\Sigma \\ U(mi+li+m+1:(m+l)(i+1),:)\Sigma \end{pmatrix} =$$
$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} U_q^T U_{12}^T U(1:mi+li,:)\Sigma \\ U(mi+li+1:mi+li+m,:)\Sigma \end{pmatrix}$$

It can be seen that this algorithm requires the computation of two singular value decompositions at each sampling instant. For this reason it may be suggested that the algorithm is restricted to systems with relatively low sampling frequencies. Alternatively the algorithm could be modified to calculate new estimates of the system matrices after a given number of sampling periods. This would lower the computational demand while retaining the adaptive nature of the algorithm.

Part II

Adaptive Control and Model Predictive Control

Chapter 5

Adaptive Control

5.1 Introduction

An adaptive control system is one in which the controller parameters may be adjusted by some mechanism. There exists a wide variety of mechanisms for adjustment, among them system identification of the control object, and some performance measure of the system such as output variance. The aim of an adaptive controller is to automatically provide a competitive controller in situations where the dynamics of the control object may be varying. This could include for example a chemical process, a paper mill or an aircraft. Adaptive control can also be useful in situations where disturbance characteristics vary in nature. A good example of this would be a ship autopilot, where the disturbances in the form of waves and currents cannot be approximated by simple disturbance models. The structure of an adaptive controller is shown in Figure 5.1.

Some of the first adaptive control techniques were developed in the 1950s for aircraft flight controllers. This was required because aircraft dynamics differ depending on flight conditions such as altitude and Mach number, meaning a fixed dynamic controller may not work well in all operating modes. Most aircraft use a technique called *Gain Scheduling* in which a control strategy is selected

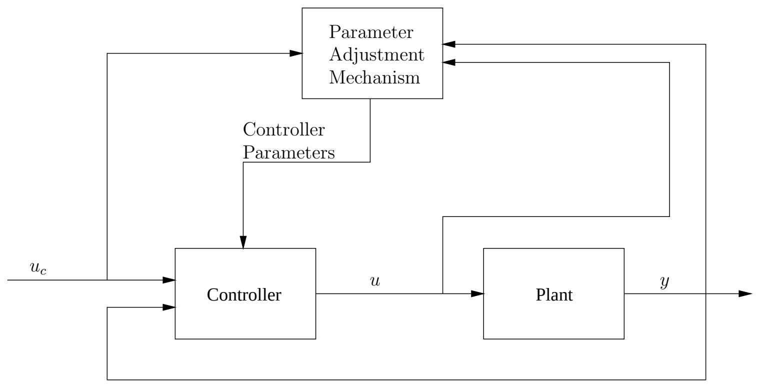

Figure 5.1: General adaptive controller structure

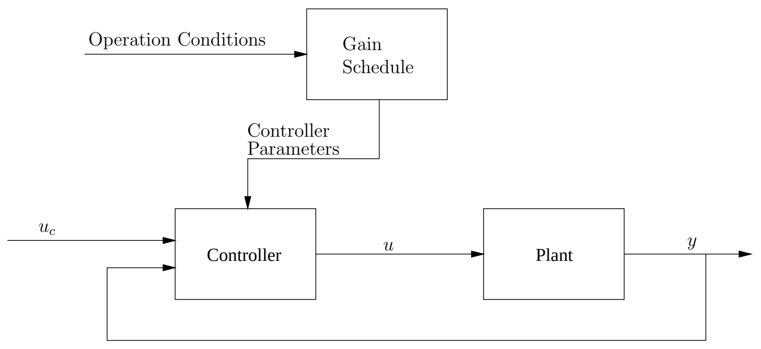

Figure 5.2: Gain scheduling controller structure

depending on the measured operating conditions. This can be considered as a form of adaptive control in which the mechanism for adjusting the control is the measurement of operating conditions. Figure 5.2 shows the basic structure of a gain scheduling controller.

In the 1970s and 1980s new schemes were developed using various structures. Among these are the *Model Reference Adaptive System* (MRAS) and the *Self Tuning Regulator* (STR). In reality these schemes are equivalent, but were developled for different standpoints and remain regarded as separate methodologies. This chapter aims to give a very brief overview of the operation of these two types of system and their variants, in order to provide comparisons and analogies with new algorithms and structures developed in the project. The reference material for this chapter is [20], *Adaptive Control* by Åström and Wittenmark, and the reader is directed to this book for further information on any adaptive control schemes.

5.2 Self Tuning Regulators

5.2.1 RLS Estimation

Many adaptive control strategies involve the estimation of parameters in an ARMA model of the plant to be controlled. The estimated model is then used in some standard control design methodology, such as pole placement design. Such schemes are known as *indirect adaptive controllers.* A common method of estimating the plant parameters is *Recursive Least Squares* (RLS). This is, as the name suggests, a recursive implementation of the least-squares estimation process, which is computationally efficient since it does not recalculate the standard least-squares solution $\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T\mathcal{Y}$ every time a new data sample is obtained, but only updates various matrices. The proof is not given here (see [7]), rather the result is stated.

Consider an ARMA model of the form:

$$\begin{aligned} y(t) = &- a_1y(t-1) - a_2y(t-2) - \ldots - a_ny(t-n) \\ &+ b_0u(t-1) + \ldots + b_m(t-m-1) \end{aligned}$$

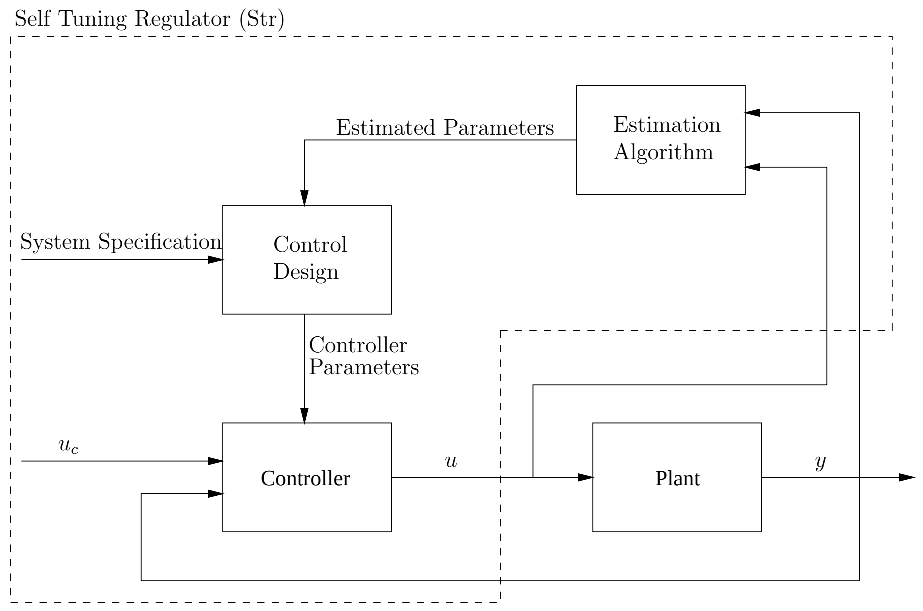

Figure 5.3: Self tuning regulator structure

Define the *regressors*:

$$\begin{aligned}\theta &= \begin{pmatrix} a_1 & a_2 & \dots & a_n & b_0 & \dots & b_m \end{pmatrix}^T \\ \phi(t-1) &= \begin{pmatrix} -y(t-1) & \dots & -y(t-n) & u(t-1) & \dots & u(t-m-1) \end{pmatrix}^T\end{aligned}$$

It is now possible to write the *regression equation*

$$y(t) = \phi^T(t-1)\theta$$

The recursive least-squares estimate is given by:

$$\begin{aligned}\hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)\epsilon(t) \\ \epsilon(t) &= y(t) - \phi^T(t-1)\hat{\theta}(t-1) \\ K(t) &= P(t-1)\phi(t-1)(\lambda + \phi^T(t-1)P(t-1)\phi(t-1))^{-1} \\ P(t) &= (I - K(t)\phi^T(t-1))P(t-1)/\lambda\end{aligned}$$

Here the factor λ is known as the *forgetting factor*, and is used to ‘discard’ old data. This can be useful in situations where the process is changing with time. A low value of λ uses fewer past data points in the calculation of the estimates. Notice however that the update law for $P(t)$ contains λ in the denominator, and this can cause an effect known as ‘P-matrix explosion’, particularly for lower values of λ.

5.2.2 Pole Placement Design

Pole placement design is a standard controller design, useful for SISO systems. It is commonly used as the design methodology in indirect adaptive controllers.

In this application it is often used with polynomial input-output representation, to correspond to the estimated transfer function of the unknown system, often obtained from RLS parameter estimation.

Consider the discrete-time system representation:

$$A(q)y(t) = B(q)(u(t) + v(t))$$

Consider also the general controller:

$$Ru(t) = Tu_c(t) - Sy(t))$$

containing both feedforward and feedback transfer functions. The closed loop system is given by:

$$\begin{aligned} y(t) &= \frac{BT}{AR+BS}u_c(t) + \frac{BR}{AR+BS}v(t) \\ u(t) &= \frac{AT}{AR+BS}u_c(t) - \frac{BS}{AR+BS}v(t) \end{aligned} \tag{5.1}$$

The closed loop characteristic polynomial is given by $AR + BS$. The idea of pole placement design is to choose R and S such that this polynomial is equal to a desired polynomial A_c. This gives rise to the *Diophantine equation*:

$$AR + BS = A_c \tag{5.2}$$

To obtain the T polynomial, consider the desired response to the reference input u_c:

$$A_m y_m(t) = B_m u_c(t)$$

From Equations 5.1 and 5.2 it can be seen that:

$$\frac{BT}{A_c} = \frac{B_m}{A_m} \tag{5.3}$$

Factor B as:

$$B = B^+B^-$$

where B^+ corresponds to stable, well-damped plant zeros that may be cancelled by the controller, and B^- to unstable or poorly damped zeros that may not be cancelled. From Equation 5.3 it can be seen that B^- must be a factor of B_m, so we may write $B_m = B^-B'_m$. If B^+ is cancelled it must therefore be a factor of A_c, as must A_m from Equation 5.3. This gives:

$$A_c = A_0 A_m B^+$$

where the polynomial A_0 has an interpretation, in the case of no zero cancellation ($B^+ = 0$), as an observer polynomial. This is justified by considering the state-space version of pole placement design, done using observer-based state feedback. For more information see [21] and [20].

From Equation 5.2 it may also may be seen that B^+ is a factor of R, i.e. that $R = B^+R'$. The Diophantine equation then becomes:

$$AR' + B^-S = A_0A_m$$

By cancelling factors in the above equations we obtain an expression for T:

$$T = A_0B'_m$$

5.2.3 An STR Example

Consider the second order process:

$$G(s) = \frac{b}{s(s+a)} \tag{5.4}$$

where the parameters a and b are unknown. For the design it is assumed that only the orders of the numerator and denominator polynomials are known, not their structure.

Let the desired response be that of a standard second order system:

$$G_m(s) = \frac{\omega_m^2}{s^2 + 2\zeta_m\omega_m + \omega_m^2} \tag{5.5}$$

where $\omega_m = 1$ and $\zeta_m = 0.7$. Let the design incorporate an observer polynomial of the form:

$$A_o(s) = (s + a_o)^\gamma$$

with $a_o = 3$, and γ the smallest possible integer. The sampling period is chosen to be $h = 0.5$. Converting the continuous time transfer functions of the process and desired response into discrete time representations yields:

$$G(q) = \frac{b_0 q + b_1}{q^2 + a_1 q + a_2} \tag{5.6}$$

and

$$G_m(q) = \frac{b_{m_0} q + b_{m_1}}{q^2 + a_{m_1} q + a_{m_2}} \tag{5.7}$$

Applying the Diophantine Equation $AR + BS = A_o A_m B^+$ to the design problem gives:

$$(q^2 + a_1 q + a_2)(q + r_1) + (b_0 q + b_1)(s_0 q + s_1) = (q^2 + a_{m_1} q + a_{m_2})(q + a_o) \tag{5.8}$$

where the controller parameters can be found by comparing coefficients. The discretized desired response is given by:

$$G_m(q) = \frac{0.09833q + 0.07778}{q^2 - 1.32q + 0.4966} \tag{5.9}$$

The $T(q)$ polynomial may then be calculated using the equation:

$$T(q) = \beta A_0(q) \tag{5.10}$$

where

$$\beta = \frac{A_m(1)}{B(1)} \tag{5.11}$$

Using the calculated values from A_m we can calculate β:

$$\beta = \frac{0.1766}{b_0 + b_1} \tag{5.12}$$

$T(q)$ is hence given by:

$$T(q) = \frac{0.1766(q + a_o)}{b_0 + b_1} \tag{5.13}$$

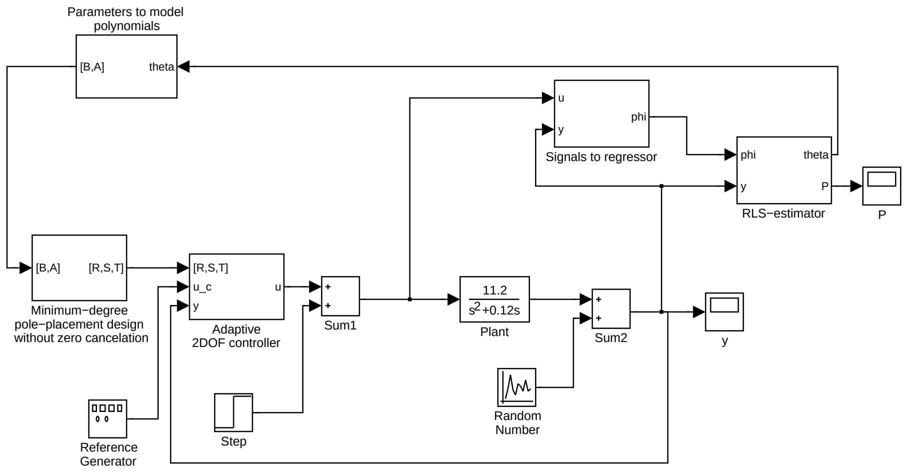

Figure 5.4: Simulink model of a second order STR

Figures 5.5 to 5.8 show the results of simulations using this design. In particular, the effect of the forgetting factor λ of the RLS estimator can be seen. In Figure 5.5 a relatively high forgetting factor, $\lambda = 0.99$ is used, and gives good results in the absence of noise and disturbances. Figure 5.6 shows the effect of a parameter change during the simulation, again with $\lambda = 0.99$. It can be seen that the system does not adapt well to the parameter change, with the estimated parameters varying slowly with time and the output failing to tract the reference signal in the desired way. In Figure 5.7 the same parameter change is performed during the simulation, but this time a forgetting factor of $\lambda = 0.8$ is used. Clearly the estimated parameters change much more quickly, with the result that the output continues to track the reference signal with minimal disturbance. However there are disadvantages with using such a low forgetting factor. Figure 5.8 shows the effect of adding output noise when $\lambda = 0.8$. Notice also the 'P-matrix explosion' effect, which although present in all the simulations, is more pronounced for lower λ. The estimated parameters are very noisy, because the system effectively adapts to the noise. In practical applications forgetting factor values between 0.97 and 0.99 are most common.

5.3 Model Reference Adaptive Systems

The MRAS solves a control problem in which system specifications are given in the form of a reference model. This could for example be the desired dynamics of a new aircraft, or the desired response of a robot arm. The model produces a desired output corresponding to the input signal, which in the aircraft example would be the pilot's control inputs. The error between the desired output and the actual output is then used to adjust the controller parameters. Figure 5.9 illustrates a model reference adaptive system structure.

The method used to adjust the control parameters must not only drive the

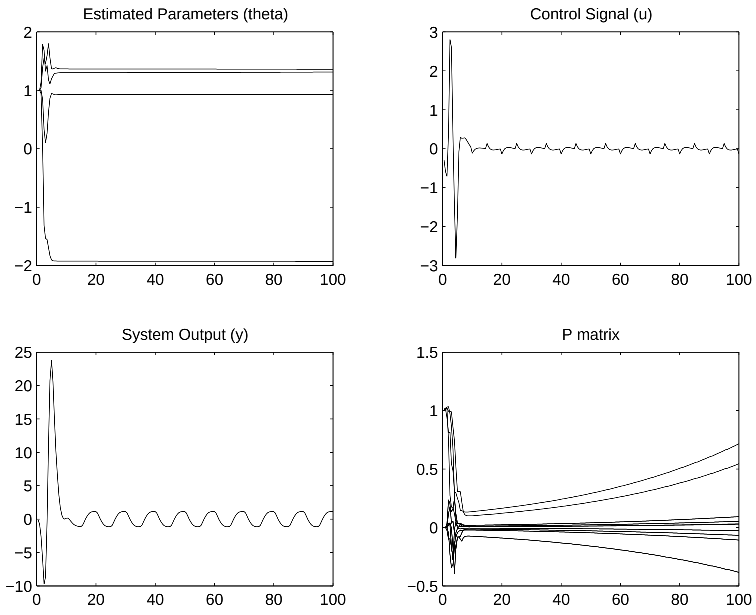

Figure 5.5: STR simulation results. $\lambda = 0.99$, input a square wave of amplitude 1

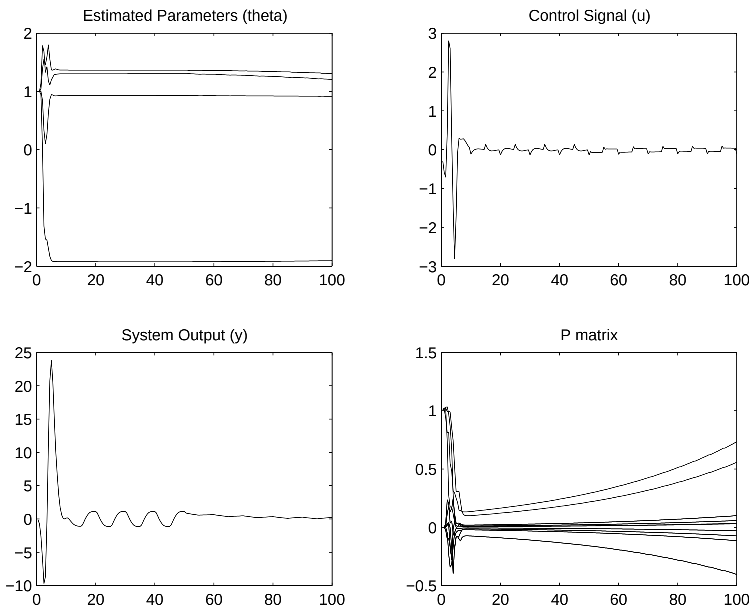

Figure 5.6: STR simulation results for process parameter change at time 50. $\lambda = 0.99$, input a square wave of amplitude 1

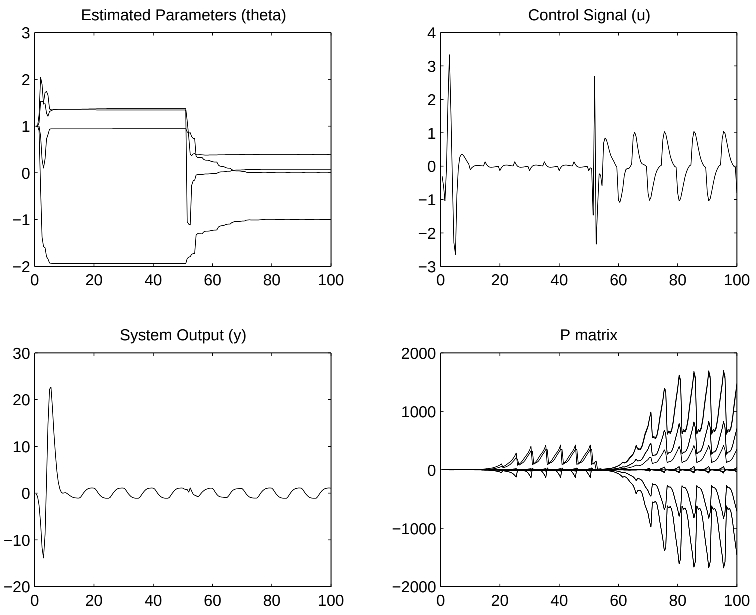

Figure 5.7: STR simulation results for process parameter change at time 50. $\lambda = 0.8$, input a square wave of amplitude 1

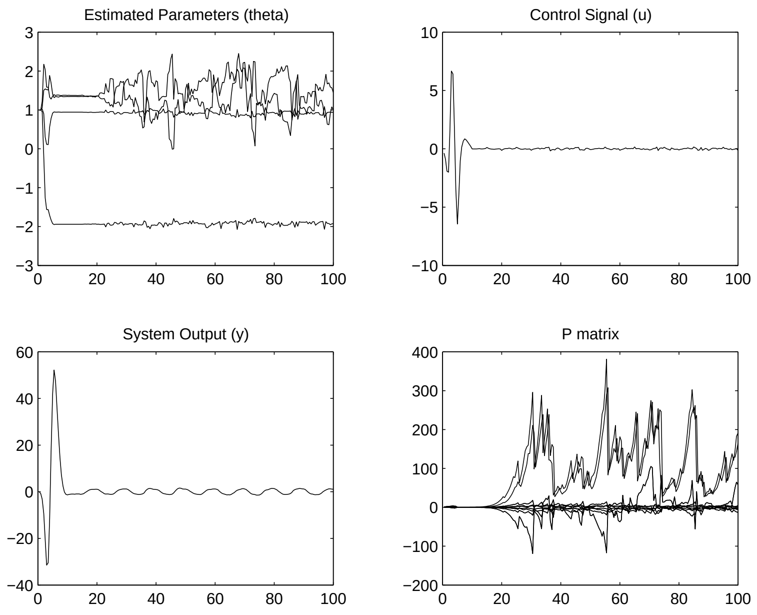

Figure 5.8: STR simulation results for noisy process noise variance 0.001. $\lambda = 0.8$, input a square wave of amplitude 1

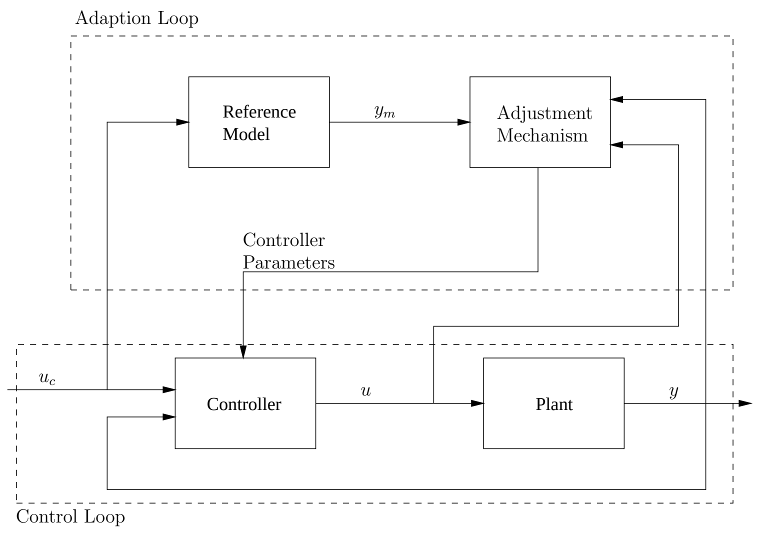

Figure 5.9: Model reference adaptive system structure

error between the model output and actual output to zero, but to ensure the system is stable. Several methods exist for doing this. One such method is a gradient method based on the error between the actual output y and the reference model output y_m. The mechanism for adjusting the control parameters is given by the *MIT rule*, discussed in Subsection 5.3.1. A design example using the MIT rule is presented in Subsection 5.3.2.

5.3.1 The MIT Rule

The MIT rule was the first adjustment mechanism used in model reference adaptive systems. It is used to adjust control parameters by reducing a loss funtion. This loss function is based on the error between the actual output y and the reference model output y_m:

$$e = y - y_m$$

For a general loss function J the MIT rule may be stated as:

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} \tag{5.14}$$

There are of course a wide variety of loss functions to be chosen. A common one is given by:

$$J = \frac{1}{2}e^2$$

which gives a parameter update law (from Equation 5.14):

$$\frac{d\theta}{dt} = -\gamma e \frac{\partial e}{\partial \theta}$$

A design example using such a parameter update law is given in the following subsection.

5.3.2 An MIT Rule Example

Consider the unknown second order process given by:

$$G(s) = \frac{k}{s^2 + a} \tag{5.15}$$

where the parameters a and k are unknown. The specification is given in the form of the second order model:

$$G_m(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \tag{5.16}$$

where $\omega = 1$ and $\zeta = 0.7$.

Consider the use of an RST controller:

$$(s + r_1)U(s) = (t_0 s + t_1)U_c(s) + (s_0 s + s_1)$$

Using the Diophantine equation $AR + BS = A_o A_m B^+$ we obtain:

$$(s^2 + a)(s + r_1) + k(s_0 s + s_1) = (s + a_o)(s^2 + 2\zeta\omega s + \omega^2)$$

Comparing coefficients:

$$\begin{array}{ll} s^2 & r_1 = 2\zeta\omega + a_o \\ s^1 & a + ks_0 = \omega^2 + 2a_o\zeta\omega \\ s^0 & ar_1 + ks_1 = a_o\omega^2 \end{array}$$

Solving these for the parameters gives:

$$\begin{aligned} r_1 &= 2\zeta\omega + a_o \\ s_0 &= \frac{\omega^2 + 2a_o\zeta\omega - a}{k} \\ s_1 &= \frac{\omega^2 - 2a\zeta\omega + aa_o}{k} \end{aligned} \tag{5.17}$$

To obtain the T polynomial, the equation:

$$T = \frac{B_m A_o}{B}$$

can be used, which yields the parameters:

$$\begin{aligned} t_0 &= \frac{\omega^2}{k} \\ t_1 &= \frac{\omega^2 a_o}{k} \end{aligned}$$

The MIT rule:

$$\frac{de}{d\theta} = -\gamma e \frac{d\theta}{dt}$$

may be used to obtain the parameter update laws. The output equation:

$$y = \frac{BTu_c}{AR + BS}$$

together with the desired output:

$$y_m = \frac{B_m u_c}{A_m}$$

can be used with the error equation:

$$e = y - y_m$$

to obtain

$$e = (\frac{BT}{AR + BS} - \frac{B_m}{A_m})u_c$$

This is then differentiated with respect to the parameters to obtain the parameter update laws:

$$\begin{aligned} \frac{dr_1}{dt} &= \frac{\gamma e B u}{A_o A_m} \\ \frac{ds_0}{dt} &= \frac{\gamma e B p y}{A_o A_m} \\ \frac{ds_1}{dt} &= \frac{\gamma e B y}{A_o A_m} \\ \frac{dt_0}{dt} &= \frac{-\gamma e B y_m}{A_o B_m} \\ \frac{dt_1}{dt} &= \frac{-\gamma e B y_m a_o}{A_o B_m} \end{aligned} \tag{5.18}$$

The next step is to determine any unknown parameters in these update laws and to make any necessary approximations in order to implement the laws. It can be seen that the only unknown parameter is the process numerator B, which appears in all the update laws. This can be compensated for in the implementation by writing:

$$\gamma' = \gamma B$$

It can also be recognised that only three of the five parameters must be estimated. The parameter r_1 is given by Equation 5.17 from the Diophantine equation; all factors in this expression are known. Also, we may observe that t_1 is a scaling of t_0 and need not be estimated separately.

Figure 5.10 shows a Simulink model used for simulations on this design. For the simulation the unknown system was chosen to be of the form in Equation 5.15 with parameters $k = 0.14$ and $a = 1$. Figure 5.11 shows the results when an adaption rate $\gamma = 2$ is used. It can be seen that parameter convergence is slow, but the correct values are reached and the system output eventually tracks the input square wave well.

One of the capabilities of an adaptive control system is that it can compensate for changes in plant dynamics. To illustrate this, the system was altered during the simulation, with the parameter a changed to 3 at time $t = 100$. Figure 5.12 shows the effects on the system for this parameter change, with $\gamma = 6$. It can be seen that the parameters converge to their new values, and that the output error decreases in magnitude, but that changes are relatively slow. Figure 5.13 shows the system response when an adaption gain $\gamma = 20$ is used, clearly the parameter changes occur much more quickly and the desired output is restored more quickly.

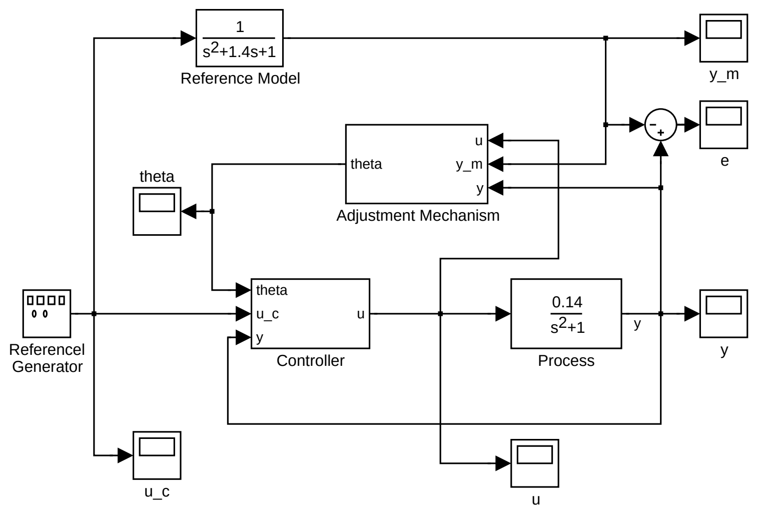

Figure 5.10: Simulink model of a second order MRAS

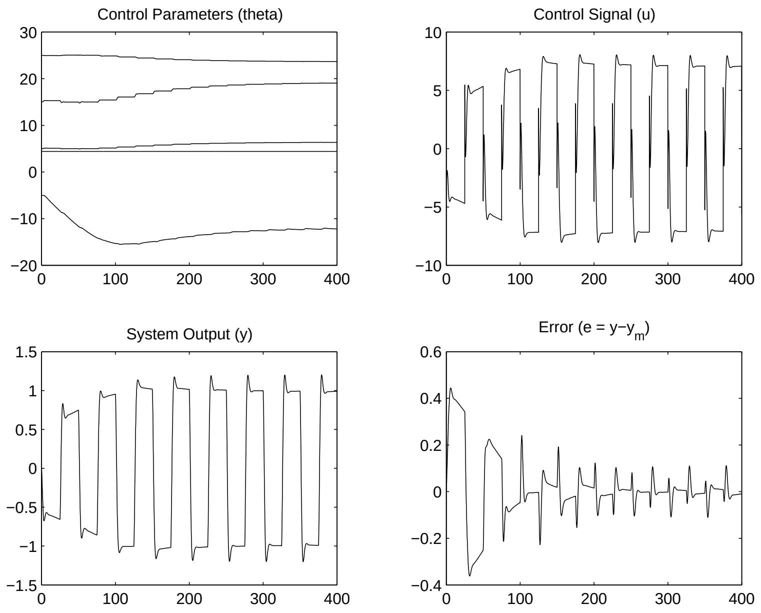

Figure 5.11: MRAS simulation results. $\gamma = 2$, input a square wave of amplitude 1

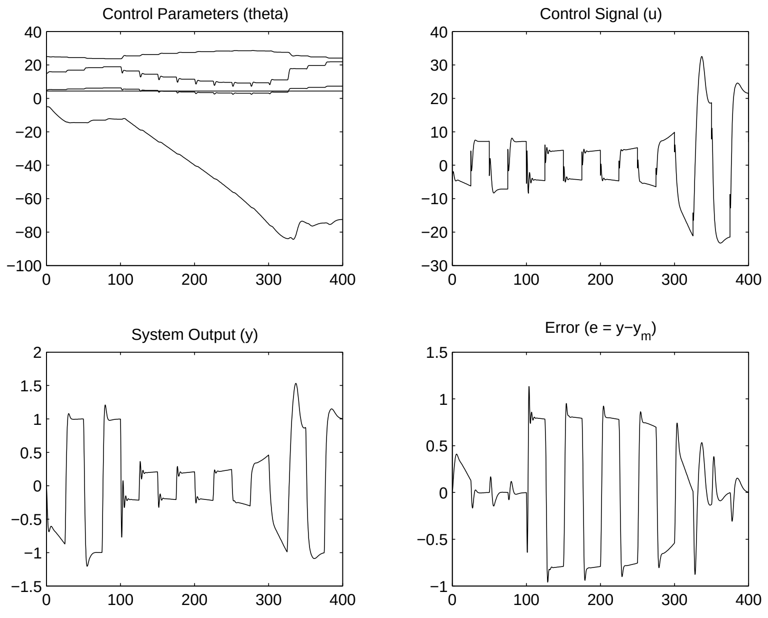

Figure 5.12: MRAS simulation results for process parameter change at time 100. $\gamma = 6$, input a square wave of amplitude 1

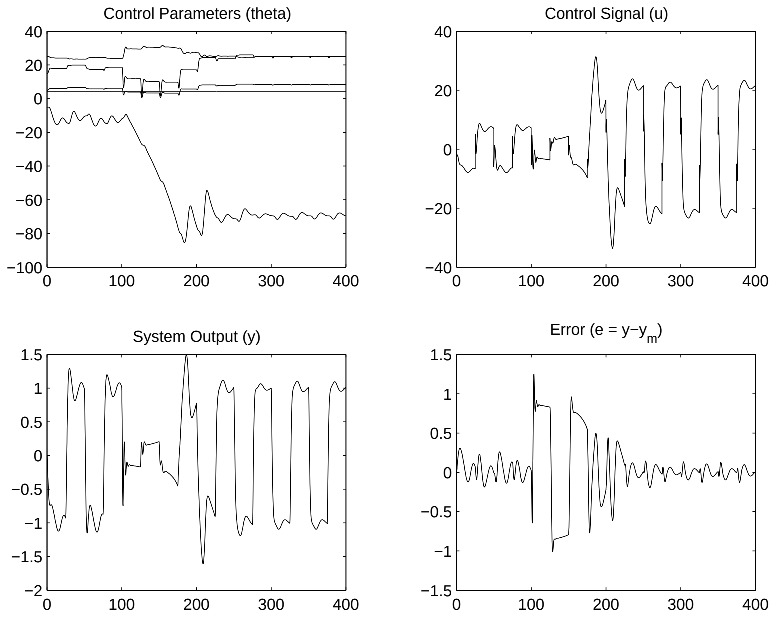

Figure 5.13: MRAS simulation results for process parameter change at time 100. $\gamma = 20$, input a square wave of amplitude 1

Chapter 6

Model Predictive Control

6.1 Introduction

Model Predictive Control (MPC) is a control strategy that utilizes a process model to provide prediction and solves an optimization problem online. A key feature of MPC is the *Receding Horizon Principle*, in which prediction is carried out over a finite set of sample points, and an optimal control sequence is obtained from online optimization. Only the first element of this sequence of control signals is actually implemented; the rest are discarded and a new optimal sequence is obtained at the next time step. This may appear at first glance to be a rather computationally-heavy control methodology, but efficient ways of solving optimizations (for example the QR decomposition) coupled with modern processing power serve to make the strategy viable. Indeed, MPC enjoys widespread use in industry, where its capability to deal with constraints on variables in a natural way allows it to be a more 'economical' controller than the traditional PID controller.

From a point of view of implementing adaptive control using subspace identification algorithms, MPC is attractive in that it is easily extensible to MIMO systems and can deal with reference signals without problems. Basic LQ control is more concerned with the *Regulator Problem* of bringing a system's states to zero than the *Servo Problem* of tracking a reference signal. Although solutions of the latter exist, for example in [1], they often require that the reference signal have certain properties, such as having been generated by a known system. This is not desirable for a general adaptive control implementation in which an arbitrary reference signal may be present.

This chapter aims to provide a brief overview of the formulation and solution of the MPC problem. In Chapter 9 a connection between MPC and subspace identification will be highlighted that will provide a new way of implementing an adaptive controller. Ideas from this chapter will be called upon further in Chapter 8. The presentation of MPC given here follows the state-space presentation shown in [13].

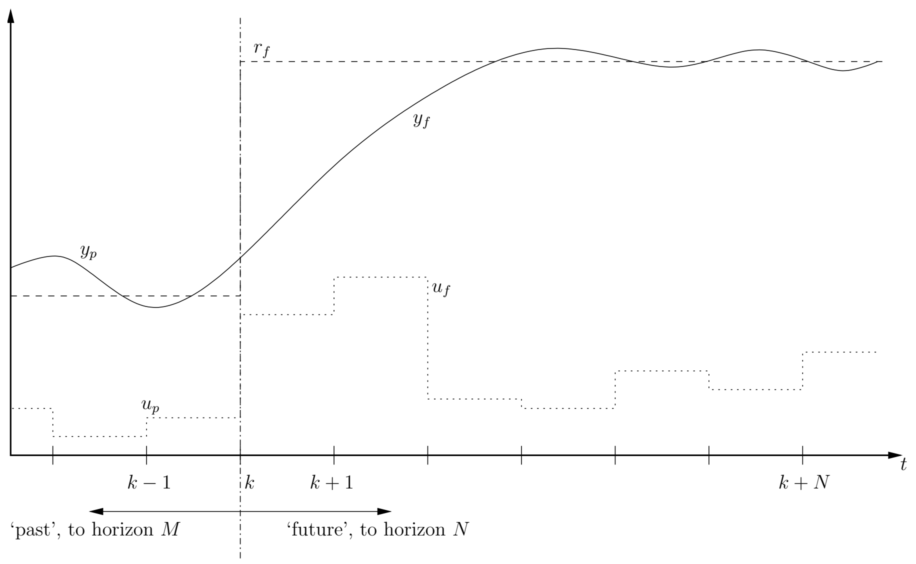

Figure 6.1: Signals for MPC control

6.2 A MPC Formulation

Let the model of the plant be a linear, discrete-time state-space representation of the form:

$$x(k+1) = Ax(k) + Bu(k) \tag{6.1}$$
$$y(k) = C_y x(k) \tag{6.2}$$
$$z(k) = C_z x(k) \tag{6.3}$$

where $y(k)$ is of size m_y and is the vector of measured outputs, and $z(k)$ is of size m_z and represents the set of outputs to be controlled. Very often we have $y(k) \equiv z(k)$, that is, the measured outputs are the same as the outputs to be controlled, and the system reduces to:

$$x(k+1) = Ax(k) + Bu(k)$$
$$z(k) = Cx(k)$$

The lack of any direct feedthrough term D is noticeable in the above equations. In [13] it is stated that since the output $u(k)$ is calculated from the measured output $y(k)$, including the direct feedthrough term complicates matters somewhat. The solution presented there involves introducing a new output variable given by:

$$\tilde{z}(k) = z(k) - Du(k) = C_z x(k)$$

and modifying the cost function slightly. It is also conceivable that $u(k)$ may be calculated from past information up to the time $k-1$, in which case the direct feedthrough problem does not arise.

6.2.1 The Cost Function

Solving optimal control problem involves the minimization of some cost function. Indeed, the term 'optimal' only refers to optimality with respect to a particular cost function. It is quite possible that the calculated 'optimal' control is in fact very bad control, due to a poor choice of cost function, or of cost function parameters.

The cost function presented in [13] is very similar to a standard LQ cost function, with the exception that it penalizes changes in input rather than the input itself. Define:

$$\Delta u(k) = u(k) - u(k-1) \tag{6.4}$$

Introduce the cost function:

$$V(k) = \sum_{i=H_w}^{H_p} \|\hat{z}(k+i|k) - r(k+i|k)\|^2_{Q(i)} + \sum_{i=0}^{H_u-1} \|\Delta\hat{u}(k+i|k)\|^2_{R(i)} \tag{6.5}$$

where $\hat{z}(k+i|k)$, $r(k+i|k)$, and $\hat{u}(k+i|k)$ are the predicted output, reference signal and control signals i time steps in the future, respectively. The weighting matrices $\mathcal{Q}$ and $\mathcal{R}$ are used to penalize the error between predicted output and reference signal, and control variation respectively. They may be represented by:

$$\mathcal{Q} = \begin{pmatrix} Q(H_w) & 0 & \dots & 0 \\ 0 & Q(H_w+1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q(H_p) \end{pmatrix}$$

$$\mathcal{R} = \begin{pmatrix} R(0) & 0 & \dots & 0 \\ 0 & R(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R(H_u-1) \end{pmatrix}$$

H_u is the control horizon, that is, the number of time steps over which an optimal control sequence will be calculated. H_p is the prediction horizon, or the number of time steps up to which the future outputs will be estimated. Since it might not be desirable to begin penalizing errors between the predicted output and reference signal immediately, the horizon H_w may be used to specify the future point at which the penalty begins to be applied. An alternative approach would be to set H_w to zero, and vary the weighting matrix $Q(i)$ such that the desired number of elements up to a given time are small or zero (although problems with the solution of the optimization arise when Q and R are not positive definite).

6.2.2 Prediction of the Future Outputs

An important element of MPC is the prediction of the future output sequence $\hat{z}(k+i|k)$. One method for obtaining this prediction is outlined here.

Begin by considering the sequence of predicted states, given by:

$$\begin{aligned}
\hat{x}(k+1|k) &= Ax(k) + B\hat{u}(k|k) \\
\hat{x}(k+2|k) &= A\hat{x}(k+1|k) + B\hat{u}(k+1|k) \\
&= A^2x(k) + AB\hat{u}(k|k) + B\hat{u}(k+1|k) \\
&\vdots \\
\hat{x}(k+H_p|k) &= A\hat{x}(k+H_p-1|k) + B\hat{u}(k+H_p-1|k) \\
&= A^{H_p}x(k) + A^{H_p-1}B\hat{u}(k|k) + B\hat{u}(k+H_p-1|k) \qquad (6.6)
\end{aligned}$$

Recalling Equation 6.4 we may also construct expressions for the future control signals:

$$\begin{aligned}
\hat{u}(k|k) &= \Delta\hat{u}(k|k) + u(k-1) \\
\hat{u}(k+1|k) &= \Delta\hat{u}(k+1|k) + \Delta\hat{u}(k|k) + u(k-1) \\
&\vdots \\
\hat{u}(k+H_u-1|k) &= \Delta\hat{u}(k+H_u-1|k) + \ldots + \Delta\hat{u}(k|k) + u(k-1) \qquad (6.7)
\end{aligned}$$

which may them be substituted in Equation 6.6 to obtain:

$$\begin{aligned}
\hat{x}(k+1|k) &= Ax(k) + B\{\Delta\hat{u}(k|k) + u(k-1)\} \\
\hat{x}(k+2|k) &= A^2x(k) + AB\{\Delta\hat{u}(k|k) + u(k-1)\} \\
&\quad + B\{\Delta\hat{u}(k+1|k) + \Delta\hat{u}(k|k) + u(k-1)\} \\
&= A^2x(k) + (A+I)B\Delta\hat{u}(k|k) + B\Delta\hat{u}(k+1|k) \\
&\quad + (A+I)Bu(k-1) \\
&\vdots \\
\hat{x}(k+H_u|k) &= A^{H_u}x(k) + (A^{H_u-1} + \ldots + A + I)B\Delta\hat{u}(k|k) + \ldots \\
&\quad + B\Delta\hat{u}(k+H_u-1|k) + (A^{H_u-1} + \ldots + A + I)Bu(k-1) \\
\hat{x}(k+H_u+1|k) &= A^{H_u+1}x(k) + (A^{H_u} + \ldots + A + I)B\Delta\hat{u}(k|k) + \ldots \\
&\quad + (A+I)B\Delta\hat{u}(k+H_u-1|k) \\
&\quad + (A^{H_u} + \ldots + A + I)Bu(k-1) \\
&\vdots \\
\hat{x}(k+H_p|k) &= A^{H_p}x(k) + (A^{H_p-1} + \ldots + A + I)B\Delta\hat{u}(k|k) + \ldots \\
&\quad + (A^{H_p-H_u} + \ldots + A + I)B\Delta\hat{u}(k+H_u-1|k) \\
&\quad + (A^{H_p-1} + \ldots + A + I)Bu(k-1) \qquad (6.8)
\end{aligned}$$

Creating vectors of future states and inputs and writing the above on matrix form, we obtain:

$$\begin{pmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+H_u|k) \\ \hat{x}(k+H_u+1|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{pmatrix} = \begin{pmatrix} A \\ \vdots \\ A^{H_u} \\ A^{H_u+1} \\ \vdots \\ A^{H_p} \end{pmatrix} x(k) + \begin{pmatrix} B \\ \vdots \\ \sum_{i=0}^{H_u-1} A^i B \\ \sum_{i=0}^{H_u} A^i B \\ \vdots \\ \sum_{i=0}^{H_p-1} A^i B \end{pmatrix} u(k-1)$$
$$+ \begin{pmatrix} B & \dots & 0 \\ AB+B & \dots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_u-1} A^i B & \dots & B \\ \sum_{i=0}^{H_u} A^i B & \dots & AB+B \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_p-1} A^i B & \dots & \sum_{i=0}^{H_p-H_u} A^i B \end{pmatrix} \begin{pmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{pmatrix} \tag{6.9}$$

To obtain the output predictions, we may use the expression for $z(k)$, including a direct feedthrough term for generality:

$$z(k) = Cx(k) + Du(k) \tag{6.10}$$

Construct the future sequences of predicted outputs, reference signals and control signal changes:

$$\mathcal{Z}(k) = \begin{pmatrix} \hat{z}(k|k) \\ \vdots \\ \hat{z}(k+H_p|k) \end{pmatrix} \tag{6.11}$$

$$\mathcal{T}(k) = \begin{pmatrix} r(k|k) \\ \vdots \\ r(k+H_p|k) \end{pmatrix} \tag{6.12}$$

$$\Delta\mathcal{U}(k) = \begin{pmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+H_u-1|k) \end{pmatrix} \tag{6.13}$$

$$\tag{6.14}$$

These are related by the expression:

$$\mathcal{Z}(k) = \Psi x(k) + \Gamma u(k-1) + \Theta \Delta\mathcal{U}(k) \tag{6.15}$$

From Equation 6.9, the matrices Ψ, Γ and Θ can be seen to be given by:

$$\Psi = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{H_u} \\ CA^{H_u+1} \\ \vdots \\ CA^{H_p} \end{pmatrix} \tag{6.16}$$

$$\Gamma = \begin{pmatrix} D \\ CB + D \\ \vdots \\ C\sum_{i=0}^{H_u-1} A^i B + D \\ C\sum_{i=0}^{H_u} A^i B + D \\ \vdots \\ C\sum_{i=0}^{H_p-1} A^i B + D \end{pmatrix} \tag{6.17}$$

$$\Theta = \begin{pmatrix} D & \dots & 0 \\ CB + D & \dots & 0 \\ C(AB + B) + D & \dots & 0 \\ \vdots & \ddots & \vdots \\ C\sum_{i=0}^{H_u-1} A^i B + D & \dots & CB + D \\ C\sum_{i=0}^{H_u} A^i B + D & \dots & C(AB + B) + D \\ \vdots & \ddots & \vdots \\ C\sum_{i=0}^{H_p-1} A^i B + D & \dots & C\sum_{i=0}^{H_p-H_u} A^i B + D \end{pmatrix} \tag{6.18}$$

6.2.3 Solution of the MPC Problem

A solution to the MPC problem, where the system's states are available and in the absence of disturbances, will be presented here. Using the above notation the cost function of Equation 6.5 may be written on matrix form:

$$V(k) = \|\mathcal{Z}(k) - \mathcal{T}(k)\|^2_{\mathcal{Q}} + \|\Delta\mathcal{U}(k)\|^2_{\mathcal{R}} \tag{6.19}$$

Define:

$$\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Gamma u(k-1) \tag{6.20}$$

This expression can be thought of as a tracking error, between the reference signal and the response of the system if no input changes occurred over the horizon. Making a substitution from Equation 6.20 into the matrix cost function in Equation 6.19 we obtain:

$$\begin{aligned} V(k) &= \|\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)\|^2_{\mathcal{Q}} + \|\Delta\mathcal{U}(k)\|^2_{\mathcal{R}} \\ &= (\Delta\mathcal{U}(k)^T\Theta^T - \mathcal{E}(k)^T)\mathcal{Q}(\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)) + \Delta\mathcal{U}(k)^T\mathcal{R}\Delta\mathcal{U}(k) \\ &= \mathcal{E}(k)^T\mathcal{Q}\mathcal{E}(k) - 2\Delta\mathcal{U}(k)^T\Theta^T\mathcal{Q}\mathcal{E}(k) + \Delta\mathcal{U}(k)^T(\Theta^T\mathcal{Q}\Theta + \mathcal{R})\Delta\mathcal{U}(k) \end{aligned}$$

Writing only the terms containing the control changes:

$$V(k) = \Delta\mathcal{U}(k)^T\mathcal{H}\Delta\mathcal{U}(k) - \Delta\mathcal{U}(k)^T\mathcal{G} + constant \tag{6.21}$$

with:

$$\mathcal{G} = 2\Theta^T \mathcal{Q}\mathcal{E}(k)$$
$$\mathcal{H} = \Theta^T \mathcal{Q}\Theta + \mathcal{R}$$

Taking the derivative of Equation 6.21, we obtain:

$$\frac{\partial V(k)}{\partial \Delta\mathcal{U}(k)} = 2\mathcal{H}\Delta\mathcal{U}(k) - \mathcal{G}$$

Setting this to zero, we obtain the optimal set of future control signal changes:

$$\Delta\mathcal{U}(k)_{optimal} = \frac{1}{2}\mathcal{H}^{-1}\mathcal{G} \tag{6.22}$$

To confirm that this result in fact yields a minimum, observe the second derivative of the cost function:

$$\begin{aligned}\frac{\partial^2 V(k)}{\partial \Delta\mathcal{U}(k)^2} &= 2\mathcal{H} \\ &= 2(\Theta^T \mathcal{Q}\Theta + \mathcal{R})\end{aligned}$$

Thus if $\mathcal{Q}$ and $\mathcal{R}$ are both positive definite, the solution given in Equation 6.22 corresponds to a minimum of the cost function.

6.2.4 Solution in the Constrained Case

MPC also allows for the solution of control problems where linear inequality constraints are present. Let these constraints take the form:

$$E\begin{pmatrix}\Delta\mathcal{U}(k) \\ 1\end{pmatrix} \leq 0$$
$$F\begin{pmatrix}\mathcal{U}(k) \\ 1\end{pmatrix} \leq 0$$
$$G\begin{pmatrix}\mathcal{Z}(k) \\ 1\end{pmatrix} \leq 0$$

It is required to express these constraints as limitations on the control signal adjustments, $\Delta\mathcal{U}(k)$. It may be shown that the above constraints can be written on the form:

$$\begin{pmatrix}\mathcal{F} \\ \mathcal{G}\Theta \\ \mathcal{W}\end{pmatrix}\Delta\mathcal{U}(k) \leq \begin{pmatrix}-\mathcal{F}_1 u(k-1) - f \\ -\mathcal{G}(\Psi x(k) + \Gamma u(k-1)) - g \\ w\end{pmatrix} \tag{6.23}$$

Again we wish to minimize the cost function:

$$V(k) = \Delta\mathcal{U}(k)^T\mathcal{H}\Delta\mathcal{U}(k) - \Delta\mathcal{U}(k)^T\mathcal{G} + constant$$

subject to the constraints in Equation 6.23. This is a problem on the form:

$$\min_{\theta} \frac{1}{2}\theta^T\Phi\theta + \phi^T\theta$$

with the constraint:

$$\Omega\theta \leq \omega$$

This is the standard form of a *Quadratic Programming* (QP) problem. A number of algorithms exist which solve the QP problem.

Part III

Subspace Methods in Adaptive Control

Chapter 7

Online Subspace Identification

7.1 Introduction

System identification may either be performed 'offline' or 'online'. In the case of offline implementation information about the system is collected in the form of input-output data or frequency response data, and is processed in some way (effectively without constraints on computational complexity or computation time) to provide a model. An online implementation on the other hand involves processing the data (normally input-output data) as it is being collected. Parametric identification, for example of ARMAX models, is often performed online using a Recursive Least Squares (RLS) algorithm, which is a 'sample-by-sample' algorithm. Since subspace algorithms are block based, they are often implemented offline using an entire set of collected input-output data. However, an adaptive control algorithm requires an online identification method, and so if subspace algorithms are to be used for adaptive control, they must first be implemented in an online way.

7.2 Online POMOESP Implementation

A starting point for the investigation of online implementations was chosen as the `smi-1.0` toolbox written by Michel Verhaegen and Bert Haverkamp at Delft University, the Netherlands. The toolbox contains a variety of functions which implement the MOESP (Multiple Output-Error State Space) set of subspace algorithms.

These functions include `dordpo`, which returns a vector of singular values S and a lower triangular matrix R of information extracted from input-output data, `dmodpo` which calculates the system matrices A and C as well as the Kalman gain K from the aforementioned R matrix and the system order obtained from an appreciation of the singular values S, and `dac2bd`, which provides estimates of the B and C matrices. These functions implement the Past-Output version of the MOESP algorithm, known as POMOESP.

These functions were utilized in a Matlab s-function to produce a simulink

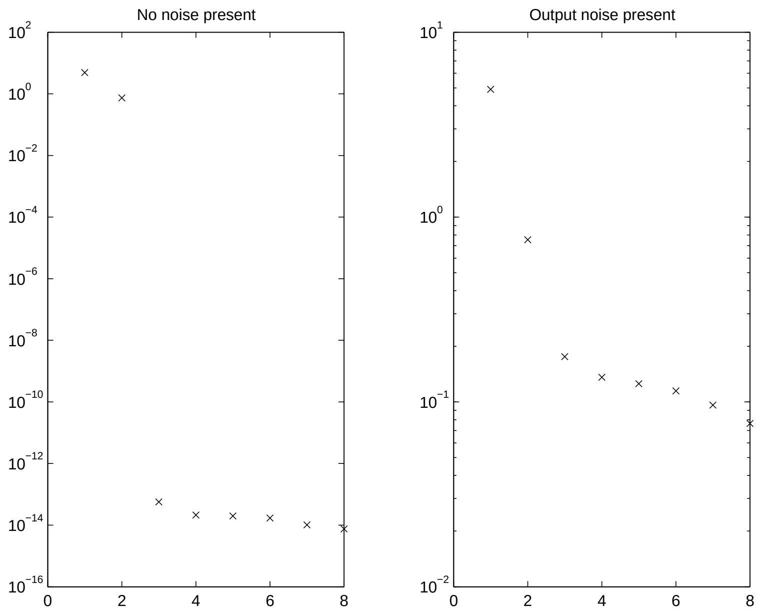

Figure 7.1: Singular value results from a second order MIMO process model in Simulink, showing the effects of output noise

block `POMOESP` which samples input-output data u and y, and outputs the obtained system order N and the identified transfer function parameters. The singular values are also plotted. It was decided (at least temporarily) to present the outputs in transfer function form because the algorithm produces a state space representation in a state space basis dependent on weighting matrices, which vary between algorithms. Thus, to confirm the estimated output it was useful to convert the identified state space system to transfer function form and compare this with the identification object (either a simulink state space or transfer function model or a real process, namely the Furuta inverted pendulum). The obtained system matrices and other information such as the R matrix are however saved as `.mat` files.

In order to automate the process it was necessary to obtain an heuristic algorithm for determining the system order from the singular values S. As a first iteration the system order was chosen by selecting the singular value which had the largest difference in magnitude between itself and the consecutive singular value. This was found to work poorly in noisy systems or when nonlinearities were present, since singular values due to these effects were often large, as can be seen from Figure 7.1. An improved version looks at the 'second differential' of the singular value plot, and sets the system order to be equal to the number of singular values below the point where the change of gradient is highest. The determination of system order from the singular values is an area which requires improvement, and much work remains to be done. It is a crucial stage of the subspace algorithm, since an incorrect choice of system order will result in poor identification results. A possible way of confirming the choice of system order could involve the use of statistical testing performed on estimated models of higher and lower order models than the suggested result. This could be implemented as some kind of 'background' process, running at a low priority, but would require a great deal of additional computational power compared to the standard algorithm.

7.3 An Example: The Furuta Inverted Pendulum

In order to test the implemented online algorithm on a real process, it was decided to identify a linear model for the small oscillations of the Furuta Inverted Pendulum process about its stable equilibrium. This section describes the process, presents the results of the identification, and compares the subspace method with other identification methods.

The process consists of a pendulum that has its point of rotation fixed to the end of a rod which rotates in a horizontal plane about an axis through its other end. The control signal is a motor voltage to the vertical shaft to which the aforementioned horizontal rod is attached. The measurable states of the system are the pendulum's angular position and angular velocity, denoted by θ and $\dot{\theta}$, as well as the angular position and angular velocity of the horizontal arm, denoted by ϕ and $\dot{\phi}$.

7.3.1 Complete Nonlinear Model

The equations of motion of the pendulum are:

$$a\ddot{\theta} - a\dot{\phi}^2 sin\theta \ cos\theta + c\ddot{\phi} \ cos\theta - dsin\theta = 0 \tag{7.1}$$

$$c\ddot{\theta} \ cos\theta - c\dot{\theta}^2 sin\theta + 2a\dot{\theta}\dot{\phi} \ sin\theta \ cos\theta + (b + asin^2\theta)\ddot{\phi} = k_u u \tag{7.2}$$

Where:

$$\begin{aligned} a &= J_p + Ml^2 \\ b &= J + Mr^2 + mr^2 \\ c &= Mrl \\ d &= lg(M + m/2) \end{aligned}$$

In the above equations l is the length of the pendulum, M is the mass of the weight at the end of the pendulum, m is the mass of the pendulum rod, r is the length of the horizontal rod, J is its moment of inertia and J_p is the moment of inertia of the horizontal rod.

7.3.2 Simplified Pendulum Model

By approximating the rotational motion in the horizontal plane by a linear motion, it is possible to model the system as an inverted pendulum on a cart, and thus simplify the model. From this model it is possible to obtain simplified system dynamics that can be used to design the controller. The resulting equations of motion are:

$$J_p \frac{d^2\theta}{dt^2} = Mgl \ sin\theta - Mla \ cos\theta \tag{7.3}$$

with:

$$a = \frac{d^2\phi}{dt^2}$$

Here a is the linear acceleration of the cart which is proportional to the angular acceleration of the horizontal arm in the actual process. The above equations can be written as:

$$\frac{d^2\theta}{dt^2} = \omega_o^2\ sin\theta - \omega_o^2 k_u u\ cos\theta$$
$$\frac{d^2\phi}{dt^2} = k_u u g$$

where:

$$\omega_o = \sqrt{\frac{Mgl}{J_p}}$$
$$k_u u = \frac{a}{g}$$

From Equation 7.3 we can perform linearizations around the two equilibrium positions and obtain from them the transfer funtions from motor voltage to pendulum angle.

The system is in a position of unstable equilibrium when the pendulum angle $\theta = 0$, that is, when the pendulum is upright. Linearization of Equation 7.3 around this position is done as follows.

Using the small angle approximations for $\theta \approx 0$:

$$sin\theta \approx \theta$$
$$cos\theta \approx 1$$

From Equation 7.3 we obtain

$$\ddot{\theta} = \omega_0^2\theta - \omega_0^2 k_u u$$

Taking Laplace Transforms and rearranging we obtain the transfer function

$$\frac{\theta(s)}{u(s)} = \frac{-\omega_0^2 k_u u}{s^2 - \omega_0^2} \tag{7.4}$$

which, as follows also from an intuitive approach, is an unstable transfer function.

The system is in a position of stable equilibrium when $\theta = \pi$, that is, when it is hanging in the downright position. Linearization of Equation 7.3 about this position is done in the same manner as above. Using the small angle approximations for $\theta \approx \pi$

$$sin\theta \approx -\theta$$
$$cos\theta \approx -1$$

From Equation 7.3 we obtain

$$\ddot{\theta} = -\omega_0^2\theta + \omega_0^2 k_u u$$

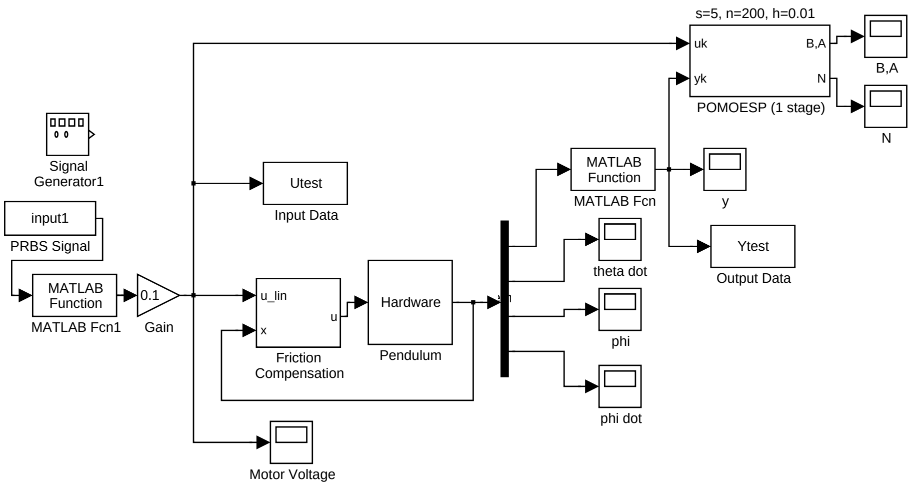

Figure 7.2: Simulink model for identification of the Furuta Inverted Pendulum process

Taking Laplace Transforms and rearranging we obtain the transfer function

$$\frac{\theta(s)}{u(s)} = \frac{\omega_0^2 k_u u}{s^2 + \omega_0^2} \tag{7.5}$$

7.3.3 Identification Results

The online POMOESP implementation was applied to the process in an attempt to obtain a second order model when operating in the approximately linear region (small θ) around the stable downright position.

Since it is necessary to maintain the pendulum within a small range of angles (arbitrarily chosen as ± 0.2 radians from the equilibrium position), smasll control inputs were therefore required. This posed problems related to friction, which is high around the arm pivot. It was decided to include friction compensation block (as can be seen in Figure 7.2) in order to cancel, as much as possible, the effects of friction and improve the identification. Clearly the friction block, if included, becomes part of the identification object along with the plant and thus must be included when any control is performed.

Figures 7.4, 7.5, 7.6 and 7.7 were obtained from square wave input with amplitude 0.3, frequency 1Hz and friction compensation coefficient $k = 0.1187$. In all the results presented here the dotted line represents the input, the dashed line the actual output and the solid line the simulated output. The data block size used in all the experiments was 200, the sampling interval was 10 milliseconds and the Hankel matrix block row size used was 10. In the POMOESP algorithm it is stated that this parameter should be approximately twice the maximum expected system order. In this case, the knowledge that a second order system should result from the identification was not used, and a maximum expected system order of 5 was supplied to the Simulink POMOESP block, yielding the figure 10 for the block row size of the Hankel matrices.

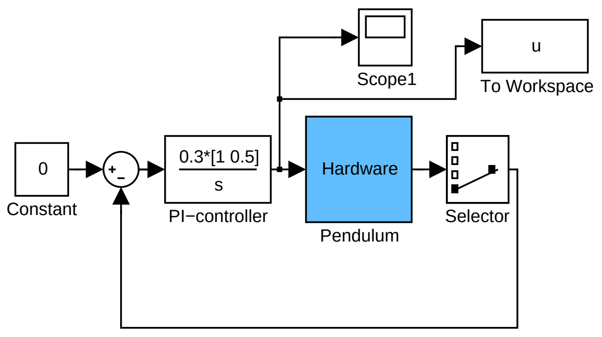

Figure 7.3: Simulink model for estimation of friction constant k for use in the friction compensation block

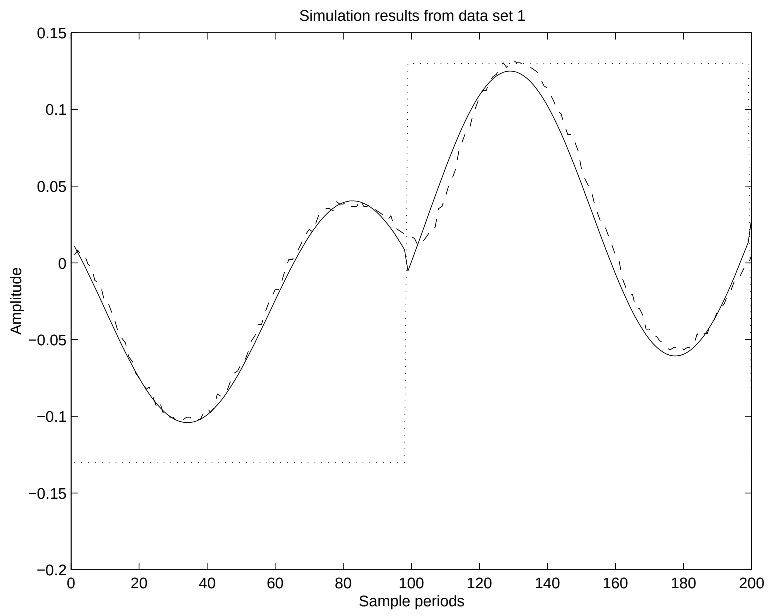

Figure 7.4: Simulation results. VAF 98.7984%

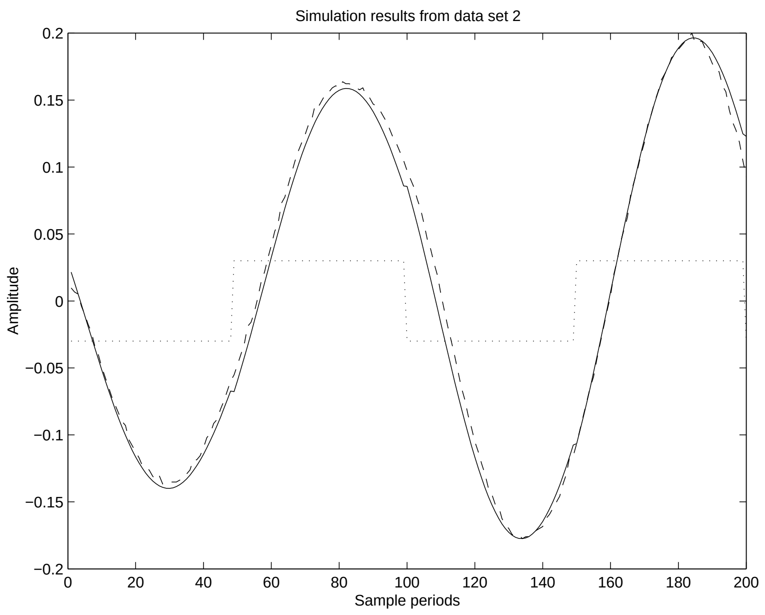

Figure 7.5: Simulation results. VAF 99.5493%

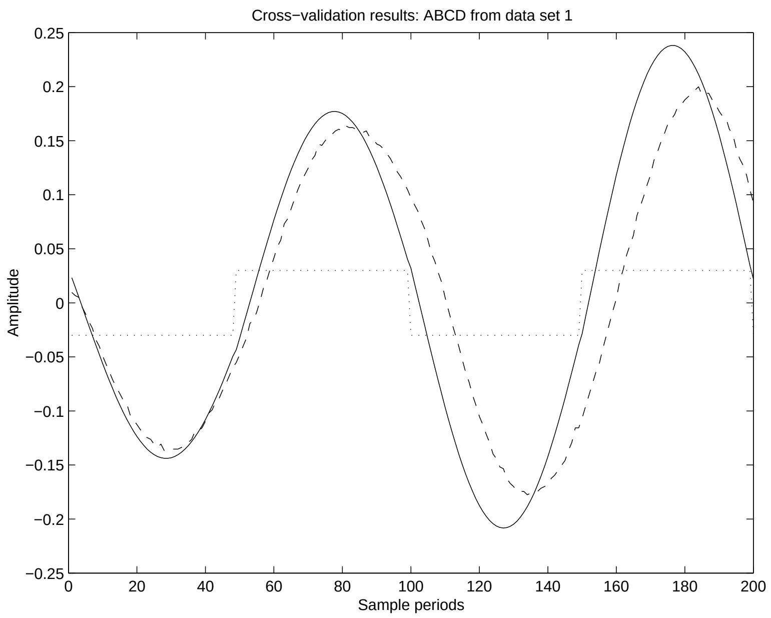

Figure 7.6: Cross-validation results. VAF 83.7963%

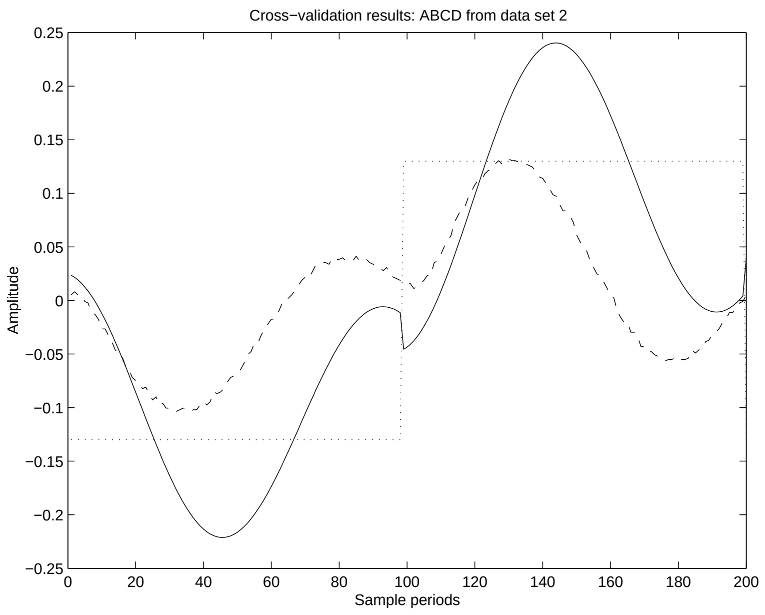

Figure 7.7: Cross-validation results. VAF 47.2825%

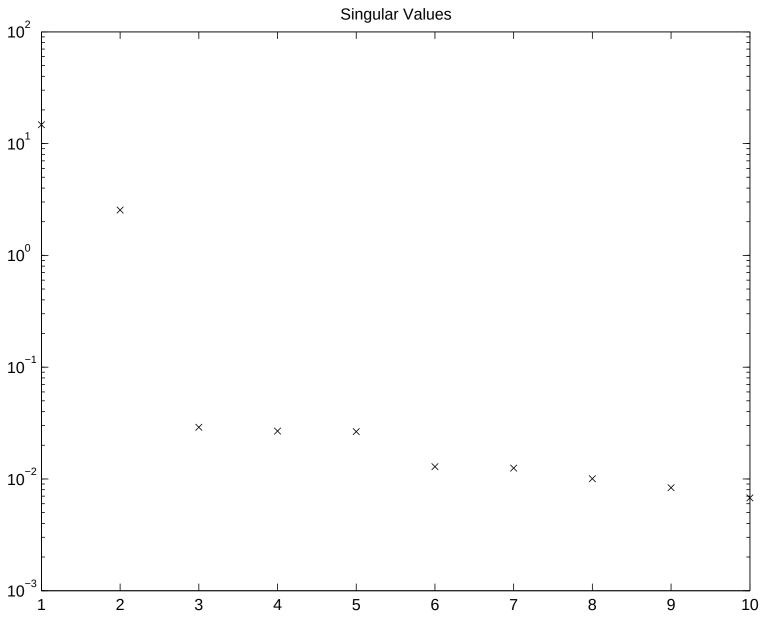

Figure 7.8: Singular Values obtained with square wave input

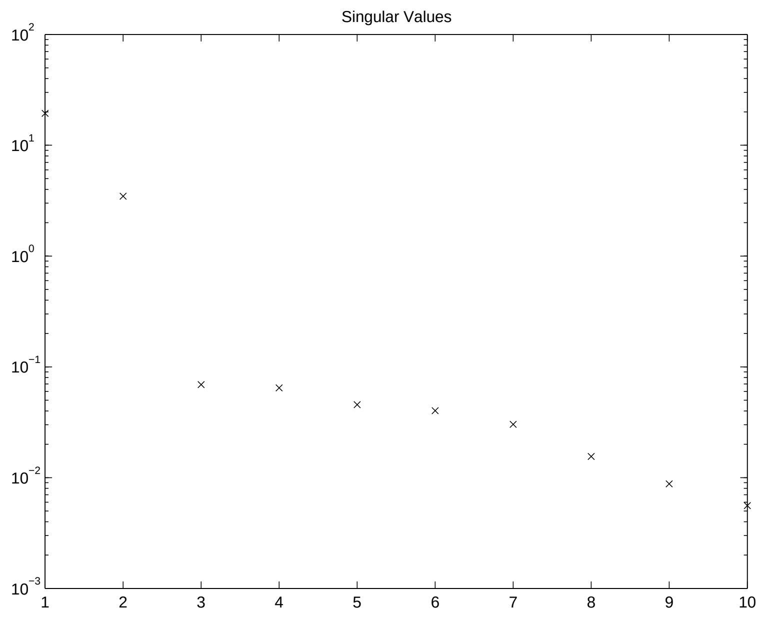

Figure 7.9: Singular values obtained with PRBS input

Figures 7.10, 7.11, 7.12 and 7.13 were obtained from Pseudo-Random Binary Signal (PRBS) input, with amplitude 0.1. For this simulation the friction constant used was $K = 0.1305$.

It can be seen that the simulation and cross-validation results for both square-wave and PRBS inputs are very good in most cases. This test highlights the ability of subspace algorithms to obtain accurate linear models of real systems despite the presence of such nonlinearities as friction. It should also be noted that the RLS solution to this identification problem is not straightforward, and in addition to requiring knowledge of the model order and structure, it is necessary to construct a modified regression model to obtain acceptable identification results.

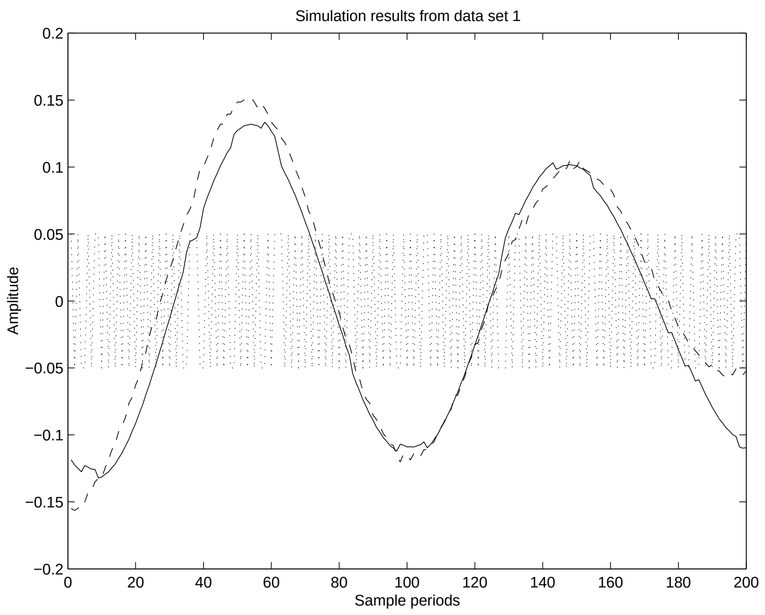

Figure 7.10: Simulation results. VAF 95.6641%

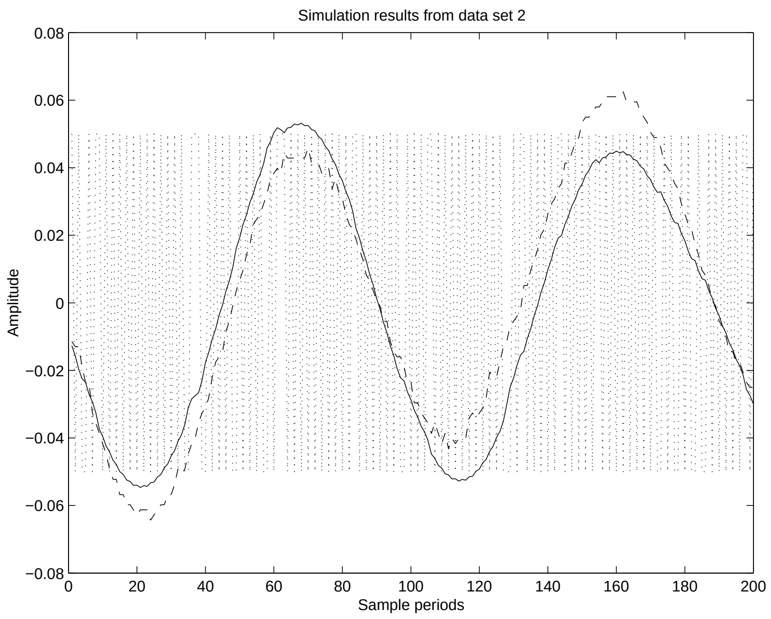

Figure 7.11: Simulation results. VAF 90.2902%

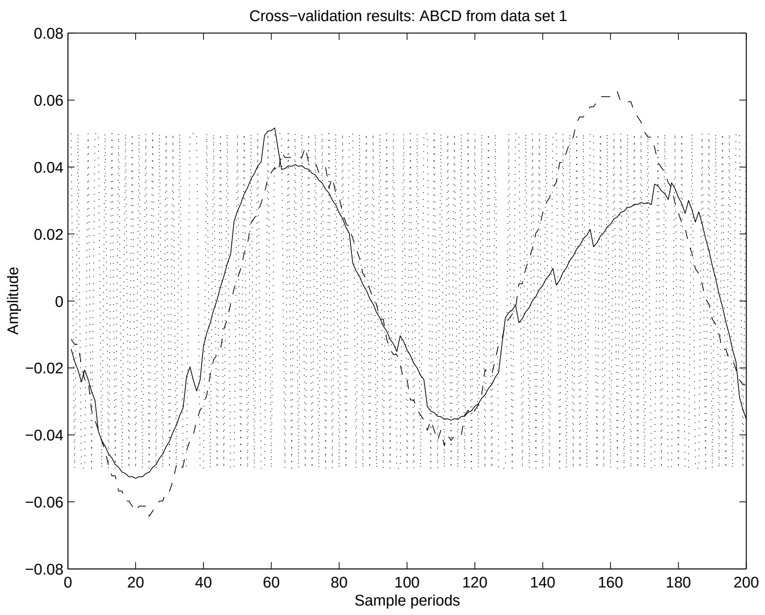

Figure 7.12: Cross-validation results. VAF 69.2815%

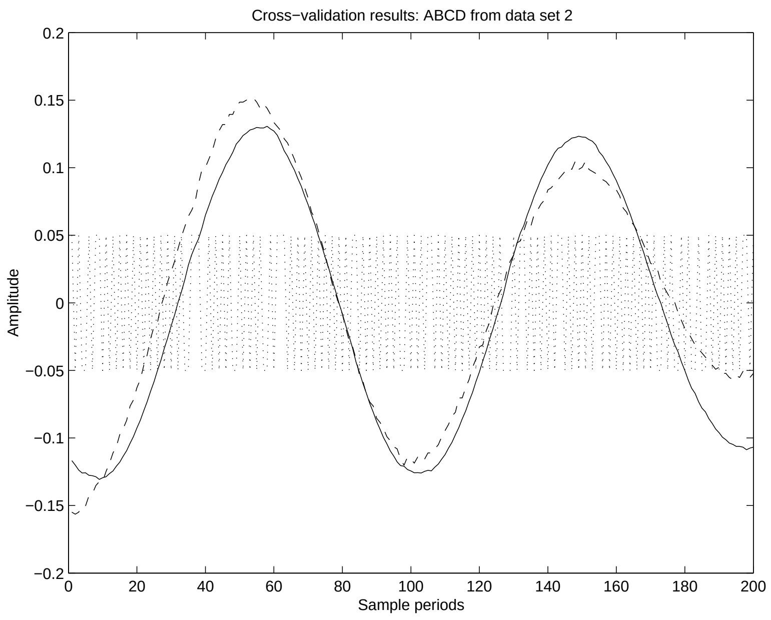

Figure 7.13: Cross-validation results. VAF 94.5587%

Chapter 8

Subspace Adaptive Control

8.1 Introduction

After obtaining an online implementation of subspace algorithms the next task is to devise a system structure for an adaptive controller. Traditional adaptive controllers using system identification elements often deal with input-output models, and perform controller design based on transfer function polynomials. With the use of subspace methods providing state-space representations, as well as information such as the Kalman gain, it would seem sensible to use state space control design methods to implement forms of indirect adaptive controllers. LQ control design is an obvious starting point, and in Section 8.2 this possibility is investigated further. The Model Predictive Control (MPC) methodology (as outlined in Chapter 6) is also well suited to the state-space control design problem, and adaptive control implemented with MPC design is explored in Section 8.3.

8.2 An Initial Structure

An intial structure could incorporate standard LQ design, implemented using a cost function of the form:

$$J = \sum y^T Q y + u^T R u$$

which may be implemented with the Matlab function `dlqry`. This function utilizes a quadratic cost on the outputs and the inputs and is therefore preferable to design methods where penalties are based on the states, since it is not possible for the user to input meaningful weighting matrices for the identified system's states. The inputs and outputs may have the necessary R and Q matrices assigned to them more easily.

A Kalman filter may be used to estimate the state, which is then multiplied by the state feedback control law to obtain the control signal. A block diagram of this structure suggestion is shown in Figure 8.1.

The main problem with this suggested controller structure was found to be the incorporation of a suitable reference signal. A reference state trajectory for the arbitrary state-space realization determined by the subspace identification

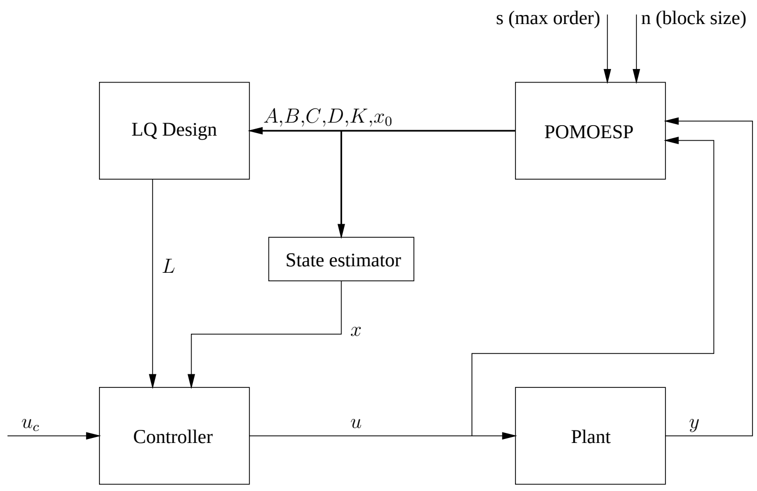

Figure 8.1: A possible structure of an adaptive controller using LQ design

cannot directly be specified, nor easily generated from a given output reference. A possible solution would be to incorporate a model inverse to generate the required control signal to give the specified output trajectory.

A second problem that comes to mind, but that was not formally proven nor shown empirically is the issue of identification in closed loop without a reference. A key assumption required for the correct operation of the subspace methods is that the intersection of the row space of the future inputs Uf and the row space of the past states X_p is empty. It can be seen that using a state feedback regulator as designed using LQ methods with the form:

$$u = -Lx$$

will violate this condition, and thus create problems for the identification stage.

Another rather more subtle issue is that the overall strategy is somewhat redundant, since it requires an observer to produce state estimates. Although the POMOESP algorithm provides the Kalman gain, the strategy lacks elegance in that the full capabilities of the subspace method are not used. In Chapter 9 we will reconsider this issue.

In summary, it was decided due to these shortcomings not to further investigate LQ-based control design for the adaptive controller. This is not to say that the strategy lacks potential, but it was decided that more effective, and perhaps more elegant strategies were more worthy of investigation.

8.3 Indirect Adaptive Control Using MPC

The model predictive control strategy is an interesting alternative for the control design method. It allows the incorporation of not only reference trajectories but also constraints on control signals and outputs. It was decided to implement an indirect adaptive control scheme using the online POMOESP algorithm described in the previous chapter along with MPC tools developed by Johan

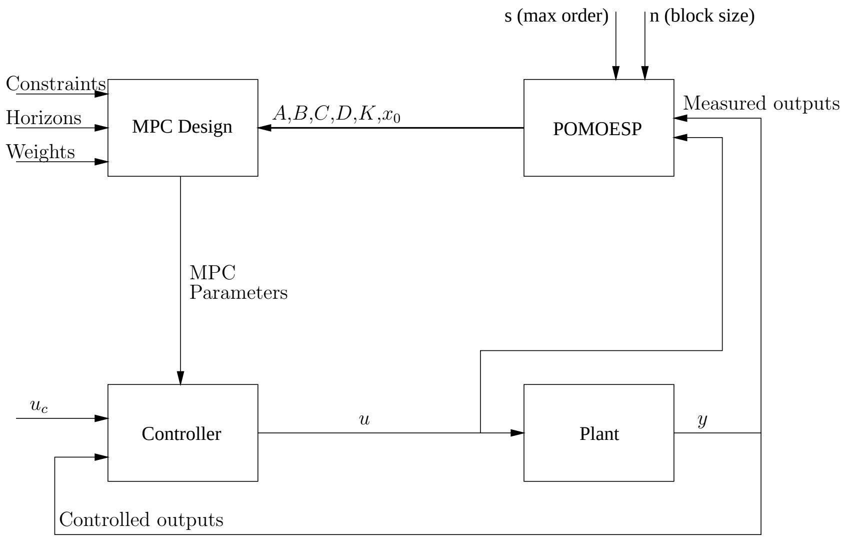

Figure 8.2: An adaptive controller structure incorporating MPC design

Åkesson at LTH. These MPC tools allow the solution of control problems with constraints on $\Delta u(k)$ (effectively limiting the 'slew rate' of the control signal), $u(k)$ (absolute limits on control signal values) and $z(k)$ (the controlled outputs). The solution follows the formulation in Chapter 6, and involves the solution of a quadratic programming problem. A block diagram of the proposed system can be seen in Figure 8.2.

The implemented algorithm uses the system matrix estimates provided by the POMOESP identification algorithm to set up the prediction matrices as in Chapter 6. This step is normally performed offline in the case where the control object is known, and represents additional computations that must be performed after each set of input-output block data is collected. A Simulink MPC controller block was modified to accept the system matrix estimates from the POMOESP block. The current implementation does not utilize the estimated Kalman gain K provided by the POMOESP algorithm, but this is an obvious extension.

The control strategy was implemented in a Simulink model, illustrated in Figure 8.3. To test the system, a simple control object was chosen. The unstable second order system given by:

$$G(s) = \frac{11.2}{s^2 + 0.12s}$$

was used. It was found that the system could be stabilized and controlled rather well, as may be seen in Figures 8.5 and 8.6.

A number of issues arose with this system. One of particular interest is that different sampling periods may be used for the identification and control stages. This functionality was implemented in the modified MPC controller block, by resampling the identified system matrices using the Matlab command `d2d`. Using different sampling rates in this manner is an attractive prospect since it allows the system matrices to be estimated at a higher rate (recall that there is in some sense a lower bound on the column size of the Hankel matrices

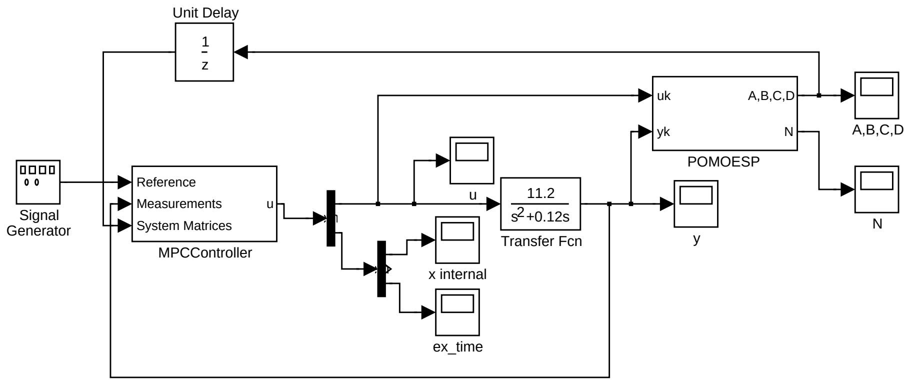

Figure 8.3: Simulink model implementing the combined POMOESP-MPC indirect adaptive controller

to ensure accurate identification results). It is also advantageous to have a relatively long sampling period for the control, since a QP problem must be solved at each time step.

Unfortunately, it was found that the control performance deteriorated as the difference in the relative sizes of the identification and control sampling intervals increased. This can be seen in the simulation results presented here. Figure 8.5 shows the results when the sampling periods for control and identification were equal, i.e. $h_c = h_{id} = 0.1$. The performance is good. However, Figure 8.6 shows the results when the identification sampling rate was set to $h_{id} = 0.01$ and the control rate to $h_c = 0.1$. The results are clearly less satisfactory than in Figure 8.5, with unexplained control activity after the step changes, where the control signal ought to be small. This is undesirable, since in this case the smaller value of h_{id} is clearly desirable for dealing with parameter changes and preventing large initial transients, occurring when the reference signal is passed through as the control, providing excitation for the first set of system matrix estimates. Increasing the relative difference in size between the control and identification sampling periods gave increasingly poor results. Indeed, the system was unstable for values of h_c above 0.1. This was disappointing because the simulations ran very slowly at this sampling rate, due to the need to solve a quadratic programming problem at each time step. While this may be an issue specific to this example, the lack of robustness with respect to the choice of sampling intervals is concerning.

Another issue that arises deals with the solution to the QP problem. It can occur that the problem is infeasible, in which case no solution is found. This is of course highly undesirable for an online control algorithm, and it implies that steps must be taken to provide some form of 'backup strategy' in this eventuality.

A more general issue with this strategy, and indeed MPC in general, is the large number of tuning parameters. Weights must be assigned to outputs and control signals (although since these are relative the problem reduces somewhat). In theory at least, these weights may also be time variant with respect to the control and prediction horizons, complicating the matter greatly. In this

Parameter	**Description**
Identification	
h_{id}	Sampling interval
N	Block data size
s	Block row size of Hankel matrices
Control	
h_c	Sampling interval
H_p	Prediction horizon
H_w	First included sample
H_u	Control horizon
$\mathcal{Q}$	Output weighting matrices
$\mathcal{R}$	Input weighting matrices
Δu_{min}, Δu_{max}	Limits on control changes
u_{min}, u_{max}	Absolute limits on control
z_{min}, z_{max}	Limits on outputs

Figure 8.4: The parameters to be chosen in the POMOESP-MPC adaptive control scheme

investigation however, the weights were assigned to be constant over the horizons. If the limits on the control signals are not 'hard' limits, they also may be considered as tuning parameters. Indeed, this is related to the problem of solving the QP problem discussed above; overly-stringent limits may make the problem infeasible, and prevent the calculation of the control sequence. Such behavior was observed during simulations, and in some cases the repeated infeasibility of the QP problem to be solved (and the lack of a 'reserve' strategy) resulted in destabilization of the system.

In addition to the control design parameters, various parameters must be chosen for the identification stage, in particular the sampling period, the block data size (which determines the column size of the Hankel matrices) and the block row size of the Hankel matrices. The parameters to be chosen with the implemented strategy are summarized in Figure 8.4.

While there are many parameters to be chosen and tuned, it must be remembered that this scheme may be applied to any system without any design work, which is in contrast to current adaptive control strategies where, for instance, parameter update laws must be derived for different systems. The prior information required about the system in question is also much lower than current strategies. It would seem reasonable to state these as the primary advantages of the proposed scheme. Another very important point is that this scheme may be extended to MIMO systems without any difficulties (although the selection of weighting matrices will become more complicated). This represents a great advantage over current schemes, where the extension to MIMO systems is no trivial task.

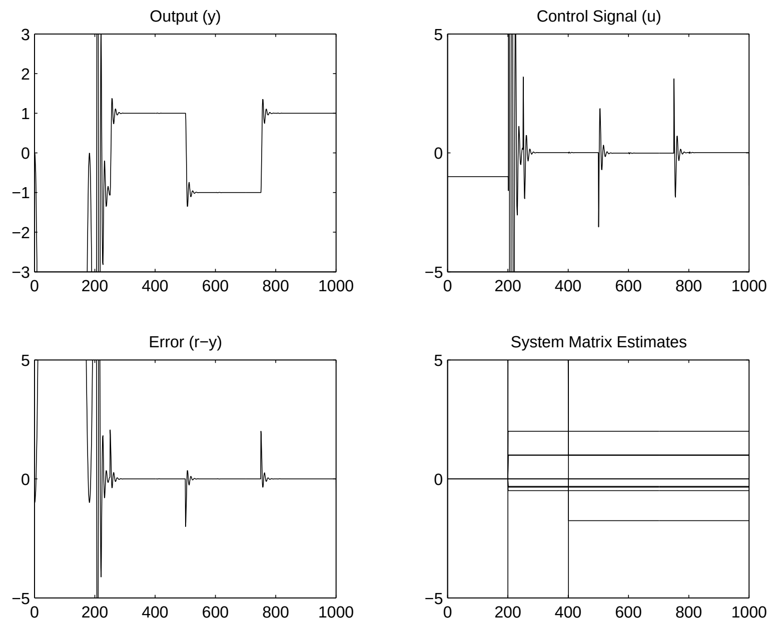

Figure 8.5: Simulation results using $h = 0.1$ for both the identification and control

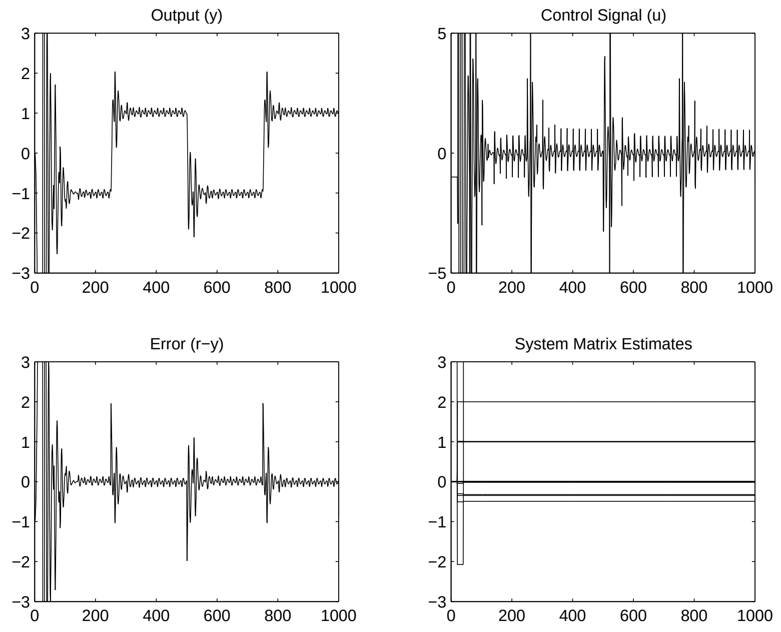

Figure 8.6: Simulation results using $h = 0.1$ for control and $h = 0.01$ for identification

Chapter 9

Subspace Predictive Control

9.1 A Relation between Subspace Model Identification and MPC

In [3] an interesting link between subspace identification and model predictive control is exploited to implement a predictive controller for an unknown system based on subspace techniques. The algorithm presented does not require computation of the system matrices, and as such this method of control design can be interpreted as 'direct' adaptive control. The derivation of the algorithm is outlined below. In addition, a modification was made to the algorithm, and this modified algorithm was successfully implemented in Simulink.

9.1.1 The Subspace Identification Problem

Recall the input-output representations derived earlier, namely:

$$Y_f = \Gamma_N X_f + H_N^d U_f + H_N^s E_f \tag{9.1}$$

$$Y_p = \Gamma_M X_p + H_M^d U_p + H_M^s E_p \tag{9.2}$$

where the block Hankel matrices are definied in the usual way:

$$U_p = \begin{pmatrix} u_1 & u_2 & \dots & u_j \\ u_2 & u_3 & \dots & u_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_M & u_{M+1} & \dots & u_{M+j-1} \end{pmatrix}$$

$$U_f = \begin{pmatrix} u_{M+1} & u_{M+2} & \dots & u_{M+j} \\ u_{M+2} & u_{M+3} & \dots & u_{M+j+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{M+N} & u_{M+N+1} & \dots & u_{M+N+j-1} \end{pmatrix}$$

in addition to the system related matrices:

$$\Gamma_N = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{pmatrix}$$

$$H_N^d = \begin{pmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ CA^{N-2}B & CA^{N-3}B & CA^{N-4}B & \dots & D \end{pmatrix}$$

$$H_N^s = \begin{pmatrix} I_l & 0 & 0 & \dots & 0 \\ CK & I_l & 0 & \dots & 0 \\ CAK & CK & I_l & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ CA^{N-2}K & CA^{N-3}K & CA^{N-4}K & \dots & I_l \end{pmatrix}$$

Recall also the shorthand notation

$$W_p = \begin{pmatrix} Y_p \\ U_p \end{pmatrix}$$

A key interpretation of subspace algorithms is that an estimate of the future outputs $\hat{Y}_f$ lies in the combined row space of the past inputs and outputs W_p and the future inputs U_f (when no noise is present the actual future outputs Y_f lie in this combined row space). This may be written as:

$$\hat{Y}_f = L_w W_p + L_u U_f \tag{9.3}$$

Thus a least squares problem may be formulated to obtain this prediction. This may be expressed as:

$$\min_{Lw,Lu} \left\| Y_f - \begin{pmatrix} L_w & L_u \end{pmatrix} \begin{pmatrix} W_p \\ U_f \end{pmatrix} \right\|_F^2 \tag{9.4}$$

Geometrically, the solution of the least squares problem is the orthogonal projection (see Appendix A.1.1) of Y_f onto the combined row space of W_p and U_f:

$$\hat{Y}_f = Y_f / \begin{pmatrix} W_p \\ U_f \end{pmatrix} = Y_f \begin{pmatrix} W_p \\ U_f \end{pmatrix}^{\dagger} \begin{pmatrix} W_p \\ U_f \end{pmatrix} \tag{9.5}$$

This projection can be implemented using an RQ decomposition, as in [3],[26] which gives:

$$\begin{pmatrix} W_p \\ U_f \\ Y_f \end{pmatrix} = \begin{pmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \end{pmatrix} \tag{9.6}$$

from which we obtain:

$$\hat{Y}_f = L \begin{pmatrix} W_p \\ U_f \end{pmatrix} \tag{9.7}$$

where

$$L = \begin{pmatrix} R_{31} & R_{32} \end{pmatrix} \begin{pmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{pmatrix}^{\dagger} \tag{9.8}$$

Thus the matrices L_w and L_u can be obtained by partitioning L in the following way:

$$\begin{aligned} L_w &= L_{\text{first } N(m+l) \text{ columns}} \\ L_u &= L_{\text{columns } N(m+l)+1 \text{ to } N(2m+l)} \end{aligned}$$

Alternatively, this may be written in Matlab notation as:

$$\begin{aligned} L_w &= L(:, 1 : N(m+l)) \\ L_u &= L(:, N(m+l)+1 : N(2m+l)) \end{aligned}$$

At this stage, the algorithm presented in [3] states that the matrix L_w should be approximated with a lower order matrix using a singular value decomposition. The reason given is that L_w would be a rank deficient matrix of order n in the absence of noise, and the approximation is made to remove some of the noise. The SVD:

$$L_w = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_{11} & 0 \\ 0 & \Sigma_{22} \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$$

is used to make the approximation:

$$L_w \approx U_1 \Sigma_1 V_1^T \tag{9.9}$$

The system order n is obtained by observing the number of dominant singular values, in this case the number of singular values in the matrix Σ_{11}. In the following section further reasoning will be provided as to why this approximation of L_w may be made.

The suggested modification to the algorithm is that L_w should not be approximated. Several reasons contributed to this decision, some of which will be described here, and the remainder in the next section. It was found that the estimate of the system order using the method described above was poor, compared to the method used in for example the POMOESP algorithm, described earlier. Another important reason for not approximating L_w by a smaller matrix is that its size determines the backwards horizon in the prediction problem, i.e. the number of past samples used. If the matrix is approximated, this horizon will become very small. Using the original matrix, the horizon remains at the level specified by the user, as the block row size of the Hankel matrices.

9.1.2 Model Predictive Control Problem

The model predictive control problem that we wish to solve in this case can be stated as follows.

Problem Formulation 9.1.1. *Given a reference output $\{r_k\}_{k=1}^N$ up to a forward horizon N, find a corresponding control sequence $\{u_k\}_{k=1}^N$ yielding the future predicted outputs $\{\hat{y}_k\}_{k=1}^N$ that minimizes the cost function:*

$$J = \sum_{k=1}^{N} (\hat{y}_k - r_k)^T Q_k (\hat{y}_k - r_k) + u_k^T R_k u_k \tag{9.10}$$

where $Q_k \in \mathbb{R}^{l \times l}$ and $R_k \in \mathbb{R}^{m \times m}$ are user defined weighting matrices for the time step k.

In this receding-horizon problem only the first control input, u_1 will be applied. A new control sequence will be calculated at every time step. For this reason it is useful to minimize the computational complexity of the control calculations.

Defining the future signals U_f, $\hat{Y}_f$ and R_f as:

$$U_f = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}, \quad \hat{Y}_f = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{pmatrix}, \quad R_f = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{pmatrix}$$

it is possible to write Equation 9.10 on matrix form as:

$$J = (\hat{Y}_f - R_f)^T Q_N (\hat{Y}_f - R_f) + U_f^T R_N U_f \tag{9.11}$$

where the matrices $Q_N \in \mathbb{R}^{Nl \times Nl}$ and $R_N \in \mathbb{R}^{Nm \times Nm}$ have the form:

$$Q_N = \begin{pmatrix} Q_1 & 0 & \dots & 0 \\ 0 & Q_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q_N \end{pmatrix}, \quad R_N = \begin{pmatrix} R_1 & 0 & \dots & 0 \\ 0 & R_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R_N \end{pmatrix}$$

Note that the weighting matrices are now effectively time variant with respect to the prediction horizon, i.e. different weights can be applied to inputs and outputs at different time steps.

The predicted future outputs $\hat{Y}_f$ can be found from Equation 9.4 as:

$$\hat{Y}_f = L_w W_p + L_u U_f \tag{9.12}$$

where W_p is defined as:

$$W_p = \begin{pmatrix} Y_p \\ U_p \end{pmatrix}$$

with:

$$Y_p = \begin{pmatrix} y_{-M+1} \\ y_{-M+2} \\ \vdots \\ y_0 \end{pmatrix}, \quad U_p = \begin{pmatrix} u_{-M+1} \\ u_{-M+2} \\ \vdots \\ u_0 \end{pmatrix}$$

From Equation 9.12 the matrix cost function in Equation 9.11 may be written as:

$$J = (L_w W_p + L_u U_f - R_f)^T Q_N (L_w W_p + L_u U_f - R_f) + U_f^T R_N U_f$$

The minimum of this cost function may be found by differentiating J with respect to the future control sequence U_f and setting the trace of this derivative to zero:

$$\mathrm{tr} \frac{\partial J}{\partial U_f} = 0$$

This gives:

$$L_u^T Q_N(L_w W_p + L_u U_f - R_f) + R_N U_f = 0$$

Which in turn yields the optimal future control sequence:

$$U_f = (R_N + L_u^T Q_N L_u)^{-1} L_u^T Q_N (R_f - L_w W_p) \tag{9.13}$$

As previously stated, only the first term in this sequence will be implemented. A new optimal sequence will be recalculated at the next time step. The first element is given by:

$$u_1 = -K_w^c W_p + K_r^c R_f \tag{9.14}$$

where:

$$K_r^c = [(R_N + L_u^T Q_N L_u)^{-1}]_{\text{first m rows}} L_u^T Q_N \tag{9.15}$$

$$K_w^c = K_r^c L_w \tag{9.16}$$

At this point we may return to the issue of approximating L_w, first considered in the previous section. It can be shown that estimates of the extended observability matrix Γ_N and the future state sequence $\hat{X}_f$ can be obtained from Equation 9.9, as:

$$\Gamma_N = U_1 \Sigma_1^{1/2} \tag{9.17}$$

$$\hat{X}_f = \Sigma_1^{1/2} V_1^T W_p \tag{9.18}$$

Consider the standard form of a state feedback controller:

$$u(k) = -Kx(k)$$

It is not necessarily obvious that Equation 9.14 fits this form. However, approximating L_w, and using Equation 9.17 and Equation 9.18, we may write:

$$\begin{aligned}\Gamma_N \hat{X}_f &= U_1 \Sigma_1 V_1^T W_p \\ &= L_w W_p\end{aligned}$$

From Equation 9.16 and Equation 9.14, ignoring for the moment the reference sequence, we may also write:

$$\begin{aligned}u_1 &= -K_r^c L_w W_p \\ &= -K_r^c \Gamma_N \hat{X}_f \\ &= -K \hat{X}_f\end{aligned}$$

Thus by making an approximation of L_w it is possible to show that the control law fits the standard form of a state feedback controller. However, it is proposed that the primary importance of L_w is to provide prediction, and as such it should not be approximated in such a way that the horizons involved are reduced. Of course, using this modified algorithm requires more computation from sample to sample since the matrices are larger. It does however avoid the need for an SVD to be performed during calculation of the prediction and control matrices.

From an implementation point of view, the algorithm is attractive, since the RQ decomposition used to calculate the matrices L_w and L_u is performed

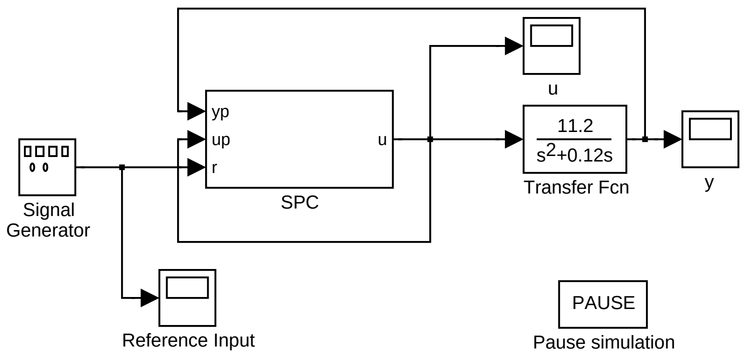

Figure 9.1: Simulink model used to implement SPC on an unstable second order SISO system

only when a new set of input-output data is obtained (typically a large number of samples). The controller matrices K_r^c and K_w^c also only require calculation once in this period. The first term of Equation 9.14 may be precalculated in an 'update-state' procedure, since it depends on information up to the time $k-1$, where u_k is the control output to be calculated. Thus only the term $K_r^c R_f$ must be calculated and added to $-K_w^c W_p$ before the control signal is available.

One issue arising with the use of block based identification for adaptive control is that of response to parameter changes. When a rapid parameter change occurs, it is reasonable to assume that the input-output data block in which the change occurred will not provide meaningful identification results (or in the case of SPC, prediction matrices L_w and L_u and thus control matrices K_r^c and K_w^c). This implies that any control designed using these results will be poor at best, and at worst could lead to system instability (on the subject of stability of adaptive systems, few results are available even for established algorithms. In this project there has been no investigation of stability of the systems proposed).

A possible solution to this problem could be to perform identification on 'overlapping' data sets, or in other words to use some form of window. From an identification point of view, this implies redundant calculations, since the data in one of the new 'shorter' sampling periods will be reused several times by the identification algorithm. However the time delay from a parameter change to the generation of accurate identification results will be much shorter. The decision of what scheme to use would depend highly on the sampling period required, the computational power available, and also the likelyhood of such a parameter change.

9.2 An Example

To test the implemented subspace predictive controller a simulink model was created to control an unstable second order SISO system. The model used is shown in Figure 9.1, and the SPC controller subsystem can be seen in Figure 9.2.

The system was simulated to test the effects of a sudden parameter change.

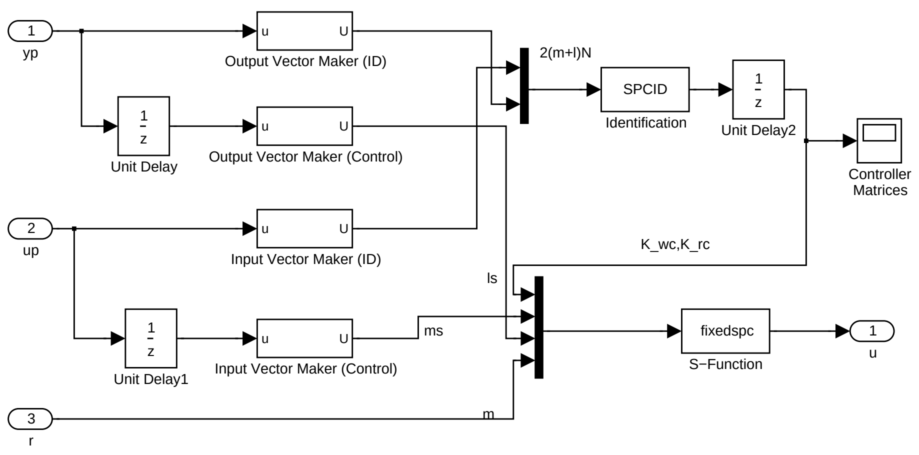

Figure 9.2: SPC Subsystem Simulink model

The system used was the same as that in the STR example in Chapter 5, namely:

$$G(s) = \frac{11.2}{s^2 + 0.12s} \tag{9.19}$$

In the simulation the following parameters were used:

Block size N	100
Hankel Matrix index/MPC horizon s	10
Sampling Interval h	0.1
Q matrix	$400I_{s\times s}$
R matrix	$10I_{s\times s}$

It was found that when the dynamics were changed during the simulation from the original system to:

$$G(s) = \frac{11.2}{s^2 + 12s} \tag{9.20}$$

the system responded well, with the control continuing to stabilize the system before the identification had collected a new set of input-output data and calculated new control matrices. Figure 9.3 shows the results of this simulation. However, when the simulation was started with the dynamics as in Equation 9.20, and the parameter was changed back to the original dynamics in Equation 9.19, the system became badly unstable for a number of sample periods before being stabilized again by the newly obtained controller matrices. The extent of the instability is shown in Figure 9.4, although it should be noted that there was no limitation on the control, and a rather unrealistically low control weighting matrix was used, for the purposes of experiment. Thus the control signals obtained were very large. (This is a good example of why the introduction of the MPC methods for dealing with constraints should be introduced to the basic SPC formulation; a higher control weighting in the cost

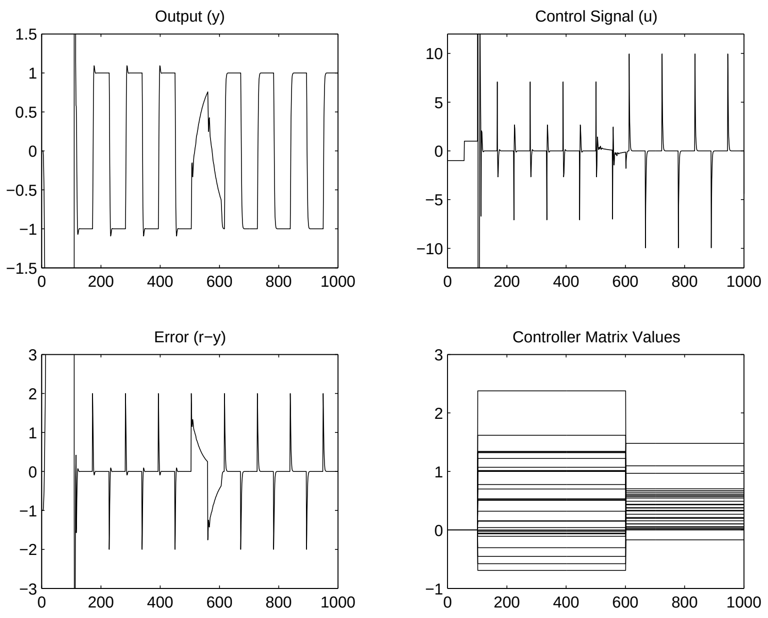

Figure 9.3: Parameter change from original dynamics to new dynamics at t = 50s (500 samples)

function would negatively affect the control performance). Figure 9.5 shows that the system is nevertheless restabilized by the new control.

A possible explanation for why the instability occurs during one transition and the other may be seen by an intuitive look at the systems in question. When the parameter change is from the original dynamics to the new dynamics (the case in which the system remains stable during the change), the dynamical change is one from slow to fast dynamics. It would appear that the designed control handles such a transfer better than in the case where the parameter transition goes the other way.

It is proposed that these instability problems arise because of the finite horizon nature of MPC. Concepts of stability are intrinsically asymptotic, and thus control designed for a finite horizon cannot have guaranteed stability inn the same way as is found in LQG control design, for example. Thus it is unknown if such instability windows will arise if the system (or the simulation) is run for an arbitrary length of time. This is clearly a disadvantage of MPC strategies.

The relatively small block data size (Hankel matrix block column size is 81) appears to give fairly poor identification in comparison with larger block sizes. But larger block sizes imply longer delays in responses to parameter changes, and can only be compensated for with smaller sampling periods, which increases the requirement on computational power.

Another possible disadvantage is the tuning of the parameters. In standard MPC, the parameters to be tuned include the weighting matrices Q and R, which are able to allocate different weights to different time steps within the forward horizon. This implies that the tuning procedure is very difficult, even for SISO systems. Indeed, it was found that tuning these matrices for the simple

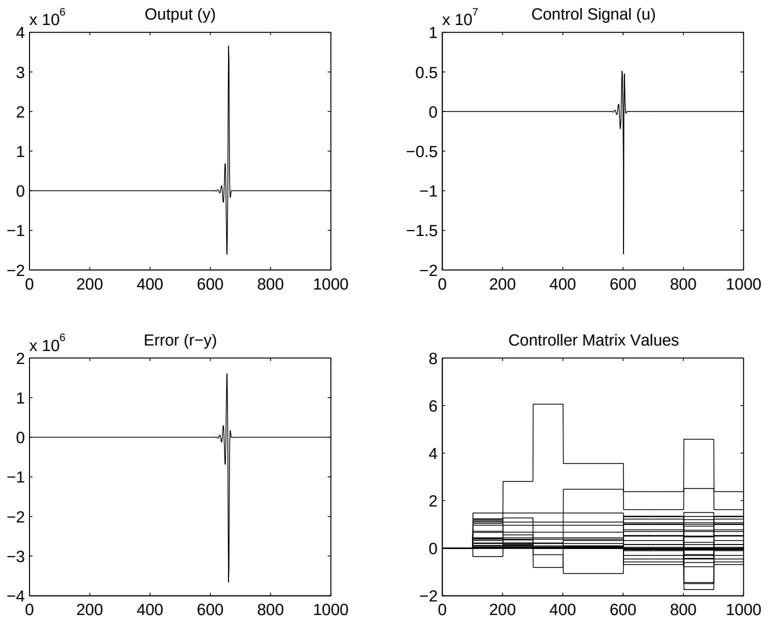

Figure 9.4: Parameter change from new dynamics to original dynamics at t = 50s (500 samples)

example above to obtain adequate control was not easy. It may therefore be concluded that the tuning process for MIMO systems would be very complicated indeed (Look at maciejowski for more details).

In SPC, further parameters must be chosen, namely, the size of the block data and the row size of the block Hankel Matrices (which in turn yields the backward and forward horizons). While these do not appear to be control parameters, they affect the identification result and thus the control design, and therefore have an effect on stability and performance of the system.

Figure 9.6 shows the results of a parameter change in the middle of a block of identification data. It can be seen that incorrect controller matrices are calculated from this set of data. This illustrates the point made previously about the use of some form of sliding window for the identification in order to minimize the time between the occurrence of a parameter change (or perhaps some disturbance) and the calculation of correct identification or control data.

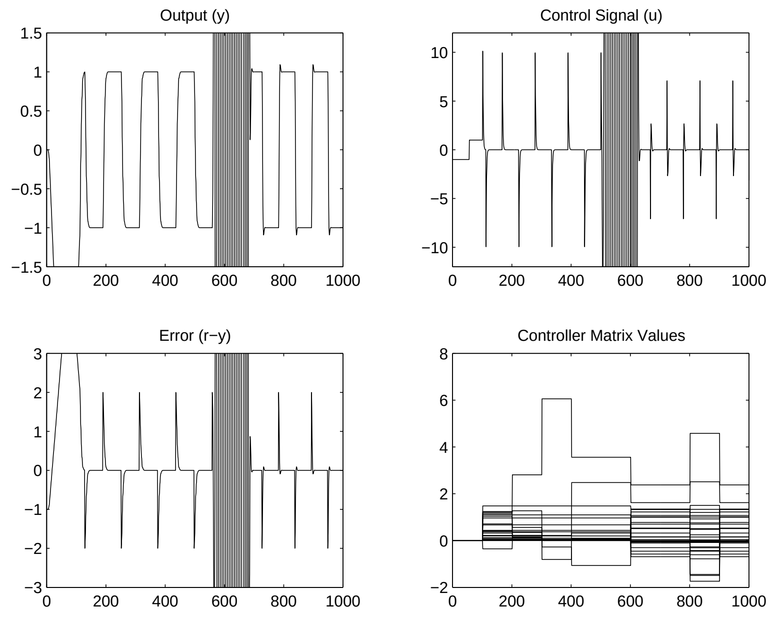

Figure 9.5: Parameter change from new dynamics to original dynamics at t = 50s (500 samples)

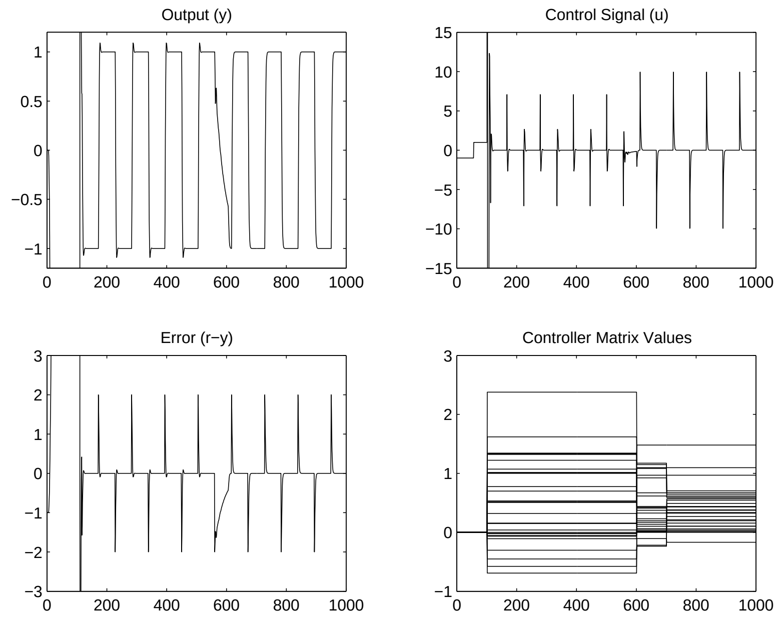

Figure 9.6: Parameter change from original dynamics to new dynamics at t = 55s (550 samples)

Chapter 10

Conclusions

10.1 Summary of Algorithms

In this section the algorithms that were derived and implemented in previous chapters will be summarized. The following sections will discuss some of the properties and relative merits of these systems.

Two subspace-adaptive control algorithms were proposed in this thesis. The first is an indirect adaptive controller utilizing the POMOESP subspace algorithm adapted for online use as well as a model predictive controller, capable of dealing with constraints on variables. The second is a direct scheme derived by considering parallels between the formulations of subspace identification and model predictive control.

10.1.1 POMOESP-MPC Indirect Adpative Controller

This algorithm involves the explicit calculation of the system matrices using a subspace method followed by control design using the MPC methodology. The algorithm can be summarized as:

- Collect a set of input-output data of length N
- Utilize the POMOESP algorithm as in Chapter 7 every $N \times h$ sample periods to obtain a new set of system matrix estimates $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$
- Set up the MPC prediction matrices as in Chapter 6
- Solve the model predictive control problem as in Chapter 6 at every time step
- Implement the first element in the calculated optimal control sequence

10.1.2 SPC Direct Adpative Controller

This algorithm combines ideas from subspace identification and MPC to provide a direct adaptive control strategy. The system matrices are not directly estimated, but subspace techniques are used to obtain prediction matrices, which are then used to solve an unconstrained MPC problem. This algorithm may be summarized as:

- Collect a set of input-output data of length N
- Form Hankel matrices of past and future inputs and outputs U_p, U_f, Y_p, Y_f,
- Obtain prediction matrices L_w and L_u through the use of an RQ decomposition solving the least squares problem:

$$\min_{Lw,Lu} \left\| Y_f - \begin{pmatrix} L_w & L_u \end{pmatrix} \begin{pmatrix} W_p \\ U_f \end{pmatrix} \right\|_F^2$$

- Calculate the control matrices K_r^c and K_w^c, given by:

$$\begin{aligned} K_r^c &= [(R_N + L_u^T Q_N L_u)^{-1}]_{\text{first m rows}} L_u^T Q_N \\ K_w^c &= K_r^c L_w \end{aligned}$$

- Calculate the first element of the optimal control sequence:

$$u_1 = -K_w^c W_p + K_r^c R_f$$

10.2 Discussion

In this section a qualitative appraisal of both of the proposed subspace-adaptive algorithms will be made. There relative merits will be discussed, both measured against each other and existing algorithms. Unfortunately, due to time constraints it was not possible to accomplish a quantative analysis of the computational complexity of the proposed algorithms.

10.2.1 Comparison of Proposed Algorithms

A direct comparison between the proposed algorithms is somewhat difficult since they fall into different categories, namely one is a direct algorithm and the other is indirect. Nevertheless some comments may be made regarding their generality, computational complexity and elegance.

The POMOESP-MPC indirect algorithm represents perhaps the more obvious method of solving the subspace adaptive control problem. The system matrices are estimated and used in an existing control design methodology. This system is perhaps the more flexible, since any subspace algorithm may be substituted for the POMOESP used in this case. It was seen empirically that the computational complexity of this algorithm was much greater than that of the SPC direct algorithm, when controlling the same system with the same sampling period.

The SPC direct adaptive algorithm is arguably the more elegant of the two. For 'faster' systems where computation time is an issue, it would be a superior choice. The primary reason for the reduced complexity of this algorithm is the absence of a quadratic programming problem to be solved at every time step. This comes at a price, since the algorithm is not able to handle constraints on variables, as the POMOESP-MPC algorithm can.

A major issue befalling both algorithms is the problem of guaranteed stability. With the SPC algorithm, providing a simple solution to a finite horizon predictive control problem, there is no guarantee of stability. There is presumably nothing to be done about this; stability is inherently an asymptotic property and a solution calculated over a finite horizon cannot provide asymptotic guarantees. This problem was observed during simulations, in the form of 'instability windows', instances where the system became unstable, but was then restabilized. It may be possible to describe the stability of the system using some optimal performance index. A cost function specified over infinite time may be rewritten as a cost over a finite time plus an additional term, known as a 'cost-to-go' term:

$$\begin{aligned} J(u^*) &= \int_0^\infty x^T Qx + {u^*}^T Ru^* dt \\ &= \int_0^T x^T Qx + u^T Rudt + V(x(T)) \end{aligned}$$

where $V(x(T))$ is the cost-to-go term. If this term is very large it would indicate poor control.

Another issue common to both of the algorithms arises from the block based nature of the majority of subspace algorithms. The system matrices are only estimated every few hundred samples, depending on the user specified block data size. For better identification a larger block size (giving 'more rectangular' Hankel matrices) is preferred, but this means of course that the system matrices are estimated fewer times in a given time interval. This has implications in situations where parameter changes occur. It may be expected that after a parameter change in the identification object, the next set of system matrix estimates (calculated from input-output data from both before and after the change) will be poor, and it will take one further data collection period to provide accurate estimates for the altered system. Such behaviour was indeed observed in simulations. The result is that the control law, based on poor identification results, will at least be poor and could in fact destabilize the altered system, as was also observed in simulations.

A possible solution to this problem could be to use some form of data windowing, whereby the system matrices are estimated at intervals which are fractions of the data block size. This would reduce the time taken before a set of accurate estimates are available, but would of course increase the computational power required. This problem is common to any algorithm using a block based subspace method of identification and is therefore an important one to be investigated.

The issue of tuning parameters is common to both algorithms, with similar numbers of parameters to select in each. The POMOESP-MPC algorithm has more, due to its ability to deal with constraints on variables. Some of the parameters, in particular the weighting matrices for outputs and control signals, may be tuned intuitively (although the capacity for these to be time variant with respect to the prediction horizon complicates the issue considerably over a basic LQG tuning case). Others have less obvious effects on performance, most notably factors such as the data block size in the identification algorithms. Currently the only way to choose this value is by trial and error, and only very rough guides can be given (namely that the column size of the Hankel matrices should be much larger than the row size).

Finally, both algorithms share the advantageous properties that they may be applied to systems where very little is known about the identification/control object, and to MIMO systems. They may also be simply ‘plugged in’ to a system without the necessity of obtaining update laws or knowledge of system structure or order. The price paid for this apparent ease of application is presumably some difficulty in tuning.

10.2.2 Comparison with Existing Algorithms

The proposed subspace-adaptive algorithms are different in many ways to existing adaptive algorithms. Several advantages of the proposed algorithms clearly present themselves. Among these are:

- Ease of application to MIMO systems
- Very low level of required knowledge about the target system
- No ‘target-specific’ design required, only tuning of parameters

The first of these is perhaps the most important. Current adaptive control strategies such as those outlined in Chapter 5 are derived using input-output models for SISO systems, and may not be directly extended to MIMO systems. However many systems of interest as control objects for adaptive controllers are MIMO systems, such as chemical processes. Such systems are also likely to be complex, and thus the proposal of model structures which may be used for parametric identification techniques is difficult. With subspace methods, virtually nothing need be known about the system in question. Indeed, only an estimate of the maximum likely system order is required. The combination of these factors makes subspace-adaptive methods much more attractive than traditional methods for multivariable, complex systems about which little is known.

Related to the issue of prior information is the topic of the amount of design required for each application. As seen in Chapter 5, some design work must be done to obtain the adaptive control laws for a certain system. This work requires the knowledge of system order, and possibly (but not always) system structure. In contrast, the proposed subspace-adaptive algorithms can be applied directly to any system. The disadvantage with this is the number of parameters to be tuned. In traditional schemes, there are few parameters, often just a forgetting factor, an observer polynomial, or a parameter adjustment gain. As has been seen, there are a great deal of parameters to be tuned with the subspace-adaptive algorithms, concerning both the identification stage and the control, not all of which have intuitive effects on system performance.

A more difficult issue to make direct comparisons about is that of stability. There exist only limited results on the stability of existing adaptive control strategies, and the proposed algorithms are no exception. With the use of finite horizon control design methods, stability guarantees cannot be made. This is clearly an issue, in particular for safety-critical systems where the lack of guarantees on system stability is a major disadvantage.

The computational complexity of the proposed algorithms is of course much greater than existing algorithms. This could be an issue when high sampling rates are required, but in such applications as chemical processes it is not uncommon that sampling rates are lower, so computation times become less of an issue.

Advantages	**Disadvantages**
Easily applied to MIMO systems	Computationally complex
Very little prior information required	Lack of stability guarantees
No offline design required	Many tuning parameters

Figure 10.1: Summary of advantages and disadvantages of the proposed algorithms

10.3 Conclusions

A great many positive and negative points have come to light during this investigation. It is clear that subspace-adaptive methods have a large number of important and useful advantages over existing techniques, primarily the applicability to MIMO systems, and the low requirements on prior system information. It is also clear that a great many problems need to be solved, including the issues of parameter tuning, response times to parameter changes, and stability.

In general the results gained from the implemented control algorithms were encouraging. It must be stated that the systems used to test the algorithms were simple and perhaps did not fully demonstrate the capacities of the schemes. Nevertheless useful results and observations were obtained. Simulations and experiments on more complex, real-world systems are of course a natural progression.

In conclusion, it is proposed that these algorithms show much promise for further development. At this stage, given the advantages observed and the problems encountered, the most likely areas of application for the proposed algorithms are multivariable processes, about which little information is known. Processes to be controlled with long sampling periods are ideal targets, since the issue of computation time is reduced in importance. Possible problems with such applications include the issue of stability, as well as the ease (or otherwise) of tuning, in particular in MIMO systems with large numbers of variables.

10.4 Future Work

Having stated that the proposed algorithms show promise, it is of course necessary to highlight the areas in which further work is deemed necessary. This is perhaps best tackled by splitting the problem into identification issues and control issues.

10.4.1 Identification

During the course of this investigation, the power of the subspace algorithms was noted, in particular the ease with which they may be applied, as well as their applicability to multivariable systems. However, many of the parameters to be chosen by the user have less than intuitive effects on the identification results, and guidelines for selecting them are rather minimal.

In addition to these general issues, a further, more specific issue arises from the need to perform identification automatically. In particular, the method of analysis of the singular values to estimate the system order is very important in the identification process. In an offline setting this is performed visually by the user, and it is the user who must decide what constitutes a 'dominant' singular value. In this thesis a fairly simple heuristic algorithm was derived and found to work reasonably well in most situations. The situation is further complicated in the presence of nonlinearities, where it was found that the ease of analysis was affected by the choice of parameters such as block data size.

A suggestion in this area is that some kind of statistical significance testing could be employed, to confirm the choice of system order. However, such as system would require a great deal of additional computation, as system matrices would have to be estimated for each system order to be analysed, and the necessary tests performed.

Another issue perhaps more related to identification than control is that of response to parameter changes. As discussed earlier, the block based nature of subspace identification implies that a considerably time may pass between the occurrence of a parameter change in the identification object and the accurate estimation of the new system matrices. The solution suggested in this thesis, but not implemented in the presented schemes, is that system matrices may be calculated at instances which are fractions of the block data size, rather than only when an entirely new set of input-output data is available. In this way the time between parameter change and accurate estimation may be reduced. Naturally, much work would be required to find some optimal way of doing this.

In summary, it would be desirable to have results concerning the effects on identification accuracy of such parameters as Hankel matrix dimensions. It is also deemed important to continue to investigate new methods of analysing the singular values to estimate the system order, since this determines the accuracy of the identification results. Additionally, methods of minimizing response times to parameter changes should be investigated.

10.4.2 Control

The control design methodology focused upon in the implementations presented in this thesis is model predictive control. This strategy was chosen largely because it was deemed the most general, with the capability to deal with reference signals in a simple way, as well as with constraints on variables. This is, of course, not to say that there are no other design strategies to be followed. The LQG strategy has its advantages over finite-horizon MPC, in particular its ability to provide asymptotically stable control laws. Some wariness was shown in this thesis to the solution of a regulator problem via LQG design when subspace methods are used for identification, because of the condition that the intersection of the row spaces of future inputs and past states be empty. However this hypothesis was not formally proven, nor empirically investigated.

Focusing attentions on the MPC strategies used in the implemented algorithms, one of the main issues arising was the lack of stability guarantees given the finite horizon nature of the strategy. Lapses in the stability of the systems were observed. Little can be said about this, other than that further investigation of a theoretical nature would be useful.

The issue of the large number of parameters to be selected and tuned is also important. This applies not only to the control parameters but also to the identification parameters, although they are fewer. Methods exist for effective tuning of parameters for MPC, though they were not utilized in this work. It may be interesting to ascertain how easily the control algorithms may be tuned for more complex, multivariable systems. If it transpires that this task is not a prohibitively difficult undertaking, it may be concluded that the proposed algorithms represent a considerable advance over existing methods.

The recommendations of directions for future work may be summarized as:

- Effects on identification results of the user defined parameters such as Hankel matrix block sizes
- Methods of analysing the singular values to estimate the system order
- Methods of reducing latency from parameter changes to accurate identification results
- Use of different control design strategies
- Stability guarantees and finite horizon control design
- Control parameter selection and tuning

Part IV

Appendices

Appendix A

Mathematical Tools

A.1 Projections

A.1.1 Orthogonal Projections

Define an operator Π_B that projects the row space of a matrix onto the row space of a matrix $B \in \mathbb{R}^{q \times j}$:

$$\Pi_B \overset{\text{def}}{=} B^T(BB^T)^\dagger B \tag{A.1}$$

where $()^\dagger$ denotes the Moore-Penrose pseudo inverse, defined as follows.

Definition A.1.1. *Given the matrices $A \in \mathbb{R}^{m \times n}$, $X \in \mathbb{R}^{n \times m}$, X is said to be the Moore Penrose pseudo inverse of A if it satisfies all of the following conditions:*

$$\begin{aligned} AXA &= A \\ XAX &= X \\ (AX)^H &= AX \\ (XA)^H &= XA \end{aligned}$$

Define the projection of the row space of the matrix $A \in \mathbb{R}^{p \times j}$ on to the row space of $B \in \mathbb{R}^{q \times j}$ as:

$$A/\mathbf{B} \overset{\text{def}}{=} A.\Pi_B \tag{A.2}$$

Consider also the projection operator $\Pi_{B^\perp}$:

$$\begin{aligned} \Pi_{B^\perp} &= I_j - \Pi_B \\ &= I_j - B^T(BB^T)^\dagger B \end{aligned} \tag{A.3}$$

It can be seen that the the operators Π_B and $\Pi_{B^\perp}$ decompose a matrix into two matrices whose row spaces are orthogonal:

$$A = A\Pi_B + A\Pi_{B^\perp} \tag{A.4}$$

This is equivalent to expressing A as the linear combination of the row space of B and the orthogonal complement of B. Define:

$$\begin{aligned} L_B.B &\overset{\text{def}}{=} A/\mathbf{B} \\ L_{B^\perp}.B^\perp &\overset{\text{def}}{=} A/\mathbf{B}^\perp \end{aligned} \tag{A.5}$$

It is then clear from the definitions that:

$$A = L_B.B + L_{B^\perp}.B^\perp \tag{A.6}$$

A.1.2 Oblique Projections

In addition to decomposing a matrix into a linear combination of the basis of another matrix and its orthogonal complement, it is possible to decompose it as linear combinations of the bases of two nonorthogonal matrices. In this case, we may express the row space of A as a linear combination of the row spaces of the nonorthogonal matrices B and C, as well as a third matrix which is orthogonal to both B and C, using the same notation as introduced in Eq. A.6:

$$A = L_B.B + L_C.C + L_{B^\perp C^\perp}.\begin{pmatrix} B \\ C \end{pmatrix}^\perp \tag{A.7}$$

Here $L_C.C$ is defined as the oblique projection of A along the row space of B onto the row space of C, that is:

$$L_C.C \overset{\text{def}}{=} A/_B\mathbf{C} \tag{A.8}$$

Clearly the orthogonal projection is a special case of the oblique projection, one in which the matrix C is made equal to $B^\perp$. The oblique projection may be interpreted as a projection of A onto the combined row spaces of the matrices B and C, with the result then decomposed onto the row spaces of one of the latter matrices. This may be written in matrix form as:

$$A/\begin{pmatrix} \mathbf{C} \\ \mathbf{B} \end{pmatrix} = A.\begin{pmatrix} C^T & B^T \end{pmatrix}.\begin{pmatrix} CC^T & CB^T \\ BC^T & BB^T \end{pmatrix}.\begin{pmatrix} C \\ B \end{pmatrix} \tag{A.9}$$

which is then to be decomposed along the row space of B or C. The oblique projection may then be defined as follows:

Definition A.1.2. *The oblique projection of the row space of A along the row space of B onto the row space of C may be defined as:*

$$A/_B\boldsymbol{C} = A.\begin{pmatrix} C^T & B^T \end{pmatrix}.\begin{pmatrix} CC^T & CB^T \\ BC^T & BB^T \end{pmatrix}^\dagger.C \tag{A.10}$$

A.2 The QR Decomposition

The QR decomposition of a matrix $A \in \mathbb{R}^{m\times n}$ is given by:

$$A = QR$$

where $Q \in \mathbb{R}^{m\times m}$ has orthonormal columns and $R \in \mathbb{R}^{m\times n}$ is upper triangular. In the case $m > n$, the final $m - n$ rows of R are zero, and thus these rows and the final $m - n$ columns of Q may be removed, resulting in a decomposition with $Q \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{n\times n}$.

One of the applications of the QR decomposition is solving orthogonal projections. To this end it is used throughout this thesis. Consider a projection operator Π_A, defined in the same manner as the operator in Equation A.1 but which projects a matrix $B \in \mathbb{R}^{m\times p}$ onto the column space of $A \in \mathbb{R}^{m\times n}$, namely:

$$\Pi_A \overset{\text{def}}{=} A(A^TA)^\dagger A^T \tag{A.11}$$

Let this multiply B from the left, giving $\Pi_A B$. This can be solved using the QR decomposition:

$$\begin{pmatrix} A & B \end{pmatrix} = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

where the right hand side is partitioned such that $Q_1 \in \mathbb{R}^{m\times n}$, $Q_2 \in \mathbb{R}^{m\times m-n}$, $R_{11} \in \mathbb{R}^{n\times n}$, $R_{12} \in \mathbb{R}^{n\times p}$ and $R_{22} \in \mathbb{R}^{m-n\times p}$. This gives:

$$\begin{aligned} A &= Q_1R_{11} \\ B &= Q_1R_{12} + Q_2R_{22} \end{aligned} \tag{A.12}$$

Using Equation A.11 and Equation A.12 and various properties of the matrices Q and R, the projection may be written as:

$$\begin{aligned} \Pi_A B &= A(A^TA)^\dagger A^T B \\ &= Q_1R_{11}(R_{11}^TQ_1^TQ_1R_{11})^\dagger R_{11}^TQ_1^T(Q_1R_{12} + Q_2R_{22}) \\ &= Q_1R_{11}(R_{11}^TR_{11})^\dagger R_{11}^TR_{12} \\ &= Q_1R_{11}R_{11}^{-1}R_{11}^{-T}R_{11}^TR_{12} \\ &= Q_1R_{12} \end{aligned}$$

In the algorithms presented in this thesis, the orthogonal projections involve projection onto row spaces, that is, projection operators of the form in Equation A.1 multiply matrices from the left. In this case an RQ decomposition can be used, which has the form:

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

A.3 The Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$ is given by:

$$A = U\Sigma V^T$$

where $U \in \mathbb{R}^{m\times m}$ and $U \in \mathbb{R}^{n\times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix, whose diagonal elements are called the *singular values* of the matrix A. The number of nonzero singular values s is the rank of A, and the first s columns of U provide a basis in the column space of A. This can be illustrated as:

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} = U_1\Sigma_1V_1^T \tag{A.13}$$

The SVD may be used to approximate a matrix with one of lower order. By observing the singular values and choosing only those deemed ‘large’, an approximation of the original matrix may be obtained as in Equation A.13. This is used in this thesis to obtain system matrices, in the presence of noise which acts to make the remaining singular values nonzero. The number of ‘large’ singular values can be used to determine the order of the system and thus to recover column spaces of certain matrices.

Appendix B

Matlab Tools

B.1 SPC Matlab files

The Matlab code used to implement the subspace predictive control (SPC) strategy is listed here. Three functions were written, two of which are Simulink M-files for s-functions in the Simulink model, and the third is a standard Matlab function.

B.1.1 SPC1.m

The function `spc1.m` calculates the prediction matrices L_w and L_u which may be used to calculate controller matrices. They are calculated using an RQ decomposition of the Hankel matrices of 'past' and 'future' inputs and outputs. The Matlab source code for this function is listed below.

```
function [L_w,L_u] = spc1(u,y,s);

% function spc1 provides the prediction matrices Lw and Lu
% which may be used to obtain the prediction:
%
%    Yf^hat = [Lw Lu][Wp ; Uf]
%
% This prediction can be used to find an optimal future control
% sequence Uf that minimizes the cost:
%
%    V = (Yf-Rf)'Q(Yf-Rf) + Uf'RUf
%
% Input:
%   u, y    The input-output data of the system to be identified.
%   s       The dimension parameter that determines the number
%           of block rows in the processed Hankel matrices. Should be
%           chosen  to be twice as large as the maximum expected
%           system order
%
% Output:
%   Lw, Lu  Prediction matrices multiplying the past inputs
%           and outputs Wp, and the future inputs Up, respectively
%
%   See also    SPCID   fixedspc
%
% Brad Schofield, LTH Reglerteknik 2003


if nargin<3
  error('Not enough input variables')
end

% Reorient block data if necessary
if size(y,2)>size(y,1)
  y=y';
end
if size(u,2)>size(u,1)
  u=u';
end
N=size(y,1); % data block size
l=size(y,2); % output dimension
m=size(u,2); % input dimension

if l==0,
  error('output required')
end

if (~(size(u,1)==N) &~isempty(u))
  error('Input and output should have same length')
end

if 2*(m+l)*s>=N-2*s+1
  error('s is chosen too large or data size is too small')
end

% construction of Hankel matrices
%
NN=N-2*s+1; % NN is the index j used in the literature
Up=zeros(NN,m*s); % these Hankel Matrices are transposed
                  % versions of the usual definitions
Uf=zeros(NN,m*s); % (used for the RQ decomposition - take
                  % a QR decomposition of the transpose)
Yf=zeros(NN,l*s);
Yp=zeros(NN,l*s);
for i=(1:s)
  if m>0
    Up(:,(i-1)*m+1:i*m)=u(i:NN+i-1,:);
    Uf(:,(i-1)*m+1:i*m)=u(s+i:NN+s+i-1,:);
  end
    Yp(:,(i-1)*l+1:i*l)=y(i:NN+i-1,:);
    Yf(:,(i-1)*l+1:i*l)=y(s+i:NN+s+i-1,:);
```

```
end

% Solve the least squares problem via an orthogonal projection
% of the future outputs (Yf) onto the combined row space of the
% past io (Wp) and the future outputs (Uf). This is implemented
% with an RQ decomposition.

R=triu(qr([Yp Up Uf Yf]));  % perform an RQ decomposition on
                            % [Yp Up Uf Yf]'

Rlower=R(1:2*(m+l)*s,1:2*(m+l)*s)'; % Trims R matrix to desired
                                    % size and transposes
                                    % to obtain a lower triangular
                                    % matrix

% Obtain [R31 R32] (=:RL1)
RL1=Rlower((2*m+l)*s+1:2*(m+l)*s,1:(2*m+l)*s);

% Obtain [R11 0; R21 R22] (=:RL2)
RL2 = Rlower(1:(2*m+l)*s,1:(2*m+l)*s);

% Calculate L = [R31 R32][R11 0; R21 R22]^+
L = RL1*pinv(RL2);

% Partition L into L_w and L_u components
L_w = L(:,1:(m+l)*s);
L_u = L(:,(m+l)*s+1:end);

% Test to see whether the prediction works.
Yest1 = L*[Yp(1,:)';Up(1,:)';Uf(1,:)']
vafsingle = vaf(Yest1,y(s+1:2*s))

% END OF THE CALCULATIONS
```

B.1.2 SPCID

The S-function `SPCID.m` is used to calculate the prediction and control matrices. These are then passed to the S-function `fixedspc.m` which implements the control from sample to sample. The Matlab code for this function is listed below.

```
function [sys,x0,str,ts] = SPCID(t,x,input,flag,N,s,h,m,l,Q,R)

% fixedspc An M-file S-function for implementing a Subspace
% Predictive Controller. The function spc1 is used to obtain
% the prediction matrices Lw and Lu, from which the controller
% matrices K_rc and K_wc are calculated using the user- defined
% (positive definite) weighting matrices Q and R.  The cost
% function minimized is:
%
```

```
%   V = (Yf-Rf)'Q(Yf-Rf) + Uf'RUf
%
% Input:
%   U,Y         Block I/O data
%
% Output:
%   K_rc, K_wc  Controller Matrices
%
% Parameters:
%   s           Hankel matrix index, serves as both the forward
%               and backward horizons. (should be twice
%               the max expected system order)
%   h           sampling interval;
%   N           Block data size
%   m           input dimension
%   l           output dimension
%   Q           Weighting matrix for future error sequence Yf-Rf
%   R           Weighting matrix for future control sequence Uf
%
%   see also    spc1    fixedspc

%%%%%%%%%%%%%%%%%%%%%%%%
% Initial Calculations %
%%%%%%%%%%%%%%%%%%%%%%%%
h_id = h*N; % increase sampling period


switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(N,s,h_id,m,l);


  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys = mdlOutputs(x,input,N,s,m,l,Q,R);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case {1,2,4,9},
    sys = []; % do nothing

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

%end SPCID

%============================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times
% for the S-function.
%============================================================%
function [sys,x0,str,ts,counter] = mdlInitializeSizes(N,s,h_id,m,l)

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0; % no states
sizes.NumOutputs     = m*s*(m+l) + m*s*l;%size of K_wc and K_rc
sizes.NumInputs      = N*(m+l);
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

x0=[];
str = [];
ts  = [h_id 0];

% end mdlInitializeSizes

%============================================================
% mdlOutputs
% output the system
%============================================================
function sys = mdlOutputs(x,input,N,s,m,l,Q,R)

% Unpack the past i/o blocks
Up = inputUnpack(N,input,m,0);
offset = N*m;
Yp = inputUnpack(N,input,l,offset);

%--------------------------------------------------------
% Find the prediction matrices Lw and Lu
[Lw,Lu] = spc1(Up,Yp,s);

% Calculate the control matrices K_rc and K_wc
tmx1 = inv(R + Lu'*Q*Lu);
K_rc = tmx1(1:m,:)*Lu'*Q; % size
K_wc = K_rc*Lw; % size m*s*(m+l)
```

```
% Pack these and output
x = mdlPack(K_rc,K_wc,s,m,l);
%--------------------------------------------------------

sys = x;

%end mdlOutputs

%===========================================================
% inputUnpack
% Unpack the input vector into a matrix of approprate dimensions
%===========================================================
function U = inputUnpack(n,u,size,offset)

    result = [];
    for i = 0:n-1
        urow = u(offset + size*i + 1 : offset + size*(i+1))';
        result = [result;urow];
    end
    U=result;
 %end inputUnpack

%===========================================================
% mdlPack
% Pack K_wc and K_rc in a row vector
%===========================================================
function x = mdlPack(K_rc, K_wc,s,m,l)
K_rcVec = [];
for i = 1:s*l
    K_rcVec = [K_rcVec;K_rc(:,i)];
end

K_wcVec = [];
for i = 1:s*(m+l)
    K_wcVec = [K_wcVec;K_wc(:,i)];
end
x = [K_rcVec;K_wcVec];
%end mdlPack
```

B.1.3 fixedspc.m

The S-function `fixedspc.m` is used to calculate the optimal future control sequence and implement the first element of this. It uses the control matrices designed by the function `SPCID.m`. The Matlab code for this function is listed below.

```
function [sys,x0,str,ts] = fixedspc(t,x,input,flag,s,h,m,l)

% fixedspc An M-file S-function for implementing a Subspace
% Predictive Controller where the controller parameters K_rc
% and K_wc have already been calculated by SPCID.m
%
% Inputs:
%   up,yp, past input, output vectors respectively
%   K_rc,  Controller matrix multiplying inputs
%   K_wc,  Controller matrix multiplying past i-o
%
% Outputs:
%   u      control signal u
%
% Parameters:
%   s,     Hankel matrix index, serves as both the forward and
%          backward horizons (equals twice the max expected
%          system order)
%   h,     sampling interval
%   m,     input dimension
%   l,     output dimension
%
% See also  spc1    SPCID

switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(s,h,m,l);



  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys = mdlOutputs(x,input,s,m,l);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case {1,2,4,9},
    sys = []; % do nothing

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

%end fixedspc
```

```
%=============================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times
% for the S-function.
%=============================================================
function [sys,x0,str,ts,counter] = mdlInitializeSizes(s,h,m,l)

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = m;
sizes.NumInputs      = m*s*(m+l) + m*s*l + s*(m+l)+l;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

x0=[];
str = [];
ts  = [h 0];

% end mdlInitializeSizes

%=============================================================
% mdlOutputs
% output the system
%=============================================================
function sys = mdlOutputs(x,input,s,m,l)

% Unpack the past i/o blocks
Up = inputUnpack(s,input,m,(m*s*(m+l) + m*s*l));
offset = s*m+m*s*(m+l) + m*s*l;
Yp = inputUnpack(s,input,l,offset);

% Define Wp
Wp = [Yp; Up];

% Unpack the reference signal
offset = s*(m+l)+ m*s*(m+l) + m*s*l;
r = input(offset+1:end);

% Unpack the controller matrices K_rc and K_wc
[K_rc K_wc] = mdlUnpack(input,s,m,l);

if (K_rc==zeros(m,l*s))
    u1 = r;
else
    % Precalculate part of control law
    pre_u = -K_wc*Wp; % could store this in a state for
```

```
                        % updatestate/calculate output
                        % separation
    % Construct the future reference sequence {r_k}
    rf = r*ones(s,1); % Assume the future references will be
                        % the same as the current
    % Calculate first control output u1 of optimal sequence u_k
    u1 = pre_u + K_rc*rf;
end

sys = u1;
%end mdlOutputs

%===========================================================
% inputUnpack
% Unpack the input vector into a matrix of approprate
% dimensions
%===========================================================
function U = inputUnpack(n,u,size,offset)
    result = [];
    for i = 0:n-1
        urow = u(offset + size*i + 1 : offset + size*(i+1))';
        result = [result;urow];
    end
    U=result;
 %end inputUnpack

%===========================================================
% mdlUnpack
% Unpack K_rc and K_wc from input
%===========================================================
function [K_rc,K_wc] = mdlUnpack(input,s,m,l)
K_rc = [];
for i = 0:s*l-1
    K_rc = [K_rc,input(i*m+1:(i+1)*m)];
end
offset = s*l*m; % number of elements in K_rc
K_wc = [];
for i = 0:s*(l+m)-1
    K_wc = [K_wc,input(offset+i*m+1:offset+(i+1)*m)];
end
%end mdlUnpack
```

B.2 Online POMOESP Implementation

B.2.1 POMOESP.m

The S-function `POMOESP.m` operates as described in Chapter 7. The Matlab code for this function is listed below.

```
function [sys,x0,str,ts] = POMOESP(t,x,u,flag,h,n,s,l,m)
```

```
% POMOESP An M-file S-function for identifying the system
% matrices [A B C D] from input-output data u(k),y(k) using
% the Past Output MOESP subspace algorithm.
%
%-----------------------------------------------------
% Generation of Singular Values and R matrix
%-----------------------------------------------------
% Uses block input-output data to calculate Singular Values
% S, triangular information matrix R using dordpo function
% by Verhaegen.
%
%-----------------------------------------------------
% Determination of System order from Singular Values
%-----------------------------------------------------
% Heuristically determins the order N of a system from
% Singular Values. Compares the 'second differential' of the
% logarithms of singular values and finds the largest value.
% The number of singular values below this maximum is taken
% to be the order of the system.
%
%-----------------------------------------------------
% Calculation of System Matrices
%-----------------------------------------------------
% Generation of the System Matrices [A B; C D] from R
% matrix, system order estimate N, and sequential inputs
% (formed into block data of arbitrary size) using dmodpo
% and dac2bd functions by Verhaegen.
%
% Requires SMI 1.0 toolbox (Verhaegen)
%
% Inputs: U(k): input vector; Y(k): output vector, both size n
% Outputs: System order N, Identified System polynomials A, B.
% Parameters: n: size of data block; s: Maximum expected
% system order, sampling interval
%
% Brad Schofield, LTH 2003

%=========================
% Input Adjustments
%=========================
N_0 = ceil(s/2);  % Arbitrary assignment of initial system
                  % order. Modify to include user specification
s = 2*s;  % Double the max expected system order, as recommended
          % for use in POMOESP algortihm
h_alg = h*n;  % Reduces sampling frequency to read in a whole
              % new data block

i_u = 1; % take the first input column (corresponding to theta for
         % pendulum) when calculating the transfer function from
         % the estimated state space model

%==========================
% Calculate Dimensions of R
%==========================

rowsizeR = s*(2*m+2*l);
colsizeR = max(4,s*(2*m+3*l));

switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(n,N_0,h_alg,s,rowsizeR,colsizeR,l,m);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys = mdlUpdate(t,x,u,s,n,rowsizeR,colsizeR,l,m);

  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys = mdlOutputs(x,s,u,n,rowsizeR,colsizeR,l,m,N_0);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case {1,4,9},
    sys = []; % do nothing

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

%end POMOESP

%=========================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for
% the S-function.
%=========================================================
function [sys,x0,str,ts] = mdlInitializeSizes(n,N_0,h,s,rowsizeR,colsizeR,l,m)
```

```
sizes = simsizes;
sizes.NumContStates  = 0;       %
sizes.NumDiscStates  = 2;       % useR,useRbd flags
sizes.NumOutputs     = s/2*s/2 + s/2*m + s/2*l + l*m+2;%
sizes.NumInputs      = l*n+m*n; % U(k) and Y(k)
sizes.DirFeedthrough = 1;       %
sizes.NumSampleTimes = 1;       %

sys = simsizes(sizes);


useR_0 = 0;
useRbd_0 = 0;
x0 = [useR_0;useRbd_0];
str = [];
ts  = [h 0];

% end mdlInitializeSizes

%===========================================================
% mdlUpdate
%===========================================================
function sys = mdlUpdate(t,x,u,s,n,rowsizeR,colsizeR,l,m)

useR = x(1,1);
useRbd = x(2,1);
if (useR == 0)
    useR=1;
else if (useRbd == 0)
        useRbd = 1;
    end
end

x = mdlPack(useR,useRbd);
sys = x

%end mdlUpdate

%===========================================================
% mdlOutputs
% Calucate System Order and System Matrices.
%===========================================================
function sys = mdlOutputs(x,s,u,n,rowsizeR,colsizeR,l,m,N_0)

[useR,useRbd] = mdlUnpack(x);

%===============================================
% unpack U and Y from input u
%===============================================
U = inputUnpack(n,u,m,0);
offset = n*m;
Y = inputUnpack(n,u,l,offset);

%-----------------------------------------------------------
% Check if this is the first time the algorithm is to be run,
% and if so, do not use R in the argument of dordpo
%-----------------------------------------------------------
useR1 = useR
if (useR==0)

    % call dordpo w/o R
    [S,R]=dordpo(U,Y,s);
     save R0 R
else
    % call dordpo with R
    load Rold;
    [S,R]=dordpo(U,Y,s,R);
end
    save Rold R
%----------------------------------------------------
% Determination of System Order from Singular Values
%----------------------------------------------------
semilogy(S,'rx');
title('Singular Values')
logS = log10(S);
diff = zeros([s-1,1]);
for j = 1:s-1
    diff(j,1) = logS(j)-logS(j+1);
end

diff2 = zeros(s-2,1);
for j = 1:s-2
    diff2(j,1) = diff(j)-diff(j+1);
end
gap = max(diff2);
for i = 1:s-2
    if (gap==diff2(i,1))
        N = i;
        break
    end
end
%end Singular Value Analysis

%----------------------------------------------------
% Calculate System Matrices and Polynomials
%----------------------------------------------------
if (x(1,1)==0)
    sys = zeros(s/2*s/2 + s/2*m + s/2*l + l*m+2,1);
else % executed on the second and successive runs
```

```
    [A,C]=dmodpo(R,N);      % calculate system matrices
    [B,D,RnewBD]=dac2bd(A,C,U,Y);
    IC = dinit(A,B,C,D,U,Y); % Find the initial conditions
                             % if desired
    save Results A B C D IC U Y;
    design_ctrl = 1;
    output = outputPack(A,B,C,D,N,design_ctrl,s,l,m);
    sys = output;
end

%end Calculate System Matrices

%end mdlOutputs

%============================================================
% mdlPack
% Pack useR, useRbd flags in x
%============================================================
function x = mdlPack(useR,useRbd)

x = [useR;useRbd];
%end mdlPack

%============================================================
% mdlUnpack
% Unpack useR, useRbd flags from x
%============================================================
function [useR,useRbd] = mdlUnpack(x)

useR = x(1,1);
useRbd = x(2,1);

%end mdlUnpack

%============================================================
% inputUnpack
% Unpack the input vector into a matrix of appropriate
% dimensions
%============================================================
function U = inputUnpack(n,u,size,offset)

    result = [];
    for i = 0:n-1
        urow = u(offset + size*i + 1 : offset + size*(i+1))';
        result = [result;urow];
    end
    U=result;
 %end inputUnpack

%============================================================
% outputPack
% Pack A, B, C, D, n, design_ctrl into a vector
%=============================================================
function x = outputPack(A,B,C,D,N,design_ctrl,s,l,m)

snew = s/2; % obtain original estimate of max system order

AVec = [];
for i = 1:N
    AVec = [AVec; A(:,i)];
end

BVec = [];
for i = 1:m
    BVec = [BVec; B(:,i)];
end

CVec = [];
for i = 1:N
    CVec = [CVec; C(:,i)];
end

DVec = [];
for i = 1:m
    DVec = [DVec; D(:,i)];
end

% output size is s/2*s/2 + s/2*m + s/2*l + l*m + 1(n) + 1(design_ctrl)
% need to 'pad' the rest of the output with zeros
if N<snew
    Pad_size = snew*snew + snew*(m+l) + l*m - (N*N + N*(m+l) + l*m);
    padVec = zeros(Pad_size,1);
else
    padVec = [];
end
x = [AVec; BVec; CVec; DVec; padVec; N; design_ctrl];
%end mdlPack
```

Bibliography

[1] Brian Anderson and John Moore, *Optimal control linear quadratic methods*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[2] H. H. J. Bloemen and T. J. J. van den Boom, *Constrained linear model-based predictive control with an infinite control and prediction horizon*, Proceedings of the 14th IFAC (1999).

[3] Wouter Favoreel, Bart De Moor, and Michel Gevers, *Spc: Subspace predictive control*, Proceedings of the 14th IFAC (1999).

[4] Magnus Gäfvert, *Modelling the furuta pendulum*, Tech. report, Department of Automatic Control, Lund Institute of Technology, April 1998.

[5] B. R. J. Haverkamp, C. T. Chou, M. Verhaegen, and R. Johansson, *Identification of continuous time mimo state space models from sampled data, in the presence of process and measurement noise*, Proceedings of the 35th Conference on Decision and Control (1996).

[6] Bert Haverkamp, *Subspace method identification, theory and practice*, Ph.D. thesis, Delft, The Netherlands, August 2000.

[7] Rolf Johansson, *System modelling and identification*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[8] Rolf Johansson, Michel Verhaegen, and C. T. Chou, *Stochastic theory of continuous-time state-space identification.*

[9] Johan Åkesson, *Safe manual control of unstable systems*, Master's thesis, Department of Automatic Control, Lund Institute of Technology, September 2000.

[10] Robin De Keyser, *A gentle introduction to model based predictive control*, International Conference on Control Engineering and Signal Processing, Piura Peru (1998).

[11] J. M. Lemos and E. Mosca, *A multipredictor-based lq self-tuning controller*, IFAC Identification and System Parameter Estimation (1985).

[12] Marco Lovera, *Subspace identification methods: Theory and applications*, Ph.D. thesis, Politecnico Di Milano, 1997.

[13] J. M. Maciejowski, *Predictive control with constraints*, November 1999.

[14] Marc Moonen, Bart De Moor, and Joos Vandewalle Vandenberghe, *On- and off-line identification of linear state space models*, International Journal of Control **49** (1989), no. 1, 219–232.

[15] Marc Moonen and Joos Vandewalle, *A qsvd approach to on- and off-line state-space identification.*

[16] C. Rowe and J. M. Maciejowski, *Tuning robust model predictive controllers using lqg/ltr*, Proceedings of the 14 IFAC (1999).

[17] Walter Rudin, *Real and complex analysis*, third ed., McGraw Hill, 1987.

[18] Wilson J. Rugh, *Linear system theory*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[19] Gilbert Strang, *Linear algebra and its applications*, Academic Press, 1976.

[20] Karl J. Åström and Björn Wittenmark, *Adaptive control*, second ed., Addison Wesley, 1995.

[21] ———, *Computer controlled systems theory and design*, third ed., Prentice Hall, Englewood Cliffs, NJ, 1997.

[22] Peter van Overschee and Bart De Moor, *Subspace identification for linear systems*, Kluwer Academic Publishers, 1996.

[23] V. Verdult, M. Verhaegen, C. T. Chou, and M. Lovera, *Efficient subspace-based identification of mimo bilinear state space models*, Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear-Modelling: Theory and Applications (1998), 216–221.

[24] Vincent Verdult and Michel Verhaegen, *Subspace-based identification of mimo bilinear systems*, Proceedings of the European Control Conference (1999).

[25] ———, *Identification of multivariable linear parameter-varying systems based on subspace techniques*, Conference on Decision and Control CDC 2000 (2000).

[26] Michel Verhaegen and Patrick Dewilde, *Subspace model identification part 1. the output-eror state-sace model identification class of algorithms*, International Journal of Control **56** (1992), no. 5, 1187–1210.

[27] ———, *Subspace model identification part 2. analysis of the elementary output-error state-space model identification algorithm*, International Journal of Control **56** (1992), no. 5, 1211–1241.

[28] Michel Verhaegen and David Westwick, *Identifying mimo wiener systems using subspace model identification methods*, Proceedings of the 34th Conference on Decision ans Control (1995).

[29] Eric A. Wan and Alexander A. Bogdanov, *Model predictive neural control with applications to a 6dof helicopter model.*