CONTINUOUS-TIME SELF-TUNING ALGORITHMS

by

HÜSEYİN DEMİRCİOĞLU

Thesis submitted for the degree of

DOCTOR OF PHILOSOPHY

Faculty of Engineering, University of Glasgow

October 1989

ACKNOWLEDGEMENTS

I wish to express my gratitude to my supervisor Prof. P.J. Gawthrop for his guidance and encouragement during the course of this study. I also wish to thank Turkish Ministry of Education for providing financial support, which made this thesis possible.

I am indebted to my parents for their continued support and encouragement throughout my education.

Table of Contents

ABSTRACT

This thesis proposes some new self-tuning algorithms. In contrast to the conventional discrete-time approach to self-tuning control, the continuous-time approach is used here, that is continuous-time design but digital implementation is used. The proposed underlying control methods are combined with a continuous-time version of the well-known discrete recursive least squares algorithms. The continuous-time estimation scheme is chosen to maintain the continuous-time nature of the algorithms.

The first new algorithm proposed is emulator-based relay control (which has already been described in a paper by the author). The algorithm is based on the idea of constructing the switching surface by emulators; that is, unrealisable output derivatives are replaced by their emulated values. In particular, the relay is forced to operate in the sliding mode. In this case, it is shown that emulator-based control and its proposed relay version become equivalent in the sense that both give the same control law.

The second new algorithm proposed is a continuous-time version of the discrete-time generalized predictive control (GPC) of Clarke *et al* (which has already been described in a paper by the author). The algorithm, continuous-time generalized predictive control (CGPC), is based on similar ideas to the GPC, however the formulation is very different. For example, the output prediction is accomplished by using the Taylor series expansion of the output and emulating the output derivatives involved.

A detailed closed-loop analysis of this algorithm is also given. It is shown that the CGPC control law only changes the closed-loop pole locations leaving the open-loop zeros untouched (except one special case). It is also shown that LQ control can be considered in the CGPC framework. Further, the CGPC is extended to include some design polynomials so that the model-following and pole-placement control can be considered in the same framework.

A third new algorithm, a relay version of the CGPC, is described. The method is based on the ideas of the emulator-based relay control and again it is shown that the CGPC and its relay version become equivalent when the relay operates in the sliding mode.

Finally, the CGPC ideas are extended to the multivariable systems and the resulting closed-loop system is analysed in some detail. It is shown that some special choice of design parameters result in a decoupled closed-loop system for certain systems. In addition, it is shown that if the system is decouplable, it is possible to obtain model-following control. It is also shown that LQ control, as in the scalar case, can be considered in the same framework.

An illustrative simulation study is also provided for all of the above methods throughout the thesis.

CHAPTER 1

INTRODUCTION

1.1. SELF-TUNING CONTROL

In general, any control system design involves two steps: system modeling and controller design. Self-tuning control may be viewed as an automation of these two steps. It consists of a recursive estimator and a controller design procedure. A block diagram of such a system is shown in figure 1.1. The recursive estimator obtains a plant model from the input/output data. The estimated model is then used in the controller design procedure to deduce the controller parameters. As the process model and the controller are updated at each sampling time, a self-tuning controller is expected to detect changes in the process and tune itself accordingly.

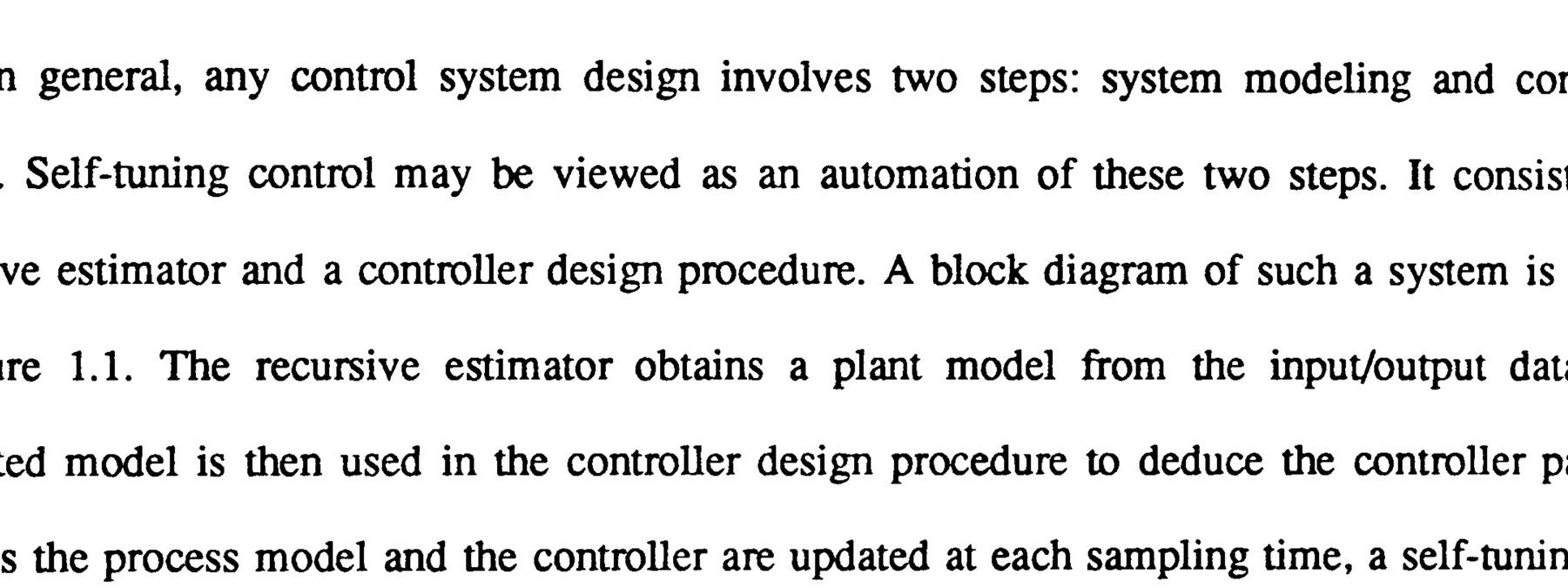


Figure 1.1 Block diagram of a self-tuning controller

A self-tuner can be thought of as composed of two loops: the inner loop which is an ordinary feedback loop having process and controller; and the outer loop (estimator and controller

design) which adjust parameters of the controller. This structure is common to any adaptive control system.

The self-tuner shown in figure 1.1 is called an indirect (or explicit) self-tuning controller as the controller parameters are updated indirectly through a design step. If the controller design step is avoided so that the controller parameters are estimated directly, then the self-tuner obtained is called direct (or implicit). This simplifies the algorithm significantly. However, it is limited to certain types of controller design methods.

1.1.1. Recursive Parameter Estimation

Recursive estimation of the process parameters is a key element in the self-tuning control. There are many recursive estimation methods such as least squares, extended least squares, instrumental variables and maximum likelihood (Astrom, 1971). Among these methods, the most common and widely used one is the recursive least squares (RLS). This is mainly due to its simple structure. Discrete estimation methods are now well established (Ljung, 1983, Ljung, 1987). In this thesis, a continuous time version of the discrete least squares will be used. Details of the algorithm is given in chapter 2.

In order to be able to track the changes in the process parameters, the estimator should discard the old data, as it may not be relevant any more to the current parameters of the process. This can be done by weighting the data accordingly, that is by putting less and less weight on the data as it gets older. The most commonly used weighting scheme is that of exponential forgetting in which the exponential weighting coefficient is refer to as *forgetting factor*. The effective memory length and thus the ability of tracking the parameter variations depend on the choice of the forgetting factor: the shorter the memory length, the faster the adaptation. On the other hand, memory length can not be further reduced from a certain length as this is essential for satisfactory estimation. This clearly puts a limit on the rate of change which can be tracked.

A typical problem with the forgetting is that of *covariance blow-up* that is exponential increase of the covariance matrix and thus the subsequent failure. This problem occurs due to the discounting of the old data while the new information comming in from the plant is not sufficient,

that is in a situation where the input signal is not persistently exciting. There are several methods to overcome this problem such as putting an upper bound on the diagonal elements or on the trace of the covariance matrix, stopping the parameter and covariance update when there is not enough information and adjusting the forgetting factor automatically. See for example (Fortescue, 1981) and (Astrom, 1984).

1.1.2. A Brief Review

Self-tuning control has been a very active domain of research during the last two decades. The idea of self-tuning seems to have been first proposed by Kalman in 1958 (Kalman, 1958). However, due to lack of technology for the implementation, the idea was abandoned until 1970s. The developments in both computer technology and control theory during 1960s led to new interest in the subject in early 1970s (Peterka, 1970, Wieslander, 1971, Astrom, 1973). These works were based on the minimum variance control strategy (Astrom, 1970) and the recursive least squares. A variant of the minimum variance (MV) self-tuner was later proposed by Clarke and Gawthrop (Clarke, 1975). This method, which is now known as generalised minimum variance (GMV), have some advantages over the MV self-tuner. In particular, use of the control weighting enables the stable control of nonminimum-phase systems. Some interpretations of this method such as model-reference adaptive control and optimal smith predictor were further given in (Gawthrop, 1977). During this period, some industrial applications were also reported (Borisson, 1974, Borisson, 1976). A review of some of these early developments and applications can be found in (Astrom, 1977). The above developments in early and mid 1970s stimulated extensive research into different types of self-tuning controllers.

MV or GMV type controllers are based on the exact knowledge of the time delay. Therefore, these type of self-tuners perform badly if the assumed time delay does not corresponds to the actual delay of the plant or if the time delay varies. In addition, although nonminimum-phase systems can be controlled by the GMV method, the choice of the control weighting involved is not straightforward. To overcome these problems, several authors proposed pole(/zero)-placement algorithms (Wellstead, 1979, Wellstead, 1979, Allidina, 1980, Astrom, 1980, Clarke, 1982). In the

same spirit, Grimble described a polynomial LQC method (Grimble, 1984), which can be treated in the pole-placement framework where the desired closed-loop poles are obtained from a spectral factorisation, rather than chosen by the designer. Although pole-placement methods overcome the above two problems, they fail to control the system if common factors in the estimated model arise (e.g. due to overparametrisation), which is of great importance from implementation and robustness point of view. Some research effort has also directed towards the state-space methods, specially LQ and LQC, in order to avoid the problems associated with GMV and pole-placement methods (Lam, 1980, Warwick, 1981, Samson, 1982, Clarke, 1985). Although these methods can cope with overparametrisation, nonminimum-phase zeros and variable time delay, they are computationally more demanding compared to GMV and pole-placement self-tuners.

Sensitivity of the GMV method to time delay variations is simply due to the fact that the method is based only on *k-step ahead* prediction, where k is the assumed time delay. Therefore, one may expect to achieve robustness against time delay variations by making predictions over a range, which covers possible time delay variations, and minimising a multi-stage cost function of these predictions. In recent years, many self-tuning controllers based on this and similar ideas have also been proposed in the literature (Peterka, 1984, Ydstie, 1984, Mosca, 1984, Keyser, 1985, Clarke, 1985, Clarke, 1987, Keyser, 1988). These algorithms are in general classified as Long Range Predictive Control (LRPC) methods and all have the common feature of being robust against time delay variations. An important point worth mentioning is that, these methods are based on a *receding-horizon* control strategy. This ensures a time invariant control law and also enhances the robustness, as it takes into account the latest information available at each time instant, unlike a *fixed-horizon* control strategy. Among these algorithms, the Generalised Predictive Control (GPC) method of Clarke *et al* appears to be the best. Its robustness, ability to cope with difficult systems, and superiority to some other self-tuners such as GMV and pole-placement have been illustrated by simulations (Clarke, 1985, Clarke, 1987, Mohtadi, 1987). Recently, some successful industrial applications of the GPC have also been reported demonstrating its effectiveness for the self-tuning control of industrial processes (Clarke, 1988).

Many self-tuning algorithms have also been extended to multivariable case. The MV self-tuner was first extended to multivariable case by Borisson (Borisson, 1979). The extension of GMV method then followed this (Koivo, 1980, Keviczky, 1981). A multivariable pole-placement self-tuner was outlined by Prager and Wellstead (Prager, 1980) and so on. See chapter 5 for the review of the developments in multivariable self-tuning controllers.

1.2. SCOPE AND OUTLINE OF THE THESIS

Most of the self-tuning literature has been devoted to the discrete-time methods. This is presumably due to the digital technology necessary for the implementation. However, there are some problems with discrete-time methods such as nonminimum-phase zeros due to fast sampling and/or fractional delay, numerical sensitivity, etc (Gawthrop, 1982, Astrom, 1984, Clarke, 1984, Sinha, 1985). These problems are addressed in chapter 2. Alternatively, one may consider to design the controller in continuous-time and implement the resulting controller digitally. This approach does not suffer from the above problems and also seems more appropriate as the physical systems are inherently continuous. Unfortunately, there has been little attention towards this approach (Egardt 1979a, 1979b; Elliott 1982a, 1982b; Gawthrop 1982, 1986, 1987). In particular, the work of Gawthrop is interesting as he reformulates the discrete-time GMV self-tuner in continuous-time by using the notion of emulator, which is referred to as *Emulator-Based Control* (EBC).

In this thesis, we are interested in developing some new continuous-time self-tuning algorithms which are robust, versatile and easy to use. For this purpose, three novel methods are proposed: emulator-based relay control (Demircioglu, 1988), continuous-time generalised predictive control (Gawthrop, 1989) and its relay version. In addition, the continuous-time generalised predictive control is extended to the multivariable case. These methods are combined with the continuous-time least squares algorithm to give their self-tuning versions. The continuous-time least squares is chosen specially to maintain the continuous-time nature of the algorithms although it is possible to use a discrete recursive estimation method. It should be noted that in this thesis we are mainly concerned with the algorithmic developments and the analysis of the proposed

underlying control methods rather than stability, convergence or robustness analysis of the associated self-tuning versions.

Chapter 2 starts with a critical review of discrete-time methods and proceeds with a summary of the necessary background material for the continuous-time methods considered in this thesis. This material includes: emulators, emulator-based control, continuous-time least squares and self-tuning emulators. The work reported in this thesis is built on this material and therefore it should be noted that the understanding of this material is important. The chapter ends with some discussion on the implementation of continuous-time self-tuning algorithms.

Chapter 3 first gives some background material on relay control and then describes the proposed emulator-based relay control method. The method is based on the idea of constructing the switching surface by emulators, that is unrealisable output derivatives are replaced by their emulated values. In particular, the relay is forced to operate in the sliding mode. In this case, it is shown that emulator-based and emulator-based relay control are equivalent. This clearly means that control methods obtained in the emulator-based control such as model-reference, pole-placement, predictive control and their detuned versions can also be obtained in the relay case. The properties of the proposed method for both nonadaptive and adaptive case are illustrated by a number of simulations. A real experiment (level control of a two cascaded tanks) is also given to show the effectiveness of the method. This chapter is based on the paper (Demircioglu, 1988).

Chapter 4 proposes a continuous-time version of the discrete-time generalised predictive control method of Clarke et al (Clarke, 1987). A detailed closed-loop analysis of the proposed method, continuous-time generalised predictive control (CGPC), is then given. Further, the relations of the CGPC with the state feedback and LQ control are established. The method is also extended to include some design polynomials so that model-following and pole-placement control can be considered in the same framework. In addition, a relay version of the CGPC is described. Again relay is forced to operate in the sliding mode. The CGPC and its relay version are shown to be equivalent in this case. Effects of the design parameters and polynomials, and the properties of the both methods are illustrated by a large number of simulations. Some of this work has been reported in a conference paper (Gawthrop, 1989).

Chapter 5 extends the CGPC method to the $p \times m$ multivariable systems. A detailed closed-loop analysis is given and the relation with the LQ control is pointed out. The conditions for decoupling and model-following type control are established. The properties of the method are illustrated by simulation.

Chapter 6 concludes the thesis and suggests possible future research areas.

Finally, the contributions of the thesis can briefly be summarised as developing and analysing some new self-tuning algorithms in a continuous-time framework for both scalar and multivariable systems. More precisely, the material given in chapters 3, 4 and 5 is original and thus forms the contribution of this thesis.

CHAPTER 2

CONTINUOUS-TIME SELF-TUNING CONTROL

2.1. INTRODUCTION

Developments in self-tuning control have been mainly within a discrete-time framework. It seems that this is partially because discrete-time methods are more appropriate for digital implementation and partially because the first developments took place in a discrete-time framework (Astrom, 1973) leading the later researchers in that direction. However, as has been pointed out in the literature (Wellstead, 1979, Gawthrop, 1980, Gawthrop, 1982, Astrom, 1984, Clarke, 1984, Sinha, 1985) there are some problems associated with discrete-time methods. These are mainly as follows:

1- *Nonminimum phase zeros:* This is the main criticism of the discrete-time methods. If a continuous-time system with relative degree > 2 is sampled at a fast sample rate with respect to system time constant then, some zeros of the corresponding discrete system will definitely be outside the unit circle. Nonminimum phase system zeros may also occur if the time delay is not integer multiple of the sample interval, due to the resulting fractional delay. Although the above problems can be overcome by choosing a larger sample interval, this would not be desirable as the resulting controller will be slow to respond to disturbances and setpoint changes. As a result, it will not be wrong to say that most of the discrete-time models of real systems will be nonminimum phase. This means that the chance of practical applications of discrete-time methods based on the cancellation of the open-loop system zeros such as model reference and minimum variance will be very limited.

2- *Numerical sensitivity:* Poles and zeros of a discrete-time system model depend on the sampling rate. For fast sampling rate, with respect to system time constant, the poles and m zeros† cluster round the $1+j0$ point in the z-plane, even though the corresponding continuous-time model may have its poles and zeros scattered widely over the s-plane. This produces a large sensitivity to numerical errors caused by truncation and round off in computation. In other words, discrete methods are numerically ill-conditioned.

3- *Sample rate selection:* From the above discussion it is clear that selection of the sample rate is an important issue in discrete control: a fast sample rate leads to nonminimum phase zeros and numerical sensitivity, on the other hand a slow sample rate leads to degraded control performance (eg. slow response to disturbances and setpoint changes). A suitable selection of the sample interval necessitates some knowledge of the system, such as time constant, closed-loop bandwidth, etc. The issue becomes even more important in the adaptive case as the estimated parameters depend on the sample interval. If the sample interval is not chosen adequately the estimates can be quite far from the actual system parameters (Sinha, 1985). So, choice of sample interval is not straightforward for discrete-time methods and it needs great care as well as some additional a priori information.

4- *Difficulty in interpretation of the discrete parameters:* The coefficients of discrete-time models depend on the sampling rate. In addition, the knowledge of the relative order is lost due to sampling. Therefore it is not easy to relate the discrete coefficients to the properties of the actual physical system.

An alternative to pure discrete-time design methods is to perform the design in a continuous-time framework and implement the resulting continuous-time controller digitally. This approach does not suffer from the above problems as the choice of sample interval is left after the design. In addition, a continuous-time approach seems more appropriate as the real systems to be controlled are inherently continuous. However, there has not been much attention towards the design of self-tuning controllers in a continuous-time framework. A noticeable work is that of

† Where m is the number of zeros of the corresponding continuous-time model

Gawthrop (Gawthrop, 1987) which is based on the author's earlier work on the subject (Gawthrop, 1980, Gawthrop, 1980, Gawthrop, 1982, Gawthrop, 1986). Some continuous-time algorithms are also given by Egardt (Egardt, 1979a, Egardt, 1979b) and Elliott (Elliott, 1982a, Elliott, 1982b).

This chapter is aimed to provide the necessary background for the work reported in this thesis. It is mainly based on the work of Gawthrop (Gawthrop 1987). Organization of the chapter is as follows. In section 2, description of the system considered is given. Section 3 introduces the idea of emulators and describe how to design emulators for the unrealisable operations such as, taking derivatives, canceling nonminimum phase zeros, and removing time delay. Section 4 examines the closed-loop system resulting from incorporating an emulator into the feedback loop. In section 5, a continuous-time version of the well-known discrete least squares is described. Section 6 combines the emulator based control methods of section 4 with the continuous-time least squares of section 5. In section 7, some implementation aspects of continuous-time self-tuning algorithms are discused.

2.2. SYSTEM DESCRIPTION

The system considered is single-input single-output and described in Laplace transform terms by the following equation

$$Y(s) = e^{-sT}\,\frac{B(s)}{A(s)}\,U(s) + \frac{C(s)}{A(s)}\,V(s) \tag{2.1}$$

where

$$A(s) = a_0 s^{n_a} + a_1 s^{n_a-1} + \ldots . + a_{n_a} \tag{2.2}$$

$$B(s) = b_0 s^{n_b} + b_1 s^{n_b-1} + \ldots . + b_{n_b} \tag{2.3}$$

$$C(s) = c_0 s^{n_c} + c_1 s^{n_c-1} + \ldots . + c_{n_c} \tag{2.4}$$

$$n_b \leq n_a \tag{2.5}$$

and e^{-sT} is the time delay term. $Y(s)$, $U(s)$ and $V(s)$ are the system output, control input and

disturbance input respectively.

No special assumptions are placed on the disturbance $V(s)$ and thus the polynomial $C(s)$ will be considered as a design polynomial having all its roots in the left-half s-plane. The degree of $C(s)$ will depend on the characteristics of the disturbances. In many cases, the disturbance component of the system is such that we would not wish to differentiate it, this can be modeled by choosing $n_c = n_a - 1$. An even worse case would be when we would not wish even to use the system output directly. This can be modeled by choosing $n_c = n_a$. It will be seen later that, the polynomial $C(s)$ acts as an observer polynomial.

2.3. EMULATORS

The notion of emulator was first introduced in (Gawthrop, 1986) to describe the dynamic systems which emulate unrealisable operations. Examples of such unrealisable operations in control systems design are: taking derivatives of the output, canceling nonminimum phase system zeros and removing time delay by an inverse delay (prediction). The idea of emulating unrealisable operations are further discussed in (Gawthrop, 1987). In this section, we will review those ideas.

2.3.1. Output Derivatives

In the presense of the noise, the operation of taking derivatives of the system output is not feasible as it amplifies the noise. In addition, all physically realisable systems have positive relative orders: they do not include any pure derivative term. Here, it will be shown that it is possible to emulate this unrealisable operation by using the system input and output.

Taking derivative of a signal in time domain corresponds to multiplication by s in Laplace domain (assuming zero initial conditions). Then, k^{th} derivative of the system output can be written in Laplace domain as

$$Y_k(s) = s^k Y(s) = \frac{s^k B(s)}{A(s)} e^{-sT} U(s) + \frac{s^k C(s)}{A(s)} V(s) \tag{2.6}$$

the s^k multiplied disturbance transfer function can be decomposed into two parts

$$\frac{s^k C(s)}{A(s)} = E_k(s) + \frac{F_k(s)}{A(s)} \tag{2.7}$$

where

$$deg\,(F_k) = n_a - 1$$

$$deg\,(E_k) = k-1 \qquad \text{if} \quad n_c = n_a - 1 \cdot$$

$$deg\,(E_k) = k \qquad \text{if} \quad n_c = n_a$$

The transfer function $\frac{F_k(s)}{A(s)}$ represents the strictly proper (realisable) part of $\frac{s^k C(s)}{A(s)}$ and $E_k(s)$ the improper remainder (unrealisable).

Using identity (2.7) $Y_k(s)$ may be written as the sum of an emulated value $Y_k^*(s)$ and the corresponding error $E_k^*(s)$

$$Y_k(s) = Y_k^*(s) + E_k^*(s) \tag{2.8}$$

where

$$Y_k^*(s) = \frac{s^k B(s)}{A(s)} e^{-sT} U(s) + \frac{F_k(s)}{A(s)} V(s) \tag{2.9}$$

and

$$E_k^*(s) = E_k(s)\, V(s) \tag{2.10}$$

Eqn. (2.9) can not be implemented as $V(s)$ is unknown. But from the system equation (2.1)

$$V(s) = \frac{A(s)}{C(s)} Y(s) - \frac{B(s)}{C(s)} e^{-sT} U(s) \tag{2.11}$$

Substituting eqn. (2.11) into eqn. (2.9) and using identity (2.7) one can then find the following expression for the emulated value of the k^{th} derivative of the output.

$$Y_k^*(s) = \frac{E_k(s)B(s)}{C(s)} e^{-sT} U(s) + \frac{F_k(s)}{C(s)} Y(s) \tag{2.12}$$

One can also easily show that when there are no disturbances ($V(s)$=0) $Y_k^*(s) = Y_k(s)$.

Notice that the relative order of $\frac{E_k(s)B(s)}{C(s)}$ is $\rho - k$, where ρ is the relative order of the system. For this term to be realisable we must have $k \leq \rho$. The transfer function $\frac{F_k(s)}{C(s)}$ is

proper.

Note that the equations leading to eqn. (2.12) are *algebraicly* equivalent to those leading to a discrete time k-step ahead predictor, but the interpretation is different.

Markov recursion

The polynomials $E_k(s)$ and $F_k(s)$ in eqn. (2.7) can be calculated recursively as follow:

$$e_{k+1} = \frac{f_{k0}}{a_0} \tag{2.13}$$

$$E_{k+1}(s) = sE_k(s) + e_{k+1} \tag{2.14}$$

$$F_{k+1}(s) = sF_k(s) - e_{k+1}A(s) \tag{2.15}$$

where f_{k0} is the coefficient of s^{n_a-1} in $F_k(s)$. The initial polynomials are given by the following identity.

$$\frac{C(s)}{A(s)} = E_0(s) + \frac{F_0(s)}{A(s)} \tag{2.16}$$

The name *'markov recursion'* comes from the fact that the coefficients of the polynomial $E_k(s)$ are the markov parameters of the transfer function $\frac{C(s)}{A(s)}$ and in this way markov parameters of any transfer function can be calculated recursively. Details of the derivation of the algorithm can be found in (Gawthrop, 1987).

2.3.2. Zero Canceling

Assume that we want to cancel the open-loop system zeros

$$Y_b(s) = \frac{1}{B(s)} Y(s) \tag{2.17}$$

if the system has some nonminimum phase zeros this operation will not be feasible as $B(s)$ is unstable. But, it can be emulated by using a similar method to the previous subsection. Consider the explicit form of eqn. (2.17)

$$Y_b(s) = \frac{1}{A(s)} e^{-sT} U(s) + \frac{C(s)}{B(s)A(s)} V(s) \tag{2.18}$$

the term $\frac{C(s)}{B(s)A(s)}$ can be divided into two parts (realisable and unrealisable)

$$\frac{C(s)}{B(s)A(s)} = \frac{E_b(s)}{B(s)} + \frac{F_b(s)}{A(s)} \tag{2.19}$$

where we impose the condition

$$deg(F_b) = n_a - 1 \tag{2.20}$$

then

$$deg(E_b) = n_b - 1 \tag{2.21}$$

The term $\frac{E_b(s)}{B(s)}$ is considered as the unrealisable part of $\frac{C(s)}{B(s)A(s)}$ as $B(s)$ is unstable. Using identity (2.19) $Y_b(s)$ can be written as the sum of an emulated value $Y_b^*(s)$ and the corresponding error $E_b^*(s)$

$$Y_b(s) = Y_b^*(s) + E_b^*(s) \tag{2.22}$$

where

$$Y_b^*(s) = \frac{1}{A(s)} e^{-sT} U(s) + \frac{F_b(s)}{A(s)} V(s) \tag{2.23}$$

and

$$E_b^*(s) = \frac{E_b(s)}{B(s)} V(s) \tag{2.24}$$

Substituting eqn. (2.11) into eqn. (2.23) and using identity (2.19), the emulated value $Y_b^*(s)$ can be written in terms of system input and output as

$$Y_b^*(s) = \frac{E_b(s)}{C(s)} e^{-sT} U(s) + \frac{F_b(s)}{C(s)} Y(s) \tag{2.25}$$

Note that both $\frac{E_b(s)}{C(s)}$ and $\frac{F_b(s)}{C(s)}$ are proper transfer functions. Note also that $Y_b^*(s) = Y_b(s)$ when $V(s) = 0$.

Remark: One may divide the polynomial $B(s)$ into two parts as $B(s) = B^-(s)B^+(s)$, where $B^-(s)$ is the unwanted part of $B(s)$ (such as nonminimum phase part), and only cancel $B^-(s)$.

Diophantine equation

Eqn. (2.19) can also be written as

$$C(s) = A(s)E_b(s) + B(s)F_b(s) \tag{2.26}$$

This equation is known as diophantine equation. It has a solution if and only if the greatest common factor of $A(s)$ and $B(s)$ divides $C(s)$. To have a unique solution at least one of the following conditions must hold (Astrom, 1984)

$$deg(F_b) < deg(A) \tag{2.27}$$

or

$$deg(E_b) < deg(B) \tag{2.28}$$

2.3.3. Prediction

The systems with time delay are difficult to control. An effective method for the control of such systems is to predict the future system output at time $t+T$ (T is the system time delay) and then used the predicted output in the feedback. This idea was suggested by Smith in late 50s (Smith, 1959). The same idea was also considered by Astrom in discrete-time minimum variance control (Astrom, 1970) which later constitute a basis for many discrete-time self-tuning algorithms.

The future output of the system at time $t+T$ can be written in Laplace transform terms as

$$Y_T(s) = e^{sT}Y(s) \tag{2.29}$$

The quantity $Y_T(s)$ can not be obtained from $Y(s)$ as e^{sT} is an unrealisable transfer function but, it may be emulated. To start with substitute the system eqn. (2.1) into eqn. (2.29)

$$Y_T(s) = \frac{B(s)}{A(s)}U(s) + e^{sT}\frac{C(s)}{A(s)}V(s) \tag{2.30}$$

As in the previous emulators design, the term $e^{sT}\frac{C(s)}{A(s)}$ can be divided into realisable and unrealisable parts

$$e^{sT}\frac{C(s)}{A(s)} = e^{sT}\bar{E}_T(s) + \frac{\bar{F}_T(s)}{A(s)} \tag{2.31}$$

$\frac{\bar{F}_T(s)}{A(s)}$ is a proper, rational transfer function representing the impulse response of $e^{sT}\frac{C(s)}{A(s)}$ for $t \geq 0$, and $\bar{E}_T(s)$ the transcendental transfer function representing (together with e^{sT}) the non-causal impulse response for $t < 0$. For example if $A(s) = s+a$ and $C(s) = 1$, then

$$\bar{E}_T(s) = \frac{1 - e^{-T(s+a)}}{s + a} \tag{2.32}$$

$$\bar{F}_T(s) = e^{-Ta} \tag{2.33}$$

After repeating the steps of the previous sections, one can obtain the following emulator

$$Y_T^*(s) = \frac{\bar{E}_T(s)B(s)}{C(s)}\,U(s) + \frac{\bar{F}_T(s)}{C(s)}\,Y(s) \tag{2.34}$$

with the corresponding error

$$E_T^*(s) = e^{sT}\bar{E}_T(s)\,V(s) \tag{2.35}$$

The problem with this emulator is that it includes a transfer function ($\bar{E}_T(s)$) which is not rational. To obtain a rational emulator $\bar{E}_T(s)$ should be approximated by a rational transfer function.

Alternatively, one may first consider approximating the time delay term by a rational transfer function so that the resulting emulator is rational. This is the approach used in the rest of the thesis. For this purpose, we will consider the pade approximation (Marshall, 1979) given by

$$e^{-sT} \approx \frac{T(-s)}{T(s)} \tag{2.36}$$

where $T(s)$ is a finite order polynomial in s

$$T(s) = t_0 s^{n_t} + t_1 s^{n_t-1} + \ldots + t_{n_t} \tag{2.37}$$

where

$$t_{n_t-i} = \frac{T(n_t-i+1)}{i(2n_t-i+1)}\,t_{n_t-i+1} \quad ; \quad t_{n_t} = 1 \tag{2.38}$$

Clearly, the approximation accuracy is determined by n_t.

Using the approximation for the time delay, the system can be approximately written as

$$Y(s) = \frac{T(-s)B(s)}{T(s)A(s)} U(s) + \frac{C(s)}{A(s)} V(s) \tag{2.39}$$

With the above approximate equations for the time delay and system, the quantity $Y_T(s)$ becomes

$$Y_T(s) = \frac{B(s)}{A(s)} U(s) + \frac{T(s)C(s)}{T(-s)A(s)} V(s) \tag{2.40}$$

The disturbance term can be divided into two parts as in the previous cases

$$\frac{T(s)C(s)}{T(-s)A(s)} = \frac{E_T(s)}{T(-s)} + \frac{F_T(s)}{A(s)} \tag{2.41}$$

where we again impose the condition

$$deg(F_T) = n_a - 1 \tag{2.42}$$

Then the resulting emulator is given by

$$Y_T^*(s) = \frac{E_T(s)B(s)}{T(s)C(s)} U(s) + \frac{F_T(s)}{C(s)} Y(s) \tag{2.43}$$

with the corresponding error

$$E_T^*(s) = \frac{E_T(s)}{T(-s)} V(s) \tag{2.44}$$

2.3.4. Generalized Emulator

One can obtain a generalization of the previous emulators by considering a quantity in the following form

$$\Phi(s) = e^{sT} \frac{P(s)}{Z(s)} Y(s) \tag{2.45}$$

where $P(s)$ and $Z(s)$ are polynomials in s and $deg(P) \geq deg(Z)$. The polynomial $Z(s)$ is divided into notionally realisable and unrealisable parts

$$Z(s) = Z^-(s)Z^+(s) \tag{2.46}$$

$Z^-(s)$ is regarded as the unrealisable part. This decomposition is not unique and particular choices

of $Z^-(s)$ and $Z^+(s)$ will depend on application. For example, if we want to cancel open-loop system zeros $Z^-(s) = B(s)$ or if we want to cancel a part of the open-loop zeros then

$$B(s) = B^-(s)B^+(s) \tag{2.47}$$

and $Z^-(s) = B^-(s)$. The following design rules are imposed:

1- $Z^+(s)$ contains no zeros with positive real part,

2- $Z(s)$ contains no zeros at $s = 0$.

Because of the same reason given in the previous subsection, the time delay term is approximated by a rational transfer function. Here, we again use pade approximation. Then the decomposition identity for the quantity $\Phi(s)$ will be

$$\frac{T(s)P(s)C(s)}{T(-s)Z(s)A(s)} = \frac{E(s)}{T(-s)Z^-(s)} + \frac{F(s)}{Z^+(s)A(s)} \tag{2.48}$$

where

$$deg(F(s)) = deg(Z^+(s)A(s)) - 1 \tag{2.49}$$

This identity leads to the following emulator

$$\Phi^*(s) = \frac{E(s)B(s)}{T(s)Z^-(s)C(s)}U(s) + \frac{F(s)}{Z^+(s)C(s)}Y(s) \tag{2.50}$$

with the corresponding error

$$E^*(s) = \frac{E(s)}{T(-s)Z^-(s)}V(s) \tag{2.51}$$

Eqn. (2.50) can be further written as

$$\Phi^*(s) = \frac{G(s)}{G_f(s)}U(s) + \frac{F(s)}{F_f(s)}Y(s) \tag{2.52}$$

where

$$G(s) = E(s)B^+(s) \tag{2.53}$$

$$G_f(s) = T(s)Z^{-+}(s)C(s) \tag{2.54}$$

$$F_f(s) = Z^+(s)C(s) \tag{2.55}$$

and $Z^{-+}(s)$ is the remaining part of $Z^{-}(s)$ after canceling out the common factors of $Z^{-}(s)$ and $B(s)$, and usually $Z^{-+}(s) = 1$.

2.4. EMULATOR BASED CONTROL (EBC)

There are some nasty components in physical systems such as time delay, nonminimum phase zeros and high relative order which make difficult to control such systems. As mentioned earlier, control of time delay systems can be simplified by using a predictor in the feedback loop. In the same way, control of the systems with nonminimum phase zeros and/or high relative order can also be simplified by using an appropriate emulator in the feedback loop. In this section, we will examine the closed-loop systems with an emulator in the feedback loop and show that with this approach a number of control methods such as model-reference, pole-placement and predictive control can be treated in a unified fashion.

The control law considered will be in the following form

$$U(s) = \frac{1}{Q(s)} \left[W(s) - \Phi^{*}(s) \right] \tag{2.56}$$

where $U(s)$, $W(s)$ and $\Phi^{*}(s)$ are the control signal, setpoint signal and emulator output; $Q(s)$ is the control weighting and $\frac{1}{Q(s)}$ is a proper transfer function. Here, we consider the generalised emulator as the others can be obtained as special cases.

The closed-loop system described by eqn. (2.56) is shown in figure 2.1. However, for simplicity in obtaining the closed-loop equations and properties of the emulator based control we will consider a notional feedback system given in figure 2.2.

Combining the equations forming figure 2.2 the following expressions for the closed-loop system are obtained:

Notional loop gain

$$L(s) = \frac{P(s)B(s)}{Q(s)Z(s)A(s)} \tag{2.57}$$

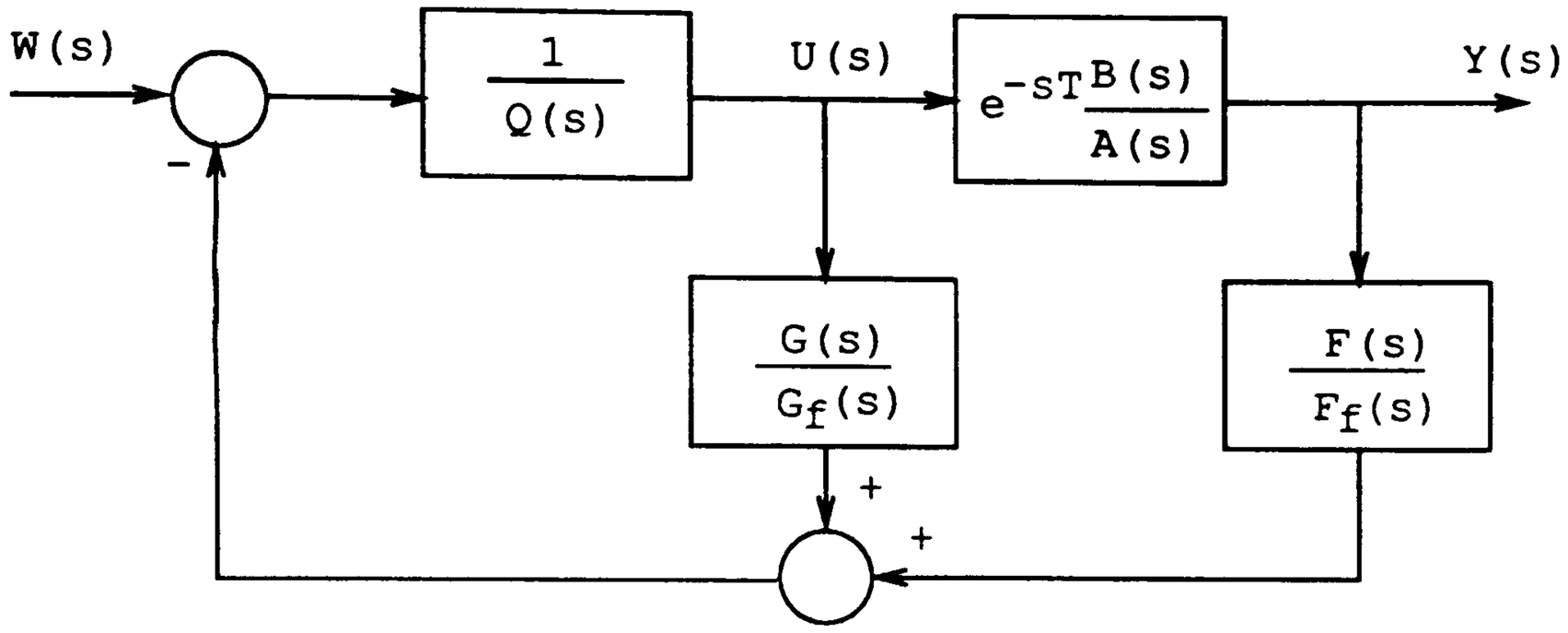


Figure 2.1 Emulator in the feedback loop

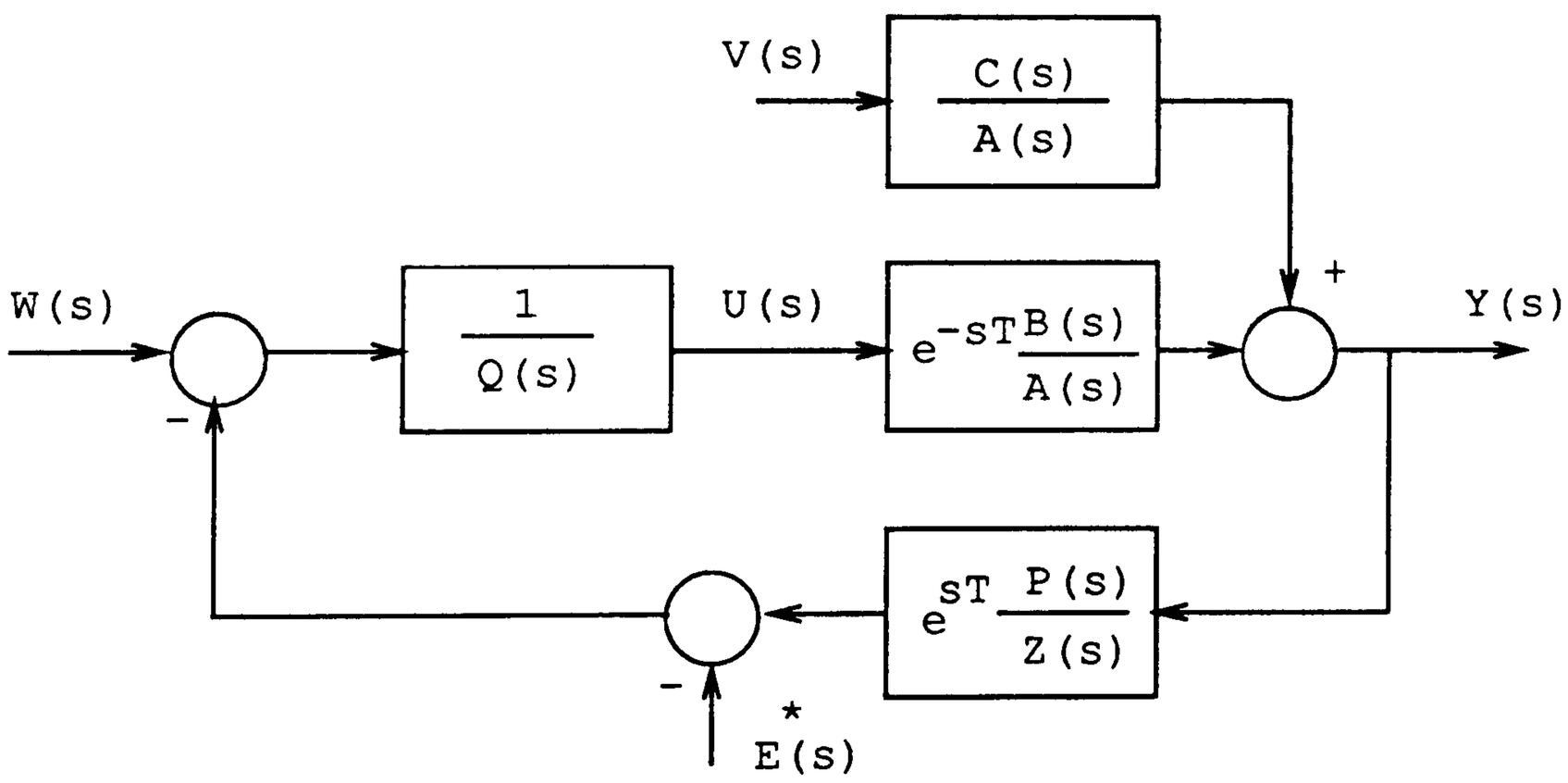


Figure 2.2 Notional feedback system

Closed-loop system output

$$Y(s) = e^{-sT}\,\frac{L(s)}{1+L(s)}\,\frac{Z(s)}{P(s)}\,[\,W(s)+E^{*}(s)\,]+\frac{1}{1+L(s)}\,\frac{C(s)}{A(s)}\,V(s) \tag{2.58}$$

$$= e^{-sT}\,\frac{B(s)Z(s)}{P(s)B(s)+Q(s)Z(s)A(s)}\,[\,W(s)+E^{*}(s)\,]+\frac{Q(s)Z(s)C(s)}{P(s)B(s)+Q(s)Z(s)A(s)}\,V(s) \tag{2.59}$$

Closed-loop system input

$$U(s) = \frac{L(s)}{1+L(s)}\,\frac{Z(s)A(s)}{P(s)B(s)}\,[\,W(s)-\frac{\bar{F}(s)}{A(s)Z^{+}(s)}\,V(s)\,] \tag{2.60}$$

First we will consider the case where there is no control weighting ($Q(s)$=0), and the systems without time delay (T=0). The closed-loop equations then becomes

Notional loop gain

$$L(s) = \infty \tag{2.61}$$

Closed-loop system output

$$Y(s) = \frac{Z(s)}{P(s)} [\, W(s) + E^*(s) \,] \tag{2.62}$$

Eqn. (2.62) shows that in this case, the closed-loop setpoint response is defined by a reference model $\frac{Z(s)}{P(s)}$.

Model-reference control

Consider the case where $B^-(s) = Z^-(s) = 1$. This gives us a model-reference control. Note that closed-loop system is not related to the open-loop system and the control signal will be stable if $B(s)$ is stable.

Pole-placement control

Consider the case where $B^-(s) = Z^-(s) = B(s)$ and $Z^+(s) = 1$. This gives us a pole-placement control. Note the zeros of the closed-loop system are identical to those of the open-loop system and the control signal will be stable even if $B(s)$ is not.

Detuned model-reference and pole-placement control

When $Q(s)$=0, the loop gain is infinite. Indication of this is that, we require exact model matching at all frequencies. However, in many cases there can be some unmodeled high frequency dynamics and thus requirement of exact model matching at all frequencies may not be met. This may also result in unstable control. Therefore, instead of choosing $Q(s)$=0, we may chose $Q(s)$ such that it is small at low frequencies but large at high frequencies. This will have the effect of giving exact model matching at low frequencies whilst not requiring exact model matching at

high frequencies. The use of $Q(s)$ in this way leads to detuned (or control weighted) version of the above control methods.

$Q(s)$ will be chosen zero at zero frequency that is, $Q(0)=0$. This is to ensure exact model matching at zero frequency and remove any offset due to control weighting.

As it is obvious from the above discussion, control weighting is important for robust control. See (Gawthrop, 1987) for the further discussion of this point.

Predictive control

For the time delay systems ($T\neq 0$), the emulator automatically includes a predictor. If $P(s)=Z(s)=1$ the emulator merely reduces to a predictor. If $P(s)\neq 1$ and $Z(s)\neq 1$, the resulting control laws can be regarded as predictive model-reference or predictive pole-placement depending on the choice of $P(s)$ and $Z(s)$. For $Q(s)\neq 0$ again detuned versions are obtained.

Integral action

The most effective method to remove offsets of any kind is to have an integral action in the controller. In the above control strategies, the integral action can automatically be obtained if the non-zero mean disturbances are modeled correctly. This corresponds to a system model as follows

$$Y(s) = e^{-sT}\,\frac{B(s)}{A(s)} + \frac{C(s)}{sA(s)}\,V(s) \tag{2.63}$$

Some remarks

1- Note that when $Q(s)=0$, the control law becomes $W(s) = \Phi^{*}(s)$.

2- It follows from the closed-loop equations that, $C(s)$ only affects the disturbance response, it has no effect on the setpoint response. It acts as an observer polynomial.

3- As one may notice, the above developments and analysis are in full analogy with the discrete generalised minimum variance control and thus emulator based control can be regarded as continuous-time version of the discrete generalised minimum variance control.

2.5. LEAST SQUARES ESTIMATION

Self-tuning control is merely a combination of a control method with a recursive estimation algorithm. The most popular and widely used estimation scheme is that of recursive least squares (RLS), which has been almost always used within a discrete-time framework: discrete RLS to estimate discrete-time model parameters. In this thesis, we are interested in continuous-time methods for the self-tuning control and so we need continuous-time model parameters. Although it is possible to use discrete RLS to estimate the parameters of a continuous-time model, it seems more consistent to use a continuous-time estimation scheme for our purpose. In this section, we will briefly review a continuous-time version of the well known discrete least squares. Details can be found in (Gawthrop, 1987).

We assume that the system to be identified can be described by the following linear-in-the-parameters model

$$y(t) = \underline{x}^T(t)\,\underline{\theta} + e(t) \tag{2.64}$$

where $y(t)$ is the scalar system output; $\underline{x}(t)$ and $\underline{\theta}$ are data and parameter vectors respectively; $e(t)$ is an error or noise term. Further we assume that $y(t)$ and $\underline{x}(t)$ can be measured (or can be obtained from measurements) and $\underline{\theta}$ is unknown. The aim is then to find an estimate $\underline{\hat{\theta}}(t)$ of θ based on all the measurements up to and including time t. For the least squares estimation, this is done by minimizing a cost function of the following form.

$$J(\underline{\hat{\theta}}(t),t) = \frac{1}{2}\, e^{-\beta t}\, (\underline{\hat{\theta}}(t) - \underline{\hat{\theta}}_0)^T S_0\, (\underline{\hat{\theta}}(t) - \underline{\hat{\theta}}_0) + \frac{1}{2} \int_0^t e^{-\beta(t-\tau)}\, \hat{e}(t,\tau)^2\, d\tau \tag{2.65}$$

where β is a non-negative scalar; S_0 is a positive definite matrix; $\underline{\hat{\theta}}_0$ is an initial parameter estimate; and the estimation error $\hat{e}(t,\tau)$ is defined as

$$\hat{e}(t,\tau) = y(\tau) - \underline{x}^T(\tau)\,\underline{\hat{\theta}}(t) \tag{2.66}$$

The exponential weighting coefficient β acts as a *forgetting factor*. As time t increases, the effect of old data at time $\tau < t$ is discounted exponentially with the elapsed time $t-\tau$. S_0 varies the weight given to the initial parameter estimate.

Minimization of the cost leads to the following solution for the parameter estimate

$$\underline{\hat{\theta}}(t) = S^{-1}(t) \left[e^{-\beta t} S_0 \, \underline{\hat{\theta}}_0 + \int_0^t e^{-\beta(t-\tau)} \, \underline{x}(\tau) \, y(\tau) \, d\tau \right] \tag{2.67}$$

where the matrix $S(t)$ is called *information matrix* and given by

$$S(t) = e^{-\beta t} S_0 + \int_0^t e^{-\beta(t-\tau)} \, \underline{x}(\tau) \, \underline{x}^T(\tau) \, d\tau \tag{2.68}$$

This equations can also be written in a recursive form as follows

$$S(t+T) = e^{-\beta T} S(t) + \int_t^{t+T} e^{-\beta(t+T-\tau)} \, \underline{x}(\tau) \, \underline{x}^T(\tau) \, d\tau \tag{2.69}$$

and

$$S(t+T)\underline{\hat{\theta}}(t+T) = e^{-\beta T} S(t) \, \underline{\hat{\theta}}(t) + \int_t^{t+T} e^{-\beta(t+T-\tau)} \, \underline{x}(\tau) \, y(\tau) \, d\tau \tag{2.70}$$

By using eqn.(2.69), eqn. (2.70) can be further written as

$$\underline{\hat{\theta}}(t+T) = \underline{\hat{\theta}}(t) + S^{-1}(t+T) \int_t^{t+T} e^{-\beta(t+T-\tau)} \, \underline{x}(\tau) \, [y(\tau) - \underline{x}^T(\tau) \, \underline{\hat{\theta}}(t)] \, d\tau \tag{2.71}$$

As one may notice, these equations are very similar to their discrete counterparts.

One can also show that equations (2.67) and (2.68) are the solutions of the following differential equations

$$S(t) \frac{d\underline{\hat{\theta}}(t)}{dt} = \underline{x}(t) \left[y(t) - \underline{x}^T(t) \, \underline{\hat{\theta}}(t) \right] \tag{2.72}$$

$$\frac{dS(t)}{dt} + \beta S(t) = \underline{x}(t) \, \underline{x}^T(t) \tag{2.73}$$

with initial conditions

$$\underline{\hat{\theta}}(0) = \underline{\hat{\theta}}_0 \quad ; \quad S(0) = S_0 \tag{2.74}$$

In eqn. (2.72) $S(t)$ needs to be inverted to obtain the parameter estimate $\underline{\hat{\theta}}(t)$. This matrix inversion can be avoided by the following reformulation.

$$\frac{d\underline{\hat{\theta}}(t)}{dt} = S^{-1}(t) \, \underline{x}(t) \left[y(t) - \underline{x}^T(t) \, \underline{\hat{\theta}}(t) \right] \tag{2.75}$$

$$\frac{-dS^{-1}(t)}{dt} + \beta S^{-1}(t) = S^{-1}(t)\underline{x}(t)\underline{x}^T(t)S^{-1}(t) \tag{2.76}$$

Clearly, in the above equations we assume that $S(t)$ is nonsingular.

2.6. SELF-TUNING CONTROL

Self-tuning version of the emulator based control method of section 2.4 can easily be obtained by combining it with a recursive parameter estimation algorithm. Here we will consider the continuous-time least squares given in the previous section. As mentioned in chapter 1, there are two types of self-tuning control method: indirect and direct. These two methods will now be considered in turn.

2.6.1. Indirect Method

In this method, first the system parameters are estimated and then based on these estimates the emulator design is performed. To be able to estimate the parameters of the system model given in eqn. (2.1), we need to write it in the linear-in-the-parameters form of eqn. (2.64).

Consider the system model with $T = 0$

$$A(s)Y(s) = B(s)U(s) + C(s)V(s) \tag{2.77}$$

Eqn. (2.77) can directly be written in linear in the parameters form but, in this case data vector will consist of pure derivatives of the input and output. These pure derivatives can be replaced by filtered ones by the following procedure: choose a filter polynomial $C_f(s)$ with the same degree as $A(s)$; divide both sides of eqn. (2.77) by $C_f(s)$; and add $Y(s)$ to both sides.

$$Y(s) + \frac{A(s)}{C_f(s)}Y(s) = \frac{B(s)}{C_f(s)}U(s) + \frac{C(s)}{C_f(s)}V(s) + Y(s) \tag{2.78}$$

Eqn. (2.78) can be rearranged as

$$Y(s) = \frac{B(s)}{C_f(s)}U(s) + \frac{C_f(s) - A(s)}{C_f(s)}Y(s) + \frac{C(s)}{C_f(s)}V(s) \tag{2.79}$$

Choose $c_0 = a_0$ where c_0 is the coefficient of highest power s term in $C_f(s)$, then

$$y(t) = \underline{x}^T(t)\,\underline{\theta} + e(t) \tag{2.80}$$

where

$$\underline{\theta}^T = [\, b_0 \;\; b_1 \;\; \ldots\ldots \;\; b_{n_b} \;\; c_1 - a_1 \;\; c_2 - a_2 \;\; \ldots\ldots \;\; c_{n_a} - a_{n_a} \,] \tag{2.81}$$

and the data vector $\underline{x}(t)$ and error term $e(t)$ are given in Laplace transform terms by

$$\underline{X}^T(s) = \frac{1}{C_f(s)} [[\, s^{n_b} \;\; s^{n_b-1} \;\; \ldots\ldots \;\; 1 \,]\, U(s) \;\; [\, s^{n_a-1} \;\; s^{n_a-2} \;\; \ldots\ldots \;\; 1 \,]\, Y(s)] \tag{2.82}$$

$$E(s) = \frac{C(s)}{C_f(s)}\, V(s) \tag{2.83}$$

This formulation is the same as that given by Gawthrop (Gawthrop, 1987) from an emulator point of view that is, designing an emulator for the system itself. A disadvantage of this formulation is that, although each entry in the data vector is filtered by $C_f(s)$, the output is not filtered. In a noisy environment, this gives rise to poor parameter estimates, specially if the signal to noise ratio is low. This problem can be avoided by a slightly different formulation: first add $Y(s)$ to both sides of eqn. (2.77) and then divide both sides by $C_f(s)$. Here, we also choose $deg\,(C_f(s)) > deg\,(A(s))$.

$$\frac{1}{C_f(s)}\, Y(s) + \frac{A(s)}{C_f(s)}\, Y(s) = \frac{B(s)}{C_f(s)}\, U(s) + \frac{C(s)}{C_f(s)}\, V(s) + \frac{1}{C_f(s)}\, Y(s) \tag{2.84}$$

This can be rearranged as

$$\frac{1}{C_f(s)}\, Y(s) = \frac{B(s)}{C_f(s)}\, U(s) + \frac{1 - A(s)}{C_f(s)}\, Y(s) + \frac{C(s)}{C_f(s)}\, V(s) \tag{2.85}$$

without lost of generality, we can take $a_{n_a} = 1$, then

$$y_f(t) = \underline{x}^T(t)\,\underline{\theta} + e(t) \tag{2.86}$$

where

$$\underline{\theta}^T = [\, b_0 \;\; b_1 \;\; \ldots\ldots \;\; b_{n_b} \;\; -a_0 \;\; -a_1 \;\; \ldots\ldots \;\; -a_{n_a-1} \,] \tag{2.87}$$

and the filtered output $y_f(t)$, data vector $\underline{x}(t)$ and $e(t)$ are given in Laplace transform terms

$$Y_f(s) = \frac{1}{C_f(s)}\, Y(s) \tag{2.88}$$

$$\underline{X}^T(s) = \frac{1}{C_f(s)} [[\, s^{n_b} \;\; s^{n_b-1} \;\; \ldots . \;\; 1 \,]\, U(s) \;\; [\, s^{n_a} \;\; s^{n_a-1} \;\; \ldots . \;\; s \,]\, Y(s)] \tag{2.89}$$

$$E(s) = \frac{C(s)}{C_f(s)} V(s) \tag{2.90}$$

Above, we assume that time delay is zero, if the time delay is not zero but is known, the control signal $U(s)$ can be replaced by a delayed version $U_T(s) = e^{-sT}U(s)$ in the above equations. If the delay is not known, it should be estimated together with system parameters. Time delay estimation in continuous-time is a quite involved problem and will not be considered here. Interested readers can refer to a recent thesis, which studies estimation and self-tuning control of time delay systems in a continuous-time framework, by Besharati-Rad (Besharati-Rad, 1988).

2.6.2. Direct Method

In this method, emulator parameters are identified directly so that the separate design phase is avoided. The emulator given by eqn. (2.52) can easily be written in the linear-in-the-parameters form required by the estimator.

$$\phi(t) = \underline{x}_e^T(t)\, \underline{\theta}_e + e^*(t) \tag{2.91}$$

where

$$\underline{\theta}_e^T = [\, g_0 \;\; g_1 \;\; \ldots . \;\; g_{n_g} \;\; f_0 \;\; f_1 \;\; \ldots . \;\; f_{n_f} \,] \tag{2.92}$$

and the data vector $\underline{x}_e(t)$ is given in Laplace transform terms

$$\underline{X}_e^T(s) = [\, \frac{1}{G_f(s)} [\, s^{n_g} \;\; s^{n_g-1} \;\; \ldots . \;\; 1 \,]\, U(s) \;\; \frac{1}{F_f(s)} [\, s^{n_f} \;\; s^{n_f-1} \;\; \ldots . \;\; 1 \,]\, Y(s)] \tag{2.93}$$

Eqn. (2.91) can not be used directly as $\phi(t)$ is not realisable but, a realisability filter $\Lambda(s)$ can be employed such that

$$\Phi_\Lambda(s) = \Lambda(s)\Phi(s) \tag{2.94}$$

is realisable. Then, the corresponding linear-in-the-parameters model

$$\phi_\Lambda(t) = \underline{x}_\Lambda^T(t)\, \underline{\theta}_e + e_\Lambda(t) \tag{2.95}$$

$$\underline{X}_{\Lambda}(s) = \Lambda(s)\underline{X}_{e}(s) \quad ; \quad E_{\Lambda}(s) = \Lambda(s)E^{*}(s) \tag{2.96}$$

is used with recursive least squares to estimate the emulator parameters. One possible choice of $\Lambda(s)$ is:

$$\Lambda(s) = e^{-sT}\,\frac{Z(s)}{P(s)} \tag{2.97}$$

giving

$$\phi_{\Lambda}(t) = y(t) \tag{2.98}$$

Note that above we assume that time delay is known.

In some control methods the polynomial $Z(s)$ includes a part (or all) of the open-loop zero polynomial $B(s)$, such as pole-placement where $Z(s)=B(s)$. In such cases two estimators are used in parallel: one for estimating $B(s)$, one for estimating the emulator parameters.

2.7. IMPLEMENTATION

As stated earlier, by continuous-time self-tuning we mean that underlying design method is continuous, not the implementation. Complexity of self-tuning controllers, without any doubt, necessitates their digital implementation. The continuous-time algorithms are essentially described by differential equations and there are many different ways of solving them numerically. In doing that, there are also many theoretical and practical considerations to be taken into account such as choice of sample interval, numerical stability, cost, speed etc. Here, our aim is not to discuss the best way of implementing continuous-time self-tuning algorithms, rather to provide the reader with some information about the implementation of simulations given in this thesis. Then, the first thing to say is that most of the simulations (except the simulations in chapter 3) were performed by using the MATLAB, a package program which is very convenient for the control system design and simulation, and we proceed as follows.

Implementation of the feedback systems

Feedback systems are merely an interconnection of different subsystems. They can be simulated either on the basis of each subsystem that is, implementing each subsystem separately and

then interconnecting them accordingly, or on the basis of an equivalent closed-loop system. We used generally the former approach as it is the convenient one for the adaptive simulation, and it is also the correct description of practical implementation. Each subsystem was implemented as follows:

1- as the subsystems in our case are transfer functions (or matrices), we first converted them into a state-space representation,

2- and choosing a proper sample interval, we obtained the corresponding discrete-time state-space representation.

The continuous to discrete conversion in MATLAB is done by assuming a zero order hold in the input and then calculating the matrix exponentials that is, for the following continuous-time system

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t) \tag{2.99}$$

the corresponding discrete-time system is:

$$\underline{x}(k+1) = \Phi\underline{x}(k) + \Gamma\underline{u}(k) \tag{2.100}$$

where

$$\Phi = e^{Ah} \quad ; \quad \Gamma = \int_0^h e^{A\tau} d\tau \, B \tag{2.101}$$

and h is the sample interval.

In the simulations, we chosed the sample interval small enough in order to approximate the corresponding continuous-time system as close as possible.

Implementation of the estimator

In the discrete least squares, inverse of the information matrix (so-called covariance matrix) is updated in order to avoid matrix inversion. For numerical reasons operation of updating is performed by factoring the covariance matrix and updating the factors such as square-root or U-D factorization algorithms (Bierman, 1977). This methods guarantee that the covariance matrix always remains positive definite and thus nonsingular. However, there may be some situations

where the system is overspecified (this is the case in some of our simulations). In such situations, the estimates are not unique (any common factors together with the actual parameters will be a solution to the estimation problem) and thus the information matrix is singular. Despite this, the above methods try to update the inverse of a matrix which does not have an inverse. This probably will give rise to some numerical problems. By taking this into account, in our simulations we updated the information matrix and then used the pseudoinverse of it to obtain the estimates (Lawson, 1974). This gives a unique solution which has a minimum Euclidean length among other solutions. Clearly, this way of implementation is not numerically efficient and thus is not suitable for the practical implementation, nevertheless we are interested in theoretical properties of the algorithms presented in this thesis and it suits our purpose well. Before giving some implementation details, it should also be noted that further work is needed to elucidate the fundamental problems arising from the essentially singular information matrix as a result of the overspecification of systems, but this is out of the scope of this thesis.

Continuous-time least squares given in section 2.5 can be implemented recursively by using either the integral (eqn. 2.69 and 2.70) or the differential (eqn. 2.72 and 2.73 or 2.76) equations formulation. It seems more sensible to implement the former ones, as they are the analytic solution of the later ones. In our implementation, we assumed that consecutive samples are connected to each other by a straight line, this gives better approximation than a zero order hold approach that is assuming that signals between two samples are constant and equal to the previous sample values. So, by taking T as the sample interval ($T=h$),equations (2.69) and (2.70) can be approximately written as follows

$$S(t+h) = e^{-\beta h}\left[S(t) + \frac{h}{2}\, \underline{x}(t)\, \underline{x}^T(t) \right] + \frac{h}{2}\underline{x}(t+h)\, \underline{x}^T(t+h) \tag{2.102}$$

$$S(t+h)\hat{\underline{\theta}}(t+h) = e^{-\beta h}\left[S(t)\, \hat{\underline{\theta}}(t) + \frac{h}{2}\, \underline{x}(t)\, y(t)\right] + \frac{h}{2}\, \underline{x}(t+h)\, y(t+h) \tag{2.103}$$

As discussed above, to obtain the parameter estimate the pseudoinverse of the information matrix was used in eqn. (2.103). It should be noted that, eqn. (2.103) is a set of linear equations and there are also other ways of solving it without explicitly taking a pseudoinverse.

CHAPTER 3

RELAY SELF-TUNING CONTROL

3.1. INTRODUCTION

Relay-based control systems have been used and analysed for many years (Flugge-Lotz, 1953, Atherton, 1982, Tsypkin, 1984). An interesting feature of such systems is that the resulting closed-loop system can be made unsensitive to parameter variations. This can be achieved in two different ways: 1) by using relay as a high gain element, 2) by forcing relay to operate in the sliding mode. The systems using the first approach are called self-oscillating adaptive system (SOAS) (Horowitz, 1974, Astrom, 1989). The problem with such systems is the existence of a limit cycle, which is unacceptable in many applications. The second approach is mainly used in variable structure systems (VSS); a more general form of switching control (Utkin, 1977, Utkin, 1978). The theory of VSS has also been used for designing robust model-following control systems (Young, 1978, Zinober, 1982). A disadvantage of such methods is the need to measure the system states in order to implement the switching surface. In this chapter, we will use the second approach that is, operating relay in the sliding mode. However, the method described here (Demircioglu, 1988) does not require measured system states: only the system output is required. There are two steps involved:

1. Implementation of the switching surface by replacing unrealisable derivatives by their emulated values.
2. Removal of the need of knowing the system parameters in the emulator design by using a self-tuning emulator.

This chapter is organised as follows. Section 2 reviews the necessary material from the theory of relay control systems. In section 3, the proposed method, *emulator based relay control,* is described and analysed. In section 4, a number of illustrative simulations for both non-adaptive and adaptive cases are given. Section 5 describes an experiment using a laboratory level control system and section 6 concludes the chapter.

3.2. RELAY CONTROL

As we mentioned earlier, the analysis and synthesis of relay control systems has a long history. Here, we will only review the material needed for the emulator-based relay control method described in this chapter.

3.2.1. System Description

The relay control system considered here is illustrated in figure 3.1 where $B(s)/A(s)$ is the transfer function of the linear open-loop system; $P(s)$ is a polynomial in the Laplace operator s; $W(s)$, $U(s)$ and $Y(s)$ are the setpoint signal, control signal, and the closed loop system output respectively; $E(s)$ is the relay input (error) signal.

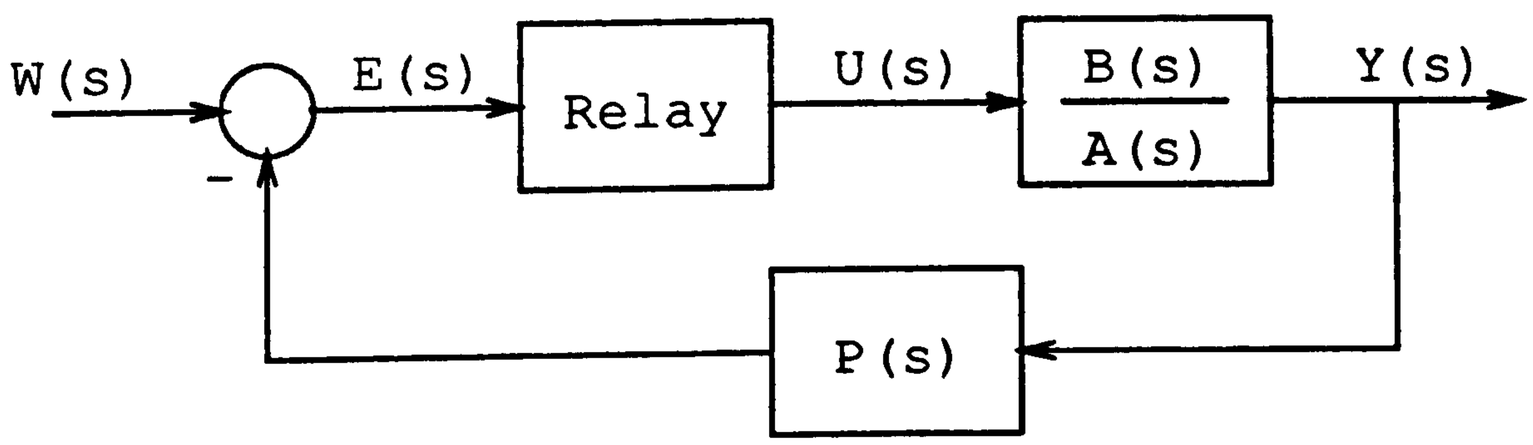


Figure 3.1 Relay control system

We assume that the relay is symmetric and ideal, that is, it has no dead zone and no hysteresis. In this case the input output relationship of the relay element is given by

$$u(t) = M\ sign(e(t)) = \begin{cases} M, & \text{if } e(t) \geq 0 \\ -M, & \text{if } e(t) < 0 \end{cases} \tag{3.1}$$

where M is the relay amplitude.

3.2.2. Sliding Motion In The Relay Control Systems

The relay output $u(t)$ is a sequence of rectangular pulses that change sign when the sign of $e(t)$ changes (figure 3.2). In figure 3.2 t_1, t_2, ... are called the switching times. Note that at the successive switching times t_k and t_{k+1} the direction of $e(t)$ is opposite, that is if $\dot{e}(t_k) > 0$, then $\dot{e}(t_{k+1}) < 0$ or vice versa where $\dot{e}(t)$ is the derivative of $e(t)$.

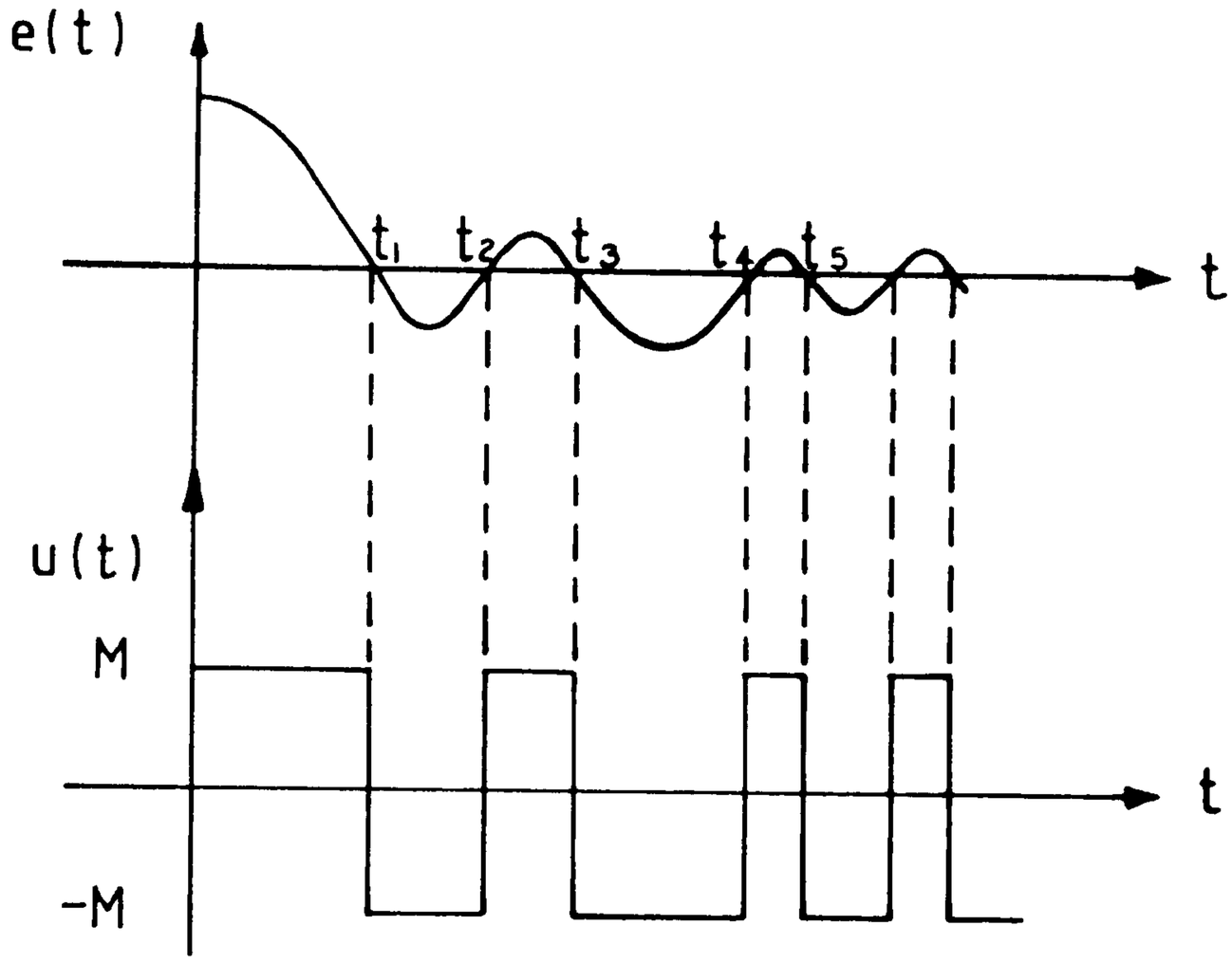


Figure 3.2 Relay input and output signals

Define

$$\dot{e}^{-}(t_1) = \dot{e}(t_1-0) \tag{3.2}$$

$$\dot{e}^{+}(t_1) = \dot{e}(t_1+0) \tag{3.3}$$

If at the first switching time t_1, $\dot{e}^{-}(t_1) \neq \dot{e}^{+}(t_1)$ and have opposite sign, that is

$$\dot{e}^{-}(t_1)\,\dot{e}^{+}(t_1) < 0 \tag{3.4}$$

then, when $e(t)$ crosses the threshold level, it immediately recrosses it giving the so-called sliding motion (Tsypkin, 1984). In the sliding motion the relay input $e(t)$ stays in the vicinity of the threshold level (zero) oscillating at a high frequency (figure 3.3) and the relay output oscillates at the same high frequency between $+M$ and $-M$.

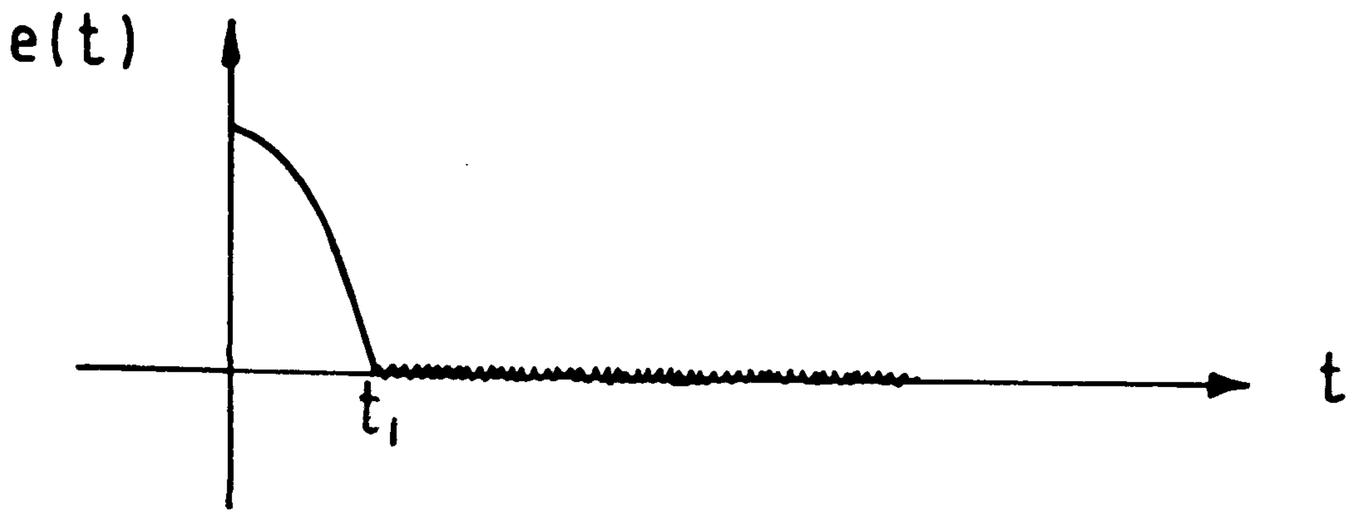


Figure 3.3 Relay input in the sliding motion

The inequality $\dot{e}^-(t_1)\,\dot{e}^+(t_1) < 0$ is the necessary and sufficient condition for the sliding motion to occur. A necessary condition for this inequality to be satisfied is that the relative order of the loop transfer function, $B(s)P(s)/A(s)$, must be unity.

For our purposes it is desirable to obtain the sliding mode just after the first switching time t_1 and keep t_1 as small as possible (this can generally be done by choosing M large). Because of this, above we consider the first switching time t_1. In fact the sliding motion which depends on the system parameters and the initial conditions occurs at the the time when the condition $\dot{e}^-(t_k)\,\dot{e}^+(t_k) < 0$ holds. This condition will certainly hold after some time if the relay control system is stable and the relative order of the loop transfer function is unity (Tsypkin, 1984).

In the sliding mode, the closed-loop system is approximately governed by the following equations

$$E(s) = 0 \tag{3.5}$$

$$E(s) = W(s) - P(s)Y(s) \tag{3.6}$$

from eqn. (3.5) and (3.6)

$$Y(s) = \frac{1}{P(s)}\,W(s) \tag{3.7}$$

It is clear that, in the sliding mode the input/output relationship of the closed loop system is independent of the open-loop system and defined by a reference model $1/P(s)$.

Similar analysis for the sliding motion can be done by using the state space theory (Utkin, 1977, Zinober, 1982). In the state space, $e(t) = 0$ defines a switching surface on which the control

signal has a discontinuity. For sliding motion to occur the following conditions must be satisfied in the neighbourhood of the switching surface.

$$\lim_{e(t)\to 0^+} \dot{e}(t)<0 \quad \text{and} \quad \lim_{e(t)\to 0^-} \dot{e}(t)>0 \tag{3.8}$$

this two conditions may be combined as

$$e(t)\dot{e}(t) < 0 \tag{3.9}$$

These conditions ensure that the motion of the state on either side of the switching surface (in the neighbourhood of the switching surface) is towards the switching surface. Thus the state remains on the switching surface and slides towards the direction indicated by the trajectories, hence the name sliding mode. Note that eqn. (3.9) and (3.4) are equivalent.

For the second order systems, state-space is reduced to a plane known as phase-plane. For such systems, the above analysis can be done graphically. More insight into the sliding motion can be gained by using this method, although it is limited to the second order systems. Below an example is given for this purpose.

An example

Consider the following system together with the feedback polynomial $P(s)$.

$$\frac{B(s)}{A(s)} = \frac{1}{s^2 - 1} \tag{3.10}$$

$$P(s) = 0.5s + 1 \tag{3.11}$$

The phase portrait of this system, which was obtained by a computer simulation, is given in figure 3.4. In this figure, it is assumed that input to the system is zero. If the input is not zero but constant, say U, the phase portrait will be still the same but about the point (-U,0)† rather than (0,0) (Atherton, 1982). In our case, input to the system is either M or $-M$ depending on the relay input.

† In general, for constant input U, the phase portrait will be about the point (δU,0) where $\delta = 1$ if the steady state gain of the system is positive, $\delta = -1$ if it is negative.

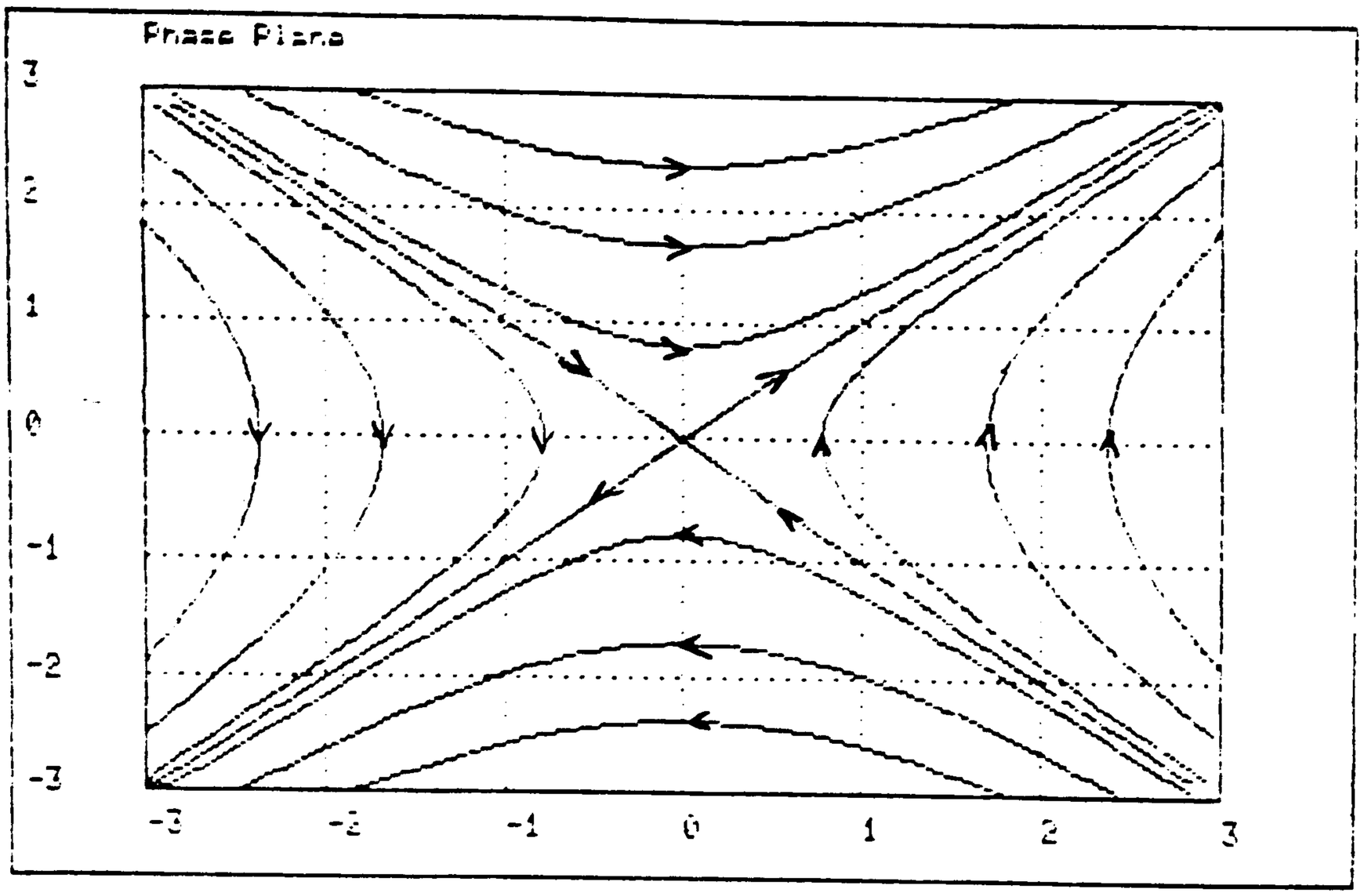


Figure 3.4 Phase portrait of $1/(s^2-1)$

The closed-loop system given by eqn. (3.10) and (3.11) is governed by the following equations in time domain

$$\ddot{y}(t) - y(t) = u(t) \tag{3.12a}$$

$$u(t) = M \ sign(e(t)) \tag{3.12b}$$

$$e(t) = w(t) - y(t) - 0.5\,\dot{y}(t) \tag{3.12c}$$

In addition to these equations, for the analysis we also need the derivative of the relay input $e(t)$; by assuming that setpoint is constant, say $w(t) = w$, it can be obtained as

$$\dot{e}(t) = -\dot{y}(t) - 0.5\,\ddot{y}(t) \tag{3.12d}$$

Recall the condition for the sliding motion (eqn. 3.9), which can be rewritten more explicitly as follows

$$when \quad e(t) > 0 \quad then \quad \dot{e}(t) < 0 \tag{3.13}$$

$$when \quad e(t) < 0 \quad then \quad \dot{e}(t) > 0 \tag{3.14}$$

By using (3.13) together with the above set of equations (3.12a to 3.12d), one can obtain the

following inequality

$$y(t) < \frac{4}{3}w + \frac{1}{3}M \tag{3.15}$$

In the same way from (3.14)

$$y(t) > \frac{4}{3}w - \frac{1}{3}M \tag{3.16}$$

These two inequalities can be combined as

$$\frac{4}{3}w - \frac{1}{3}M < y(t) < \frac{4}{3}w + \frac{1}{3}M \tag{3.17}$$

This interval of $y(t)$ corresponds to the sliding mode and it can be represented on the switching line as a line segment. Note that the interval is proportional to the relay amplitude M. As an example, consider the following choice of w and M

$$w = 1 \text{ and } M = 2 \tag{3.18}$$

then

$$0.66 < y(t) < 2 \tag{3.19}$$

and the switching line

$$y(t) + 0.5\,\dot{y}(t) = 1 \tag{3.20}$$

These can be plotted on the phase plane as illustrated in figure 3.5. The line segment KL of the switching line in the figure corresponds to the sliding mode region. Starting with zero initial conditions, the first switching time t_1 is obtain as $t_1 = 0.59$ sec and at the first switching time the output is $y(t_1) = 0.366$. These results are also shown in figure 3.5. It is clear that, when $M = 2$ and $w = 1$, the sliding mode will not occur after the first switching time t_1. However, as illustrated in the figure, following the second switching time the state enters the sliding mode region and slides down to the point (1,0).

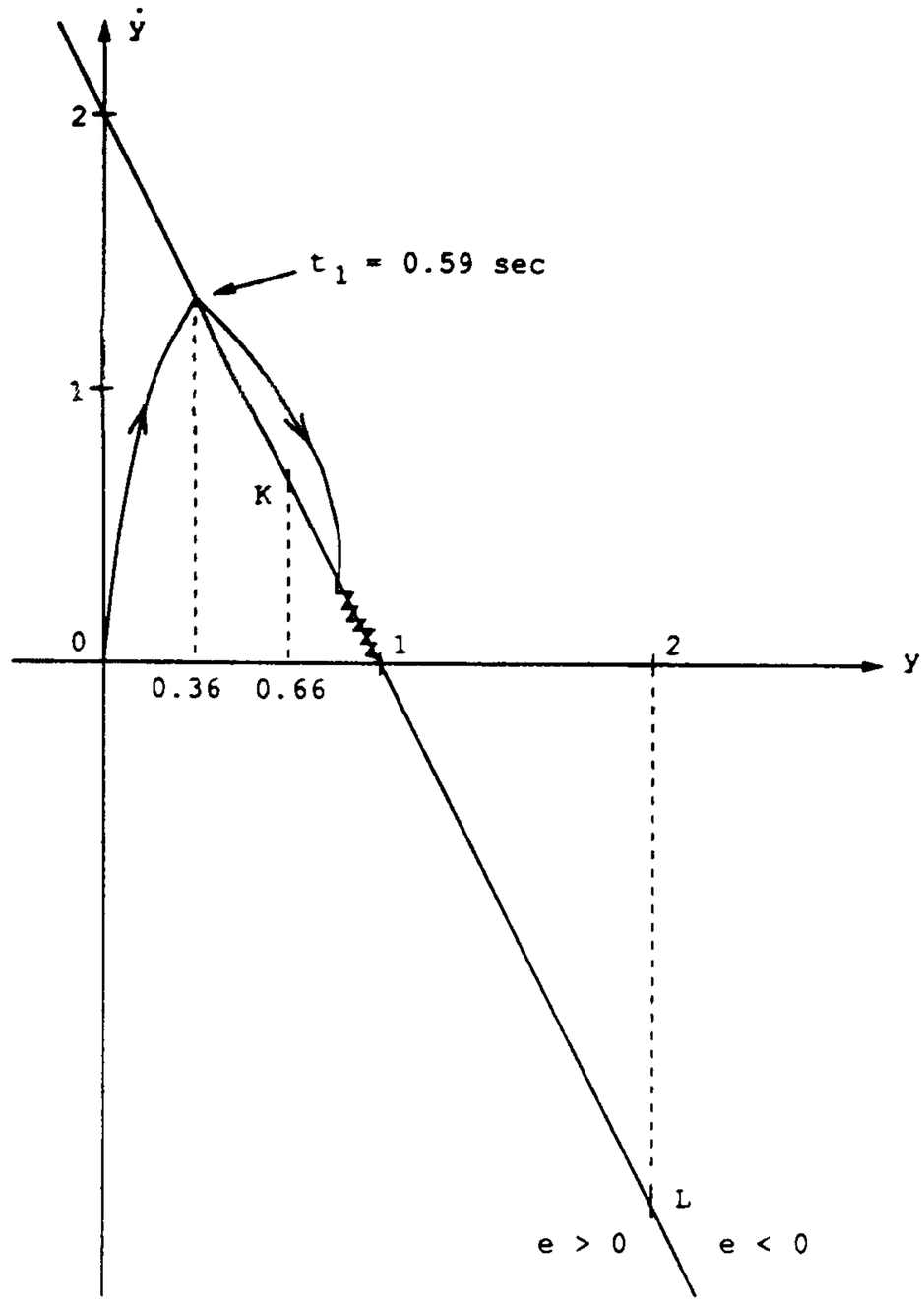


Figure 3.5 Analysis of sliding motion on the phase plane

By finding the value of the output at the first switching time t_1, $y(t_1)$, in terms of w and M and using inequality (3.16), the condition for the sliding mode to occur after the first switching time can be obtained as

$$M > 3w \tag{3.21}$$

In the calculations leading to (3.21), the initial conditions were assumed to be zero. It should be noted that the first switching time depends on the initial conditions.

For $M = 4$ and $w = 1$ the phase portrait of the closed-loop system is given in figure 3.6. In this case the sliding region corresponds to the interval $0 < y(t) < 2.66$. Note that at the first switching time the output is in the sliding region (starting with the zero initial conditions).

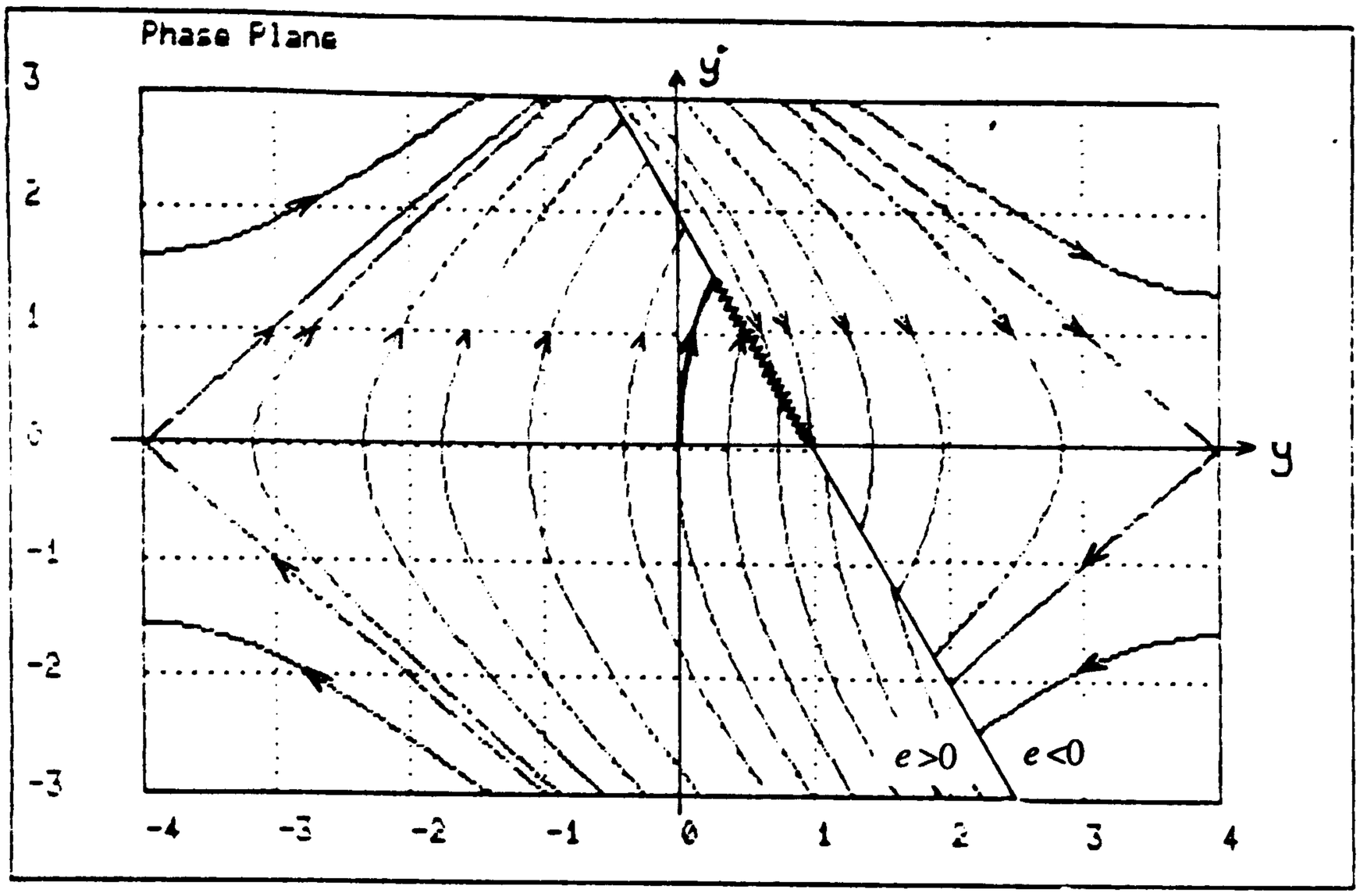


Figure 3.6 Phase portrait of the closed-loop system with M=4 and w=1

3.3. EMULATOR BASED RELAY CONTROL

The relay control method described in the previous section is based on availability of the output derivatives or the states. If, as is assumed here, only the system output is available then the quantity

$$\Phi(s) = P(s)Y(s) \tag{3.22}$$

is not physically realisable and thus the method is not feasible. However, if the quantity $\Phi(s)$ is replaced by the emulator output $\Phi^*(s)$ described in chapter 2 then the method becomes feasible. This is illustrated in figure 3.7. Recall that the emulated value, $\Phi^*(s)$, of $\Phi(s)$ is given as follows.

$$\Phi^*(s) = \frac{G(s)}{C(s)}\,U(s) + \frac{F(s)}{C(s)}\,Y(s) \tag{3.23}$$

where

$$G(s) = E(s)B(s) \tag{3.24}$$

$$P(s)C(s) = E(s)A(s) + F(s) \tag{3.25}$$

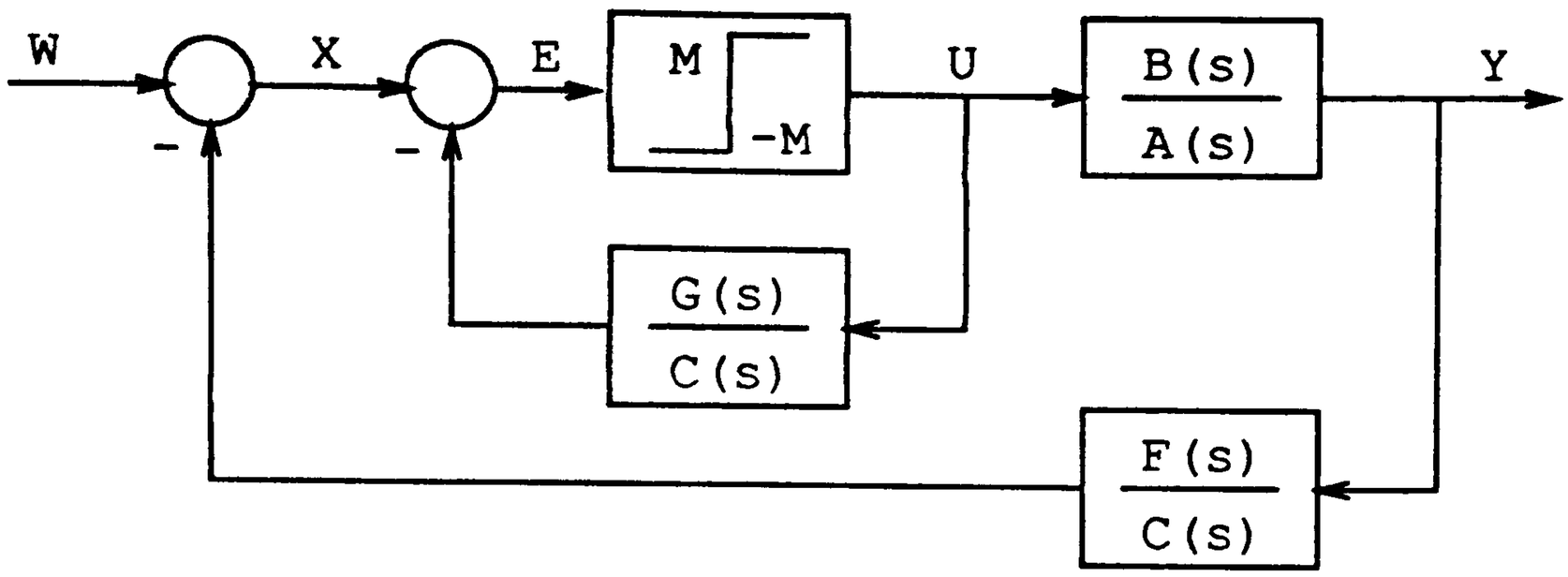


Figure 3.7 Emulator-based relay control system

Suppose that the system is operating in the sliding mode then the relationship relating $x(t)$ (see figure 3.7) to $u(t)$ will be defined by the transfer function $C(s)/G(s)$. Using this linear relationship, the response of the closed loop system to the setpoint can be obtained as

$$Y(s) = \frac{C(s)}{E(s)A(s) + F(s)} W(s) \tag{3.26}$$

from eqn. (3.25)

$$Y(s) = \frac{1}{P(s)} W(s) \tag{3.27}$$

As a result, when $P(s)$ is replaced by its emulator the input/output relationship of the closed loop system, in the sliding mode, will still be defined by a reference model $1/P(s)$. Equations in the sliding mode

$$e(t) = 0 \tag{3.28}$$

$$e(t) = w(t) - \phi^*(t) \tag{3.29}$$

from eqn. (3.28) and (3.29)

$$w(t) = \phi^*(t) \tag{3.30}$$

Note that this is the same control law obtained in chapter 2 by setting $Q(s)$ equal to zero. Namely, in emulator based control replacing the control weighting $Q(s)$ by a relay and operating the system in the sliding mode has the same effect as setting $Q(s)$ equal to zero. This implies that, not only the model-reference control, also the other control approaches obtained in chapter 2 by

setting $Q(s)$ equal to zero, such as pole-placement and predictive control, are also obtainable (by using the corresponding emulator) in this case.

In the above analysis, it is assumed that the system is operating in the sliding mode. For this system to operate in the sliding mode the condition $e(t)\dot{e}(t) < 0$ must hold. In order for this condition to be satisfied the relative order of the relay loop transfer function $G(s)/C(s)$ must be unity. The relative order of $G(s)/C(s)$ is equal to that of $P(s)B(s)/A(s)$. Therefore, we need to choose $P(s)$ such that the relative order of $P(s)B(s)/A(s)$ is unity, that is

$$deg(P) = deg(A) - deg(B) - 1. \tag{3.31}$$

There may be some cases where $P(s)B(s)/A(s)$ may have zero relative order, thus $G(s)/C(s)$ has also zero relative order. In such cases a transfer function with unit relative degree, for example a first order low pass filter, can be inserted into the relay loop before or after the relay to make the relative order of the relay loop transfer function unity. Note that this filter acts as a tool to obtain the sliding motion. It has no effects on the closed-loop system response when the system is in the sliding mode; that is in the sliding mode, despite this filter, the relationship relating $x(t)$ to $u(t)$ will still be defined by the transfer function $C(s)/G(s)$.

So far, in order not to obscure the main points, we only considered the emulator for the quantity $\Phi(s) = P(s)Y(s)$. As we stated above, it is also possible to use the emulator for the quantity (see chapter 2)

$$\Phi(s) = e^{sT}\,\frac{P(s)}{Z(s)}\,Y(s) \tag{3.32}$$

in the feedback-loop of the relay control system, which will enable us to obtain the same control laws as obtained in chapter 2. Here, we assume that the system has a time delay. In this case, relative order of $P(s)B(s)/Z(s)A(s)$ must be chosen unity to make relative order of the relay loop transfer function $G(s)/G_f(s)$ unity. Again if the relative order is zero, a first order low pass filter can be inserted into the relay loop as explained above.

3.3.1. Detuned Version of The Algorithms

As is explained by Gawthrop (Gawthrop, 1987, Gawthrop, 1987a) detuning is crucial for the robustness in the self-tuning control and as mentioned in chapter 2, it can be obtained by choosing $Q(s) \neq 0$. If the detuning is so important, then a question immediately arises as to how the detuned versions of the algorithms can be obtained in the relay case. The answer to this question is given in figure 3.8 where the control weighting $Q(s)$ is fed back around the relay. In this configuration if the relay operates in the sliding mode then the relay loop will be equivalent to the transfer function $1/Q(s)$. Thus, figure 3.8 will be reduced to figure 2.1 of chapter 2 which has the control law

$$U(s) = \frac{1}{Q(s)}[W(s) - \Phi^*(s)] \tag{3.33}$$

As is explained in chapter 2 this will give the detuned version of the algorithms if $Q(s)$ is not zero.

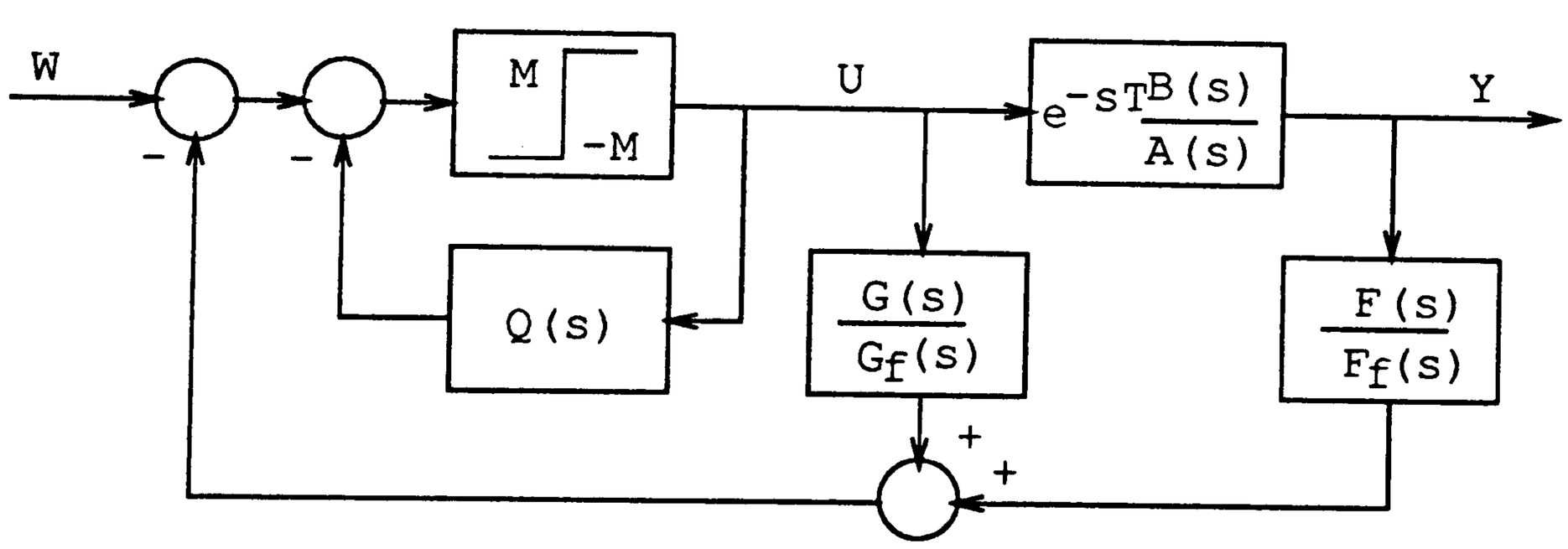


Figure 3.8 Detuned relay control

Note when $Q(s) = 0$ figure 3.8 reduces to figure 3.7 which is equivalent (in the sliding mode) to emulator based control with $Q(s) = 0$. Therefore, figure 3.8 is the general form of the emulator based relay control system.

In this case to obtain the sliding mode the transfer function $[Q(s)G_f(s)+G(s)]/G_f(s)$ must have a unity relative order. As $Q(s)$ is generally chosen to have zero relative order a first order low pass filter needs to be used in the relay loop.

3.4. SIMULATIONS

This section presents a number of simulated examples which illustrate the properties of the emulator based relay control for both non-adaptive and adaptive cases.The examples simulated are as follows:

Model-reference control

Example 1: $\frac{B(s)}{A(s)} = \frac{1}{s^2}$

$$Z(s) = 1$$

$$P(s) = 0.5s + 1$$

$$C(s) = 0.5s + 1$$

Example 2: $\frac{B(s)}{A(s)} = \frac{1}{s(s^2 + 1)}$

$$Z(s) = 1$$

$$P(s) = 0.333s^3 + 1.666s^2 + 0.833s + 1$$

$$C(s) = 0.5s^2 + s + 1$$

$$Low\text{-}pass\ filter = \frac{1}{s+1}$$

Pole-placement control

Example 3: $\frac{B(s)}{A(s)} = \frac{1-s}{s^2}$

$$Z(s) = 1-s$$

The rest is as in example 1

Example 4: $\frac{B(s)}{A(s)} = \frac{(1-s)}{s(s^2 + 1)}$

$$Z(s) = 1-s$$

The rest is as in example 2

Predictive model-reference control

Example 5: $\frac{B(s)}{A(s)} = \frac{e^{-s}}{(s+1)^2}$

The rest is as in example 1

Detuned model-reference control

Example 6: $\frac{B(s)}{A(s)} = \frac{2}{s+1}$

$N(s) = \frac{100}{s^2 + 8s+100}$ *(neglected dynamics)*

$Z(s) = 1$

$P(s) = 0.3s+1$

$C(s) = 0.3s+1$

$Q(s) = \frac{0.15s}{0.02s+1}$

Low–pass filter $= \frac{1}{10s+1}$

Simulations were performed by using a PASCAL programme† running on a SUN 3 workstation. In these simulations the followings are common:

1- the sample interval is 0.01 time unit;

2- each figure consists of three graphs for non-adaptive simulations:

(a) the upper graph shows the setpoint (square wave), the actual system output and the model output; the model output $Y_m(s)$ corresponds to

$$Y_m(s) = e^{-sT} \frac{Z(s)}{P(s)} W(s) \tag{3.34}$$

(b) the middle graph shows the control signal (relay output);

(c) the lower graph shows the relay input;

† The programme called CSTC was written by Gawthrop. Its implementation details together with a large number of simulated examples are soon going to be published as an accompanying volume to the first volume of his book 'Continuous-time self-tuning control'.

and four graphs for adaptive simulations:

(d) the above three plus estimated parameters;

3- all adaptive simulations start with a set of wrong parameters.

3.4.1. Non-adaptive Simulations

Example 1 was simulated for two different relay amplitudes, $M = 5$ and $M = 30$, to illustrate the effect of the relay amplitude. Simulation results are given in figure 3.9 and 3.10 respectively. Note that at the initial setpoint change, the system is in the sliding mode following the first switching time t_1 for both $M = 5$ and $M = 30$. However, as the setpoint changes from 1 to -1 or vice versa, the sliding motion occurs after the second switching time for $M = 5$ whereas it remains the same for $M = 30$. It is clear that, when M is small, the first switching time t_1 is large and this prevents the system output from following the model output perfectly. When M is large, t_1 becomes very small and the system output perfectly follows the model output.

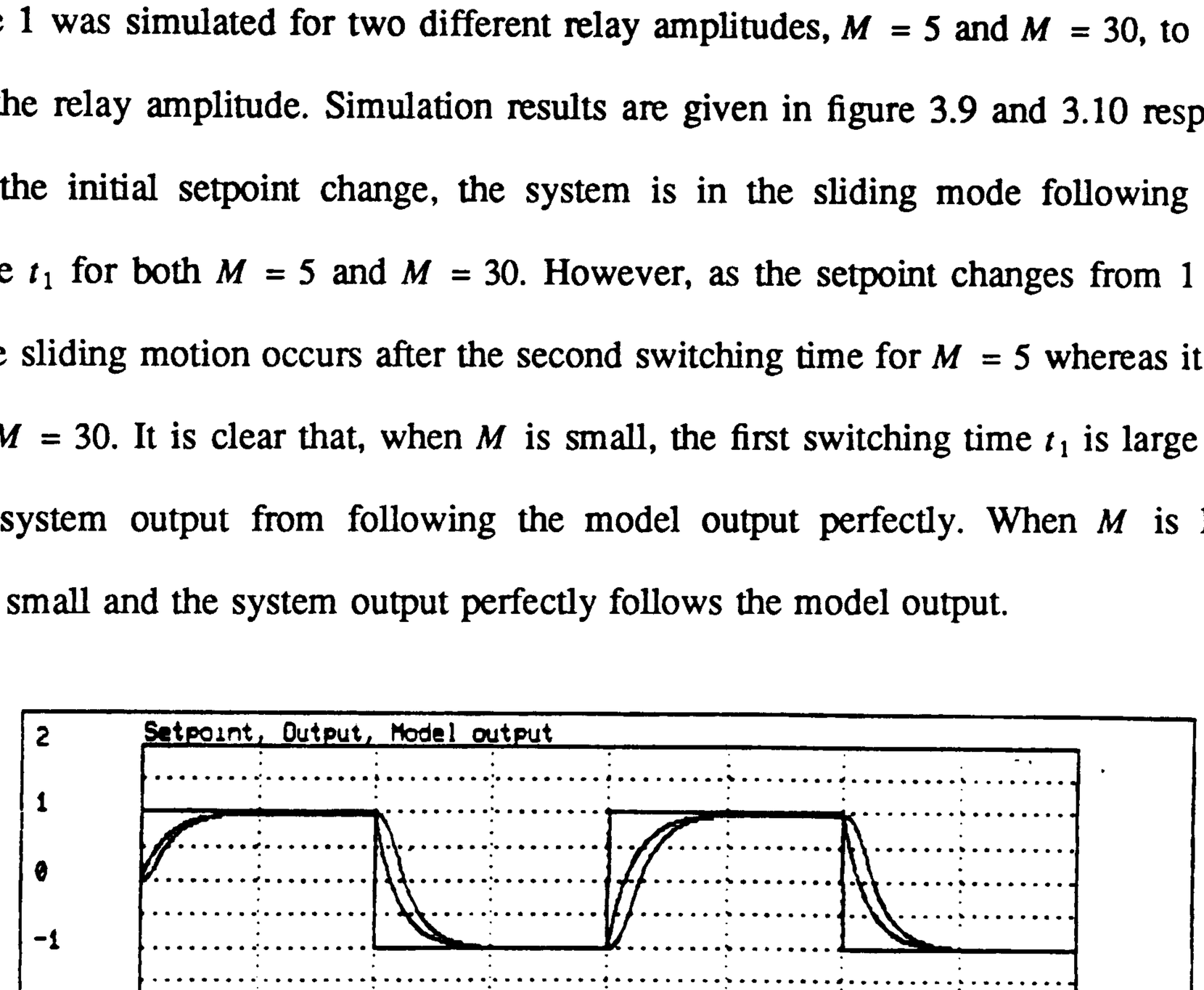


Figure 3.9 Example 1 with $M = 5$

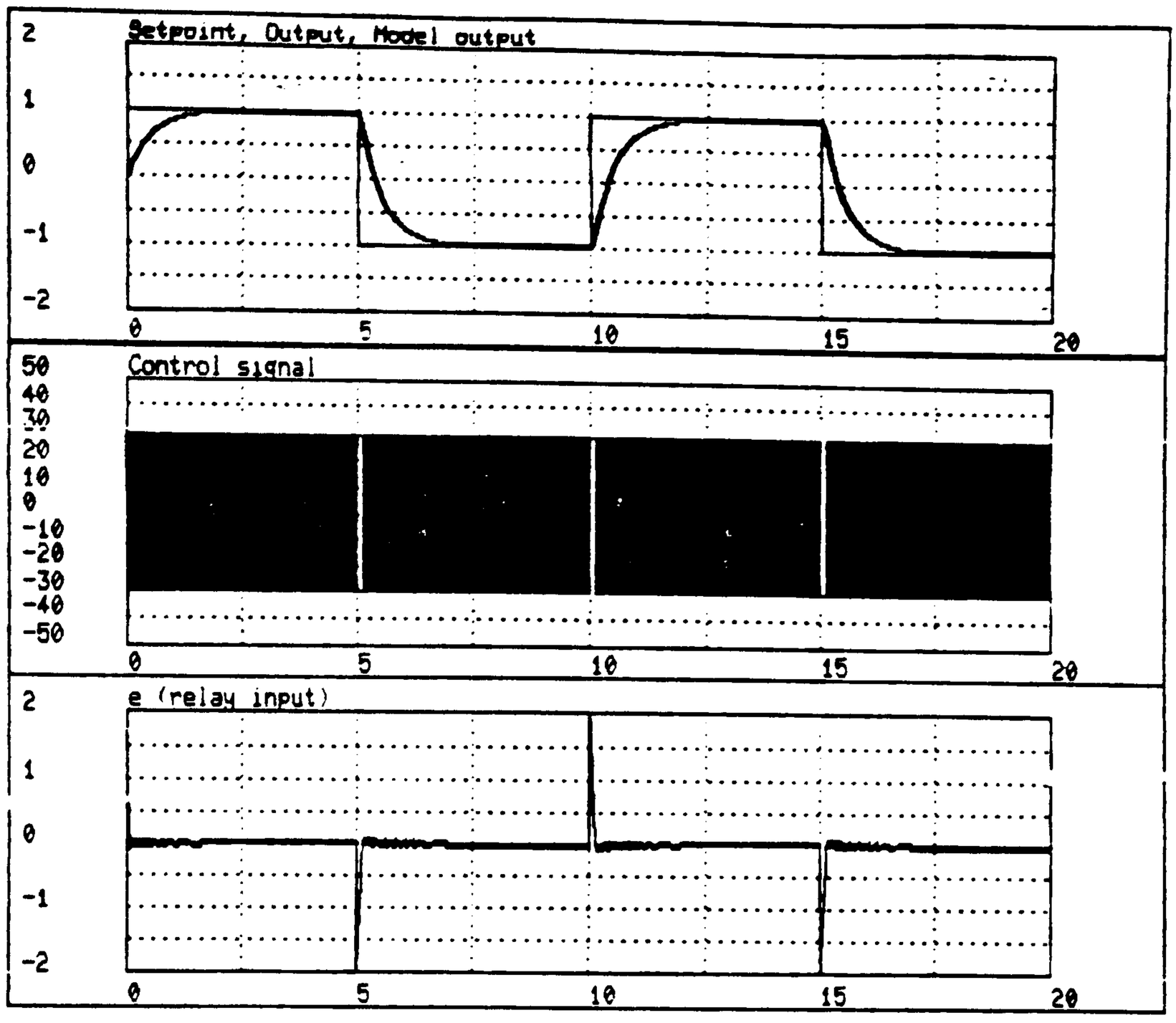


Figure 3.10 Example 1 with $M = 30$

The simulation result of example 2 is given in figure 3.11 for $M = 10$. From the lower graph it can be seen that the relay input is always in the vicinity of zero indicating that system is always in the sliding mode. This gives perfect model following as can be seen from the upper graph. Note that in this example a first order low-pass filter was used to obtain the sliding motion since $P(s)B(s)/A(s)$ has a zero relative order.

In the simulation of example 3 and 4, the pole-placement algorithm was used since both examples are non-minimum phase system. The simulation result of example 3 for $M = 30$ and of example 4 for $M = 10$ are given in figure 3.12 and 3.13, respectively. Both systems are in the sliding mode and their outputs perfectly follow those of the models. Note that the models and the systems have the same numerator polynomial. In example 4 a first order low-pass filter is used for the sliding mode.

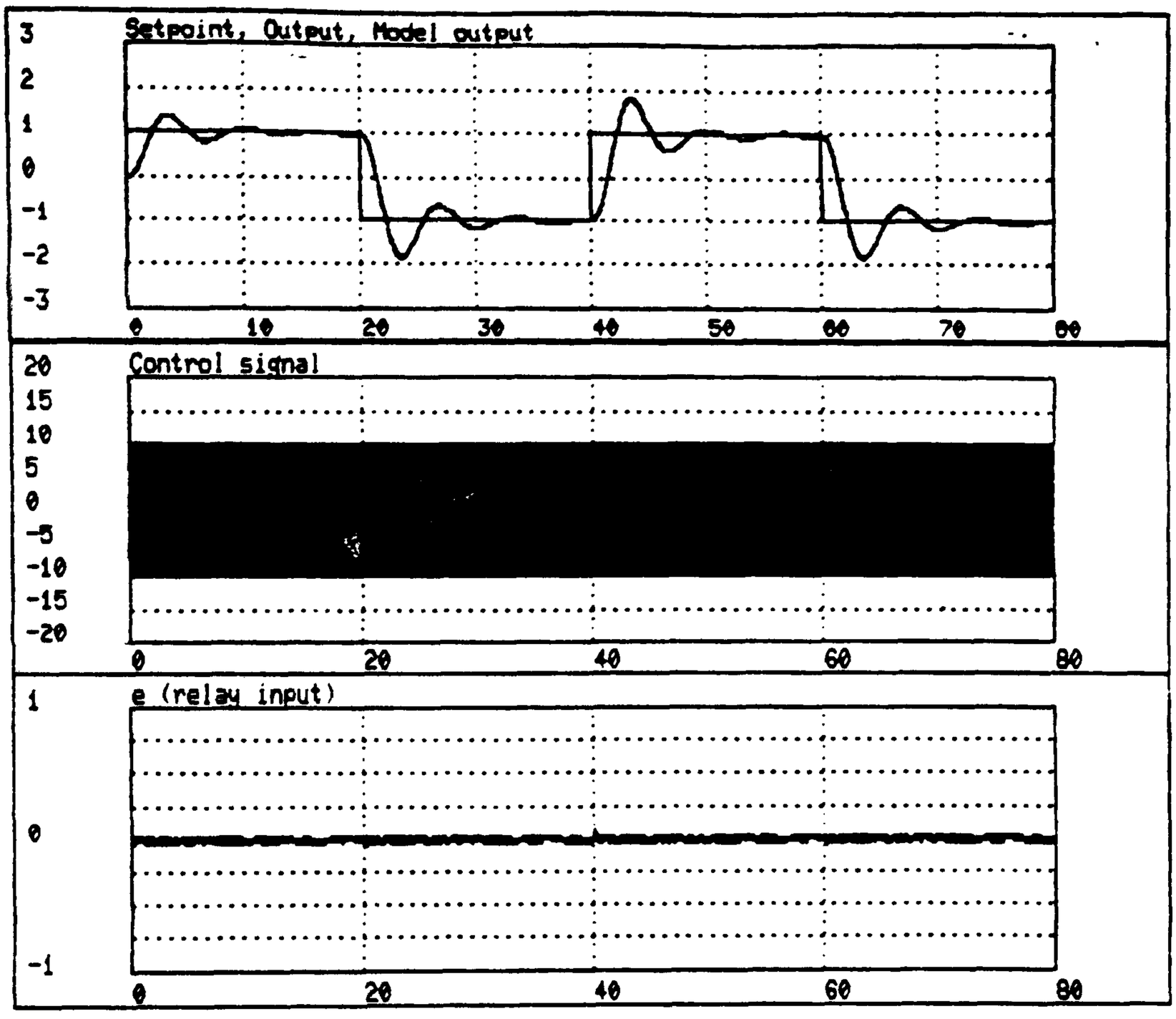


Figure 3.11 Example 2

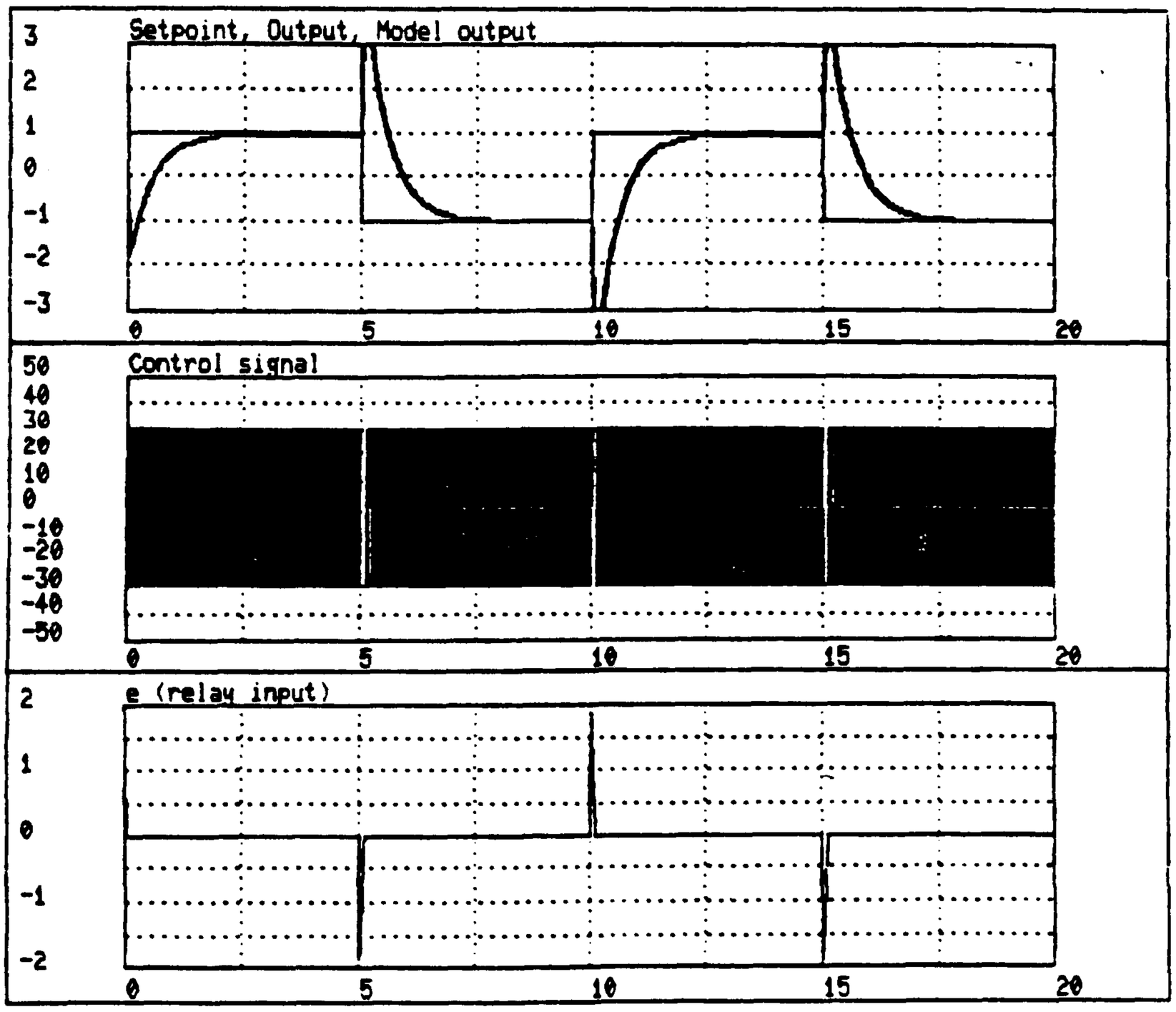


Figure 3.12 Example 3

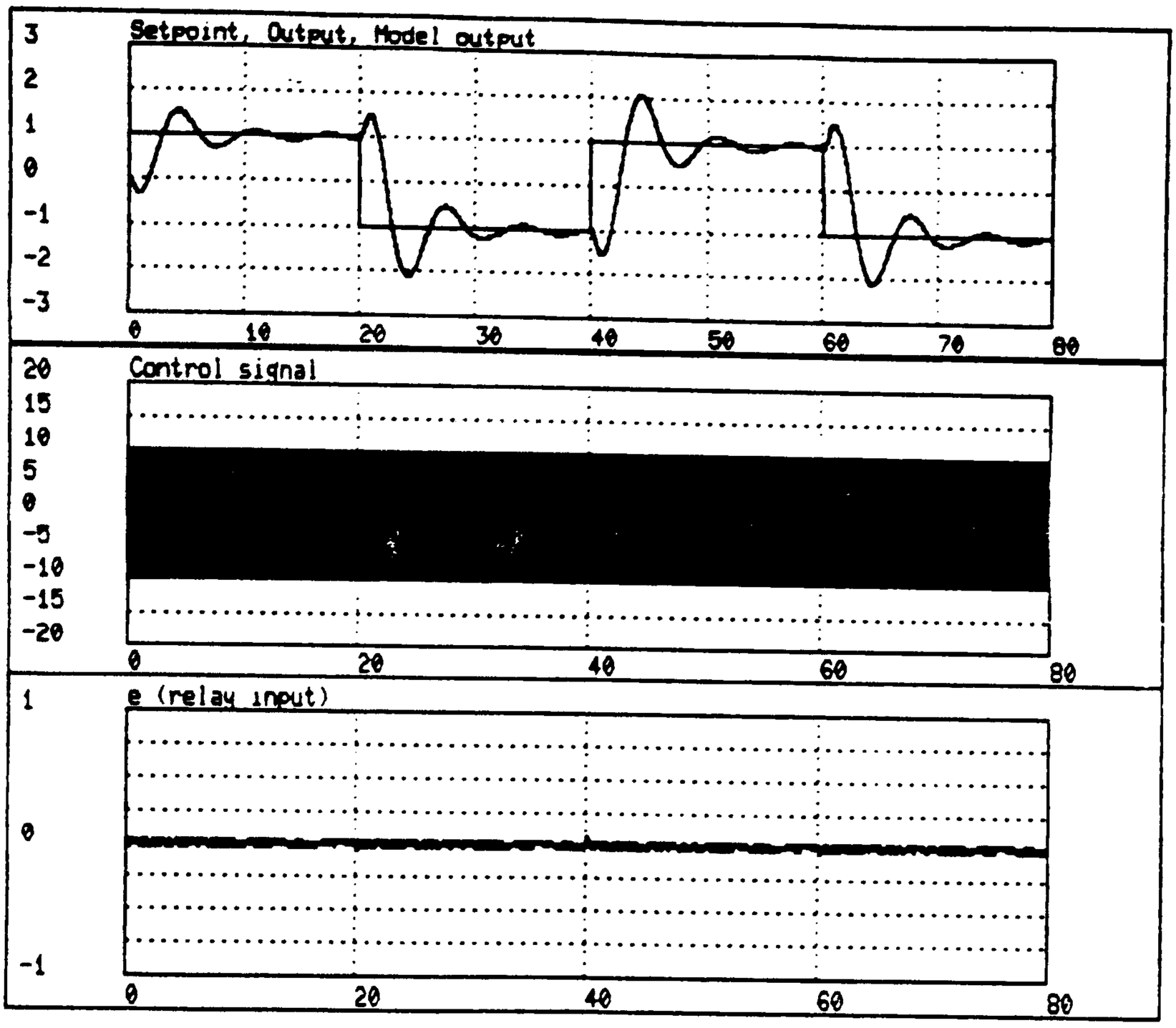


Figure 3.13 Example 4

In order to give an example for the predictive control we simulated example 5. As we mentioned in chapter 2, predictive control strategies can be combined with the model-reference or pole-placement control. Here, we used the model-reference approach. The emulator design was performed by using a second order pade approximation of the delay. But, in the simulation exact time delay was used. Simulation result is given in figure 3.14. As can be seen from the figure, after the first switching time system is in the sliding mode and the system output follows the model output closely. Much closer model-following can be obtained by increasing the relay amplitude.

The reason why we can not see the sharp peaks in the relay input of example 2 and example 4 when the setpoint changes its level is the low pass filter before the relay.

It is possible to obtain the sliding mode for a very large range of M but when M is increased we need to decrease the sample interval to obtain quasi continuous-time behaviour.

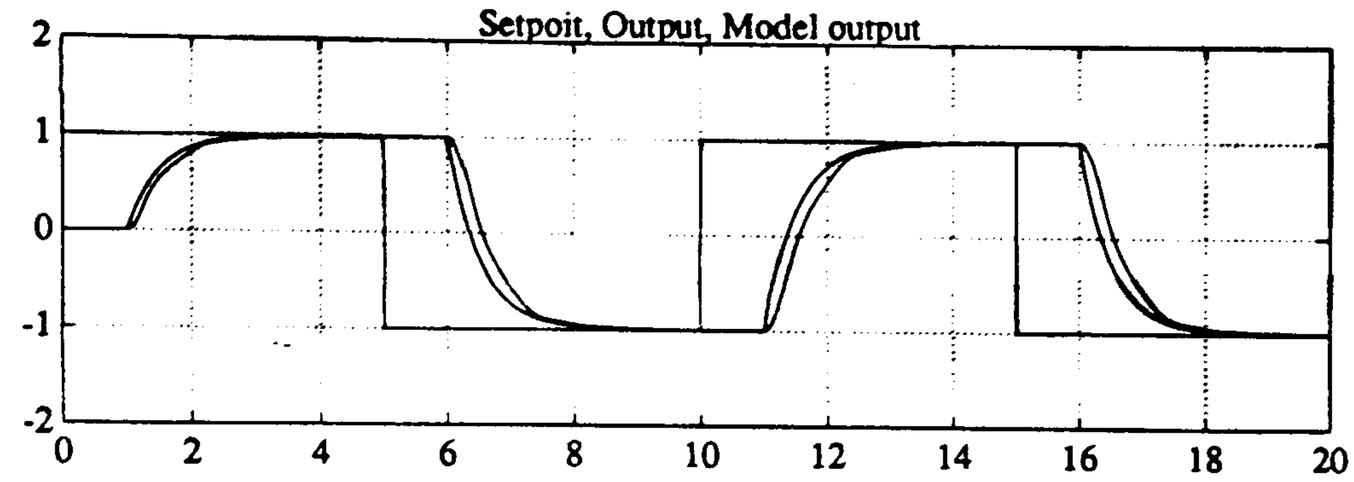


Figure 3.14 Example 5

3.4.2. Adaptive Simulations

Self-tuning version of the above emulator-based relay controller can easily be obtained by combining a recursive parameter estimator to the emulator. Here, the recursive least squares given in chapter 2 is used. As described in chapter 2, there are two approaches to the self-tuning control: direct and indirect. In this section, we will consider both approaches in order to see their relative performance.

Examples 2 and 4, the former for model-reference and the latter for pole-placement control, were simulated. The simulation results of example 2 for indirect and direct self-tuning are given in figure 3.15 and 3.16, respectively. The simulation results of example 4 are given in figure 3.17 for indirect and in figure 3.18 for direct self-tuning.

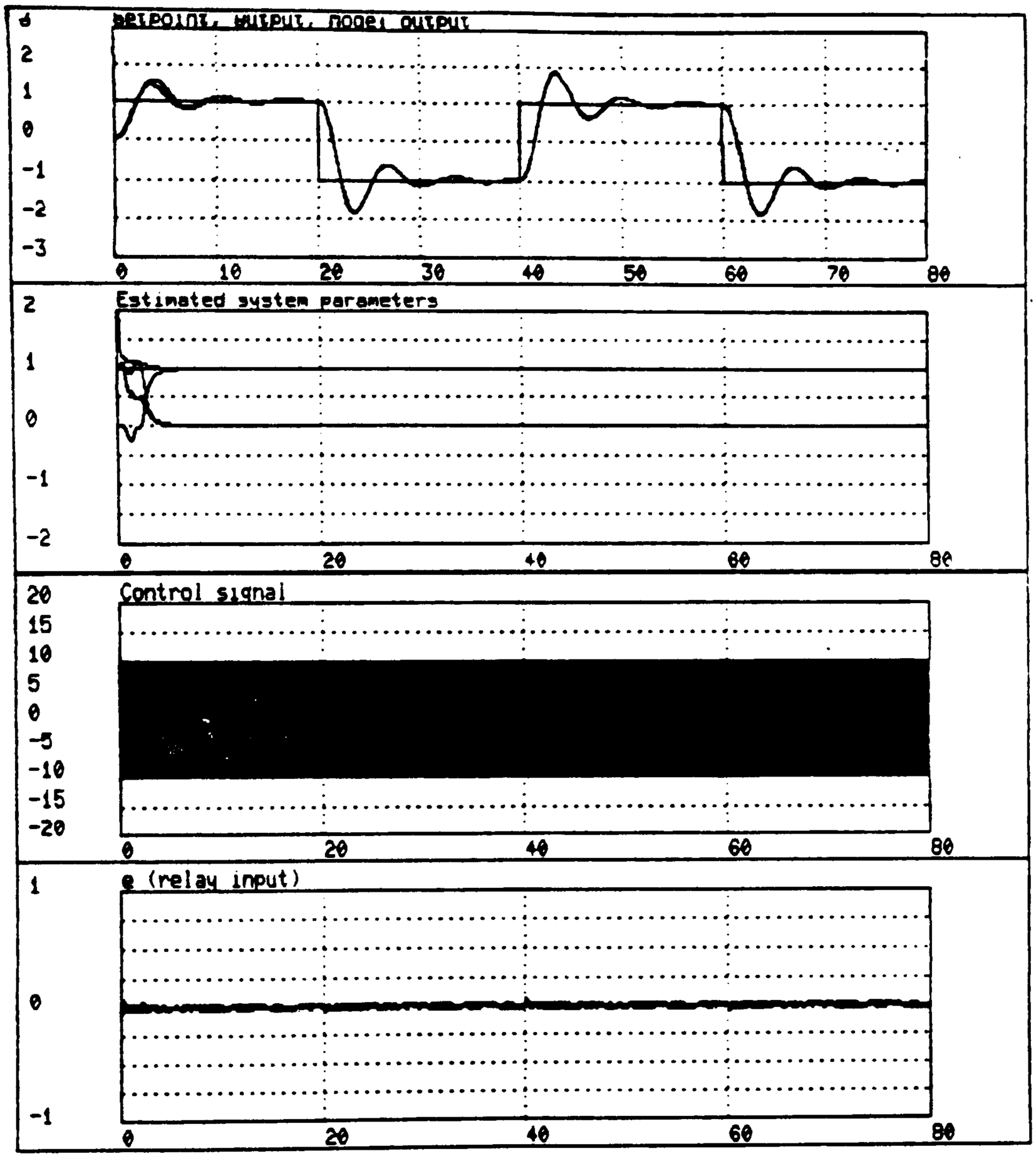


Figure 3.15 Example 2 (indirect self-tuning)

As can be seen from the figures, the parameter estimates rapidly converge to their true values. The systems are always in the sliding mode giving the same control law of the self-tuning EBC. If these examples are simulated by using the self-tuning EBC algorithms of chapter 2 with $Q(s) = 0$, exactly the same results will be obtained. As can be seen from the upper graphs, after transients caused by parameter variations at the beginning, the system outputs perfectly follow those of models.

In the direct pole-placement case (figure 3.18), we need to identify system numerator $B(s)$ in order to identify the emulator parameters. The large variations in the emulator parameters at the beginning, in comparison to others, is due to this two step identification.

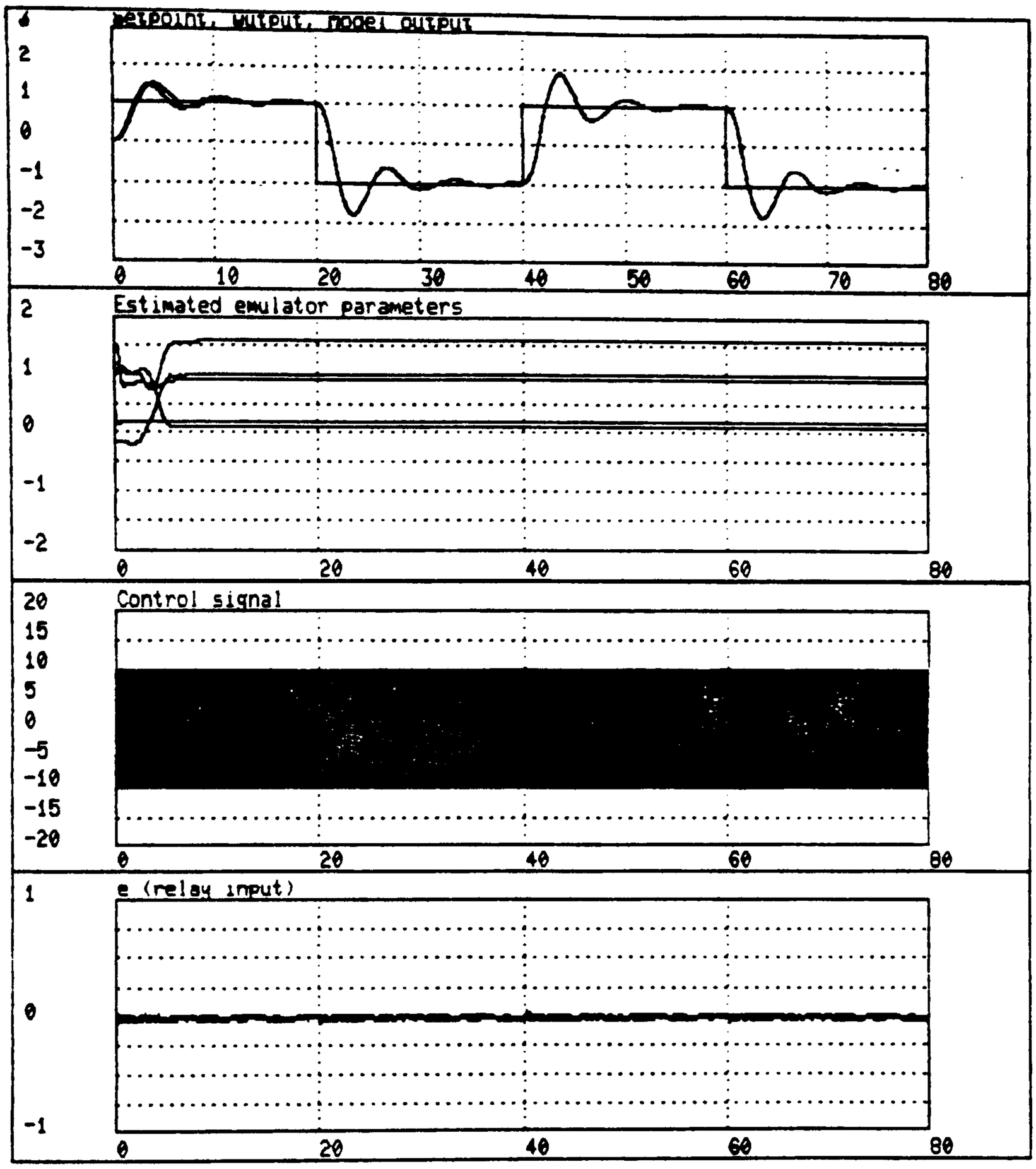


Figure 3.16 Example 2 (direct self-tuning)

Robustness properties of the emulator-based controller have been studied by Gawthrop (Gawthrop, 1987, Gawthrop, 1987a). His conclusion is the importance of the control weighting for the robustness, that is the need to use the detuned versions of the algorithms for the robust control. Detuned version of the emulator-based relay controller was derived in section 3.3.1 and it was shown that both emulator-based and emulator-based relay controller are equivalent if the relay operates in the sliding mode. Clearly, this suggests similar robustness properties for both. Our aim here to compare the robustness properties of two methods by simulation. For this purpose example 6† was simulated by using the direct self-tuning algorithm for both with and without

† This example was first used by Rohrs et al (Rohrs, 1985) to illustrate the poor robustness of a class of model-reference control algorithms. It was also used by Gawthrop in his work.

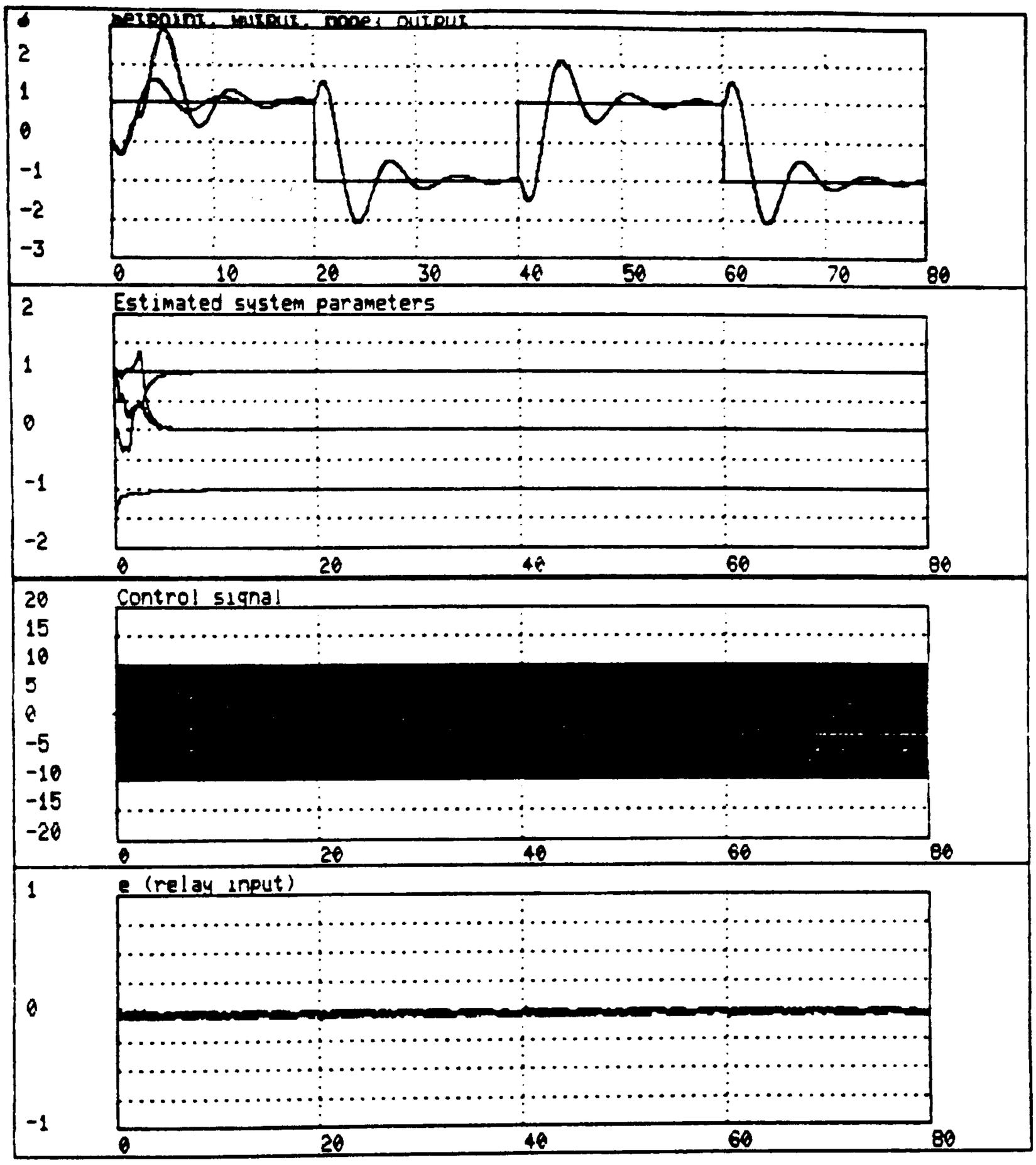


Figure 3.17 Example 4 (indirect self-tuning)

relay, and the simulation results are given in figure 3.19 and 3.20, respectively. Figure 3.20 shows that self-tuning EBC is robust for this chosen $Q(s)$ despite the second-order neglected dynamics in the example system. In figure 3.19, as can be seen from the control signal and relay input, the relay is operating in the sliding mode. This gives the same control law and thus the same output (see upper graphs in figure 3.19 and 3.20). In both figures emulator parameters also show the same convergence performance. As a result figure 3.19 and 3.20 suggest that these two self-tuning algorithms are equivalent and thus have the same robustness properties if the relay operates in the sliding mode.

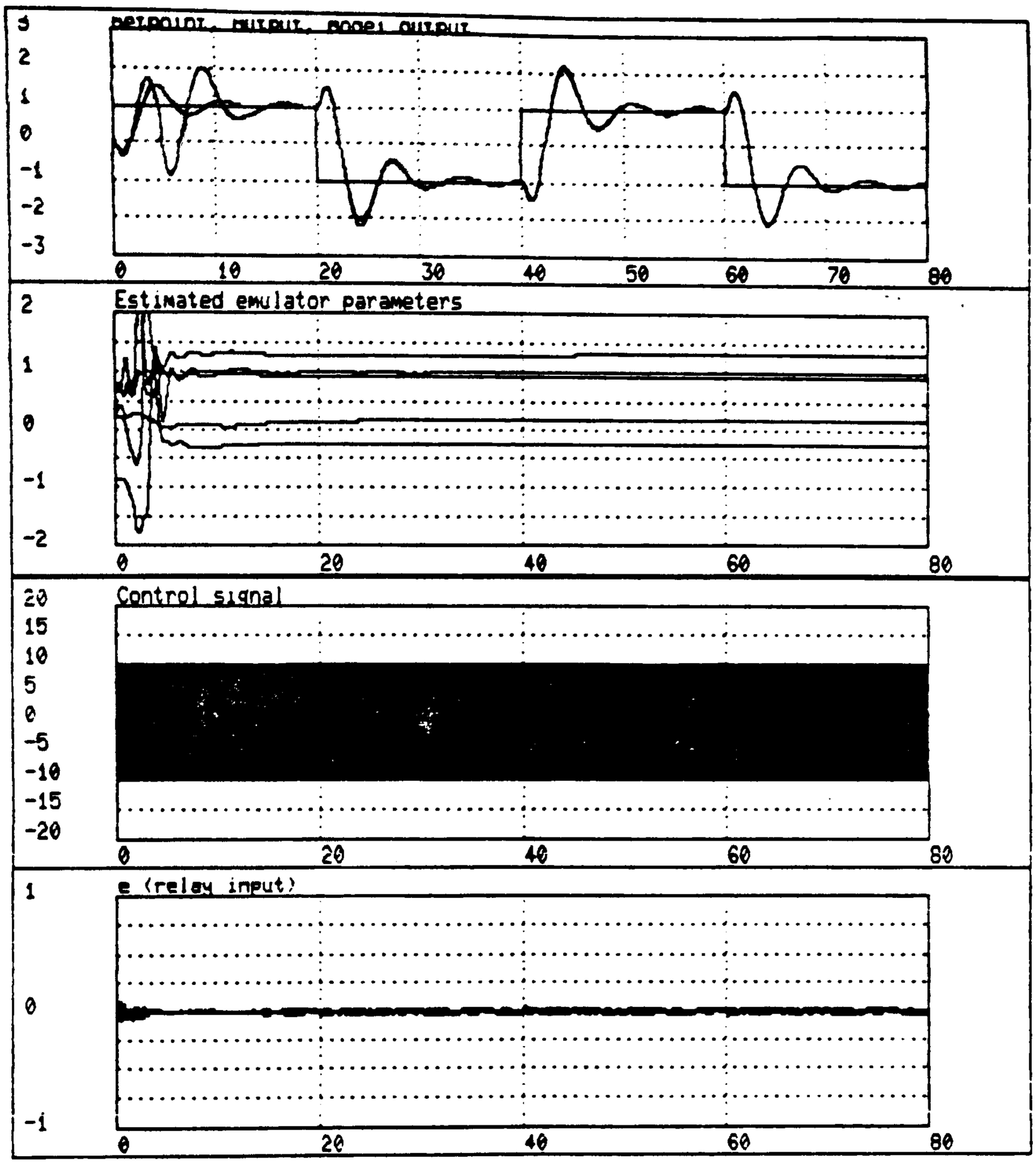


Figure 3.18 Example 4 (direct self-tuning)

Note that in all of the simulations systems are in the sliding mode at the beginning, despite parameter variations. This is due to the fact that the closed-loop system is less sensitive to parameter variations in the sliding mode.

3.5. EXPERIMENT ON A REAL SYSTEM

To see the performance of the relay self-tuning control on a real system, two cascaded tanks as shown in figure 3.21 were used in a level control experiment. This system can be approximated as a linear second order system although it has nonlinearities.

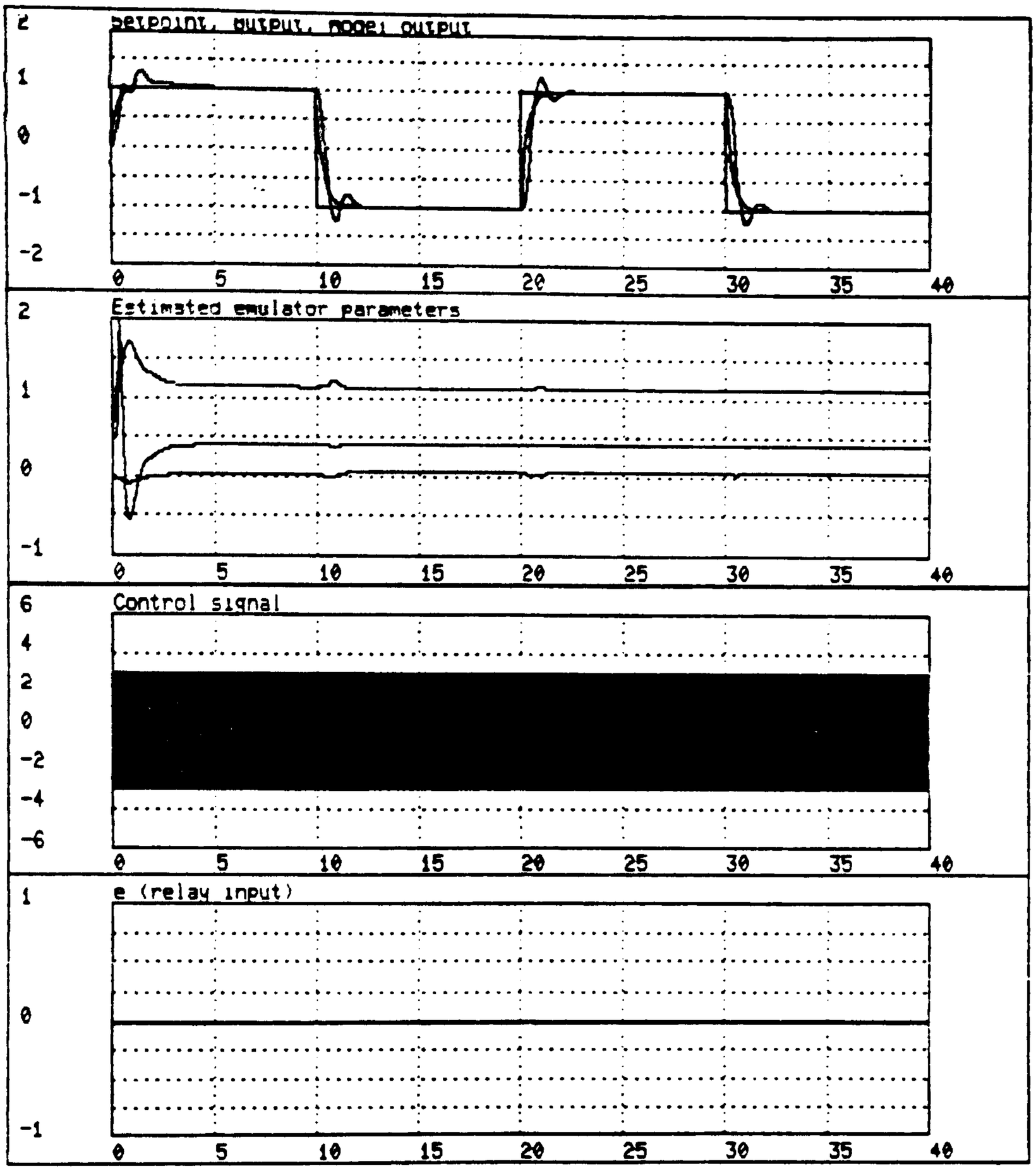


Figure 3.19 Example 6 with relay

To imitate a symmetric relay, two pumps were used: one for pumping water into the first tank and the other for pumping water out of the first tank. The level of the second tank was controlled by using the indirect relay self-tuning control algorithm. The algorithm was the same as the one used for the simulations. The interface between tank to SUN 3 workstation was made through a MOTOROLA MVME315 microsystem which had a A/D and D/A converter. The run time for experiments was around 40 minutes.

The results of the experiments showed that the relay self-tuning performed well. One of the results is given in figure 3.22. Since the system is second order four parameters were estimated. Note that parameter estimates rapidly converge to their correct values. Also notice that after

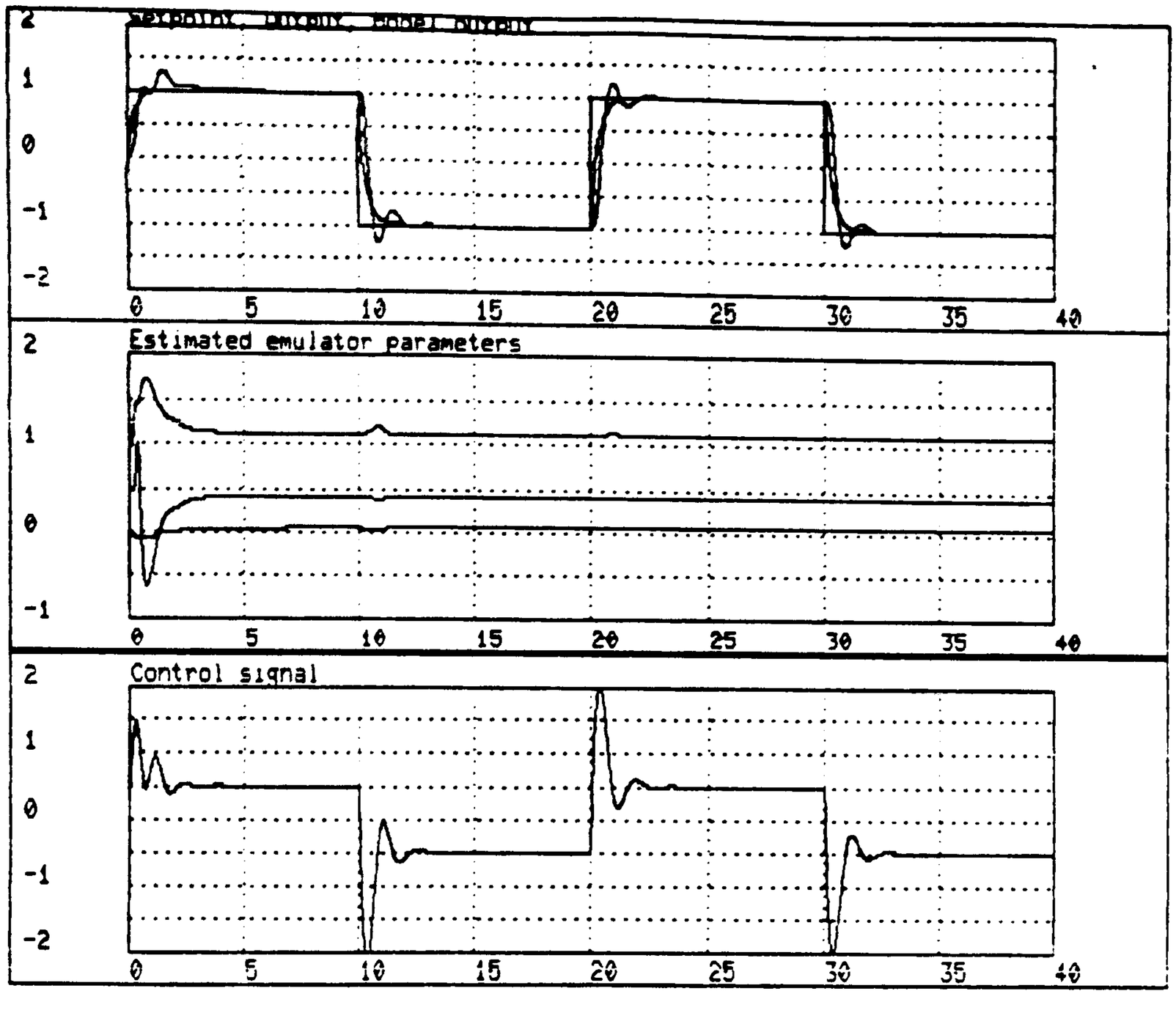


Figure 3.20 Example 6 without relay

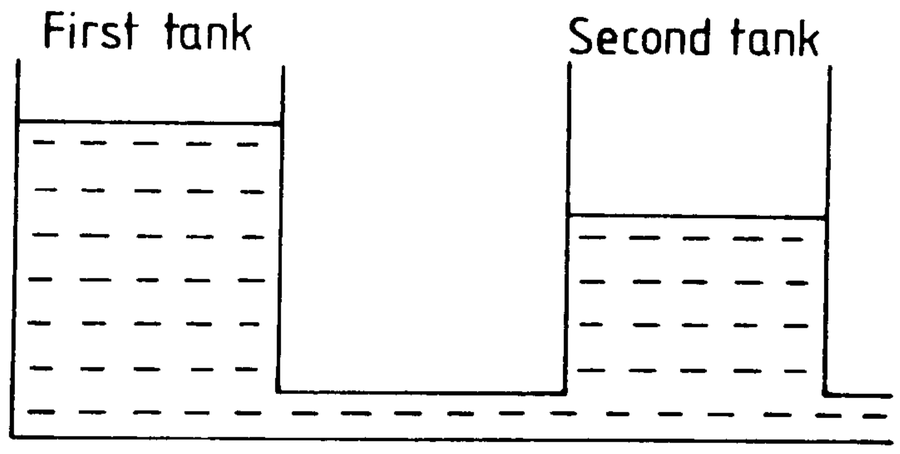


Figure 3.21 Two cascaded tanks

a short time, the relay starts to operate in the sliding mode and the system output perfectly follows the model output when the correct parameters are reached.

It was observed that the relay self-tuning controller performed better than the corresponding self-tuning controller without a relay. The reason for this is that the pump is strongly non-linear. This has no effect on the relay control as only two points on the non-linear characteristic are used; but the usual self-tuning controller has a strongly non-linear system to identify.

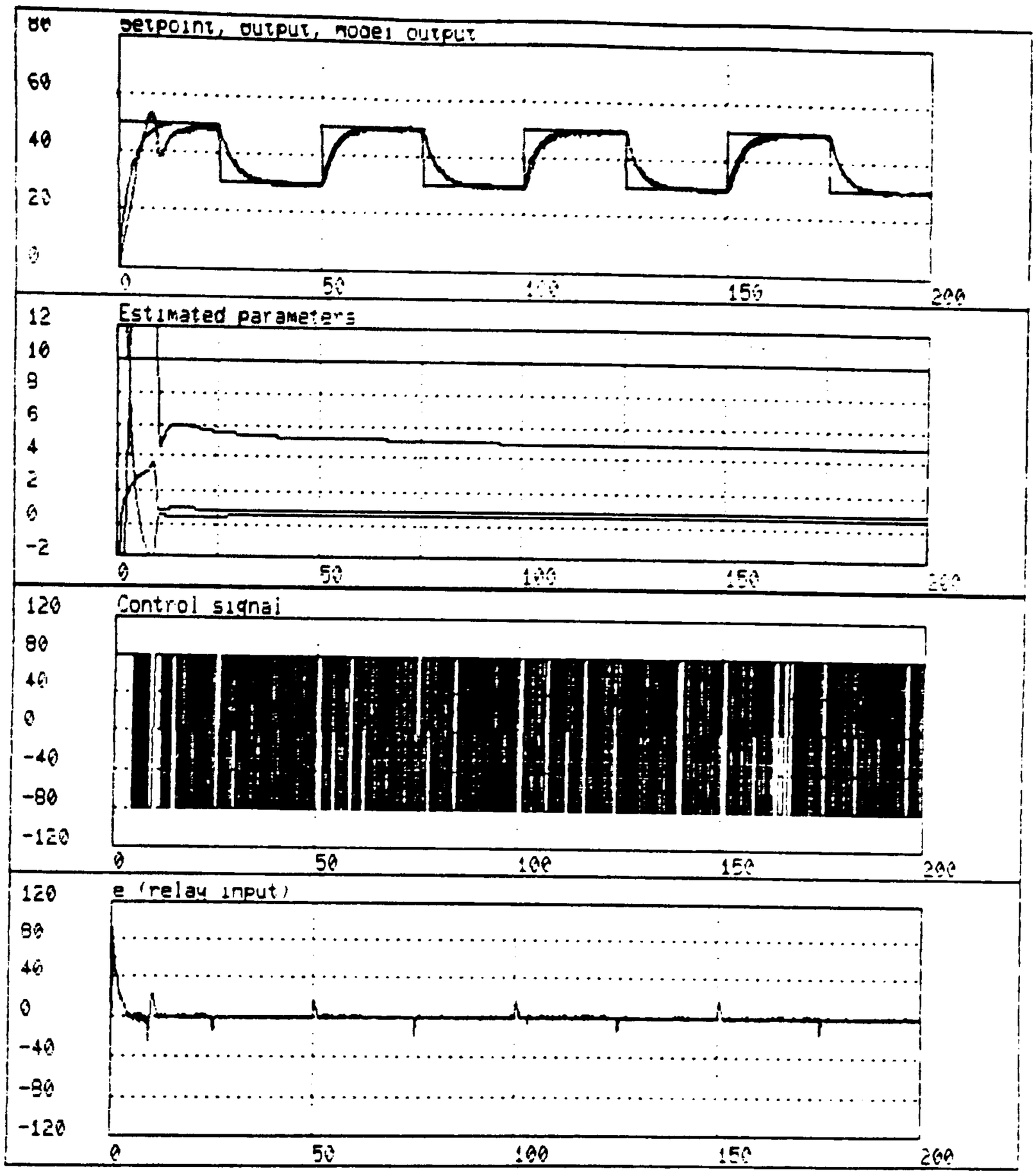


Figure 3.22 Level control of two cascaded tanks

3.6. CONCLUSIONS

Relay control of single-input single-output systems with unknown parameters has been considered. The method developed here, *emulator-based relay control,* removes the need to know the system states to implement the switching surface, unlike the variable structure design. The switching surface is implemented by replacing unrealisable derivatives by an emulator. The need to know system parameters for the emulator design is removed by using self-tuning emulators.

It has been shown that emulator-based control and its proposed relay version are equivalent if the relay operates in the sliding mode. Thus the same control approaches, such as model-

reference pole-placement and predictive control, obtained in the first case can also be obtained in the second case. The results are illustrated by a number of simulated examples. Algorithms are also tested on a real experiment using a laboratory level control system.

CHAPTER 4

CONTINUOUS-TIME GENERALIZED PREDICTIVE CONTROL

4.1. INTRODUCTION

The shortcomings of the early self-tuners have lead researchers to look for better algorithms for the self-tuning control. This research effort gave rise to a new group of algorithms called Long Range Predictive Control (LRPC) (Richalet, 1978, Cutler, 1980, Keyser, 1981, Peterka, 1984, Mosca, 1984, Keyser, 1985, Clarke, 1987, Clarke, 1987, Lelic, 1987, Keyser, 1988). These algorithms differ from each other in the assumed system model which the design is based on and in formulation, but all are based on the same basic ideas which can be summarized as follows:

1. Predict the system output over a range of future times.
2. Assuming that future setpoint is known, choose a set of future controls which minimize the future errors between the predicted future output and the future setpoint.
3. Use the first element u(t) as a current input and repeat the whole procedure at the next time instant; that is a receding horizon strategy is used.

These algorithms, as a natural result of different formulation and choice of different system model for the design, have different properties, but all have an important common property of being robust against time delay variations due to the long range prediction of the output.

One of the most recent of these algorithms is the Generalized Predictive Control (GPC) developed by Clarke and his co-workers (Clarke, 1987). Robustness of the algorithm against time delay and parameter variations, the choice of system order and its ability to control difficult systems have been illustrated by simulations (Clarke, 1984, Clarke, 1985, Clarke, 1987, Mohtadi,

1987). In addition some successful applications of the algorithm have been reported (Lambert, 1987, Lambert, 1987). As a result, GPC generally seems to be the best among the other self-tuning control and LRPC methods.

As with most of the other self-tuning control methods, GPC was also developed in discrete-time. It was noted in chapter 2 that, in general, the continuous-time approach has advantages over the discrete-time approach. Hence, we believe that it would be desirable to have a continuous-time version. Therefore, this chapter is devoted to the development and analysis of a continuous-time generalized predictive control (CGPC) (Gawthrop and Demircioglu 1988, 1989).

This chapter is organized as follows. In section 2 the basic CGPC algorithm is introduced and the relation of the resulting control law with the state feedback is discussed. Section 3 examines the CGPC closed-loop system in detail. The choices and the effects of the CGPC parameters to the closed-loop system response are discussed in section 4. In section 5 the relation of CGPC with LQ control is examined. In section 6 some extensions are introduced to the basic algorithm. A relay version of the CGPC is described in section 7 and an illustrative simulation study is given in section 8. Finally, the chapter is finished by some conclusions in section 9.

4.2. THE BASIC CGPC ALGORITHM

4.2.1. System Description

The system model to be employed in the development of the algorithm will be the same as in the previous chapters except here the system will be assumed to be strictly proper and the time delay term e^{-sT} will not be considered explicitly.

$$Y(s) = \frac{B(s)}{A(s)} U(s) + \frac{C(s)}{A(s)} V(s) \tag{4.1}$$

However, it will be assumed that the polynomials $B(s)$ and $A(s)$ include a rational approximation of the time delay term e^{-sT} when it exists. Thus the time delay systems will be approximately modeled with a higher order system without a delay (Marshall, 1979, Gawthrop, 1987, Souza, 1988, Besharati-Rad, 1988). Notice, this approach resembles the one in discrete-time where the

order of numerator polynomial B is increased to accommodate the time delay.

4.2.2. Output Prediction

The development of the CGPC algorithm involves two main steps as in the other predictive methods:

1. output prediction,
2. control law calculation based on this prediction.

In this section we will consider the first of these steps. Before introducing the details, let us first point out some facts.

1. In discrete-time, predictor design is based on the fact that the discrete transform variable z corresponds to a forward time shift (Astrom, 1970). This enables us to distinguish between the future and the past. Therefore, removing the unknown future noise terms, a *j–step ahead* prediction of the output at time t can be easily obtained.
2. In discrete-time, there are a finite number of points in a given range of prediction horizon which means that a finite set of output predictors are needed.

Obviously, these facts do not hold in continuous-time; the Laplace variable s corresponds to derivative operation, not to a forward shift, and there are an infinite number of points in a given range of prediction horizon indicating that the continuous-time predictor should depend on a continuous future variable, say T. Therefore, the predictor design seems less obvious in continuous-time. However, one may note that the current derivatives of a smooth continuous-time signal imply the future development of that signal. Therefore, if the output derivatives at time t are known, it is possible to predict the future system output. This *T–ahead* predictor is given by a truncated Maclaurin series as follows.†

† A function f(t) can be approximated about a specified point t_0 by its derivatives at that point. This is known as the Taylor series expansion of that function about that point. If t_0=0 the series is called Maclaurin series. In our case, the derivatives of the output y(t) (at time t) are known. So we are defining a new time axis T (taking the time t as the origin) and doing the expansion in this new variable T.

$$\hat{y}(t+T) = y(t) + \sum_{k=1}^{N_y} y_k(t)\frac{T^k}{k!} \tag{4.2}$$

where

$$y_k(t) = \frac{d^k \hat{y}(t+T)}{dT^k} \ (at\ T=0) = \frac{d^k y(t)}{dt^k} \tag{4.3}$$

N_y = predictor order

Obviously, for a given T, prediction accuracy depends on N_y. We will leave the discussion of the choice of N_y to section 4.4.3. For the time being, it is sufficient to say that the larger T the larger N_y should be for a good prediction.

As stated above, for the predicted value of the future output at time $t+T$, the derivatives of the output at time t are needed. However, as we discused in chapter 2 taking derivatives of the output is not desirable, because of noise amplification, so the derivative operation is emulated Recall that if $y_k^*(t)$ is the emulated value of $y_k(t)$, then it is given in the Laplace domain by the following equation.

$$Y_k^*(s) = \frac{E_k(s)B(s)}{C(s)} U(s) + \frac{F_k(s)}{C(s)} Y(s) \tag{4.4}$$

where $E_k(s)$ and $F_k(s)$ polynomials are found from the following identity:

$$\frac{s^k C(s)}{A(s)} = E_k(s) + \frac{F_k(s)}{A(s)} \tag{4.5}$$

The term in (4.4), $\frac{E_k(s)B(s)}{C(s)}$ is not a proper transfer function for $k > \rho$ where ρ is the relative order of the system. This term can be decomposed into two parts by using polynomial long division

$$\frac{E_k(s)B(s)}{C(s)} = H_k(s) + \frac{G_k(s)}{C(s)} \tag{4.6}$$

where $G_k(s)$ and $H_k(s)$ are polynomials in s, $\frac{G_k(s)}{C(s)}$ is strictly proper and $H_k(s)$ is the remainder. Then the emulator equation (4.4) becomes,

$$Y_k^*(s) = H_k(s)U(s) + \frac{G_k(s)}{C(s)} U(s) + \frac{F_k(s)}{C(s)} Y(s) \tag{4.7}$$

Assuming that the degree of $C(s)$ is one less than that of $A(s)$, the degree of the polynomials involved in eqn. (4.7) are:

$$deg\,(H_k(s)) = k - \rho$$

$$deg\,(G_k(s)) = n - 2$$

$$deg\,(F_k(s)) = n - 1$$

$$n = n_a = deg\,(A(s))$$

If $deg\,(C(s)) = deg\,(A(s))$ then the only difference will be $deg\,(G_k(s)) = n - 1$. Notice that the emulator equation has two parts. One part can be realized by using proper transfer functions the other part can not; it can be rewritten as

$$Y_k^*(s) = H_k(s)U(s) + Y_k^o(s) \tag{4.8}$$

where

$$Y_k^o(s) = \frac{G_k(s)}{C(s)} U(s) + \frac{F_k(s)}{C(s)} Y(s) \tag{4.9}$$

is the realisable part. It is interesting to note that, the equations leading to (4.8) are *algebraicly* equivalent to those leading to a discrete-time k-step ahead predictor; but the interpretation is different. In the time domain equation (4.8) becomes

$$y_k^*(t) = \underline{h_k}\,\underline{u} + y_k^o(t) \tag{4.10}$$

where $\underline{h_k}$ is a row vector and contains the coefficients of the $H_k(s)$ polynomial and $\underline{u}$ is a column vector which contains the input derivatives.

$$\underline{u} = [u(t)\ u_1(t)\ .....\ u_{k-\rho}(t)]^T \tag{4.11}$$

$$u_k(t) = \frac{d^k u(t)}{dt^k} \tag{4.12}$$

If the output derivatives in eqn. (4.2) are replaced by their emulated values then the *T–ahead* predictor is given by

$$y^*(t+T) = y(t) + \sum_{k=1}^{N_y} y_k^*(t) \frac{T^k}{k!} \tag{4.13}$$

This equation can be rearranged in a matrix form

$$y^*(t+T) = \underline{T}_{N_y} \underline{Y}^* \tag{4.14}$$

where

$$\underline{T}_{N_y} = [\, 1 \;\; T \;\; \frac{T^2}{2!} \;\; ..... \;\; \frac{T^{N_y}}{N_y!} \,] \tag{4.15}$$

$$\underline{Y}^* = [\, y(t) \;\; y_1^*(t) \;\; ..... \;\; y_{N_y}{}^*(t) \,]^T \tag{4.16}$$

Using eqn. (4.10), the vector $\underline{Y}^*$ can be written in an explicit form

$$\underline{Y}^* = H\underline{u} + \underline{Y}^o \tag{4.17}$$

where $\underline{Y}^o$ is a $(N_y+1)\times 1$ column vector

$$\underline{Y}^o = [\, y(t) \;\; y_1^o(t) \;\; ..... \;\; y_{N_y}{}^o(t) \,]^T \tag{4.18}$$

and H is a $(N_y+1)\times(N_y - \rho+1)$ lower triangular Toeplitz matrix which contains the coefficients of $H_k(s)$ polynomials. When $\rho = 1$, the H matrix is given as follows

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & . & . & 0 \\ h_1 & 0 & 0 & 0 & . & . & 0 \\ h_2 & h_1 & 0 & 0 & . & . & 0 \\ h_3 & h_2 & h_1 & 0 & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ h_{N_y} & . & . & . & . & . & h_1 \end{bmatrix} \tag{4.19}$$

where the h_ks are the Markov parameters of the open-loop system $\frac{B(s)}{A(s)}$. This can easily be proven by using the identities (4.5) and (4.6). If $\rho > 1$ then $h_k = 0$ for $k < \rho$ and as a result the number of the H matrix columns decreases with ρ. Note that the dimension of vector $\underline{u}$ is now $(N_y - \rho+1)\times 1$.

Substitution of eqn. (4.17) into eqn. (4.14) results in the following predictor equation.

$$y^*(t+T) = \underline{T}_{N_y} H \underline{u} + \underline{T}_{N_y} \underline{Y}^o \tag{4.20}$$

The eqn. (4.20) is the basic equation in the development of CGPC algorithm, however the following Laplace form of it will be needed in the development of the closed-loop system.

$$Y_T^*(s) = \underline{T}_{N_y} H \underline{S}_H U(s) + \underline{T}_{N_y} \underline{Y}^o(s) \tag{4.21}$$

where

$$\underline{S}_H = [\ 1\ s\ s^2\ .....\ s^{N_y - \rho}\ ]^T \tag{4.22}$$

$$\underline{Y}^o(s) = \frac{G\ \underline{S}_G}{C(s)} U(s) + \frac{F\ \underline{S}_F}{C(s)} Y(s) \tag{4.23}$$

where G and F are $(N_y+1)\times(n-1)$, $(N_y+1)\times n$ coefficient matrices of the polynomials $G_k(s)$ and $F_k(s)$ respectively and $\underline{S}_G$, $\underline{S}_F$ are corresponding s vectors.

$$\underline{S}_G = [\ s^{n-2}\ s^{n-3}\ .....\ s\ 1\ ]^T \tag{4.24}$$

$$\underline{S}_F = [\ s^{n-1}\ s^{n-2}\ .....\ s\ 1\ ]^T \tag{4.25}$$

The subscript T is used in eqn. (4.21) to indicate that the Laplace transform is taken with respect to t and T is left as a parameter. Note that $\underline{T}_{N_y} H \underline{S}_H$, $\underline{T}_{N_y} G \underline{S}_G$ and $\underline{T}_{N_y} F \underline{S}_F$ are polynomials with coefficients dependent on T

$$H_T(s) = \underline{T}_{N_y} H \underline{S}_H \quad , \quad G_T(s) = \underline{T}_{N_y} G \underline{S}_G \quad , \quad F_T(s) = \underline{T}_{N_y} F \underline{S}_F \tag{4.26}$$

with these polynomials, the predictor (eqn. 4.21) can be written in a transfer function form.

$$Y_T^*(s) = H_T(s)U(s) + \frac{G_T(s)}{C(s)} U(s) + \frac{F_T(s)}{C(s)} Y(s) \tag{4.27}$$

The polynomials $H_T(s)$, $G_T(s)$ and $F_T(s)$ satisfy the following identities

$$\frac{T_T(s)C(s)}{A(s)} = E_T(s) + \frac{F_T(s)}{A(s)} \tag{4.28}$$

$$\frac{E_T(s)B(s)}{C(s)} = H_T(s) + \frac{G_T(s)}{C(s)} \tag{4.29}$$

where

$$e^{sT} \simeq T_T(s) = 1 + sT + \ ..... + s^{N_y} \frac{T^{N_y}}{N_y!} \tag{4.30}$$

4.2.2.1. Recursion of the identities

Computations involved in the output prediction can be reduced significantly by using recursive equations for $E_k(s)$, $F_k(s)$, $H_k(s)$ and $G_k(s)$ polynomials. Recursions for the $E_k(s)$ and $F_k(s)$ polynomials (markov recursion) are given in chapter 2. Here, a recursive formulation for the $H_k(s)$ and $G_k(s)$ polynomials will be developed. Consider the second identity (eqn. 4.6)

$$\frac{E_{k+1}(s)B(s)}{C(s)} = H_{k+1}(s) + \frac{G_{k+1}(s)}{C(s)} \tag{4.31}$$

if the recursive equation for $E_{k+1}(s)$ (eqn. 2.14) is substituted in this identity and the resulting equation is pursued further, the following recursive equations are obtained.

$$H_{k+1}(s) = sH_k(s) + e_{k+1}B_e(s) + h_{k+1} \tag{4.32}$$

$$G_{k+1}(s) = sG_k(s) + e_{k+1}B_f(s) - h_{k+1}C(s) \tag{4.33}$$

where $B_e(s)$ and $B_f(s)$ satisfies the following identity

$$\frac{B(s)}{C(s)} = B_e(s) + \frac{B_f(s)}{C(s)} \tag{4.34}$$

and

$$h_{k+1} = \frac{g_{k0}}{c_0} \quad , \quad e_{k+1} = \frac{f_{k0}}{a_0} \tag{4.35}$$

where g_{k0}, f_{k0}, c_0 and a_0 are the coefficients of the highest power s term of $G_k(s)$, $F_k(s)$ $C(s)$ and $A(s)$ polynomials respectively. The initial polynomials for these recursions are given by

$$\frac{E_0(s)B(s)}{C(s)} = H_0(s) + \frac{G_0(s)}{C(s)} \tag{4.36}$$

Note, if $deg(Ek(s)B(s)) < deg(C(s))$, then $H_k(s) = 0$ and $G_k(s) = E_k(s)B(s)$

4.2.2.2. Control order

In the discrete-time GPC after a future time instant, which is called control horizon (Nu), the predicted control† increments are constrained to be zero (Clarke et al 1987). This constraint is

† The term predicted control is used for the future control which is to be calculated at time t from the predictor model in order to minimize specified cost function. The calculations are based on the receding horizon strategy that is predicted control $u_r^*(t,T)$ is not applied to the system over the time which cost

convenient for the following reasons:

1- It reduces the dimension of the matrix involved in the control law calculations and thus the computational burden.

2- It enables the control of non-minimum phase systems with zero control weighting.

3- It can be used to adjust the system transients.

As discussed by Clarke et al (1987), the effect of Nu on the output response is that the larger Nu, the more active control action, thus the faster response or vice versa. As a result, Nu is a useful design parameter.

In discrete-time, the predictor equation explicitly includes the future controls whereas in continuous-time (eqn. 4.20) the future control is implicitly included in terms of current input derivatives and the future variable T. More precisely, the future control (predicted control) appears to be a *polynomial* in T. Therefore the above discrete-time constraint is not as appropriate for the continuous-time case. Instead, we use the constraint that input derivatives of order greater than N_u are zero that is

$$u_k(t) = 0 \qquad \text{for} \qquad k > N_u \tag{4.37}$$

We will call this value of N_u as control order because of the obvious reason that the predicted control is constrained to be a polynomial of order N_u. For example, the predicted control will be a constant for N_u=0, a ramp for N_u=1 and so on.

The control order (N_u) is algebraically equivalent to the control horizon (Nu) because it reduces the dimension of the vector $\underline{u}$ to $(N_u+1)\times 1$ and the dimension of the matrix H to $(N_y+1)\times(N_u+1)$. However, they are not physically equivalent. Despite this, the constraint (4.37) has similar effects in continuous-time (control of non-minimum phase systems with zero control weighting, the larger N_u the faster response etc.). This will be discused in detail later in this chapter.

---

function is minimized just only its value at T=0 $u_r^*(t,0)$ is applied. Therefore predicted control and the real future control are not the same. This matter will be clearer in section 4.2.4.

4.2.2.3. Interpretation of the predictor

At any time t, the future response of any linear system can be divided into three parts.

$$y(t+T) = y_u(t,T) + y_i(t,T) + v(t+T) \tag{4.38}$$

where:

- $y_u(t,T)$ is the response to the input after time t assuming zero initial conditions at time t
- $y_i(t,T)$ is the response to initial conditions at time t created by the past data, assuming zero input after time t
- $v(t+T)$ is the future noise component.

At time t, $y_i(t,T)$ is exactly known, $y_u(t,T)$ depends on the future input and $v(t+T)$ is not known. Thus assuming that future noise is zero a predictor model for the system depending on the future control is obtain as

$$\tilde{y}(t+T) = h(T)*u(t+T) + y_i(t,T) \tag{4.39}$$

where $h(T)$ is the impulse response of the system and * denotes the convolution integral with respect to T.

Now, note that the predictor of eqn. (4.20) is exactly in the same form. The term $\underline{T}_{N_y}\underline{Y}^o$ entirely depends on the past data and can be calculated at time t; thus this part is related to the initial condition response. The part $\underline{T}_{N_y}H\underline{u}$ depends on the future input in terms of input derivatives at time t and the future variable T. Thus it is an approximation to $h(T)*u(t+T)$. To make this point more clear consider the following approximate functions

$$h(T) \simeq \tilde{h}(T) = h_1 + h_2T + h_3\frac{T^2}{2!} + \ldots\ldots + h_{N_y}\frac{T^{(N_y-1)}}{(N_y-1)!} \tag{4.40}$$

$$u(t+T) \simeq \tilde{u}(t+T) = u(t) + u_1(t)T + u_2(t)\frac{T^2}{2!} + \ldots\ldots + u_{N_u}(t)\frac{T^{N_u}}{N_u!} \tag{4.41}$$

it can be shown that

$$\tilde{h}(T)*\tilde{u}(t+T) = \underline{T}_{N_y}H\underline{u} + \underline{T}_rH_r\underline{u} \tag{4.42}$$

where

$$\underline{T}_r = [\ \frac{T^{(N_y+1)}}{(N_y+1)!} \cdot \cdot \cdot \cdot \cdot \frac{T^{(N_y+N_u)}}{(N_y+N_u)!}\ ] \tag{4.43}$$

$$H_r = \begin{bmatrix} 0 & h_{N_y} & h_{(N_y-1)} & \cdot & \cdot & h_{(N_y-N_u+1)} \\ 0 & 0 & h_{N_y} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & h_{(N_y-1)} \\ 0 & 0 & 0 & \cdot & \cdot & h_{N_y} \end{bmatrix} \tag{4.44}$$

From eqn. (4.42) it is obvious that

$$\underline{T}_{N_y} H \underline{u} \approx h(T)*u(t+T) \tag{4.45}$$

As a result, eqn. (4.20) is an approximation to eqn. (4.39) and approximation accuracy depends on N_y, N_u and T.

4.2.3. Reference Output

The objective of the CGPC control law, as in the discrete-time GPC, is to drive the predicted future output as close as possible to the future setpoint subject to the input constraints. This implies that the future setpoint needs to be known, which is the case in some applications such as robotics, however in many applications future setpoint is not known. In this case, one may consider a constant setpoint w into the future, but trying to match the predicted output to a constant value might give an excessive control action or overshoot at the output. The better approach may be to consider a reference output which goes smoothly from the current output $y(t)$ to w as illustrated in figure 4.1. As will be seen later, this approach indeed has the effect of reducing the overshoot and the control activity, in addition it enables us to obtain model-following type control (even sometimes exact model-following) with the right choice of CGPC design parameters.

The reference output $w_r(t,T)$ will be taken as the output of a rational transfer function (reference model) with numerator R_n and denominator R_d, namely

$$W_r(t,s) = \frac{R_n(s)}{R_d(s)} \frac{w(t)-y(t)}{s} \tag{4.46}$$

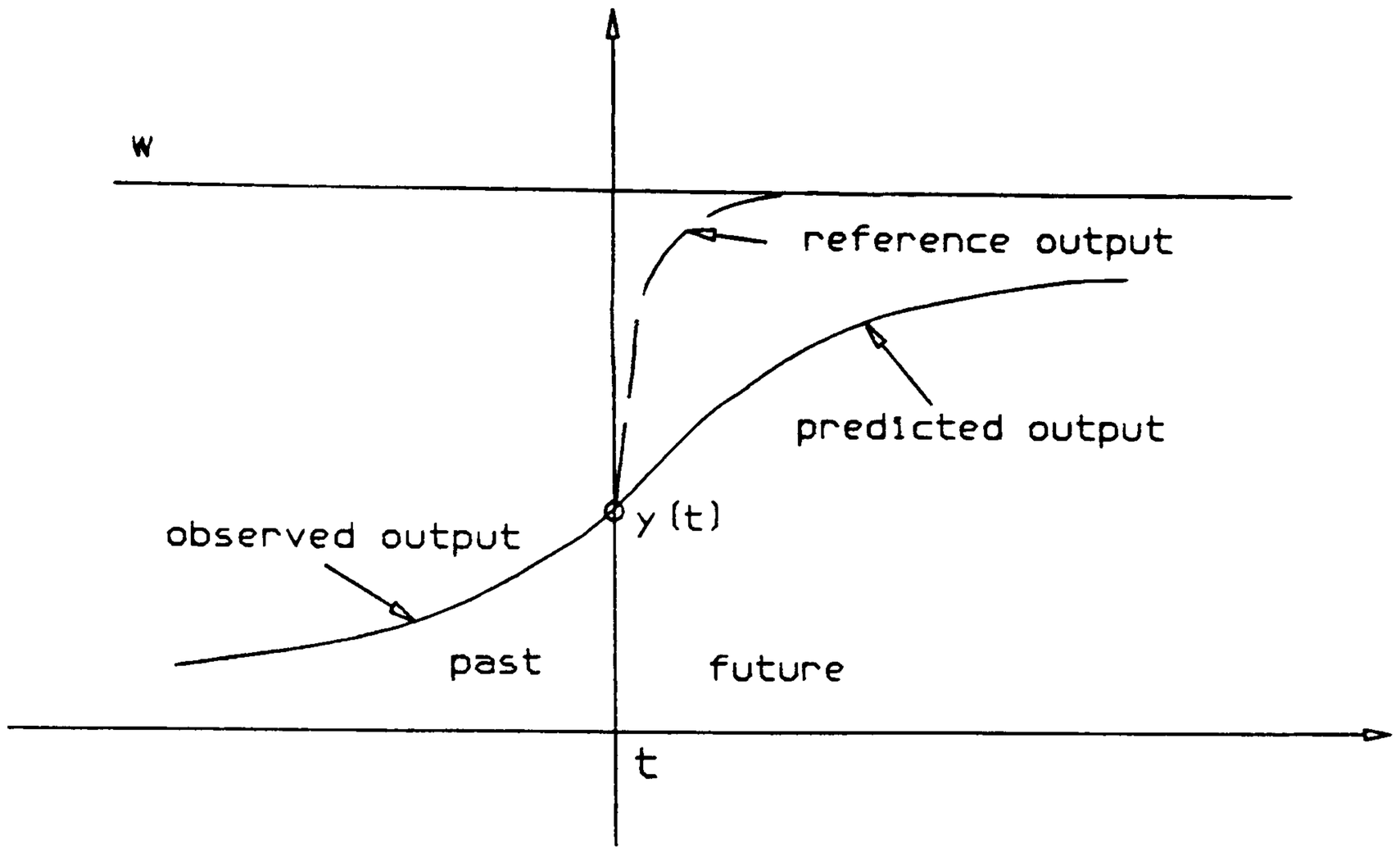


Figure 4.1 Graphical illustration of the CGPC strategy

In order to have the same structure as the output predictor (eqn. 4.20), the reference output is approximated as a truncated Maclaurin series

$$w_r^*(t,T) = \sum_{k=0}^{N_y} w_k \frac{T^k}{k!} \tag{4.47}$$

where

$$w_k = r_k[w(t)-y(t)] \tag{4.48}$$

and r_k is the k^{th} markov parameters of $\frac{R_n(s)}{R_d(s)}$. In the control law calculation the following matrix form of the eqn. (4.47) is more appropriate

$$w_r^*(t,T) = \underline{T}_{N_y}\ \underline{w} \tag{4.49}$$

where

$\underline{T}_{N_y}$ is a row vector as defined before

$$\underline{w} = [\, w_0\ w_1 \ldots w_{N_y}\,]^T \tag{4.50}$$

or

$$\underline{w} = \underline{r}\,[\,w(t)-y(t)\,] \tag{4.51}$$

where $\underline{r}$ is a column vector which contains the markov parameters of $\frac{R_n(s)}{R_d(s)}$.

$$\underline{r} = [\,r_0\, r_1 \ldots r_{N_y}\,]^T \tag{4.52}$$

If the future setpoint is known, then this can be used instead of reference output. To do this, the future setpoint should be approximated as a truncated Maclaurin series as in the form of eqn. (4.49). This can then be used in the control law calculations. However, note that if the future setpoint is not a continuous function in the given time frame, this approximation can be a problem. Therefore, setpoints which are discontinuous, such as a square wave, are not easily incorporated into the design unlike discrete GPC.

4.2.4. Control Law

The CGPC, like the GPC, is based on a receding time frame (Thomas, 1975, Kwon, 1977, Chen, 1982, Longchamp, 1983, Yaz, 1984, Selbuz, 1987). That is at a give time t the cost function minimization occurs not with respect to t but with respect to a receding time frame whose origin is at time t. Thus for each time t, a pseudo input $u_r(t,T)$, a pseudo output $y_r(t,T)$ and a pseudo setpoint $w_r(t,T)$ are considered where T is the receding time variable and t is a constant for that time frame. These pseudo variables are defined so as to be directly related to the actual system variables at T=0:

$$y_r(t,0) = 0 \tag{4.53}$$

$$u_r(t,0) = u(t) \tag{4.54}$$

These pseudo variables are undefined for $T < 0$ and have no direct relationship with the actual variables. In particular, it is *not* generally true that $u_r(t,T) = u(t+T)$.

In the discrete-time GPC, output prediction† depends on the future controls which are to be

† The term 'output prediction' is shortly used for the j-step ahead (T-ahead for the continuous-time case) output prediction based on the information available at time t, where j (T) is any integer (real) number greater than the system time delay.

determined. Suppose that the future controls are known, then the predicted output can be calculated. Reverse operation is also possible: given a predicted output over a time frame the corresponding future controls can be calculated. These future controls will be called predicted controls. GPC does this reverse operation by minimizing a cost function over the given time frame. The first element $u(t)$ of the predicted controls is then applied to the system and the same procedure is repeated at the next time instant.

However, in continuous-time the predicted output depends on the input $u(t)$ and its derivatives (see predictor equation 4.20). In other words, future control (predicted control) is a polynomial of order N_u in T. If the input and its derivatives are known the predicted output can be calculated or given the output prediction over a time frame corresponding input and input derivatives can be calculated. The objective of the CGPC is then to find the input and its derivatives such that predicted output is as close as possible to the reference output. This is done by minimizing a cost function, similar to the one in discrete case, over the given time frame. Having obtained the Maclaurin representation of the pseudo control (predicted control), only the first term of the series is used in computing $u(t)$ from (4.54).

Now consider the following cost function

$$J = \int_{T_1}^{T_2} [y_r^*(t,T) - w_r^*(t,T)]^2 dT + \lambda \int_0^{T_2 - T_1} [u_r^*(t,T)]^2 dT \tag{4.55}$$

where

$$y_r^*(t,T) = y^*(t+T) - y(t) \tag{4.56}$$

$$u_r^*(t,T) = \sum_{k=0}^{N_u} u_k(t) \frac{T^k}{k!} \tag{4.57}$$

or in the matrix form

$$u_r^*(t,T) = \underline{T}_{N_u}\, \underline{u} \tag{4.58}$$

$$\underline{T}_{N_u} = [\, 1 \; T \; \frac{T^2}{2!} \; \cdots \; \frac{T^{N_u}}{N_u!} \,] \tag{4.59}$$

$$\underline{u} = [\, u(t) \; u_1(t) \; \cdots \; u_{N_u}(t) \,]^T \tag{4.60}$$

T_1 = *minimum prediction horizon*

T_2 = *maximum prediction horizon*

λ = *control weighting*

Note that $y_r^*(t,T)$ is exactly given by the same predictor equation (eqn. 4.20) except that the first element of $\underline{Y}^o$ is set to zero, this new vector will de denoted by $\bar{\underline{Y}}^o$.

$$\bar{\underline{Y}}^o = [\,0 \;\; y_1^o(t) \;\ldots\ldots\; y_{N_y}^o(t)\,]^T \tag{4.61}$$

For the Laplace domain equation (eqn. 4.21) this is equivalent to setting the first row of the matrix F to zero.

Note that we include a minimum prediction horizon T_1 in the cost. In the general case, T_1 is zero but, if the system has a time delay, T_1 can be set equal to the delay. If the time delay is not known then T_1 can be set equal to the largest possible delay. Since the predicted input $u_r^*(t,T)$ for $T > T_2 - T_1$ will have effect on $y_r^*(t,T)$ for $T > T_2$ (which are not included in the cost) when T_1 = time delay, then there is no point in weighting the $u_r^*(t,T)$ for $T > T_2 - T_1$. This is also included in the cost, which has not been taken in consideration in the discrete GPC.

The CGPC control law can be restated as follows:

1. Find the vector $\underline{u}$ which minimizes the above cost (4.55).

2. Use the first element of $\underline{u}$ $u(t)$ as control input.

With the substitution of the eqn. (4.20) , eqn. (4.49) and eqn. (4.58) into eqn. (4.55), the cost becomes

$$J = \int_{T_1}^{T_2} (\underline{T}_{N_y} H \,\underline{u} + \underline{T}_{N_y} \bar{\underline{Y}}^o - \underline{T}_{N_y} \underline{w})^2 dT + \lambda \int_0^{T_2 - T_1} \underline{u}^T \underline{T}_{N_u}^T \underline{T}_{N_u} \underline{u}\, dT \tag{4.62}$$

The minimization of the J results in

$$\underline{u} = K(\underline{w} - \bar{\underline{Y}}^o) \tag{4.63}$$

where

$$K = (H^T T_y H + \lambda T_u)^{-1} H^T T_y \tag{4.64}$$

$$T_y = \int_{T_1}^{T_2} \underline{T}_{N_y}^T \underline{T}_{N_y} \, dT \qquad is \quad (N_y+1)\times(N_y+1) \tag{4.65}$$

$$T_u = \int_{0}^{T_2 - T_1} \underline{T}_{N_u}{}^T \underline{T}_{N_u} \, dT \qquad is \quad (N_u+1)\times(N_u+1) \tag{4.66}$$

Note that T_y and T_u are symmetric. Evaluation of the integrals in eqn. (4.65) and eqn. (4.66) reveals that the ij^{th} element of the matrix T_y is given by

$$t^y{}_{ij} = \frac{T_2^{i+j+1} - T_1^{i+j+1}}{(i+j+1)i\,!\,j\,!} \tag{4.67}$$

and the ij^{th} element of the matrix T_u is given by

$$t^u{}_{ij} = \frac{(T_2 - T_1)^{i+j+1}}{(i+j+1)i\,!\,j\,!} \tag{4.68}$$

Note that T_u becomes submatrix of T_y when T_1=0. Let the first row of K be $\underline{k}$, then CGPC control law is given by

$$u(t) = \underline{k}\ [\ \underline{w} - \overline{\underline{Y}}^o\ ] \tag{4.69}$$

In the Laplace domain

$$U(s) = \underline{k}\ [\ \underline{r}\ [W(s) - Y(s)] - \underline{Y}^o(s)] \tag{4.70}$$

eqn. (4.70) can be rearranged in a transfer function form

$$U(s) = g\,[W(s) - Y(s)] - \frac{G_c(s)}{C(s)}\,U(s) - \frac{F_c(s)}{C(s)}\,Y(s) \tag{4.71}$$

where the scalar gain g and the polynomials $G_c(s)$ and $F_c(s)$ are given by

$$g = \underline{k}\ \underline{r} \tag{4.72}$$

$$G_c(s) = \underline{k}\ G\ \underline{S}_G \tag{4.73}$$

$$F_c(s) = \underline{k}\ F\ \underline{S}_F \tag{4.74}$$

The feedback system given by the CGPC control law (eqn. 4.71) is illustrated in figure 4.2.

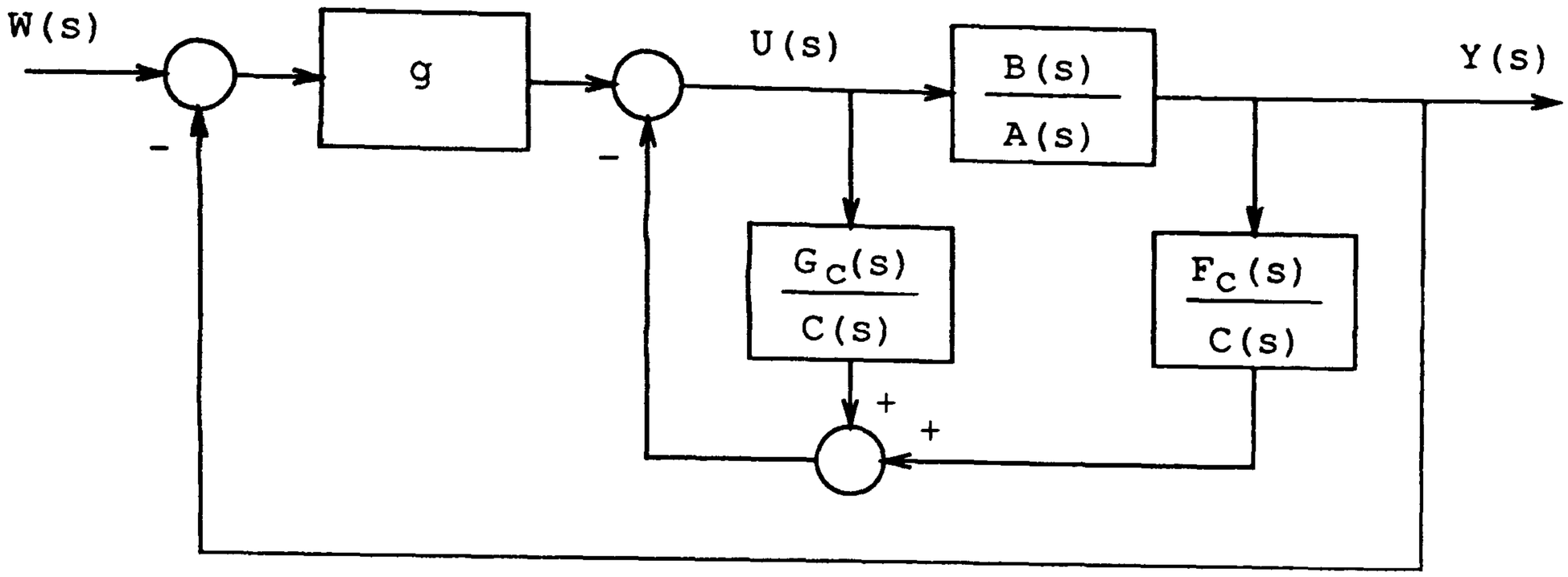


Figure 4.2 The feedback system of CGPC

The CGPC control law can be rewritten in the following form

$$W(s) = \Phi_e(s) \tag{4.75}$$

where

$$\Phi_e(s) = \frac{G_e(s)}{C(s)}\,U(s) + \frac{F_e(s)}{C(s)}\,Y(s) \tag{4.76}$$

$$G_e(s) = \frac{G_c(s) + C(s)}{g} \tag{4.77}$$

$$F_e(s) = \frac{F_c(s)}{g} + C(s) \tag{4.78}$$

Remark 1: Note that the CGPC control law is obtained by setting the output of an equivalent emulator equal to the setpoint (eqn. 4.75). The equivalent emulator polynomials satisfy the pole-placement identity $PC = G_eA + F_eB$, but P is implicitly specified by the CGPC algorithm.

Remark 2: An important result of using the receding time frame is that, although the control law required to realize $u_r^*(t,T)$ (for fixed t and varying T) would vary with T, the actual control law required to realize $u(t) = u_r^*(t,0)$ for variable t and fixed $T=0$ does not depend on t. Thus, like the GPC, the CGPC is a time invariant control law.

Remark 3: In the development of the above control law it is assumed that $deg(C) = deg(A) - 1$. If $deg(C) = deg(A)$, then the output $y(t)$ should be replaced by its emulated value.

Remark 4: The eqn. (4.63) has a unique solution, if the matrix $(H^T T_y H + \lambda T_u)$ is nonsingular. The matrices T_y and T_u are positive definite, and matrix H has always full rank because of its structure. This means that the matrix $(H^T T_y H + \lambda T_u)$ is also positive definite and thus nonsingular. However, the above argument is purely based on a theoretical viewpoint, numerically speaking, for small values of T_2 the matrix $(H^T T_y H + \lambda T_u)$ can be singular. This can be easily overcome by increasing T_2 or choosing a smaller N_u.

Minimum of the cost function

The cost function (eqn. 4.62) can be written as follows

$$J = (H\underline{u} + \underline{\bar{Y}}^o - \underline{w})^T T_y (H\underline{u} + \underline{\bar{Y}}^o - \underline{w}) + \lambda \underline{u}^T T_u \underline{u} \tag{4.79}$$

this can be rearranged as

$$J = \underline{u}^T (H^T T_y H + \lambda T_u) \underline{u} + 2 (\underline{\bar{Y}}^o - \underline{w})^T T_y H \underline{u} + (\underline{\bar{Y}}^o - \underline{w})^T T_y (\underline{\bar{Y}}^o - \underline{w}) \tag{4.80}$$

substitution of the control law (eqn. 4.63) into eqn. 4.80 gives the following minimum of the cost.

$$J_{\min} = (\underline{w} - \underline{\bar{Y}}^o)^T [T_y - T_y H (H^T T_y H + \lambda T_u)^{-1} H^T T_y] (\underline{w} - \underline{\bar{Y}}^o) \tag{4.81}$$

More on the control law

Consider the system model (eqn. 4.1) with zero disturbance input, that is

$$Y(s) = \frac{B(s)}{A(s)} U(s) \tag{4.82}$$

then, the realisable part of the emulator (eqn. 4.9) can be written as

$$Y_k^o(s) = \frac{G_k(s)A(s) + F_k(s)B(s)}{C(s)A(s)} U(s) \tag{4.83}$$

using the decomposition identities (eqn. 4.5 and eqn. 4.6), it can be shown that

$$G_k(s)A(s) + F_k(s)B(s) = C(s)L_k(s) \tag{4.84}$$

where $L_k(s)$ satisfies the following identity

$$\frac{s^k B(s)}{A(s)} = H_k(s) + \frac{L_k(s)}{A(s)} \tag{4.85}$$

hence, $Y_k^o(s)$ becomes

$$Y_k^o(s) = \frac{L_k(s)}{A(s)} U(s) \tag{4.86}$$

This reveals that when there are no disturbances, k^{th} derivative emulator gives exact k^{th} derivative of the output, that is

$$Y_k(s) = \frac{s^k B(s)}{A(s)} U(s) = H_k(s)U(s) + \frac{L_k(s)}{A(s)} U(s) = Y_k^*(s) \tag{4.87}$$

We will use this fact to establish a relationship between $\underline{Y}^o$ and the states of the system. To do this, consider a state-space representation of the eqn. (4.82)

$$\underline{x}^{\cdot}(t) = A\ \underline{x}(t) + b\ u(t) \tag{4.88}$$

$$y(t) = c\ \underline{x}(t) \tag{4.89}$$

Consecutive derivatives of the output can be arranged in the following matrix form (Kailath, 1980)

$$\underline{y} = H_x\ \underline{u} + Q\ \underline{x}(t) \tag{4.90}$$

where

$$\underline{y} = [\ y(t)\ \ y_1(t)\ .....\ y_{N_y}(t)\ ]^T \tag{4.91}$$

$$\underline{u} = [\ u(t)\ \ u_1(t)\ .....\ u_{N_y-1}(t)\ ]^T \tag{4.92}$$

$$Q = \begin{bmatrix} c \\ cA \\ cA^2 \\ . \\ . \\ cA^{N_y} \end{bmatrix} \tag{4.93}$$

and H_x is a lower triangular Toeplitz matrix

$$H_x = \begin{bmatrix} 0 & 0 & 0 & 0 & . & . & 0 \\ cb & 0 & 0 & 0 & . & . & 0 \\ cAb & cb & 0 & 0 & . & . & 0 \\ cA^2b & cAb & cb & 0 & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ cA^{N_y-1}b & . & . & . & . & . & cb \end{bmatrix} \tag{4.94}$$

where $cA^{k-1}b$ is the k^{th} markov parameter of the system, that is $h_k = cA^{k-1}b$ so $H_x = H$. Comparing eqn. (4.90) with the eqn. (4.17) one can see that, when there are no disturbances, they are equivalent and thus

$$\underline{Y}^o = Q\ \underline{x}(t) \tag{4.95}$$

and

$$\overline{Y}^o = \overline{Q}\ \underline{x}(t) \tag{4.96}$$

where

$$\overline{Q} = \begin{bmatrix} \underline{0} \\ cA \\ cA^2 \\ \cdot \\ \cdot \\ cA^{N_y} \end{bmatrix} \tag{4.97}$$

where $\underline{0}$ is a zero row vector with appropriate dimension. Substitution of the eqn. (4.96) into the control law (eqn. 4.69) results in

$$u(t) = \underline{k}\ \underline{w} - \underline{k}\ \overline{Q}\ \underline{x}(t) \tag{4.98}$$

this can be further written as

$$u(t) = g\ w(t) - \underline{f}\ \underline{x}(t) \tag{4.99}$$

where g is as defined before (eqn. 4.72) , $\underline{f}$ is the feedback gain vector and is given as follows.

$$\underline{f} = \underline{k}\ (\underline{r}\ c\ + \overline{Q}\ ) \tag{4.100}$$

The effect of $\frac{R_n(s)}{R_d(s)}$ is reflected by the $\underline{r}$ vector in the feedback gain $\underline{f}$, note that if $\frac{R_n(s)}{R_d(s)} = 1$ then $\underline{f}$ becomes

$$\underline{f} = \underline{k}\ Q \tag{4.101}$$

This is an interesting result showing that CGPC control law is actually a state feedback where the state feedback gain $\underline{f}$ is chosen by the CGPC algorithm in order to meet some specifications stated by the CGPC design parameters (T_1, T_2, N_u, λ, R_n/R_d). This also means that CGPC control law only alters the pole locations (zeros stay the same) which enables the control of non-minimum

phase systems with zero control weighting. Notice that this state feedback is made possible by the $C(s)$ polynomial, which acts as an observer polynomial as discussed in chapter 2, enabling us to obtain the state information from the input output data. The above results will also be obtained from a transfer function analysis of the CGPC closed-loop system in the following section.

4.3. THE CLOSED-LOOP SYSTEM

4.3.1. General Closed-loop Equations

Closed-loop setpoint response

Application of the CGPC control law (eqn. 4.71) to a system given by the eqn. (4.1) results in the following closed-loop setpoint response.

$$Y(s) = \frac{gB(s)C(s)}{A(s)C(s) + G_c(s)A(s) + F_c(s)B(s) + gB(s)C(s)} W(s) \tag{4.102}$$

Considering the eqn. (4.84) together with the equations for the $G_c(s)$ and $F_c(s)$ polynomials (eqn. 4.73 and eqn. 4.74) one can easily see the following relationship

$$G_c(s)A(s) + F_c(s)B(s) = L_c(s)C(s) \tag{4.103}$$

where the $L_c(s)$ polynomial is given as follows

$$L_c(s) = \underline{k}\ L\ \underline{S_L} \tag{4.104}$$

where L is the $(N_y + 1)\times n$ coefficients matrix of the $L_k(s)$ polynomials given by the identity (4.85) and $\underline{S_L}$ is the corresponding $n\times 1$ s vector, that is

$$L = \begin{bmatrix} \underline{0} \\ \underline{L}_1 \\ \underline{L}_2 \\ \cdot \\ \cdot \\ \underline{L}_{N_y} \end{bmatrix} \tag{4.105}$$

$$\underline{L_k} = [\, l_{k0}\ l_{k1} \ldots l_{k(n-1)} \,] \quad ; \qquad k = 1,2, \ldots, N_y \tag{4.106}$$

$$\underline{S_L} = [\, s^{n-1}\ s^{n-2} \ldots s\ 1 \,]^T \tag{4.107}$$

where $\underline{0}$ is an appropriate dimension zero row vector. It follows from eqn. (4.103) that $C(s)$ is a factor of both numerator and denominator of the closed-loop system hence cancellation of this common factor results in the following closed-loop transfer function.

$$Y(s) = \frac{gB(s)}{A(s) + L_c(s) + gB(s)} W(s) \tag{4.108}$$

The closed-loop transfer function shows that CGPC control law only alters the pole locations leaving the zeros untouched. In addition to this we see that the closed-loop system has the same degree as the open-loop system (degree of $L_c(s)$ is n - 1). Notice that these are also properties of a state feedback.

The feedback system given by the eqn. (4.108) is shown in figure 4.3. In this figure, the inner loop actually corresponds to a state feedback where the partial state and its derivatives (Wolovich, 1974, Kailath, 1980) are fed back by the gain vector $\underline{kL}$. In addition to this state feedback in the figure there is also an output feedback, but this output feedback can be incorporated into the state feedback by modifying the figure 4.3 as in figure 4.4. The state feedback gain then becomes

$$\underline{f_c} = \underline{k} \ (L + \underline{r} \ \underline{B} \ ) \tag{4.109}$$

where $\underline{B}$ is a $1 \times n$ row vector which contains the coefficients of the $B(s)$ polynomial. Note that if $deg\,(B(s)) < n-1$ then corresponding leading elements of $\underline{B}$ will be zero. This result is a special case of the general result of the previous section (eqn. 4.100): here the controller canonical realization is assumed and subscript c denotes this. Note that when $\frac{R_n(s)}{R_d(s)} = 1$ $\underline{f_c}$ becomes

$$\underline{f_c} = \underline{k} \ \overline{L} \tag{4.110}$$

where

$$\overline{L} = \begin{bmatrix} \underline{B} \\ \underline{L}_1 \\ \underline{L}_2 \\ \cdot \\ \cdot \\ \underline{L}_{N_y} \end{bmatrix} \tag{4.111}$$

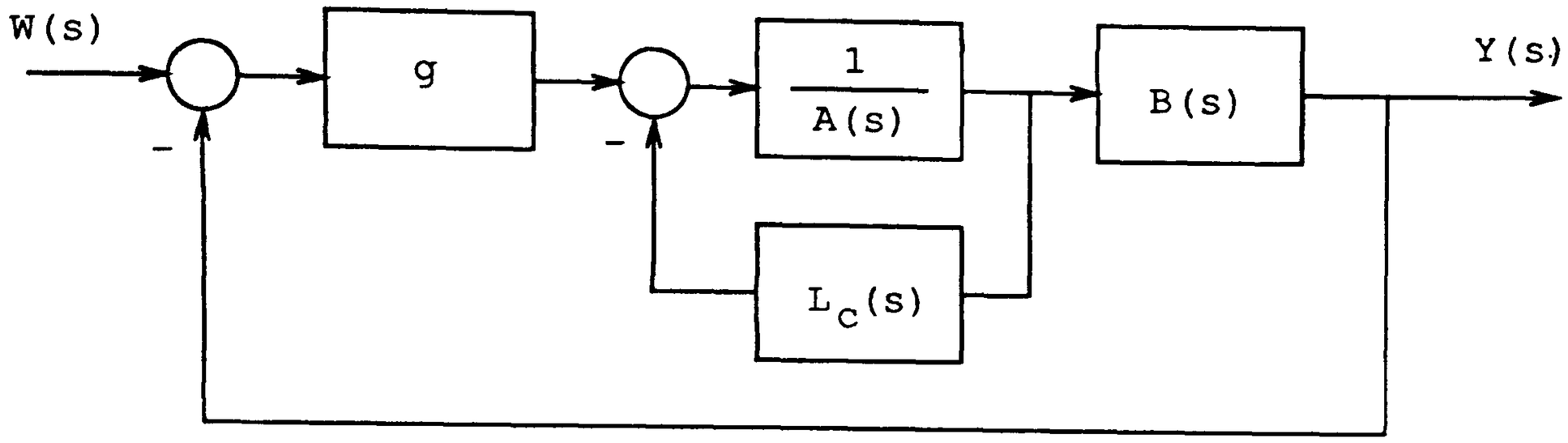


Figure 4.3 Equivalent CGPC feedback system

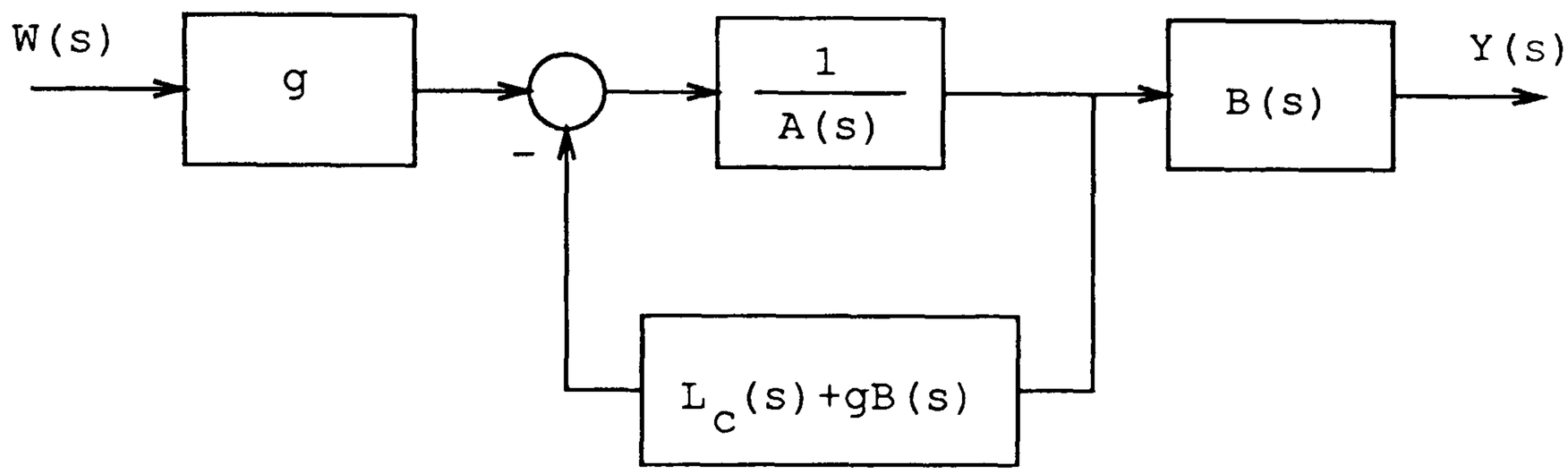


Figure 4.4 CGPC feedback system in the state feedback form

So when $\frac{R_n(s)}{R_d(s)} = 1$ the closed-loop system can be written as

$$Y(s) = \frac{k_0 B(s)}{A(s) + \bar{L}_c(s)} W(s) \tag{4.112}$$

where

$$\bar{L}_c(s) = \underline{k}\ \bar{L}\ \underline{S_L} \tag{4.113}$$

and k_0 is the first element of the row $\underline{k}$.

An interesting result of the above argument is that $L_k(s)$ polynomials can be expressed in terms of the matrices of the controller canonical realization (A_c, b_c, c_c), that is

$$L_k(s) = \underline{L_k}\ \underline{S_L} = c_c\ A_c^k\ \underline{S_L} \tag{4.114}$$

This also suggests the following recursion for $\underline{L_k}$

$$\underline{L_{k+1}} = \underline{L_k}\ A_c \tag{4.115}$$

with the initial coefficient vector

$$\underline{L}_0 = \underline{B} = c_c \tag{4.116}$$

Note that using the identity (4.85) the $L_c(s)$ polynomial can be written as

$$L_c(s) = B(s)\,\underline{k}\;\underline{S} - A(s)\,\underline{k}\;H_f\,\underline{S}_{H_f} \tag{4.117}$$

where

$$\underline{S} = [\,0\;s\;s^2\;.\;.\;.\;.\;s^{N_y}\,]^T \tag{4.118}$$

and H_f is the full H matrix, namely the H matrix when $N_u = N_y - \rho$ and $\underline{S}_{H_f}$ is the corresponding s vector. $L_c(s)$ further can be written as

$$L_c(s) = B(s)Z_c(s) - A(s)H_c(s) \tag{4.119}$$

where the polynomials $Z_c(s)$ and $H_c(s)$ are defined as follows

$$Z_c(s) = \underline{k}\;\underline{S} \tag{4.120}$$

$$H_c(s) = \underline{k}\;H_f\,\underline{S}_{H_f} \tag{4.121}$$

with the substitution of eqn. (4.119) into eqn. (4.108) a new expression for the closed-loop system is obtained as follows

$$Y(s) = \frac{gB(s)}{A(s)\,(1 - H_c(s)) + B(s)\,(Z_c(s) + g)}\,W(s) \tag{4.122}$$

This expression will be used to prove some properties of the closed-loop system in the following subsections.

Closed-loop disturbance response

The closed-loop disturbance response is give by

$$Y(s) = \frac{(G_c(s) + C(s))\,A(s)}{(A(s) + L_c(s) + gB(s))\,C(s)}\,\Psi(s) \tag{4.123}$$

where $\Psi(s)$ is the direct disturbance at the output and given by $\Psi(s) = \frac{C(s)}{A(s)}\,V(s)$. The eqn.

(4.123) can be divided into two parts

$$Y(s) = \frac{A(s)}{A(s) + L_c(s) + gB(s)} \Psi(s) + \frac{G_c(s)A(s)}{(A(s) + L_c(s) + gB(s))\, C(s)} \Psi(s) \qquad (4.124)$$

Although the second part of the closed-loop disturbance response may be adjusted by $C(s)$ polynomial without effecting the closed-loop setpoint response, as a whole we do not have enough flexibility to adjust the disturbance transients separately from the closed-loop setpoint response.

Closed-loop control input

The closed-loop control input is given by

$$U(s) = \frac{gA(s)}{A(s) + L_c(s) + gB(s)} W(s) - \frac{(gC(s) + F_c(s))\, A(s)}{(A(s) + L_c(s) + gB(s))\, C(s)} \Psi(s) \qquad (4.125)$$

4.3.2. A Special Case

There is a special case where $\lambda = 0$ and $N_u = N_y - \rho$ then CGPC control law becomes a cancellation law. Namely, closed-loop pole polynomial has B as a factor. This obviously gives unstable control for the non-minimum phase systems. In this special case, the closed-loop system is given as follows.

$$Y(s) = \frac{1}{Z(s)} W(s) \qquad (4.126)$$

where

$$Z(s) = \frac{Z_c(s) + g}{g} \qquad (4.127)$$

The proof is simple: consider the equation for the gain matrix K (eqn. 4.64), then it can easily be shown that when $\lambda = 0$ and $N_u = N_y - \rho$, $H_c(s)$ polynomial is equal to 1, that is $H_c(s) = 1$. This ends the proof (see eqn. 4.122). Note that $deg(Z(s)) = \rho$.

In general, the analysis of the closed-loop pole locations in terms of the CGPC design parameters seems impossible analytically. In this special case, however, this can be done to some extent, which may also provide some insight into the general case.

Now consider the H matrix, it can be decomposed as follows

$$H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \tag{4.128}$$

where H_1 is a zero matrix with the dimension $\rho \times (N_y - \rho + 1)$ and H_2 is a lower triangular square matrix in the following form.

$$H_2 = \begin{bmatrix} h_\rho & 0 & . & . & 0 \\ h_{\rho+1} & h_\rho & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ h_{N_y} & . & . & . & h_\rho \end{bmatrix} \tag{4.129}$$

The matrix T_y can also be decomposed appropriately as

$$T_y = \begin{bmatrix} T_{y11} & T_{y12} \\ T_{y21} & T_{y22} \end{bmatrix} \tag{4.130}$$

then, it can be shown that the gain matrix K (eqn. 4.64) can be written as follows

$$K = H_2^{-1} \; [\; T_{y22}^{-1} T_{y21} \quad I \;] \tag{4.131}$$

where I is a unit matrix with appropriate dimension. Since H_2 is lower triangular its inverse is also lower triangular, hence the first row of K is given as

$$\underline{k} = [\; \frac{1}{h_\rho} \; 0 \ldots . 0 \;] \; [\; T_{y22}^{-1} T_{y21} \quad I \;] \tag{4.132}$$

or

$$\underline{k} = \frac{1}{h_\rho} \; [\; \underline{T_c} \; 1 \; 0 \ldots . 0 \;] \tag{4.133}$$

where $\underline{T_c}$ is the first row of $T_{y22}^{-1} T_{y21}$. Note that the dimension of $\underline{T_c}$ is $1 \times \rho$.

The elements of $\underline{T_c}$ are non-linear functions of different powers of T_1 and T_2, so even in this simple case a general expression for the closed-loop pole polynomial (in this case $Z(s)$) will be quite complex. Therefore we will not examine $Z(s)$ in general case instead we will consider two special cases where $\rho = 1$ and $\rho = 2$.

Case 1: Here ρ=1, thus $\underline{T}_c$ is scalar say t_c and we assume a first order R_n/R_d as follows

$$\frac{R_n(s)}{R_d(s)} = \frac{1}{rs + 1} \tag{4.134}$$

then the first markov parameter r_0 of R_n/R_d is zero, that is

$$\underline{r} = [\, 0\, r_1\, r_2 \ldots . r_{N_y}\,]^T \tag{4.135}$$

Using eqn. (4.127) it is easy to show that, in this case, $Z(s)$ is given as follows

$$Z(s) = \frac{1}{r_1}s + 1 \tag{4.136}$$

one may also note that $r_1 = \frac{1}{r}$ hence $Z(s)$ becomes

$$Z(s) = rs + 1 \tag{4.137}$$

This means that under these circumstances we obtain exact model-following regardless to the choice of T_1 and T_2. Note that if $R_n/R_d = 1$ then $Z(s)$ will be

$$Z(s) = \frac{1}{t_c}s + 1 \tag{4.138}$$

and pole location can be adjusted by T_1 and T_2 : generally T_1 is chosen as zero and T_2 is used for adjustment.

Case 2: Here ρ = 2 thus $\underline{T}_c = [t_{c1}\ \ t_{c2}]$ and we assume a second order R_n/R_d as follows

$$\frac{R_n(s)}{R_d(s)} = \frac{1}{r_a s^2 + r_b s + 1} \tag{4.139}$$

then $\underline{r}$ will be

$$\underline{r} = [\, 0\, 0\, r_2 \ldots . r_{N_y}\,]^T \tag{4.140}$$

It can be shown that in this case $Z(s)$ is given by

$$Z(s) = \frac{1}{r_2}s^2 + \frac{t_{c2}}{r_2}s + 1 \tag{4.141}$$

one may note that $r_2 = \frac{1}{r_a}$ then $Z(s)$ becomes

$$Z(s) = r_a s^2 + r_a t_{c2}\, s + 1 \tag{4.142}$$

If t_{c2} is adjusted to be $t_{c2} = \frac{r_b}{r_a}$ by T_1 and T_2 ,which is always possible, then we obtain exact model following. This type of exact model-following is not possible for $\rho > 2$ since more than one t_c terms appears in $Z(s)$ making it impossible to adjust them independently.

If R_n/R_d is chosen to be a first order as in case 1, then $Z(s)$ will be given as follows.

$$Z(s) = (rs + 1)(\frac{r}{rt_{c2} - 1}\, s + 1) \tag{4.143}$$

This equation shows that, in this case, irrespective of T_1 and T_2, one of the closed-loop poleswill be at the location defined by R_d. The other pole can be replaced far away from the imaginary axis by the proper choice of T_1 and T_2. This results in a very close model-following. This is also true for $\rho > 2$, that is one of the closed-loop poleswill be at the location defined by R_d and replacing the other poles further left from this pole, model-following type control can be obtained to some extent. Note that if $R_n/R_d = 1$ then $Z(s)$ is given by

$$Z(s) = \frac{1}{t_{c1}}(s^2 + t_{c2}\, s + t_{c1}) \tag{4.144}$$

4.3.3. The Effect of Common Factors

In self-tuning control if the system model is overspecified, a common factor will appear in the estimated model. Therefore, it is important to examine the case where the system does not have a common factor but the model which the design is based on has a common factor. Consider the following model

$$A(s) = A'(s)\, X(s) \quad ; \quad B(s) = B'(s)\, X(s) \tag{4.145}$$

where $A'(s)$ and $B'(s)$ are the actual system polynomials, $X(s)$ is a common factor. There are two questions to be answered:

1- Does the common factor create any problem in the control law calculations?

2- How does the common factor effect the control law?

Examination of the decomposition identities (eqn. 4.5 and 4.6) shows that common factors will not create any problem in the solution but, for different common factors we will have different $F_k(s)$ and $G_k(s)$ polynomials. Although $H_k(s)$ polynomials and thus the vector $\underline{k}$ do not depend on the common factor (this is apparent from the identity (4.85)), this gives rise to different $G_c(s)$ and $F_c(s)$ polynomials for different common factors. Note that gain g is independent of common factor.

As the control law is applied to the actual system, the closed-loop system (eqn. 4.102) will be given in terms of actual system polynomials $A'(s)$ and $B'(s)$. So we will have the term $G_c(s)A'(s) + F_c(s)B'(s)$ in the denominator instead of $G_c(s)A(s) + F_c(s)B(s)$. Then the question is what this term will be as $G_c(s)$ and $F_c(s)$ are different for different common factors. The examination of the identity (4.85) shows that

$$L_c(s) = L_c'(s)\, X(s) \tag{4.146}$$

where $L_c'(s)$ is the $L_c(s)$ polynomial when $X(s) = 1$. It then follows from eqn. (4.103) that

$$G_c(s)\, A'(s) + F_c(s)\, B'(s) = L_c'(s)\, C(s) \tag{4.147}$$

This equation shows that whatever the common factor is, we end up with the same closed-loop system and thus the same control law.

Note that in the above analysis we did not make any distinction between the stable and unstable common factors: the above result is true for both cases. Note that the above result is also true when the system itself has a common factor but, in this case unstable common factors will result in unstable control.

So, an important feature of CGPC is that, unlike pole-placement it does not suffer from the ill effects of the pole-zero cancellation.

4.3.4. The Offset Problem

The CGPC closed-loop system has unit steady-state gain, regardless to the choice of parameters and the open-loop system, when $\lambda = 0$. The proof is as follows. Using the equation for the gain matrix K (eqn. 4.63) it can be shown that the constant coefficient of the $H_c(s)$ polynomial (eqn. 4.121) is 1 when $\lambda = 0$. Also note that $Z_c(s)$ polynomial (eqn. 4.120) has s as a factor. These two facts guarantee that the constant coefficient of the $L_c(s)$ polynomial (see eqn. 4.119) is $-a_n$ (a_n is the constant coefficient of $A(s)$) when $\lambda = 0$. This results in a closed-loop system with unit steady-state gain. Therefore, there will be no offset due to setpoint when $\lambda = 0$. This is not true when $\lambda \neq 0$ and the resulting offset is called $\lambda\text{–}offset$. This can be overcome by replacing λ by a transfer function $Q(s)$ such that $Q(0) = 0$. This type of control weighting is called dynamic control weighting and we will show how to include $Q(s)$ in the CGPC design later.

An other reason for the offset is the disturbances (specially dc or stepwise disturbances or the ones with a dc level). To remove the offset due to disturbances closed-loop disturbance transfer function must have a zero steady-state gain. However, examination of the CGPC disturbance transfer function (eqn. 4.123) reveals that the above type of disturbances will cause an offset at the output. This offset problem can easily be overcome by modeling the disturbances correctly that is, by taking into account dc type disturbances. This can be done by augmenting $A(s)$ and $B(s)$ polynomials by s as follows (Gawthrop, 1986, Gawthrop, 1987).

$$Y(s) = \frac{sB(s)}{sA(s)} U(s) + \frac{C(s)}{sA(s)} V(s) \tag{4.148}$$

This results in a disturbance transfer function with zero steady-state gain and thus remove this kind of offsets. The proof is as follows. Examination of the identity (4.5) reveals that $F_c(s)$ polynomial will have s as a factor. Then, from eqn. (4.103), by noting that the constant coefficient of the $L_c(s)$ is $-a_n$, one can see that the constant coefficient of $G_c(s)A(s)$ is $-a_n c_n$ where c_n is the constant coefficient of the $C(s)$ polynomial. Hence $G_c(s)A(s) + C(s)A(s)$ has s as a factor. This ends the proof.

4.4. THE EFFECTS AND CHOICE OF CGPC PARAMETERS

4.4.1. The Minimum Prediction Horizon T_1

This parameter is usually chosen as to be zero, but it is useful to choose $T_1 > 0$ when the system has a time delay or when it is non-minimum phase. If the system has a time delay then there is no point in setting T_1 less than the time delay since the corresponding output can not be effected by $u(t)$. If the time delay is not known then T_1 may be chosen equal to the largest possible delay. For the non-minimum phase systems T_1 may be chosen such that the negative going part is excluded. Although it is possible to obtain reasonable control for the time delay and non-minimum phase systems with $T_1 = 0$, the above choice of $T_1 > 0$ will improve the control performance for each case. T_1 corresponds to N_1 in the discrete time formulation (Clarke et al 1987).

4.4.2. The Maximum Prediction Horizon T_2

This parameter is equivalent to N_2 in the discrete-time formulation (Clarke et al 1987) and has the same effect in continuous-time. In general, the smaller value of T_2 corresponds to the faster output response and thus the more active control action. For the larger T_2, the slower output response is obtained. Therefore, T_2 can be used as a *knob* to adjust the rise time of the closed-loop output response. However, if a reference trajectory is specified by R_n/R_d the choice of T_2 needs to agree with the rise time of the model. If the system has an initially negative-going non-minimum phase response, the minimum value of T_2 should cover the later positive-going part.

An interesting limiting case is that when $\lambda = 0$ and $T_2 \to 0$, $Y(s) \to W(s)$. The argument of this point is similar to the one which will be given in section 4.6.2 under the heading *'Relation of CGPC with EBC'*. Clearly this gives unstable control for the non-minimum phase systems as the controller tries to cancel the system by its inverse.

4.4.3. The Predictor Order N_y

In the predictor design, the future output is approximated by a N_y^{th} order truncated Maclaurin series. It is obvious that approximation accuracy depends on N_y.† So N_y needs to be chosen such that a good approximation can be obtained over the range in which T varies. In other words, the sum of the terms $y_i(t)\frac{T_2^i}{i!}$ for $i > N_y$ should be reasonably small. However, as $y_i(t)$s are not known, it seems impossible to choose N_y on the basis of this argument.

Intuitively one may argue that if N_y is chosen such that we have a good approximation of the open-loop system step (or impulse) response over the range $0 < T < T_2$, then this will also result in a good approximation for the output predictor as the predictor design is based on the open-loop system. This intuitive argument can be supported as follows. Consider the output predictor (eqn. 4.20) when $N_u = 0$

$$y^*(t+T) = y_s^*(T)u(t) + \underline{T}_{N_y}\underline{Y}^o \tag{4.149}$$

where

$$y_s^*(T) = \underline{T}_{N_y}H = h_1T + h_2\frac{T^2}{2!} + \ldots . + h_{N_y}\frac{T^{N_y}}{N_y!} \tag{4.150}$$

As one may notice, $y_s^*(T)$ is the approximate step response (truncated Maclaurin series) of the open-loop system. We believe that if N_y is chosen such that $y_s^*(T)$ is approximated well over the range $0 < T < T_2$, $\underline{T}_{N_y}\underline{Y}^o$ will also be approximated well over the same range as it is the initial condition response of the same system.

As a conclusion, choosing N_y such that the error between the real and approximate step responses of the open-loop system over the range $0 < T < T_2$ is reasonably small will be a good criterion. This is also supported by simulation results.

---

† Approximation of a function f(t) with Taylor series about a specified point t_0 will be very good near to that point and very poor away from that point. Of course, the range ,in which the approximation is good, depends on the order of Taylor series (in our case N_y).

It may be useful to illustrate the procedure with an example. Consider the following system

$$\frac{A(s)}{B(s)} = \frac{-0.2s + 1}{s(s^2 + 1)} \tag{4.151}$$

The actual and approximate step responses of this system for various N_y over the range $0 < T < 5$ is given in figure 4.5. As can be seen form the figure , for a larger T_2 a larger N_y is needed and vice versa. For example, for $T_2 = 5$ N_y should be 12 whereas for $T_2 = 3$ $N_y = 6$ will be sufficient.

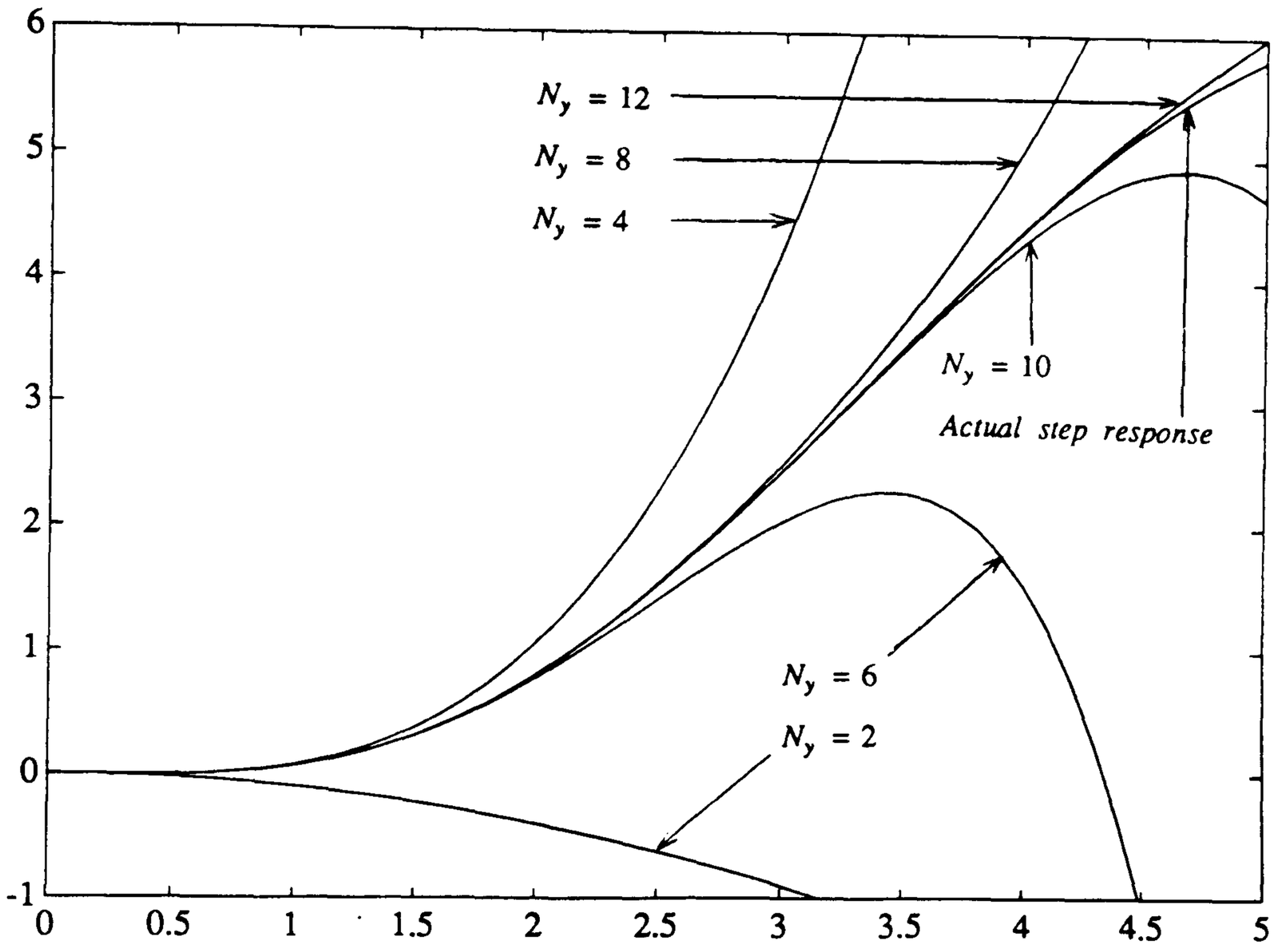


Figure 4.5 Graphical illustration of the choice of N_y with an example

As it is obvious from the above discussion, there is a close link between T_2 and N_y. Therefore these two parameters should always be considered together. However simulation studies showed that a large number of systems can be controlled reasonably well with the value of $N_y = 6$. For the simple plants this value can be reduced even further. But, for the complex systems (at the same time open-loop unstable, non-minimum phase and higher order) N_y can be as large as 30 depending on the complexity. This is the case generally with the time delay systems since their approximation becomes non-minimum phase and higher order. For example for the double integrator with the unit time delay $(\frac{e^{-s}}{s^2})$ N_y needs to be chosen around 20 when $T_2 = 2$.

Note that we do not use the step response representation in the predictor design, we introduce the step response only to give a criteria in order to choose N_y. However, it is interesting to note that N_y will never be as large as in the control methods based on step or impulse response representation such as DMC or IDCOM (Cutler, 1980, Richalet, 1978), since N_y is chosen in order to approximate a part of step (or impulse) response not all of it.

There is no *physical* equivalent to N_y in discrete time but *algebraically* N_y is equivalent to N_2 since both has the same effect on the dimension of the matrices.

4.4.4. The Control Order N_u

Control order N_u can be seen as a parameter to constrain the predicted control $u_r^*(t,T)$ since the form of the predicted control is defined by N_u; for example $N_u = 0$ assumes a constant into the future, $N_u = 1$ a ramp and so on. The smaller the N_u, the higher the constraint on the predicted control and vice versa. Note that in this way we indirectly constrain the control $u(t)$. No mathematical argument has yet been devised but it seems reasonable that more constraint on the predicted control $u_r^*(t)$ means more constrainton $u(t)$ or vice versa. As a result a small value of N_u gives less active control $u(t)$ and slow output response. Increasing N_u makes the control and the corresponding output more active until a stage is reached where any further increase in N_u makes little difference. Simulation results show that a value of $N_u = 0$ gives generally acceptable control for a large variety of systems but, an increased value of N_u is needed for the complex systems (higher order open-loop unstable and non-minimum phase systems).

The control horizon Nu in discrete time is an other way of constraining the predicted control. Because of this they have similar effects but, they are physically not the same. However N_u and Nu are algebraically equivalent since both have the same effect on the dimension of matrices and thus on the control law calculations.

4.4.5. The Control Weighting λ

In general, control weighting has the effect of reducing control activity and thus input energy. For the control of non-minimum phase systems λ is not needed and the effects of λ can also be accomplished by N_u and T_2 without associated drawback of λ–$offset$. Therefore we will

not consider λ as an important design parameter and it will usually be chosen as zero. However if LQ type control, which is given in the next section, is desired then λ needs to be $\lambda \neq 0$.

4.4.6. The Model R_n/R_d

The closed-loop system response can be specified by a reference output defined by a model R_n/R_d. Then CGPC control law tries to match the system response to the model output. But, in general, it is not possible to obtain exact model-following since CGPC only changes the closed-loop pole locations. Some special cases where exact model-following is obtained are examined in case 1 and case 2 of section 4.3.2. A first order R_n/R_d generally seems more appropriate to use. In this case CGPC places one of the closed-loop poles at the location specified by R_d, the rest away (the distance depends on the choice of T_2) from the imaginary axis when $N_u = N_y - \rho$ and $\lambda = 0$. Thus it is possible to obtain a very close model-following with the right choice of T_2. This model-following property becomes less accurate as N_u decreases from $N_y - \rho$. For a small N_u such as 0 or 1, no longer model-following is obtain instead the effect of R_n/R_d is that it smooths the response by penalizing the overshoot.

4.5. RELATION OF CGPC TO LQ CONTROL

The CGPC cost function is similar to the LQ cost function and previously it was also shown that CGPC control law correspond to a state feedback. Clearly, these facts suggest a relation between CGPC and LQ control and indeed CGPC becomes LQ control with the following setting of the design parameters.

$$R_n = R_d = 1$$

$$N_u = N_y - \rho$$

$$N_y \rightarrow \infty$$

$$T_1 = 0$$

$$T_2 \rightarrow \infty$$

However, the firm proof of the above argument seems difficult because of the fact that CGPC

design is based on transfer function representation whereas LQ design is based on state-space representation. Here we only give a heuristic discussion of this point. In section 2.2.3 , we discussed the relationship between the predicted future output $y^*(t+T)$ and the actual future output $y(t+T)$ and it was shown that $y^*(t+T)$ is an approximation to the noise-free future output $\tilde{y}(t+T)$ (eqn. 4.39). It was also noted that the approximation accuracy depends on N_y, N_u and T. From this argument, it is clear that when $N_u = N_y - \rho$ (largest possible N_u for a given N_y) and $N_y \rightarrow \infty$ the predicted output $y^*(t+T)$ approaches to $\tilde{y}(t+T)$ and by assuming no future noise this can be replaced by $y(t+T)$, and in the same way the predicted control can be replaced by $u(t+T)$. By considering that t is the initial time and choosing $T_1 = 0$, then the CGPC cost function can be written as

$$J = \int_{t}^{t+T_2} [y^2(T) + \lambda u^2(T)]dT \tag{4.152}$$

where setpoint has been omitted for simplicity. Note that this is also receding horizon LQ cost function for single-input single-output systems (Kwon, 1977, Longchamp, 1983). So CGPC can be considered as an approximation to the receding horizon LQ and it can be argued that when $N_u = N_y - \rho$, $R_n/R_d = 1$, $T_1 = 0$ and $N_y \rightarrow \infty$ this relation becomes exact. It is well known that when $T_2 \rightarrow \infty$ receding horizon LQ reduces to standard LQ see (Longchamp, 1983).

An interesting point, which supports the above argument , is that the minimum of the CGPC cost function and the LQ cost function are similar type. To make this point more clear consider the minimum of the CGPC cost (eqn. 4.81) when setpoint is zero

$$J_{\min} = \underline{x}^T(t)\, P\, \underline{x}(t) \tag{4.153}$$

where the symmetric matrix P is given as follows.

$$P = Q^T [\, T_y - T_y H\, (H^T T_y H + \lambda T_u)^{-1} H^T T_y \,] Q \tag{4.154}$$

By considering t is the initial time in the minimization, one can immediately notice that the eqn. (4.153) has the same form as the minimum of the LQ cost function (Kwakernaak, 1972). This also suggest a relation between the matrix P (eqn. 4.154) and the solution of the algebraic Riccati equation. Numerical calculations show that P converges to the solution of the algebraic Riccati

equation when $R_n/R_d = 1$, $N_u = N_y - \rho$, $N_y \to \infty$, $T_1 = 0$ and $T_2 \to \infty$.

4.6. SOME EXTENSIONS TO THE BASIC ALGORITHM

4.6.1. Inclusion of Systems With Zero Relative Order

In the development of the basic algorithm the system was assumed to be strictly proper as the intention was to keep the formulation simple. Although the basic formulation can also handle systems with zero relative order, for a proper formulation of the CGPC algorithm for these systems some modifications are needed. This is because systems with zero relative order has a feedforward from input to output unlike strictly proper systems and this needs to be taken into account in the design. The modifications are as follows:

1- choose $deg\,(C(s)) = deg\,(A(s))$,

2- emulate the output and replace the output by its emulated value throughout the basic design.

This enables us to separate the feedforward term from the rest. Let $y^*(t)$ be the emulated value of $y(t)$, then in the Laplace domain

$$Y^*(s) = H_0(s)U(s) + Y^o(s) \tag{4.155}$$

$$Y^o(s) = \frac{G_0(s)}{C(s)}\,U(s) + \frac{F_0(s)}{C(s)}\,Y(s) \tag{4.156}$$

where $H_0(s)$, $G_0(s)$ and $F_0(s)$ satisfies the following identities.

$$\frac{C(s)}{A(s)} = E_0(s) + \frac{F_0(s)}{A(s)} \tag{4.157}$$

$$\frac{E_0(s)B(s)}{C(s)} = H_0(s) + \frac{G_0(s)}{C(s)} \tag{4.158}$$

Note that $\frac{F_0(s)}{C(s)}$ is strictly proper, this prevents any feedforward from input to $y^o(t)$ and $H_0(s)U(s)$ corresponds to the feedforward from the input. The eqn. (4.155) can be written in the time domain as

$$y^*(t) = \underline{h}_0\underline{u} + y^o(t) \tag{4.159}$$

where

$$\underline{h}_0 = [\, h_0 \;\; 0 \;\; . \;\; . \;\; . \;\; 0 \,] \tag{4.160}$$

The differences which arise from the replacement of $y(t)$ by $y^*(t)$ in the predictor (eqn. 4.20) are: the first element of $\underline{Y}^o$ (eqn. 4.18) is now $y^o(t)$ instead of $y(t)$ and the first row of the H matrix is now $\underline{h}_o$, that is

$$H = \begin{bmatrix} h_0 & 0 & 0 & . & . & 0 \\ h_1 & h_0 & 0 & . & . & 0 \\ h_2 & h_1 & h_0 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & h_0 \\ . & . & . & . & . & . \\ h_{N_y} & . & . & . & . & h_{(N_y - N_u)} \end{bmatrix} \tag{4.161}$$

We also need to replace $y(t)$ by $y^*(t)$ in the reference output

$$w_r^*(t,T) = \underline{T}_{N_y}\, \underline{r}\, [w(t) - y^*(t)] \tag{4.162}$$

and in the $y_r^*(t,T)$

$$y_r^*(t,T) = y^*(t + T) - y^*(t) \tag{4.163}$$

Note that $y_r^*(t,T)$ is the same as before that is, the first row of the H matrix and the first element of the $\underline{Y}^o$ will be zero in the predictor equation for $y_r^*(t,T)$. The only difference in the cost function (eqn. 4.55) then arise from the existence of the vector $\underline{u}$ in the reference output. By taking this into account, the minimization of the cost function results in a similar control law for proper systems.

$$\underline{u} = K\, [\, \underline{r}(w(t) - y^o(t)) - \underline{\bar{Y}}^o \,] \tag{4.164}$$

where K is now given by

$$K = (H_p^T T_y H_p + \lambda T_u)^{-1} H_p^T T_y \tag{4.165}$$

where

$$H_p = H + \underline{r}\, \underline{h}_0 \tag{4.166}$$

Note that for strictly proper systems H_p reduces to H and thus matrix K becomes the same as before.

The closed-loop setpoint response for systems with zero relative order is then given as follows.

$$Y(s) = \frac{gB(s)}{A(s) + L_c(s) + gL_0(s)} W(s) \tag{4.167}$$

where g and $L_c(s)$ are defined as before and $L_0(s)$ satisfies the following identity.

$$\frac{B(s)}{A(s)} = H_0(s) + \frac{L_0(s)}{A(s)} \tag{4.168}$$

For strictly proper systems $H_0(s) = 0$, $L_0(s) = B(s)$, so eqn. (4.167) reduces to eqn. (4.108).

In the special case where $\lambda = 0$ and $N_u = N_y$, the open-loop zeros are canceled out by the closed-loop poles and thus the closed-loop transfer function becomes unity that is

$$Y(s) = W(s) \tag{4.169}$$

The proof is similar to the previous one and will not be repeated here. Notice that in the special case R_n/R_d must have zero relative order. This is because if R_n/R_d is strictly proper then the first row of H_p will be zero and thus H_p will be singular. As the transfer function in the special case is independent of R_n/R_d it can be taken as unity.

4.6.2. Auxiliary Output Approach

The use of an auxiliary output, instead of output itself, in the predictor design was first suggested by Clarke and Gawthrop in the GMV design (Clarke, 1975, Gawthrop, 1977, Clarke, 1979). This enables us to consider different control approaches such as model-reference and pole-placement in the same frame work. Later the same ideas were extended to the continuous-time with the notion of an emulator, which was explained in chapter 2. The auxiliary output approach is also used by Clarke et al (Clarke, 1987) in the GPC design to add some new design polynomials and thus to increase the capabilities of the algorithm. The extension of this approach to the CGPC is also straightforward.

Consider the following auxiliary output

$$\Phi(s) = \frac{P_n(s)}{B^-(s)P_d(s)}\, Y(s) \tag{4.170}$$

where $B^-(s)$ is the part of $B(s)$ which we do not want to cancel out. So, as in chapter 2, $B(s)$ is decomposed as $B(s) = B^-(s)B^+(s)$ where $B^-(s)$ contains all the zeros with positive real part and it may also contain zeros with negative real part. $\frac{P_n(s)}{B^-(s)P_d(s)}$ is chosen such that $\frac{P_n(s)B(s)}{B^-(s)P_d(s)A(s)}$ is *proper* and the steady-state gain of $\frac{P_n(s)}{B^-(s)P_d(s)}$ is unity (to ensure offset-free control), namely $\frac{P_n(0)}{B^-(0)P_d(0)} = 1$.

Now, the design steps of section 2 should be repeated for $\Phi(s)$ that is, design a T-ahead predictor $\phi^*(t+T)$ for $\phi(t)$ and minimize the cost function (eqn. 4.55) with $y^*(t+T)$ and $y(t)$ replaced by $\phi^*(t+T)$ and $\phi^*(t)$ respectively. This results in the following similar control law.

$$U(s) = g[\, W(s) - \Phi^o(s)\,] - \Phi_c(s) \tag{4.171}$$

where $\Phi^o(s)$ is the realisable part of the emulated value $\Phi^*(s)$ of $\Phi(s)$

$$\Phi^*(s) = H_0(s)U(s) + \Phi^o(s) \tag{4.172}$$

$$\Phi^o(s) = \frac{G_0(s)}{C(s)}\, U(s) + \frac{F_0(s)}{P_d(s)C(s)}\, Y(s) \tag{4.173}$$

$H_0(s)$, $G_0(s)$ and $F_0(s)$ are obtained from the following identities

$$\frac{P_n(s)C(s)}{B^-(s)P_d(s)A(s)} = \frac{E_0(s)}{B^-(s)} + \frac{F_0(s)}{P_d(s)A(s)} \tag{4.174}$$

$$\frac{E_0(s)B^+(s)}{C(s)} = H_0(s) + \frac{G_0(s)}{C(s)} \tag{4.175}$$

and

$$\Phi_c(s) = \frac{G_c(s)}{C(s)}\, U(s) + \frac{F_c(s)}{P_d(s)C(s)}\, Y(s) \tag{4.176}$$

Scalar gain g , $G_c(s)$ and $F_c(s)$ polynomials are defined as before. Note that $H_0(s) = 0$ and

$G_0(s) = E_0(s)B^+(s)$ when $\dfrac{P_n(s)B(s)}{B^-(s)P_d(s)A(s)}$ is strictly proper. The control law (eqn. 4.171) can be rearranged as

$$W(s) = \Phi_e(s) \tag{4.177}$$

$$\Phi_e(s) = \frac{G_e(s)}{C(s)}\,U(s) + \frac{F_e(s)}{P_d(s)C(s)}\,Y(s) \tag{4.178}$$

where

$$G_e(s) = G_0(s) + \frac{G_c(s) + C(s)}{g} \tag{4.179}$$

$$F_e(s) = F_0(s) + \frac{F_c(s)}{g} \tag{4.180}$$

Differences in the development of the above control law are as follows:

1- For $\phi^*(t+T)$, the identities (4.5) and (4.6) should be modified as

$$\frac{s^k P_n(s)C(s)}{B^-(s)P_d(s)A(s)} = \frac{E_k(s)}{B^-(s)} + \frac{F_k(s)}{P_d(s)A(s)} \tag{4.181}$$

$$\frac{E_k(s)B^+(s)}{C(s)} = H_k(s) + \frac{G_k(s)}{C(s)} \tag{4.182}$$

thus the emulator for the k^{th} derivative of $\phi(t)$ becomes

$$\Phi_k^*(s) = H_k(s)U(s) + \Phi_k^o(s) \tag{4.183}$$

$$\Phi_k^o(s) = \frac{G_k(s)}{C(s)}\,U(s) + \frac{F_k(s)}{P_d(s)C(s)}\,Y(s) \tag{4.184}$$

where the degrees of the polynomials involved are :

$$deg\,(F_k) = deg\,(P_dA) - 1$$

$$deg\,(G_k) = deg\,(C) - 1$$

$$deg\,(H_k) = k + deg\,(P_n) - deg\,(B^-P_d) - \rho$$

2- In equation (4.46), for the reference output, $y(t)$ should be replaced by $\phi^*(t)$.

The CGPC closed-loop setpoint response, in this case, given by the following equation and the corresponding feedback system is illustrated in figure 4.6.

$$Y(s) = \frac{gB(s)P_d(s)}{A(s)P_d(s) + L_c(s) + gL_0(s)} W(s) \tag{4.185}$$

where

$$L_c(s) = \underline{k}\ L\ \underline{S_L} \tag{4.186}$$

$$L\underline{S_L} = [\ 0\ \ L_1(s)\ \ L_2(s)\ \ .\ .\ .\ \ L_{N_y}(s)\ ]^T \tag{4.187}$$

$L_k(s)$ polynomial is obtained from the following identity

$$\frac{s^k P_n(s)B^+(s)}{P_d(s)A(s)} = H_k(s) + \frac{L_k(s)}{P_d(s)A(s)} \tag{4.188}$$

and $L_0(s)$ polynomial satisfies

$$\frac{P_n(s)B^+(s)}{P_d(s)A(s)} = H_0(s) + \frac{L_0(s)}{P_d(s)A(s)} \tag{4.189}$$

Note that both $L_c(s)$ and $L_0(s)$ are of degree $n + deg(P_d) - 1$. So the degree of the close-loop system is $n + deg(Pd)$

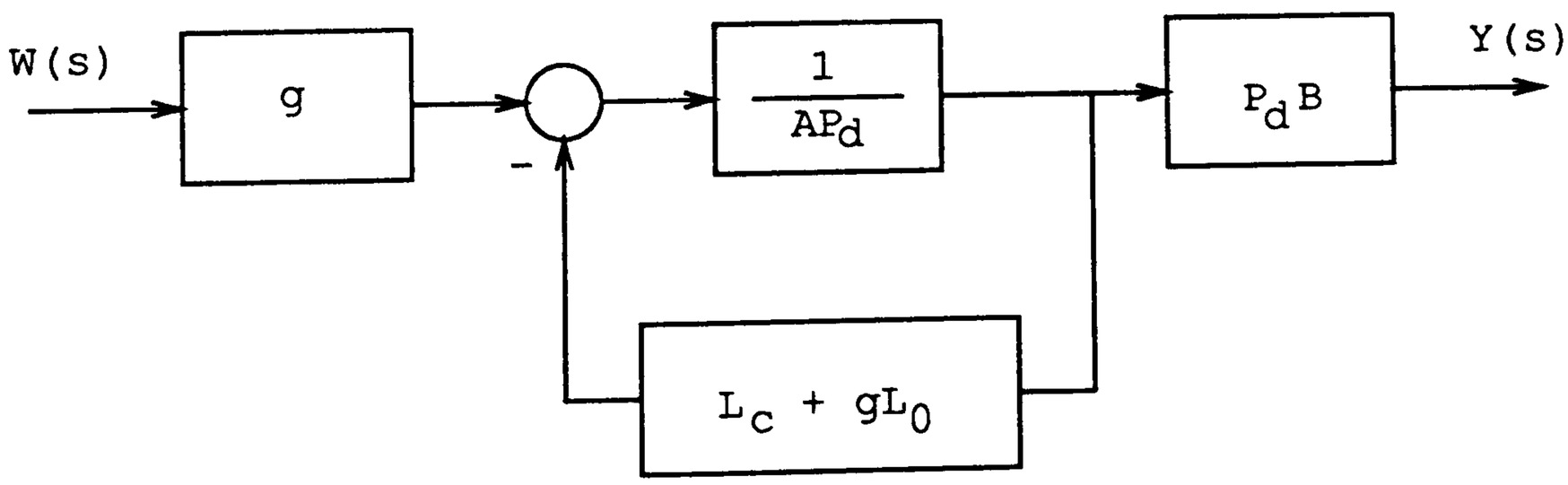


Figure 4.6 Equivalent feedback system of CGPC with auxiliary output

A special case of section 4.3.2 is also true for this case when $\lambda = 0$ and $N_u = N_y + deg(P_n) - deg(B^- P_d) - \rho$. Then the closed-loop system equation becomes

$$Y(s) = \frac{g\ B^-(s)\ P_d(s)}{(Z_c(s) + g)\ P_n(s)} W(s) \tag{4.190}$$

or

$$Y(s) = \frac{B^{-}(s)\, P_d(s)}{Z(s)\, P_n(s)}\, W(s) \tag{4.191}$$

where $Z_c(s)$ and $Z(s)$ are defined as before and $deg\,(Z) = \rho + deg\,(B^{-}P_d) - deg\,(P_n)$.

Model-following control

Model-following control is a special case of the above algorithm defined by

$$B^{-}(s) = 1$$

$$N_u = N_y + deg\,(P_n) - deg\,(P_d) - \rho$$

$$\lambda = 0$$

the corresponding closed-loop system is then

$$Y(s) = \frac{P_d(s)}{Z(s)\, P_n(s)}\, W(s) \tag{4.192}$$

So $\frac{P_n(s)}{P_d(s)}$ specifies the inverse model. The $Z(s)$ polynomial does not effect the output response significantly, because the CGPC control law tries to place the poles of $Z(s)$ faraway from the imaginary axis. The distance of these poles from the imaginary axis depends on the choice of T_2. When $T_2 \rightarrow 0$ the model-following relationship becomes exact as the poles of the $Z(s)$ polynomial moves towards $-\infty$. Note that degree of the $Z(s)$ polynomial is equal to the relative order of $\frac{P_n(s)B(s)}{P_d(s)A(s)}$. If $\frac{P_n(s)}{P_d(s)}$ is chosen such that $\frac{P_n(s)B(s)}{P_d(s)A(s)}$ has zero relative order then $Z(s)$ becomes unity, which means that irrespective of T_2 exact model-following is obtain.

When N_u is less than the above value, $\frac{P_n(s)}{P_d(s)}$ acts as an approximate inverse model. For large N_u the relationship is very close and it may be interpreted as detuned model-reference. For small N_u the effect of P_n/P_d is that: it reduces the overshoot at the output, giving better and smoother response.

Remark: 1: It is clear that the control signal will only be stable if $B(s)$ is stable.

Remark 2: Although R_n/R_d and P_n/P_d seem to have similar effects for small N_u, the use of P_n/P_d has greater effect than the use of R_n/R_d. This is easily understood by considering the closed-loop transfer function where the effect of R_n/R_d implicitly appears in the scalar gain g, whereas P_n/P_d explicitly appears in the transfer function itself.

Pole-placement control

Pole-placement control is a special case of the above algorithm defined by

$$B^{-}(s) = B(s)$$

$$P_d(s) = 1$$

$$N_u = N_y + deg(P_n) - deg(A)$$

$$\lambda = 0$$

the corresponding closed-loop system is then

$$Y(s) = \frac{B(s)}{Z(s)\,P_n(s)}\,W(s) \tag{4.193}$$

So there is no cancellation of the $B(s)$ polynomial in this case and a part of the closed-loop poles are given by the $P_n(s)$ polynomial. The $Z(s)$ polynomial does not effect the output response significantly, because CGPC control law tries to place the poles of $Z(s)$ faraway from the imaginary axis. The poles of the $Z(s)$ polynomial move towards $-\infty$ when $T_2 \rightarrow 0$, thus for $T_2 \rightarrow 0$ exact pole-placement is obtained. Note that $deg(Z) = n - deg(P_n)$ in this case, if $deg(P_n)$ is chosen equal to n then $Z(s)$ becomes unity, which means that exact pole-placement is obtained regardless of T_2. This pole-placement relationship becomes less accurate when N_u is reduced from $N_u = N_y + deg(P_n) - deg(A)$. However, for a large N_u the relationship is quite close and it may be interpreted as a detuned pole-placement.

Relation of CGPC with EBC

Consider the case where $T_1 = 0$, $T_2 \rightarrow 0$ then $\phi^*(t+T) \rightarrow \phi^*(t)$. So for very small T_2, without introducing a considerable error, we can write

$$\phi^*(t+T) = \phi^*(t) \tag{4.194}$$

and similarly

$$u_r^*(t,T) = u(t) \tag{4.195}$$

By assuming $R_n/R_d = 1$, then CGPC cost function can be written as

$$J = \int_0^{T_2} [\phi^*(t) - w(t)]^2 dT + \lambda \int_0^{T_2} u^2(t) dT \tag{4.196}$$

after evaluating the integral, the cost becomes

$$J = [\phi^*(t) - w(t)]^2 T_2 + \lambda\, u^2(t)\, T_2 \tag{4.197}$$

Recall that

$$\phi^*(t) = h_0\, u(t) + \phi^o(t) \tag{4.198}$$

then, the control law which minimize the above cost is given by

$$u(t) = \frac{h_0}{\lambda} [w(t) - \phi^*(t)] \tag{4.199}$$

Clearly, this is equivalent to the control law of EBC (eqn. 2.56) if the control weighting $Q(s)$ is chosen as $Q(s) = \dfrac{\lambda}{h_0}$. When $\lambda = 0$ the control law can be written as

$$w(t) = \phi^*(t) \tag{4.200}$$

Thus for $R_n/R_d = 1$ and $T_2 \to 0$ CGPC $\to$ EBC.

4.6.3. Dynamic Control Weighting

We mentioned before that λ will not be regarded as an important design parameter, as the effects of λ can be obtained by the other CGPC parameters without associated drawback of λ–$offset$. Therefore, instead of a constant control weighting λ, a dynamic control weighting $Q(s) = \dfrac{Q_n(s)}{Q_d(s)}$ can be considered which has two main advantages as discussed in chapter 2 :

1- it removes the λ–$offset$ if $Q(0) = 0$,

2- it may be used to improve robustness of the algorithm in the precence of neglected high frequency dynamics by choosing it to penalize the effects of these dynamics.

However, in this thesis we will not examine the effect of $Q(s)$ on the robustness of the CGPC; it is only covered for completeness.

There are two possible ways of incorporating dynamic control weighting into the CGPC algorithm:

1- during the design phase,

2- after the design phase.

The first one is obviously the natural way of incorporating the control weighting and it will be called natural dynamic control weighting (NDCW). The second one will be called forced dynamic control weighting (FDCW) as the control weighting is forced into the algorithm after the design phase.

Natural dynamic control weighting

In this case, a filtered predicted control $U_f(t,s) = Q(s)U_r(t,s)$ instead of $\sqrt{\lambda}\, u_r(t,T)$ is considered in the cost. Let $q(t)$ be the impulse response of $Q(s)$, in the time domain the filtered predicted control is then given by the following convolution integral.

$$u_f(t,T) = \int_0^T q(\tau)\, u_r(t,T-\tau)\, d\tau \tag{4.201}$$

Recall that

$$u_r(t,T) \approx u_r^*(t,T) = u(t) + u_1(t)T + u_2(t)\frac{T^2}{2!} + \ldots\ldots + u_{N_u}(t)\frac{T^{N_u}}{N_u!} \tag{4.202}$$

and consider the following approximate impulse response of $Q(s)$

$$q(T) \approx q^*(T) = q_0\delta(T) + q_1 + q_2T + q_3\frac{T^2}{2!} + \ldots\ldots + q_{N_y}\frac{T^{(N_y-1)}}{(N_y-1)!} \tag{4.203}$$

where q_k is the k^{th} markov parameter of $Q(s)$ and $\delta(T)$ is the unit impulse function. Then it can be shown that

$$u_f^*(t,T) = \underline{T}_{N_y} Q_m \underline{u} + \underline{T}_r Q_r \underline{u} \tag{4.204}$$

where $\underline{T}_{N_y}$, $\underline{u}$ and $\underline{T}_r$ (eqn. 4.43) are defined as before, Q_m and Q_r are given as follows.

$$Q_m = \begin{bmatrix} q_0 & 0 & 0 & . & . & 0 \\ q_1 & q_0 & 0 & . & . & 0 \\ q_2 & q_1 & q_0 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & q_0 \\ . & . & . & . & . & . \\ q_{N_y} & . & . & . & . & q_{(N_y - N_u)} \end{bmatrix} \tag{4.205}$$

$$Q_r = \begin{bmatrix} 0 & q_{N_y} & q_{(N_y - 1)} & . & . & q_{(N_y - N_u + 1)} \\ 0 & 0 & q_{N_y} & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & q_{(N_y - 1)} \\ 0 & 0 & 0 & . & . & q_{N_y} \end{bmatrix} \tag{4.206}$$

The second term of the right hand side of eqn. (4.204) will be ignored as it corresponds to $\frac{T^j}{j!}$ terms where $j > N_y$. So a N_y^{th} order truncated Maclaurin series approximation of $u_f(t,T)$ is obtained as

$$u_f^*(t,T) = \underline{T}_{N_y} Q_m \underline{u} \tag{4.207}$$

Minimization of the cost function (eqn. 4.55) with $\sqrt{\lambda}\, u_r^*(t,T)$replaced by $u_f^*(t,T)$ then results in the following control law

$$\underline{u} = K(\underline{w} - \underline{\bar{Y}}^o) \tag{4.208}$$

now K is given as follows

$$K = [\, H^T T_y H + Q_m^T T_q Q_m \,]^{-1} H^T T_y \tag{4.209}$$

where

$$T_q = \int_0^{T_2 - T_1} \underline{T}_{N_y}^T \underline{T}_{N_y}\, dT \tag{4.210}$$

Note that when $T_1 = 0$, $T_q = T_y$.

Forced dynamic control weighting

In this case we will consider the CGPC control law eqn. (4.75) which a control weighting can easily be incorporated as follows

$$Q(s)U(s) = W(s) - \Phi_e(s) \tag{4.211}$$

or

$$U(s) = \frac{1}{Q(s)} [\, W(s) - \Phi_e(s) \,] \tag{4.212}$$

where, $Q(s)$ can be used similarly as in EBC or GMV to detune the CGPC algorithm. Clearly, the effect of $Q(s)$ here will be different from the first case.

Remark: $Q(s)$ in general will have a simpler structure than system: a suitable choice for $Q(s)$ is suggested by Gawthrop (Gawthrop, 1987) as $Q(s) = \frac{\lambda s}{qs + 1}$. So eqn. (4.203) will be a good approximation of $q(T)$ over the prediction range as the N_y is chosen to approximate the system sufficiently well over the same range.

4.7. RELAY-CGPC

In chapter 3, a relay control strategy is described for the emulator-based control. The same strategy can also be employed in the CGPC without any difficulty. This is shown in figure 4.7 where $G_e(s)$ and $F_e(s)$ are the equivalent emulator polynomials (eqn. 4.77 and 4.78) and $R_f(s)$ is a transfer function with a unit relative degree. The aim again is to operate relay in the sliding mode. The transfer function $R_f(s)$, therefore, is needed to make the relative degree of the overall relay loop transfer function unity, as $deg(G_e) = deg(C)$. Recall that this is the necessary condition for the sliding mode to occur. As stated in chapter 3, a suitable choice of $R_f(s)$ is a first order low pass filter and it can also be replaced after the relay.

Suppose that relay is operating in the sliding mode, the relay input $e_f(t)$ will then force to stay in the vicinity of zero. This will approximately give

$$W(s) = \Phi_e(s) \tag{4.213}$$

which is control law of the CGPC (eqn. 4.75). As a result, the relay version of the CGPC given in figure 4.7 will be equivalent to the CGPC, if relay operates in the sliding mode. This will also be illustrated by simulation.

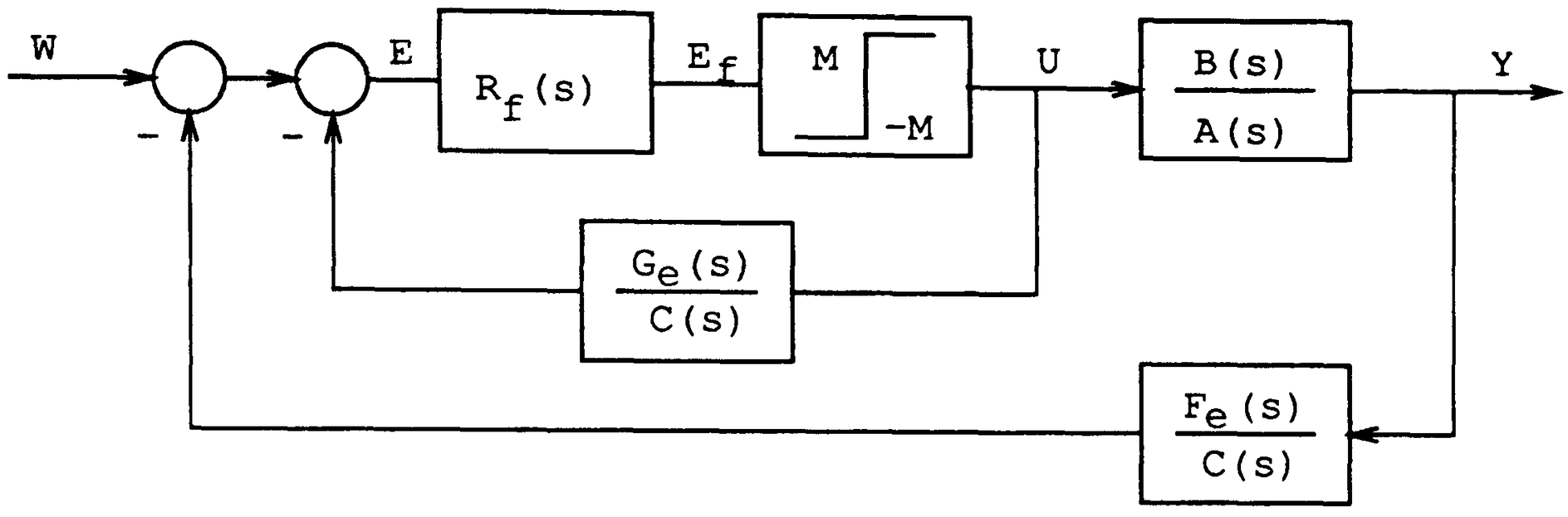


Figure 4.7 Block diagram of the Relay-CGPC

4.8. A SIMULATION STUDY

4.8.1. Non-Adaptive Simulations

CGPC design parameters (N_y, N_u, T_1, T_2, λ) and transfer functions (R_n/R_d and $P_n/B \cdot P_d$) will have the same effects in both adaptive and non-adaptive case. In this section a set of non-adaptive simulations were performed to illustrate the effects of these parameters and transfer functions, and the properties of non-adaptive CGPC. Simulations for the self-tuning CGPC are given in section 4.8.2. All of the simulations were performed using the MATLAB package program running on a Sun 3 workstation. Examples used in the adaptive and non-adaptive simulations are tabulated below.

Example 1: $\dfrac{B(s)}{A(s)} = \dfrac{1}{s(s^2+1)}$

$C(s) = 0.2s^2 + s + 1$

Example 2: $\dfrac{B(s)}{A(s)} = \dfrac{s+1}{s(s^2+1)}$

$$C(s) = 0.2s^2 + s + 1$$

Example 3: $\frac{B(s)}{A(s)} = \frac{1}{(s^2 + 1)(1.5s^2 + 1)}$

$$C(s) = (s+1)^3$$

Example 4: $\frac{B(s)}{A(s)} = \frac{-0.2s + 1}{s(s^2 + 1)}$

$$C(s) = 0.2s^2 + s + 1$$

Example 5: $\frac{B(s)}{A(s)} = \frac{e^{-s}}{s^2}$

$$C(s) = (s+1)^3$$

Example 6: $\frac{B(s)}{A(s)} = \frac{1}{s^2 + 1}$

$$C(s) = s + 1$$

.1.1. The effects of T_2 and N_u

Example 1 was simulated to illustrate the effects of T_2 and N_u. Since the effects of design ameters are of interest the design transfer functions R_n/R_d and $P_n/B \cdot P_d$ were set to unity. The trol weighting λ and T_1 were chosen to be zero and the sample interval to be 0.1sec.

Figure 4.8 illustrates the effect of T_2 where T_2 varies from 1 to 9 with an increment of 2. : predictor order N_y and the control order N_u were chosen to be 6 and 3 respectively, that is =6, N_u=3. In the figure, the upper graph shows the step responses and the lower graph shows closed-loop poles locus as T_2 varies. The fastest response in figure 4.8 corresponds to T_2=1. can be seen from the graphs the response becomes slower and poles move towards the origin T_2 increases.

If a model R_n/R_d is specified, the effect of T_2 may be different. This is because the cification given by the model and T_2 contradict with each other. In this case T_2 should be sen by considering the time constant of the model.

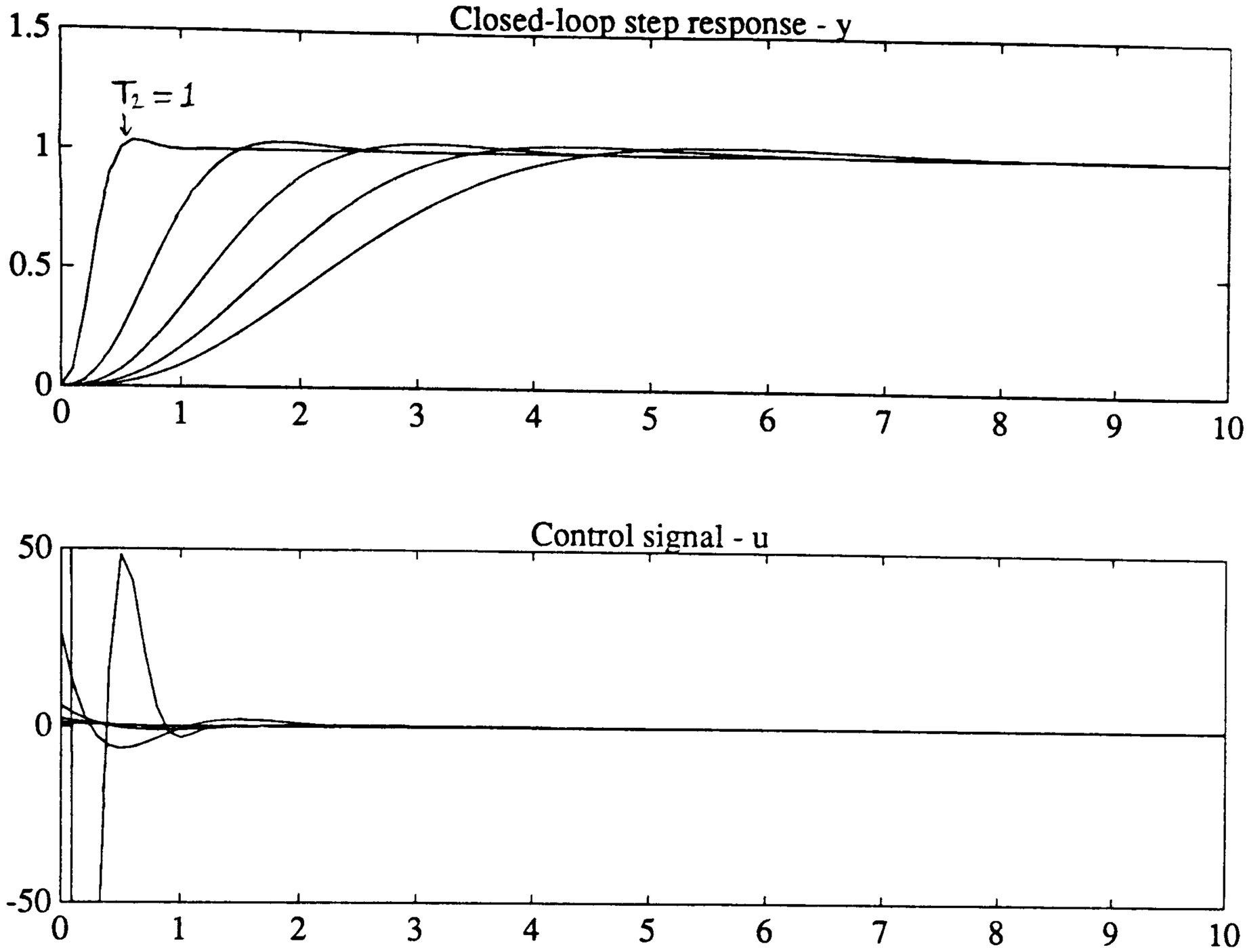


Figure 4.8 Illustration of the effect of T_2

The effect of N_u is shown in figure 4.9. For this example, N_y and T_2 were chosen to be 6 and 2 respectively and N_u is varied from 0 to 3. The upper graph shows the step responses and the lower one the poles locus as N_u varies. For N_u=0, the output response and the corresponding poles are marked on the graphs. The output response becomes faster and the poles move away from the imaginary axis as N_u increases. Note that N_u and T_2 have opposite effects that is, when N_u is increased response becomes faster whereas when T_2 is increased response becomes slower.

4.8.1.2. The effect of R_n/R_d

As mentioned in section 4.4.6, R_n/R_d has two functions: it can either be used as an approximate model or to penalize the overshoot. In this section examples 1 and 2 were simulated to illustrate these two functions of R_n/R_d. In the simulations R_n, P_n, B^*P_d were chosen to be 1, λ and T_1 to be 0, N_y to be 6 and the sample interval to be 0.1sec.

Figure 4.10 shows the simulation result of example 1 with $R_d = s+1$ (model = $\frac{1}{s+1}$), T_2=1 and N_u=3. In the figure, poles are the corresponding closed-loop poles, ym is the model output. Note that CGPC control law placed one of the poles at -1, the others quite away from the imaginary axis in order to approximate the model. Also note the impulsive behaviour of the control at the beginning, this is because a third order system was made to closely follow a first order model. In the example N_u was chosen to be $N_u = N_y - \rho$ in order to obtain the best approximation to the model. If N_u is chosen smaller this relationship becames less accurate.

Example 2 was simulated to illustrate the point discussed in the case 2 of section 4.3.2. For this purpose R_d and N_u were chosen to be $R_d=(0.5s+1)^2$ and $N_u = N_y - \rho = 4$. This choice of N_u will result in a cancellation law giving a second order closed-loop system. Since the model was chosen as second order, there is a value of T_2 which gives exact model matching. This value of T_2 was found as 5.625. The simulation result of example 2 and the corresponding closed-loop poles are shown in figure 4.11. Note that, as expected, one of the poles is equal to the closed-loop zero and the others are equal to the model poles. As a result exact model matching is obtained for $T_2 = 5.625$. Figure 4.12 shows, for the same example, the closed-loop poles locus as T_2 varies

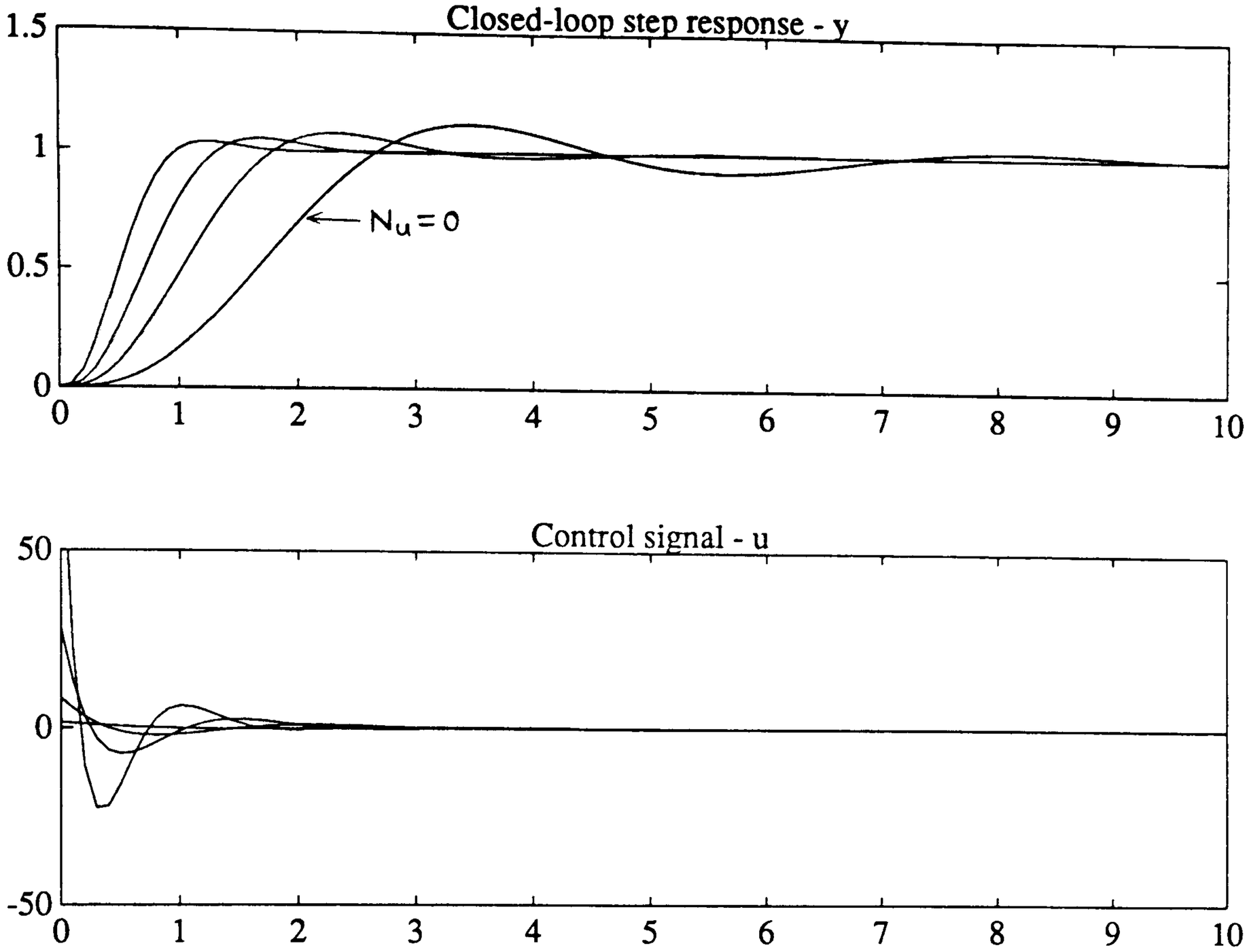


Figure 4.9 The effect of the control order N_u

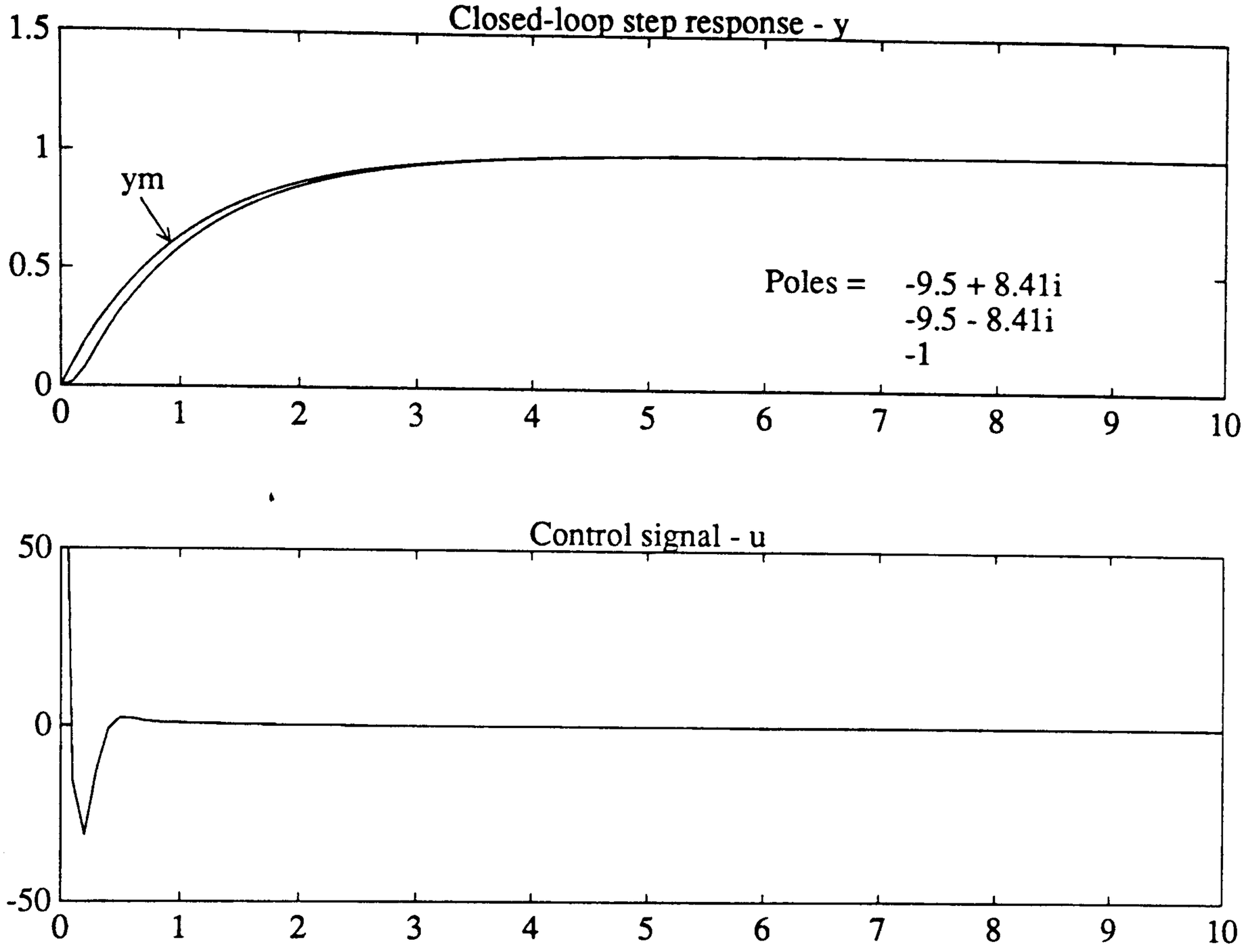


Figure 4.10 Model-following with R_n/R_d

from 3 to 10. The poles at the breakaway point, where $T_2 = 5.625$, corresponds to the model poles.

The use of R_n/R_d in order to penalize the overshoot is shown in figure 4.13. In this simulation example 1 was used with $N_u = 0$, $T_2 = 1$. The upper graph shows the step response when $R_d = 1$, the lower graph shows the step response when $R_d = 0.7s+1$. Note that the use of R_d, for this example, completely removed the overshoot.

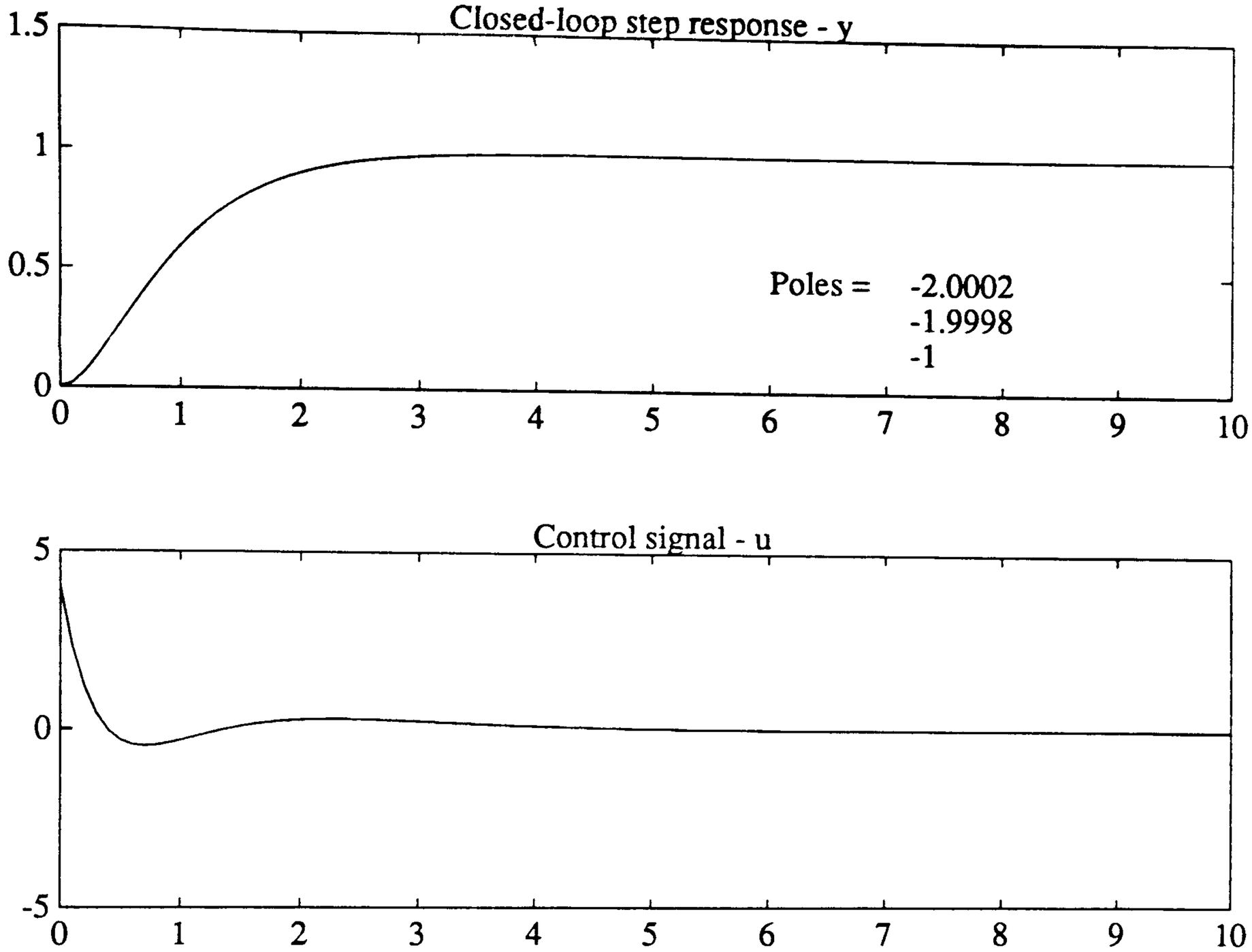


Figure 4.11 Exact model-following with R_n/R_d

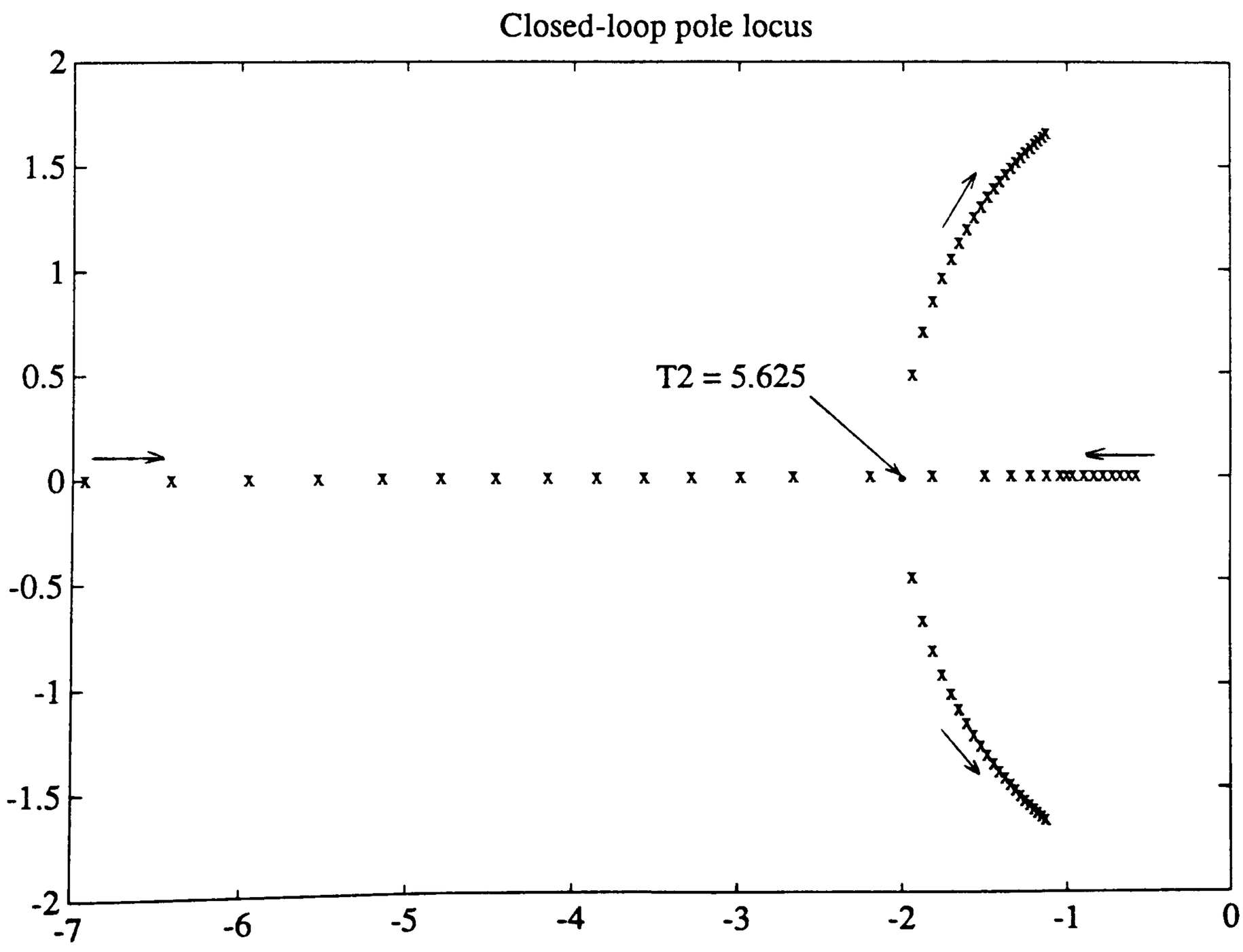


Figure 4.12 Closed-loop pole locus of example 2

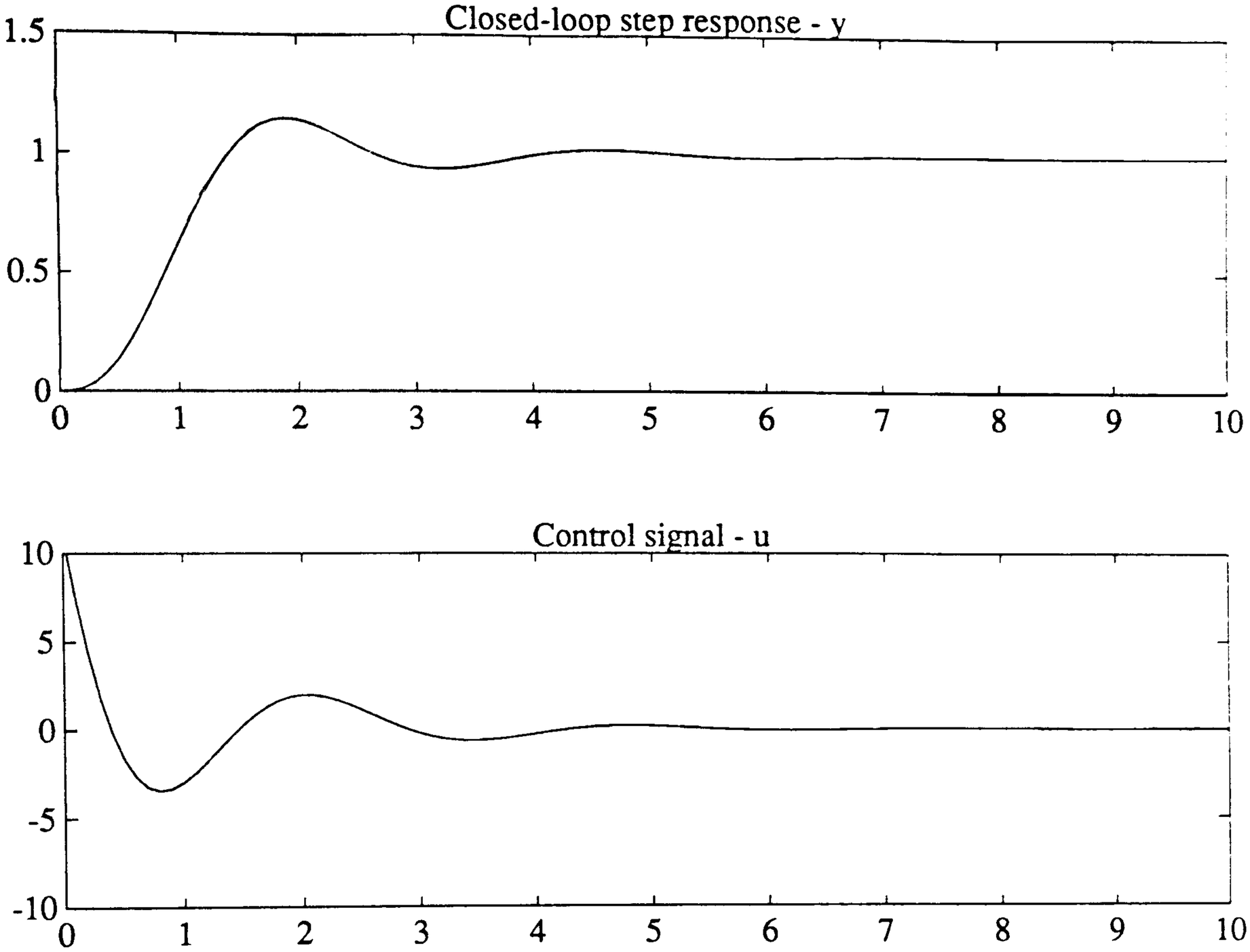


Figure 4.13 The use of R_n/R_d to penalize the overshoot

4.8.1.3. The effect of P_n/B^-P_d

As we discussed in section 4.6.2, P_n/B^-P_d can be used in three different ways: model-following control, pole-placement control and to penalize the overshoot. Here, pole-placement control will not be considered, hence $B^- = 1$. In the following simulations the sample interval was chosen to be 0.1sec, the control weighting λ and T_1 to be 0.

Example 1 was simulated to illustrate the model-following control with P_n/P_d. In the simulation R_n, R_d, P_d were chosen to be 1, N_y to be 6, T_2 to be 1. Figure 4.14 shows the simulation result for a first order model $P_n = s+1$. The corresponding closed-loop poles are also given with the figure. Note that, one of the poles is equal to the model pole, the others quite away from the imaginary axis (the distance of these poles from the imaginary axis depends on the value of T_2 for $T_2 \rightarrow 0$ these poles moves towards infinity giving the exact model matching). As seen from the figure the response is very close to the model output ym. However, forcing a third order system to match a first order model resulted in an impulsive control. In the simulation N_u was chosen to be $N_u = N_y - \rho + deg(P_n) = 4$ to obtain the best approximation. Making N_u smaller gives a less accurate relationship but removes the impulsive behaviour of the control, this may be interpreted as detuning the control. The same example was also simulated for a second order model $P_n = (0.5s+1)^2$ and the result is given in figure 4.15. In this simulation N_u was 5. Note that here two of the closed-loop poles are the model poles.

Example 3, a double oscillator, was simulated to show the use of P_n/P_d to penalize the overshoot. In the simulation the following choice of parameters were used:

$R_n/R_d = 1, \quad P_d = 1, \quad N_y = 8, \quad N_u = 0, \quad T_2 = 1$

The simulation result is given in figure 4.16. The upper graph shows the step response when $P_n = 1$, the lower graph shows the step response when $P_n = 0.7s+1$. As seen from the figure, the overshoot is much reduced. The same example was also simulated for $R_d = 0.7s+1$, $P_n = 1$ to give a comparison between R_n/R_d and P_n/P_d. The result is given in figure 4.17, which illustrates the greater effect of P_n/P_d.

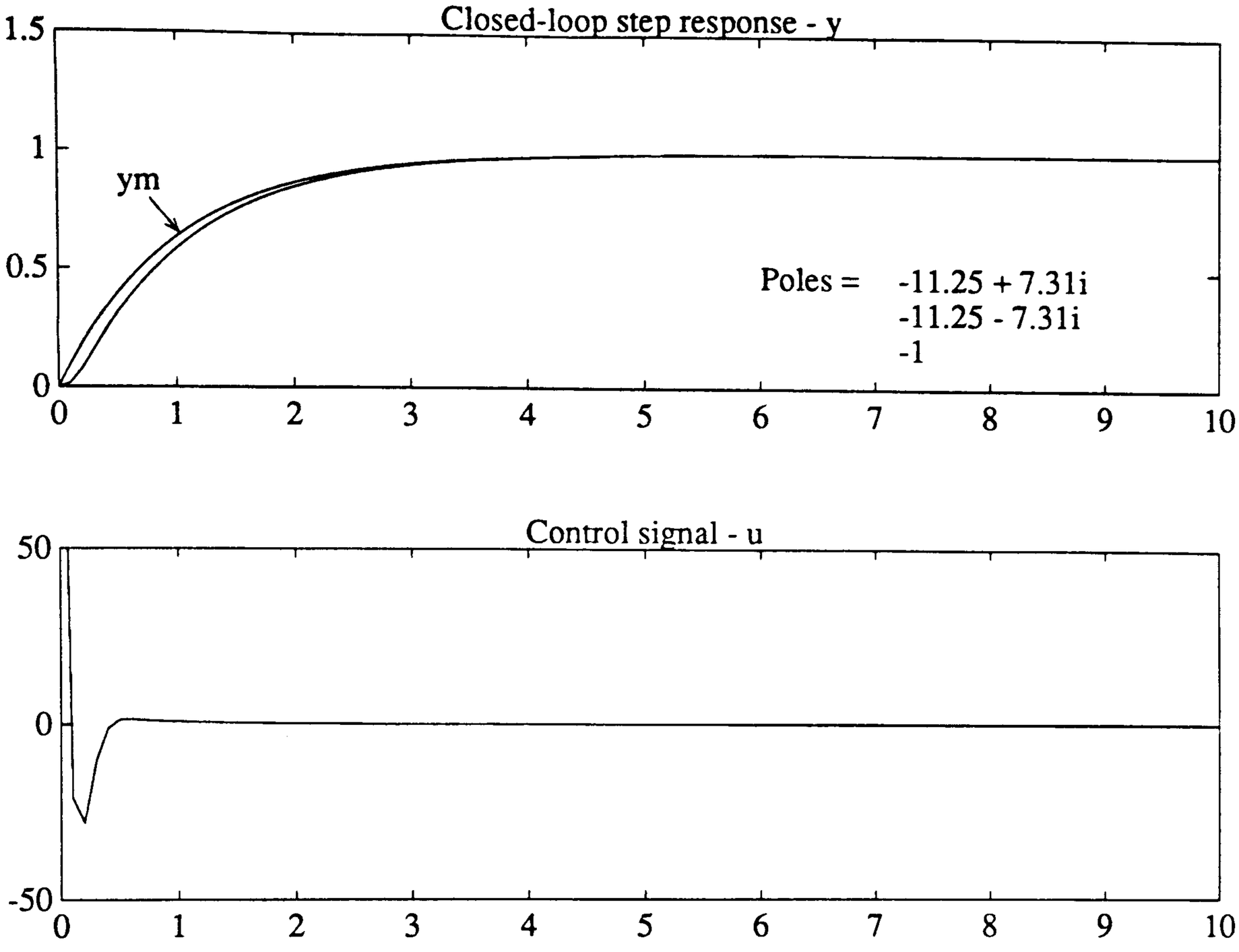


Figure 4.14 Model-following control with P_n/P_d (first order model)

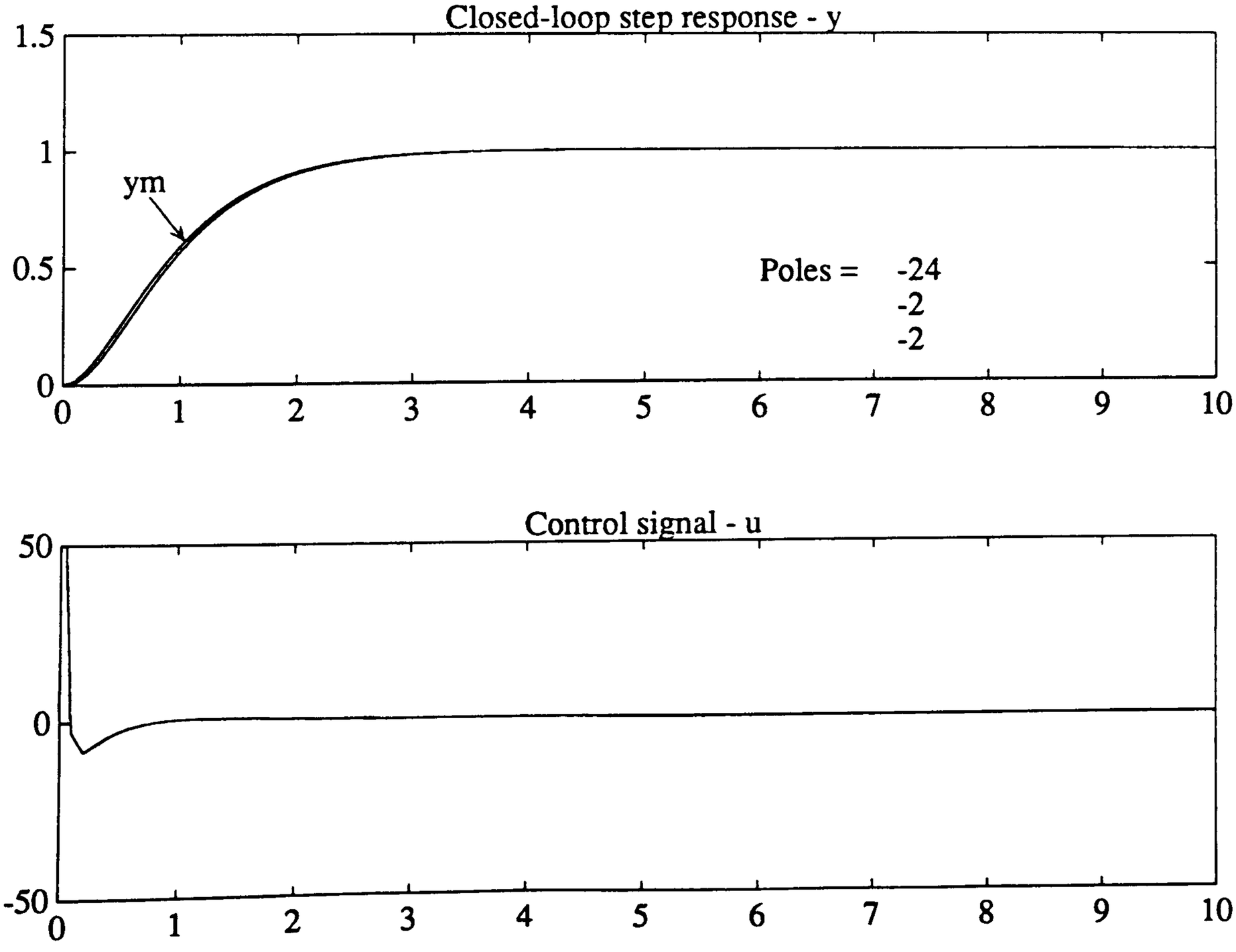


Figure 4.15 Model-following control with P_n/P_d (second order model)

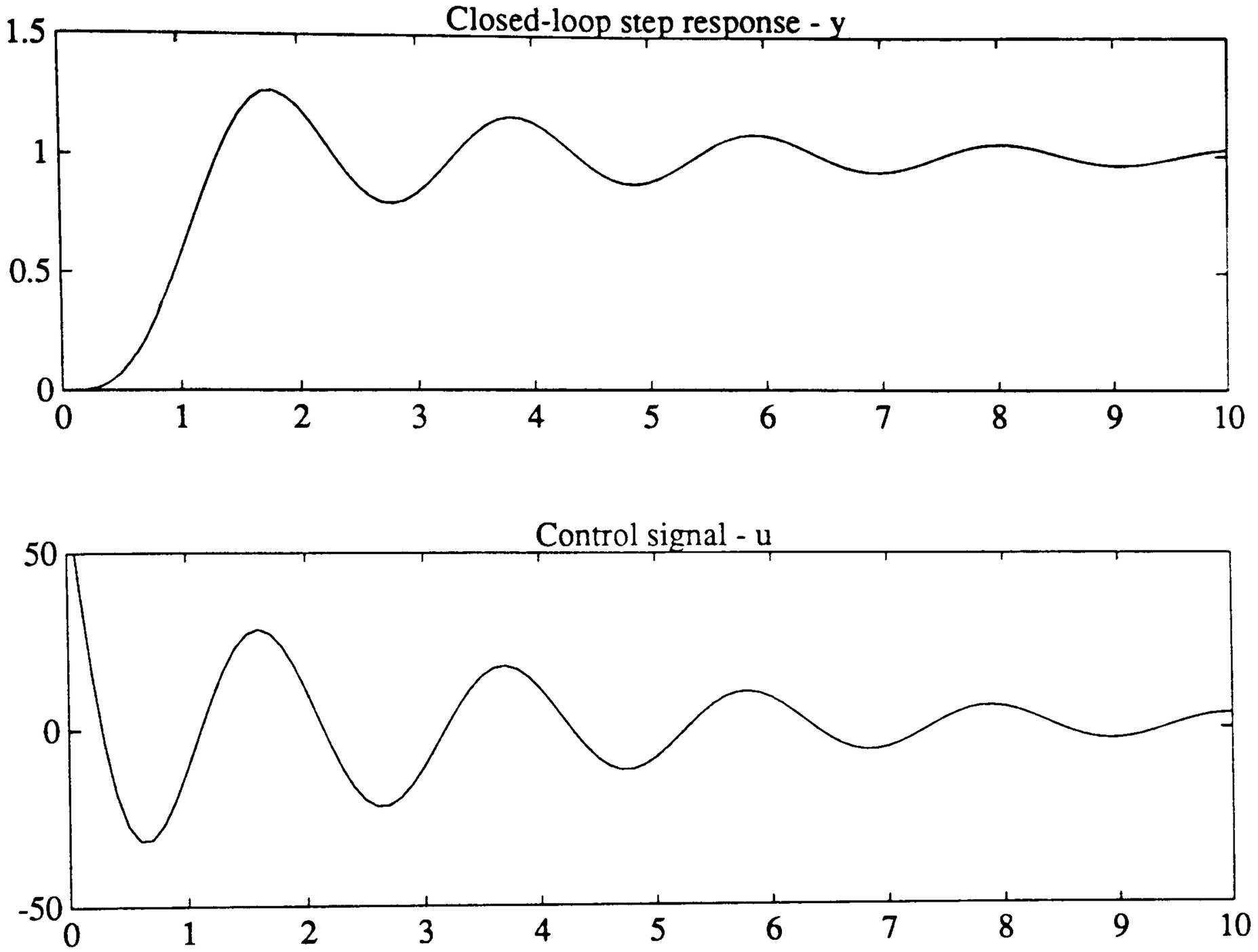


Figure 4.16 The use of P_n/P_d to penalize the overshoot

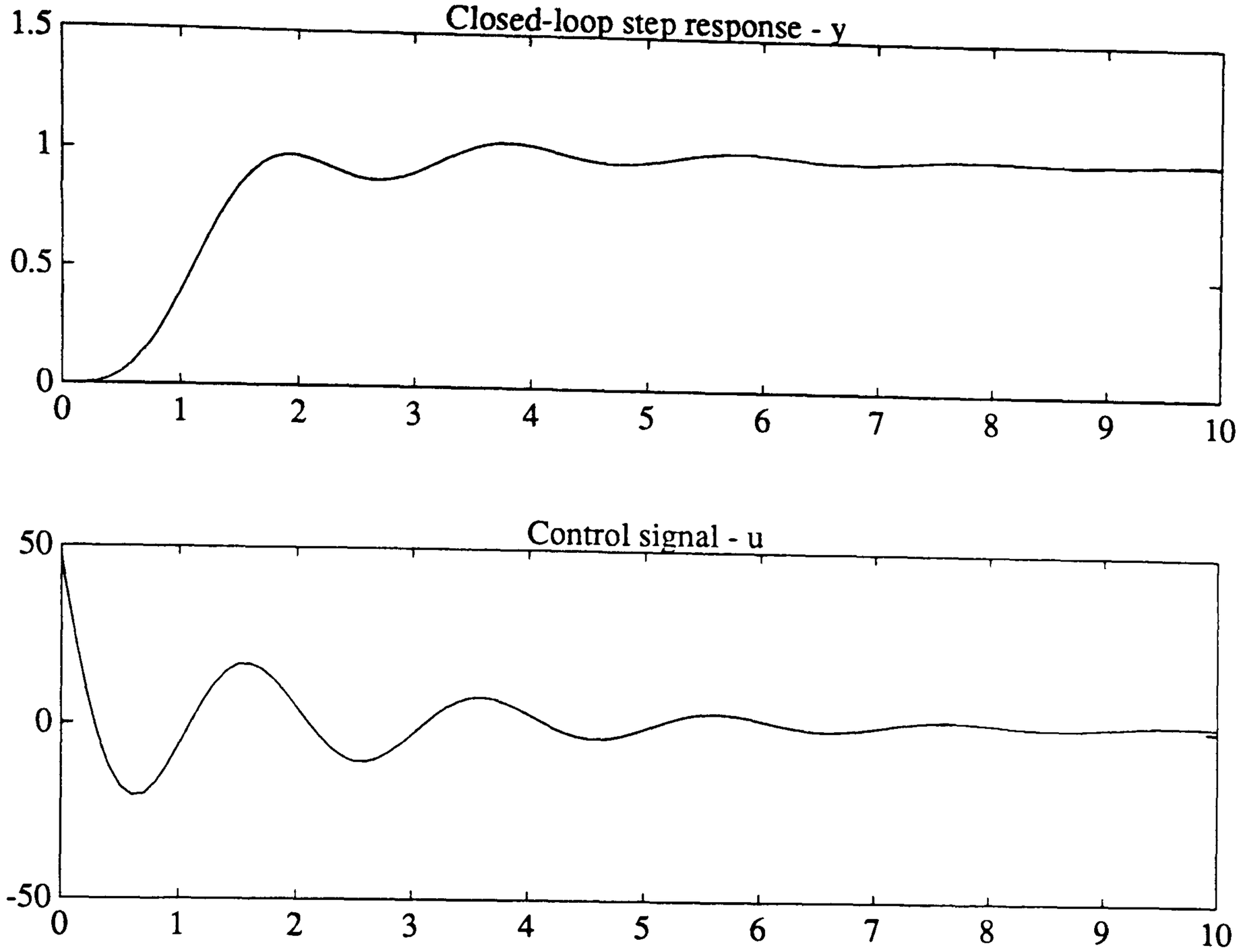


Figure 4.17 Example 3 with R_d=0.7s + 1, P_n = 1

4.8.1.4. Non-minimum phase systems

In contrast to the GMV algorithm, one of the properties of the GPC is that it is able to control non-minimum phase systems with zero control weighting. The CGPC also has this property. Example 4 was simulated to illustrate the control of non-minimum phase systems. In the simulation, sample interval was chosen to be 0.05sec and the following choice of parameters were used:

$R_n/R_d = 1,\ P_n/P_d = 1,\ N_y = 6,\ N_u = 0,\ T_1 = 0,\ T_2 = 3,\ \lambda = 0.$

The simulation result is given in figure 4.18. As seen from the figure output response is reasonable but, it can further be improved by increasing N_u and using one of the design transfer functions to remove the overshoot. This is illustrated in figure 4.19 where $N_u = 2$ and $R_d = 1.5s+1$. As seen from the figure the response is much improved. Note that the choice of $N_u = N_y - \rho$ is not possible (when λ=0) for the non-minimum phase systems since it removes the system zeros.

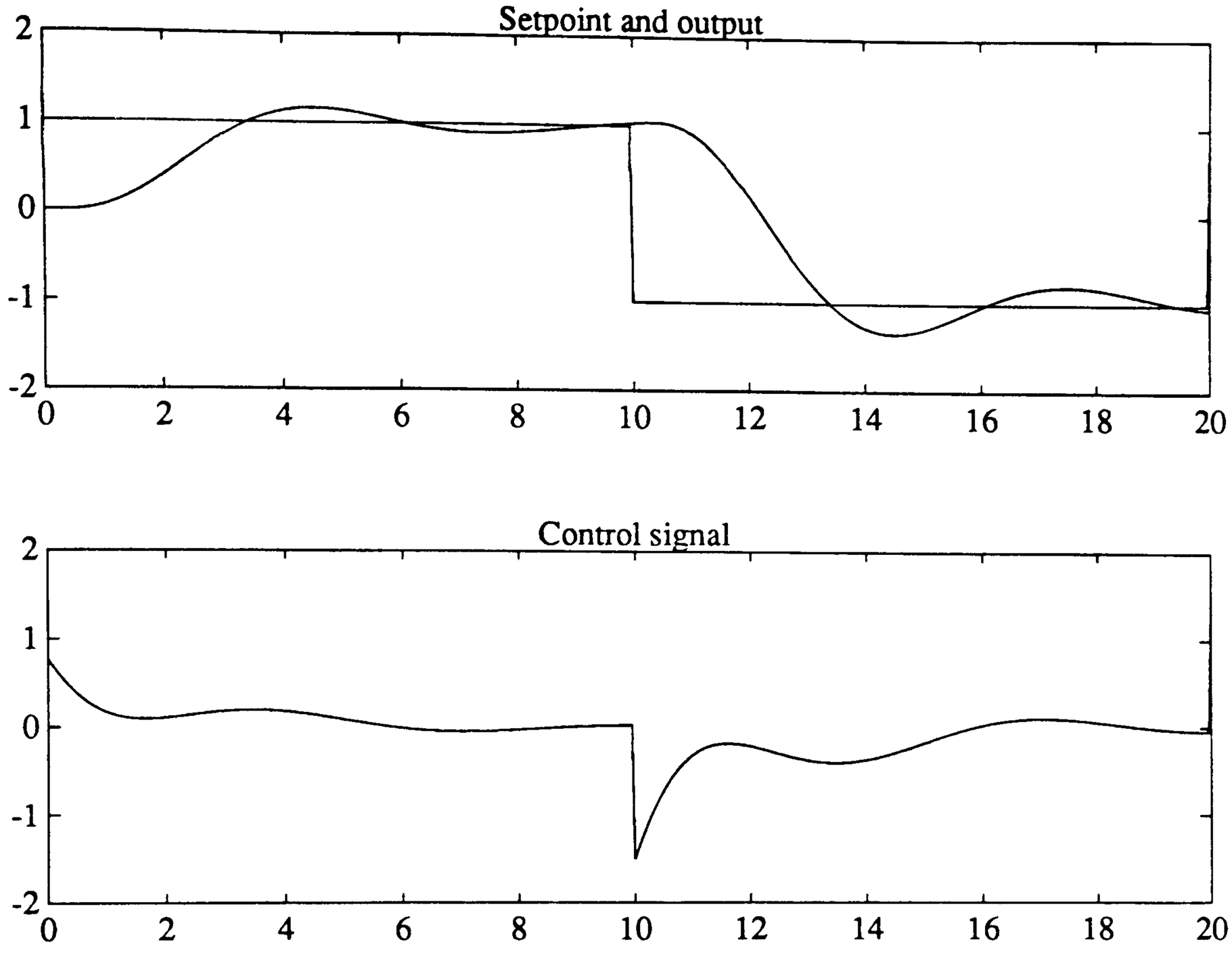


Figure 4.18 Control of a non-minimum phase system

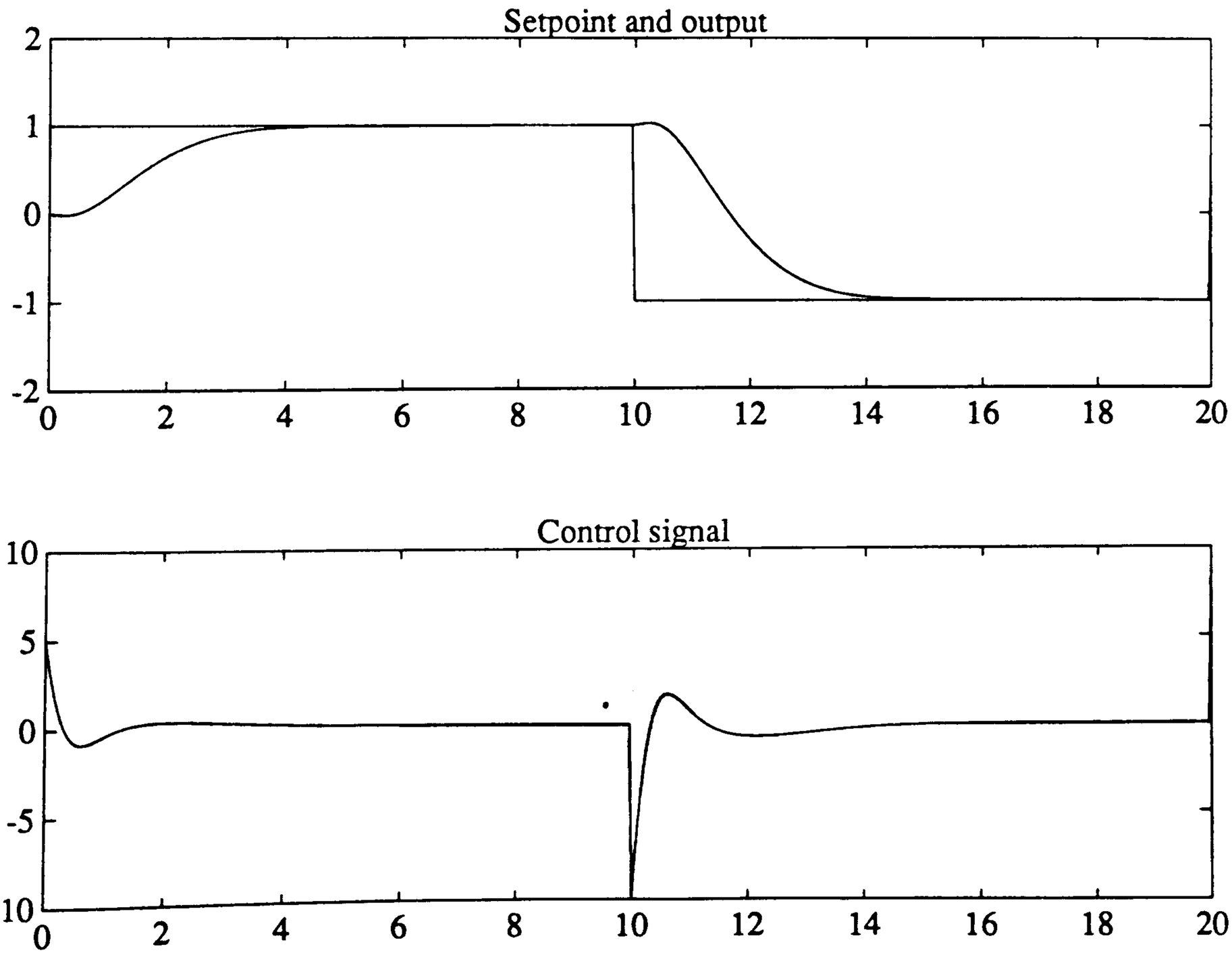


Figure 4.19 Example 4 with $N_u = 2$, $R_d = 1.5s + 1$

4.8.1.5. Time delay systems

Example 5 (a double integrator with unit time delay) was simulated to illustrate the ability of the CGPC to control time delay systems. A second order Pade approximation of the time delay was incorporated into A and B polynomials. CGPC design was based on the resulting 4^{th} order approximate system without time delay but, the simulation was performed by using the exact time delay. The CGPC parameters used in the simulation are as follows:

$R_n = 1,\ R_d = 0.6s + 1,\ P_n/P_d = 1,\ T_1 = 1,\ T_2 = 2, N_y = 20,\ N_u = 0,\ \lambda = 0$

In the simulation sample interval was chosen to be 0.05sec. The simulation result is shown in figure 4.20. As can be seen from the figure control performance is good but, this performance is obtained at the expense of increased N_y. However, this increase in N_y did not increase the computational burden significantly since $N_u = 0$. Although T_1 is chosen equal to the system time delay, it is also possible to obtain good control when $T_1 = 0$. However, in general, chosing T_1 equal to the system delay improves the control performance.

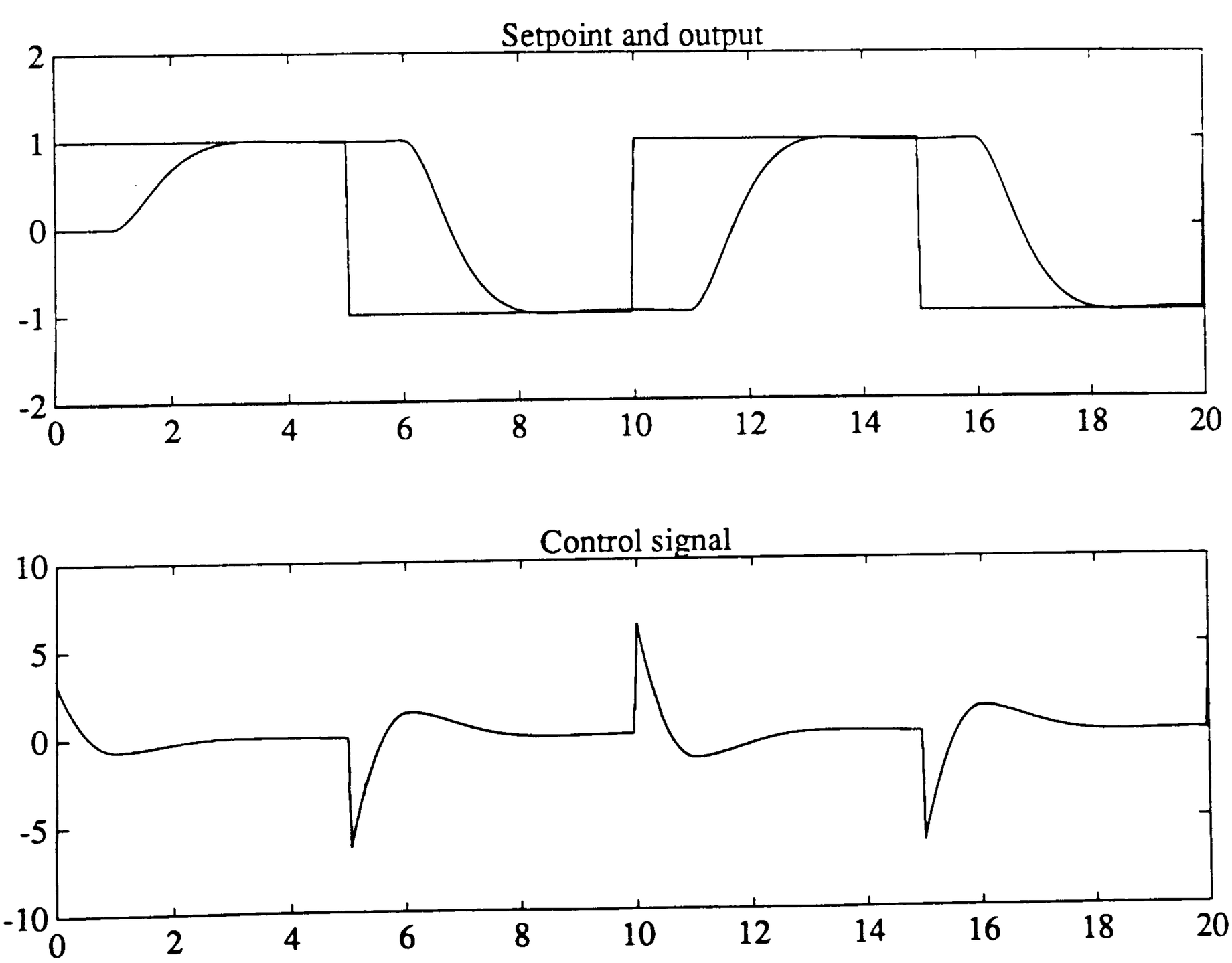


Figure 4.20 Control of a time delay system

4.8.1.6. LQ Control

As discused in section 5, the following setting of the CGPC parameters gives LQ control.

$$R_n/R_d = 1$$

$$P_n/B^{-}P_d = 1$$

$$N_u = N_y - \rho$$

$$N_y \rightarrow \infty$$

$$T_2 \rightarrow \infty$$

$$T_1 = 0$$

A double integrator (example 5 without time delay and $C(s) = s+1$) with the following choice of CGPC parameters was employed to illustrate this relationship.

$N_y = 12, \quad N_u = 10, \quad \lambda = 1$

The closed-loop LQ poles for the above choice of λ are found as: $-0.7071 \pm 0.7071i$

The closed-loop CGPC poles were calculated for $T_2 = 1$ to $T_2 = 10$ to see whether they converge to the LQ poles and the resulting poles are give below.

$T_2 = 1$	$-0.1546 \pm 0.6654i$
.	$-0.6021 \pm 0.7920i$
.	$-0.6941 \pm 0.6987i$
.	$-0.6963 \pm 0.7002i$
.	$-0.7030 \pm 0.7071i$
.	$-0.7065 \pm 0.7075i$
.	$-0.7071 \pm 0.7071i$
.	$-0.7071 \pm 0.7071i$
.	$-0.7071 \pm 0.7071i$
$T_2 = 10$	$-0.7070 \pm 0.7072i$

As seen from the CGPC poles for $T_2 > 6$ CGPC poles and LQ poles become the same to a four figure accuracy.

4.8.2. Adaptive Simulations

In the first part of the simulations (section 4.8.1), properties of the non-adaptive CGPC and, effects of the CGPC design parameters and transfer functions were illustrated. Since these properties will remain the same in the self-tuning case, in this section we did not reconsider them, instead we simulated several examples in order to illustrate properties of the self-tuning CGPC. In the simulations, parameter $\hat{a}_0$ of the highest power s term of $\hat{A}$ was fixed to 1 and thus one less A parameters were estimated. The following are common in the simulations.

1- All simulations start with a set of wrong parameters.

2- Estimator parameters: forgetting factor and initial inverse covariance are 0.2 and $0.00001I$ (where I is the unit matrix) respectively.

3- Sample interval is $0.05sec$.

4- Each figure consists of four graphs: the first one is the setpoint and output, the second one is the control signal, the third and fourth ones are the estimated A and B parameters.

4.8.2.1. An example

A non-minimum phase system (example 4) was simulated to give an example for the self-tuning CGPC. Simulation was performed with the same CGPC parameters as in the non-adaptive simulation corresponding to figure 4.19. Two B and three A parameters, same number as actual parameters, were estimated. Simulation result is shown in figure 4.21. As can be seen from the figure, parameters rapidly converge to their true values. Much more rapid convergence can be obtained if the initial inverse covariance is chosen zero. However, this results in large variations in the parameters at the beginning and this may give rise to initially large output. Note that, after convergence, the output is the same for adaptive and non-adaptive simulations as expected.

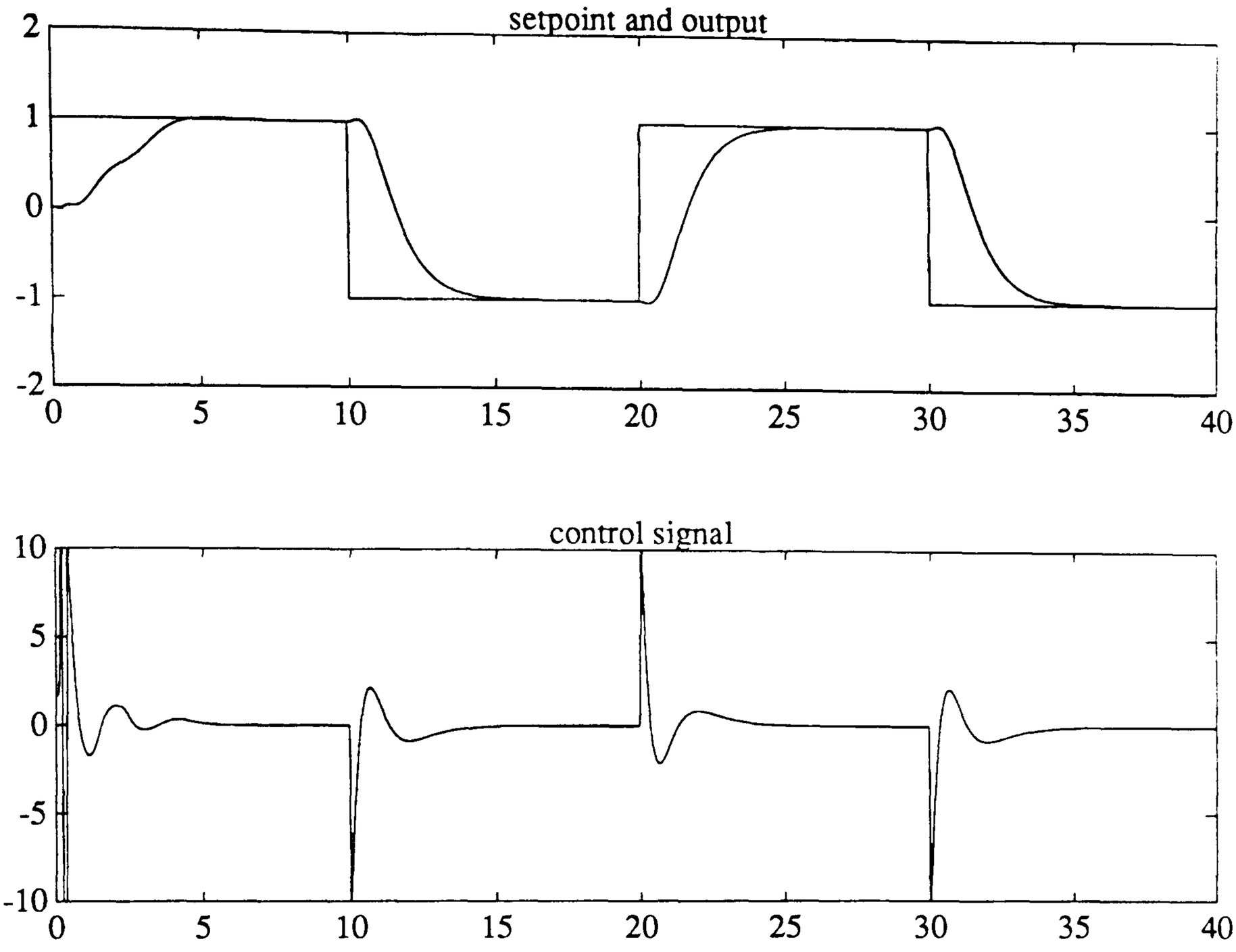


Figure 4.21 Self-tuning CGPC (example 4)

4.8.2.2. Effect of the noise

In practice, control systems are subject to disturbances of all kinds such as stepwise load disturbances or high frequency sensor or thermal noise etc. In self-tuning control, although the underlying design method may have a good setpoint response and disturbance rejection, these disturbances may give rise to wrong parameter estimates and thus result in a bad control performance or even an unstable system. In general this problem can be avoided if the signal to noise ratio is high.

In order to see the performance of the CGPC when noises are present, we simulated example 4 with a added random disturbance (Gaussian white noise with zero-mean and a standard deviation of 0.1) direct at the output. In the simulation exactly the same parameters as in section 4.8.2.1 were used. Simulation result is shown in figure 4.22. As can be seen from the figure despite the noise parameter estimates and the control performance is good.

4.8.2.3. Time delay systems

In non-adaptive CGPC, time delay systems are approximated by a higher order system without a delay. This is done by approximating the delay and incorporating $e^{-\lambda t}$ into system polynomials. The CGPC design is then based on the resulting approximate system. However, in adaptive case knowledge of the time delay is not available. One possible approach to this problem is to estimate the delay together with the system parameters (Besharati-Rad, 1988). Then the above design procedure can be applied. However, there are two drawbacks of this method: first it will increase the complexity of the estimation algorithm, second each time instant we need to approximate the delay and then incorporate into system polynomials to obtain the approximate system. The second approach, which removes these two drawbacks, is to let the estimator obtain the approximate system by specifying a higher order model to the estimator. This is the approach taken in the self-tuning CGPC. Note that this approach resembles the approach in discrete GPC where the time delay is taken into account by estimating a higher order B polynomial. Here we take the delay into account by estimating higher order B and A polynomials.

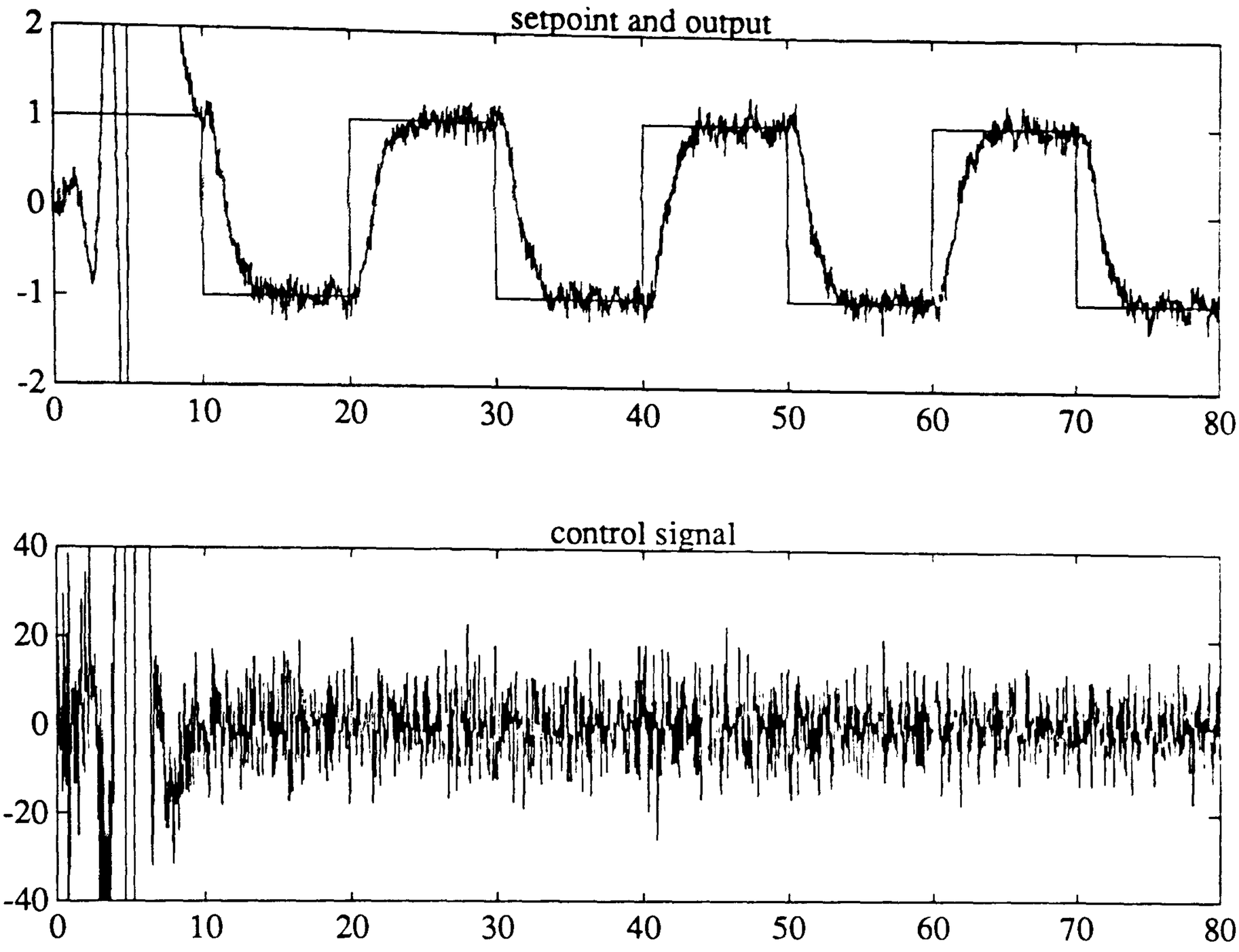


Figure 4.22 Example 4 with added random disturbances

Example 5 was simulated to illustrate the control of time delay systems by self-tuning CGPC. The CGPC design parameters and polynomials were chosen as in non-adaptive simulation (section 4.8.1.5) expect that R_d was chosen here $R_d = 0.8s + 1$ since $R_d = 0.6s + 1$ gave very slight overshoot. Four A and four B parameters were estimated. Simulation result is shown in figure 4.23. As can be seen from the figure control performance is very good. Note that estimated system is non-minimum phase as expected.

4.8.2.4. Over parameterization

There may be some situations where the parameters of the model are overspecified. This results in common factors in the estimated model. Pole-placement algorithms (Wellstead, 1979) fail under these circumstances. In section 4.3.3 the effect of common factors is examined and shown that, as for the GPC it is not a problem for the CGPC. This will also be illustrated by simulation. Example 6 was simulated for this purpose with the following CGPC parameters:

$R_n = 1, \quad R_d = 1.5s + 1, \quad P_n = 1, \quad P_d = 1, \quad T_1 = 0, \quad T_2 = 1, \quad N_y = 6, \quad N_u = 0, \quad \lambda = 0$

One more A and B (3 A and 2 B) parameters were estimated. Simulation result is shown in figure 4.24. As can be seen from the figure there is a common factor at s=0 in the estimated system model. This common factor did not effect the control performance. If the simulation is repeated with exact parameterization, it can be seen that the same output response is obtained.

4.8.2.5. Relay-CGPC

Finally, the simulation given in section 4.8.2.2 was repeated for the relay-CGPC. In the simulation, the filter $R_f(s)$ and the relay amplitude M were chosen to be $R_f(s) = 1/(0.5s+1)$ and $M = 5$ respectively. The simulation result is given in figure 4.25. Note that initially relay is not operating in the sliding mode. Relay may force to operate in the sliding mode earlier or from the beginning, if necessary, by choosing a larger relay amplitude. Note also that, relay is in the sliding mode following the first switching time when the setpoint changes. A larger relay amplitude will also reduce the first switching time.

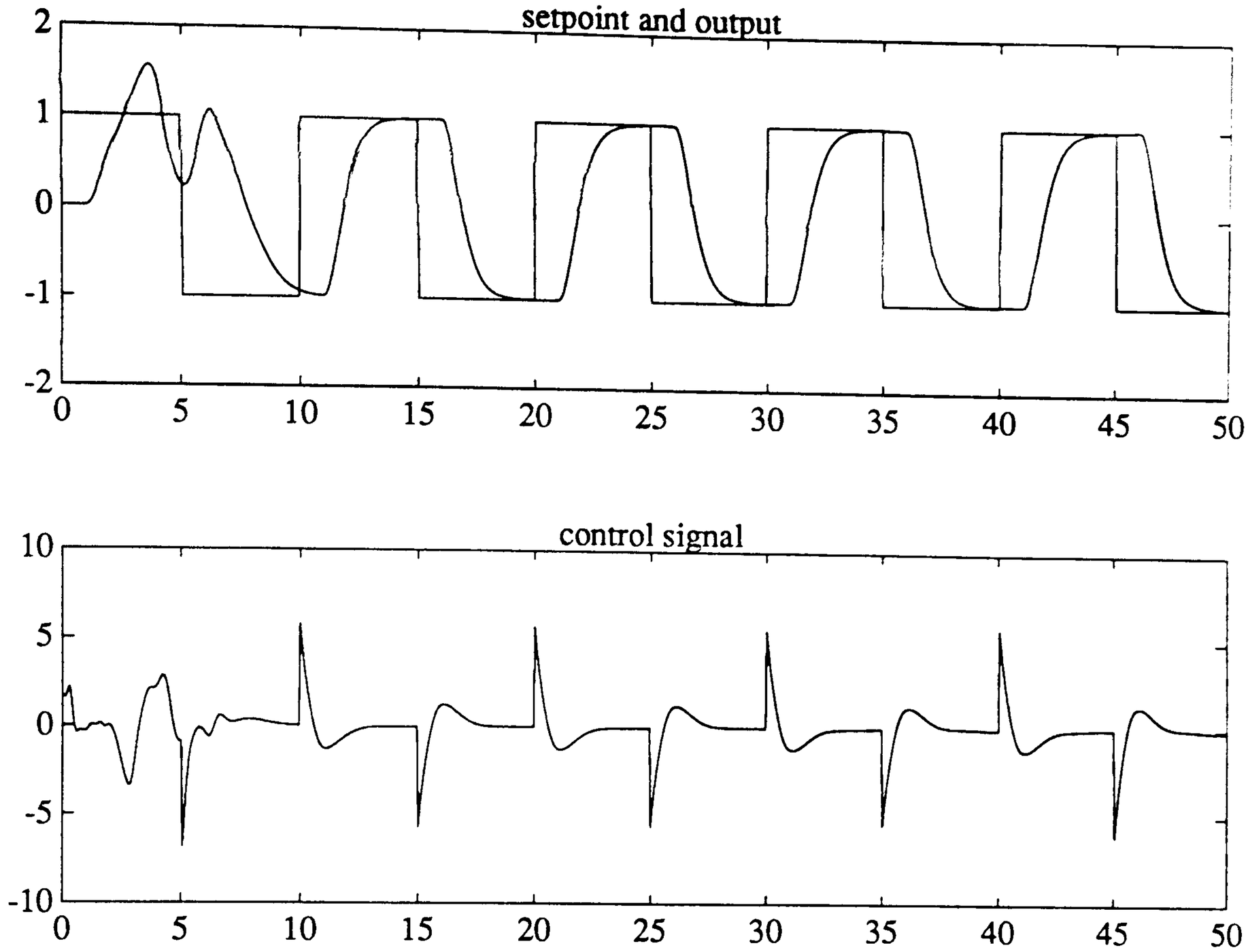


Figure 4.23 Self-tuning control of a time delay system

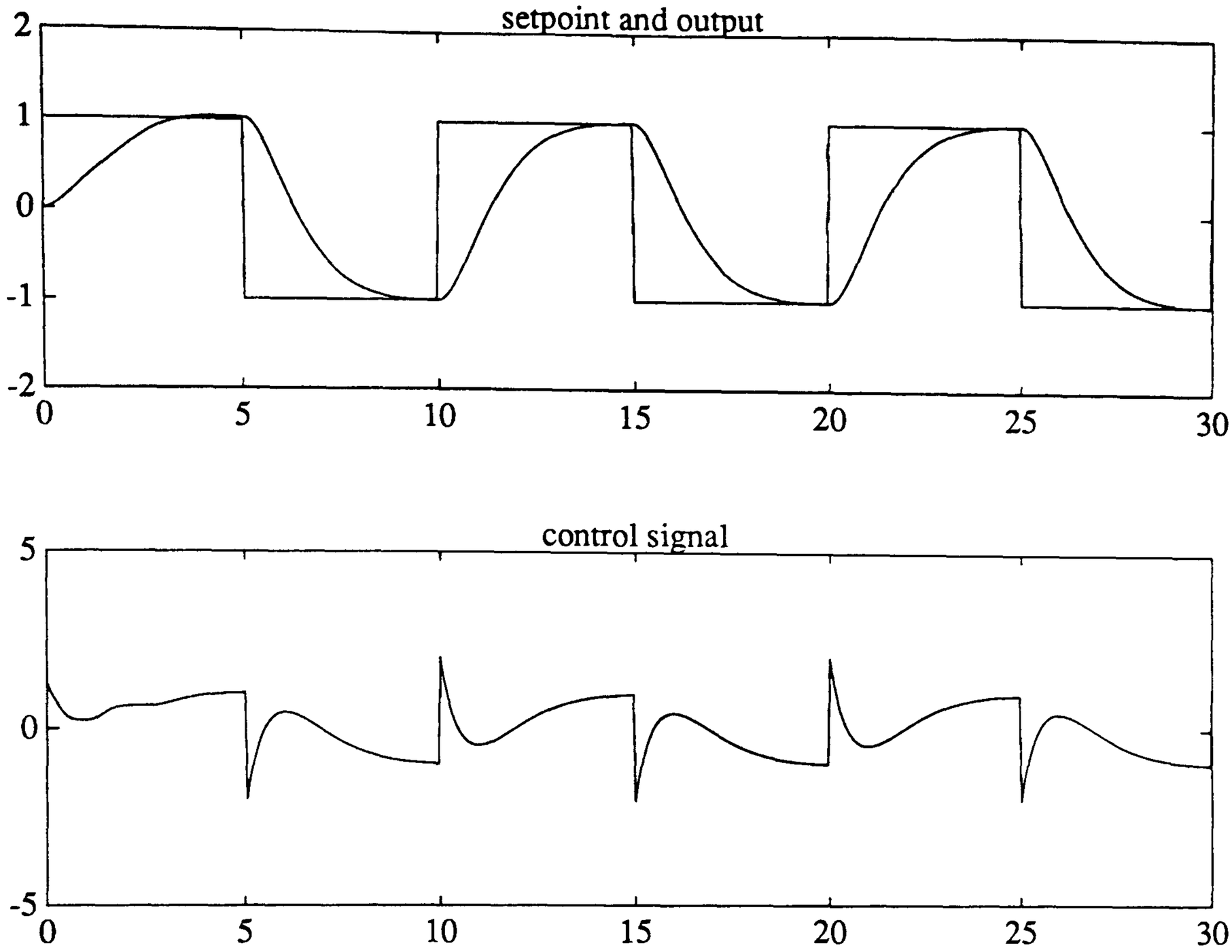


Figure 4.24 Self-tuning control with an over parameterized model

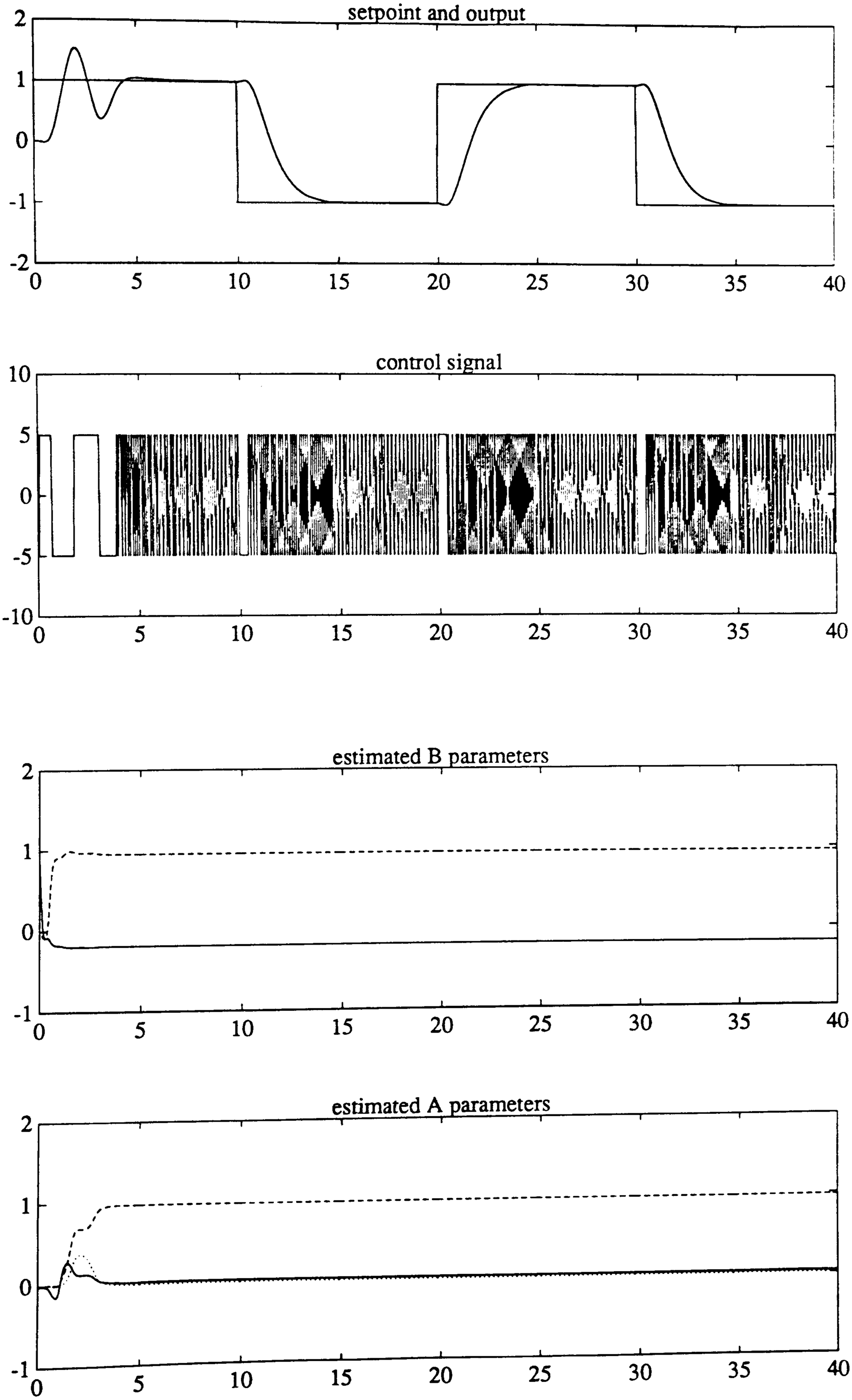


Figure 4.25 An example for the relay-CGPC

By comparing figures 4.25 and 4.21, one can see that the output responses in both cases are very similar (after the initial transients). This demonstrates the equivalence of the CGPC and the relay-CGPC when the relay operates in the sliding mode. It is also interesting to note that the parameter convergence are very similar in both cases.

4.9. CONCLUSIONS

In this chapter the CGPC, a continuous-time version of the GPC developed by Clarke et al, is presented and analysed. The basic CGPC algorithm appears to be a very useful design method. A large variety of systems can be controlled reasonably well by only adjusting the prediction horizons T_1 and T_2 keeping the control order N_u as zero but, for the more complex systems (at the same time higher order, open-loop unstable, non-minimum phase) an increased value of N_u is needed. In general, it is advisable to keep N_u small in order not to increase the computational burden, instead use reference-model R_n/R_d to adjust the transients. Apart from this R_n/R_d can also be used to obtain model-following type control (sometimes exact model-following) with a large N_u. The relationship of CGPC with LQ control is also examined and it is shown that LQ control can be considered as a subalgorithm of the CGPC. In addition, it is shown that time-delay systems can be controlled in a similar way to GPC by increasing the system order in order to accommodate a rational approximation to the delay.

The basic CGPC algorithm is also enhanced by using an auxiliary output instead of output itself. This made it possible to consider the model-reference and pole-placement control (and also their detuned versions) in the CGPC frame work. It is also shown that emulator based control (EBC) of Gawthrop can be considered as a subalgorithm of the CGPC. In addition, the case of systems with canceling pole/zero pairs is considered and shown to cause no difficulty unlike pole-placement control. An important feature is that control weighting is not necessary for the control of non-minimum phase systems unlike GMV or EBC.

In these respects, the CGPC is superficially similar to its discrete-time counterpart, but there are important differences in the way in which output prediction and control weighting is accomplished. An example of this is that, whereas the GPC constrains the predicted control difference to

be zero after N_u samples, the CGPC constrains the predicted control so that the derivatives of order greater than N_u are zero.

A relay version of the CGPC based on the ideas of chapter 3 is also described and shown to be equivalent to the CGPC, if relay operates in the sliding mode.

In brief, CGPC appears to be a very suitable control algorithm for the self-tuning control applications for a large variety of systems.

CHAPTER 5

MULTIVARIABLE CONTINUOUS-TIME GENERALIZED PREDICTIVE CONTROL

5.1. INTRODUCTION

One approach to the control of multivariable systems is to reduce the multivariable control problem to a set of scalar control problems so that the scalar control techniques can directly be applied. A straightforward way of doing this is simply to design a scalar controller for each loop by treating the interactions from the other loops as disturbances. The work of Gawthrop (Gawthrop, 1985) can be given as an example of this type of self-tuning multivariable control in which a feedforward term is also estimated to reduce the effect of interactions. This technique was also applied to a number of real plants and some encouraging results were reported (Nomikos, 1988). An alternative way is to decompose the multivariable system into a set of scalar systems by some means and design a controller for each. The method proposed by Cloud and Kouvaritakis (Cloud, 1988) is an interesting example of this type of self-tuning multivariable control where the multivariable system is decoupled by spectral decomposition and GPC is used as a control technique for each scalar subsystem.

A more common approach to the control of multivariable systems is to design a multivariable controller. Most of the extension of the scalar self-tuning algorithms to the multivariable case are based on this approach. A quick review of some of the important ones are as follows:

Minimum variance controller of Astrom and Wittenmark (Astrom, 1973) was first extended to the multivariable case by Borisson (Borisson, 1979). The extension of the GMV controller of

Clarke and Gawthrop (Clarke, 1975) then followed this (Koivo, 1980, Keviczky, 1981). In these methods the time delay is assumed to be the same in all channels, which prevented them from being considered as the complete generalization of the corresponding scalar techniques. Goodwin, Ramadge and Caines (Goodwin, 1980) took this generalization effort one step further by assuming different delays at different outputs in their design. Generalization of the MV type controller appeared to be completed by Goodwin and Long (Goodwin, 1980) using the so-called interactor matrix (Wolovich, 1976) which provided a proper generalization of the time delay to the discrete multivariable systems.† It then became apparent that the earlier results of Borisson (1979), Koivo (1980), and Goodwin, Ramadge and Caines (1980) were special cases in which the interactor matrix was taken to be a scaled identity matrix and a diagonal matrix respectively. Implementation of the Goodwin and Long's algorithm (1980) requires complete knowledge of the interactor matrix, which is a severe practical limitation. This limitation was partially overcome by the method suggested by Dugard, Goodwin and Souza (Dugard, 1983) and Elliott and Wolovich (Elliott, 1984) where the off-diagonal elements of the interactor matrix are estimated together with the system parameters. Singh and Narendra (Singh, 1984) suggested an alternative method in which a suitably chosen precompensator is applied to the system in order to transform the interactor to a diagonal matrix. As a further improvement, an extended horizon approach (Ydstie, 1984) was adopted by Dugard, Goodwin and Xianya (Dugard, 1984). This reduced the prior knowledge about the interactor matrix to an upper bound on the maximum forward shift in the interactor matrix.

There have also been attempts of generalizing other types of scalar self-tuning algorithms to the multivariable case. A multivariable pole-placement self-tuner was outlined by Prager and Wellstead (Prager, 1980). Elliott and Wolovich (Elliott, 1982) proposed a model matching algorithm based on the pole-placement approach in which the open-loop zeros are canceled out by specifying them as part of the closed-loop poles and a setpoint filter is used to meet the model matching requirement. Elliott and Wolovich (Elliott, 1984) also proposed a direct multivariable pole-

† Interactor matrix in the continuous-time corresponds to the generalization of the relative order.

placement method to avoid the transformation from the estimated left MFD (matrix fraction description) to the right MFD, however this considerably increased the number of parameters to be estimated making it practically unrealistic. A further method based on a pole-placement approach was proposed again by Elliott and Wolovich (Elliott, 1984) with an attempt to unify minimum variance, pole-placement and model matching strategies. In addition to these algorithms, a number of state-space multivariable self-tuners have also appeared in the literature (Hesketh, 1982, Shieh, 1983, Bezanson, 1984, Shieh, 1989).

As expected, the above multivariable strategies suffer from the same limitations as their scalar counterpart: MV and GMV methods are very sensitive to the variations in the time delay structure of the plant; MV is only applicable to the minimum phase systems; pole-placement methods require exact knowledge of the controllability indexes and an upper bound on the observability index to assure a unique and physically realisable solution to the multivariable diophantine equation. These limitations seem to be overcome by LRPC methods in the scalar case. In particular, the GPC provided considerable improvement over the GMV and pole-placement strategies in these aspects. Motivated by this success, a multivariable generalization of GPC was outlined by Mohtadi, Shah and Clarke (Mohtadi, 1986) and Mohtadi (Mohtadi, 1987). As mentioned previously, continuous-time approach has some advantages over the discrete-time approach and we believe that it will be worthwhile to generalize the scalar CGPC concept to the multivariable case. Therefore this chapter is devoted to the development and analysis of a multivariable CGPC (MCGPC).

This chapter is organized as follows. In section 2 a multivariable generalization of the scalar CGPC is presented. In section 3 the resulting MCGPC closed-loop system is analysed in some detail and conditions for decoupling and exact model-following are established. Some illustrative simulations are given in section 4 and section 5 concludes the chapter.

5.2. DEVELOPMENT OF THE MCGPC ALGORITHM

5.2.1. System Description

The development of the algorithm will be based on the following linear multivariable system model †

$$\mathbf{A}(s)\underline{Y}(s) = \mathbf{B}(s)\underline{U}(s) + \mathbf{C}(s)\underline{V}(s) \tag{5.1}$$

where $\underline{Y}(s)$, $\underline{U}(s)$ and $\underline{V}(s)$ are $p\times1$ output, $m\times1$ input and $p\times1$ disturbance vectors respectively; $\mathbf{B}(s)$ is a $p\times m$ polynomial matrix; $\mathbf{A}(s)$ and $\mathbf{C}(s)$ are $p\times p$ *diagonal* polynomial matrices. Elements of $\mathbf{C}(s)$ are all stable polynomials with a degree of one less or equal to that of corresponding element of $\mathbf{A}(s)$. $C(s)$ polynomials are chosen by the designer and can be interpreted as *observer* polynomials as in the scalar case. The time delay problem will be tackled in a way similarly as in chapter 4, that is it will be assumed that polynomial matrices $\mathbf{A}(s)$ and $\mathbf{B}(s)$ include a rational approximation of any time delay term between any input output pair when they exist. For simplicity it will also assumed that multivariable system is strictly proper.

Note that the assumption that $\mathbf{A}(s)$ and $\mathbf{C}(s)$ are diagonal very much simplifies the predictor design as it reduces multivariable identities involved in the design to a set of scalar identities and as diagonal matrices commute.

The above system model corresponds to a left MFD (matrix fraction description) (Kailath, 1980) of a multivariable system having transfer matrix $\mathbf{T}(s)$, that is

$$\mathbf{T}(s) = \mathbf{A}(s)^{-1}\mathbf{B}(s) \tag{5.2}$$

$\mathbf{T}(s)$ can also be written in a right MFD form

$$\mathbf{T}(s) = \mathbf{B}_R(s)\mathbf{A}_R(s)^{-1} \tag{5.3}$$

This later form will be employed in the evaluation of a closed-loop equation for the MCGPC in section 3. There are many MFD (left or right) representations of a given multivariable system,

† Throughout this chapter matrices will be denoted by the bold letters and vectors will be underlined.

however in our case it is unique as $\mathbf{A}(s)$ matrix is assumed to be diagonal.

5.2.2. Output Prediction

As in chapter 4, the Maclaurin series expansion of the output will be used for the prediction. Each output of the multivariable system will be considered separately as this enables us to choose different predictor orders for different outputs. This is important from the computational point of view as the predictor order needed for each output can be considerably different. Now consider the truncated Maclaurin expansion of the i^{th} output

$$\hat{y}_i(t+T) = y_i(t) + \sum_{k=1}^{N_{y_i}} y_{ik}(t)\frac{T^k}{k!} \quad ; \quad i = 1,2,\cdots,p \tag{5.4}$$

where

$$y_{ik}(t) = \frac{d^k \hat{y}_i(t+T)}{dT^k} \ (at\ T=0) = \frac{d^k y_i(t)}{dt^k} \tag{5.5}$$

N_{y_i} = predictor order for the i^{th} output †

Eqn. (5.4) can be rewritten in a matrix form

$$\hat{y}_i(t+T) = \underline{T}_{N_{y_i}} \underline{Y}_i \tag{5.6}$$

where

$$\underline{T}_{N_{y_i}} = [\,1 \ \ T \ \ \frac{T^2}{2!} \ \ldots\ \frac{T^{N_{y_i}}}{N_{y_i}!}\,] \tag{5.7}$$

$$\underline{Y}_i = [\,y_i(t) \ \ y_{i1}(t) \ \ y_{i2}(t) \ \ldots\ y_{iN_{y_i}}(t)\,]^T \tag{5.8}$$

Using eqn. (5.6) the predicted future output vector can be written as

$$\underline{\hat{y}}(t+T) = \mathbf{T}_{N_y} \underline{Y} \tag{5.9}$$

---

† The rules given in the scalar case (section 4.4.3) for the choice of the predictor order also apply to the choice of N_{y_i} and will not be repeated here.

where

$$\mathbf{T}_{N_y} = \begin{bmatrix} \underline{T}_{N_{y_1}} & \underline{0} & \cdots & \underline{0} \\ \underline{0} & \underline{T}_{N_{y_2}} & \cdots & \underline{0} \\ \vdots & \vdots & \cdots & \vdots \\ \underline{0} & \underline{0} & \cdots & \underline{T}_{N_{y_p}} \end{bmatrix} \tag{5.10}$$

and

$$\underline{Y} = \begin{bmatrix} \underline{Y}_1 \\ \underline{Y}_2 \\ \vdots \\ \underline{Y}_p \end{bmatrix} \tag{5.11}$$

In eqn (5.10) $\underline{0}$ denotes an appropriate dimension zero row vector. The outputs derivatives in eqn. (5.9) should be emulated because of the reasons given earlier. Consider the i^{th} output

$$Y_i(s) = \frac{1}{A_i(s)} \sum_{j=1}^{m} B_{ij}(s)U_j(s) + \frac{C_i(s)}{A_i(s)} V_i(s) \tag{5.12}$$

If $y_{ik}^*(t)$ represents the emulated value of the k^{th} derivative of the i^{th} output $y_{ik}(t)$, then it is given in the Laplace domain as

$$Y_{ik}^*(s) = \frac{E_{ik}(s)}{C_i(s)} \sum_{j=1}^{m} B_{ij}(s)U_j(s) + \frac{F_{ik}(s)}{C_i(s)} Y_i(s) \tag{5.13}$$

where the polynomials $E_{ik}(s)$ and $F_{ik}(s)$ satisfies the following identity

$$\frac{s^k C_i(s)}{A_i(s)} = E_{ik}(s) + \frac{F_{ik}(s)}{A_i(s)} \tag{5.14}$$

The terms $\frac{E_{ik}(s)B_{ij}(s)}{C_i(s)}$ in eqn. (5.13) are not proper transfer functions for $k > \rho_{ij}$ $(j=1,2, \cdots ,m)$ where ρ_{ij} is the relative order of the ij^{th} element of the system transfer matrix. This term can be decomposed into two parts by using polynomial long division

$$\frac{E_{ik}(s)B_{ij}(s)}{C_i(s)} = H_{ijk}(s) + \frac{G_{ijk}(s)}{C_i(s)} \quad ; \quad j=1,2, \cdots ,m \tag{5.15}$$

where $\frac{G_{ijk}(s)}{C_i(s)}$ is a strictly proper transfer function and $H_{ijk}(s)$ is the remainder polynomial. Then

the emulator eqn. (5.13) becomes

$$Y_{ik}^{*}(s) = \sum_{j=1}^{m} H_{ijk}(s)U_j(s) + \frac{1}{C_i(s)}\sum_{j=1}^{m} G_{ijk}(s)U_j(s) + \frac{F_{ik}(s)}{C_i(s)}\, Y_i(s) \tag{5.16}$$

assuming that the degree of $C_i(s)$ is one less than that of $A_i(s)$, the degree of the polynomials involved in eqn. (5.16) are:

$$deg\,(H_{ijk}(s)) = k - \rho_{ij}$$

$$deg\,(G_{ijk}(s)) = n_i - 2$$

$$deg\,(F_{ik}(s)) = n_i - 1$$

$$n_i = deg\,(A_i(s))$$

If $deg\,(C_i(s)) = deg\,(A_i(s))$ then the only difference will be $deg\,(G_{ijk}(s)) = n_i - 1$. Notice that the emulator eqn. (5.16) has two parts: one part can be realized by using proper transfer function, the other part can not. Hence it can be rewritten as

$$Y_{ik}^{*}(s) = \sum_{j=1}^{m} H_{ijk}(s)U_j(s) + Y_{ik}^{o}(s) \tag{5.17}$$

where the realisable part $Y_{ik}^{o}(s)$ is given by

$$Y_{ik}^{o}(s) = \frac{1}{C_i(s)}\sum_{j=1}^{m} G_{ijk}(s)U_j(s) + \frac{F_{ik}(s)}{C_i(s)}\, Y_i(s) \tag{5.18}$$

Choosing a *control order* for each input, eqn. (5.17) can be written in the time domain as follows

$$y_{ik}^{*}(t) = \sum_{j=1}^{m} \underline{H}_{ijk}\,\underline{u}_j + y_{ik}^{o}(t) \tag{5.19}$$

where $\underline{H}_{ijk}$ is a row vector containing coefficients of the polynomial $H_{ijk}(s)$ and $\underline{u}_j$ is a column vector of the derivatives of the j^{th} input

$$\underline{u}_j = [\, u_j(t) \;\; u_{j1}(t) \;\; u_{j2}(t) \ldots . \; u_{jN_{u_j}}(t) \,]^T \tag{5.20}$$

$$u_{jk}(t) = \frac{d^k u_j(t)}{dt^k} \tag{5.21}$$

where N_{u_j} is the *control order* for the j^{th} input $u_j(t)$. Eqn. (5.19) can be rearrange in a matrix form

$$y_{ik}^{*}(t) = \underline{H_{ik}}\,\underline{u} + y_{ik}^{o}(t) \tag{5.22}$$

where $\underline{H_{ik}}$ is a row vector of the row vectors $\underline{H_{ijk}}$

$$\underline{H_{ik}} = [\, \underline{H_{i1k}} \;\; \underline{H_{i2k}} \;\; \ldots . \; \underline{H_{imk}} \,] \tag{5.23}$$

and $\underline{u}$ is a column vector of the column vectors $\underline{u_j}$

$$\underline{u} = \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ . \\ . \\ \underline{u}_m \end{bmatrix} \tag{5.24}$$

Using eqn. (5.22), the column vector

$$\underline{Y_i^{*}} = [\, y_i(t) \;\; y_{i1}^{*}(t) \;\; y_{i2}^{*}(t) \ldots . \; y_{iN_{y_i}}^{*}(t) \,]^T \tag{5.25}$$

can be written as

$$\underline{Y_i^{*}} = \mathbf{H}_i\,\underline{u} + \underline{Y_i^{o}} \tag{5.26}$$

where

$$\underline{Y_i^{o}} = [\, y_i(t) \;\; y_{i1}^{o}(t) \;\; y_{i2}^{o}(t) \ldots . \; y_{iN_{y_i}}^{o}(t) \,]^T \tag{5.27}$$

and

$$\mathbf{H}_i = \begin{bmatrix} \underline{0} \\ \underline{H_{i1}} \\ \underline{H_{i2}} \\ . \\ . \\ . \\ \underline{H_{iN_{y_i}}} \end{bmatrix} \tag{5.28}$$

where $\underline{0}$ is an appropriate dimension zero row vector. Further using eqn. (5.26) the vector of the emulated outputs derivatives

$$\underline{Y^{*}} = \begin{bmatrix} \underline{Y_1^{*}} \\ \underline{Y_2^{*}} \\ . \\ . \\ \underline{Y_p^{*}} \end{bmatrix} \tag{5.29}$$

can be written in an explicit form

$$\underline{Y}^* = \mathbf{H}\underline{u} + \underline{Y}^o \tag{5.30}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ . \\ . \\ \mathbf{H}_p \end{bmatrix} \quad ; \quad \underline{Y}^o = \begin{bmatrix} \underline{Y}_1^o \\ \underline{Y}_2^o \\ . \\ . \\ \underline{Y}_p^o \end{bmatrix} \tag{5.31}$$

Finally, replacing $\underline{Y}$ in eqn. (5.9) by its emulated value $\underline{Y}^*$ (eqn. 5.30), the output predictor for the multivariable system is obtained as fallows.

$$\underline{y}^*(t+T) = \mathbf{T}_{N_y}\mathbf{H}\,\underline{u} + \mathbf{T}_{N_y}\,\underline{Y}^o \tag{5.32}$$

It is important to note that the matrix **H** is in the following form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & . & . & . & \mathbf{H}_{1m} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & . & . & . & \mathbf{H}_{2m} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \mathbf{H}_{p1} & \mathbf{H}_{p2} & . & . & . & \mathbf{H}_{pm} \end{bmatrix} \quad ; \quad (p+\sum_{i=1}^{p} N_{y_i}) \times (m+\sum_{j=1}^{m} N_{u_j}) \tag{5.33}$$

where each subblock matrix $\mathbf{H}_{ij}$, which corresponds to the ij^{th} element of the system transfer matrix, is in the form as in the scalar case, that is

$$\mathbf{H}_{ij} = \begin{bmatrix} 0 & 0 & 0 & . & . & . & 0 \\ h_{ij1} & 0 & 0 & . & . & . & . \\ h_{ij2} & h_{ij1} & 0 & . & . & . & . \\ h_{ij3} & h_{ij2} & h_{ij1} & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & h_{ij1} \\ . & . & . & . & . & . & . \\ h_{ijN_{y_i}} & . & . & . & . & . & h_{ij(N_{y_i}-N_{u_j})} \end{bmatrix} \quad ; \quad (N_{y_i}+1) \times (N_{u_j}+1) \tag{5.34}$$

where h_{ijk} is the k^{th} Markov parameter of the ij^{th} transfer function $\frac{B_{ij}(s)}{A_i(s)}$ of the system transfer matrix.

As in the scalar case (see section 4.2.2.3), the second term $\mathbf{T}_{N_y}\underline{Y}^o$ in eqn. (5.32) corresponds approximately to the response of the system to the initial conditions at time t, and the first term

$\mathbf{T}_{N_y}\mathbf{H}\,\underline{u}$ corresponds approximately to the response to a future input $\underline{u}(t+T)$, that is

$$\mathbf{T}_{N_y}\mathbf{H}\,\underline{u} \approx \mathbf{h}(T)*\underline{u}(t+T) \tag{5.35}$$

where $\mathbf{h}(T)$ is the impulse response matrix of the system and * denotes the convolution integral with respect to T. Clearly the approximation accuracy depends on the parameters T, N_{y_i} $(i=1,2,\ \cdots\ ,p)$ and N_{u_j} $(j=1,2,\ \cdots\ ,m)$.

Remark: Note that in the above development a different control order is used for each input. This is important as it enables us to adjust control inputs independently, to some extent. The effect of the control orders here are similar to the scalar case, that is the larger the control orders the more active the control signals and vice versa. They have also effects on interactions: in general, larger control orders result in less interaction.

5.2.3. Reference Outputs

In the scalar case it was seen that a reference output approach is useful as it provides us with an extra design transfer function which can be used either to reduce the overshoot or to obtain a model-following type control depending on the choice of the other design parameters. This approach is also useful for the same reasons in the multivariable case. As one may expect, here a reference output for each of the system outputs needs to be considered which means that we simply have p scalar cases. Let for the i^{th} output, the reference output $w_{ri}(t,T)$ be defined by the transfer function $\dfrac{R_{ni}(s)}{R_{di}(s)}$. Then $N_{y_i}^{th}$ order Maclaurin expansion $w_{ri}^{*}(t,T)$ of $w_{ri}(t,T)$ is given as follows (see section 4.2.3)

$$w_{ri}^{*}(t,T) = \underline{T}_{N_{y_i}}\underline{w}_i \tag{5.36}$$

where $\underline{T}_{N_{y_i}}$ is defined as in eqn. (5.7) and $\underline{w}_i$ is given by

$$\underline{w}_i = \underline{r}_i\ [\ w_i(t) - y_i(t)\ ] \tag{5.37}$$

where

$$\underline{r}_i = [\ r_{i0}\ \ r_{i1}\ \ r_{i2} \ldots . r_{iN_{y_i}}\ ]^T \tag{5.38}$$

where r_{ik} is the k^{th} markov parameter of $\frac{R_{ni}(s)}{R_{di}(s)}$. Using eqn. (5.36), it is then easy to show that the reference output vector

$$\underline{w}_r^*(t,T) = [\, w_{r1}^*(t,T) \;\; w_{r2}^*(t,T) \ldots . \; w_{rp}^*(t,T) \,]^T \tag{5.39}$$

is given as follows

$$\underline{w}_r^*(t,T) = \mathbf{T}_{N_y}\, \underline{w} \tag{5.40}$$

where $\mathbf{T}_{N_y}$ is defined as in eqn. (5.10) and $\underline{w}$ is defined as

$$\underline{w} = \begin{bmatrix} \underline{w}_1 \\ \underline{w}_2 \\ . \\ . \\ \underline{w}_p \end{bmatrix} \tag{5.41}$$

It can be shown that $\underline{w}$ can be further written as follows

$$\underline{w} = \mathbf{R}\,(\,\underline{w}(t) - \underline{y}(t)\,) \tag{5.42}$$

where $\underline{w}(t)$ is the set point vector, $\underline{y}(t)$ is the output vector and $\mathbf{R}$ is the block diagonal Markov parameters matrix of the reference models $\frac{R_{ni}(s)}{R_{di}(s)}$.

$$\mathbf{R} = \begin{bmatrix} \underline{r}_1 & \underline{0} & . & . & . & \underline{0} \\ \underline{0} & \underline{r}_2 & . & . & . & \underline{0} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \underline{0} & \underline{0} & . & . & . & \underline{r}_p \end{bmatrix} \tag{5.43}$$

where $\underline{0}$ is an appropriate dimension column vector.

5.2.4. MCGPC Control Law

As in the scalar case, the control objective here is to determine at time t a (predicted) future input vector $\underline{u}_r(t,T)$ such that the predicted outputs are as close as possible to the reference outputs. This is done again by minimizing a cost function of the future errors, between the predicted outputs and reference outputs, and the future inputs. Once the predicted future input vector is

determined, only its value at $T=0$ $\underline{u}_r(t,0)$ is applied, which is known as receding horizon strategy. The cost function considered is in the following form

$$J = \int_{T_1}^{T_2} [\, \underline{y}_r^*(t,T) - \underline{w}_r^*(t,T)\,]^T [\, \underline{y}_r^*(t,T) - \underline{w}_r^*(t,T)\,]\, dT + \int_0^{T_2-T_1} \underline{u}_r^*(t,T)^T \Lambda\, \underline{u}_r^*(t,T)\, dT \qquad (5.44)$$

where

$$\underline{y}_r^*(t,T) = \underline{y}^*(t+T) - \underline{y}(t) \qquad (5.45)$$

which is the same as eqn. (5.32) except that the first elements of the vectors $\underline{Y}_i^o$ $(i=1, .. p)$ in eqn (5.27) are set to zero, in the sequel these new vectors will be denoted by $\bar{\underline{Y}}_i^o$

$$\bar{\underline{Y}}_i^o = [\, 0 \;\; y_{i1}^o(t) \;\; y_{i2}^o(t) \ldots . \; y_{iN_{y_i}}^o(t)\,]^T \qquad (5.46)$$

and the vector of these vectors in eqn. (5.31) will be denoted by $\bar{\underline{Y}}^o$

$$\bar{\underline{Y}}^o = \begin{bmatrix} \bar{\underline{Y}}_1^o \\ \bar{\underline{Y}}_2^o \\ . \\ . \\ \bar{\underline{Y}}_p^o \end{bmatrix} \qquad (5.47)$$

Λ is a diagonal control weighting matrix

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & . & . & 0 \\ 0 & \lambda_2 & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & . & . & . & \lambda_m \end{bmatrix} \qquad (5.48)$$

and $\underline{u}_r^*(t,T)$ is the truncated Maclaurin series expansion form of the predicted future input $\underline{u}_r(t,T)$

$$\underline{u}_r^*(t,T) = \mathbf{T}_{N_u} \underline{u} \qquad (5.49)$$

where

$$\mathbf{T}_{N_u} = \begin{bmatrix} \underline{T}_{N_{u_1}} & \underline{0} & . & . & \underline{0} \\ \underline{0} & \underline{T}_{N_{u_2}} & . & . & \underline{0} \\ . & . & . & . & . \\ . & . & . & . & . \\ \underline{0} & \underline{0} & . & . & \underline{T}_{N_{u_m}} \end{bmatrix} \qquad (5.50)$$

$$\underline{T}_{N_{u_i}} = [\,1 \;\; T \;\; \frac{T^2}{2!} \;\; \ldots\ldots \;\; \frac{T^{N_{u_i}}}{N_{u_i}!}\,] \tag{5.51}$$

and the inputs derivatives vector $\underline{u}$ is defined as in eqn. (5.24).

In the cost (eqn. 5.44), we included a minimum prediction horizon T_1 for completeness. It is not an important design parameter and will be usually chosen to be zero however, if there is a common delay between all the input output pairs, T_1 can be set to the common delay. The effect of the maximum prediction horizon T_2 is similar to the scalar case that is, the larger the T_2 the slower the outputs response and vice versa. Note that the prediction horizons (T_1,T_2) are the same for all the outputs. A drawback of this is that, outputs transients can not be adjusted independently by T_2. But, this is not a problem as the output transients can always be adjusted independently by the reference models ($R_{ni}(s)/R_{di}(s)$) and/or control orders. It is also possible to consider different prediction horizons for each output however, it is omitted here to retain simplicity.

With the substitution of the eqn. (5.32) (*with* $\underline{Y}^o$ *replaced by* $\bar{\underline{Y}}^o$), eqn. (5.40) and eqn. (5.49) into eqn. (5.44), the cost becomes

$$J = \int_{T_1}^{T_2} (\mathbf{T}_{N_y}\mathbf{H}\underline{u} + \mathbf{T}_{N_y}\bar{\underline{Y}}^o - \mathbf{T}_{N_y}\underline{w})^T(\mathbf{T}_{N_y}\mathbf{H}\underline{u} + \mathbf{T}_{N_y}\bar{\underline{Y}}^o - \mathbf{T}_{N_y}\underline{w})dT + \int_0^{T_2-T_1} \underline{u}^T\mathbf{T}_{N_u}^T\Lambda\mathbf{T}_{N_u}\underline{u}\; dT \tag{5.52}$$

and the minimization of the cost (eqn. 5.52) results in

$$\underline{u} = \mathbf{K}\,(\,\underline{w} - \bar{\underline{Y}}^o\,) \tag{5.53}$$

where

$$\mathbf{K} = (\mathbf{H}^T\mathbf{T}_y\mathbf{H} + \mathbf{T}_u)^{-1}\mathbf{H}^T\mathbf{T}_y \tag{5.54}$$

$$\mathbf{T}_y = \int_{T_1}^{T_2} \mathbf{T}_{N_y}^T\mathbf{T}_{N_y}\; dT \tag{5.55}$$

$$\mathbf{T}_u = \int_0^{T_2-T_1} \mathbf{T}_{N_u}^T\Lambda\mathbf{T}_{N_u}\; dT \tag{5.56}$$

Note that the matrices $\mathbf{T}_y$ and $\mathbf{T}_u$ are block diagonal and symmetric.

$$\mathbf{T}_y = \begin{bmatrix} \mathbf{T}_{y_1} & 0 & \cdots & 0 \\ 0 & \mathbf{T}_{y_2} & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdots & \mathbf{T}_{y_p} \end{bmatrix} \tag{5.57}$$

$$\mathbf{T}_u = \begin{bmatrix} \lambda_1\mathbf{T}_{u_1} & 0 & \cdots & 0 \\ 0 & \lambda_2\mathbf{T}_{u_2} & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdots & \lambda_m\mathbf{T}_{u_m} \end{bmatrix} \tag{5.58}$$

where the submatrices are given as in the scalar case

$$\mathbf{T}_{y_i} = \int_{T_1}^{T_2} \underline{T}_{N_{y_i}}^T \underline{T}_{N_{y_i}} \, dT \quad ; \quad (N_{y_i}+1)\times(N_{y_i}+1) \quad ; \quad i=1,2, \cdots ,p \tag{5.59}$$

$$\mathbf{T}_{u_j} = \int_{0}^{T_2-T_1} \underline{T}_{N_{u_j}}^T \underline{T}_{N_{u_j}} \, dT \quad ; \quad (N_{u_j}+1)\times(N_{u_j}+1) \quad ; \quad j=1,2, \cdots ,m \tag{5.60}$$

and 0 is an appropriate dimension block zero matrix. Notice also that, if $\mathbf{T}_{y_i}^{max}$ represents the matrix $\mathbf{T}_{y_i}$ with the maximum dimension, then one can see that rest of the matrices $\mathbf{T}_{y_i}$ are the submatrices of $\mathbf{T}_{y_i}^{max}$, having the first $N_{y_i}+1$ rows and columns. The same is also true for the matrices $\mathbf{T}_{u_j}$. Note that the matrices $\mathbf{T}_{u_j}$ also become submatrices of $\mathbf{T}_{y_i}^{max}$ when $T_1 = 0$. The elements of the matrices $\mathbf{T}_{y_i}$ and $\mathbf{T}_{u_j}$ are given by eqn. (4.67) and (4.68) respectively.

Let $\mathbf{k}$ be the matrix formed from the rows of matrix $\mathbf{K}$ corresponding to the inputs $u_1(t), u_2(t), \ldots, u_m(t)$ then, the MCGPC control law is given by the following equation.

$$\underline{u}(t) = \mathbf{k}\,(\,\underline{w} - \overline{\underline{Y}}^o) \tag{5.61}$$

In the Laplace domain the control law becomes

$$\underline{U}(s) = \mathbf{k}\,\mathbf{R}\,[\,\underline{W}(s) - \underline{Y}(s)\,] - \mathbf{k}\,\overline{\underline{Y}}^o(s) \tag{5.62}$$

where $\overline{\underline{Y}}^o(s)$ is the Laplace transform of $\overline{\underline{Y}}^o$ and one can show that it is given as follows

$$\overline{\underline{Y}}^o(s) = \mathbf{G}(s)\,\overline{\mathbf{C}}(s)^{-1}\,\underline{U}(s) + \mathbf{F}(s)\,\mathbf{C}(s)^{-1}\,\underline{Y}(s) \tag{5.63}$$

where $\mathbf{G}(s)$ and $\mathbf{F}(s)$ are the block diagonal polynomial matrices of the polynomials $G_{ijk}(s)$ and $F_{ik}(s)$ respectively

$$\mathbf{G}(s) = \begin{bmatrix} \mathbf{G}_1(s) & 0 & \cdots & 0 \\ 0 & \mathbf{G}_2(s) & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdots & \mathbf{G}_p(s) \end{bmatrix} \quad ; \quad \mathbf{F}(s) = \begin{bmatrix} \mathbf{F}_1(s) & 0 & \cdots & 0 \\ 0 & \mathbf{F}_2(s) & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdots & \mathbf{F}_p(s) \end{bmatrix} \tag{5.64}$$

where 0 is an appropriate dimension zero matrix and the polynomial submatrices $\mathbf{G}_i(s)$ and $\mathbf{F}_i(s)$ $(i=1, \cdots, p)$ are given as follows

$$\mathbf{G}_i(s) = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ G_{i11}(s) & G_{i21}(s) & \cdots & G_{im1}(s) \\ G_{i12}(s) & G_{i22}(s) & \cdots & G_{im2}(s) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ G_{i1N_{y_i}}(s) & G_{i2N_{y_i}}(s) & \cdots & G_{imN_{y_i}}(s) \end{bmatrix} \quad ; \quad \mathbf{F}_i(s) = \begin{bmatrix} 0 \\ F_{i1}(s) \\ F_{i2}(s) \\ \cdot \\ \cdot \\ F_{iN_{y_i}}(s) \end{bmatrix} \tag{5.65}$$

and $\overline{\mathbf{C}}(s)^{-1}$ is given by

$$\overline{\mathbf{C}}(s)^{-1} = \begin{bmatrix} \frac{\mathbf{I}_m}{C_1(s)} \\ \frac{\mathbf{I}_m}{C_2(s)} \\ \cdot \\ \cdot \\ \frac{\mathbf{I}_m}{C_p(s)} \end{bmatrix} \tag{5.66}$$

which is the right inverse of

$$\overline{\mathbf{C}}(s) = \frac{1}{p} \left[\mathbf{I}_m C_1(s) \;\; \mathbf{I}_m C_2(s) \; \ldots \; \mathbf{I}_m C_p(s) \right] \tag{5.67}$$

where $\mathbf{I}_m$ is $m \times m$ unit matrix. With substitution of eqn. (5.63) into eqn. (5.62), the final form of the MCGPC control law is obtained as follows

$$\underline{U}(s) = \mathbf{G}_n [\underline{W}(s) - \underline{Y}(s)] - \mathbf{G}_c(s) \, \overline{\mathbf{C}}(s)^{-1} \, \underline{U}(s) - \mathbf{F}_c(s) \, \mathbf{C}(s)^{-1} \, \underline{Y}(s) \tag{5.68}$$

where

$$\mathbf{G}_n = \mathbf{k}\,\mathbf{R} \tag{5.69}$$

$$\mathbf{G}_c(s) = \mathbf{k}\,\mathbf{G}(s) \tag{5.70}$$

$$\mathbf{F}_c(s) = \mathbf{k}\,\mathbf{F}(s) \tag{5.71}$$

and the resulting MCGPC feedback system is illustrated in figure 5.1.

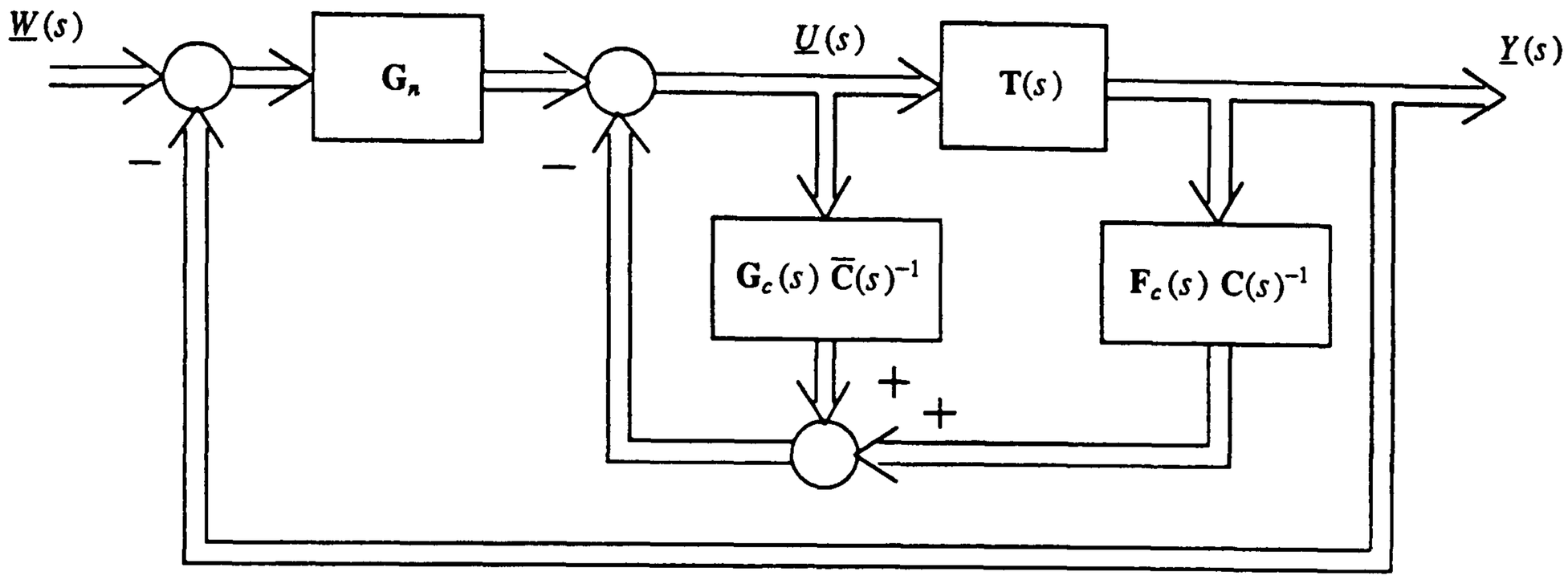


Figure 5.1 MCGPC feedback system

Remark 1: Note that, as a result of the receding horizon control strategy, the MCGPC control law is time invariant.

Remark 2: In the development of the above control law it is assumed that $deg\,(C_i(s)) = deg\,(A_i(s))$ -1. If $deg\,(C_i(s)) = deg\,(A_i(s))$, then the output $y_i(t)$ should be replaced by its emulated value $(i=1, .. ,p)$.

Remark 3: With a similar argument to the one given in section 4.2.4 under the heading *'More on the control law'*, it can be shown that MCGPC control law, as in the scalar case, corresponds to a state feedback however, this point will not be pursued here.

5.3. ANALYSIS OF THE MCGPC CLOSED LOOP SYSTEM

In this section, we will derive the closed-loop equations and examine the properties of the closed-loop system. In particular, it will be shown that for a specific setting of the design parameters, the MCGPC control law becomes a cancellation law as in the scalar case. It will also be shown that in some cases the closed-loop system is decoupled. In the sequel the following definitions will be needed.

Definition 1: Define the relative order matrix ρ as follows

$$\rho = \begin{bmatrix} \rho_{11} & \rho_{12} & \cdot & \cdot & \cdot & \rho_{1m} \\ \rho_{21} & \rho_{22} & \cdot & \cdot & \cdot & \rho_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \rho_{p1} & \rho_{p2} & \cdot & \cdot & \cdot & \rho_{pm} \end{bmatrix} \tag{5.72}$$

where ρ_{ij} is the relative order of the ij^{th} transfer function of the system transfer matrix $\mathbf{T}(s)$.

Definition 2: Define the predictor order vector $\underline{N}_y$ as

$$\underline{N}_y = [\, N_{y_1} \;\; N_{y_2} \ldots . N_{y_p} \,]^T \tag{5.73}$$

Definition 3: Let $N_{u_j}^{max}$ denotes the maximum possible value of the j^{th} control order N_{u_j} for a given $\underline{N}_y$, then it will be defined as

$$N_{u_j}^{max} = \max\, (\, \underline{N}_y - \underline{\rho}_{cj} \,) \tag{5.74}$$

where $\underline{\rho}_{cj}$ is the j^{th} column of ρ and $max(.)$ denotes the maximum element of the argument vector.

Definition 4: Let $\mathbf{H}_f$ denotes the full $\mathbf{H}$ matrix, then it will be defined as the $\mathbf{H}$ matrix corresponding to the control orders $N_{u_j} = N_{u_j}^{max}$; $j=1,2, \ldots ,m$.

5.3.1. Closed-Loop System Equations

The control input, as a result of application of the MCGPC control law (eqn. 5.68) to the system described by eqn. (5.1) (with zero disturbance input), is given as follows

$$\underline{U}(s) = [\, \mathbf{I}_m + \mathbf{G}_c(s)\overline{\mathbf{C}}(s)^{-1} + \mathbf{F}_c(s)\mathbf{C}(s)^{-1}\mathbf{A}(s)^{-1}\mathbf{B}(s) + \mathbf{G}_n\mathbf{A}(s)^{-1}\mathbf{B}(s) \,]^{-1}\mathbf{G}_n \, \underline{W}(s) \qquad (5.75)$$

In this equation the term $\mathbf{G}_c(s)\overline{\mathbf{C}}(s)^{-1} + \mathbf{F}_c(s)\mathbf{C}(s)^{-1}\mathbf{A}(s)^{-1}\mathbf{B}(s)$ is the key term for reducing this equation further and for rewriting it in a MFD form. So this term should be examined first.

Using identities (eqn. 5.14) and (eqn. 5.15), it can be shown that $C_i(s)$ polynomials cancel out from $\mathbf{G}_c(s)\overline{\mathbf{C}}(s)^{-1} + \mathbf{F}_c(s)\mathbf{C}(s)^{-1}\mathbf{A}(s)^{-1}\mathbf{B}(s)$, and it is given by

$$\mathbf{G}_c(s)\overline{\mathbf{C}}(s)^{-1} + \mathbf{F}_c(s)\mathbf{C}(s)^{-1}\mathbf{A}(s)^{-1}\mathbf{B}(s) = \mathbf{L}_c(s)\overline{\mathbf{A}}(s)^{-1} \qquad (5.76)$$

where

$$\mathbf{L}_c(s) = \mathbf{k}\,\mathbf{L}(s) \qquad (5.77)$$

where the block diagonal polynomial matrix $\mathbf{L}(s)$ is given as

$$\mathbf{L}(s) = \begin{bmatrix} \mathbf{L}_1(s) & 0 & \cdots & 0 \\ 0 & \mathbf{L}_2(s) & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdots & \mathbf{L}_p(s) \end{bmatrix} \qquad (5.78)$$

where 0 is an appropriate dimension zero matrix and the polynomial submatrices $\mathbf{L}_i(s)$ $(i=1, .. ,p)$ are given as follows

$$\mathbf{L}_i(s) = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ L_{i11}(s) & L_{i21}(s) & \cdots & L_{im1}(s) \\ L_{i12}(s) & L_{i22}(s) & \cdots & L_{im2}(s) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ L_{i1N_{y_i}}(s) & L_{i2N_{y_i}}(s) & \cdots & L_{imN_{y_i}}(s) \end{bmatrix} \qquad (5.79)$$

where the polynomials $L_{ijk}(s)$ $(j=1, .. ,m;\ k=1, .. ,N_{y_i})$ satisfy the following identity

$$\frac{s^k B_{ij}(s)}{A_i(s)} = H_{ijk}(s) + \frac{L_{ijk}(s)}{A_i(s)} \quad ; \quad deg\,(L_{ijk}(s)) = deg\,(A_i(s)) - 1 \qquad (5.80)$$

and $\overline{\mathbf{A}}(s)^{-1}$ is given by

$$\overline{\mathbf{A}}(s)^{-1} = \begin{bmatrix} \dfrac{\mathbf{I}_m}{A_1(s)} \\ \dfrac{\mathbf{I}_m}{A_2(s)} \\ . \\ . \\ \dfrac{\mathbf{I}_m}{A_p(s)} \end{bmatrix} \tag{5.81}$$

which is the right inverse of

$$\overline{\mathbf{A}}(s) = \frac{1}{p} [\, \mathbf{I}_m A_1(s) \;\; \mathbf{I}_m A_2(s) \ldots . \mathbf{I}_m A_p(s) \,] \tag{5.82}$$

where $\mathbf{I}_m$ is $m \times m$ unit matrix.

One may also show the following relationship by using eqn. (5.80)

$$\mathbf{S}\mathbf{A}(s)^{-1}\mathbf{B}(s) = \mathbf{H}_f \mathbf{S}_{H_f} + \mathbf{L}(s)\overline{\mathbf{A}}(s)^{-1} \tag{5.83}$$

where

$$\mathbf{S} = \begin{bmatrix} \underline{S}_{N_{y_1}} & \underline{0} & . & . & . & \underline{0} \\ \underline{0} & \underline{S}_{N_{y_2}} & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \underline{0} & \underline{0} & . & . & . & \underline{S}_{N_{y_p}} \end{bmatrix} \tag{5.84}$$

$$\underline{S}_{N_{y_i}} = [\, 0 \;\; s \;\; s^2 \ldots . s^{N_{y_i}} \,]^T \quad ; \quad i=1, .. ,p \tag{5.85}$$

and $\mathbf{H}_f$ is the full $\mathbf{H}$ matrix and $\mathbf{S}_{H_f}$ is the corresponding s matrix.

$$\mathbf{S}_{H_f} = \begin{bmatrix} \underline{S}_{N_{u_1}} & \underline{0} & . & . & . & \underline{0} \\ \underline{0} & \underline{S}_{N_{u_2}} & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \underline{0} & \underline{0} & . & . & . & \underline{S}_{N_{u_m}} \end{bmatrix} \tag{5.86}$$

$$\underline{S}_{N_{u_j}} = [\, 1 \;\; s \;\; s^2 \ldots . s^{N_{u_j}} \,]^T \quad ; \quad j=1, .. ,m \tag{5.87}$$

In the above equations $\underline{0}$ denotes an appropriate dimension zero column vector. Further, by

multiplying both sides of eqn. (5.83) by the matrix **k**, the following important relationship is obtained

$$\mathbf{Z}_c(s)\mathbf{A}(s)^{-1}\mathbf{B}(s) = \mathbf{H}_c(s) + \mathbf{L}_c(s)\overline{\mathbf{A}}(s)^{-1} \tag{5.88}$$

where the polynomial matrices $\mathbf{Z}_c(s)$ and $\mathbf{H}_c(s)$ are given by

$$\mathbf{Z}_c(s) = \mathbf{k}\,\mathbf{S} \qquad ; \qquad (m\times p) \tag{5.89}$$

$$\mathbf{H}_c(s) = \mathbf{k}\,\mathbf{H}_f\;\mathbf{S}_{H_f} \qquad ; \qquad (m\times m) \tag{5.90}$$

Using a right MFD form (eqn. 5.3) for the open-loop system in eqn. (5.88), one obtains the following relationship

$$\mathbf{L}_c(s)\overline{\mathbf{A}}(s)^{-1} = \mathbf{L}_R(s)\mathbf{A}_R(s)^{-1} \tag{5.91}$$

where

$$\mathbf{L}_R(s) = \mathbf{Z}_c(s)\mathbf{B}_R(s) - \mathbf{H}_c(s)\mathbf{A}_R(s) \tag{5.92}$$

Note that the polynomial matrix $\mathbf{L}_R(s)$ is $m\times m$.

Then it follows from eqn. (5.76), (5.91) and (5.3) that the closed-loop control input (eqn. 5.75) can be written as follows

$$\underline{U}(s) = [\,\mathbf{I}_m + \mathbf{L}_R(s)\mathbf{A}_R(s)^{-1} + \mathbf{G}_n\mathbf{B}_R(s)\mathbf{A}_R(s)^{-1}\,]^{-1}\mathbf{G}_n\;\underline{W}(s) \tag{5.93}$$

By rearranging eqn. (5.93), one can obtain the following right MFD for the closed-loop control input

$$\underline{U}(s) \;=\; \mathbf{A}_R(s)\,[\,\mathbf{A}_R(s) + \mathbf{L}_R(s) + \mathbf{G}_n\mathbf{B}_R(s)\,]^{-1}\mathbf{G}_n\;\;\underline{W}(s) \tag{5.94}$$

and it follows from eqn. (5.94) that a right MFD for the closed-loop system output is then given by

$$\underline{Y}(s) \;=\; \mathbf{B}_R(s)\,[\,\mathbf{A}_R(s) + \mathbf{L}_R(s) + \mathbf{G}_n\mathbf{B}_R(s)\,]^{-1}\mathbf{G}_n\;\;\underline{W}(s) \tag{5.95}$$

Note that the closed-loop zeros are the same as the open-loop zeros, which shows that MCGPC control law only changes the pole locations as in the scalar case. So non-minimum phase systems can be controlled by MCGPC without any problem. The closed-loop system described by eqn.

(5.95) is shown in figure 5.2. One may note that the feedback configuration in the figure actually corresponds to a state feedback where the partial state vector and its derivatives are fed back through the gain matrix defined by the coefficients of the polynomial matrix $\mathbf{L}_R(s) + \mathbf{G}_n\mathbf{B}_R(s)$ (Kailath, 1980, Wolovich, 1974).

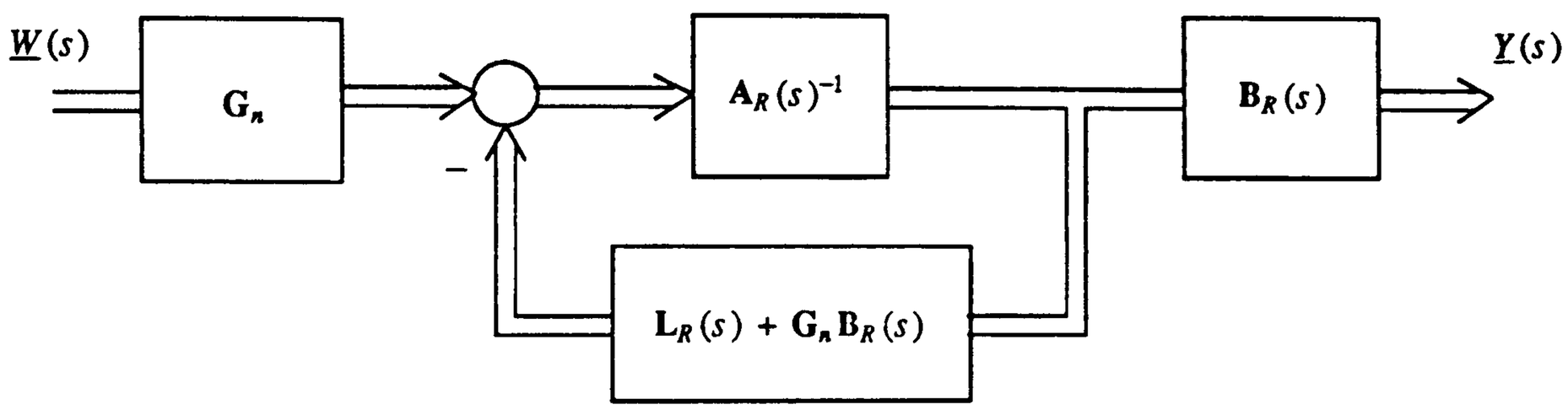


Figure 5.2 Equivalent MCGPC feedback system

As it is known, any right MFD can also be written in a left MFD form. However, a left MFD of eqn. (5.95) in terms of known polynomial matrices, in general, is not obvious but, for the square systems (assuming system is invertible) eqn. (5.95) can be easily rearranged to give the following left MFD of the closed-loop system.

$$\underline{Y}(s) = [\, \mathbf{A}(s) + \mathbf{L}_L(s) + \mathbf{B}(s)\mathbf{G}_n \,]^{-1}\mathbf{B}(s)\mathbf{G}_n \;\; \underline{W}(s) \tag{5.96}$$

where

$$\mathbf{L}_L(s) = \mathbf{B}(s)\mathbf{L}_R(s)\mathbf{B}_R(s)^{-1} = \mathbf{B}(s)\,[\, \mathbf{Z}_c(s) - \mathbf{H}_c(s)\mathbf{B}(s)^{-1}\mathbf{A}(s) \,] \tag{5.97}$$

5.3.2. A Special Case

Consider the case where

$$\Lambda = 0$$

$$N_{u_j} = N_{u_j}^{\max} \qquad ; \qquad j=1,2,\ldots,m$$

then the gain matrix **K** (eqn. 5.54) becomes

$$\mathbf{K} = (\mathbf{H}_f^T \mathbf{T}_y \mathbf{H}_f)^{-1} \mathbf{H}_f^T \mathbf{T}_y \tag{5.98}$$

If $\mathbf{H}_f$ has full rank, then $\mathbf{H}_f^T \mathbf{T}_y \mathbf{H}_f$ will be nonsingular. In this case, the closed-loop system is given as follows

$$\underline{Y}(s) = \mathbf{B}_R(s) [\, (\mathbf{Z}_c(s) + \mathbf{G}_n)\mathbf{B}_R(s) \,]^{-1} \mathbf{G}_n \;\; \underline{W}(s) \tag{5.99}$$

The proof is:

It follows from eqn. (5.92) and (5.95) that the closed-loop system can also be written as

$$\underline{Y}(s) = \mathbf{B}_R(s) [\, (\mathbf{I}_m - \mathbf{H}_c(s))\mathbf{A}_R(s) + (\mathbf{Z}_c(s) + \mathbf{G}_n)\mathbf{B}_R(s) \,]^{-1} \mathbf{G}_n \;\; \underline{W}(s) \tag{5.100}$$

using eqn. (5.98), it can easily be shown that the polynomial matrix $\mathbf{H}_c(s)$ becomes a unit matrix in this case

$$\mathbf{H}_c(s) = \mathbf{I}_m \tag{5.101}$$

then substitution of eqn. (5.101) into eqn. (5.100) results in eqn. (5.99).

Note that for the systems which have more inputs than outputs ($p < m$), $\mathbf{H}_f^T \mathbf{T}_y \mathbf{H}_f$ will always be singular. Therefore, eqn. (5.99) is not valid for such systems. This is an expected result as there are more than one input vector which gives the same output vector for the systems with $p < m$, that is the solution to the minimisation problem is not unique for such systems. To obtain a unique solution, some constraints on the inputs are needed. This means that the control orders should reduce from their maximum values ($N_{u_j}^{\max}$ s). In other words, for such systems the control orders can not be chosen larger than certain values where $\mathbf{H}^T \mathbf{T}_y \mathbf{H}$ becomes singular.

One may also note that, for square invertible systems the open-loop zeros are canceled out and thus the closed-loop system becomes

$$\underline{Y}(s) = (\, \mathbf{Z}_c(s) + \mathbf{G}_n \,)^{-1} \mathbf{G}_n \;\; \underline{W}(s) \tag{5.102}$$

Clearly, this gives unstable control for non-minimum-phase systems.

5.3.3. Decoupling in MCGPC

In MCGPC closed-loop system, interactions between different loops can be reduced by choosing larger value for control orders and/or smaller value for the maximum prediction horizon T_2 and/or using reference models for the outputs $(\frac{R_{ni}(s)}{R_{di}(s)})$. For some systems, it is even possible to obtain completely decoupled closed-loop system in some cases, which will be given as a theorem following some definitions. Here we will consider only $(p \times p)$ square systems.

Definition 1: Let ρ_{ri} denotes the relative order of the i^{th} row of $\mathbf{T}(s)$, then it will be defined as

$$\rho_{ri} = min(\varrho_{ri}) \tag{5.103}$$

where ϱ_{ri} is the i^{th} row of the relative order matrix ρ and $min(.)$ denotes the minimum element of the argument vector.

Definition 2: The matrix $\mathbf{H}$ can be decomposed as follows

$$\mathbf{H} = \begin{bmatrix} 0_1 \\ \overline{\mathbf{H}}_1 \\ 0_2 \\ \overline{\mathbf{H}}_2 \\ . \\ . \\ 0_p \\ \overline{\mathbf{H}}_p \end{bmatrix} \tag{5.104}$$

where 0_i is a zero matrix with dimension $\rho_{ri} \times (m + \sum_{j=1}^{m} N_{u_j})$ and $\overline{\mathbf{H}}_i$ does not have any zero row $(i=1, \ldots, p)$. Then define the matrix $\overline{\mathbf{H}}$ as

$$\overline{\mathbf{H}} = \begin{bmatrix} \overline{\mathbf{H}}_1 \\ \overline{\mathbf{H}}_2 \\ . \\ . \\ \overline{\mathbf{H}}_p \end{bmatrix} \tag{5.105}$$

and denote the $\overline{\mathbf{H}}$ matrix for the full $\mathbf{H}$ matrix $(\mathbf{H}_f)$ as $\overline{\mathbf{H}}_f$.

Note that the matrix $\overline{\mathbf{H}}_f$ will be in the following form

$$\overline{\mathbf{H}}_f = \begin{bmatrix} \overline{\mathbf{H}}_{f11} & \overline{\mathbf{H}}_{f12} & \cdots & \overline{\mathbf{H}}_{f1p} \\ \overline{\mathbf{H}}_{f21} & \overline{\mathbf{H}}_{f22} & \cdots & \overline{\mathbf{H}}_{f2p} \\ \vdots & \vdots & \cdots & \vdots \\ \overline{\mathbf{H}}_{fp1} & \overline{\mathbf{H}}_{fp2} & \cdots & \overline{\mathbf{H}}_{fpp} \end{bmatrix} \tag{5.106}$$

where each submatrix $\overline{\mathbf{H}}_{fij}$ is a lower triangular matrix with dimension $(N_{y_i}+1-\rho_{ri})\times(N_{u_j}^{max}+1)$.

Definition 3: Let h_{ij} be the $(1,1)$ element of the submatrix $\overline{\mathbf{H}}_{fij}$ of $\overline{\mathbf{H}}_f$, then define the matrix $\tilde{\mathbf{H}}$ as follows

$$\tilde{\mathbf{H}} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1p} \\ h_{21} & h_{22} & \cdots & h_{2p} \\ \vdots & \vdots & \cdots & \vdots \\ h_{p1} & h_{p2} & \cdots & h_{pp} \end{bmatrix} \tag{5.107}$$

note that some of the h_{ij}s could be zero.

Remark 1 : Assume that $\overline{\mathbf{H}}_f$ is a square matrix and nonsingular, then $\overline{\mathbf{H}}_f^{-1}$ will also have a lower triangular block structure where the dimension of each triangular block (submatrix) will be $(N_{u_j}^{max}+1)\times(N_{y_i}+1-\rho_{ri})$ and the matrix formed from the 11^{th} elements of the each triangular submatrix of $\overline{\mathbf{H}}_f^{-1}$ as in definition 3, say $\mathbf{H}^*$, will be the inverse of $\tilde{\mathbf{H}}$

$$\mathbf{H}^* = \tilde{\mathbf{H}}^{-1} \tag{5.108}$$

Theorem 5.1 : For $p\times p$ invertible systems if MCGPC parameters Λ and N_{u_j} are chosen to be

$$\Lambda = 0$$

$$N_{u_j} = N_{u_j}^{max} \qquad ; \qquad j=1,2,\ldots,p$$

and if the matrix $\overline{\mathbf{H}}_f$ is square and nonsingular, then MCGPC control law results in a decoupled closed-loop system.

Note that the matrix $\overline{\mathbf{H}}_f$ being square and nonsingular implies that $\mathbf{H}_f$ has full rank, but if $\mathbf{H}_f$ has full rank, it does not imply that $\overline{\mathbf{H}}_f$ is square. This is the point which makes a distinction from the special case of the previous section. More precisely this means that, to decouple a system the conditions of the special case must be satisfied but, not every system can be decoupled, only the systems with square $\overline{\mathbf{H}}_f$ can.

Proof: Consider the gain matrix $\mathbf{K}$ (eqn. 5.98) when $\mathbf{\Lambda} = 0$ and $N_{u_j} = N_{u_j}^{\max}$. Assume that $\mathbf{H}_f$ is decomposed as in eqn. (5.104) and decompose the matrix $\mathbf{T}_y$ accordingly as follows

$$\mathbf{T}_y = \begin{bmatrix} \mathbf{T}_{y_{111}} & \mathbf{T}_{y_{112}} & 0 & 0 & \cdots & 0 & 0 \\ \mathbf{T}_{y_{121}} & \mathbf{T}_{y_{122}} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{T}_{y_{211}} & \mathbf{T}_{y_{212}} & \cdots & \cdot & \cdot \\ 0 & 0 & \mathbf{T}_{y_{221}} & \mathbf{T}_{y_{222}} & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdots & \mathbf{T}_{y_{p11}} & \mathbf{T}_{y_{p12}} \\ 0 & 0 & \cdot & \cdot & \cdots & \mathbf{T}_{y_{p21}} & \mathbf{T}_{y_{p22}} \end{bmatrix} \tag{5.109}$$

where 0s are appropriate dimension zero matrices. Then it can be shown that the matrix $\mathbf{K}$ can be written as follows

$$\mathbf{K} = \overline{\mathbf{H}}_f^{-1}\, \overline{\mathbf{T}}_y \tag{5.110}$$

where

$$\overline{\mathbf{T}}_y = \begin{bmatrix} \mathbf{T}_{y_{122}}^{-1}\mathbf{T}_{y_{121}} & \mathbf{I} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{T}_{y_{222}}^{-1}\mathbf{T}_{y_{221}} & \mathbf{I} & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdots & \mathbf{T}_{y_{p22}}^{-1}\mathbf{T}_{y_{p21}} & \mathbf{I} \end{bmatrix} \tag{5.111}$$

where 0s and Is are appropriate dimension zero and unit matrices respectively. As we are only interested in rows of $\mathbf{K}$ corresponding to $u_1(t), u_2(t), \ldots, u_p(t)$, it follows from eqn. (5.110) and remark 1 give in the previous page that in this special case the matrix $\mathbf{k}$ can be written as follows

$$\mathbf{k} = \tilde{\mathbf{H}}^{-1}\, \tilde{\mathbf{T}} \tag{5.112}$$

where

$$\tilde{\mathbf{T}} = \begin{bmatrix} \underline{\tilde{T}}_1 & \underline{0} & \cdots & \underline{0} \\ \underline{0} & \underline{\tilde{T}}_2 & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \underline{0} & \cdot & \cdots & \underline{\tilde{T}}_p \end{bmatrix} \quad ; \quad \underline{\tilde{T}}_i = [\, \underline{t}_{y_i} \;\; 1 \;\; 0 \ldots 0 \,] \qquad 1\times(N_{y_i}+1) \tag{5.113}$$

where $\underline{t}_{y_i}$ $(1\times\rho_{ri})$ is the first row of $\mathbf{T}_{y_{i22}}^{-1}\mathbf{T}_{y_{i21}}$ $(i=1, .. ,p)$.

Now consider the closed-loop transfer matrix when $\Lambda = 0$, $N_{u_j} = N_{u_j}^{\max}$ and $\mathbf{H}_f$ has full rank

$$\mathbf{T}_c(s) = (\, \mathbf{Z}_c(s) + \mathbf{G}_n \,)^{-1}\mathbf{G}_n \tag{5.114}$$

It follows from eqn. (5.89), (5.69) and (5.112) that in this case $\mathbf{Z}_c(s)$ and $\mathbf{G}_n$ can be written as

$$\mathbf{Z}_c(s) = \tilde{\mathbf{H}}^{-1}\,\tilde{\mathbf{Z}}_c(s) \tag{5.115}$$

$$\mathbf{G}_n = \tilde{\mathbf{H}}^{-1}\,\tilde{\mathbf{G}}_n \tag{5.116}$$

where

$$\tilde{\mathbf{Z}}_c(s) = \tilde{\mathbf{T}}\,\mathbf{S} \tag{5.117}$$

$$\tilde{\mathbf{G}}_n = \tilde{\mathbf{T}}\,\mathbf{R} \tag{5.118}$$

Then substitution of eqn. (5.115) and (5.116) into eqn (5.114) results in

$$\mathbf{T}_c(s) = (\, \tilde{\mathbf{Z}}_c(s) + \tilde{\mathbf{G}}_n \,)^{-1}\tilde{\mathbf{G}}_n \tag{5.119}$$

Since $\tilde{\mathbf{T}}$, $\mathbf{S}$ and $\mathbf{R}$ are block diagonal matrices, $\tilde{\mathbf{Z}}_c(s)$ is a diagonal polynomial matrix and $\tilde{\mathbf{G}}_n$ is a diagonal gain matrix. Then it is obvious that the closed-loop transfer matrix $\mathbf{T}_c(s)$ is diagonal. This ends the proof. Note that the order of i^{th} diagonal element $T_{ci}(s)$ of $\mathbf{T}_c(s)$ will be ρ_{ri}.

Remark 2: Wolovich gives a theorem concerning the decouplability of a system by a linear state feedback in (Wolovich, 1974). We will rewrite this theorem for our case as follows:

A $p\times p$ invertible system can be decoupled by a linear state feedback alone if and only if $\mathbf{B}^*$ is nonsingular. Where the matrix $\mathbf{B}^*$ is defined as follows:

$$\mathbf{B}^* = \lim_{s \to \infty} \mathbf{D}(s)\mathbf{T}(s) \tag{5.120}$$

where

$$\mathbf{D}(s) = \begin{bmatrix} s^{\rho_{r1}} & 0 & . & . & . & 0 \\ 0 & s^{\rho_{r2}} & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & . & . & . & . & s^{\rho_{rp}} \end{bmatrix} \tag{5.121}$$

One may note that the matrix $\tilde{\mathbf{H}}$ and $\mathbf{B}^*$ are the same. One may also note that to have a square and nonsingular $\bar{\mathbf{H}}_f$, the matrix $\tilde{\mathbf{H}}$ must be nonsingular. This result is not surprising as MCGPC control law is a linear state feedback. So MCGPC can only decouple the systems which can be decoupled by a linear state feedback.

Model-following control

In addition to the above decoupling conditions, assume that system has unit row relative orders (ρ_{ri}) and we choose first order models as follows

$$\frac{R_{ni}(s)}{R_{di}(s)} = \frac{r_i}{s + r_i} \qquad ; \qquad i=1, \ldots p \tag{5.122}$$

it is easy to show that

$$\frac{r_i}{s + r_i} = 0 + r_i s^{-1} - r_i^2 s^{-2} + r_i^3 s^{-3} - \ldots . \tag{5.123}$$

It follows from eqn. (5.117), (5.118) and (5.123) that in this case $\tilde{\mathbf{Z}}_c(s)$ and $\tilde{\mathbf{G}}_n$ will be

$$\tilde{\mathbf{Z}}_c(s) = \begin{bmatrix} s & 0 & . & . & . & 0 \\ 0 & s & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & . & . & . & . & s \end{bmatrix} \qquad ; \qquad \tilde{\mathbf{G}}_n = \begin{bmatrix} r_1 & 0 & . & . & . & 0 \\ 0 & r_2 & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & . & . & . & . & r_p \end{bmatrix} \tag{5.124}$$

then from eqn. (5.119) it is obvious that

$$\mathbf{T}_c(s) = \begin{bmatrix} \frac{r_1}{s+r_1} & 0 & \cdots & 0 \\ 0 & \frac{r_2}{s+r_2} & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdots & \frac{r_p}{s+r_p} \end{bmatrix} \tag{5.125}$$

Hence, in this case exact model-following is obtained. If system's row relative orders are not unity, it can be shown (as in scalar case) that one of the poles of transfer functions of $\mathbf{T}_c(s)$ will be equal to corresponding model pole. As the other poles can be placed faraway from the imaginary axis by proper choice of T_2, it is also possible to obtain a very close model following when $\rho_{ri} > 1 \quad (i=1, \ldots, p)$.

5.3.4. Relation of MCGPC to LQ Control

One can argue in a similar way as in section 4.5 that MCGPC becomes LQ control with the following setting of the parameters.

$$R_{ni} = R_{di} = 1$$

$$N_{u_j} = N_{u_j}^{\max}$$

$$N_{y_i} \to \infty$$

$$T_1 = 0$$

$$T_2 \to \infty$$

$$i=1, \ldots, p \qquad j=1, \ldots, m$$

Here we will illustrate this relationship with an example. First consider the LQ cost function

$$J = \int_0^\infty [\, \underline{x}^T(t)\mathbf{Q}\underline{x}(t) + \underline{u}^T(t)\mathbf{\Lambda}\underline{u}(t)]dt \tag{5.126}$$

for the system

$$\dot{\underline{x}}(t) = \mathbf{A}\, \underline{x}(t) + \mathbf{B}\, \underline{u}(t) \tag{5.127}$$

$$\underline{y}(t) = \mathbf{C}\, \underline{x}(t) \tag{5.128}$$

and choose $\mathbf{Q} = \mathbf{C}^T\mathbf{C}$. Then the optimal closed-loop poles are the left half plane characteristic values of the matrix **M** (Kwakernaak, 1972)

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & -\mathbf{B}\Lambda^{-1}\mathbf{B}^T \\ -\mathbf{C}^T\mathbf{C} & -\mathbf{A}^T \end{bmatrix} \tag{5.129}$$

Example: Consider the following 2×2 system

$$\mathbf{T}(s) = \begin{bmatrix} \frac{s+2}{s^2+1} & \frac{1}{s^2+1} \\ \frac{1}{s^2-1} & \frac{s+1}{s^2-1} \end{bmatrix} \tag{5.130}$$

with the following setting of MCGPC parameters

$N_{y_1} = N_{y_2} = 11$

$N_{u_1} = N_{u_2} = 10$

$T_1 = 0$

$R_{n1} = R_{d1} = R_{n2} = R_{d2} = 1$

$\mathbf{C}(s) = (s+1)\,\mathbf{I}$

$\mathbf{\Lambda} = 0.1\,\mathbf{I}$

It follows from eqn. (5.129) that for this example the optimal closed-loop poles are:

$$-2.5047 \pm 1.3853i \qquad -0.4799 \qquad -3.3257$$

The closed-loop MCGPC poles were calculated for several values of T_2 to see whether they converge to LQ poles and resulting poles are given below.

$T_2 = 3.5$	$-2.5073 \pm 1.3794i$	-0.4629	-3.3195
$T_2 = 4$	$-2.5076 \pm 1.3720i$	-0.4691	-3.3143
$T_2 = 4.5$	$-2.5057 \pm 1.3600i$	-0.4732	-3.3063
$T_2 = 5$	$-2.4997 \pm 1.3429i$	-0.4759	-3.2954

These result shows that closed-loop MCGPC poles for this range of T_2 are very close to the closed-loop LQ poles which verifies the above argument.

5.4. SOME ILLUSTRATIVE SIMULATIONS

In this section, some simulation results are presented to illustrate the properties and performance of the MCGPC algorithm. Simulations are arranged in two groups. The first group is aimed to illustrate the effects of different design parameters and some properties of the MCGPC algorithm, such as decoupling and model following. These simulations were performed non-adaptively, as these properties will be the same in the self-tuning case. The second group is aimed to illustrate the performance of the self-tuning MCGPC. All of the simulations were performed by using the MATLAB package running on a SUN 3 workstation. Examples used in the simulations are given below.

Example 1:

$$\mathbf{T}(s) = \begin{bmatrix} \frac{s+2}{s^2+1} & \frac{1}{s^2+1} \\ \frac{1}{s^2-1} & \frac{s+1}{s^2-1} \end{bmatrix} \quad ; \quad \mathbf{C}(s) = \begin{bmatrix} s+1 & 0 \\ 0 & s+1 \end{bmatrix}$$

Example 2:

$$\mathbf{T}(s) = \begin{bmatrix} \frac{1}{(s^2+1)(0.5s^2+1)} & \frac{s+2}{(s^2+1)(0.5s^2+1)} \\ \frac{1}{2s^2+1} & \frac{s+1}{2s^2+1} \end{bmatrix} \quad ; \quad \mathbf{C}(s) = \begin{bmatrix} (s+1)^3 & 0 \\ 0 & s+1 \end{bmatrix}$$

Example 3:

$$\mathbf{T}(s) = \begin{bmatrix} \frac{s-4}{(s^2-1)(s^2+1)} & \frac{s+1}{(s^2-1)(s^2+1)} \\ \frac{1}{(s-0.5)(s^2-4)} & \frac{s+3}{(s-0.5)(s^2-4)} \end{bmatrix} \quad ; \quad \mathbf{C}(s) = \begin{bmatrix} (s+1)^3 & 0 \\ 0 & (s+1)^2 \end{bmatrix}$$

Example 4:

$$\mathbf{T}(s) = \begin{bmatrix} \frac{s+1}{s^2+s+1} & \frac{s+2}{s^2+s+1} & \frac{4}{s^2+s+1} \\ \frac{s+3}{(s+2)(s-1)} & \frac{1}{(s+2)(s-1)} & \frac{s+2}{(s+2)(s-1)} \end{bmatrix} \quad ; \quad \mathbf{C}(s) = \begin{bmatrix} s+1 & 0 \\ 0 & s+1 \end{bmatrix}$$

5.4.1. Non-Adaptive Simulations

In this group of simulations the following are common:

1- Sample interval is 0.05 sec,

2- $\underline{R}_n(s) = [\ 1\ \ 1\ ]$ (model numerators),

3- $T_1 = 0$ and $\Lambda = 0$,

4- Each figure consists of four graphs: the upper two graphs show the system outputs for various design parameters and the setpoints (square waves). The lower two graphs show the corresponding control inputs.

5.4.1.1. The effects of T_2 and N_u

As we mentioned earlier, the maximum prediction horizon T_2 has similar effects as in the scalar case that is, the smaller the T_2, the faster the closed-loop system outputs and vice versa. As T_2 gets smaller, the controller gain becomes higher this also reduces the interactions. In the limiting case when $T_2 \rightarrow 0$, the controller gain tends to infinity and thus $\underline{Y}(s) \rightarrow \underline{W}(s)$. This obviously gives unstable control for the non-minimum phase systems. For this type of systems T_2 should cover the negative going part of the output responses. Example 1 was simulated the illustrate the effects of T_2 with the following design parameters.

$$\underline{R}_d(s) = [\ 1\ \ 1\ ]$$

$$\underline{N}_y = [\ 6\ \ 6\ ]$$

$$\underline{N}_u = [\ 2\ \ 2\ ]$$

The simulation results are given in figure 5.3. In the figure, solid line corresponds to $T_2 = 1$, dashed line $T_2 = 2$ and dotted line $T_2 = 3$. As can be seen from the figure, as T_2 gets smaller, the control inputs become more active, the output responses become faster and the interactions become less and less. If T_2 is chosen very small, such as 0.01, it can be seen that the closed-loop outputs coincide with the setpoints.

The effects of the control orders are also similar to the scalar case that is, the larger the control orders the faster the closed-loop outputs and vice versa. In addition, the larger control orders result in less interactions for the decouplable systems. Again we simulated example 1 to illustrate

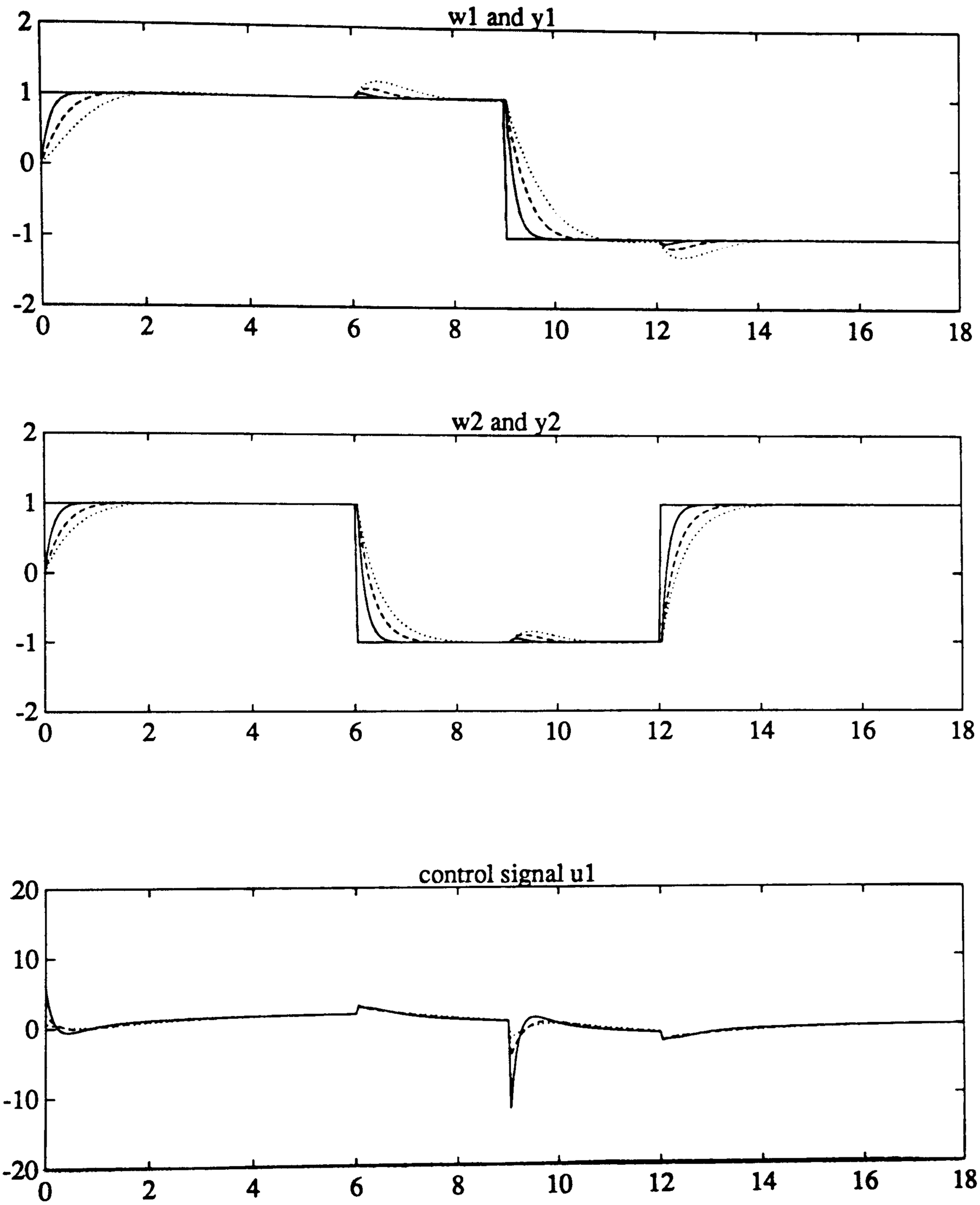


Figure 5.3 Illustration of the effects of T_2

the effects of the control orders. In the simulations $\underline{R}_d(s)$ and $\underline{N}_y$ were chosen to be as in the previous simulation and T_2 to be 2. The simulation results are given in figure 5.4. In the figure solid line corresponds to $\underline{N}_u$ = [3 3], dashed line $\underline{N}_u$ = [2 2] and dotted line $\underline{N}_u$ = [1 1]. As can be seen from the figure, as the control orders increase, the controls become more active, the output responses become faster and the interactions become less and less. Note that in these simulations the control orders were chosen to be the same for both inputs. Control orders can also be chosen different for different inputs, which is useful as it enables us to adjust the output transients separately. An example of this is given in figure 5.5 where $\underline{N}_u$ = [1 3]. As a result, second control input is much more active than the first control input and thus the second output is much faster than the first output, the interaction on the second output is much less than the interaction on the first output.

5.4.1.2. The effects and use of the reference models

Reference models $R_{ni}(s)/R_{di}(s)$ can be used in three different ways:

1- to penalize the overshoots and reduce the interactions,

2- to adjust the output transients separately,

3- to obtain model-following type control or exact model-following (exact model-following is not possible for every system).

Example 2, which is a highly oscillatory and interacting system, was simulated to illustrate the first use of the reference models. In the simulations the following design parameters were used:

$\underline{N}_y$ = [8 8]

$\underline{N}_u$ = [1 1]

T_2 = 2

The simulation results are shown in figure 5.6. In the figure, dashed line corresponds to $\underline{R}_d(s)$ = [1 1] and solid line $\underline{R}_d(s)$ = [s+1 s+1]. As can be seen from the figure, the use of reference models completely removed the overshoots and reduced the interactions.

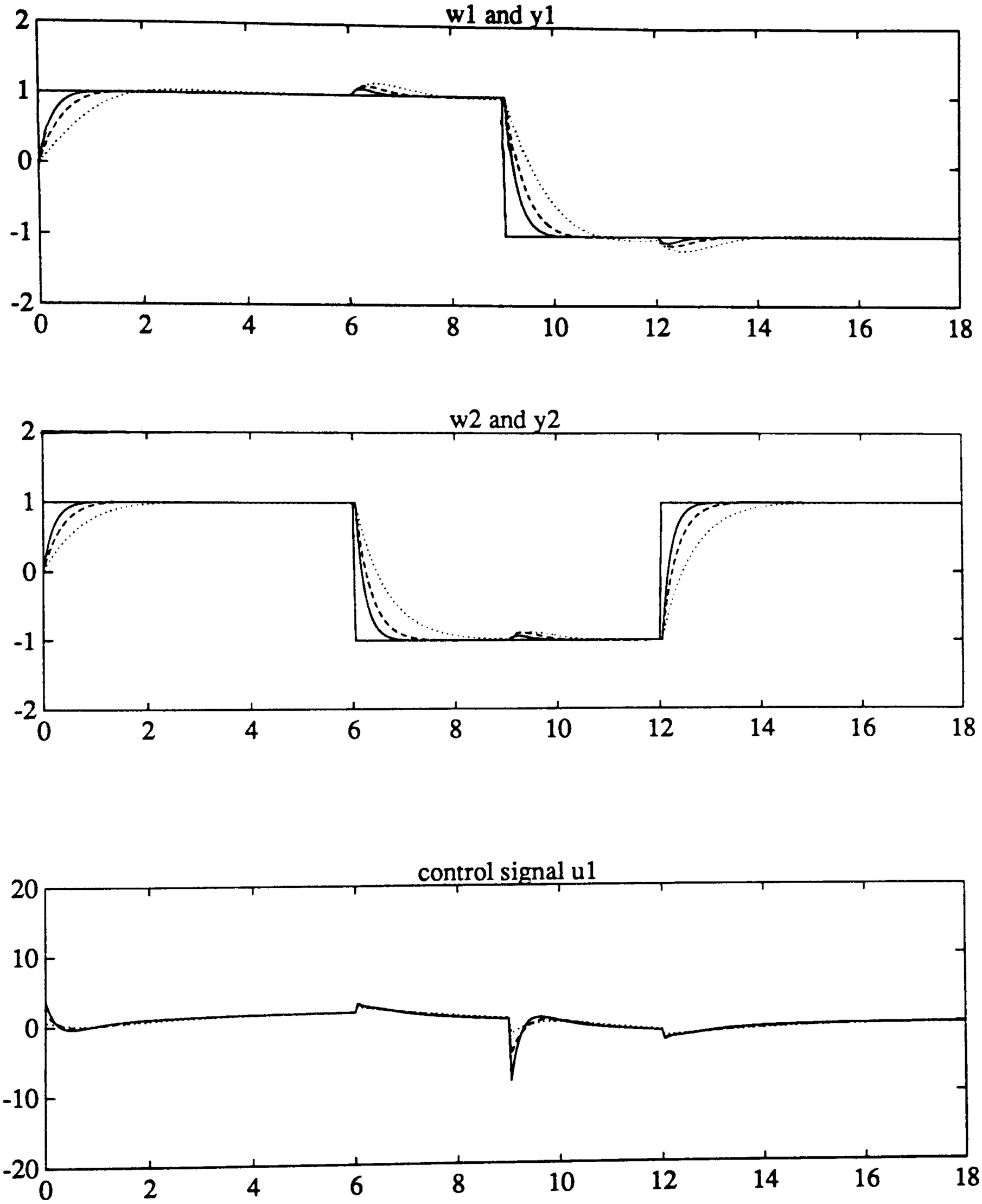


Figure 5.4 Illustration of the effects of the control orders

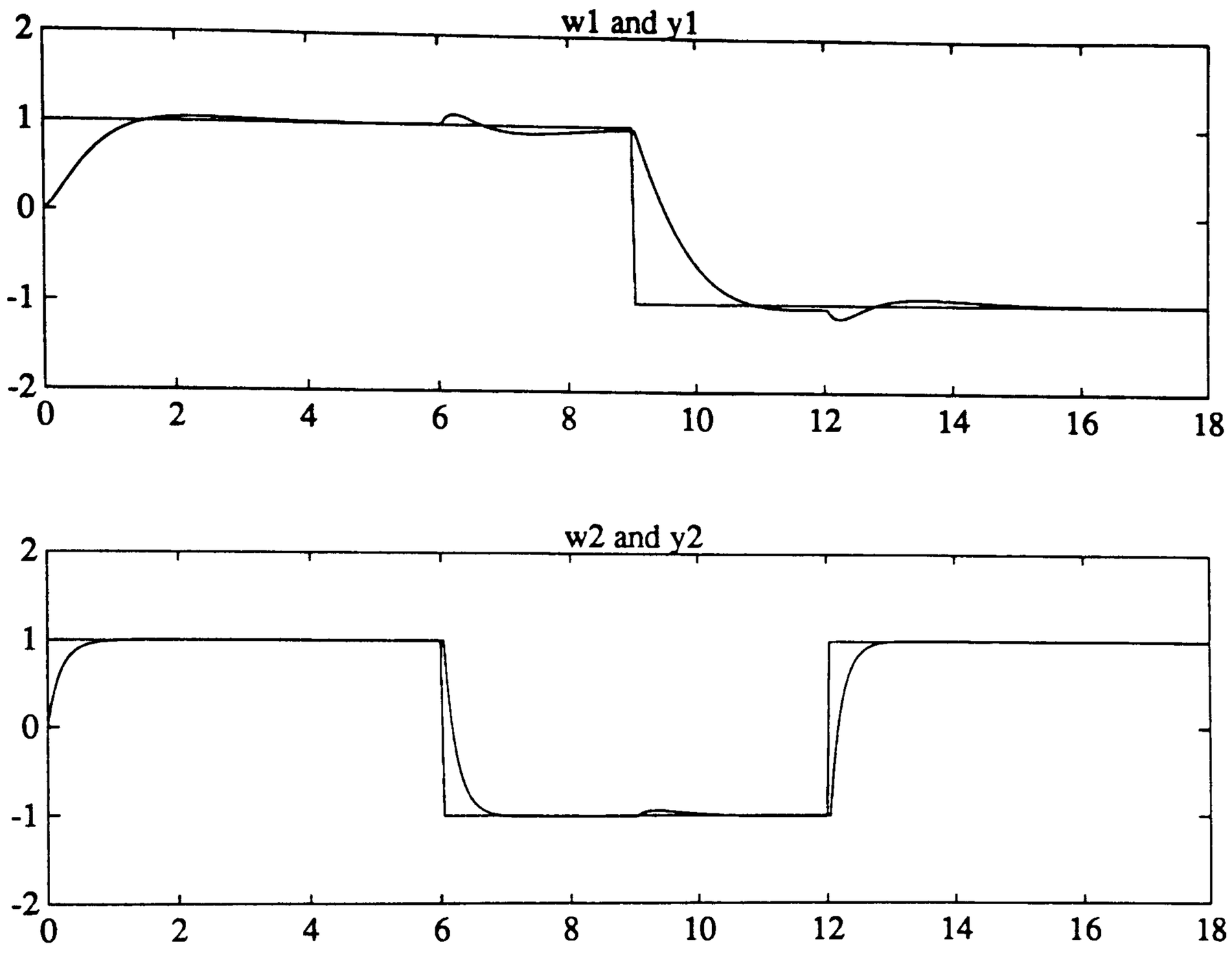


Figure 5.5 Effects of the different control orders for different inputs

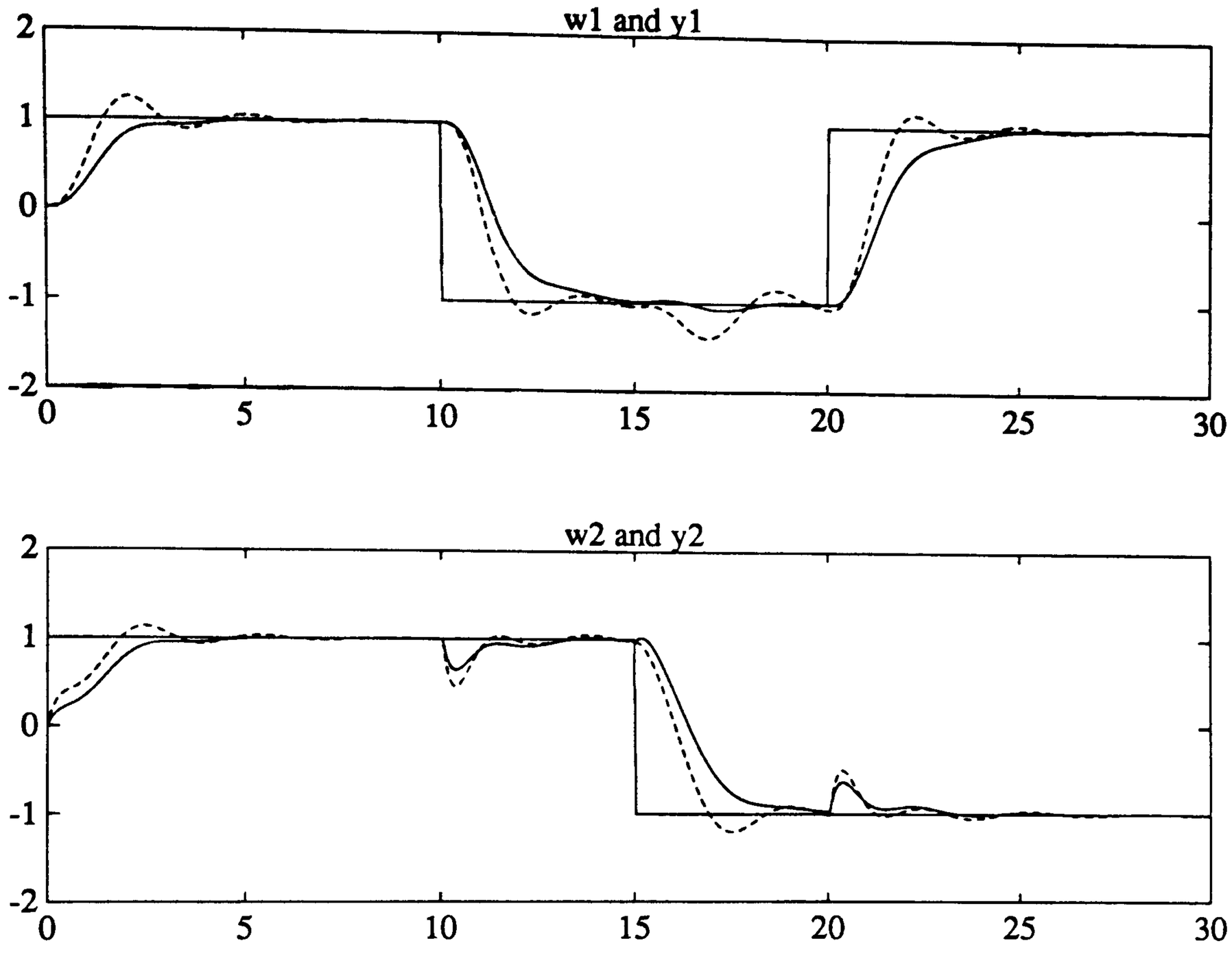


Figure 5.6 Use of the reference models to penalize overshoots and reduce interactions

Example 1 was simulated to show the second use of the reference models with the following parameters.

$$\underline{N}_y = [\,6 \;\; 6\,]$$

$$\underline{N}_u = [\,1 \;\; 1\,]$$

$$T_2 = 1$$

The simulation results are given in figure 5.7. In the figure, dashed line corresponds to $\underline{R}_d(s) = [\,1 \;\; 1\,]$ and solid line corresponds to $\underline{R}_d(s) = [\,1 \;\; s+1\,]$. As we did not specify any reference model for the first output it remained the same, but the second output was modified as specified by the corresponding reference model.

The use of the reference models to obtain model-following type control and exact model-following will be given in the following section together with decoupling.

5.4.1.3. Decoupling and model-following

In section 5.3.3 the conditions for decoupling and exact model-following are given. Here we will illustrate these properties by simulation. For this purpose example 1 was simulated. Note that the matrix $\mathbf{B}^*$ of example 1 is nonsingular so the system is decouplable and also note that row relative orders of the system (ρ_{r1}, ρ_{r2}) are unity so it is also possible to obtain exact model-following. In the simulation the following parameters were used:

$$\underline{R}_d(s) = [\,0.5s+1 \quad s+1\,]$$

$$\underline{N}_y = [\,6 \;\; 6\,]$$

$$\underline{N}_u = [\,5 \;\; 5\,]$$

$$T_2 = 2$$

Note that $N_{u_j} = N_{u_j}^{\max}$ $(J=1,2)$. The simulation result is given in figure 5.8. As seen from the figure, the closed-loop system is decoupled and exact model-following is obtained.†

† As model outputs and system outputs completely overlap, they are not distinguishable from each other in the figure.

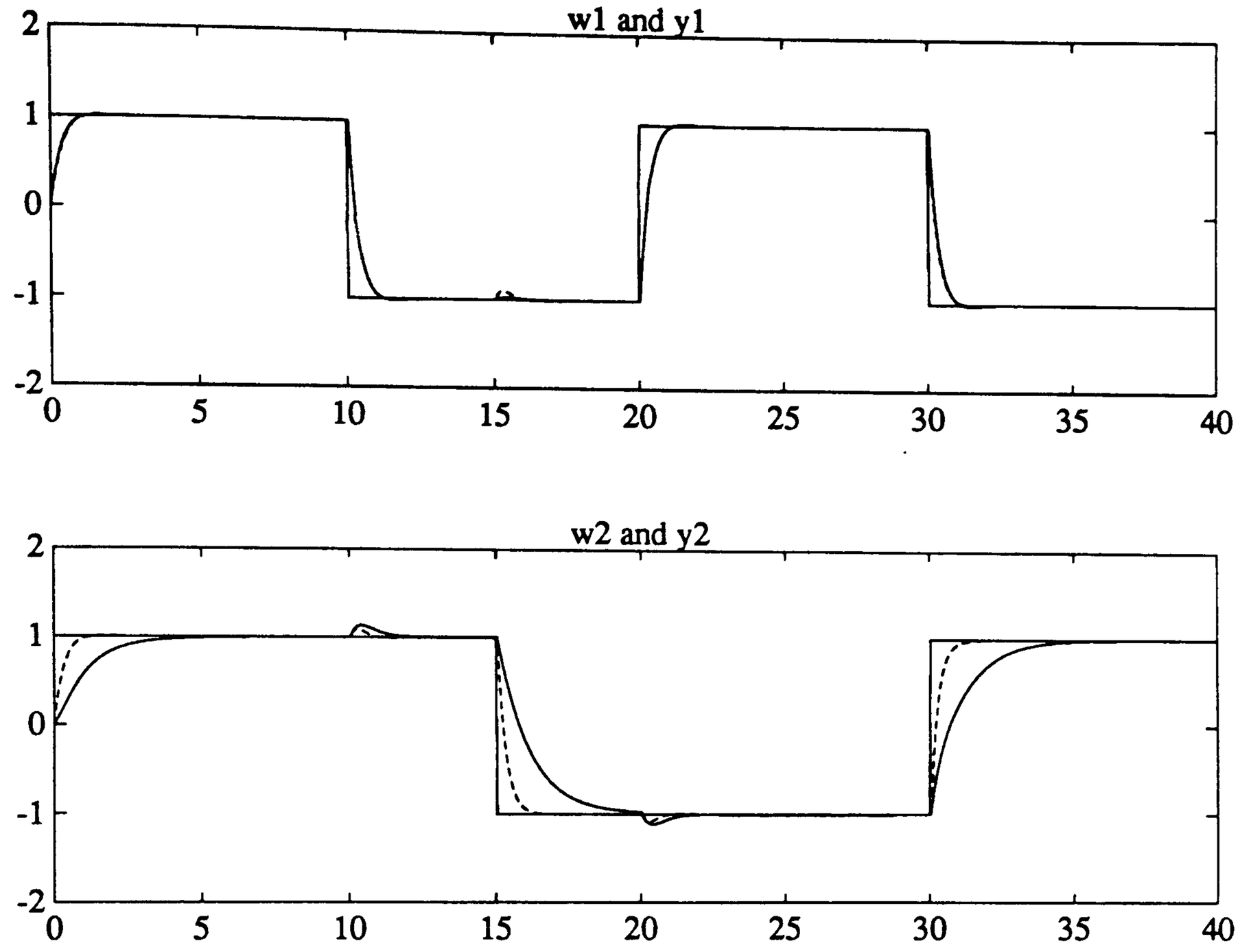
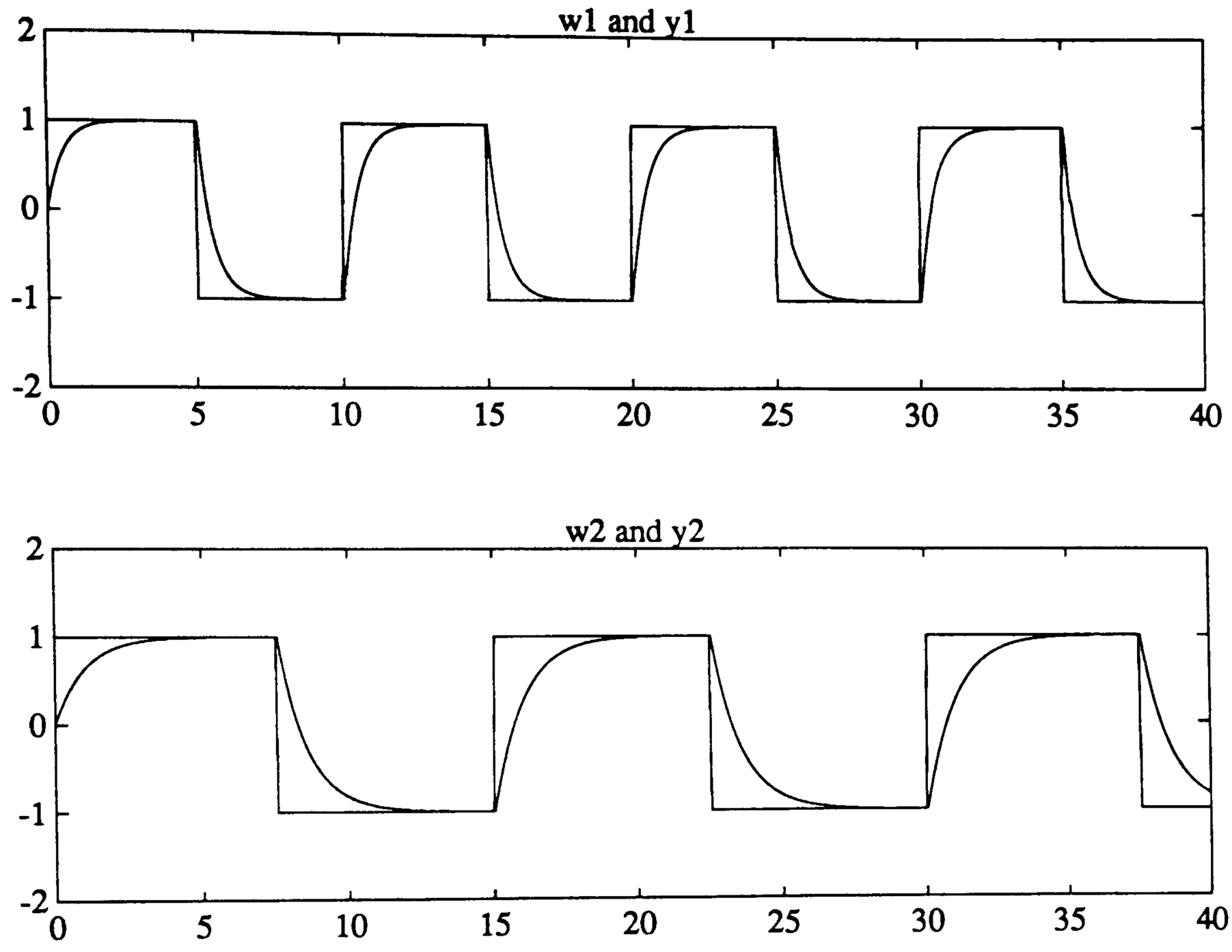


Figure 5.7 Adjusting output transients by reference models



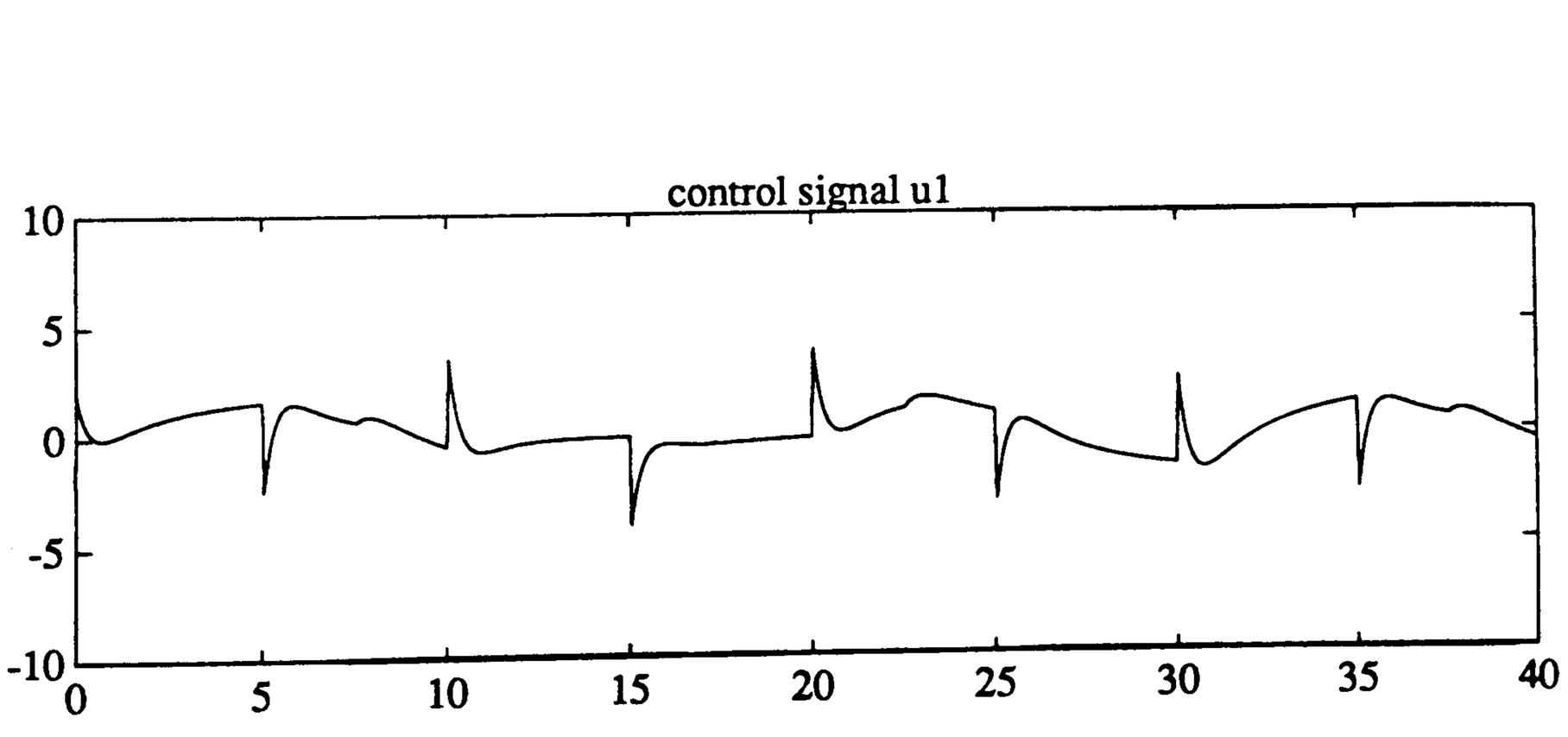


Figure 5.8 Decoupling and exact model-following

As mentioned in section 5.3.3, if the system is decouplable but the system row relative orders are greater than unity, then it is still possible to obtain a very close model-following, as one of the poles of the transfer functions of the closed-loop transfer matrix will be at the position of the corresponding model pole and the other poles can be placed faraway from the imaginary axis by the proper choice of T_2 (the smaller the T_2, the more faraway the poles). To illustrate this relationship example 1 was simulated with the following modifications:

$$B_{11}(s) = 2 \quad \text{instead of} \quad s+2$$

$$B_{22}(s) = 2 \quad \text{instead of} \quad s+1$$

Note that now $\rho_{r1} = \rho_{r2} = 2$. In the simulation the parameters and models were the same as the previous simulation except $\underline{N}_u$. Control orders were rechosen to satisfy the condition $N_{u_j} = N_{u_j}^{\max}$, that is $\underline{N}_u = [\,4 \;\; 4\,]$. The simulation result is given in figure 5.9. In the figure dashed line corresponds to the model outputs. As seen from the figure, a very close model-following is obtained. Notice the impulsive behaviour of the control signals at setpoint step changes. This is because the models have lower relative orders than the corresponding system row relative orders.

The above decoupling and model-following properties become less accurate as N_{u_j} reduces from $N_{u_j}^{\max}$. However, for large control orders interactions are very small and the model-following relationship is close enough so that it still can be considered as a model-following type control.

5.4.2. Adaptive Simulations

In general, any control method can be combined with a recursive estimation algorithm to give its self-tuning version. Here, we will consider the continuous-time least-squares algorithm given in chapter 2. The multivariable estimation problem will be formulated as follows. Each output of the multivariable system will be written in a linear in the parameters form

$$y_i(t) = \underline{X}_i^T \, \underline{\theta}_i \tag{5.131}$$

where $\underline{\theta}_i$ is the parameter vector containing the coefficients of the polynomials $B_{i1}(s)$, $B_{i2}(s)$, ... , $B_{im}(s)$, $A_i(s)$ and $\underline{X}_i^T$ is the corresponding filtered data vector.† The degree of the filter polynomial

† The equations leading to the above linear in the parameters form (eqn. 5.131) are very similar to the scalar case and will not be given here, for the details see chapter 2.

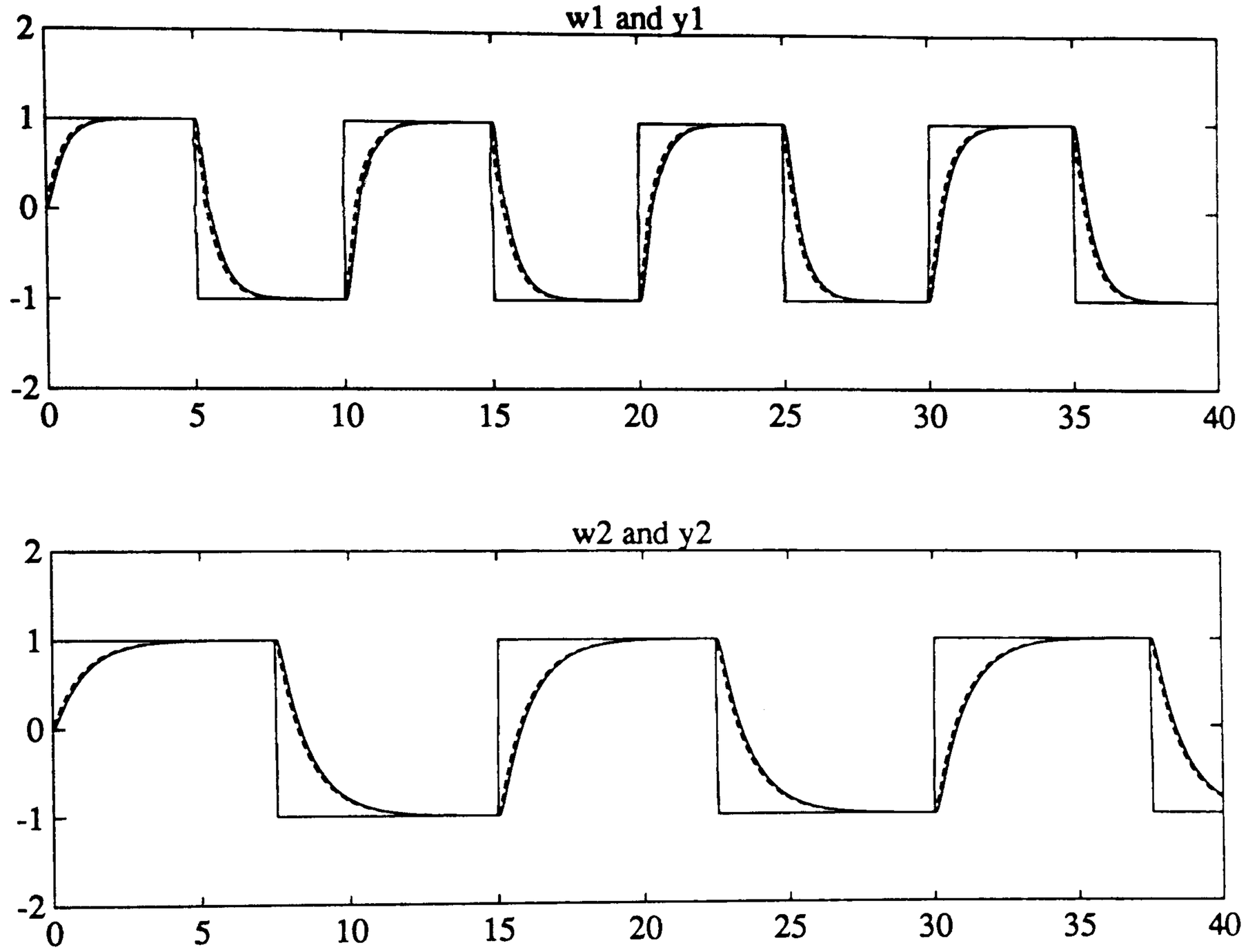


Figure 5.9 Model-following control

$C_{fi}(s)$ is equal to the degree of $A_i(s)$ and both polynomials $(C_{fi}(s), A_i(s))$ are assumed to be monic. Then there will be as many estimators running in parallel as there are outputs. The following are common in the simulations:

1- sample interval is 0.05 sec,

2- $\underline{R}_n(s) = [\ 1 \quad 1\ ]$,

3- $\underline{N}_y = [\ 6 \quad 6\ ]$

4- $T_1 = 0$ and $\Lambda = 0$,

5- the control signals are limited with ± 10000,

6- all simulations start with a set of wrong parameters,

7- estimator parameters forgetting factor and initial inverse covariance are 0.2 and 0.0001 I respectively (for all the estimators).

As a simple example, we first simulated example 1 with the following parameters:

$\underline{R}_d(s) = [\ 1 \quad 1\ ]$

$\underline{N}_u = [\ 1 \quad 1\ ]$

$T_2 = 1$

$\underline{C}_f(s) = [\ s^2+s+1 \quad s^2+s+1\ ]$

The closed-loop system outputs, control inputs and setpoints are given in figure 5.10(a) and the corresponding estimated parameters in figure 5.10(b). As can be seen from the figures, parameter estimates rapidly converge to their true values and despite the initial variations in the estimates, the initial output responses are very smooth. If quicker parameter convergence is desired, this can be accomplished by choosing smaller initial inverse covariance. However, this results in larger variations in the parameters initially and may results in more overshoots at the outputs in the tuning phase.

To see the performance of the self-tuning MCGPC when it is subject to stochastic disturbances, we also simulated example 1 with added random disturbances (Gaussian white noise with zero mean and a standard deviation of 0.1) direct at the outputs. In the simulation the design

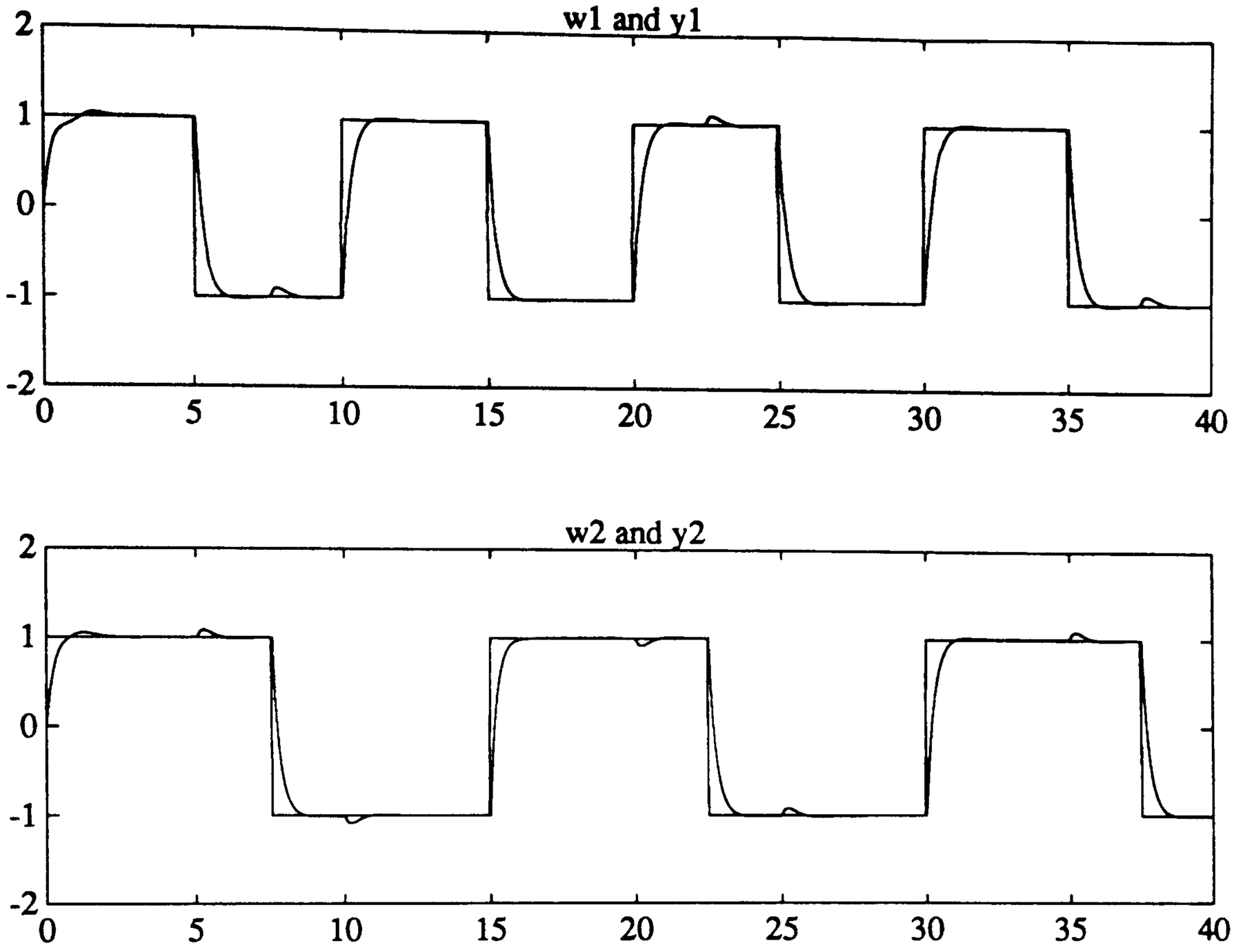


Figure 5.10(a) Self-tuning control of example 1

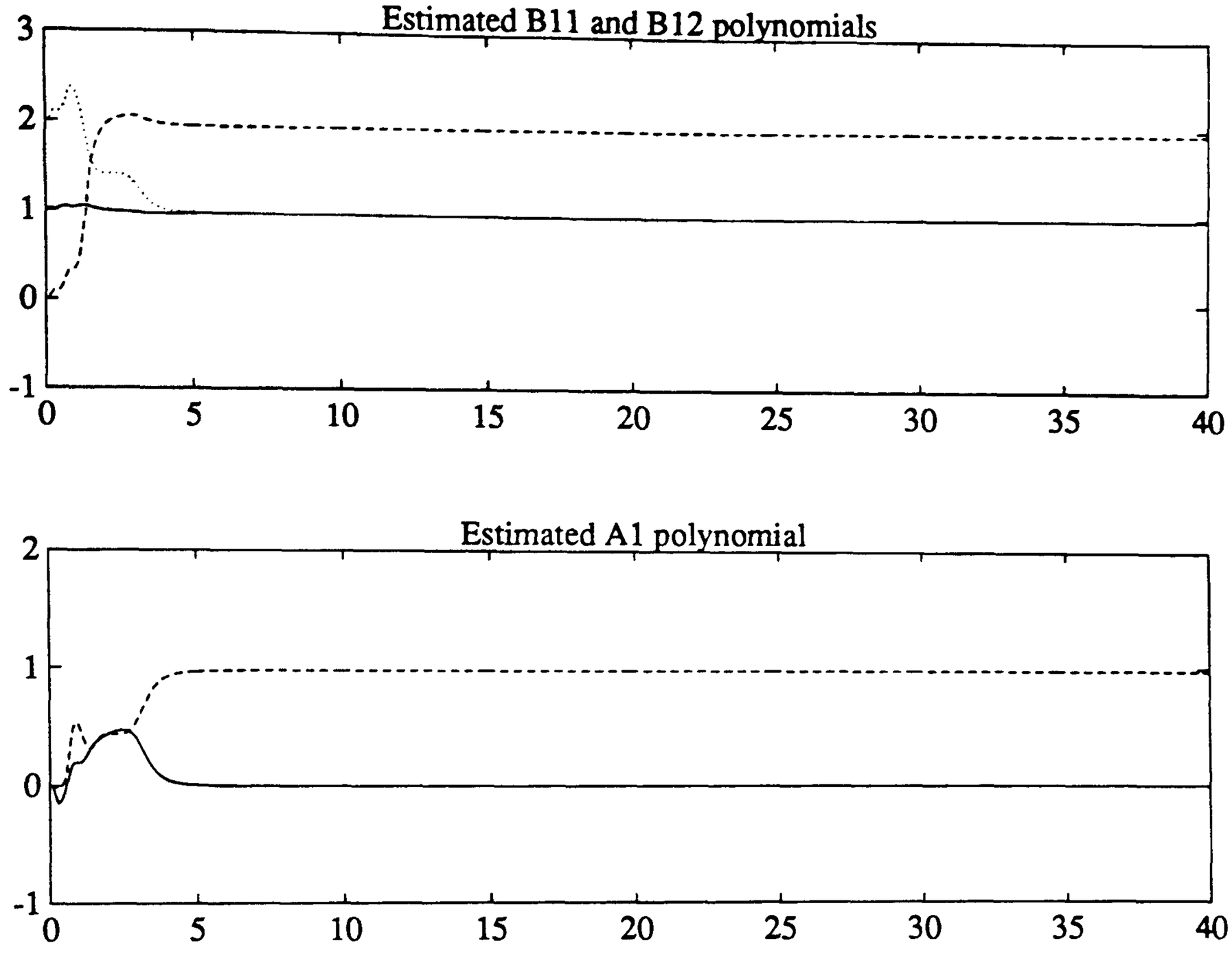


Figure 5.10(b) Self-tuning control of example 1

parameters were exactly the same as in the previous simulation. The simulation results are given in figure 5.11(a) and 5.11(b). Note that despite the noise, parameter estimates and the control performance are very good.

Second simulation was example 3. In this simulation the aim is to show the performance of the MCGPC on a more complicated system. Note that system has 2 stable, 3 unstable and a pair of complex poles on the imaginary axis. It is also non-minimum phase having two zeros at $s_1 = 4.7417$ and $s_2 = -2.7417$. In the simulation the following parameters were used:

$$\underline{R}_d(s) = [\ 1.5s+1 \quad s+1\ ]$$

$$\underline{N}_u = [\ 2 \quad 2\ ]$$

$$T_2 = 2$$

$$\underline{C}_f(s) = [\ s^4+2s^3+3s^2+2s+1 \quad s^3+2s^2+2s+1\ ]$$

The closed-loop outputs, control inputs and setpoints are given in figure 5.12(a) and the parameter estimates in figure 5.12(b). As can be seen from the figures, in the tuning phase there are large overshoots and large variations in the control signals due the wrong parameters. Once the estimates converge, a good control with very little interactions is obtained despite the fact that system is highly unstable and non-minimum phase. Note also that there is no control weighting.

Finally, example 4 was simulated as an example of the control of nonsquare systems. In the simulation the parameters used were:

$$\underline{R}_d(s) = [\ 1 \quad 1\ ]$$

$$\underline{N}_u = [\ 1 \quad 1 \quad 0\ ]$$

$$T_2 = 2$$

$$\underline{C}_f(s) = [\ s^2+2s+1 \quad s^2+2s+1\ ]$$

The simulation results are given in figure 5.13(a) and 5.13(b). Here again after initial tuning phase, the control performance is very good and there are almost no interactions.

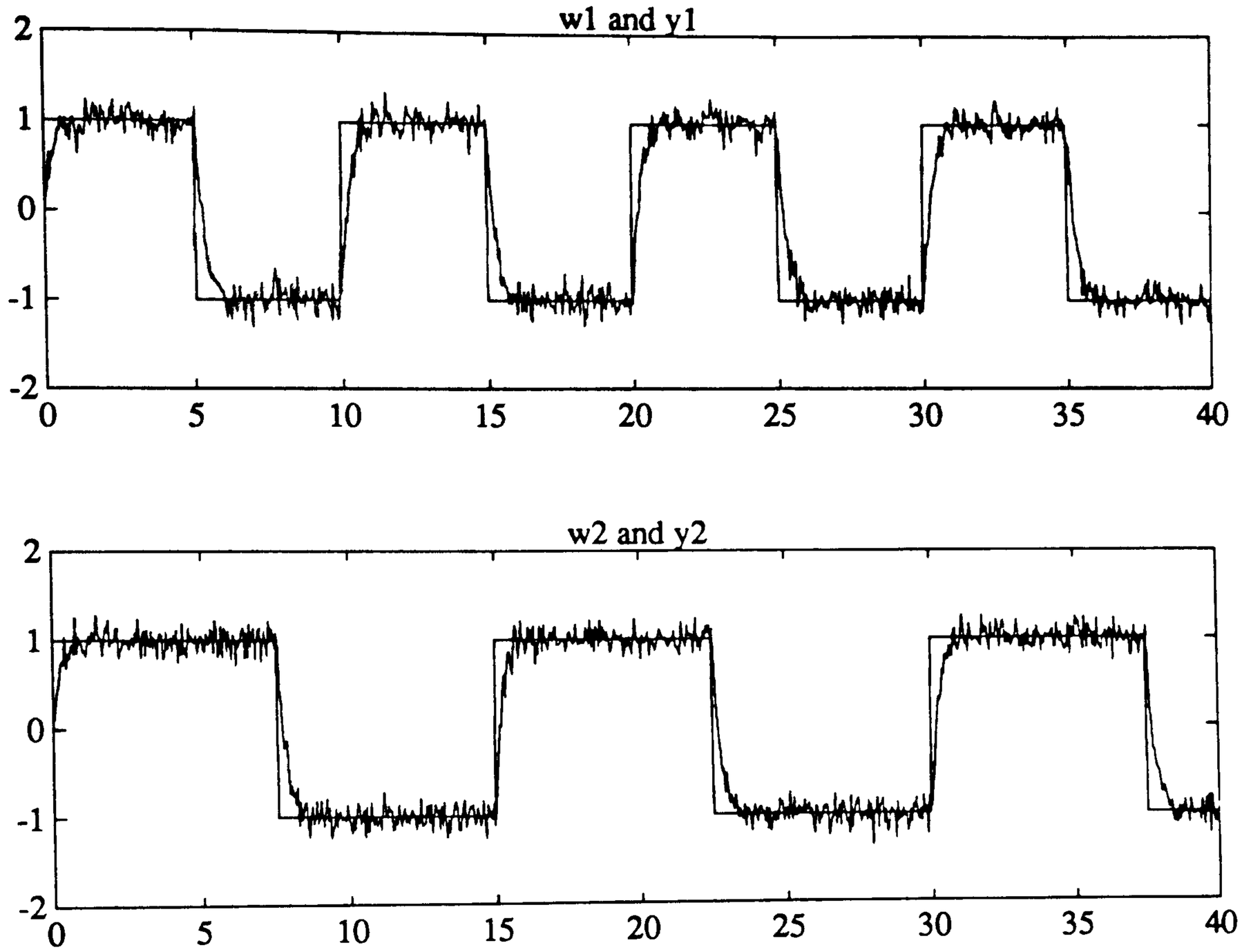


Figure 5.11(a) Self-tuning control in the presence of noise

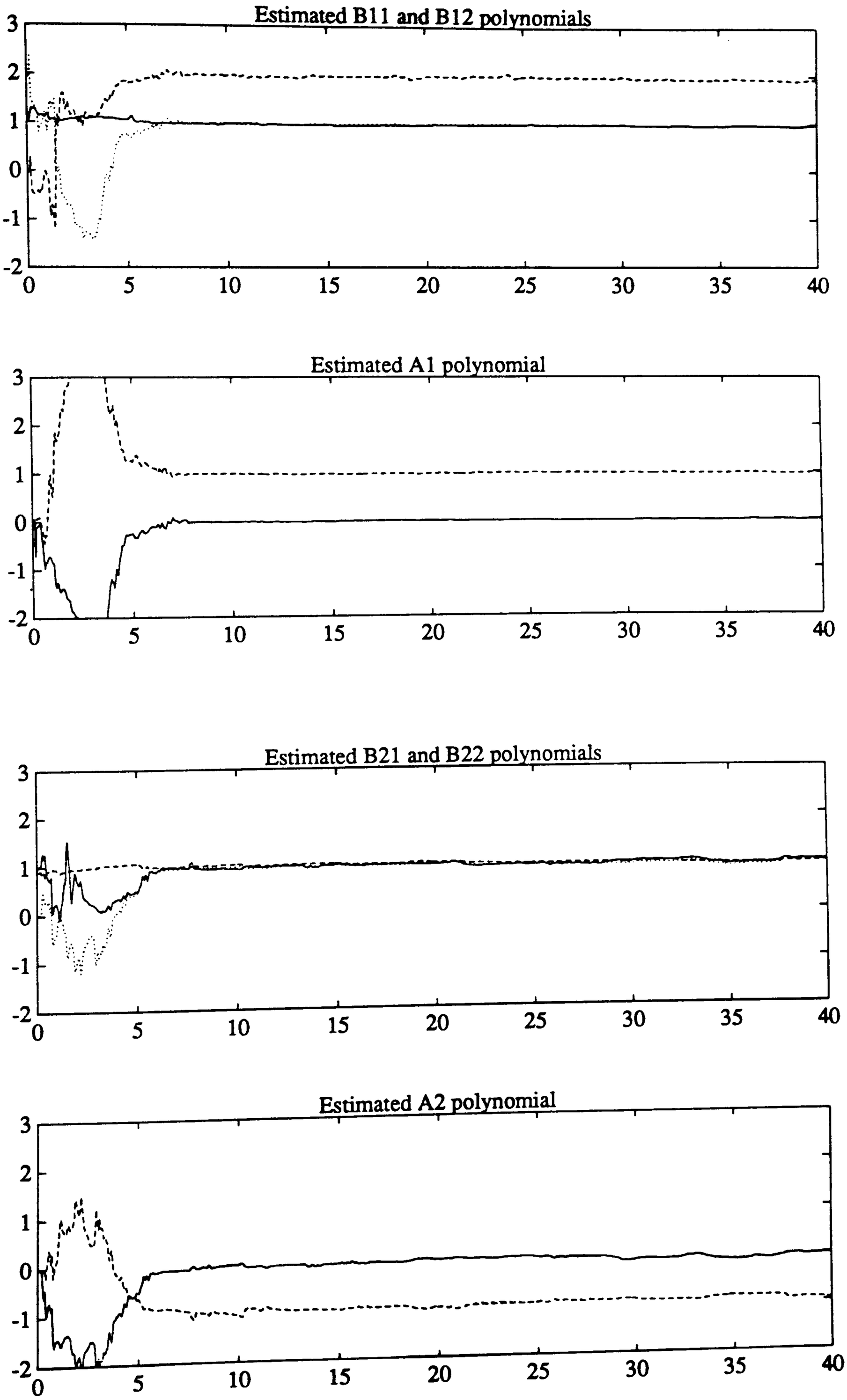


Figure 5.11(b) Self-tuning control in the presence of noise

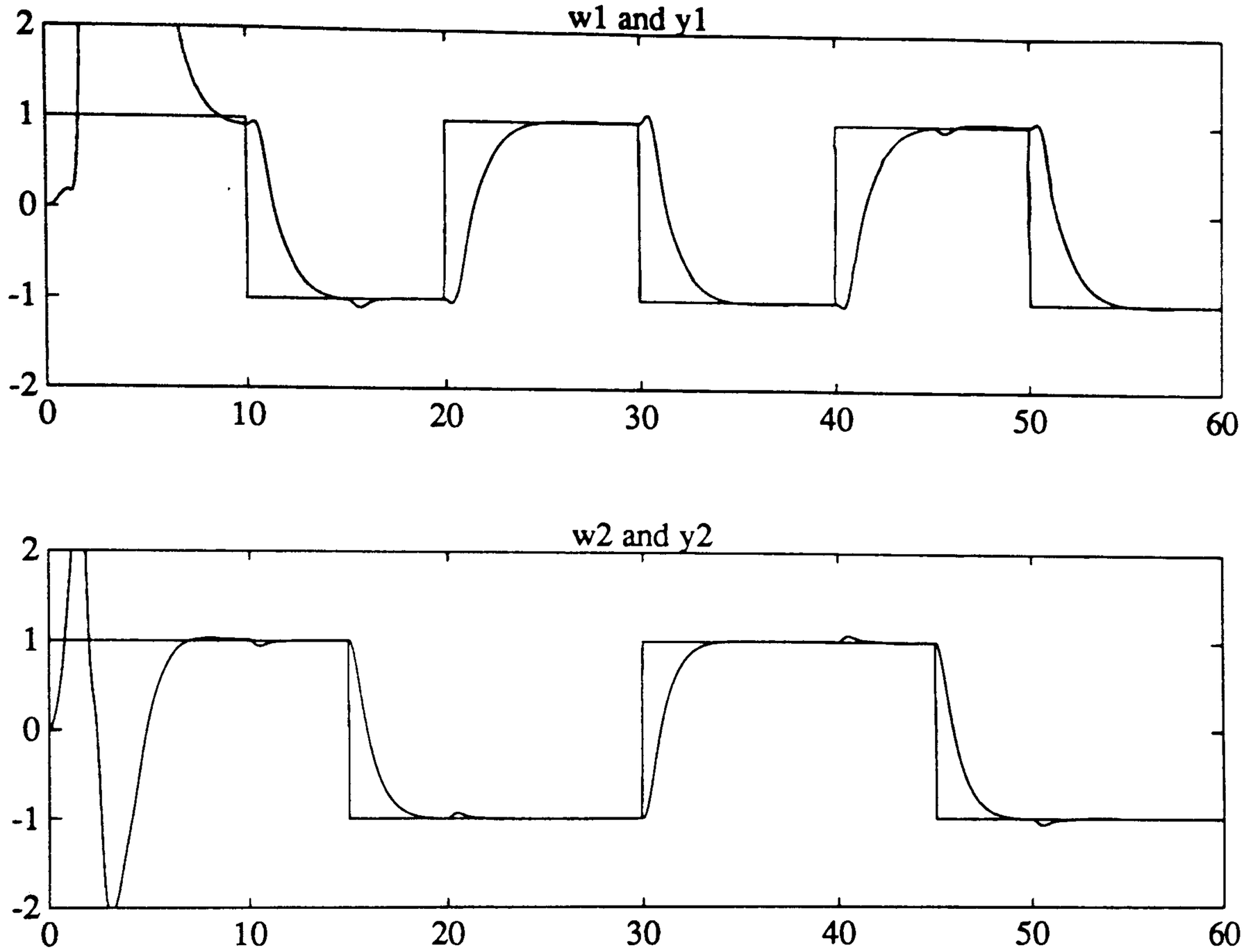


Figure 5.12(a) Self-tuning control of example 3

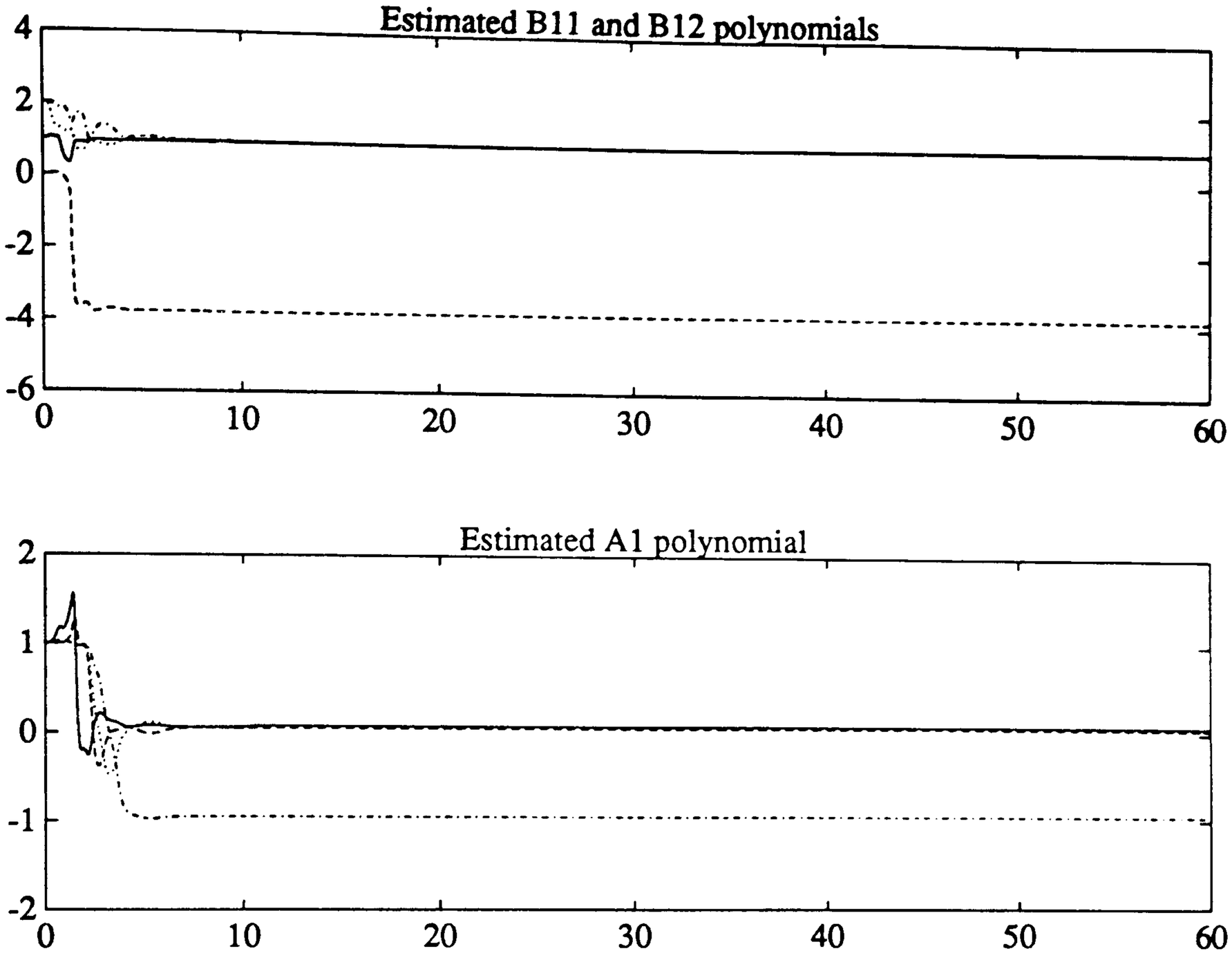


Figure 5.12(b) Self-tuning control of example 3

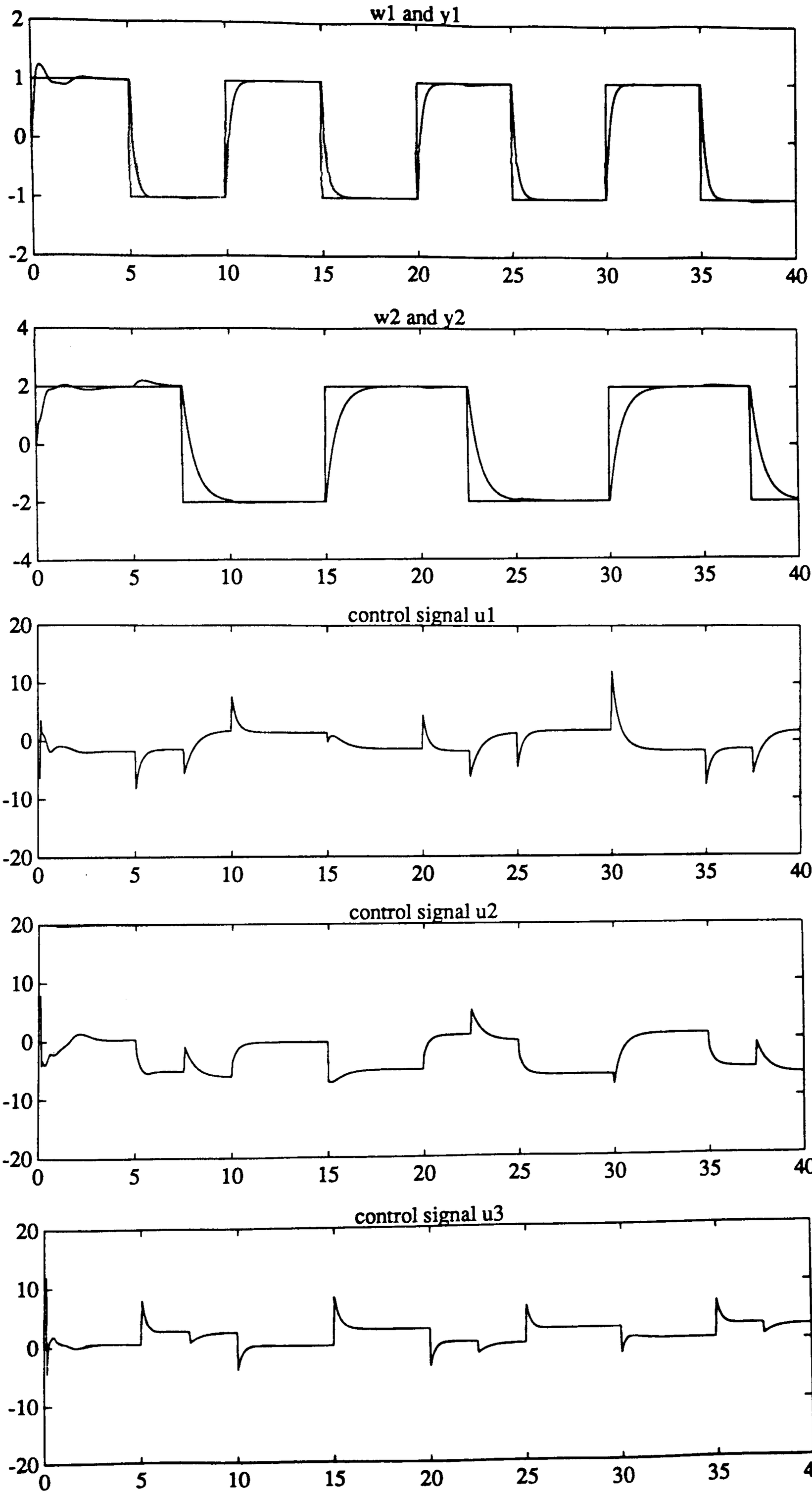


Figure 5.13(a) Self-tuning control of a nonsquare system

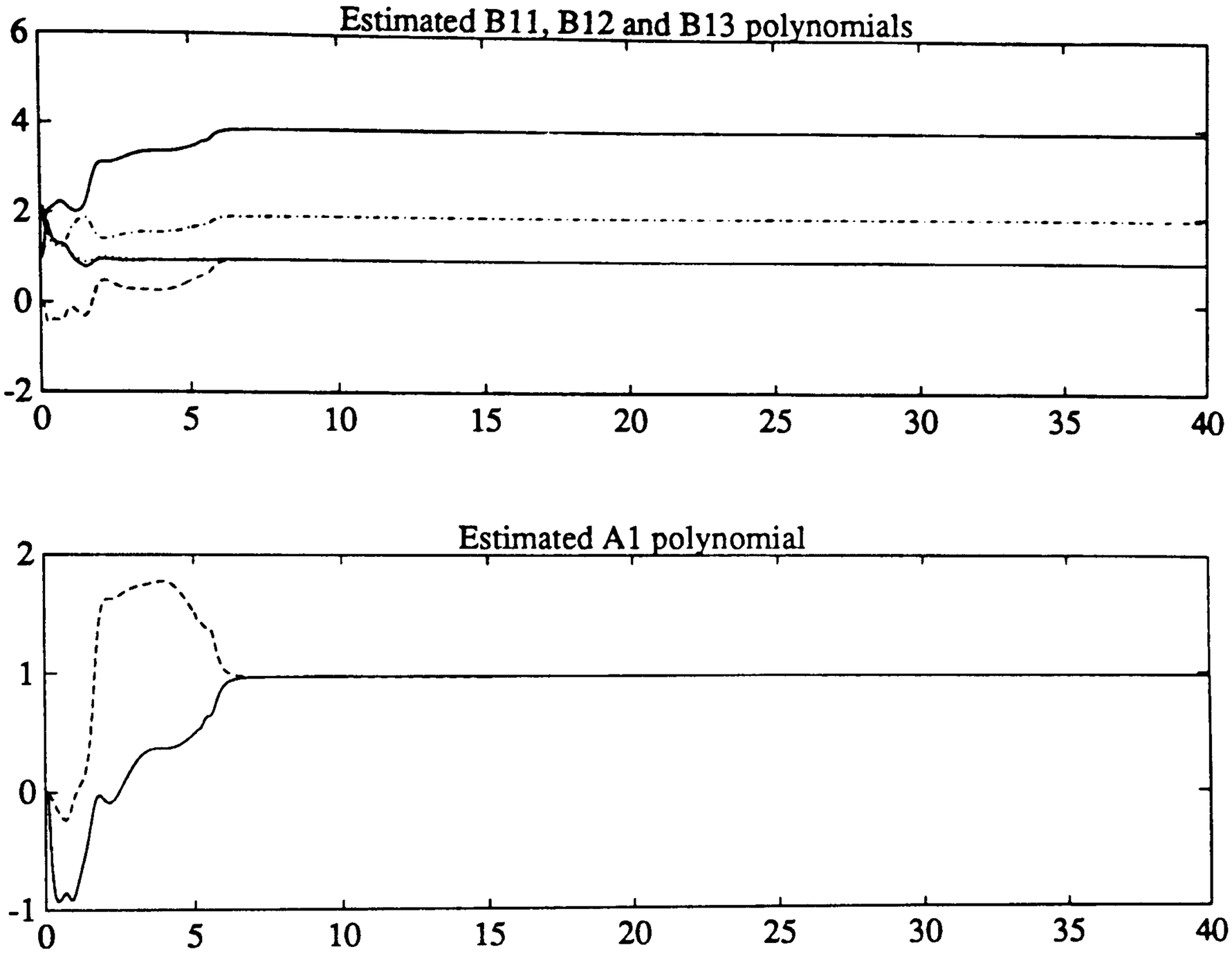


Figure 5.13(b) Self-tuning control of a nonsquare system

5.5. CONCLUSIONS

In this chapter a multivariable generalization of the CGPC (MCGPC) is presented and the resulting closed-loop system is analysed in some detail. In the development of the algorithm, a general ($p \times m$) system is considered unlike the most of the self-tuning multivariable algorithms which only consider the square systems. The conditions for decoupling and exact model-following are established. The relation with the LQ control is pointed out and the argument is supported by a numerical example. It is also shown that the MCGPC algorithm is capable of controlling non-minimum phase systems with zero control weighting. The effects of different design parameters and the performance of the algorithm are illustrated by simulations.

Although, for simplicity, we did not consider here, it is possible to generalize the algorithm further by using an auxiliary output approach and by incorporating a dynamic control weighting. This will possibly enable us to consider the pole-placement control in the MCGPC framework and will possibly improve the performance and robustness of the algorithm further.

On the basis of the analysis and simulations given in this chapter, our conclusion is that the MCGPC is a high performance algorithm and it seems to be very suitable for the self-tuning applications for a large variety of multivariable systems.

CHAPTER 6

CONCLUSIONS AND FURTHER WORK

In this thesis, some new continuous-time self-tuning algorithms are presented. These are: emulator-based relay control (Demircioglu, 1988), continuous-time generalized predictive control (CGPC) (Gawthrop, 1989) and its relay version. The CGPC is also extended to the multivariable systems. These methods are combined with a continuous-time version of the well-known discrete recursive least squares algorithm to give their self-tuning versions. The thesis mainly concentrates on the development and closed-loop analysis of the proposed underlying control methods, rather than stability, convergence or robustness analysis of the corresponding self-tuning algorithms.

The emulator-based relay control is described in chapter 3. The method removes the need to know the system states to implement the switching surface, unlike variable structure design. The switching surface is implemented by replacing the unrealisable output derivatives by their emulated values. It is shown that emulator-based control and its relay version are equivalent when the relay operates in the sliding mode. Thus, the control methods obtained in the first case, such as model-reference, pole-placement, predictive control and their detuned versions, can also be obtained in the second case. It was observed in a real experiment that (level control of a two cascaded tank), the relay self-tuning controller performed better than the corresponding self-tuning controller without a relay. The reason for this is that the pump is strongly nonlinear. This has no effect on the relay control as only two points on the nonlinear characteristic are used; but the usual self-tuning controller has a strongly nonlinear system to identify and control. This is probably true for many cases where the actuator has a nonlinear characteristic.

The CGPC, a continuous-time version of the discrete-time GPC developed by Clarke et al (Clarke, 1987), is presented in chapter 4. It has very similar properties to those of the discrete-time GPC although there are important differences in the way in which output prediction and control constraining is accomplished. The CGPC design parameters, as with the GPC, are directly related to the closed-loop output response. For example, the effect of the maximum prediction horizon T_2 is that the larger the T_2 the slower the output response and vice versa. The effect of the control order N_u is opposite, that is a larger N_u corresponds to a faster output response whereas a smaller N_u a slow output response. This feature of the CGPC makes it easy for the user to adjust the output response as desired. In general, it is advisable to keep the control order N_u as to be zero in order not to increase the computational burden and adjust the output response by using T_1 (minimum prediction horizon), T_2 and the reference model R_n/R_d. In this way, it is possible to obtain a good control performance for a large variety of systems, but for the more complex systems (e.g. at the same time higher order, open-loop unstable and non-minimum phase) an increased value of N_u is needed. The reference model R_n/R_d, apart from adjusting the output transients, can also be used to obtain model following type control (sometimes exact model following) with a large N_u.

The CGPC, unlike pole-placement control, does not suffer from the ill effects of the common factors due to overparameterization. It can control non-minimum phase systems without any difficulty even if the control weighting is zero. In addition, it is shown that time delay systems can be controlled in a similar way to the GPC by increasing the system order in order to accommodate a rational approximation to the delay. Therefore it is robust against time delay variations. It is also shown that a special setting of the CGPC parameters result in LQ control. The CGPC method is further extended to include some design transfer functions, which enables us to consider the model-reference and pole-placement control in the CGPC framework and also enhances the properties of the algorithm. As a result, the CGPC has all the potential and power of the GPC, without having its drawbacks due to the discrete-time formulation.

The relay version of the CGPC is based on the ideas of chapter 2. It is shown again that the CGPC and its relay version becomes equivalent when the relay operates in the sliding mode.

Hence, the relay-CGPC can be seen as an implementation of the CGPC control law using the switching control.

The CGPC ideas are extended to the multivariable systems in chapter 5. The method, multivariable CGPC (MCGPC), is developed for a general $p \times m$ multivariable system, unlike most of the multivariable self-tuning which only consider the square systems. The MCGPC has similar design parameters to those of the scalar CGPC. In particular, it has different control orders and reference models for each inputs and outputs respectively. This is important as it enables us to adjust the closed-loop system outputs independently, to some extent. The interactions can also be reduced by using these parameters, for example larger control orders result in less interactions. It is shown that certain setting of the MCGPC parameters leads to a decoupled closed-loop system, if the system is decouplable by a state feedback alone. Further, it is shown that if the system is decouplable, it also possible to obtain model following control. In addition, it is argued that LQ control can be considered in the MCGPC framework. Moreover, as in the scalar case, over-parametrization (non-minimum realization) and non-minimum phase systems are not a problem for the MCGPC.

We will end the thesis by suggesting some possible further research areas. These are:

1- Extension of the relay self-tuning ideas to multivariable systems for both the EBC and CGPC strategies;

2- Extension of the MCGPC further by using an auxiliary output approach and incorporating a dynamic control weighting into the method;

3- Convergence, stability and robustness analysis of the algorithms; in particular it will be interesting to see the effect of the dynamic control weighting $Q(s)$ on the robustness of the CGPC.

4- Industrial applications of the algorithms;

5- Comparing relative performance, advantages and disadvantages of the discrete and continuous-time versions of the methods by both simulations and practical applications.

REFERENCES

Allidina, A.Y. and Hughes, F.M. (1980): "Generalised self-tuning controller with pole assignment," *IEE Proc. Pt. D* **127**(1) pp. 15-18.

Astrom, K.J. (1970): *Introduction to stochastic control theory*, Academic Press., New York

Astrom, K.J. and Eykhoff, P. (1971): "System identification, a survey," *Automatica* **7** pp. 123-169.

Astrom, K.J. and Wittenmark, B. (1973): "On self-tuning regulators," *Automatica* **9** pp. 185-199.

Astrom, K.J., Borisson, U., Ljung, L., and Wittenmark, B. (1977): "Theory and applications of self-tuning regulators," *Automatica* **13** pp. 457-476.

Astrom, K.J. and Wittenmark, B. (1980): "Self-tuning controllers based on pole-zero placement," *IEE Proc. Pt. D* **127** pp. 120-130.

Astrom, K.J. (1983): "Theory and applications of adaptive control - A survey," *Automatica* **19**(5) pp. 471-486.

Astrom, K.J., Hagander, P., and Sternby, J. (1984): "Zeros of sampled systems," *Automatica* **20**(1) pp. 31-38.

Astrom, K.J. and Wittenmark, B. (1984): *Computer controlled systems*, Prentice Hall.

Astrom, K.J. and Wittenmark, B. (1984): "Practical Issues in the Implementation of Self-tuning Control," *Automatica* **20**(5) pp. 595-605.

Astrom, K.J. (1987): "Adaptive feedback control," *Proc. IEEE* **75**(2) pp. 185-217.

Astrom, K.J. and Wittenmark, B. (1989): *Adaptive Control*, Addison Wesley.

Atherton, D.P. (1982): *Nonlinear control engineering*, Van nostrand reinhold.

Besharati-Rad, A. (1988): "Identification and Adaptive Control of Retarded Systems - A Continuous-time Approach," Ph.D Thesis, School of Engineering and Applied Sciences., University of Sussex

Bezanson, L.W. and Harris, S.L. (1984): "State-space design of multivariable self-tuning regulators," *Int. J. Control* **39**(2) pp. 395-411.

Bierman, G.J. (1977): *Factorization methods for discrete sequential estimation*, Academic Press., New York

Borisson, U. and Wittenmark, B. (1974): "An industrial application of a self-tuning regulator," in *IFAC Symposium on digital computer applications to process control*, , Zurich.

Borisson, U. and Syding, R. (1976): "Self-tuning control of an ore crusher," *Automatica* **12** pp. 1-7.

Borisson, U. (1979): "Self-tuning regulators for a class of multivariable systems," *Automatica* **15** pp. 209-215..

Chen, C.C. and Shaw, L. (1982): "On receding horizon feedback control," *Automatica* **18**(3) pp. 349-352.

Clarke, D.W and Gawthrop, P.J. (1975): "Self-tuning controller," *IEE Proc. Pt. D* **122**(9) pp. 929-934.

Clarke, D.W and Gawthrop, P.J. (1979): "Self-tuning control," *IEE Proc. Pt. D* **126**(6) pp. 633-640.

Clarke, D.W. (1982): "Model following and pole-placement self-tuners," *Opt. Control App. and Methods* **3** pp. 323-335.

Clarke, D.W., Hodgson, A.J.F., and Tuffs, P.S. (1983): "Offset problem and k-incremental predictors in self-tuning control," *IEE Proc. Pt. D* **130**(5) pp. 217-225.

Clarke, D.W. (1984): "Self-tuning control of nonminimum-phase systems," *Automatica* **20**(5) pp. 501-518.

Clarke, D.W., Mohtadi, C., and Tuffs, P.S. (1984): "Generalized predictive control part 1 and part 2," OUEL report, No. 1555/84 and 1557/84.

Clarke, D.W., Kanjilal, P.P., and Mohtadi, C. (1985): "A generalised LQG approach to self-tuning control. Part 1: Aspects of design," *Int. J. Control* **41**(6) pp. 1509-1523.

Clarke, D.W., Tuffs, P.S., and Mohtadi, C. (1985): "Self-tuning control of a difficult process," in *7th IFAC Syposium of identification and system parameter estimation*, , York, U.K.

Clarke, D.W., Mohtadi, C., and Tuffs, P.S. (1987): "Generalised Predictive Control. Part 1 : the basic algorithm and Part 2: extensions and interpretations," *Automatica* **23**(2) pp. 137-160.

Clarke, D.W. and Zhang, L. (1987): "Long-range predictive control using weighting-sequence models," *IEE Proc. Pt. D* **134**(3) pp. 187-195.

Clarke, D.W. (April 1988): "Application of generalised predictive control to industrial processes," *IEEE Control Systems Magazine*, pp. 49-55.

Cloud, D.J. and Kouvaritakis, B. (1988): "Characteristic decomposition and the multivariable generalisation of predictive self-tuning control," *IEE Proc. Pt. D* **135**(3) pp. 165-181.

Cutler, C.R. and Ramaker, B.L. (1980): "Dynamic matrix control: a computer control algorithm," in *Joint Automatic Control Conf.*, , San Francisco, U.S.A.

Demircioglu, H. and Gawthrop, P.J. (1988): "Continuous-time relay self-tuning control," *Int. J. Control* **47**(4) pp. 1061-1080.

Dugard, L., Goodwin, G.C., and De Souza, C.E. (1983): "Prior knowledge in model reference adaptive control of multi-input multi-output systems," in *22nd CDC Conference*, , San Antonio, U.S.A.

Dugard, L., Goodwin, G.C., and Xianya, X. (1984): "The role of the interactor matrix in multivariable stochastic adaptive control," *Automatica* **20**(5) pp. 701-709.

Egardt, B. (1979a): "Unification of some continuous-time adaptive control schemes," *IEEE Trans. on Auto. Control* **AC-24** pp. 588-592.

Egardt, B. (1979b): *Stability of adaptive controllers*, Springer.

Elliott, H. and Wolovich, W.A. (1982): "A parameter adaptive control structure for linear multivariable systems," *IEEE Trans. on Auto. Control* **AC-27**(2) pp. 340-352.

Elliott, H. (1982a): "Hybrid adaptive control of continuous time systems," *IEEE Trans. on Auto. Control* **AC-27**(2) pp. 419-426.

Elliott, H. (1982b): "Direct adaptive pole placement with application to nonminimum phase systems," *IEEE Trans. on Auto. Control* **AC-27**(3) pp. 720-721.

Elliott, H. and Wolovich, W.A. (1984): "Arbitrary adaptive pole placement for linear multivariable systems," *IEEE Trans. on Auto. Control* **AC-29**(3) pp. 221-229.

Elliott, H. and Wolovich, W.A. (1984): "Parameterization issues in multivariable adaptive control," *Automatica* **20**(5) pp. 533-545.

Flugge-Lotz, I. (1953): *Discontinuous automatic control*, Princeton University press.

Fortescue, T.R, Kershenbaum, L.S, and Ydstie, B.E (1981): "Implementation of self-tuning regulators with variable forgetting factors," *Automatica* **17**(6) pp. 831-835.

Gawthrop, P.J. (1977): "Some interpretations of the self-tuning controller," *IEE Proc. Pt. D* **124**(10) pp. 889-894.

Gawthrop, P.J. and Clarke, D.W. (1980): "Hybrid self-tuning control and its interpretation," in *Proceedings of 3rd IMA Conference on Control Theory, Academic Press*,

Gawthrop, P.J. (1980): "Hybrid self-tuning control," *IEE Proc. Pt. D* **127**(5) pp. 229-236.

Gawthrop, P.J. (1982): "Self-tuning PI and PID Controllers," in *Proceedings of the IEEE conference on "Applications of Adaptive and Multivariable Control"*, , Hull, U.K.

Gawthrop, P.J. (1982): "A continuous-time approach to discrete-time self-tuning control," *Opt. Control App. and Methods* **3**(4) pp. 399-414.

Gawthrop, P.J. (1985): "Robust self-tuning control of n-input n-output systems," in *Preprints of the 7th IFAC Symposium on Identification and System Parameter Identification*, , York, U.K.

Gawthrop, P.J. (1986): "Self-tuning PID controllers: Algorithms and implementation," *IEEE Trans. on Auto. Control* **AC-31**(3) pp. 201-209.

Gawthrop, P.J. (July 1986): "Continuous-time self-tuning control - A Unified Approach.," in *Preprints of the 2nd IFAC Workshop on Adaptive Systems in Control and Signal Processing*, , Lund, Sweden.

Gawthrop, P.J. (1987): *Continuous-time Self-tuning Control*, Research Studies Press., Letchworth, England.

Gawthrop, P.J. (1987a): "Robust stability of a continuous-time self-tuning controller," *Int. J. of Adaptive Control and Signal Processing* **1** pp. 31-48.

Gawthrop, P.J. and Demircioglu, H. (1988): "Continuous-time Generalised Predictive Control (CGPC)," Control Engineering Report 88.2., University of Glasgow, Department of Mechanical Engineering

Gawthrop, P.J. and Demircioglu, H. (April 1989): "Continuous-time generalised predictive control," pp. 123-128 in *Preprints of IFAC symposium on adaptive systems in control and signal processing*, , Glasgow, UK.

Goodwin, G.C., Ramadge, P.J., and Caines, P.E. (1980): "Discrete-time multivariable adaptive control," *IEEE Trans. on Auto. Control* **AC-25**(3) pp. 449-456.

Goodwin, G.C. and Long, R.S. (1980): "Generalization of results on multivariable adaptive control," *IEEE Trans. on Auto. Control* **AC-25**(6) pp. 1241-1245.

Goodwin, G.C. and Sin, K.S. (1984): *Adaptive filtering prediction and control*, Prentice-Hall., Englewood Cliffs, New Jersey, USA

Grimble, M.J. (1984): "Implicit and Explicit LQG Self-tuning Controllers," *Automatica* **20**(5) pp. 661-669.

Harris, C. and Billings, S. (1981): *Self-tuning and adaptive control - theory and applications*, Peter Peregrinus.

Hesketh, T. (1982): "State-space pole-placing self-tuning regulator using input-output values," *IEE Proc. Pt. D* **129**(4)

Horowitz, I., Smay, J., and Shapiro, A. (1974): "A Synthesis theory for self-oscillating adaptive systems," *Automatica* **10** pp. 381-392.

Kailath, T. (1980): *Linear Systems*, Prentice-Hall.

Kalman, R.E. (1958): "Design of a self-optimizing control system," *Trans. ASME* **80** p. 468.

Keviczky, L. and Kumar, K.S.P. (1981): "Multivariable self-tuning regulator with generalized cost-function," *Int. J. Control* **33**(5) pp. 913-921.

De Keyser, R.M.C. and Van Cauwenberghe, A.R. (1981): "A self-tuning multistep predictor application," *Automatica* **17**(1) pp. 167-174.

De Keyser, R.M.C. and Van Cauwenberghe, A.R. (1985): "Extended prediction self-adaptive control," in *IFAC Identification and System Parameter Estimation*, , York, U.K.

De Keyser, R.M.C., Van de Velde, Ph.G.A., and Dumortier, F.A.G. (1988): "A comparative study of self-adaptive long-range predictive control methods," *Automatica* **24**(2) pp. 149-163.

Koivo, H.N. (1980): "A multivariable self-tuning controller," *Automatica* **16** pp. 351-366.

Kwakernaak, H. and Sivan, R. (1972): *Linear optimal control systems*, Wiley.

Kwon, W.H. and Pearson, A.E. (1977): "A modified quatratic cost problem and feedback stabilization of a linear system," *IEEE Trans. on Auto. Control* **AC-22**(5)

Lam, K.P. (1980): "Implicit and explicit self-tuning controllers," D.Phil Thesis and OUEL Report No.1334/80., Oxford University

Lambert, E.P. (1987): "Process control applications of long-range prediction," D.Phil Thesis and OUEL Report No.1715/87., Oxford University

Lambert, M. (1987): "Adaptive control of flexible systems," D.Phil Thesis and OUEL Report No.1707/87., Oxford University

Lawson, C.L. and Hanson, R.J. (1974): *Solving Least Squares Problems*, Prentice-Hall.

Lelic, M.A. and Zarrop, M.B. (1987): "Generalized pole-placement selff-tuning controller, Part1. Basic algorithm," *Int. J. Control* **46**(2) pp. 547-568.

Ljung, L. and Soderstrom, T. (1983): *Theory and practice of recursive identification*, MIT press., London

Ljung, L. (1987): *System identification: theory for the user*, Prentice-Hall.

Longchamp, R. (1983): "Singular perturbation analysis of a receding horizon controller," *Automatica* **19**(3) pp. 303-308.

Marshall, J.E. (1979): *Control of time-delay systems*, Peter Peregrinus Ltd..

Mohtadi, C., Shah, S.L., and Clarke, D.W. (1986): "Generalized predictive control of multivariable systems," Report No. OUEL 1640/86., Oxford University, Department of Engineering Science

Mohtadi, C. (1987): "Advanced self-tuning algorithms," D.Phil Thesis and OUEL Report No.1689/87., Oxford University

Mosca, E., Zappa, G., and Manfredi, C. (1984): "Multistep horizon self-tuning controllers: the MUSMAR approach," in *IFAC 9th World Congress*, , Budapest, Hungary.

Nomikos, P.E. (1988): "Multivariable self-tuning controllers for industrial applications," D.Phil. Thesis., University of Sussex

Peterka, V (1970): "Adaptive digital regulation of noisy systems," in *IFAC Symposium on Identification and process parameter Estimation*, , Prague.

Peterka, V. (1984): "Predictor-based self-tuning control," *Automatica* **20**(1) pp. 39-50.

Prager, D.L. and Wellstead, P.E. (1980): "Multivariable pole-assignment self-tuning regulators," *IEE Proc. Pt. D* **128**(1) pp. 9-18.

Richalet, J., Rault, A., Testud, J.L., and Papon, J. (1978): "Model predictive heuristic control: applications to industrial processes," *Automatica* **14** pp. 413-428.

Rohrs, C.E., Valavani, L., Athans, M., and Stein, G. (1985): "Robustness of continuous-time adaptive control in the presence of unmoddeled dynamics," *IEEE Trans. on Auto. Control* **AC-30**(9) pp. 881-889.

Samson, C. (1982): "An adaptive LQ controller for nonminimum-phase systems," *Int. J. Control* **35**(1) pp. 1-28.

Selbuz, H. and Eldem, V. (1987): "Stabilizing and deadbeat properties of receding horizon controllers," *Int. J. Control* **45**(6) pp. 1931-1939.

Shieh, L.S., Wang, C.T., and Tsay, Y.T. (1983): "Fast suboptimal state-space self-tuner for linear stochastic multivariable systems," *IEE Proc. Pt. D* **130**(4)

Shieh, L.S., Bao, Y.L., and Chang, F.R. (1989): "State-space self-tuning regulators for general multivariable stochastic systems," *IEE Proc. Part D* **136**(1)

Singh, R.P. and Narendra, K.S. (1984): "Prior information in the design of multivariable adaptive controllers," *IEEE Trans. on Auto. Control* **AC-29**(12) pp. 1108-1111.

Sinha, N.K. and Puthenpura, S. (1985): "Choice of the sampling interval for the identification of

continuous-time systems from samples of input/output data," *IEE Proc. Pt. D* **132**(6) pp. 263-267.

Smith, O.J.M. (1959): "A controller to overcome dead-time," *ISA transactions* **6**(2) pp. 28-33.

De Souza, C.E., Goodwin, G.C., Mayne, D.Q., and Palaniswami, M. (1988): "An Adaptive Control Algorithm for Linear Systems having Unknown Time Delay," *Automatica* **24**(3) pp. 327-341.

Thomas, Y.A. (1975): "Linear quadratic optimal estimation and control with receding horizon," *Electronics Letter* **11**(1) pp. 19-21.

Tsypkin, Y.Z. (1984): *Relay control systems*, Cambridge university press.Traslated by Constanda, C.

Utkin, V.I. (1977): "Variable structure systems with sliding modes," *IEEE Trans. on Auto. Control* **AC-22**(2) pp. 212-222.

Utkin, V.I. (1978): *Sliding modes and their application in variable structure systems*, Mir Publishers Moscow.Translated by Parnakh, A.

Warwick, K. (1981): "Self-tuning regulators: a state-space approach," *Int. J. Control* **33**(5) pp. 839-858.

Wellstead, P.E., Edmunds, J.E., Prager, D., and Zanker, P. (1979): "Self-tuning pole/zero assignment regulators," *Int. J. Control.* **30**(1) pp. 1-26.

Wellstead, P.E., Prager, D., and Zanker, P. (1979): "Pole assignment self-tuning regulator," *IEE Proc. Pt. D* **126**(8) pp. 781-787..

Wieslander, J. and Wittenmark, B. (1971): "An approach to adaptive control using real time identification," *Automatica* **7** pp. 211-217.

Wolovich, W.A. (1974): *Linear multivariable systems*, Springer-Verlag.

Wolovich, W.A. and Falb, P.L. (1976): "Invariants and canonical forms under dynamic compensation," *SIAM J. Control and Optimization* **14**(6) pp. 996-1008.

Yaz, E. and Selbuz, H. (1984): "A note on the receding horizon control method," *Int. J. Control* **39**(4) pp. 853-855.

Ydstie, B.E. (1984): "Extended horizon adaptive control," in *IFAC 9th World Congress* , , Budapest, Hungary.

Young, K.K.D. (1978): "Design of variable structure model-following control systems," *IEEE Trans.* **AC-23** pp. 1079-1085.

Zinober, A.S.I (1981): "Controller design using the theory of variable structure systems," pp. 204-229 in *Self-tuning and adaptive control: Theory and applications*, ed. Harris, C.J. and Billings S.A., Peter peregrinus

Zinober, A.S.I., El-Ghezawi, O.M.E., and Billings, S.A. (1982): "Multivariable variable-structure adaptive model-following control systems," *IEE Proc. Pt. D* **129**(1) pp. 6-12.