CODEN: LUTFD2/(TFRT-5457)/1–43/(1992)

Adaptive Control of Piecewise Linear Systems

Martin Balters

Department of Automatic Control
Lund Institute of Technology
March 1992

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Master Thesis

Date of issue
March 1992

Document Number
CODEN: LUTFD2/(TFRT–5457)/1-43/(1991)

Author(s)
Martin Balters

Supervisor
Karl Johan Åström, LTH

Sponsoring organisation

Title and subtitle
Adaptive control of piecewise linear systems

Abstract

This thesis treats several problems in control of piecewise linear systems.

A parameter estimation algorithm for piecewise linear systems is presented. Some problems with the algorithm are discussed and suggestions with regard to the input requirements are made in order to overcome the problems encountered.

A self-tuning regulator for piecewise linear systems is presented, using pole placement design based on input-output models.

Key words
Adaptive control, Parameter estimation, Piecewise linear system, Time-out

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
43

Recipient's notes

Security classification

Preface

As a student at ETH Zürich I wrote this master thesis at the Department of Automatic Control at Lund Institute of Technology in Sweden. It has been in every sense an enriching experience to life in Sweden and I can really recommend to other students to do similar exchanges.

The exceptional helpfulness and competence of the people at the department have greatly contributed to the success of this work. I would like to take the opportunity to thank them all.

Especially I would like to thank my supervisor Professor Karl Johan Åström. He suggested the thesis subject and provided an excellent guidance during the work.

I also would like to thank the PhD students Henrik Olsson, Jörgen Malmborg, and Ulf Jönsson very much. They spent a lot of time to answer my innumerable questions and I have learned a lot during those discussions.

Table of contents

1. Introduction 5

2. Problem description 6
2.1 A prototype model 6
2.2 Experiences with the PI-control of the piecewise linear model . . 9

3. Recursive estimation of piecewise linear systems 13
3.1 Introduction to least squares 13
3.2 Estimation of piecewise linear systems 15
3.3 The "time-out" effect 23
3.4 Introducing a forgetting factor λ 28

4. Controller design for piecewise linear systems 31
4.1 Introduction to pole placement 31
4.2 Pole placement for piecewise linear systems 32

5. Conclusions 36

6. References 37

Appendix A 38
1. A brief introduction to SIMNON 38
2. The programs for the self-tuning regulator 38

1. Introduction

Research on adaptive control has so far mostly been focused on linear systems. Significant progress has been made in development of algorithms, analysis of stability, and convergence analysis. Adaptive techniques have also been transferred to industry and a number of adaptive controllers have appeared.

Lately there has also been significant progress in nonlinear control theory, e.g. feedback linearization and use of optimal control. A natural step is then to start investigations of nonlinear adaptive control. This can be expected to be quite a difficult task. As an intermediate step we can consider adaptive control of special classes of nonlinear systems. This is the task of this thesis where we consider adaptive control of piecewise linear systems.

The starting is a practical problem that frequently occurs in industry, namely control of systems with heating and cooling. Such problems are common in heating ventilating and air-conditioning systems. A practical solution has been to use conventional PID controllers with different settings for heating and cooling. Such systems are naturally more difficult to tune and adapt than conventional PID controllers. They are however not as difficult to deal with as general nonlinear systems.

The thesis is organized as follows. In chapter 2, a prototype model of a piecewise linear process is treated. Some difficulties when controlling it with a conventional PI-controller are pointed out. Chapter 3 investigates the identification of piecewise linear systems. A parameter estimation algorithm is developed and some problems with it are discussed. Suggestions are made how to overcome these problems. In chapter 4 a controller design algorithm for piecewise linear systems using pole placement based on input-output models is developed. The estimation algorithm and the controller design algorithm are combined in a self-tuning regulator to control piecewise linear systems adaptively. Conclusions and references are given in chapters 5 and 6 respectively. Some explanations about SIMNON and some program listings can be found in appendix A.

2. Problem description

Many industrial processes are nonlinear. A particular type of nonlinearity occurs in processes with heating and cooling. Such processes are common in HVAC (heating ventilation and airconditioning) systems, in plasticextruders etc. In these cases the processes can be approximated by piecewise linear systems, one system for heating and another for cooling. In this section a simplified process model with this property will be described.

Such processes are currently controlled by PID type of controllers which may have two settings, one for heating and one for cooling. It is a tedious process to tune the controllers because of the switches between operating regions. The parameters may also change in a way which makes adaptation necessary.

2.1 A prototype model

The features to be included in the prototype model are given below:

1. As mentioned earlier, the model should be piecewise linear.
2. It should represent a process that can be controlled by heating and cooling.
3. In this first model, heating and cooling should not be performed at the same time, as this would mean that we have to deal with a multiple input system.
4. Conventional controllers should give inferior performance while controlling this process, because only this justifies the design of a rather complicated adaptive controller.

One application we had in mind when looking for a good model was the following heating system (see Fig.2.1).

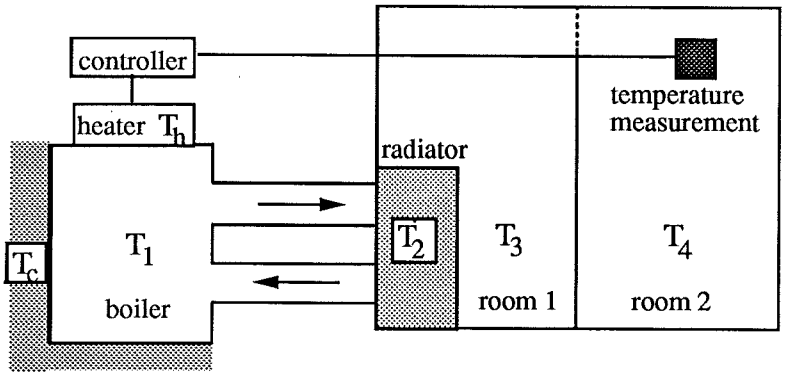

Figure 2.1 A Room temperature control system.

The goal is to control the temperature T_4 in room 2. In an abstract model, the boiler with temperature T_1, the radiator with temperature T_2, room 1 with temperature T_3, and room 2 with temperature T_4 can each be considered as

first order systems connected to each other. Together this is a fourth order system. With the heater set at a certain temperature T_h, the water temperature of the boiler can be raised. As the boiler looses heat to the environment which is at temperature T_c, this is a cooling process which is of course much slower than the heating process. In an abstraction the environmental temperature can be regarded same as the heater temperature as a control variable. It is assumed that heat losses of the rooms and the pipes can be neglected. It is also assumed that while heating, losses of the boiler can be neglected. Therefore, either heating or cooling have to be considered, but not both at the same time (this is exactly what was demanded above). A simplified model of the system could consist of four subsystems connected to each other in the following way:

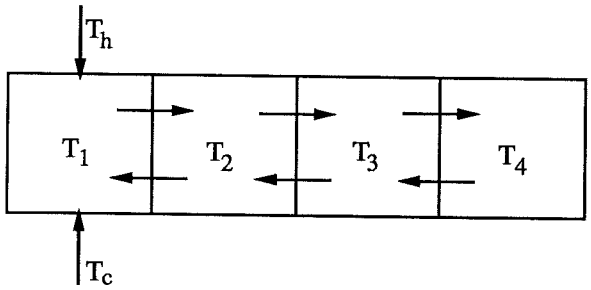

Figure 2.2 A simplified model of the heating system.

In a further simplification it is assumed that all subsystems are "equally sized" and that the heat transfer coefficients between them in both directions are equally k. The heat transfer equations are then the following:

1. Heating

$$\frac{dT_1}{dt} = g_h(T_h - T_1) + k(T_2 - T_1) \tag{2.1}$$

2. Cooling

$$\frac{dT_1}{dt} = g_c(T_c - T_1) + k(T_2 - T_1) \tag{2.2}$$

where g_h is the transfer coefficient for heating and g_c is the transfer coefficient for cooling.

3. Other heat transfers

$$\frac{dT_2}{dt} = k(T_1 - T_2) + k(T_3 - T_2) \tag{2.3}$$

$$\frac{dT_3}{dt} = k(T_2 - T_3) + k(T_4 - T_3) \tag{2.4}$$

$$\frac{dT_4}{dt} = k(T_3 - T_4) \tag{2.5}$$

In a final simplification it is assumed that $k = 1$. Replacing $\frac{d}{dt}$ with the Laplace-operator s, equations (2.1)-(2.5) are transformed to obtain

$$sT_1 = g_h(T_h - T_1) + T_2 - T_1 \tag{2.6}$$

$$sT_1 = g_c(T_c - T_1) + T_2 - T_1 \tag{2.7}$$

$$sT_2 = T_1 + T_3 - 2T_2 \tag{2.8}$$

$$sT_3 = T_2 + T_4 - 2T_3 \tag{2.9}$$

$$sT_4 = T_3 - T_4 \tag{2.10}$$

Eliminating T_1, T_2, and T_3 from the equations 2.6, 2.8, 2.9, and 2.10, the total transfer function $G(s)$ between the input $U(s)$ (which is T_h for heating and T_c for "cooling") and the output Y(s) (which is T_4) is received (here for the case of heating).

$$G(s) = \frac{Y(s)}{U(s)} = \frac{g_h}{s^4 + (6 + g_h)s^3 + (10 + 5g_h)s^2 + (4 + 6g_h)s + g_h} = \frac{T_4}{T_h} \tag{2.11}$$

The transfer function for cooling is obtained by replacing g_h with g_c. Figure 2.3 shows the block diagram of the model. The nonlinear block (see Fig.2.4) has a gain of g_h for positive and a gain of g_c for negative values of i(t).

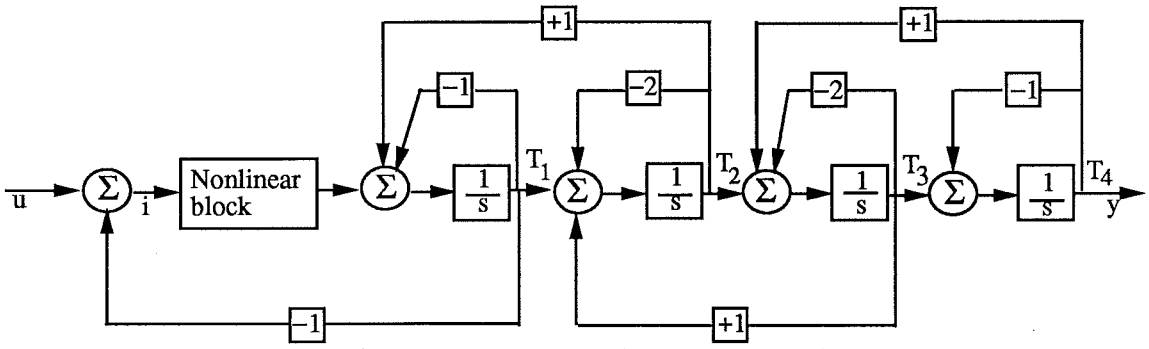

Figure 2.3 The block diagram of the nonlinear model.

Is this a good model? This model neglects a lot of properties of a real heating system. It also contains several very rough approximations. However it captures qualitatively all the key features that were described above.

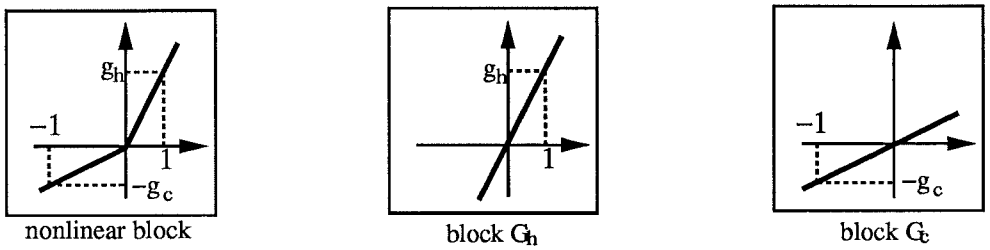

Figure 2.4 The nonlinear block and the linear blocks it was replaced with when optimizing the PI-controllers.

2.2 Experiences with the PI-control of the piecewise linear model

To learn more about the models properties and behaviour its control with different PI-controllers (see Fig.2.5) was simulated using the simulation language SIMNON [Elmqvist, Åström, Schönthal, and Wittenmark, 1990]. One PI-controller was optimized for the model when the nonlinear block is replaced with the linear block G_h. The other PI-controller was optimized for the case when the nonlinear block is replaced with the linear block G_c (see Fig.2.4).

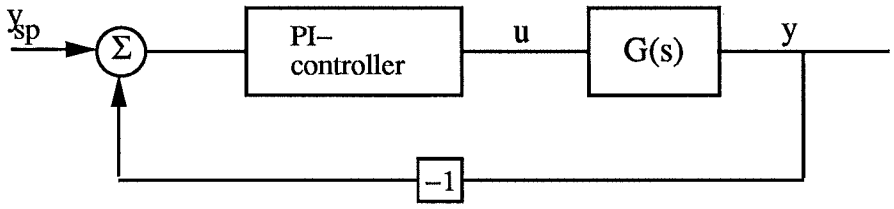

Figure 2.5 The closed control loop used for the experiment.

Each of these controllers then controlled the linear model they were designed for and the nonlinear model (see Fig.2.3). This means that four different combinations of controllers and process models were used (see Fig.2.6).

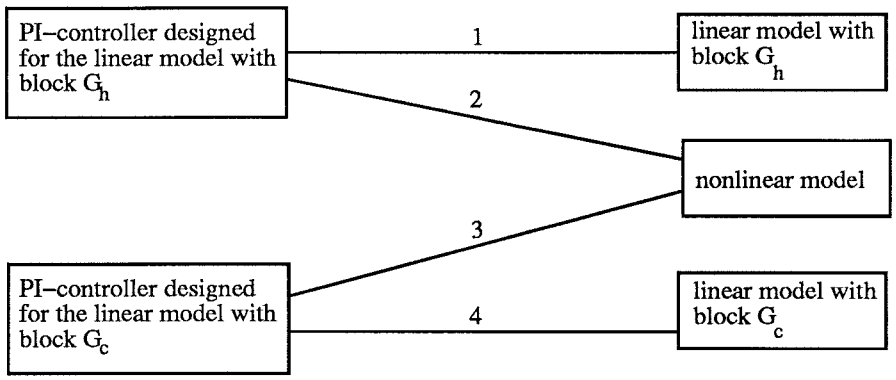

Figure 2.6 The four different combinations of controllers and process models that were used for the simulations.

The behaviours of the four systems were compared with respect to step responses for steps of the set point, load disturbances interacting at the beginning of the process, and measurement noise which was added at the end of the process. Basically two situations could be expected to be interesting.

1. How would the PI-controller designed for the linear model with block G_h (high gain linearity) perform, when the nonlinear block is in the domain with gain g_c (low gain)? How does this compare to the control of the linear model with block G_h?
2. How would the PI-controller designed for the linear model with block G_c (low gain linearity) perform, when the nonlinear block is in the domain with gain g_h (high gain)? How does this compare to the control of the linear model with block G_c?

During simulations the variable i(t) (see Fig.2.3) was the indicator to see in which domain of the nonlinear block the model is at any point in time. When i(t) is positive, the model is in the domain with gain g_h which shall be called the

positive domain. When i(t) is negative, the model is in the domain with gain g_c which shall be called the negative domain hereafter. For the simulations, the parameters g_h and g_c were chosen to be 1 and 0.02 respectively. This quite big ratio of 50 : 1 should allow to see in a "black and white" contrast what happens when switching from the linear to the nonlinear system.

EXAMPLE 2.1

In a first experiment the response of the nonlinear system to a negative unit step of the set point y_{sp} (see Fig.2.5) was simulated. Figure 2.7 shows the results for the case when the controller has been optimized for the linear model with block G_c. It compares the response of the output $y(t)$ of the nonlinear model (NL) with the one of $y(t)$ of the linear model (L) with block G_c.

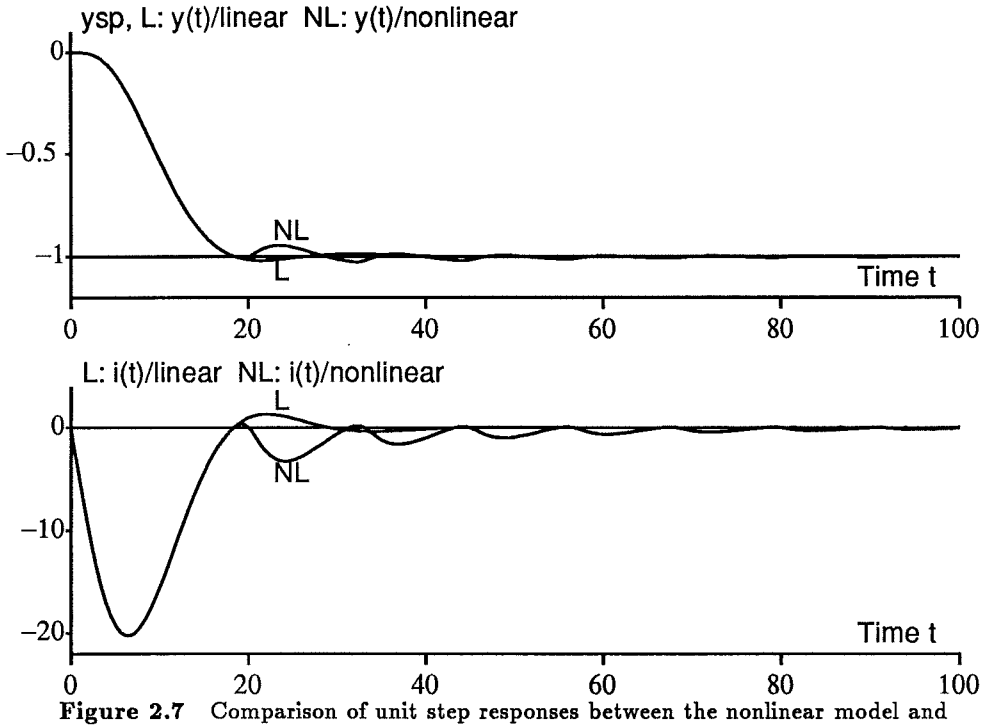

Figure 2.7 Comparison of unit step responses between the nonlinear model and the linear model with block G_c.

In the nonlinear case $y(t)$ oscillates more than in the linear case, but the difference is not significant. The explanation for this similar behaviour is found in the lower diagram of Fig.2.7. "Luckily" the indicator signal $i(t)$ of the nonlinear model is with a few exceptions always negative. Therefore the nonlinear model is almost always in the negative (low gain) domain and this is what the controller has been designed for. □

EXAMPLE 2.2

Figure 2.8 shows the results for the case when the controller has been optimized for the linear model with block G_h (high gain). It compares the response of the output $y(t)$ of the nonlinear model with the one of $y(t)$ of the linear model with block G_h.

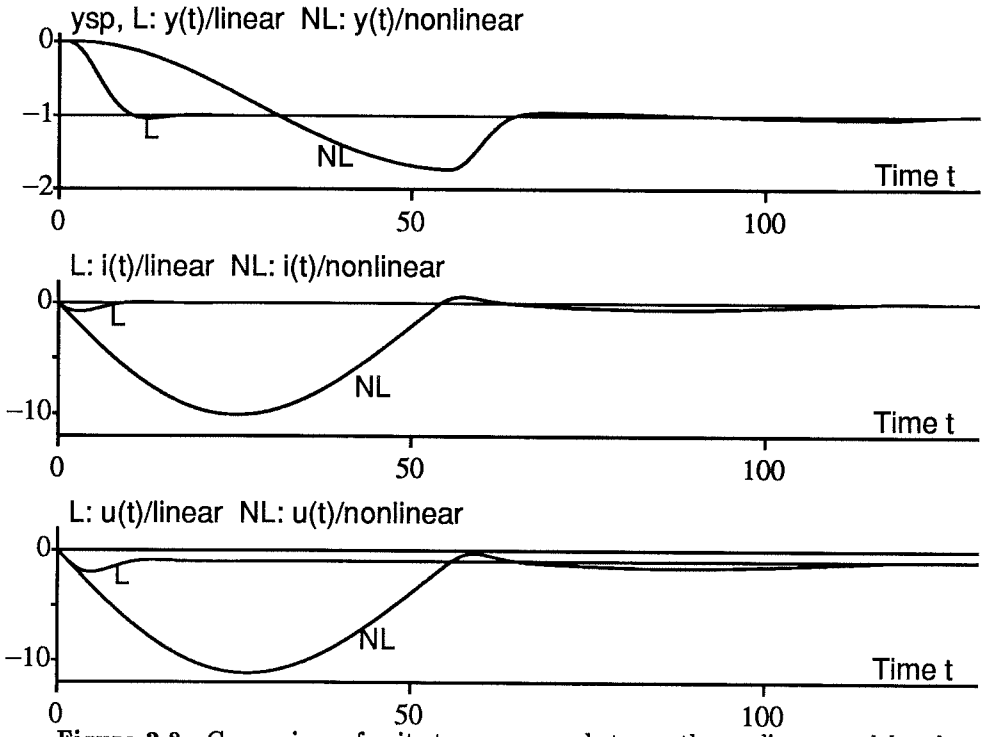

Figure 2.8 Comparison of unit step responses between the nonlinear model and the linear model with block G_h.

This time the responses are very different for the two models. For the nonlinear model $y(t)$ responds very slowly. It has a big overshoot and a very long settling time. Again the behaviour can be explained by looking at the indicator signal. As long as $i(t)$ is negative the nonlinear system is poorly controlled. The nonlinear model is in the negative (low gain) domain. Therefore the control signal $u(t)$ has to be much larger than for the linear system with block G_h (high gain) (see bottom diagram in Fig.2.8). The controller does not manage to produce big enough control signals because it has not been designed for the gain G_c (low gain) domain of the nonlinear block. However, as soon as the indicator signal becomes positive (at time$\approx$54) the nonlinear system responds very well, because now the nonlinear block is in the domain the controller has been designed for. □

In a second experiment the responses of the nonlinear system to a positive unit step of the set point y_{sp} were simulated for the different controllers. This time the controller designed for the linear model with block G_h (high gain) performed better than the other one. However it did not perform as well as it did when controlling the linear model with block G_h. Especially the settling time was several times as long as the one for the linear model. The reason is again that the indicator $i(t)$ became negative and therefore the nonlinear model entered the domain the controller was not designed for. When using the controller designed for the linear model with block G_c (low gain) the output of the nonlinear model oscillated heavily.

Similar experiences were made when adding load disturbance and measurement noise to the model. Every time the nonlinear block entered the domain the PI-controller was not designed for performance was poor.

These results could have been expected. The evidence found in the experiments demands a different approach to the problem. One obvious solution

would be to switch between different controllers. The decision on which controller to use would be based on the sign of the indicator $i(t)$. If the indicator can not be accessed this approach would no longer be possible.

During the experiments it was also found that the system is very sensitive to parameter changes. When one of the parameters g_h or g_c of the nonlinearity (depending on which gain the controller had been designed for) was changed a little bit control performance became worse. Therefore it must be concluded that when the parameters g_h or g_c (or other parameters of the system) vary with time, switching between different "fixed" controllers would not work well. In this case an adaptive controller would be required. To find a suitable adaptive controller two basic problems have to be solved.

1. Identification of the model assuming that different parts of the process are known.
2. Implementation of a controller assuming that the process has been identified and is therefore known.

This project will deal mainly with the first problem as it is the base for the implementation of an adaptive controller.

3. Recursive estimation of piecewise linear systems

In this project it is assumed that the process parameters vary continuously and that their variation can not be determined a priori. Therefore they have to be estimated recursively and the regulator parameters will be adjusted accordingly. An adaptive controller that captures these features is called a self-tuning regulator (see Fig.3.1), [Åström and Wittenmark, 1989].

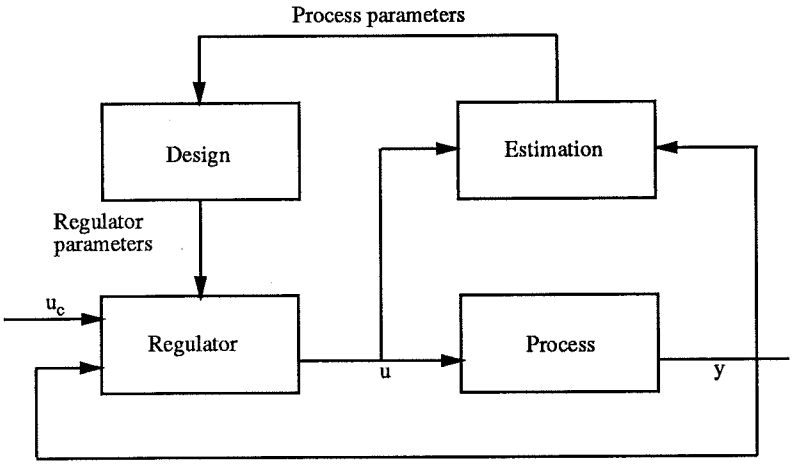

Figure 3.1 Block diagram of a self-tuning regulator.

The block labeled design in Fig.3.1 determines the new regulator parameters, treating the estimated process parameters as if they were true. This is called the certainty equivalence principle.

Several recursive parameter estimation methods have been developed such as maximum-likelihood estimation, Bayesian estimation (both stochastic approximations), and recursive least squares (using deterministic criteria). Least squares has been chosen for this project as it has several advantages. It has been applied successfully to a large variety of problems. Also when the model is linear in the parameters (as is the case for our problem) it is simple to apply because the least squares estimates can be calculated analytically.

3.1 Introduction to least squares

Assume the following model is given.

$$y(t) = \phi_1(t)\theta_1 + \phi_2(t)\theta_2 + ... + \phi_n(t)\theta_n = \phi(t)^T\theta \qquad (3.1)$$

y is the observed variable, $\theta_1, \theta_2, ..., \theta_n$ are the unknown parameters to be estimated, and $\phi_1, \phi_2, ..., \phi_n$ are known functions called regression variables. Introduce the vectors $\phi^T(t) = (\phi_1(t)\phi_2(t)...\phi_n(t))$ and $\theta^T = (\theta_1\theta_2...\theta_n)$ and the matrix

$$\Phi(t) = \begin{pmatrix} \phi^T(1) \\ \vdots \\ \phi^T(t) \end{pmatrix}$$

Pairs of observations and regressors $(y(i), \phi(i), i = 1, 2, ..t)$ are collected over a period of time. The aim is to determine parameters $\theta_1, \theta_2, ..., \theta_n$ such that the sum of the squared errors

$$V(\theta, t) = \frac{1}{2}\sum_{i=1}^{t}(y(i) - \phi^T(i)\theta)^2 \tag{3.2}$$

between the measured variable y(i) and the computed output from Eq.3.1 is minimized. For systems with time-varying parameters the least squares criterion of Eq. 3.2 can be replaced with

$$V(\theta, t) = \frac{1}{2}\sum_{i=1}^{t}\lambda^{t-i}(y(i) - \phi^T(i)\theta)^2 \tag{3.3}$$

where λ is a parameter called the forgetting factor such that $0 < \lambda \leq 1$. It gives unit weight to the most recent data, but data that is n time units old is only weighted by λ^n. The solution to this problem in recursive form is given by the following theorem, [Åström and Wittenmark, 1989].

Theorem 1: Recursive least-squares estimation with exponential forgetting
Assume that the matrix $\Phi(t)$ has full rank for $t \geq t_0$. The parameter θ, which minimizes 3.3 , is given recursively by

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \phi^T(t)\hat{\theta}(t-1)) \tag{3.4}$$

$$K(t) = P(t)\phi(t) = P(t-1)\phi(t)(\lambda I + \phi^T(t)P(t-1)\phi(t))^{-1} \tag{3.5}$$

$$P(t) = (I - K(t)\phi^T(t))P(t-1)/\lambda \tag{3.6}$$

$$= P(t-1) - P(t-1)\phi(t)(I + \phi^T(t)P(t-1)\phi))^{-1}\phi^T(t)P(t-1)$$

Recursive estimation of linear systems

First and second order linear systems were considered. A general first order system has the pulse-transfer operator

$$H(q) = \frac{b_1}{q + a_1} \tag{3.7}$$

where q is the forward shift operator. The corresponding difference equation is

$$y_t = -a_1 y_{t-1} + b_1 u_{t-1} \tag{3.8}$$

$$= (-y_{t-1}u_{t-1})\begin{pmatrix} a_1 \\ b_1 \end{pmatrix} = \phi_t^T\theta_1 \tag{3.9}$$

A general second order system has the pulse-transfer operator

$$H(q) = \frac{b_1 q + b_2}{q^2 + a_1 q + a_2} \tag{3.10}$$

and a corresponding difference equation of

$$y_t = -a_1 y_{t-1} - a_2 y_{t-2} + b_1 u_{t-1} + b_2 u_{t-2} \tag{3.11}$$

$$= (-y_{t-1} - y_{t-2}u_{t-1}u_{t-2})\begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \phi_t^T\theta_2 \tag{3.12}$$

The algorithm given by Theorem 1 was implemented in SIMNON to estimate the parameter vectors θ_1 and θ_2. The systems had the transfer functions $G(s) = 1/(s+1)$ and $G(s) = 1/(s+1)^2$ and were assumed to be time invariant, so the forgetting factor was set $\lambda = 1$ which means no forgetting. These continuous transfer functions were converted to their discrete representations (using the sampling time of the estimator), so that the true values of the parameter vectors θ_1 and θ_2 could be obtained as a reference. This was done for all the following examples. As input signals steps and square waves were used. Estimates showed to be quite accurate and had short convergence times for both systems.

3.2 Estimation of piecewise linear systems

The piecewise linear systems that were looked at have a transfer function $G(s) = G_1(s)g_h/(s+g_h)$ for the positive domain and a transfer function $G(s) = G_1(s)g_c/(s+g_c)$ for the negative domain, where $G_1(s)$ is linear and $g_h \neq g_c$. The following figure shows the situation for the parameter estimation.

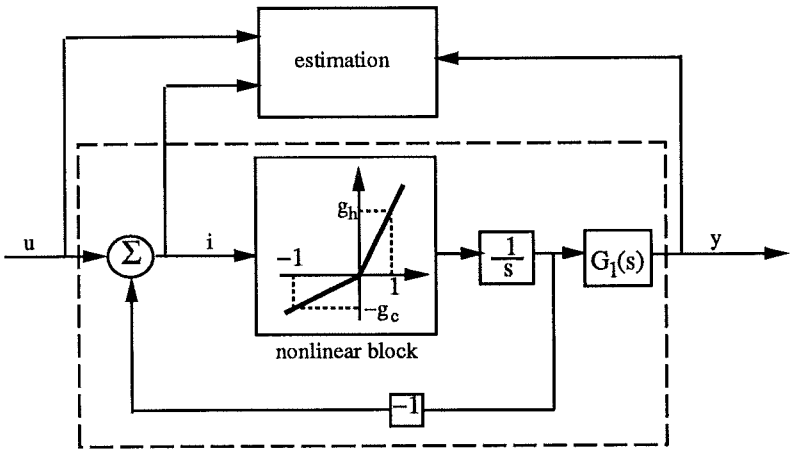

Figure 3.2 Parameter estimation of piecewise linear systems.

It was assumed that the indicator signal is accessible, so the three signals $u(t)$, $i(t)$, and $y(t)$ could be used for the parameter estimation. What should change in the recursive estimation algorithm?

Two "pairs" of parameters $\theta_h = (a_{1h}..a_{nh}\ b_{1h}..b_{nh})$ and $\theta_c = (a_{1c}..a_{nc}\ b_{1c}..b_{nc})$ (compare with Eqs.3.9 and 3.12) were defined corresponding to the two domains. How about the regression vector ϕ ?

$$\phi_t^T = (-y_{t-1} - y_{t-2} \cdots - y_{t-n}\ \ u_{t-1}u_{t-2}...u_{t-n})$$

The regression vector contains the old values of u and y. When the system moves from one domain into the other one, the regression vector still contains old values of u and y of the "old" domain which are of no use in the "new" domain. Therefore care must be exercised when the system switches between domains. It is necessary to interrupt estimation for a number of sampling periods (for an n-th order system n sampling periods) until the regression vector is filled with data that corresponds to the correct domain. This phenomenon will be called "time-out" hereafter.

Finally two P-matrices P_h for the positive and P_c for the negative domain had to be defined. The reason is that for the updating of a P-matrix the regression vector is needed (see Eq.3.6), which is different for the two domains.

The algorithm given by Theorem 1 had to be modified as follows. When the indicator $i(t)$ is negative (which means the system is in the negative domain) it is

$$K(t) = P_c(t)\phi(t) = P_c(t-1)\phi(t)(\lambda I + \phi^T(t)P_c(t-1)\phi(t))^{-1} \quad (3.13)$$

$$\hat{\theta}_c(t) = \hat{\theta}_c(t-1) + K(t)(y(t) - \phi^T(t)\hat{\theta}_c(t-1)) \quad (3.14)$$

$$\hat{\theta}_h(t) = \hat{\theta}_h(t-1) \quad (3.15)$$

$$P_c(t) = (I - K(t)\phi^T(t))P_c(t-1)/\lambda \quad (3.16)$$

$$P_h(t) = P_h(t-1) \quad (3.17)$$

Else, when the indicator $i(t)$ is positive it is

$$K(t) = P_h(t)\phi(t) = P_h(t-1)\phi(t)(\lambda I + \phi^T(t)P_h(t-1)\phi(t))^{-1} \quad (3.18)$$

$$\hat{\theta}_h(t) = \hat{\theta}_h(t-1) + K(t)(y(t) - \phi^T(t)\hat{\theta}_h(t-1)) \quad (3.19)$$

$$\hat{\theta}_c(t) = \hat{\theta}_c(t-1) \quad (3.20)$$

$$P_h(t) = (I - K(t)\phi^T(t))P_h(t-1)/\lambda \quad (3.21)$$

$$P_c(t) = P_c(t-1) \quad (3.22)$$

This algorithm can be regarded as an estimation of two linear systems with different gains in parallel. At any point in time the parameters are estimated as if the system was linear.

Estimation of systems with different gains

The algorithm given above was implemented in SIMNON for the estimation of first order piecewise linear systems. The implementation of recursive estimation algorithms for higher order piecewise linear systems in SIMNON was then just a matter of adding additional dimensions to the algorithm for first order systems (see Appendix A for program listings).

As a first step a first order piecewise linear system with the transfer function $G(s) = g_h/(s + g_h)$ for the positive domain and the transfer function $G(s) = g_c/(s + g_c)$ for the negative domain was considered. The parameters g_h and g_c of the nonlinear block were chosen as 1 and 0.02 respectively, same as in the previous chapter.

EXAMPLE 3.1—A first order piecewise linear system

Consider the system in Fig.3.2 with $G_1(s) = 1$. Let the input $u(t)$ be a square wave with period 10. Choose the sampling period $T = 0.2$ and let the initial covariance matrices P_h and P_c be $100I$ where I is the identity matrix. The parameter estimates are given in Fig.3.3.

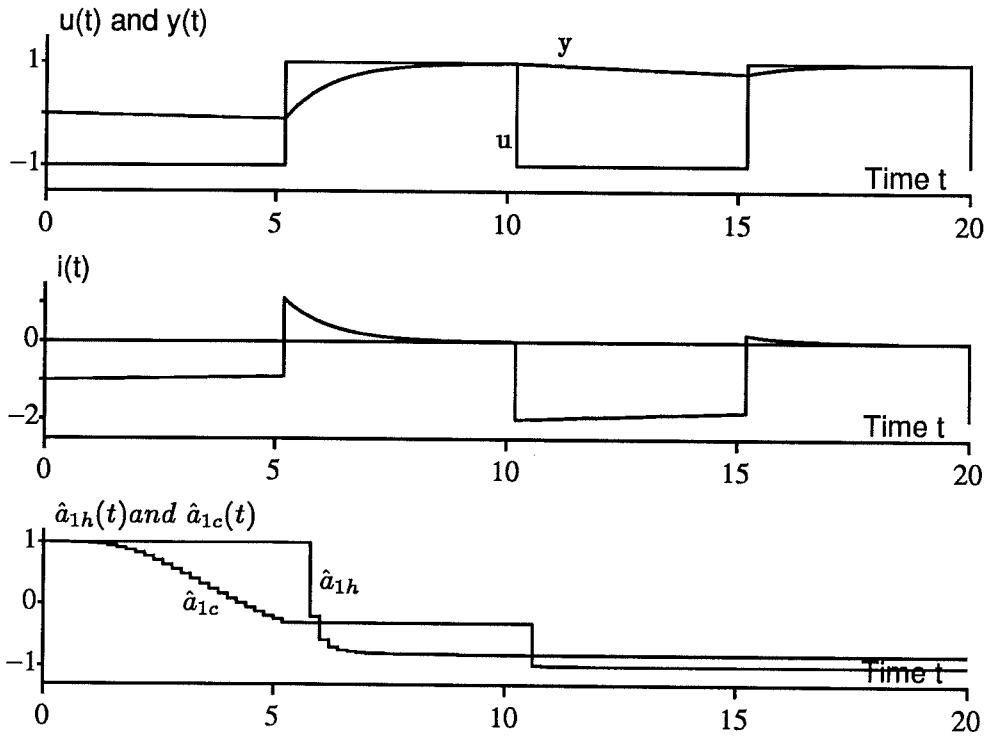

Figure 3.3 RLS parameter estimation of the first order piecewise linear system.

In the figure $\hat{a}_{1h}$ denotes the estimate of parameter a_1 in Eq.3.7 for the system when it is in the positive domain (the domain with gain g_h), $\hat{a}_{1c}$ is the estimate of parameter a_1 when the system is in the negative domain (the domain with gain g_c). Both parameter vectors $\hat{\theta}_h$ and $\hat{\theta}_c$ had the initial values $(1, 0)$.

The square wave $u(t)$ starts with a negative step which makes the indicator $i(t)$ negative. Therefore the estimation of the parameters for the negative domain starts (see lower diagram in Fig.3.3). Parameter $\hat{a}_{1c}$ converges slowly and has not reached its true value when the system moves to the positive domain (after 5 time units). It then stays constant until the system returns to the negative domain (after 10 time units). Then it converges at a value of -0.994 (true value: -0.996).

Parameter $\hat{a}_{1h}$ stays at its initial value until the indicator signal becomes positive (after 5 time units). Then it converges very quickly at -0.813 (true value -0.819). The input signal $u(t)$ was chosen to illustrate how the estimation algorithm works and not to get the best possible estimates. With more exciting input signals even more accurate estimates were obtained.

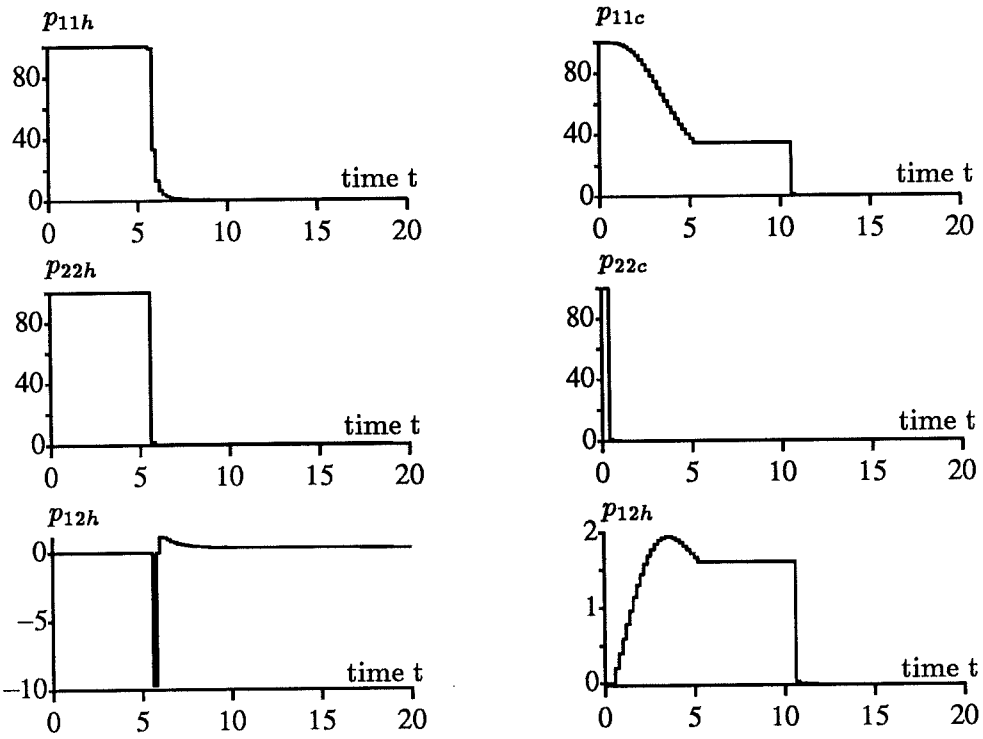

Figure 3.4 The elements of the P-matrices for the two domains.

In Fig.3.4 the elements of the P-matrices for the two domains are shown. On the left side all elements of P_h (positive domain) move very quickly to zero. On the right side the elements p_{11c} and p_{12c} move slowly during the first 5 time units. This speed reflects the speed of convergence of the parameters $\hat{a}_{1c}$ and $\hat{b}_{1c}$.

The experiment was repeated for different gains g_c. It was found that the more g_c is increased, the faster the parameter vector $\hat{\theta}_c$ would converge and the quicker the elements of the P_c-matrix would converge to zero. For $g_c = g_h$ parameters would converge at the same speed in both domains.

An explanation for this can be found from Eq.3.5. The filter gain is given by $K(t) = P(t)\phi(t)$, where $\phi(t) = (-y(t-1) \quad u(t-1))$. As the gain g_c is increased, the system is faster and the output $y(t)$ is larger. Therefore the regression vector $\phi(t)$ will be larger and the filter gain $K(t)$ will be higher. It follows that $\hat{\theta}(t)$ can move quicker.

Notice in Fig.3.3 how small the output $y(t)$ is during the first five time units. This is due to the low gain of the system in the negative domain.

Another way to get higher convergence rates for the parameters $\hat{\theta}_c$ and $\hat{\theta}_h$ was to increase the amplitude of the input $u(t)$. The reasons for this are similar to the ones just described in connection with increased gains. As the amplitude of $u(t)$ is increased, $y(t)$ increases and therefore $K(t)$ is larger. It follows again from Eq.3.4 that $\hat{\theta}(t)$ can change its value faster. Choosing the amplitude of $u(t)$ 30 times as big as previously $\hat{\theta}_c$ would converge almost as fast as $\hat{\theta}_h$.

Other ways to get a faster convergence of $\hat{\theta}_c$ were to use larger initial P-matrices and more exciting inputs $u(t)$. □

As a second step a second order piecewise linear system was considered. The true parameter vectors are now $\theta_h = (a_{1h}, a_{2h}, b_{1h}, b_{2h})$ for the positive and $\theta_c = (a_{1c}, a_{2c}, b_{1c}, b_{2c})$ for the negative domain (compare with Eq.3.12).

EXAMPLE 3.2—Parameter estimation of a second order piecewise linear system

Consider a system with the transfer functions $G(s) = g_h/((s+g_h)(s+1))$ and $G(s) = g_c/((s+g_c)(s+1))$ for the two domains ($g_h = 1, g_c = 0.02$). Let the input $u(t)$ be a square wave with period $T_u = 2\pi$ and amplitude 2. Choose the sampling period $T = 0.2$ and let the initial covariance matrices P_h and P_c be $10^4 I$. Choose the initial parameter vectors $\hat{\theta}_h$ and $\hat{\theta}_c$ both as $(0,0,0,0)$. The results for this experiment are given in Fig.3.5.

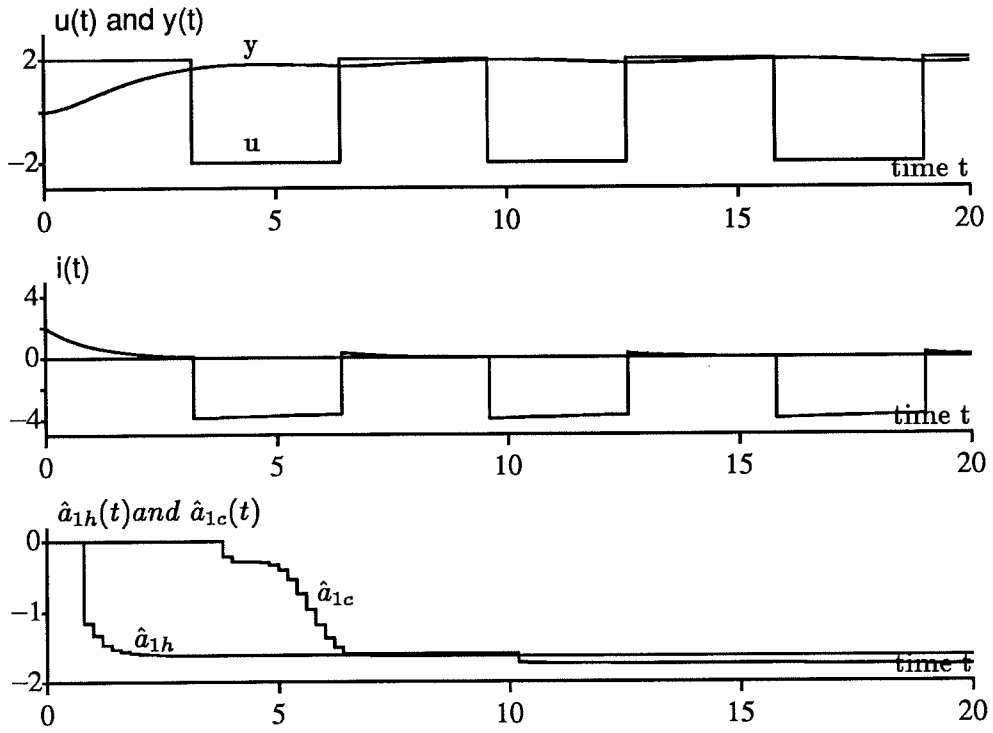

Figure 3.5 RLS parameter estimation of a second order piecewise linear system.

Parameter $\hat{\theta}_h$ converges quickly and is quite accurate after a simulation time of only 20 time units ($\hat{a}_{1h} = -1.631$ compared with a true value of -1.638), while $\hat{\theta}_c$ converges more slowly and is less accurate ($\hat{a}_{1c} = -1.76$ compared with a true value of -1.815). One reason for this is that $\hat{\theta}_c$ is the parameter estimate for a very slow linear process (gain $g_c = 0.02$). Therefore the output $y(t)$ has a smaller rate of change than in the high gain domain. This makes convergence slow.

It was found that $\hat{b}_{1h} = \hat{b}_{2h}$ and $\hat{b}_{1c} = \hat{b}_{2c}$. It will be shown in section 3.3 that this is not a coincidence. The elements of the lower right corners of the P-matrices ($p_{33h}, p_{34h}, p_{44h}, p_{33c}, p_{34c}$ and p_{44c}) were still at values around 5000 after 20 time units of simulation which is too large and indicates that something is wrong. It just did not have such a drastic effect for this system as the true parameters are not so far apart ($b_{1h} = 0.0175, b_{2h} = 0.0153, b_{1c} = 0.3741E-3, b_{2c} = 0.3495E-3$). □

One way to get around this problem was to use more exciting inputs $u(t)$.

EXAMPLE 3.3—Using a more exciting input signal $u(t)$

In Fig.3.6 the experiment from example 3.2 was repeated adding another square wave to the input $u(t)$ with period $T_{u2} = 5$ and amplitude 1.5.

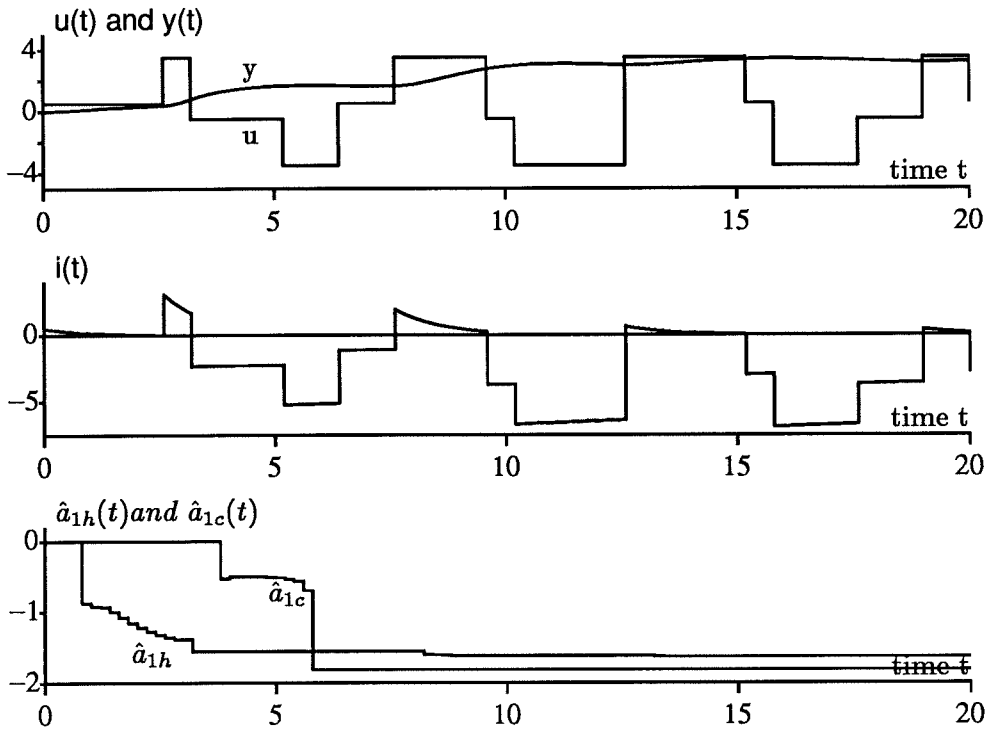

Figure 3.6 RLS parameter estimation of a second order piecewise linear system.

The results were very good this time. Parameter $\hat{a}_{1h} = -1.631$ compared with a true value of -1.638, $\hat{a}_{1c} = -1.813$ compared with a true value of -1.815. The $\hat{b}_{ih}$'s and $\hat{b}_{ic}$'s were estimated equally precisely this time. All elements of the P-matrices came down to small numbers (below 10). Notice that when the indicator becomes negative the first time, the parameter of the new domain ($\hat{a}_{1c}$) stays constant a little longer. The reason is that it has to be waited two sampling periods until the regression vector is filled with values from the new domain before any new estimation can be made. □

Other ways to speed up the convergence of the parameters were a) to use inputs with larger amplitudes and b) to use larger initial P-matrices. This would however not change the problems with the $\hat{b}_{ih}$'s and $\hat{b}_{ic}$'s. It was still found that $\hat{b}_{1h} = \hat{b}_{2h}$ and $\hat{b}_{1c} = \hat{b}_{2c}$ using a single square wave input.

Further experiments with second order systems were made varying the gains of the two domains. Similar experiences were made as for first order systems peviously. It was found again that the higher the gain of a domain was, the faster its parameters would converge to their true values.

One interesting result is the following. It was found that for piecewise linear systems the parameters of the low gain domain would converge faster than for linear systems with the same low gain.

EXAMPLE 3.4—Comparing parameter convergences for two systems

The experiment from example 3.3 was repeated for a second order piecewise linear system with the gains $g_h = 1$ and $g_c = 0.02$ for the two domains. The parameters of the negative domain (with gain g_c) were found to converge faster than for the linear system with gain g_c (see Fig.3.7).

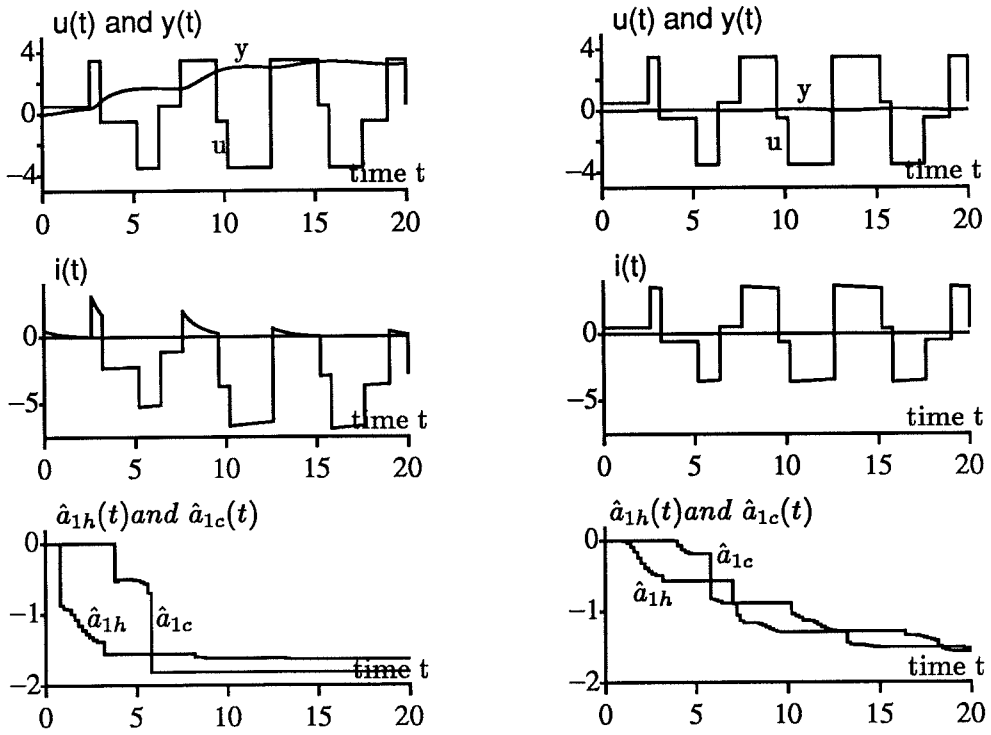

Figure 3.7 Comparing parameter convergences for second order linear and piecewise linear systems.

One reason is that in the piecewise linear system the output $y(t)$ is much larger when the system is in the negative (low gain) domain. Therefore the filter gain $K(t)$ will be larger and the parameter estimate $\hat{a}_{1c}$ can move quicker away from its initial value. Notice on the right side that the values of the output $y(t)$ are very small because the system has a very low gain in both domains.

A difficulty

In the previous examples the indicator $i(t)$ is in the positive domain (high gain domain) for a short time compared with the negative domain for piecewise linear systems. The reason for this can be found from Fig.3.2. Remember that g_h and g_c were chosen as 1 and 0.02. When $u(t)$ makes a positive step the indicator $i(t)$ follows it immediately and the same value is found at the output of the piecewise linear block. It then only takes a very short time until $y(t)$ takes on the same value as $u(t)$, so the indicator $i(t)$ moves back to zero very quickly. On the other hand, when $u(t)$ makes a negative step it takes much longer time for the indicator i(t) to go to zero, because of the low gain (0.02) of the nonlinear block. Another way to see this is that the time constant of the loop consisting of the nonlinearity and the integrator (see Fig.3.2) is 1 in the positive and 50 in the negative domain. This can be expected to cause some problems, especially for higher order piecewise linear system where the estimator has to wait longer after changing domain before any estimation can be done.

The easiest solution to this problem would be to add a ramp to the input signal with a gain such that the indicator spends half the time in the positive and the other half in the negative domain. But this is of course useless for an adaptive controller, as a system can not be controlled this way.

EXAMPLE 3.5—A difficulty with piecewise linear systems

As described previously, when entering a new domain the estimation can only start after a number of sampling periods, so for certain input signals $u(t)$ the indicator $i(t)$ might not be long enough positive in order to obtain good estimates. As an example consider Fig.3.4 where the first order piecewise linear system has a sinusoidal input $u(t)$ with amplitude 0.4 and period $T_u = 2\pi$. The sampling period is $T_s = 0.2$. The initial covariance matrices P_h and P_c are chosen as $100I$. The initial parameter vectors $\hat{\theta}_h$ and $\hat{\theta}_c$ are both $(1, 0)$.

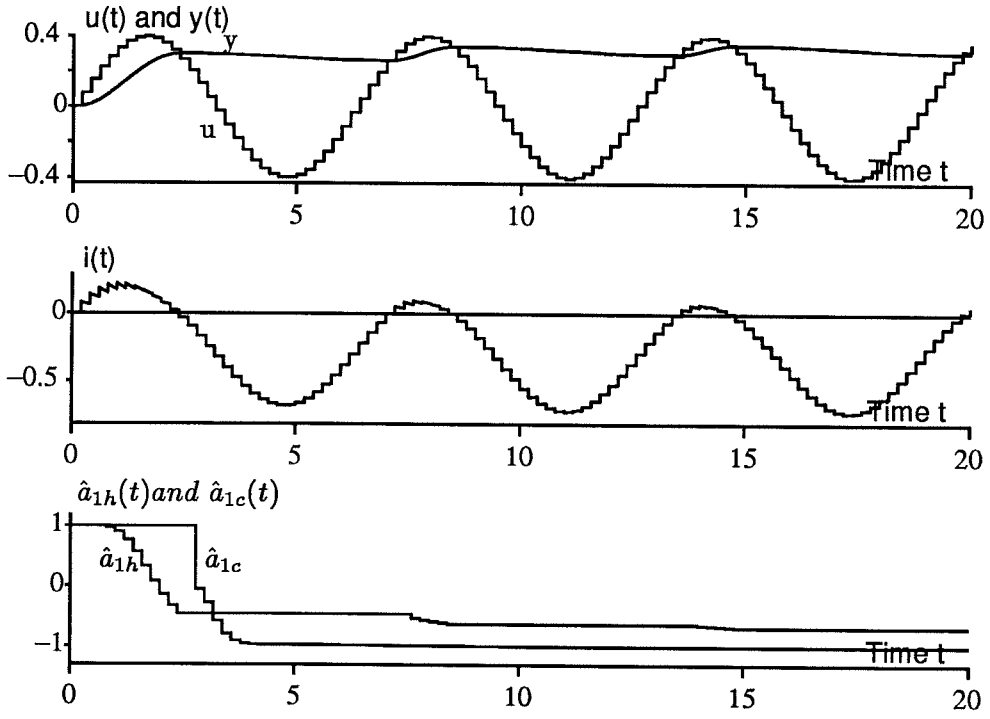

Figure 3.8 RLS parameter estimation using sinusoidal input.

The parameters of the negative domain are estimated quite accurately (i.e. $\hat{a}_{1c} = -0.993$ compared with a true value of -0.996), while the parameters of the positive domain are estimated poorly ($\hat{a}_{1h} = -0.67$ compared with -0.819). This behaviour was found independently of the frequency of $u(t)$ over a wide range of frequencies. □

Several experiments were made to investigate by what means the estimates could be improved using sinusoidal input. The following observations were made. In order to get more accurate estimates three different ways were found.

1. Increase the amplitude of the input signal $u(t)$. In example 3.2 the estimates in the positive domain are improved significantly ($\hat{a}_{1h}$ =-0.812 compared with a true value of -0.819) by increasing the amplitude of the input signal 5 times. This is the same effect as described in example 3.1. It can be explained with the filter gain $K(t)$ being higher which allows the parameter estimates to move faster away from their initial values (see Eqs.3.4 and 3.5).
2. Use longer simulation times. An estimate $\hat{a}_{1h} = -0.809$ was obtained after a simulation over 10000 time units. This is close to the true value of -0.819.

The last measure increases the data used for estimation. Thus more information is available and the estimate becomes more accurate.

3. Increase the initial P-matrix. The initial P-matrix is usually chosen as $P(0) = \rho I$, where ρ is a large number and I is the identity matrix. It reflects the confidence in the initial estimate $\hat{\theta}(0)$, see [Söderström and Stoica, 1989]. By increasing $P(t)$, $K(t)$ will increase (see Eq.3.4 and 3.5) and therefore the parameters can move much faster away from their initial values $\hat{\theta}(0)$ when the error $(y(t) - \phi^T(t)\hat{\theta}_h(t-1))$ is different from zero. Choosing $\rho = 10^4$ for the example from above almost perfect estimates were obtained ($\hat{a}_{1h}$ =-0.817 compared with a true value of -0.819, $\hat{a}_{2h}$ =0.182 compared with a true value of 0.181).

3.3 The "time-out" effect

An interesting phenomenon that occurs in identification of piecewise linear systems called the "time-out" effect was explained in section 3.2. It was found that every time the system changes domain (that is the indicator $i(t)$ changes sign), estimation has to be interrupted for a number of sampling periods. It is well known in identification of linear systems that good data for estimation is obtained in transients. The time-out means that these periods are significantly reduced for piecewise linear systems. What are the effects of this? It will be shown that the problems found in example 3.2 are due to the time-out in the estimator.

The change of domain of the system is usually caused by large variations in the input signal $u(t)$ (i.e. steps). Significant information about parameters is obtained during these time periods. For piecewise linear systems the estimator is however blocked at the changes, because of the need of filling the regression vector with appropriate data. Thus the estimator does not operate during the times when the data contains the most valuable information for system identification.

To see one possible effect, consider an n-th order system with the regression vector $\phi_t^T = (-y_{t-1} - y_{t-2} \ldots - y_{t-n}\ u_{t-1}u_{t-2}\ldots u_{t-n})$ and a corresponding parameter vector $\theta_t = (a_1 a_2 \ldots a_n\ b_1 b_2 \ldots b_n)$ for one domain. Assume a square wave input $u(t)$ (amplitude c_1). After changing domain the estimator waits n sampling periods until the regression vector is filled with data from the new domain. The components of the regression vector that corresponds to $u(t)$ are then all equal and constant. This has some unexpected consequences.

It will be shown that for certain initial P-matrices, the part of the filter gain $K(t)$ corresponding to the b-parameters will always consist of equal values.

To see this partition the vectors θ and ϕ into parts corresponding to the a- and b-parameters. Partition the P-matrix in a similar way. Assume that the regression vector has been filled with data from the positive domain for the first time at time t=1. It then has the form

$$\phi(1) = \begin{pmatrix} \phi_a \\ c_1 \epsilon \end{pmatrix} \tag{3.23}$$

where ϵ is a vector with all components equal to one and c_1 is a constant. Introduce the vector $\delta^T = (d_1 \ldots d_n)$, where the Parameters $d_1 \ldots d_n$ are constants.

Assume further that the initial P-matrix has the form

$$P(0) = \begin{pmatrix} P_a & \delta\epsilon^T \\ \epsilon\delta^T & \rho I + c_2 M \end{pmatrix} \tag{3.24}$$

where I is the identity matrix and M is a matrix with all components equal to one (both have dimension n). Notice that the identity matrix is of this type.

It follows from Eqs.3.23 and 3.24 that

$$\begin{aligned} P(0)\phi(1) &= \begin{pmatrix} P_a & \delta\epsilon^T \\ \epsilon\delta^T & \rho I + c_2 M \end{pmatrix} \begin{pmatrix} \phi_a \\ c_1\epsilon \end{pmatrix} \\ &= \begin{pmatrix} P_a\phi_a + c_1 n\delta \\ \epsilon\delta^T\phi_a + c_1\rho\epsilon + c_1 c_2 n\epsilon \end{pmatrix} =: \begin{pmatrix} \mu \\ \nu \end{pmatrix} \end{aligned} \tag{3.25}$$

The filter gain is given by

$$K(t) = P(t-1)\phi(t)(I + \phi^T(t)P(t-1)\phi(t))^{-1} \tag{3.26}$$

(see Eq.3.5) which we partition as $K(t) = \binom{K_a(t)}{K_b(t)}$ corresponding to the a- and b-parameters. It follows for t=1 from Eqs.3.25 and 3.26 that

$$\begin{aligned} K(1) &= P(0)\phi(1)(I + \phi^T(1)P(0)\phi(1))^{-1} \\ &= \begin{pmatrix} \mu \\ \nu \end{pmatrix} \frac{1}{1+\alpha} = \begin{pmatrix} K_a(1) \\ K_b(1) \end{pmatrix} \end{aligned} \tag{3.27}$$

where the scalar α is given by $\alpha = \phi^T(1)P(0)\phi(1)$. Therefore the filter gains $K_b(1)$ associated with the b-parameters are the same after one recursive step.

The proof is only complete if the updated P-matrix has the same form as its preceeding one. To show this consider that the new P-matrix is given by

$$P(t) = P(t-1) - P(t-1)\phi(t)(I + \phi^T(t)P(t-1)\phi))^{-1}\phi^T(t)P(t-1) \tag{3.28}$$

(see Eq.3.6). For t=1 it follows that

$$\begin{aligned} P(1) &= P(0) - P(0)\phi(1)(I + \phi^T(1)P(0)\phi))^{-1}\phi^T(1)P(0) \\ &= \begin{pmatrix} P_a & \delta\epsilon^T \\ \epsilon\delta^T & \rho I + c_2 u \end{pmatrix} - \begin{pmatrix} \mu \\ \nu \end{pmatrix} \frac{1}{1+\alpha} (\mu^T \nu^T) \\ &= \begin{pmatrix} P_a & \delta\epsilon^T \\ \epsilon\delta^T & \rho I + c_2 u \end{pmatrix} - \frac{1}{1+\alpha} \begin{pmatrix} \mu\mu^T & \mu\nu^T \\ \nu\mu^T & \nu\nu^T \end{pmatrix} \end{aligned} \tag{3.29}$$

It now remains to show that $\mu\nu^T$, $\nu\mu^T$, and $\nu\nu^T$ do not change the form of the P-matrix. Consider ν in Eq.3.25 which is

$$\nu = \epsilon\delta^T\phi_a + c_1\rho\epsilon + c_1 c_2 n\epsilon$$

Since $\delta^T\phi_a$ is a scalar, the equation can be rewritten as

$$\nu = (\delta^T\phi_a + c_1\rho + c_1 c_2 n)\epsilon = k_1\epsilon$$

where k_1 is a scalar. It follows that $\nu\nu^T = k_1^2\epsilon\epsilon^T = k_1^2 M$. This matrix only changes the value of c_2 in Eq.3.1 which does not effect the form of the submatrix $\rho I + c_2 U$.

Now consider $\mu\nu^T$. Rewrite μ as $\mu = f + c_1 n\delta$, where the vector $P_a\phi_a$ has simply been renamed for convenience. Thus

$$\mu\nu^T = (f + c_1 n\delta)k_1\epsilon^T = k_1(f\epsilon^T + c_1 n\delta\epsilon^T)$$

This matrix is of the same form as $\delta\epsilon^T$ and by adding them the form remains the same. Because $(\delta\epsilon^T)^T = \epsilon\delta^T$ and $(\mu\nu^T)^T = \nu\mu^T$ the same thing applies to $\epsilon\delta^T$. The proof now follows by induction.

Therefore the filter gains $K_b(t)$ associated with the b-parameters are always all equal. If all b-parameters are equal initially it follows that they remain equal during the estimation. It thus follows that a square wave signal is not a good input for estimating the parameters of a piecewise linear system. This explains the results from example 3.2 where it was found that $\hat{b}_{1h} = \hat{b}_{2h}$ and $\hat{b}_{1c} = \hat{b}_{2c}$.

In addition the output $y(t)$ will also change very little in the low gain domain. The excitation is therefore poor and the a-parameters are poorly estimated. Hence the only parameter that we can expect to estimate accurately under these circumstances is the static gain of the system.

A comparison between estimators with and without time-out

To illustrate the effects of the time-out first and second order linear systems were investigated. For linear systems there is no need to stop estimation when changing domain and wait until the regression vector is filled with data from the correct domain. The data from both domains reflects the same linear system and can therefore be used in both domains. The estimators for piecewise linear systems were modified such that they still estimate two sets of parameters for the two different domains, but do not interrupt estimation when switching domain. These two sets of parameters should then converge to the same values for linear systems.

Comparisons were made between the estimator with and without time-out for first order linear systems with different gains. Remember that for first order systems the estimator only waits one sampling period when changing domain which is a very short time.

When comparing the two estimators for a first order linear system with gain 1 virtually no difference could be found. The parameter estimates converged just as fast in the two cases. For first order systems with lower gains significant differences could be found.

EXAMPLE 3.6—The effects of the time-out for a first order system.

Consider Fig.3.9 where the two estimators are compared for a first order linear system with gain 0.02. A square wave with period 10 was chosen as the input $u(t)$. The estimator had a sampling period $T = 0.2$. The initial covariance matrices $P_h(0)$ and $P_c(0)$ were both $1000I$. The initial parameter vectors $\hat{\theta}_h$ and $\hat{\theta}_c$ were equally $(1, 0)$.

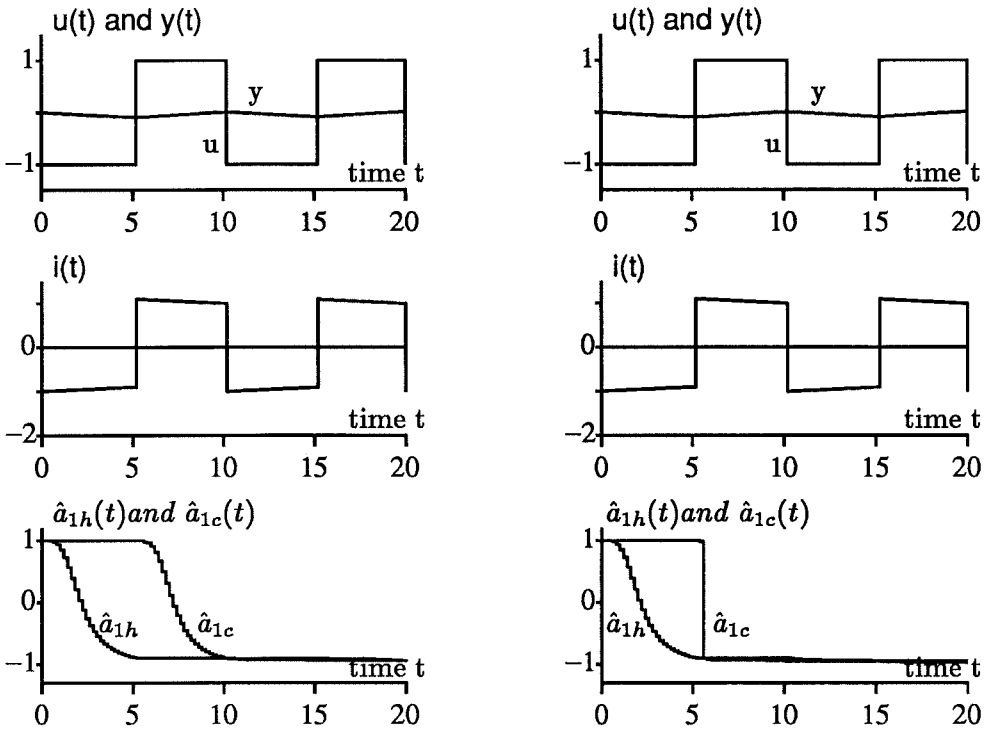

Figure 3.9 A comparison between a parameter estimation with time-out (left) and without time-out (right).

On the left side the estimation with time-out is shown, on the right side the one without time-out. Parameter $\hat{a}_{1h}$ converges slowly in both cases. To explain this consider Fig.3.10 where the output $y(t)$ is shown with different scales.

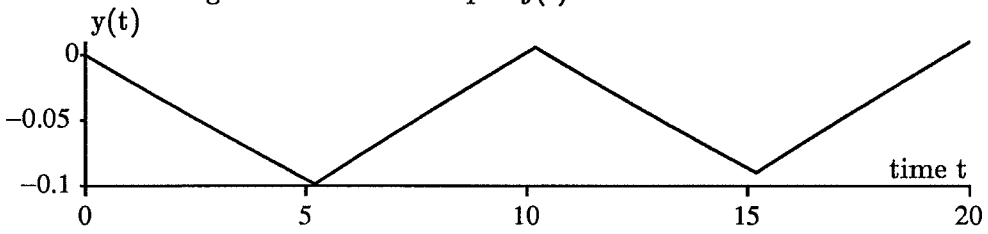

Figure 3.10 The output signal $y(t)$

At time 0 the regression vector is $\phi^T(0) = (0, 0)$. As the initial output signal $y(0)$ is zero, it follows that the correction term

$$e(0) = (y(0) - \phi^T(0)\hat{\theta}_h(0))$$

(see Eq.3.4) will be zero. Thus the estimates can not change. The estimator without time-out can not use the excitation in $u(t)$ ($u(t)$ makes a step at time 0). Therefore $\hat{a}_{1h}$ converges at the same rate for both estimators. Since the system has a low gain the output $y(t)$ has a small rate of change and parameters converge slowly.

The speed of convergence of $\hat{a}_{1c}$ on the other hand is very different in the two cases. The reason is that when the system moves in the positive domain (after 5 time units), the correction term

$$e(t) = (y(t) - \phi^T(t)\hat{\theta}(t-1))$$

is quite large, because $y(t)$ is approximately -0.1. The estimator without time-out can change the estimate $\hat{\theta}_h$ rapidly, making use of the excitation in the input step. Parameter $\hat{\theta}_h$ converges fast in this case.

When the estimator with time-out can start estimation the output $y(t)$ has a very small rate of change. As there is not any excitation in the input $u(t)$ parameters converge slowly. What makes the difference is that the most valuable part of the excitation in the input step (after 5 time units) can not contribute to a better identification when the estimator with time-out is used. When the estimator without time-out is used it can.

As already mentioned earlier the filter gain $K(t)$ can be raised by increasing the amplitude of the input $u(t)$. When increasing the amplitude of $u(t)$ it was found that the parameters of both domains would converge faster. □

Improved excitation

Having understood the phenomenon of time-out, we can now suggest some ways to overcome the problem. Since it is due to the lack of excitation in connection with switches from one domain to another, we will simply make sure that after a switch, the input is changed so that proper excitation is obtained while the system remains in the same domain.

EXAMPLE 3.7
In Fig.3.11 a square pulse was added to the square wave after the regression vector was filled with data from the positive domain (after a little more than 5 time units).

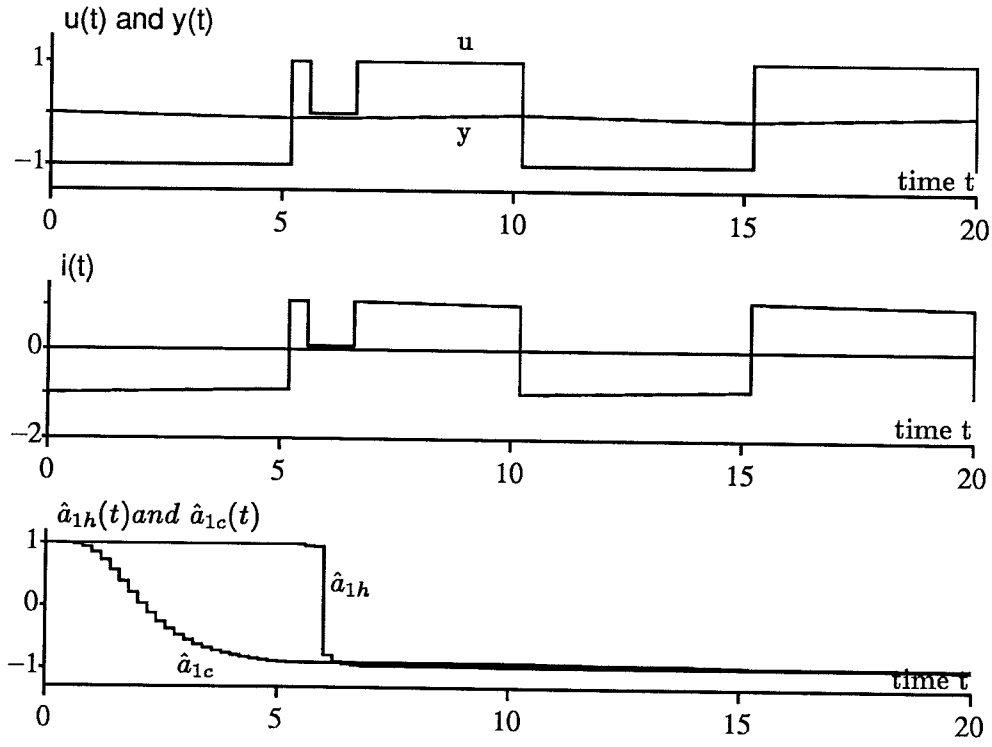

Figure 3.11 The parameter estimation when adding a square pulse to the input $u(t)$.

The estimate $\hat{a}_{1h}(t)$ converges very fast this time. This is the result of just a little extra excitation. By adding more square pulses when the system is in the negative domain the parameter $\hat{a}_{1c}(t)$ would also converge faster. □

Comparisons were also made between estimators for second order systems with and without time-out. Similar results were obtained as for first order systems. For a linear system with gain $g_h = 1$ very little difference could be found between the two estimators. When using a linear system with a gain $g_c = 0.02$, the estimator without time-out performed a little better than the other one. The difference was however much smaller than for first order systems.

Same as for first order systems the speed of convergence could be raised by a) using inputs $u(t)$ with bigger amplitudes, b) using better excited inputs $u(t)$, and c) choosing more suitable initial P-matrices.

Summary

As a conclusion, the input signal $u(t)$ must capture more features for the parameter estimation of piecewise linear systems than for the parameter estimation of linear systems.

1. Same as for linear systems, it should be excited enough in order to obtain accurate estimates. However there must be excitation during the periods of time when estimation can be performed. As the steps of the input $u(t)$ that cause a change of domain always fall into the time-out period of the estimator, the most valuable part of their excitation can not contribute to a better identification of a piecewise linear system. Therefore steps within the domains that do not cause another change of domain are needed.
2. Another new result is that the input $u(t)$ must bring the indicator signal long enough in the high gain domain, so that enough estimations can be performed there.

3.4 Introducing a forgetting factor λ

So far all experiments were made using $\lambda = 1$ which means no forgetting. This was reasonable because the systems were time invariant. As the estimator designed in this project shall be used in self-tuning regulators to control time variant piecewise linear systems, experiments were made using forgetting factors $\lambda < 1$. The question was whether this would effect the estimator for piecewise linear systems in the same way as it effects estimators for linear systems.

In a first step first order linear and piecewise linear systems were investigated. For high gain linear systems (i.e. with gain $g_h = 1$) the estimator was not found to be sensitive to changes in λ . The results were virtually the same for $0.9 < \lambda < 1$. For low gain linear systems however the estimator was found to be sensitive to different choices of λ. Estimates converged faster for smaller choices of λ.

For the piecewise linear process with gains $g_h = 1$ and $g_c = 0.02$ slightly better results were obtained for smaller λ's.

In a second step second order linear and piecewise linear systems were looked at. For second order high gain linear systems (i.e. with gain $g_h = 1$) estimates converged a little faster as λ was reduced. For second order low gain linear systems (i.e. with gain $g_c = 0.02$) significant differences could be found.

EXAMPLE 3.8

In Fig.3.12 the estimator performances are compared for a linear system with the transfer function $G(s) = \frac{1}{(s+0.02)(s+1)}$ for $\lambda = 1$ and $\lambda = 0.9$. Two square waves with periods $T_{u1} = 2 * pi$ and $T_{u2} = 5$ and amplitudes a1=2 and a2=1.5 were used as the input $u(t)$. The sampling period of the estimator was $T_s = 0.2$ and the P-matrices had the initial values $P_{h,c} = \rho 10^4$. The initial parameter vectors $\hat{\theta}_h$ and $\hat{\theta}_c$ were both chosen as $(0, 0, 0, 0)$.

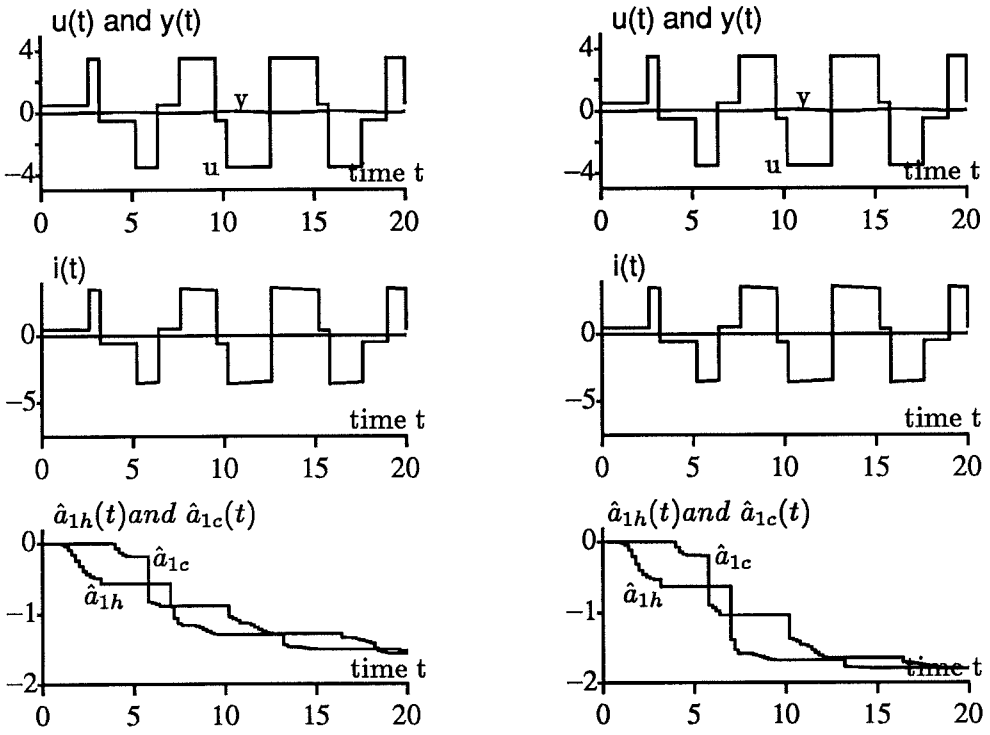

Figure 3.12 Comparing estimation of a linear second order system for $\lambda = 1$ and $\lambda = 0.9$.

Using $\lambda = 1$ the parameters converged slowly (see left side). Parameter $\hat{a}_{1h} = -1.527$ and $\hat{a}_{1c} = -1.563$ after 20 time units of simulation. These values are quite far from the true value of -1.815.

On the other hand, using $\lambda = 0.9$ results were much better. Parameter $\hat{a}_{1h} = -1.799$ and $\hat{a}_{1c} = -1.803$. These values are close to the true value.

The reason for this is that the smaller λ increases the P-matrices. The filter gain $K(t)$ will be larger (see Eqs.3.4 and 3.5) and the parameter estimates can therefore move faster away from their initial values. For a higher gain system the difference is less visible, as the rate of convergence is quite big anyway. □

EXAMPLE 3.9

In Fig.3.13 a comparison was made between the estimator designed in this project for piecewise linear systems and a standard least squares estimator designed for linear systems. The standard least squares estimator only estimates one set of parameters, performing estimation in both domains and all the time.

The performances of the two estimators were compared when estimating the parameters of the same second order low gain linear system as in the last example. Also the same input signal $u(t)$ and the same initial conditions were

used as in the last example. Fig.3.13 shows the results using $\lambda = 0.9$ for both estimators.

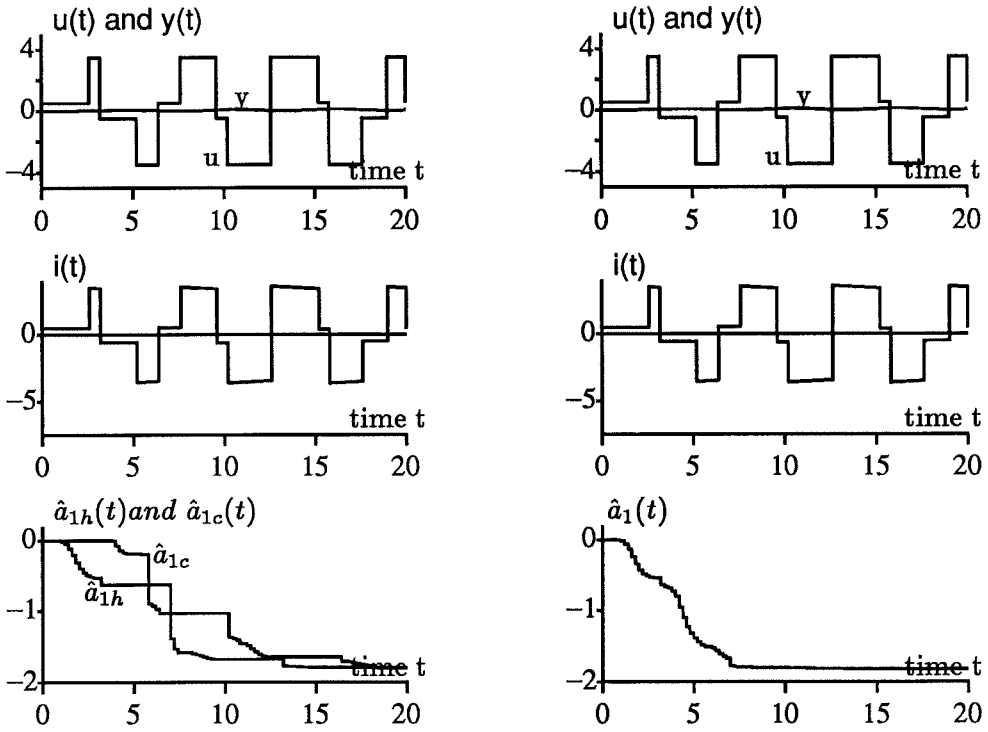

Figure 3.13 Comparison of the two estimators with $\lambda = 0.9$.

This example illustrates quite nicely that the convergence rate of the parameters can be twice as long when estimating two sets of parameters (see left side) compared to estimating just one set (see right side).

When using the estimator without time-out estimates would converge a little faster. □

For the piecewise linear system it was found again that by using smaller λ's parameters would converge a little faster.

In a good estimator care must be taken to ensure that the estimator does not diverge when forgetting is used. This can be achieved by a) covariance resetting, b) a constant trace algorithm, or c) directional forgetting. See [Söderström and Stoica, 1989].

Parameter estimations for fourth order systems were also carried out. A general fourth order system has the pulse-transfer operator

$$H(q) = \frac{b_1 q^3 + b_2 q^2 + b_3 q + b_4}{q^4 + a_1 q^3 + a_2 q^2 + a_3 q + a_4} \tag{3.30}$$

Notice that 8 parameters have to be estimated at the same time so the estimation can be expected to be a lot more difficult.

Using $\lambda = 1$ the parameters of the fourth order linear system $G(s) = g_h/((s + g_h)(s + 1)^3)$ $(g_h = 1)$ converged very slowly and only when using very exciting inputs $u(t)$. With smaller forgetting factors parameters would converge significantly faster. No good results were obtained for the parameter estimation of fourth order piecewise linear systems. This could not be explained.

4. Controller design for piecewise linear systems

In the previous chapter recursive least squares was introduced and applied to estimate the parameters of piecewise linear systems. These estimates shall now be used to design appropriate regulators to control these systems (compare with Fig.3.1). There exist a number of different controller design methods as pole placement, linear quadratic, minimum variance and model-following, see [Åström and Wittenmark, 1989]. It was decided to use pole placement based on input-output models. One reason is that an input-output model is obtained from the recursive estimation algorithm. Another advantage is that it is simple and easy to implement.

4.1 Introduction to pole placement

This design method is applicable to single input, single output systems. The relation between the input u and the output y of the system is given by the pulse-transfer function

$$H(z) = \frac{B(z)}{A(z)} \tag{4.1}$$

$A(z)$ and $B(z)$ are assumed to be polynomials without common factors. Also the desired closed-loop pulse-transfer function must be specified in the form

$$H_M(z) = \frac{B_M(z)}{A_M(z)} \tag{4.2}$$

where $B_M(z)$ and $A_M(z)$ should again not have any common factors. Finally a characteristic polynomial A_o must be given that contains observer dynamics. The control law for a regulator with the two inputs y_{sp} (the command signal) and y (the measured output) and the output u can be written as

$$R(q)u(k) = T(q)y_{sp}(k) - S(q)y(k) \tag{4.3}$$

R, T, and S are polynomials in the forward shift operator. R is assumed to be monic which means that the coefficient in the highest power in R is unity. Conditions on the control law are $\deg R \geq \deg T$ and $\deg R \geq \deg S$, both for causality reasons. The degrees are usually chosen such that equality holds, as it will be the case in this project.

It is now the goal to specify the polynomials R, T, and S for the control law given by Eq.4.3 such that the closed-loop system (see Fig.4.1) satisfies the input-output relation given by Eq.4.2 and an observer with the characteristic polynomial $A_o(z)$. Notice that disturbances and modeling errors are neglected.

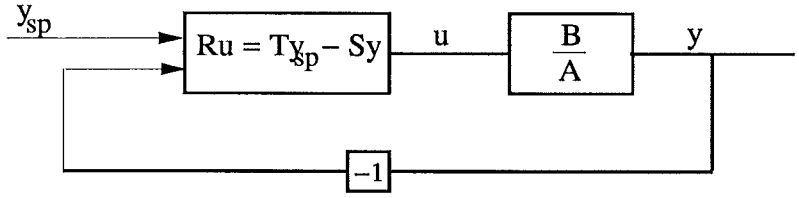

Figure 4.1 Block diagram of the closed-loop system.

Eliminating u between Eqs.4.1 and 4.3 gives

$$\frac{y}{y_{sp}} = \frac{BT}{AR + BS} = \frac{B_m}{A_m} \tag{4.4}$$

There are many different solutions to Eq.4.4 by different choice of R,S, and T.

Usually sufficiently well damped zeros of the open-loop system may be canceled by closed-loop poles. Because the self-tuning regulator shall be used for systems where information about poles might not be available it was decided not to cancel any zeros.

The pole placement design problem can be solved by the following algorithm [Åström and Wittenmark, 1984 and 1989].

ALGORITHM 4.1—Pole placement design with no zeros canceled
Conditions on the control law in order to get a causal solution are:

$$\deg A_m - \deg B_m \geq \deg A - \deg B$$
$$\deg A_o \geq \deg A - \deg A_m - 1$$

A further condition is: B divides B_m

Step 1: Solve the equation (Diophantine equation)

$$AR + BS = A_o A_m$$

with respect to R and S. Choose a solution such that

$$\deg S < \deg A$$

and

$$\deg R = \deg A_o + \deg A_m - \deg A$$

Step 2: Form

$$T = A_o B_m / B$$

The control law is

$$Ru = Ty_{sp} - Sy$$

4.2 Pole placement for piecewise linear systems

At first sight only one extra step has to be added to the algorithm described above in order to apply it to piecewise linear systems. It must be decided which linearity of the system the controller shall be designed for. Obviously this decision should be based on the sign of the indicator $i(t)$. When $i(t)$ is positive the input-output model $H(z) = \frac{B(z)}{A(z)}$ for the positive domain must be chosen. Otherwise the model for the negative domain is the proper one to use.

With this little modification the algorithm was implemented in SIMNON for second order piecewise linear systems. The following algebraic problem arose when trying to start the simulation. In the system the indicator $i(t)$ is a function of $u(t)$. In the controller design algorithm the decision on with input-output model to use depends on the indicator $i(t)$. Therefore the new controller output $u(t)$ (which is the system input) depends on the indicator which itself depends on the controller output $u(t)$. As there are no time-delays an algebraic loop occurs.

The only way that was found to overcome this problem was to introduce a delay somewhere in the design algorithm. It was decided to base the decision on which input-output model to use on the old and not the actual indicator $i(t)$. This way the controller will calculate outputs $u(t)$ for the old linearity for one extra sampling period when the system changes domain. This is not very satisfactory, but no better solution could be thought of. The controller was implemented this way.

EXAMPLE 4.1—Adaptive control of a linear system.
In Fig.4.2 the linear system with the transfer function $G(s) = 1/(s+1)^2$ was controlled for 200 time units. A square wave set point $y_{sp}(t)$ with period 40 was used. The sampling period was $T = 0.2$. The initial covariance matrices P_h and P_c for the estimator were both chosen as 10^4I, where I is the identity matrix. The initial parameter estimates $\hat{\theta}_h$ and $\hat{\theta}_c$ were both $(0, 0, 0.1, 0.1)$. The desired closed-loop damping was 0.85 and the desired closed-loop natural frequency was 1.2. The observer pole was set at -0.4. The control variable $u(t)$ was limited to ± 2.5 because of the uncertainty about the initial estimates of the system parameters. This in turns made it necessary to implement antireset windup [Åström and Wittenmark, 1984].

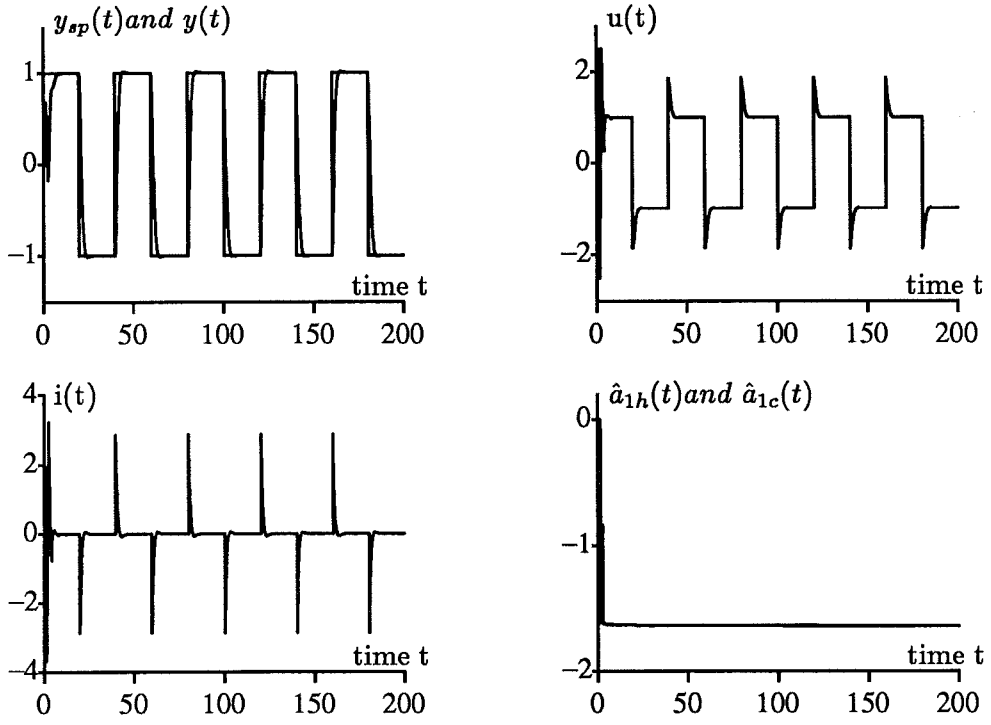

Figure 4.2 Adaptive control of a second order linear system.

The parameter estimates converged very quickly to accurate values and the controller performed fine then. The delay in the controller design algorithm

when changing domain does not show any effect here, because the models for the two domains are identical. □

The experiment was repeated for linear systems with lower gains. The results were just as good. However bigger control signals $u(t)$ were needed which makes sense.

EXAMPLE 4.2—A nonlinear system
The experiment was carried out for a piecewise linear system with the transfer functions $G(s) = g_h/((s+g_h)(s+1))$ and $G(s) = g_c/((s+g_c)(s+1))$ for the two domains ($g_h = 1, g_c = 0.02$). Notice that there is a factor of 50 of difference between the two gains.

The limitations for the input signal $u(t)$ were put up to 300. All other values were kept the same. The parameters are given in Fig.4.3.

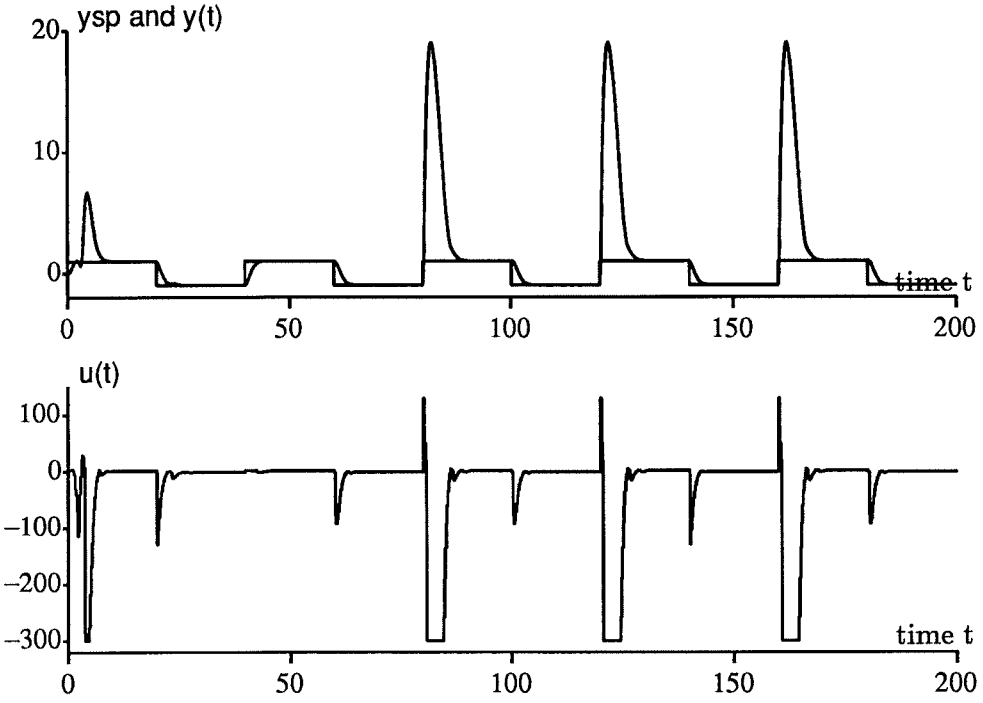

Figure 4.3 Adaptive control of a second order piecewise linear system.

The results are very bad this time, but they are as expected. Every time the system moves from the negative to the positive domain big overshoots occur for the system output $y(t)$. The reason is that in the negative domain big control signals are required, as the system has a very low gain. When it moves into the positive domain the controller uses the system parameters of the negative domain for another sampling period. The control signal is then much too large for the positive domain with a high gain of one. □

Improving the control algorithm with prediction

For a square wave set point $y_{sp}(t)$ it can be predicted that the system changes domain after every step. The algorithm was modified as follows. When the old set-point $y_{sp}(t)$ was negative and the new set point is positive (which means that a positive step occurred) it uses immediately the estimated system parameters for the positive domain. When the old setpoint was positive and the new set point is negative (which means that a negative step occurred) it uses immediately the estimated system parameters for the negative domain.

EXAMPLE 4.3
In the next figure the same system as in the previous example is controlled using prediction as described above.

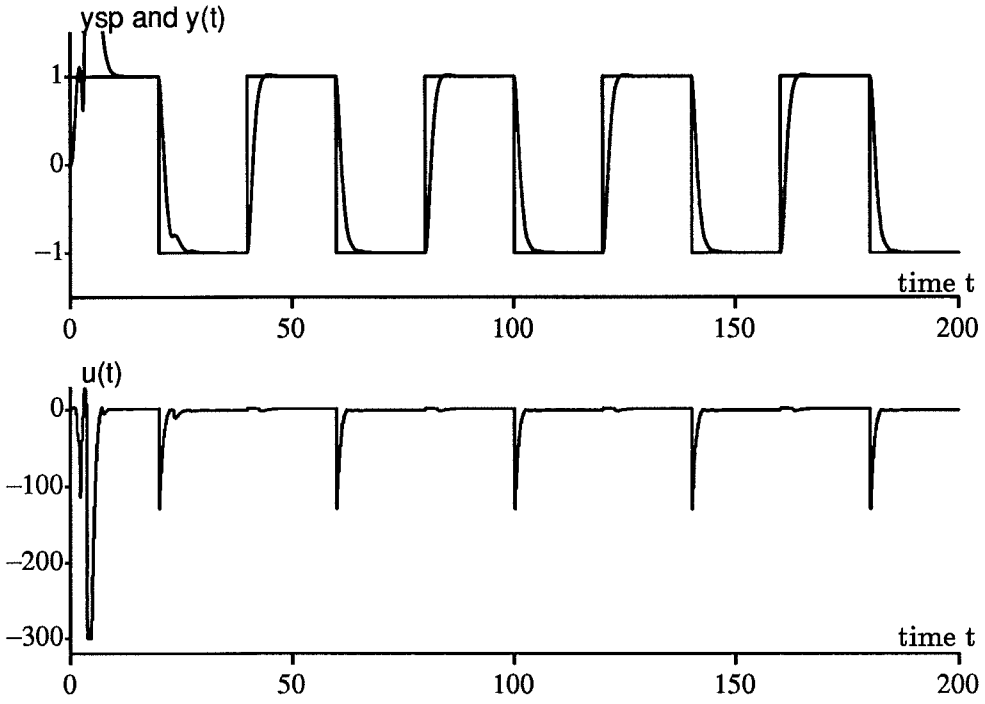

Figure 4.4 Improved control with prediction.

The results are very good this time. The parameters converge quickly for both domains and control performance is good. The experiment was repeated for even larger ratios $g_h : g_c$ and the results were just as good.

For systems with such big differences in gains between the two domains two sets of desired closed-loop performances in terms of desired closed-loop natural frequency and desired closed-loop damping should be specified in order to optimize the control algorithm. The controller should then switch between these two sets when changing domain.

The prediction method as described above is only applicable for special set points $y_{sp}(t)$ as square waves and steps. For other kinds of set point signals it might not detect all changes of domain. Or it might detect a change of domain by mistake. In order to reduce these risks the prediction method would have to be modified, i.e. it could look at differences between the values of new and recent set points or at rates of change of $y_{sp}(t)$.

The idea of prediction as described above could also be used for systems where the indicator $i(t)$ is not accessible. All decisions in the estimator that were based on the sign of the indicator $i(t)$ in this project would then be based on the new and recent values of the set point $y_{sp}(t)$.

Summary

It was found that when the change of domain can be predicted, the modified control algorithm performed well. For problems where this is not the case no satisfying solution has been found and it would be necessary to look for other control strategies.

5. Conclusions

Several problems in control of piecewise linear systems have been treated in this thesis. As a result some insights were obtained about the behaviour of this kind of systems.

A parameter estimation algorithm for piecewise linear systems was developed and tested for first and second order systems. It was found to be necessary to introduce a time-out mechanism in the estimator when the system changes domain in order to fill up the regression vector with appropriate data. This is an important consideration for all piecewise linear systems. Several problems caused by the time-out in the estimator were found and explained.

One difficulty arises for domains where the gain is high. The response of the system is very fast in such domains. In extreme cases the system might leave a high gain domain before any estimation could be performed. But also when this is not the case, the periods during which estimation can be performed are significantly reduced. This can slow down the parameter convergence considerably.

For square wave inputs it was found that their most valuable excitation can not contribute to system identification when the estimator with time-out is used. The reason is that the step changes in the square wave will cause a change of domain of the system which makes a time-out necessary.

With the knowledge gained some suggestions were made how to overcome these problems. These suggestions contain requirements about the nature of inputs of the system. A key issue here is that for the estimation of piecewise linear systems, excitation alone might not be sufficient in order to obtain accurate estimates. There must be excitation during the periods of time when estimation can be performed. Illustrations with examples were given.

Finally an adaptive controller for piecewise linear systems was developed, using pole placement design based on input-output models. An algebraic loop problem in the algorithm had to be overcome by introducing a time-delay. Problems as a result of this time-delay were solved using a prediction method. With this prediction method second order piecewise linear systems were successfully controlled for certain set-point signals.

This thesis was a first investigation about problems occuring when controlling piecewise linear systems adaptively. Further research must follow to obtain results that can be used in practice.

6. References

ÅSTRÖM, K. J. and B. WITTENMARK (1989): *Adaptive Control*, Addison-Wesley, Reading, Mass.

ÅSTRÖM, K. J. and B. WITTENMARK (1984): *Computer controlled systems*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

ELMQVIST, E., K.J. ÅSTRÖM, T. SCHÖNTHAL, and B. WITTENMARK (1990): *SimnonTM User's Guide for the MS-DOS Computers*, SSPA SYSTEMS, Box 24 001, S-400 22 Göteborg.

KNUTH, D. E. (1985): *The TEXbook*, Addison-Wesley, Reading, Mass.

SÖDERSTRÖM, T. and P. STOICA (1989): *System identification*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

Appendix A

1. A brief introduction to SIMNON

SIMNON is an interactive program that can simulate dynamical systems consisting of *continuous* and *discrete* subsystems. The subsystems are connected with a *connecting* system. It defines how the inputs and outputs of the various subsystems are interconnected (see Fig.1). For further information we refer to [Elmkvist, Åström, Schönthal and Wittenmark, 1990].

2. The programs for the self-tuning regulator

To simulate the control of a piecewise linear process with a self-tuning regulator three subsystems were defined.

1. The second order system *procs* is a *continuous* system and simulates the process which is controlled.
2. The regulator *strs* (self-tuning regulator for second order systems) is a *discrete* system and consists of two parts. One part is the recursive estimation algorithm. It estimates the parameters of the process model using the process input $u(t)$ and the process output $y(t)$. Notice that it is identical with the estimator used in chapter 3. The other part is the pole-placement algorithm. It uses the estimated model parameters to design an appropriate regulator and calculates the new process input $u(t)$.
3. The *connecting* system *conn* connects the regulator *strs* and the process *procs* and closes the control loop.

Fig.1 shows how the three systems exchange signals.

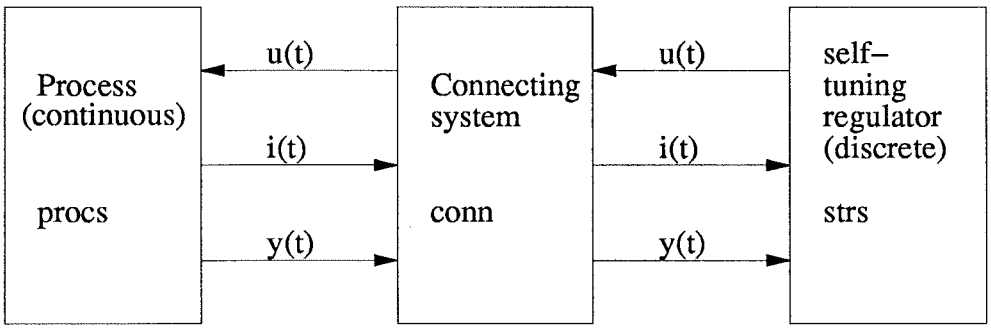

Figure 1. The exchange of signals between the process, the connecting system and the self-tuning regulator.

The program listings of the three subsystems are given on the following pages.

Notice that to simulate just the parameter estimation of the process model the same system can be used.

```
CONTINUOUS SYSTEM proc
"Second order system procs

INPUT u
OUTPUT y i
STATE  x1  x2
DER    dx1 dx2

y=x2
dx2=x1-x2
i=u-x1
dx1=IF i<0 THEN b*i ELSE a*i

a:1"0.02
b:0.02

END

CONNECTING SYSTEM conn
TIME t

u[proc]=u[est]
y[est]=y[proc]
i[est]=i[proc]

END


DISCRETE SYSTEM reg
"File called strs.t
"Indirect Self-Tuning Regulator based on the model
"   H(q)=(b1*q+b2)/(q^2+a1*q+a2)
"using standard recursive least squares estimation and pole placement
"design without cancellation of process zeros
"Polynomial Am has degree 2 and
"Polynomial Ao has degree 1
"Specifications are given in terms of continuous w and z

INPUT y i                       "set point, process output and indicator
OUTPUT u                        "control variable

STATE ysp1 y1 u1 v1             "controller states

STATE i1 i2                     "old indicator signals

STATE th1a th2a th3a th4a       "parameter estimates
STATE th1b th2b th3b th4b

STATE f1 f2 f3 f4               "regression variables

STATE p11a p12a p13a p14a       "covariance matrix a
STATE      p22a p23a p24a
STATE           p33a p34a
STATE                p44a

STATE p11b p12b p13b p14b       "covariance matrix b
STATE      p22b p23b p24b
STATE           p33b p34b
STATE                p44b

NEW nysp1 ny1 nu1 nv1
NEW ni1 ni2
NEW nth1a nth2a nth3a nth4a
NEW nth1b nth2b nth3b nth4b
```

```
NEW nf1 nf2 nf3 nf4

NEW n11a n12a n13a n14a
NEW      n22a n23a n24a
NEW           n33a n34a
NEW                n44a

NEW n11b n12b n13b n14b
NEW      n22b n23b n24b
NEW           n33b n34b
NEW                n44b

TIME t
TSAMP ts

INITIAL
"Compute sampled Am ------------------------------------------------

a=EXP(-z*w*h)
am1=-2*a*COS(w*h*SQRT(1-z*z))
am2=a*a

th1a=0                        "initial estimates
th2a=0
th3a=0.1
th4a=0.1

th1b=0
th2b=0
th3b=0.1
th4b=0.1

p11a=10000                   "initial covariances
p22a=10000
p33a=10000
p44a=10000

p11b=10000
p22b=10000
p33b=10000
p44b=10000

SORT
"1.0 Parameter Estimation ------------------------------------------

ysp=IF MOD(t,per)<per/2 THEN step ELSE -step

"1.1 Computation of P*f and estimator gain k

signal=abs(sign(i)+sign(i1)+sign(i2)) "check to see if the indicator has
                                      "been long enough in the new domain

tb1=p11b*f1+p12b*f2+p13b*f3+p14b*f4
ta1=p11a*f1+p12a*f2+p13a*f3+p14a*f4

pf1=IF signal<2.5 THEN 0 ELSE IF i<0 THEN tb1 ELSE ta1

tb2=p12b*f1+p22b*f2+p23b*f3+p24b*f4
ta2=p12a*f1+p22a*f2+p23a*f3+p24a*f4

pf2=IF signal<2.5 THEN 0 ELSE IF i<0 THEN tb2 ELSE ta2
```

```
tb3=p13b*f1+p23b*f2+p33b*f3+p34b*f4
ta3=p13a*f1+p23a*f2+p33a*f3+p34a*f4

pf3=IF signal<2.5 THEN 0 ELSE IF i<0 THEN tb3 ELSE ta3

tb4=p14b*f1+p24b*f2+p34b*f3+p44b*f4
ta4=p14a*f1+p24a*f2+p34a*f3+p44a*f4

pf4=IF signal<2.5 THEN 0 ELSE IF i<0 THEN tb4 ELSE ta4

denom=lambda+f1*pf1+f2*pf2+f3*pf3+f4*pf4

k1=pf1/denom
k2=pf2/denom
k3=pf3/denom
k4=pf4/denom

"1.2 Update estimates and covariances -------------------------------

tb9=y-f1*th1b-f2*th2b-f3*th3b-f4*th4b
ta9=y-f1*th1a-f2*th2a-f3*th3a-f4*th4a
eps=IF i<0 THEN tb9 ELSE ta9

nth1a=IF i<0 THEN th1a ELSE th1a+k1*eps
nth2a=IF i<0 THEN th2a ELSE th2a+k2*eps
nth3a=IF i<0 THEN th3a ELSE th3a+k3*eps
nth4a=IF i<0 THEN th4a ELSE th4a+k4*eps

nth1b=IF i<0 THEN th1b+k1*eps ELSE th1b
nth2b=IF i<0 THEN th2b+k2*eps ELSE th2b
nth3b=IF i<0 THEN th3b+k3*eps ELSE th3b
nth4b=IF i<0 THEN th4b+k4*eps ELSE th4b

n11a=IF signal<2.5 THEN p11a ELSE IF i<0 THEN p11a
                                  ELSE (p11a-pf1*k1)/lambda
n12a=IF signal<2.5 THEN p12a ELSE IF i<0 THEN p12a
                                  ELSE (p12a-pf1*k2)/lambda
n13a=IF signal<2.5 THEN p13a ELSE IF i<0 THEN p13a
                                  ELSE (p13a-pf1*k3)/lambda
n14a=IF signal<2.5 THEN p14a ELSE IF i<0 THEN p14a
                                  ELSE (p14a-pf1*k4)/lambda

n22a=IF signal<2.5 THEN p22a ELSE IF i<0 THEN p22a
                                  ELSE (p22a-pf2*k2)/lambda
n23a=IF signal<2.5 THEN p23a ELSE IF i<0 THEN p23a
                                  ELSE (p23a-pf2*k3)/lambda
n24a=IF signal<2.5 THEN p24a ELSE IF i<0 THEN p24a
                                  ELSE (p24a-pf2*k4)/lambda

n33a=IF signal<2.5 THEN p33a ELSE IF i<0 THEN p33a
                                  ELSE (p33a-pf3*k3)/lambda
n34a=IF signal<2.5 THEN p34a ELSE IF i<0 THEN p34a
                                  ELSE (p34a-pf3*k4)/lambda

n44a=IF signal<2.5 THEN p44a ELSE IF i<0 THEN p44a
                                  ELSE (p44a-pf4*k4)/lambda

n11b=IF signal<2.5 THEN p11b ELSE IF i<0 THEN (p11b-pf1*k1)/lambda
                                  ELSE p11b
n12b=IF signal<2.5 THEN p12b ELSE IF i<0 THEN (p12b-pf1*k2)/lambda
                                  ELSE p12b
n13b=IF signal<2.5 THEN p13b ELSE IF i<0 THEN (p13b-pf1*k3)/lambda
                                  ELSE p13b
```

```
n14b=IF signal<2.5 THEN p14b ELSE IF i<0 THEN (p14b-pf1*k4)/lambda
                                 ELSE p14b

n22b=IF signal<2.5 THEN p22b ELSE IF i<0 THEN (p22b-pf2*k2)/lambda
                                 ELSE p22b
n23b=IF signal<2.5 THEN p23b ELSE IF i<0 THEN (p23b-pf2*k3)/lambda
                                 ELSE p23b
n24b=IF signal<2.5 THEN p24b ELSE IF i<0 THEN (p24b-pf2*k4)/lambda
                                 ELSE p24b

n33b=IF signal<2.5 THEN p33b ELSE IF i<0 THEN (p33b-pf3*k3)/lambda
                                 ELSE p33b
n34b=IF signal<2.5 THEN p34b ELSE IF i<0 THEN (p34b-pf3*k4)/lambda
                                 ELSE p34b

n44b=IF signal<2.5 THEN p44b ELSE IF i<0 THEN (p44b-pf4*k4)/lambda
                                 ELSE p44b

"2.0 Control design ------------------------------------------------

"2.1 Choice of model parameters with prediction---------------------

a1tp=IF i1<0 THEN th1b ELSE th1a
a1=IF ysp1<0 AND ysp>0 THEN th1a ELSE IF ysp1>0 AND ysp<0 THEN th1b
                                 ELSE a1tp

a2tp=IF i1<0 THEN th2b ELSE th2a
a2=IF ysp1<0 AND ysp>0 THEN th2a ELSE IF ysp1>0 AND ysp<0 THEN th2b
                                 ELSE a2tp

b1tp=IF i1<0 THEN th3b ELSE th3a
b1=IF ysp1<0 AND ysp>0 THEN th3a ELSE IF ysp1>0 AND ysp<0 THEN th3b
                                 ELSE b1tp

b2tp=IF i1<0 THEN th4b ELSE th4a
b2=IF ysp1<0 AND ysp>0 THEN th4a ELSE IF ysp1>0 AND ysp<0 THEN th4b
                                 ELSE b2tp

"2.1 Solve the polynomial identity AR+BS=(B+)AoAm -------------------
bs=b1+b2
as=1+am1+am2
bmo=as/bs

ao=-aop

n=b2*b2-a1*b1*b2+a2*b1*b1

t0=bmo
t1=bmo*ao
r1=b2/b1+(b2*b2-am1*b1*b2+am2*b1*b1)*(-b2+ao*b1)/n/b1
s0=(ao+am1-a1-r1)/b1
s1=(ao*am2-a2*r1)/b2

"3.0 Control law with anti-windup ----------------------------------

v=-ao*v1+ysp*t0+t1*ysp1-s0*y-s1*y1+(ao-r1)*u1

u=IF v<-ulim THEN -ulim ELSE IF v<ulim THEN v ELSE ulim

"1.3 Update regression vector and old indicators-------------------

nf1=-y
nf2=f1
```

```
nf3=u
nf4=f3

ni1=i
ni2=i1

e=0

"3.1 Update controller state ----------------------------------------
ny1=y
nu1=u
nv1=v
nysp1=ysp

"4.0  Update sampling time ------------------------------------------

temp=IF abs(i)<0.01 THEN h/6 ELSE h
ts=t+h"temp

"Parameters ---------------------------------------------------------
lambda:0.99              "forgetting factor
h:0.2                    "sampling time
pi:3.141592
aop:0.4                  "observer pole
z:0.85"0.7               "desired closed loop damping
w:1.2"1                  "desired closed loop natural frequency

ulim:2.5                 "limitation of control signal

per:40                   "period of set point
step:1                   "amplitude of set point

END
```