Linköping Studies in Science and Technology. Dissertations
No. 802

Local and Piecewise Affine Approaches to System Identification

Jacob Roll

Department of Electrical Engineering
Linköping University, SE–581 83 Linköping, Sweden

Linköping 2003

Local and Piecewise Affine Approaches to System Identification

`roll@isy.liu.se`
`http://www.control.isy.liu.se`
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 91-7373-608-2 ISSN 0345-7524

Printed by Bokakademin, Linköping, Sweden 2003

Abstract

Identification of nonlinear systems is a multifaceted research area, with many diverse approaches and methods. This thesis considers two different approaches: (nonparametric) local modelling, and identification of piecewise affine systems.

Local models and methods predict the system behavior by constructing function estimates from observations in a local neighborhood of the point of interest. For many local methods, it turns out that the function estimates are in practice weighted sums of the observations, so a central question is how to choose the weights. Many of the existing methods are designed using asymptotic (in the number of observations) arguments, which may lead to problems when only few data are available. To avoid this, an approach named *direct weight optimization* is proposed, where an upper bound on the worst-case mean squared error is minimized directly with respect to the weights of a linear or affine estimator. It is shown that the estimator will have a finite bandwidth, and that it keeps several of the properties of an asymptotically optimal estimator.

The case when bounds on the estimated function and its derivatives are known *a priori* is also studied, and it is shown that one can sometimes, but not always, benefit from this extra information. The problem of estimating the function derivatives is also considered.

Another way of approaching the nonlinear system identification problem is to use a parameterized model class. *Piecewise affine systems* are an interesting class for this purpose. They have universal approximation properties, and are also closely related to hybrid systems. Here, an overview of different approaches appearing in the literature is presented, and a new identification method based on mixed-integer programming is proposed. One notable property of the latter method is that the global optimum is guaranteed to be found within a finite number of steps. The complexity of the mixed-integer programming approach is discussed, and its relations to existing approaches are pointed out. The special case of identification of Wiener models is considered in detail, since this model structure makes it possible to reduce the computational complexity. Some suboptimal modifications of the mixed-integer programming approach are also investigated.

As for hybrid systems in general, there has been a growing interest for piecewise affine systems in recent years, and they occur in many application areas. In many cases, safety is an important issue, and there is a need for tools that prove that certain states are never reached, or that some states are reached in finite time. The process of proving these kinds of statements is called *verification*. Many verification tools for hybrid systems have emerged in the last ten years. They all depend on a model of the system, which will in practice be an approximation of the real system. Therefore, it would be desirable to learn how large the model errors can be, before the verification is not valid anymore. In this thesis, a verification method for piecewise affine systems is presented, where bounds on the allowed model errors are given along with the verification.

Acknowledgments

Acknowledgments usually follow a rather standardized pattern. The main reason for this is probably that it is difficult to always find new ways of expressing one's gratitude. This acknowledgment is no exception. Nevertheless, even if the phrases are used many times before, I do mean all of them.

With this in mind, first of all I would like to thank my supervisor Professor Lennart Ljung, for excellent guidance and support during my time here at the Division of Automatic Control. I would also like to thank Professor Alexander Nazin, for a fruitful collaboration with many nice and interesting discussions, and for many valuable remarks on preliminary versions of the thesis. Also Dr. Alberto Bemporad deserves many thanks, for a nice and rewarding collaboration and for letting me visit the control group in the wonderful city of Siena.

The thesis has been proofread by Martin Enqvist, Markus Gerdin, David Lindgren, and Fredrik Tjärnström, for which I am extremely grateful. Earlier versions of the text have also been read by Ola Härkegård, Frida Gunnarsson, and Johan Löfberg. Your comments and remarks have been of great value and without doubt improved the quality of the thesis considerably.

I would also like to thank Måns Östring, Gustaf Hendeby, Rickard Karlsson, and Mikael Norrlöf for help with LaTeX problems and similar issues. Ulla Salaneck has always been helpful when it comes to administrative and practical problems, and deserves much gratitude. Ola Härkegård and Fredrik Tjärnström also deserve special thanks for putting up with all kinds of questions with a never-ending enthusiasm. In addition, many other people have been helpful during various phases of my work, and I thank all of you for your assistance.

This work has been supported by the ECSEL graduate school in Linköping, which is gratefully acknowledged.

The spirit and atmosphere in the Control and Communication group is a great source of inspiration, not to be forgotten, and I would like to thank everyone in the group for being a part of this.

Finally, I would like to thank Karin, for all the happiness we share and for always being there when I need you. You are the most valuable part of my life.

Linköping, March 2003

Jacob Roll

Contents

Notation

The following lists of symbols, acronyms, etc. are mainly intended to list notation that is used frequently in this thesis. Note that the same symbol may sometimes be used for different purposes. The numbers in the right column refer to sections where the notation is used or explained.

Symbols, Operators and Functions

$\mathbb{R}$	the set of real numbers
v_i	(if v is a vector) the ith element of v
M_i	(if M is a matrix) the ith row of M
M_{ij}	(if M is a matrix) the element in the ith row and jth column of M
I	identity matrix
$\mathrm{diag}(v)$	the diagonal matrix with the elements from the vector v as diagonal elements
$E[\cdot]$, $E[\cdot\vert\cdot]$	expectation, conditional expectation
$P(\cdot)$, $P(\cdot\vert\cdot)$	probability, conditional probability
$\Vert\cdot\Vert$	Euclidean norm
$\triangleq$	equal by definition
$\in$	belongs to

$S \subseteq P$	S is a subset of P	
$u(t)$	input at time t	1.1
$y(t)$	output at time t	1.1
$e(t)$	noise at time t	1.1
$x(t)$	state vector at time t	1.1
$f(\cdot)$	system function	1.1
$\varphi(t)$	regression vector at time t	1.1
$\dot{x}$	time derivative of x	1.1
$y_{t_1}^{t_2}$	$\{y(t_1), y(t_1+1), \ldots, y(t_2)\}$	1.1
$Z_{t_1}^{t_2}$	$\{u_{t_1}^{t_2}, y_{t_1}^{t_2}\}$	1.1
θ	parameter vector	1.1
n	dimension, e.g., of the state vector	1.1
N	number of data (observations)	1.2
n_a	number of output lags in $\varphi(t)$	1.2.1
n_b	number of input lags in $\varphi(t)$	1.2.1
∇f	gradient of f	1.2.3
φ_0	point of estimation	1.2.3
$\widetilde{\varphi}(k)$	$\varphi(k) - \varphi_0$	2.1
w_k	weights of a linear or affine estimator	2.1
σ	standard deviation of $e(t)$	2.1
L	Lipschitz constant	2.1
$\hat{f}(\varphi_0)$	estimate of $f(\varphi_0)$	2.1
$\mathcal{F}_{p+1}(L)$	function class with Lipschitz continuous pth derivatives	2.1
$\Sigma(\beta, L)$	Hölder class	2.1
$\mathcal{G}_{p+1}(L)$	function class of Taylor expansion type	2.1
$\mathcal{F}_{p+1}(L, \delta, \Delta)$	like $\mathcal{F}_{p+1}(L)$, but with bounds on $f(\varphi_0)$ and $\nabla f(\varphi_0)$	2.1
a	*a priori* estimate of $f(\varphi_0)$	2.1
δ	bound on $\lvert f(\varphi_0) - a\rvert$	2.1
b	*a priori* estimate of $\nabla f(\varphi_0)$	2.1
Δ	bound on $\lVert\nabla f(\varphi_0) - b\rVert$	2.1
$K(\cdot)$	kernel function	2.2
$K_h(\cdot)$	$K(\cdot/h)/h$	2.2
h	bandwidth of a kernel function K_h	2.2
$\Psi_k(K)$	$\int u^k K(u) du$	2.2
$r(K)$	$\int K^2(u) du$	2.2
$p_\varphi(\varphi(k))$	probability density function of $\varphi(k)$	2.2
Φ	matrix constructed of powers of $\widetilde{\varphi}(k)$	2.3
$\bar{K}_h$	$\operatorname{diag}(K_h(\widetilde{\varphi}(1)), \ldots, K_h(\widetilde{\varphi}(N)))$	2.3
Y	$\begin{pmatrix} y(1) & \ldots & y(N) \end{pmatrix}^T$	2.3
$\mathbf{e}_i$	ith standard basis vector	2.3
$\widehat{f^{(j)}}(\varphi_0)$	estimate of $f^{(j)}(\varphi_0)$	2.3
$O(h)$	$a(h) = O(h)$ as $h \to 0$ if $a(h)/h$ is bounded in a neighborhood of $h = 0$	2.4.1
$o(h)$	$a(h) = o(h)$ as $h \to 0$ if $a(h)/h \to 0$ as $h \to 0$	2.4.1

h_{AMSE}	asymptotically optimal (local) bandwidth	2.4.1
h_{AMISE}	asymptotically optimal (global) bandwidth	2.4.1
$P(\hat{f})$	mean squared prediction error	2.4.2
$\hat{P}(\hat{f})$	resubstitution estimate of $P(\hat{f})$	2.4.3
H	hat matrix	2.4.3
$\mathrm{tr}(M)$	trace of M	2.4.3
$\mathrm{infl}(\varphi_0)$	influence function	2.4.3
$\mathrm{sgn}(t)$	sign function	2.5.1
$\mu(\varphi)$	mean function of Gaussian process	2.6
$C(\varphi(i), \varphi(j))$	covariance function of Gaussian process	2.6
s	slack variables	3.1.1
w^*	optimal value of w	3.1.1
g	nonnegative highest-degree coefficient of the weight function	3.2.1
μ	Lagrangian multipliers corresponding to equality constraints	3.2.1
$\lambda^{\pm}$	Lagrangian multipliers corresponding to inequality constraints	3.2.1
r_k	$\mathrm{sgn}(w_k^*)$	3.2.2
n	number of nonzero weights w_k (for univariate function estimates)	3.2.2
1_n	n-dimensional vector with all elements equal to 1	3.2.2
$\widetilde{\varphi}_{1:n}$	$\big(\widetilde{\varphi}(1) \quad \dots \quad \widetilde{\varphi}(n)\big)^T$	3.2.2
ζ	denominator of the explicit expressions for w^*	3.2.2
Φ_n	$\big(1_n \quad \widetilde{\varphi}_{1:n}\big)$	3.2.4
$\asymp$	$a_N \asymp b_N \;\Leftrightarrow\; a_N/b_N \to 1,\ N \to \infty$	3.2.5
$f^{(p)}_{i_1 \dots i_p}$	partial pth derivative of f	4.1
$\mu^{(j)}_{i_1 \dots i_j}$	Lagrangian multipliers corresponding to equality constraints	4.2
$U_{\mathcal{F}}(w_0, w)$, $U_{\mathcal{F}}(w)$	upper bounds on the worst-case MSE for the function class $\mathcal{F}$	5.1.1
$\mathrm{sgn}_p(t)$	sign function with $\mathrm{sgn}_p(0) = p$	5.1.6
$U^1_{\mathcal{F}}(w_0, w)$, $U^1_{\mathcal{F}}(w)$	upper bounds on the worst-case MSE for a derivative estimator and the function class $\mathcal{F}$	6.1.1
$\hat{y}(t\|\theta)$	prediction of $y(t)$	8.1
$\varepsilon(t, \theta)$	residual, $\varepsilon(t, \theta) = y(t) - \hat{y}(t\|\theta)$	8.1
$V(\theta, Z_1^N)$	criterion function	8.1
$\ell_2(\varepsilon)$	$\ell_2(\varepsilon) = \varepsilon^2$	8.1
$\hat{\theta}$	parameter estimate	8.1
$\hat{\theta}^{(i)}$	value at iteration i in, e.g., a Newton algorithm	8.2
$\nabla^2 f$	Hessian of f	8.2
$A(v), B(v)$, $b(v), C(v)$, $D(v), d(v)$	system matrices in a piecewise affine system	9.1

v	key vector	9.1
C, d	matrices defining the set of switching hyperplanes	9.1.1
$X(v)$	regions of the different affine subsystems	9.1.1
$\{-1, 0, 1\}^M$	the set of vectors in $\mathbb{R}^M$, where each element has one of the values -1, 0, and 1	9.1.1
M	number of switching hyperplanes or hinges	9.1.1
$\theta(v)$	parameter vector of the subsystem v	9.2.1
M^+	number of positive hinges	9.2.2
θ_i	parameter vector of the ith hinge function	9.2.2
$\ell_1(\varepsilon)$	$\ell_1(\varepsilon) = \lvert\varepsilon\rvert$	10
$L_i(t)$, $U_i(t)$	lower and upper bounds on $\varphi^T(t)\theta_i$	11.1
$\prec, \preccurlyeq$	componentwise inequalities (for vectors)	11.1
V_1	1-norm criterion function	11.1.1
V_2	2-norm criterion function	11.1.1
$z_i(t)$	auxiliary real variables in MILP/MIQP reformulations	11.1.1
$\delta_i(t)$	auxiliary binary variables in MILP/MIQP reformulations	11.1.2
μ	arbitrarily small positive number, in practice chosen, e.g., to the machine precision	11.1.2
σ_i	auxiliary binary variables in MILP/MIQP reformulations	11.2.1
$\zeta_i(t)$	auxiliary real variables in MILP/MIQP reformulations	11.2.1
$\binom{N}{k}$	binomial term, $\binom{N}{k} = \frac{N!}{k!(N-k)!}$	11.3.1
q	delay operator, $q^{-1}x(t) = x(t-1)$	11.4
a_h, b_k	coefficients in the linear part of the Wiener model	11.4
α_i, β_i	coefficients in the nonlinearity of the Wiener model	11.4
L	length of the sliding window	11.6
$\Phi(t, \delta(t))$	$\big(\varphi^T(t) \quad \pm\delta_1(t)\varphi^T(t) \quad \dots \quad \pm\delta_M(t)\varphi^T(t)\big)^T$	11.7
$\Delta(v)$	uncertainty in $A(v)$	12.3
$\delta(v)$	uncertainty in $b(v)$	12.3
γ	uncertainty in d defining the position of the switching hyperplanes	12.3
$\overline{X}$	closure of the set X	12.4.1
λ	coefficients of convex combinations	12.4.2
$P_{[v^j]}$	matrix picking out the rows corresponding to the zero entries of v^j	12.4.4
$Q_{[v^j]}$	matrix picking out and scaling the rows corresponding to the nonzero entries of v^j	12.4.4
$f(n, N)$	number of regions into which $\mathbb{R}^n$ can be divided by N hyperplanes through the origin	A.5

Acronyms

AMISE	Asymptotic mean integrated squared error	2.4.1
AMSE	Asymptotic mean squared error	2.4.1
ARX	Autoregressive exogenous	1.1.1
DWO	Direct weight optimization	3.1
HL CPWL	High level canonical piecewise linear	9.2.3
HS	Hinging sigmoid	10.1.2
KKT	Karush-Kuhn-Tucker	3.2.1
LCV	Localized cross-validation	2.4.3
LP	Linear program(ming)	A.4
MILP	Mixed-integer linear program(ming)	A.4
MIQP	Mixed-integer quadratic program(ming)	A.4
MISE	Mean integrated squared error	2.4.1
MLD	Mixed logical dynamical	9.2
MSE	Mean squared error	2.4.1
NARX	Nonlinear autoregressive exogenous	1.1.1
NFIR	Nonlinear finite impulse response	1.1.2
PWA	Piecewise affine	1.1.2
PWARX	Piecewise autoregressive exogenous	9.2.1
QP	Quadratic program(ming)	3.1.1
SOCP	Second-order cone program(ming)	5.2
WMSE	Worst-case mean squared error	2.4.5

1 Introduction

Modelling, identification, and prediction are ubiquitous phenomena. Through our senses, we gather information about the world; we interpret, predict, and then react according to our perceptions. In natural science, long series of experiments and observations have led us to formulate laws of nature, which describe different aspects of the world and let us predict (again from observations) all sorts of things, like planet movements or tomorrow's weather. Also in technology, modelling and identification have much to offer. Everywhere around us, there is a need for automatic control mechanisms (to a higher or lower degree): in aeroplanes, cars, chemical process plants, mobile phones, heating of houses etc. However, to be able to control a system (like the heating of a house or a process plant), one needs to know at least something about how it behaves and reacts to different actions taken on it (control inputs). Hence, we need a model of the system.

A *system* can informally be defined as an entity which interacts with the rest of the world (other systems) through more or less well-defined input and output channels. A *model* is then a (more or less approximate) description of the system. For example, a car may be described as being affected by the maneuvers the driver makes with the steering wheel and pedals. These could hence be seen as input signals. The position and velocity of the car can be seen as output signals. By describing the relationship between the driver's maneuvers and the position and velocity of the car, we get a model of it. The relationships may be more or less accurately described, depending on what approximations are made. Of course, the model can also be made more detailed, e.g., by considering the aerodynamics of the car, friction, amount of petrol left in the tank, etc.

An ideal model should be both simple, accurate, and general. By simplicity, we could mean that it is simple to understand and interpret, easy to use, and that it leads to simple computations when making predictions. By accuracy is meant that the model describes the system well and makes accurate predictions. Generality means that the model should be able to handle many kinds of different situations. Unfortunately, there is an inherent conflict between these three ideal properties, which forces us to trade off between them. In the car model example, a simple model that does not consider aerodynamics and friction may work well when going straight forward with a low velocity. However, for high velocities and sharp turns, the effects of friction and aerodynamics can no longer be neglected, and we need a more complex model.

Instead of having one *global* model, which is general and covers many situations, and thus probably becomes complex, an alternative is to use different *local* models for each situation, or even to use a new model for each prediction. In this way the local models can be made simpler while still being able to make accurate predictions within their domains.

In this thesis, different approaches to the nonlinear system identification problem are considered, namely using *local modelling* and using *piecewise affine systems.* This chapter gives a brief introduction and an outline to the rest of the thesis. Apart from system identification, a *verification* method for piecewise affine systems is also considered. An introduction to verification can be found in Section 1.3.

1.1 Nonlinear Systems and Models

As mentioned, a general system has *output signals* $y(t)$, which can be observed/ measured, and *input signals* $u(t)$, with which the system can be affected. The system may also be disturbed by *noise* $e(t)$. Often (but not always) the system is assumed to be *causal*, which means that the outputs are determined only from what has happened in the past, not what happens in the future. A common way of describing systems is to use *state-space models.* In this kind of models, the history of the system is reflected in the *state vector* $x(t)$. The output $y(t)$ of the system thus depends on $x(t)$ (representing what has happened in the past), the input signal $u(t)$, and the noise $e(t)$. A fairly general, *continuous time* state-space model takes the form

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), e(t)) \\ y(t) &= g(x(t), u(t), e(t)) \end{aligned} \tag{1.1}$$

where $x(t) \in \mathbb{R}^n$, and $\dot{x}(t)$ as usual denotes the time derivative of $x(t)$. A special case of this form is considered in Part III.

In the model (1.1), the signals are time continuous, i.e., they are defined for all time points. This is of course a very natural way of describing physical signals. However, when observing the system, the signals can only be measured at a finite number of time points. Often the signals are sampled at regular time points, and so it makes sense to consider *discrete time* models. A discrete time state-space model can be written as[*]

$$\begin{aligned} x(t+1) &= f(x(t), u(t), e(t)) \\ y(t) &= g(x(t), u(t), e(t)) \end{aligned} \tag{1.2}$$

An alternative form is

$$y(t) = f(y(t-1), y(t-2), \dots, u(t-1), u(t-2), \dots, e(t), e(t-1), \dots) \tag{1.3}$$

i.e., $y(t)$ is a function of all previous input and output signals, and of the noise. For notational convenience, we will use the notation

$$\begin{aligned} y_{t_1}^{t_2} &\triangleq \{y(t_1), y(t_1+1), \dots, y(t_2)\} \\ Z_{t_1}^{t_2} &\triangleq \{u_{t_1}^{t_2}, y_{t_1}^{t_2}\} \end{aligned} \tag{1.4}$$

In this way, we can write (1.3) as

$$y(t) = f(Z_{-\infty}^{t-1}, e_{-\infty}^{t}) \tag{1.5}$$

A common assumption about the noise, covering most of the systems considered in this thesis, is given in the class of systems described by

$$y(t) = f(Z_{-\infty}^{t-1}) + e(t) \tag{1.6}$$

where $E[e(t)] = 0$ and $e(t)$ is independent of $Z_{-\infty}^{t-1}$ (see Section 1.1.3 for some more discussion about noise). Although being general, the form (1.6) is not very convenient in practice, though, since $Z_{-\infty}^{t-1}$ contains infinitely many elements. Instead, one can, e.g., use a model where the output $y(t)$ is a function of a fixed-length *regression vector* $\varphi(t)$[†]:

$$y(t) = f(\varphi(t)) + e(t) \tag{1.7}$$

The regression vector $\varphi(t)$ can be composed of, e.g., old inputs and outputs. This kind of models in regression form is going to be used in Parts I and II.

If a model class can be represented by a model which is completely determined by a number of *parameters*, we speak about a *parameterized* model class. The parameters are often collected in a *parameter vector* θ. Examples of this are given in the following sections.

1.1.1 Linear Models

Linear models (in the data) form a subclass of the general models (1.1), (1.2), and (1.5). For these models f is a linear function of the data, so that, e.g., a linear regression model (1.7) can be written as

$$y(t) = \varphi^T(t)\theta + e(t) \tag{1.8}$$

[*]It should be emphasized that f and g in this section represent general functions, and are not (necessarily) the same functions in (1.1) and (1.2).

[†]Another alternative is to use the notion of an initial state $x(0)$, and let the system class be described by $y(t) = f(Z_1^{t-1}, x(0)) + e(t)$

where $\varphi(t)$ consists of past data which enters linearly, and θ is the parameter vector. Depending on what elements are included in the regression vector $\varphi(t)$, one distinguishes between different model classes (see [142, 146]):

- FIR (Finite Impulse Response) models have a regression vector which consists only of old inputs $u(t-\tau)$, i.e.,

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

  for a given positive integer n_b.

- AR (AutoRegressive) models include only old outputs $y(t-\tau)$, $\tau > 0$, in $\varphi(t)$.

- ARX (AutoRegressive eXogenous) models include both old inputs and outputs in $\varphi(t)$.

Apart from these options, the regression vector may also depend on the parameters, i.e.,

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

This is not a true linear regression form, but is often called *pseudo-linear regression*. Examples of such models are ARMAX (AutoRegressive Moving Average eXogenous) models, OE (Output Error) models, and BJ (Box-Jenkins) models. For more details, see, e.g., [94].

Often, one refers to a model like (1.8) as being *linear in the parameters*, since $y(t)$ depends linearly on θ. A model may of course be linear in the parameters but nonlinear in the data, e.g., if $\varphi(t)$ in (1.8) is a nonlinear function of past data.

1.1.2 Some Specific Nonlinear Model Classes

Let us now briefly mention some specific nonlinear model classes. First, let us return to the nonlinear regression models (1.7). Like in the linear case, we can distinguish between different model classes depending on the construction of the regression vector [142]: If $\varphi(t)$ consists only of old inputs $u(t-\tau)$, the model is called an NFIR (Nonlinear Finite Impulse Response) model; if $\varphi(t)$ consists of both old inputs and outputs it is an NARX model, etc. Models of these types will be used throughout the thesis.

A special class of nonlinear systems is obtained if the functions f and g in (1.1) or (1.2) are piecewise affine. In the last years, there has been a growing interest in these *piecewise affine (PWA) systems*, partly because of their close relationship to *hybrid systems*. Hybrid systems are systems that have both continuous and discrete dynamics. A simple example could be a physical system with continuous dynamics, controlled by a discrete controller. The continuous dynamics of hybrid systems is typically associated with physical systems (possibly controlled by continuous controllers), while the discrete dynamics for instance may come from discrete controllers, inherent nonlinearities in the physical system, on/off switches, or external discrete events influencing the system.

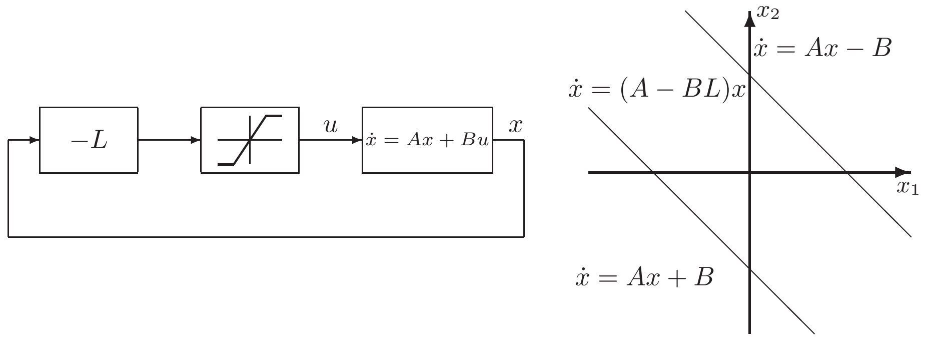


Figure 1.1: A linear system, controlled by linear feedback. Since the control signal is bounded, the system is a piecewise affine system.

Due to this hybrid structure, it can be hard to use, e.g., linearization techniques for control design and analysis of such systems. Therefore, there has been a need for developing the theory of hybrid systems.

The piecewise affine systems can be described as consisting of several affine subsystems, between which switchings occur at different occasions. Such systems occur in many applications, e.g., when there are physical limits (such as a tank that can get full or empty), bounds on the control signal, dead-zones, or when the system contains switches and thresholds. Piecewise affine systems can also be used to approximate nonlinear systems, since they have *universal approximation properties*, which essentially means that any (sufficiently smooth) nonlinear function can be arbitrarily well approximated by a piecewise affine function.

Example 1.1 (Bounded signal) *Consider the system in Figure 1.1, where a linear system is controlled by linear feedback, but where the control signal is bounded by $|u| \leq 1$. As long as the control signal is kept within the bounds, the system is linear, but when $|Lx| > 1$, the dynamics is changed from $\dot{x} = (A - BL)x$ to $\dot{x} = Ax \pm B$. We get a piecewise affine system with three subsystems, as shown in the figure.*

Example 1.2 (Temperature control) *To control the temperature in a house, the heating system often uses thermostats, that switch the radiators on or off when the temperature reaches certain thresholds. This will give a piecewise affine behavior, where one affine system is given by the radiators being turned off, and the other by the radiators being turned on.*

Piecewise affine systems are described in more detail in Chapter 9.

Many classes of piecewise affine systems can be seen as special cases of the class of function expansion type models, which can be written in the form

$$y(t) = \sum_{k=1}^{r} \alpha_k g_k(\varphi(t), \beta_k, \gamma_k) \tag{1.9}$$

This is a very broad class of models, and includes (apart from many piecewise affine systems), e.g., feedforward neural networks, radial basis networks, Fourier series approximations, and wavelets. See [142] for more details.

1.1.3 Noise

The noise of the system can be modelled in several different ways. A common way in linear modelling is to model the noise as an additive term, constructed by independent, identically distributed stochastic variables with zero mean (white noise), filtered through a linear filter (the different classes ARX, ARMAX, OE, and BJ mentioned in Section 1.1.1 basically differ by their different types of noise models). In nonlinear modelling, there are several ways in which this could be extended. For example, the noise could be filtered through a nonlinear filter, it could enter the system in a multiplicative way (e.g., we could have products like $u(t)e(t)$), etc. As already mentioned, in this thesis we are going to restrict the noise models to an additive term

$$y(t) = f(Z_{-\infty}^{t-1}) + e(t)$$

where $E[e(t)] = 0$ and $e(t)$ is independent of $Z_{-\infty}^{t-1}$ (in Part I, $e(t)$ should be independent of all regression vectors). However, in Chapter 12, some terms could be interpreted as multiplicative noise, as we will see. The Wiener systems in Section 11.4 will also have a slightly different (although still additive) noise contribution, where the noise enters into the middle of the system (see (11.48)).

In Chapter 12, it will also be natural to assume that the noise is *unknown but bounded*, i.e., instead of specifying a probability density function for the noise, we specify a certain region within which the noise may take any value. This is a common approach, e.g., in set membership identification and robust control [52, 107].

1.2 System Identification

In general, the system identification problem is the problem of mathematically describing the relation between a collection of signals as well as possible, given experimental sets of data from the signals. Here, as in most settings, we are interested in the relation between the input signals, $u(t)$, and the output signals, $y(t)$. We assume that the experimental data consists of two time-series, $u(t)$ and $y(t)$, $t = 1, \dots, N$. The goal is to describe $y(t)$ as a function of previous outputs and inputs, together with some noise, $e(t)$, i.e., like in (1.5). As already mentioned, in this thesis we will mostly consider the form (1.6). For a more thorough introduction to system identification, see, e.g., [94].

1.2.1 Prediction Error Methods

To be able to identify a system, we must know, or at least assume, something about its structure. If nothing whatsoever is known about the system function, the identification problem is meaningless – the only things we can say are statements like: "At time t and for the input $u(1), \dots, u(t)$, the output was $y(t)$ (at least this time!)". In other words, the best description of the system we can get is the set of experimental data itself.

If, however, we can assume that the system can be described (reasonably well) by a model belonging to a certain model class, more can be said. The model class can be specified in different ways. For instance, the classes described in Section 1.1.1 and by (1.9) are examples of *parameterized* model classes, i.e., the class can be represented by a parameterized model. When using such a model class, the system identification problem amounts to finding values of the parameters, such that the resulting model describes the system as well as possible. Such methods are called *parametric methods.* One family of identification methods, containing many well-known and commonly used approaches, are the *prediction error methods (PEM)*, which are described in Chapter 8. The methods in Part II are also of this kind. Basically, the prediction error methods form parameterized predictions $\hat{y}(t|\theta)$, where θ is the *parameter vector.* From these the *residuals*

$$\varepsilon(t, \theta) = y(t) - \hat{y}(t|\theta)$$

i.e., the differences between the true output and the predicted output are formed. The parameter vector is then chosen to minimize the residuals according to some criterion. A very common criterion is the *least-squares criterion*

$$V(\theta, Z_1^N) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta) = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t|\theta))^2$$

Let us conclude this section by considering a simple example.

Example 1.3 (ARX model) *As mentioned in Section 1.1.1, an ARX model is a model with the following structure:*

$$\begin{aligned} y(t) &= -a_1 y(t-1) - \dots - a_{n_a} y(t-n_a) \\ &\qquad + b_1 u(t-1) + \dots + b_{n_b} u(t-n_b) + e(t) \\ &= \varphi^T(t)\theta + e(t) \end{aligned}$$

where

$$\begin{aligned} \varphi(t) &= \begin{pmatrix} -y(t-1) & \dots & -y(t-n_a) & u(t-1) & \dots & u(t-n_b) \end{pmatrix}^T \\ \theta &= \begin{pmatrix} a_1 & \dots & a_{n_a} & b_1 & \dots & b_{n_b} \end{pmatrix}^T \end{aligned}$$

The natural predicted output is

$$\hat{y}(t|\theta) = \varphi^T(t)\theta$$

so the least squares criterion is given by

$$V(\theta, Z_1^N) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon^2(t,\theta) = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \varphi^T(t)\theta)^2$$

which is minimized by

$$\hat{\theta} = \left(\sum_{t=1}^{N} \varphi(t)\varphi^T(t)\right)^{-1} \sum_{t=1}^{N} \varphi(t)y(t)$$

Hence

$$\begin{aligned}\hat{y}(t|\hat{\theta}) &= \varphi^T(t)\left(\sum_{l=1}^{N} \varphi(l)\varphi^T(l)\right)^{-1} \sum_{k=1}^{N} \varphi(k)y(k) \\ &= \sum_{k=1}^{N} w_k(\varphi(t))y(k)\end{aligned}$$

where

$$w_k(\varphi(t)) = \varphi^T(t)\left(\sum_{l=1}^{N} \varphi(l)\varphi^T(l)\right)^{-1} \varphi(k) \tag{1.10}$$

In other words, we can see the prediction $\hat{y}(t|\hat{\theta})$ as a weighted sum of the observed outputs. This perspective will be relevant in Part I.

1.2.2 Identification of Piecewise Affine Systems

Many tools and methods for verification, as well as for control, stability analysis, etc., of hybrid systems have emerged in recent years. To be able to use these tools, however, a model of the system is needed.

Identification of hybrid systems (e.g., piecewise affine systems) is an area that is related to many other research fields within nonlinear system identification. In particular, one can find several different methods and approaches which are applicable, or at least related to the piecewise affine system identification problem. Some examples of approaches that result in piecewise affine systems are neural networks with piecewise affine perceptrons [8, 51, 86], Chua's canonical representation and hinging hyperplanes [22, 27, 28, 77, 79, 80, 124], self-exciting threshold autoregressive (SETAR) models for time-series analysis [104, 105], special-purpose methods for physical applications [81], and some function approximation approaches [76, 78]. In [142], which is a good overview of different nonlinear identification techniques, the relations between several different approaches are explored.

The piecewise affine system identification problem will be considered in Part II, where an overview of different approaches occurring in the literature in Chapter 10 will be given. An approach based on mixed-integer programming will also be presented in Chapter 11. This approach guarantees that an optimal model (with respect to the particular criterion used and the experimental data available) is found, but this guarantee comes at a price of greater computational complexity.

In Chapters 10 and 11, we will mostly consider models in regression form like in (1.7). It turns out that once the partitioning of the state-space is known, the piecewise affine system identification problem reduces to a problem, comparable to a linear system identification problem in terms of complexity. However, finding the best partition may be a very complex problem. Hence, there are two fundamental approaches: Either an a priori partition can be used, or the partitioning can be estimated along with the different subsystems. The latter can be done simultaneously or iteratively. The first approach gives a simple estimation process, but the number of regions needed to give enough flexibility in the model structure may be very large. In the second approach, the number of subsystems can be kept low, but the estimation will be more complex. This dilemma will be treated more in detail in Chapters 10 and 11.

Approximating Nonlinear Systems by Piecewise Affine Systems

As previously mentioned, many classes of piecewise affine systems have universal approximation properties, which make them suitable for approximating arbitrary nonlinear functions. Therefore, an efficient identification method for piecewise affine systems would also be of interest for identification of nonlinear functions. In this case, the exact shape of the regions is of less direct importance, since the true system does not consist of affine subsystems. Instead, being able to approximate the nonlinear function well, using few parameters and a representation that is easy to handle, becomes the main question.

1.2.3 Nonparametric Methods and Local Modelling

In Section 1.2.1, we saw that one way of specifying a model class is to represent it by a parameterized model. An alternative way could be to specify different properties that the model should satisfy. For instance, we could assume that the models can be described by (1.7), where f is continuously differentiable, and the derivative satisfies a *Lipschitz condition*

$$\|\nabla f(\varphi(1)) - \nabla f(\varphi(2))\| \leq L\|\varphi(1) - \varphi(2)\| \quad \forall\, \varphi(1), \varphi(2) \in \mathbb{R}^n \tag{1.11}$$

Here, and throughout this thesis, $\|\cdot\|$ stands for the Euclidean norm (if nothing else is stated). We can also assume that the noise variance is finite (and given). In this case, it is hard to parameterize the model class, so either we have to approximate the model class by a parameterized class, or we have to use a *nonparametric* identification method.

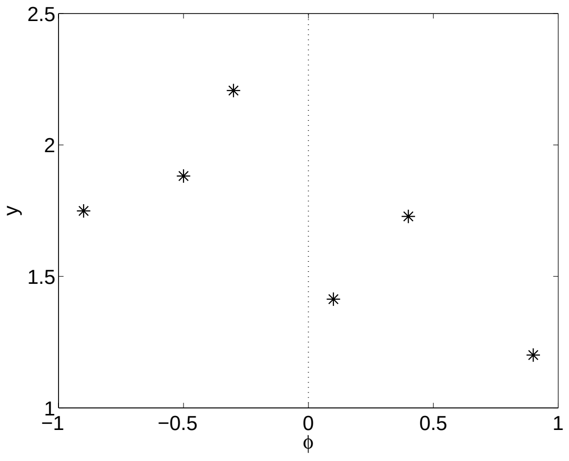


Figure 1.2: Data set for Example 1.4.

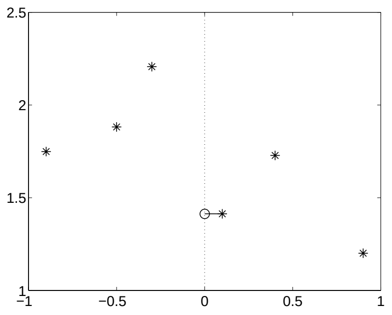
Specifying a Lipschitz condition limits the possible variation of the system function f. This means, that if we would like to estimate $f(\varphi_0)$ at a certain point φ_0, we can find some information about the value of $f(\varphi_0)$ by looking at the values $y(t)$ corresponding to regression vectors $\varphi(t)$ that are close to φ_0. Hence, it is natural to use some kind of *local modelling* to describe the system. In statistics, the interest has been focused on various local methods, like kernel methods, [113, 156], local polynomial approaches, [47], and trees, [24] (see also [138]). Some local modelling approaches are described in Chapter 2, and in Chapter 3 a local modelling method, named a *direct weight optimization (DWO)* approach, is proposed.

To get a first feeling of the problem, let us consider some very simple estimation methods.

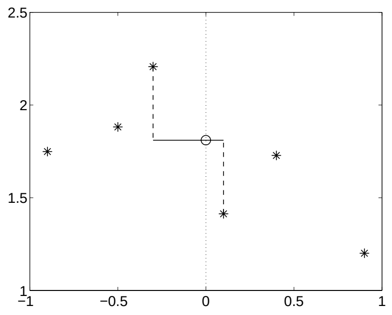
Example 1.4 (Function estimation) *Assume that we are given a set of noisy data samples* $(\varphi(t), y(t))$, $t = 1, \dots, N$, *coming from an unknown function* $f(\varphi)$ *in accordance with* (1.7), *and would like to estimate the value of* $f(0)$. *Let us denote the estimated value by* $\hat{f}(0)$. *An example data set is plotted in Figure 1.2.*

There are many ways to estimate $f(0)$. *A very simple option is the* nearest neighbor *method illustrated in Figure 1.3(a), where we consider the data sample which is closest to* 0, *and take its value as the estimate of* $f(0)$. *This approach is apparently very sensitive to noise, as it considers just one data sample. A slight generalization is the* k-nearest neighbor *method, where the estimate is given by the mean of the* k *closest data samples. This is illustrated in Figure 1.3(b) for* $k = 2$.

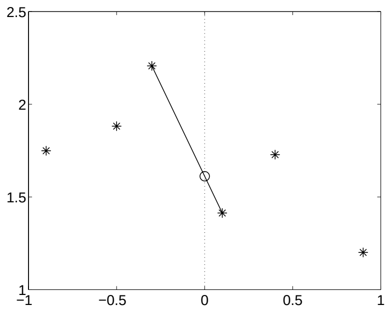
We can notice that both these methods can be regarded as taking a weighted sum of the experimental function values to obtain the desired estimate $\hat{f}(0)$. *In*



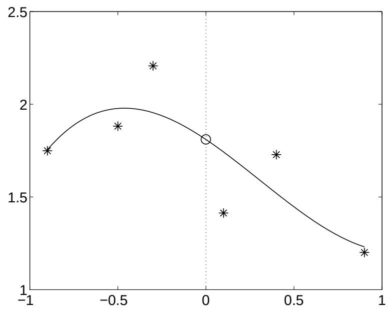
(a) Nearest neighbor.



(b) k-nearest neighbor.



(c) Linear interpolation.



(d) Cubic regression.

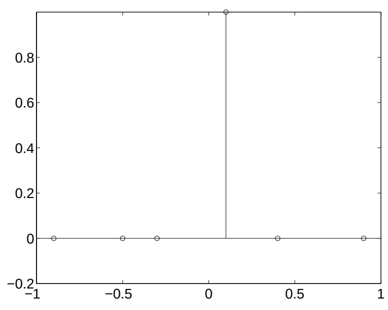
Figure 1.3: The different methods for function estimation described in Example 1.4. The estimate $\hat{f}(0)$ is marked by a circle in each case.

other words, we can write both estimates in the form

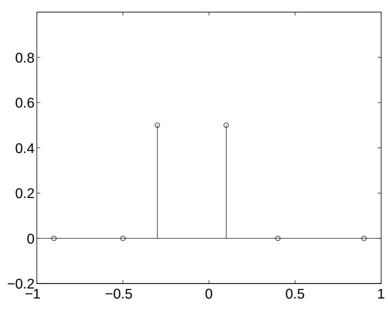
$$\hat{f}(0) = \sum_{t=1}^{N} w_t y(t) \tag{1.12}$$

where the values of the weights w_t *depend on the method we use, and of the values of* $\varphi(t)$*. For the nearest neighbor approach, one weight (the one corresponding to the closest data sample) will be equal to* 1*, and the rest of the weights will be zero. For the* k-nearest neighbor*, on the other hand,* k *weights (corresponding to the* k *closest data samples) will equal* $1/k$*, and the others are zero. The weights are plotted in Figure 1.4(a) and Figure 1.4(b), respectively.*

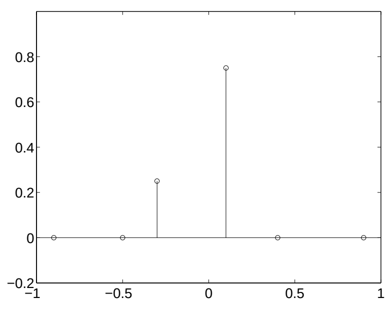
A third option, shown in Figure 1.3(c), is to make a linear interpolation to obtain the estimate. It turns out that this method also results in an estimate in the form (1.12)*: Let* $\varphi(i)$ *and* $\varphi(j)$ *be the two values which are closest to* 0 *(one*



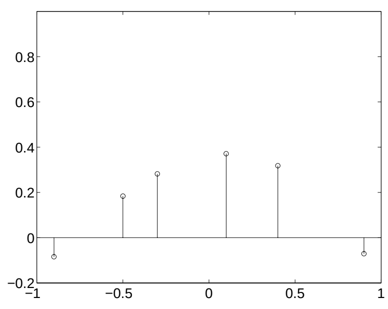
(a) Nearest neighbor.



(b) k-nearest neighbor.



(c) Linear interpolation.



(d) Cubic regression.

Figure 1.4: The corresponding weights for the methods in Figure 1.3.

positive and one negative). The line between $(\varphi(i), y(i))$ *and* $(\varphi(j), y(j))$ *can be written as*

$$l(\varphi) = \frac{\varphi - \varphi(j)}{\varphi(i) - \varphi(j)} y(i) + \frac{\varphi - \varphi(i)}{\varphi(j) - \varphi(i)} y(j)$$

and since we are interested in the value at $\varphi = 0$*, the function estimate becomes* $\hat{f}(0) = l(0)$*. Hence, the weights we get are*

$$w_i = \frac{\varphi(j)}{\varphi(j) - \varphi(i)}, \qquad w_j = \frac{\varphi(i)}{\varphi(i) - \varphi(j)}$$

The weights are plotted in Figure 1.4(c).

Yet another approach is to fit, e.g., a third-order polynomial to the data by minimizing a least-squares criterion (see Figure 1.3(d)). Simple calculations (similar to the ones in Example 1.3) show that also this method gives an estimate in the form (1.12)*, with the weights shown in Figure 1.4(d).*

Note that the first three methods of Example 1.4 are local methods, while the last method is global. However, also the last method can be made local, by only considering the data samples which are closest to the point of interest ($\varphi_0 = 0$ in Example 1.4) when fitting the polynomial. The method then becomes a *local polynomial modelling* method. This kind of methods is described in greater detail in Chapter 2.

Since many methods lead to a linear estimator of the kind given in (1.12) (see also the predictor in Example 1.3), we can view the estimation problem as a problem of finding appropriate weights for (1.12). No matter what type of local modelling approach is taken, the central problem is which of the data samples should be taken into consideration when forming the estimate (i.e., which weights should be nonzero). Intuitively, it is clear that the answer must depend on three items:

1. How many data are available (and how are they spread)?

2. How smooth is the function surface (supposed to be)?

3. How much noise is there in the observations?

This problem has been studied extensively in the statistical literature, and there are several solutions based on asymptotic (in the number of observations) analysis.

In Part I, another solution, which is not based on the asymptotic behavior of the estimates, is proposed. Based on the smoothness measure given by the Lipschitz condition (1.11) and the noise variance, we compute a uniform upper bound of the mean squared error (MSE) of a linear estimate, as a function of the estimator parameters. This upper bound is then minimized directly with respect to the weights in (1.12). It turns out that this problem can be reformulated as a quadratic programming (QP) problem, which can be solved efficiently. It also turns out that this solution has many of the key features of the asymptotically optimal estimators, but for finite number of observations it produces better guaranteed error bounds.

1.3 Verification

In many application areas, such as in chemical industry and aeronautics, safety is a very important issue. For example, given certain assumptions on the system, one would like to be able to ensure that the system never reaches certain specified bad (or dangerous) states. Typical requirements for control design might therefore include that the system should never reach some specific (possibly dangerous) states, that the system should reach a certain region in the state-space, and/or that there should be invariant regions (once you get there, you will never leave the region). After the design process, one would often like to make sure that the specified requirements are satisfied. This process is known as *verification.*

Even if a system model is given, it is mostly just an approximation – good or bad – of the real system. Therefore, in Part III, the verification problem is considered for piecewise affine systems with model uncertainties. A *robust verification* method is presented, where upper bounds on the uncertainties that can be tolerated are computed along with the verification process. This section provides a short description of the concept of verification.

Solving a verification problem exactly is in general possible only for some restricted classes of hybrid systems [2, 4]. Many verification methods for the problem of avoiding bad states found in the literature are based on computing a conservative approximation of the system [2, 3, 14, 29–31, 43, 84, 150]. This means that either the model of the system is replaced by a (computationally) simpler model, which is an *outer approximation* of the original system, or that the trajectories from a given initial set of states are replaced by an outer approximation. An outer approximation is an approximation that guarantees that if a trajectory is allowed by the original system, it is also allowed in the simplified model. In this way it can be guaranteed that if a certain bad state is never reached in the simple model, it is never reached in the original system either. A good overview over numerous different approaches is given in [44].

For the problem of assuring that a certain region is reached, an *inner approximation* can be used analogously to what is described above (see [43]). If a transition is guaranteed to occur in an inner approximation, it is also guaranteed to occur in the original system. However, other techniques might be needed to compute the inner approximation.

1.3.1 Robust Verification

The verification method presented in Chapter 12 is partly built upon a method presented in [43]. For this method, we will consider continuous time piecewise affine systems in state-space form, where the dynamics depend on in which region of the state-space the current state $x(t)$ is (see (9.2)). The regions (denoted $X(v)$) are assumed to be polyhedral. To verify the desired properties, the behavior of the vector field $\dot{x}(t)$ at the borders of the regions $X(v)$ is analyzed. Specifically, questions such as "At a given face of the polyhedron $X(v)$, is there a point, x_0, such that $\dot{x}_0$ is pointing out of $X(v)$, or are all trajectories at this face going into $X(v)$?" are answered (this kind of computations has also been used by others, e.g., by [72]). The information obtained is used to determine which transitions between different regions are possible, which transitions are guaranteed to *occur nondeterministically* (i.e., one transition out of a set of transitions from a given polyhedron is guaranteed to occur) and which are not. Then finite automata are constructed, showing the guaranteed or possible transitions. The finite automata give an approximation of the system, and can be used for different kinds of verification. For example, we can guarantee that certain states in the original system are not reachable from some other initial states, by proving that there is no sequence of possible transitions in the finite automata, taking the system state from the region of the initial states to the region of the final states.

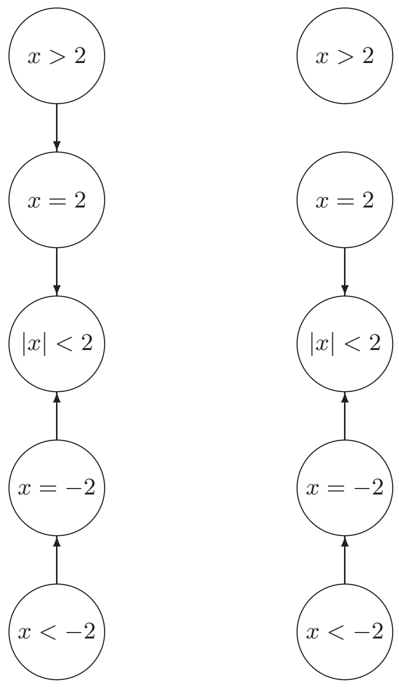


Figure 1.5: Automata showing possible (left figure) and guaranteed (right figure) transitions for the system in Example 1.5.

Example 1.5 (Verification) *Consider the simple system*

$$\dot{x} = \begin{cases} -2x & x < -2 \\ -x+1 & -2 \le x \le 2 \\ x-3 & x > 2 \end{cases}$$

Suppose that we would like to make sure that if $x(0) \ge -2$, $x(t)$ will never get below -2. This can easily be verified by considering the trajectories at $x = -2$. Here, $\dot{x} = -x + 1 = -(-2) + 1 = 3$, which means that if x gets close to -2 (from above), it will increase, and hence never pass the border $x = -2$.

By making calculations like above, we can construct a finite automaton showing what transitions between the three regions are possible (see the left automaton in Figure 1.5). This automaton is an outer approximation of the original system. Apart from the property shown above, we can see, e.g., that once we get into the region $|x| < 2$ we never leave it.

We can also construct an automaton which is an inner approximation of the system (the right automaton in Figure 1.5). With the help of this automaton we can guarantee that we will end up in the region $|x| < 2$, if starting with $x \le 2$.

Like all other methods mentioned above, the method in [43] assumes that a model of the system is given. It would be desirable to be able to perform the verification, and simultaneously compute how sensitive the verification proof (e.g., the approximating automata) is to changes in the underlying systems, both in the dynamics and in the switching surfaces. This is what is called *robust verification* in this thesis. Such information could be used to get a measure of how robust the verification process is to model errors, or as an aid in a control design process, if we would like to adjust the system dynamics without losing the verified property. Sometimes we would only be interested in that some crucial transitions should not change, whereas in other cases we might want the entire approximating automata to remain invariant.

Since the approximating method in [43] considers the behavior of $\dot{x}(t)$ at the borders of the regions $X(v)$, we must determine how this behavior changes with varying dynamics of the submodel corresponding to $X(v)$, and with translations of the surfaces that bound $X(v)$. How this can be done is the topic of Chapter 12.

1.4 Thesis Outline

The thesis consists of three main parts:

- Local modelling and the direct weight optimization (DWO) approach can be found in Part I. Chapter 2 gives an overview of the problem and of existing methods. In Chapter 3 the DWO approach is introduced for a special class of once differentiable, univariate functions. This is then extended in Chapter 4 to multivariate functions, with higher degrees of differentiability. Chapter 5 considers the case when bounds on the function value and derivatives are known. Some other extensions are given in Chapter 6 and conclusions in Chapter 7.

- In Part II, the problem of identification of piecewise affine systems is studied. Chapter 8 gives a brief introduction to prediction error methods. In Chapter 9, different classes of piecewise affine systems are presented. Chapter 10 gives a survey of existing identification approaches, while Chapter 11 considers the identification of piecewise affine systems using mixed-integer linear or quadratic programming.

- Part III, which consists of Chapter 12, concerns robust verification for piecewise affine systems.

Apart from this, some mathematical preliminaries are given in Appendix A.

1.5 Contributions

The main contributions of this thesis are:

- A method for finding the function estimate that minimizes an upper bound on the worst-case mean squared error (MSE) through quadratic programming (QP), as outlined in the DWO approach in Chapters 3 (for once differentiable, univariate functions) and 4 (for p times differentiable, multivariate functions).
- Extension of the DWO approach to the case when bounds on the function and its derivatives are known *a priori*, presented in Chapter 5.
- Extension of the DWO approach to the estimation of derivatives using QP, given in Section 6.1.
- The algorithm in Section 6.2 for finding the function estimate minimizing the exact worst-case MSE for once differentiable, univariate functions, with derivatives satisfying a Lipschitz condition.
- The techniques in Chapter 11 of using mixed-integer linear/quadratic programming (MILP/MIQP) to identify piecewise affine systems. A technique of reformulating products of continuous affine functions and discrete variables as linear inequalities, described in Section 11.1.2.
- Extension of an identification algorithm proposed by Hush and Horne [69], and showing its relationship to the MILP/MIQP formulations. This is found in Section 11.7.
- The robust verification methods in Chapter 12.

Some of the material in this thesis has been presented previously. The DWO approach in Part I has been presented for once differentiable, univariate functions in

> J. Roll, A. Nazin, and L. Ljung. A non-asymptotic approach to local modelling. In *The 41st IEEE Conference on Decision and Control*, pages 638–643, Dec. 2002

The extension to multivariate functions was considered in

> J. Roll, A. Nazin, and L. Ljung. Local modelling of nonlinear dynamic systems using direct weight optimization. Accepted for the 13th IFAC Symposium on System Identification, Rotterdam, Aug. 2003

and

> J. Roll, A. Nazin, and L. Ljung. Direct weight optimization for nonparametric estimation of a regression function at a given point. Submitted to Scandinavian Journal of Statistics, 2003

The case when bounds on the function value and the derivative are given is presented in

> J. Roll, A. Nazin, and L. Ljung. Local modelling with a priori known bounds using direct weight optimization. Submitted to the European Control Conference, Cambridge, Sept. 2003

Much of the material in Sections 11.1, 11.2, 11.4, and 11.6 in Part II is joint work with Dr. Alberto Bemporad. This material is also published in

> A. Bemporad, J. Roll, and L. Ljung. Identification of hybrid systems via mixed-integer programming. In *The 40th IEEE Conference on Decision and Control*, pages 786–792, Dec. 2001.

and

> J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. Provisionally accepted for Automatica, 2003

Some of the material in Chapter 12 has previously been published in

> J. Roll. Invariance of approximating automata for piecewise linear systems with uncertainties. In *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 396–406. Springer-Verlag, 2000.

Other parts of Chapter 12 also appear in

> J. Roll. Robust verification of piecewise affine systems. In *15th IFAC World Congress on Automatic Control, Session T-We-A21*, July 2002.

Part I

Local Modelling Using Direct Weight Optimization

2 Nonparametric Methods and Local Modelling

As mentioned in Section 1.2, the system identification problem is a problem of describing the relationship between input and output signals. This problem can of course be regarded as a kind of function approximation or a regression problem: Given a set of regression vectors $\varphi(k)$ and output signals $y(k)$, we can assume that the outputs are generated as noisy measurements of an unknown function

$$y(k) = f(\varphi(k)) + e(k)$$

and our goal is to recover the function $f(\varphi)$.

A very common approach in nonlinear system identification is to use some kind of *local models* (see e.g., [112]) and/or methods. A local model or method builds the function estimate or prediction from observations in a local neighborhood of the point of interest. Also most function expansion methods are of this character: A radial basis neural network is built up from basis functions with local support, and the standard sigmoidal (one hidden layer feed-forward) network is local around certain hyperplanes in the regressor space (see [94]).

If the model classes used are parameterized, this means that the identification problem is reduced to finding the parameters that makes the model match the observed data as well as possible. This is referred to as *parametric* methods. In contrast, *nonparametric* methods do not use a parameterized model class, but make pointwise estimates of e.g., the frequency response, the step response, or, as in this part of the thesis, the system function f. However, the boundary between the different categories is not always sharp.

In this part, a nonparametric *local modelling* approach to the regression problem is considered. Following the definition in [47], the local modelling approach can be described as follows: For any given point φ_0, for which we would like to estimate $f(\varphi_0)$, we should model f around φ_0 using only the data that are close to φ_0. There are two main points in this:

- The local modelling approach is a local method, and uses only data from the neighborhood of φ_0.
- For each new point in which f should be estimated, a "new model" is formed, in contrast to, e.g., the piecewise affine models in Part II, where each submodel has a certain validity region.

The local modelling approach and similar ideas have occurred in many contexts [18, 47, 61, 122, 141, 155] under names such as *Model on Demand* [146], *lazy learning* and *least commitment learning* [5, 6, 17]. A central question is how many points should be used when forming the estimate (also referred to as the *bandwidth* question). The answer to this mainly depends on three factors:

1. The number of data available (and the actual spreading of the regression vectors).
2. The smoothness of the function f.
3. The variance of the noise $e(k)$.

This problem has been studied extensively in the statistical literature, and there are several solutions based on asymptotic (in the number of observations) analysis.

This chapter gives an overview of the problem and some existing methods, mainly based on the presentations in [47, 146]. Other good overviews can be found in [5, 155]. The following chapters then present a *direct weight optimization (DWO)* approach, which is not based on asymptotic analysis.

2.1 Introduction and Problem Formulation

Throughout this part of the thesis, we will assume that we are given a set of input-output pairs $\{(\varphi(k), y(k))\}_{k=1}^N$, coming from the relation

$$y(k) = f(\varphi(k)) + e(k) \tag{2.1}$$

The function $f : \mathbb{R}^n \to \mathbb{R}$ (we will often consider the univariate case $n = 1$) is assumed to be unknown. The noise terms $e(k)$ are independent random variables with $E[e(k)] = 0$ and $E[e^2(k)] = \sigma^2$, and should be independent of the regression variables $\varphi(k)$.

One usually distinguishes between two ways of viewing the regression variables. If they are viewed as independent, identically distributed random variables with a certain probability density function p_φ, this is referred to as *random design*. If, on the other hand, $\varphi(k)$ are deterministic, we call this *fixed design*. A special case of the latter is the *equally spaced fixed design*, where the distance between two neighboring samples is constant (e.g., $\varphi(k) = k/N$ in the univariate case).

The problem we consider is the problem of estimating the value $f(\varphi_0)$ at a given point φ_0. In the sequel, it will be convenient to also use the notation

$$\widetilde{\varphi}(k) = \varphi(k) - \varphi_0 \tag{2.2}$$

As we saw in Section 1.2, many methods for estimating $f(\varphi_0)$ lead to a *linear estimator* in the form

$$\hat{f}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) \tag{2.3}$$

where $\hat{f}(\varphi_0)$ is our estimate of $f(\varphi_0)$. In words, the estimate $\hat{f}(\varphi_0)$ is a weighted sum of the observations $y(k)$. As it will turn out in Chapter 5, in some cases it will also be useful to consider an *affine estimator* given by

$$\hat{f}(\varphi_0) = w_0 + \sum_{k=1}^{N} w_k y(k) \tag{2.4}$$

In both these cases, the weights $w_0 \in \mathbb{R}$ and $w = (w_1 \ \ldots \ w_N)^T \in \mathbb{R}^N$ can be functions of φ_0 and the regression vectors $\varphi(k)$, $k = 1, \ldots, N$. In many methods, they also depend of the noise variance σ^2 and of some measure of the smoothness of f. However, it is worth noting that the weights in most methods do not depend on $y(k)$.

Assuming one of the forms (2.3) or (2.4), the problem then reduces to finding good weights w_k, which give reasonably small bias and variance of the estimate. In Section 2.4, different criteria are given for assessing the estimators. A popular criterion is the *mean squared error (MSE)*

$$\mathit{MSE}(\hat{f}(\varphi_0)) \triangleq E[(\hat{f}(\varphi_0) - f(\varphi_0))^2 | \varphi_1^N] \tag{2.5}$$

where $\varphi_1^N = \{\varphi(1), \ldots, \varphi(N)\}$. Sometimes when considering random design, the MSE is defined as the corresponding unconditional expectation, but in this thesis, only the definition (2.5) will be used. In the case of fixed design, the conditioning will of course have no effect. The *worst-case MSE* over a class $\mathcal{F}$ of functions

$$\mathit{WMSE}(\hat{f}(\varphi_0), \mathcal{F}) \triangleq \sup_{f \in \mathcal{F}} \mathit{MSE}(\hat{f}(\varphi_0)) \tag{2.6}$$

is another commonly used criterion. In this way, one can get a guaranteed upper bound on the MSE (assuming that all assumptions hold). In statistics, the worst-case MSE is also called *maximum MSE*. Using this criterion is often referred to as a *minimax* approach (see, e.g., [85, 88, 137, 138]). The worst-case MSE (or rather an upper bound on the worst-case MSE) will be used in the DWO approach presented in this thesis.

Depending on the method, different assumptions are made about f. In the following, some common function classes will be presented.

Class $\mathcal{F}_{p+1}(L)$

A common assumption is that f is p times differentiable. Another frequently used assumption is that the pth derivative is *Lipschitz continuous*. For $p = 1$, this means that there is a constant L (called the *Lipschitz constant*) such that

$$\|\nabla f(\varphi(1)) - \nabla f(\varphi(2))\| \leq L\|\varphi(1) - \varphi(2)\| \quad \forall\, \varphi(1), \varphi(2) \in \mathbb{R}^n \tag{2.7}$$

where $\|\cdot\|$ denotes the Euclidean norm. For more general cases, see Chapter 4. The class of p times differentiable functions with Lipschitz continuous pth derivatives, having a Lipschitz constant L, will be denoted by $\mathcal{F}_{p+1}(L)$. (With some abuse of notation, we will use this notation regardless of the value of n, i.e., n is supposed to be fixed and known.)

Hölder Class $\Sigma(\beta, L)$

A generalization of this class for univariate functions is the *Hölder* class [88]. The univariate function f is said to belong to the Hölder smoothness class $\Sigma(\beta, L)$ if f is p times differentiable, $\beta = p + \alpha$ with $0 < \alpha \leq 1$, and

$$|f^{(p)}(\varphi(1)) - f^{(p)}(\varphi(2))| \leq L|\varphi(1) - \varphi(2)|^{\alpha} \quad \forall\, \varphi(1), \varphi(2) \in \mathbb{R} \tag{2.8}$$

We can notice that for univariate functions and integers β, we have $\Sigma(\beta, L) = \mathcal{F}_{\beta}(L)$.

Class $\mathcal{G}_{p+1}(L)$

A related assumption is given in [85], where the univariate function f is supposed to take the following form:

$$f(\varphi) = a_0 + a_1\widetilde{\varphi} + \cdots + a_p\widetilde{\varphi}^p + c(\widetilde{\varphi})\widetilde{\varphi}^{p+1} \tag{2.9}$$

where $a_i \in \mathbb{R}$ and $\sup_{\widetilde{\varphi}} |c(\widetilde{\varphi})| \leq M$. Note that under this assumption, f does not need to be continuous (except at φ_0). Thus, for univariate functions, the function class $\mathcal{F}_{p+1}((p+1)!M)$ is a proper subclass of this class. However, the function class is connected to the point φ_0 of the function estimate, and thus is not so useful in practice, if one would like to estimate f in several points. We will denote the function class by $\mathcal{G}_{p+1}((p+1)!M)$.

Class $\mathcal{F}_{p+1}(L, \delta, \Delta)$

In some situations, one might have some prior knowledge of what would be a reasonable value for the function and/or its derivatives. This can be incorporated in the DWO approach by assuming that there are known constants a, δ, and Δ, and a vector b, such that

$$|f(\varphi_0) - a| \leq \delta \tag{2.10a}$$

$$\|\nabla f(\varphi_0) - b\| \leq \Delta \tag{2.10b}$$

The function classes satisfying (2.7) and (2.10) will be denoted by $\mathcal{F}_{p+1}(L, \delta, \Delta)$, and will be studied (for the case $p = 1$) in Chapter 5.

When the data come from a dynamic system, such that the regression vectors depend on old values of y, this means that $\varphi(i)$ and $e(j)$ will not be independent for all values of i, j anymore. However, we will neglect this fact for the time being, and discuss the implications further in Chapter 7. Note also that for NFIR models (see Section 1.1.2), such problems will not occur.

2.2 Kernel Estimators

A classic family of methods to decide the weights of (2.3) are the *kernel estimators*. Here, a *kernel function* K, which usually is a symmetric probability density function, is used to determine the weights. For simplicity, we will only treat the univariate case in this section. Some common choices of kernel functions are the *Gaussian* kernel

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \tag{2.11}$$

and the *Epanechnikov* kernel [45]

$$K(u) = \frac{3}{4} \max\{1 - u^2, 0\} \tag{2.12}$$

For more details about the kernel functions, see Section 2.5.

The width of the kernel is determined by introducing a bandwidth parameter h, and letting

$$K_h(\cdot) = \frac{1}{h} K(\cdot/h) \tag{2.13}$$

As an example of a kernel estimator, the *Nadaraya-Watson estimator* [113, 156] (see also [123] for a related estimator) is given by

$$\hat{f}^{NW}(\varphi_0) = \frac{\sum_{k=1}^{N} K_h(\widetilde{\varphi}(k)) y(k)}{\sum_{i=1}^{N} K_h(\widetilde{\varphi}(i))} \tag{2.14}$$

Comparing this expression with (2.3), we can see that the weights are given by

$$w_k = \frac{K_h(\widetilde{\varphi}(k))}{\sum_{i=1}^{N} K_h(\widetilde{\varphi}(i))} \tag{2.15}$$

Apparently, the denominator makes the estimate a weighted average of $y(k)$ by normalizing so that

$$\sum_{k=1}^{N} w_k = 1$$

Another kernel estimator is the *Gasser-Müller* estimator [53]. Assuming that the data $\varphi(k)$ are ordered in ascending order, the Gasser-Müller estimator is defined by

$$\begin{aligned}\hat{f}^{GM}(\varphi_0) &= \sum_{k=1}^{N} \int_{u_{k-1}}^{u_k} K_h(u-\varphi_0)du\, y(k) \\ u_0 = -\infty,\ u_N = \infty,\ u_k &= \frac{\varphi(k)+\varphi(k+1)}{2} \text{ for } k = 1,\dots,N-1\end{aligned} \tag{2.16}$$

One advantage with this estimator compared to the Nadaraya-Watson estimator is, as pointed out in [47], that no normalizing denominator is needed, since

$$\sum_{k=1}^{N} \int_{u_{k-1}}^{u_k} K_h(u-\varphi_0)du = \int_{-\infty}^{\infty} K_h(u-\varphi_0)du = 1$$

if K_h is a probability density function.

For both these methods, the choice of bandwidth is a bias/variance trade-off problem, which will be discussed more in detail in Section 2.4. The asymptotic (for $N \to \infty$) bias and variance for the methods are given in Table 2.1, taken from [47]. Here a random design is assumed, where the samples $\varphi(k)$ are taken from a distribution with a differentiable probability density function p_φ. Furthermore, it is assumed that the true function f is two times differentiable. For notational simplicity, the definitions

$$\Psi_k(K) = \int u^k K(u)du \tag{2.17}$$

$$r(K) = \int K^2(u)du \tag{2.18}$$

are used (it is assumed that these integrals exist and are finite). These definitions will also be needed in later sections.

Table 2.1: Asymptotic bias and variance for the Nadaraya-Watson and Gasser-Müller estimators.

Estimator	Bias	Variance
Nadaraya-Watson	$\frac{1}{2}\left(f''(\varphi_0) + \frac{2f'(\varphi_0)p'_\varphi(\varphi_0)}{p_\varphi(\varphi_0)}\right) h^2\Psi_2(K)$	$\frac{\sigma^2}{p_\varphi(\varphi_0)Nh} r(K)$
Gasser-Müller	$\frac{1}{2}f''(\varphi_0)h^2\Psi_2(K)$	$\frac{3}{2}\frac{\sigma^2}{p_\varphi(\varphi_0)Nh} r(K)$

The expressions in Table 2.1 also assume that φ_0 is an *interior point* [47], which means that the support of the kernel function (asymptotically) is within the support of p_φ. Points that are not interior points are called *boundary points.* Unfortunately, the kernel estimators are not very flexible, and cannot automatically adapt to cases when data are asymmetrically distributed around φ_0, e.g., when φ_0 is a boundary point. This often leads to a large increase in the bias, and is referred to as *boundary effects.* The phenomenon can be understood intuitively by considering the problem of estimating an increasing function in a point φ_0 such that $\varphi_0 > \varphi(k)$ for all $k = 1, \dots, N$. Using the Nadaraya-Watson or Gasser-Müller estimator with a nonnegative kernel function, the estimate $\hat{f}(\varphi_0)$ will be a weighted average of the observations $y(k) = f(\varphi(k)) + e(k)$, and never gets larger than $\max_k y(k)$, although it probably should.

To get around this problem, many methods have been proposed. For instance, [53, 54] consider special *boundary kernels* to reduce the bias.

2.3 Local Polynomial Modelling

A slightly more sophisticated alternative to the kernel estimators is the *local polynomial modelling* approach (see, e.g., [32, 47, 61, 147, 155]). In this approach, the estimator is determined by locally fitting a polynomial to the given data via minimization of a weighted least-squares problem, which in the univariate case takes the form

$$\hat{\beta} = \arg\min_{\beta} \sum_{k=1}^{N} K_h(\widetilde{\varphi}(k)) \left(y(k) - \sum_{j=0}^{p} \beta_j \widetilde{\varphi}^j(k) \right)^2 \tag{2.19}$$

where K_h is given by (2.13). The resulting estimator is obtained as $\hat{f}^{LP}(\varphi_0) = \hat{\beta}_0$.

When $p = 0$, it turns out that $\hat{f}^{LP}(\varphi_0)$ will be the Nadaraya-Watson estimator. The Gasser-Müller estimator is instead obtained if $K_h(\widetilde{\varphi}(k))$ in (2.19) is replaced by $\int_{u_{i-1}}^{u_i} K_h(u - \varphi_0) du$. Since we can obtain these estimators by locally fitting a constant term to the observed data (i.e., by using (2.19) with $p = 0$), we can refer to them as *local constant approximators* [47].

If we introduce

$$\Phi = \begin{pmatrix} 1 & \widetilde{\varphi}(1) & \cdots & \widetilde{\varphi}^p(1) \\ \vdots & \vdots & & \vdots \\ 1 & \widetilde{\varphi}(N) & \cdots & \widetilde{\varphi}^p(N) \end{pmatrix} \tag{2.20}$$

$$\bar{K}_h = \begin{pmatrix} K_h(\widetilde{\varphi}(1)) & 0 & \cdots & 0 \\ 0 & K_h(\widetilde{\varphi}(2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_h(\widetilde{\varphi}(N)) \end{pmatrix} \tag{2.21}$$

and

$$Y = \begin{pmatrix} y(1) \\ \vdots \\ y(N) \end{pmatrix}, \qquad \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix} \tag{2.22}$$

we can rewrite the problem (2.19) as

$$\min_{\beta} (Y - \Phi\beta)^T \bar{K}_h (Y - \Phi\beta) \tag{2.23}$$

with the solution

$$\hat{\beta} = (\Phi^T \bar{K}_h \Phi)^{-1} \Phi^T \bar{K}_h Y \tag{2.24}$$

The function estimate $\hat{f}^{LP}(\varphi_0)$ is given by

$$\hat{f}^{LP}(\varphi_0) = \hat{\beta}_0 = \mathbf{e}_1^T(\Phi^T \bar{K}_h \Phi)^{-1}\Phi^T \bar{K}_h Y \tag{2.25}$$

Comparing once again to (2.3), this will correspond to

$$w = \bar{K}_h \Phi(\Phi^T \bar{K}_h \Phi)^{-1}\mathbf{e}_1 \tag{2.26}$$

These weights are often referred to as *equivalent weights* (see, e.g., [47]), to separate them from the weights $K_h(\widetilde{\varphi}(k))$ of the weighted least-squares problem (2.19). Note also that, for $j \leq p$, we have

$$\sum_{k=1}^{N} w_k \widetilde{\varphi}^j(k) = \mathbf{e}_j^T \Phi^T w = \mathbf{e}_j^T \Phi^T \bar{K}_h \Phi(\Phi^T \bar{K}_h \Phi)^{-1}\mathbf{e}_1 = \mathbf{e}_j^T \mathbf{e}_1 \tag{2.27}$$

This is a desirable property, and will be shared by the DWO estimator in Chapters 3 and 4. For instance, as we will see in Chapter 3, they imply that the worst-case MSE over $\mathcal{F}_{p+1}(L)$ is finite.

The case $p = 1$ may be worth a closer look. In this case, the estimator is called a *local linear estimator*, and can be expressed explicitly as

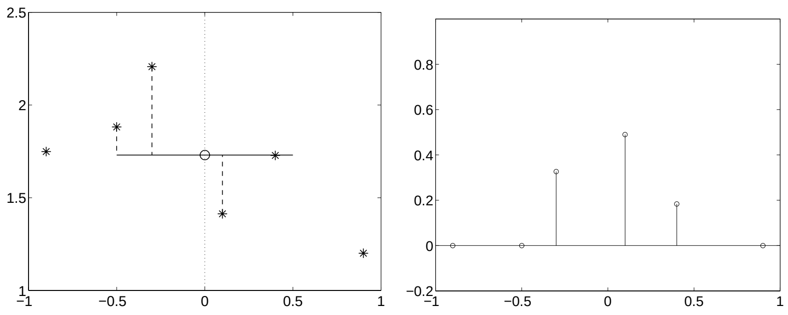
$$\begin{aligned} \hat{f}^{LP}(\varphi_0) &= \frac{\sum_{k=1}^{N} \alpha_k y(k)}{\sum_{k=1}^{N} \alpha_k}, \\ \alpha_k &= K_h(\widetilde{\varphi}(k)) \left(\sum_{i=1}^{N} K_h(\widetilde{\varphi}(i))\widetilde{\varphi}^2(i) - \widetilde{\varphi}(k) \sum_{i=1}^{N} K_h(\widetilde{\varphi}(i))\widetilde{\varphi}(i) \right) \end{aligned} \tag{2.28}$$

For the local linear estimator, (2.27) implies that

$$\sum_{k=1}^{N} w_k = 1, \qquad \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 0 \tag{2.29}$$

Another interesting thing to note is that if K_h is an even function, and the data samples are lying symmetrically, i.e., if the nonzero $\widetilde{\varphi}(k)$ can be paired so that for each pair $(\widetilde{\varphi}(i), \widetilde{\varphi}(j))$ we have $\widetilde{\varphi}(i) = -\widetilde{\varphi}(j)$, then the local linear estimator (2.28) and the Nadaraya-Watson estimator will coincide.

The local polynomial models (for $p \geq 1$) are better than the kernel estimators at adapting automatically to data lying asymmetrically, e.g., at boundaries [47]. However, since the number of data samples taken into account is determined only by the choice of bandwidth, there might (as pointed out by [34]) still be problems if the latter choice is only asymptotically based and does not take the actual values of $\varphi(k)$ into account. Furthermore, the asymptotically optimal kernel functions are different depending on if φ_0 is an interior point or a boundary point.



(a) Function estimate (○). (b) Equivalent weights.

Figure 2.1: The Nadaraya-Watson estimate from Example 2.1.

Example 2.1 (The Nadaraya-Watson and local linear estimators) *Consider again the data set in Example 1.4. The values of $\varphi(t)$ are listed in Table 2.2.*

Table 2.2: The values of $\varphi(t)$ for the data set in Example 1.4.

t	$\varphi(t)$
1	-0.9
2	-0.5
3	-0.3
4	0.1
5	0.4
6	0.9

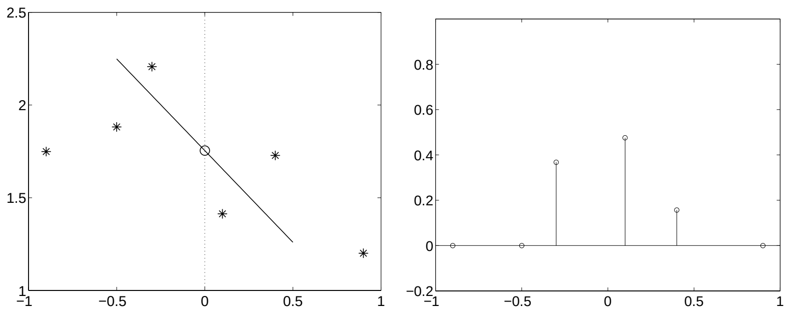
Figure 2.1 shows the Nadaraya-Watson estimate of $f(0)$ and the equivalent weights, using the Epanechnikov kernel with bandwidth $h = 0.5$. In Figure 2.2 the local linear estimate (using the same kernel) is shown together with its equivalent weights.

An advantage with the local polynomial approach is that we can get estimates not only of the function value $f(\varphi_0)$, but also of the derivatives $f^{(j)}(\varphi_0)$. Since f is locally approximated by a polynomial with the coefficients $\hat{\beta}_j$, a natural estimate of the jth derivative (where $j \leq p$) is

$$\widehat{f^{(j)}}(\varphi_0) = j!\hat{\beta}_j = j!\mathbf{e}_j^T(\Phi^T\bar{K}_h\Phi)^{-1}\Phi^T\bar{K}_hY \tag{2.30}$$

The equivalent weights for this estimator become

$$w^{(j)} = j!\bar{K}_h\Phi(\Phi^T\bar{K}_h\Phi)^{-1}\mathbf{e}_j \tag{2.31}$$



(a) Function estimate (○). (b) Equivalent weights.

Figure 2.2: The local linear estimate from Example 2.1.

Estimates of the derivative using the DWO approach will be studied in Chapter 6.

So far in this section, only univariate functions f have been considered. However, the extension to the multivariate case is immediate for the local polynomial modelling methods (including the Nadaraya-Watson estimator), given multivariate kernel functions. Such functions will be described in Section 2.5.

2.4 Different Performance Criteria

To be able to select the weights of an estimator (which for the local polynomial estimators corresponds to determining the kernel function and the bandwidth), one needs a *criterion function* to measure the quality of the estimates. The criterion function gives a value which is somehow related to the performance of the estimator, and the task then becomes that of finding the weights that give the optimal performance value. The criterion can be global, which could be useful if we would like, e.g., a common bandwidth for the entire regressor space, or local (pointwise), which, e.g., allows us to select a bandwidth for each estimation point. In this section, some of the most common criteria will be presented.

2.4.1 The MSE and MISE Criteria

A commonly used pointwise criterion is the *mean squared error (MSE)*, which is defined as

$$MSE(\hat{f}(\varphi_0)) \triangleq E[(\hat{f}(\varphi_0) - f(\varphi_0))^2|\varphi_1^N] \tag{2.32}$$

A nice property of the MSE is that it can be decomposed into a squared bias part and a variance part

$$MSE(\hat{f}(\varphi_0)) = \underbrace{\Big(E[\hat{f}(\varphi_0)|\varphi_1^N] - f(\varphi_0)\Big)^2}_{\text{Bias}^2} + \underbrace{E[\Big(\hat{f}(\varphi_0) - E[\hat{f}(\varphi_0)|\varphi_1^N]\Big)^2 |\varphi_1^N]}_{\text{Variance}} \tag{2.33}$$

For a linear estimator in the form (2.3), the bias can be written as

$$B(w) = \sum_{k=1}^{N} w_k f(\varphi(k)) - f(\varphi_0) \tag{2.34}$$

The variance becomes

$$v(w) = \sigma^2 \sum_{k=1}^{N} w_k^2 \tag{2.35}$$

A global criterion can be obtained by integrating the MSE over φ_0. This criterion is called the *mean integrated squared error (MISE)*:

$$MISE(\hat{f}) \triangleq E[\int (\hat{f}(\varphi) - f(\varphi))^2 d\varphi|\varphi_1^N] = \int MSE(\hat{f}(\varphi))d\varphi \tag{2.36}$$

Often when using random design, the following alternative definition of the MISE is used [155]:

$$MISE_p(\hat{f}) \triangleq \int MSE(\hat{f}(\varphi))p_\varphi(\varphi)d\varphi \tag{2.37}$$

i.e., the integral is weighted by the probability density function p_φ of φ. According to the definition in [47], an arbitrary nonnegative weight function can be used instead of p_φ.

When using a local polynomial modelling method to calculate the weights, a drawback with the MSE is that it depends on the bandwidth h in a complicated way. One way of getting around this is to consider the case when N is large, which leads to the *asymptotic mean squared error (AMSE)*. To derive this criterion, it is assumed that the data samples are equally spaced on a bounded interval (similar expressions can also be derived for random design with a sufficiently smooth probability density function p_φ; see, e.g., [155]). For a univariate local linear estimator, we get the following expression for the bias

$$B(K, h) = \frac{1}{2}h^2 f''(\varphi_0)\Psi_2(K) + o(h^2) + O(\frac{1}{N}) \tag{2.38}$$

and the variance

$$v(K, h) = \frac{1}{Nh}r(K)\sigma^2 + o(\frac{1}{Nh}) \tag{2.39}$$

Taking only the first parts of these expressions, we can form the AMSE

$$AMSE(\hat{f}(\varphi_0)) = \left(\frac{1}{2}h^2 f''(\varphi_0)\Psi_2(K)\right)^2 + \frac{1}{Nh}r(K)\sigma^2 \tag{2.40}$$

which for large Nh and small h will be a good approximation of the MSE (more precisely, as $h \to 0$, $Nh \to \infty$, the ratio between the AMSE and the MSE will approach 1). This expression can now be minimized with respect to h to get the asymptotically optimal local bandwidth. Since, as we can see, the bias part increases and the variance part decreases with h, this is a classic bias/variance tradeoff. The minimizing h is given by

$$h_{AMSE} = \left(\frac{r(K)\sigma^2}{(f''(\varphi_0))^2 \Psi_2^2(K)} \right)^{\frac{1}{5}} N^{-\frac{1}{5}} \tag{2.41}$$

If we instead are interested in an asymptotically optimal global bandwidth, we can do the same procedure with the MISE. The *asymptotic mean integrated squared error (AMISE)* becomes:

$$AMISE(\hat{f}) = \frac{1}{4} h^4 r(f'') \Psi_2^2(K) + \frac{1}{Nh} r(K)\sigma^2 \tag{2.42}$$

and we get the asymptotically optimal global bandwidth

$$h_{AMISE} = \left(\frac{r(K)\sigma^2}{r(f'') \Psi_2^2(K)} \right)^{\frac{1}{5}} N^{-\frac{1}{5}} \tag{2.43}$$

The multivariate AMSE and AMISE for local linear estimators are defined analogously. For details about this, see [136, 146].

2.4.2 The Mean Squared Prediction Error and Risk Function

Another global criterion is the *mean squared prediction error*

$$P(\hat{f}) = \frac{1}{N} \sum_{k=1}^{N} \Big(f(\varphi(k)) - \hat{f}(\varphi(k)) \Big)^2 \tag{2.44}$$

This is closely related to the *risk function*

$$R(\hat{f}) = \frac{1}{N\sigma^2} \sum_{k=1}^{N} E[\Big(f(\varphi(k)) - \hat{f}(\varphi(k)) \Big)^2 |\varphi_1^N] = \frac{1}{\sigma^2} E[P(\hat{f})|\varphi_1^N] \tag{2.45}$$

A common problem with the criteria mentioned so far is that they all depend on the unknown true function f. To handle this, many different alternative criteria have appeared that approximate one of the above criteria. The methods of finding the approximations can be divided into two classes:

- *Classical methods*, for example cross-validation, generalized cross-validation, Akaike's criteria, and Mallows' C_p criterion, are extensions of methods used in parametric system identification.
- In *direct plug-in methods*, the unknown variables of the AMSE and AMISE formulas are estimated from data, and the estimates are "plugged in" into the expressions (2.41) or (2.43) for finding an asymptotically optimal bandwidth.

2.4.3 Classical Methods

Most classical methods use the mean squared prediction error function $P(\hat{f})$ or the risk function $R(\hat{f})$ as a starting point and try to approximate these using only known quantities. A very simple approximation of $P(\hat{f})$ is the *resubstitution estimate* of the prediction error, obtained by replacing the true function values $f(\varphi(k))$ in $P(\hat{f})$ by the measured observations $y(k)$:

$$\hat{P}(\hat{f}) = \frac{1}{N} \sum_{k=1}^{N} \Big(y(k) - \hat{f}(\varphi(k)) \Big)^2 \tag{2.46}$$

Unfortunately, this is a biased estimate of $P(\hat{f})$, and for local polynomial modelling methods this function is an increasing function of h. The reason for this is that we try to estimate the observations $y(k)$ using the observations themselves (recall that $\hat{f}$ is a weighted sum of $y(k)$, according to (2.3)), and so we get a perfect fit if the estimate of $y(k)$ is chosen to be $y(k)$ itself (i.e., $w_k = 1$, while $w_i = 0$ for $i \neq k$). This can be thought of as an overfitting phenomenon.

To overcome this problem, one can use a cross-validation strategy, or one can multiply $\hat{P}(\hat{f})$ by a *penalizing function* to better mimic $P(\hat{f})$.

Before we go into the different procedures for doing this, however, we will need some additional notation, which is given in [146] for local polynomial modelling methods. Let $w(\varphi)$ denote the vector of weights for estimating f in the point φ. Now we can define the so-called *hat matrix* or smoothing matrix

$$H \triangleq \begin{pmatrix} w^T(\varphi(1)) \\ \vdots \\ w^T(\varphi(N)) \end{pmatrix} \tag{2.47}$$

This matrix is interesting in several aspects. Note that

$$\begin{pmatrix} \hat{f}(\varphi(1)) \\ \vdots \\ \hat{f}(\varphi(N)) \end{pmatrix} = HY \tag{2.48}$$

with Y defined by (2.22). Furthermore, as explained in [146], $\operatorname{tr}(H)$ can be interpreted as a *degrees-of-freedom* measure. To see this intuitively, consider the most localized possible estimator, where only $w_k(\varphi(k)) = 1$ and the rest of the weights are zero. For this estimator with many degrees of freedom, $\operatorname{tr}(H) = N$. On the other hand, when fitting a global constant model to the data (one degree of freedom), so that all weights $w_i(\varphi(k)) = 1/N$, we get $\operatorname{tr}(H) = 1$.

A quantity related to the hat matrix is the *influence function* [96], which is defined as

$$\operatorname{infl}(\varphi_0) = \mathbf{e}_1^T (\Phi^T \bar{K}_h \Phi)^{-1} \mathbf{e}_1 K_h(0) \tag{2.49}$$

where Φ was defined in (2.20). Note that the diagonal elements of H are given by $w_k(\varphi(k)) = \operatorname{infl}(\varphi(k))$.

Cross-Validation

A simple way to get around the overfitting problem of the estimator (2.46) is to use *one leave-out cross-validation* [61], where the cross-validation function

$$CV(\hat{f}) = \frac{1}{N}\sum_{k=1}^{N}\Big(y(k) - \hat{f}_{-k}(\varphi(k))\Big)^2 \tag{2.50}$$

is used. Here $\hat{f}_{-k}(\varphi(k))$ stands for the estimate obtained when leaving out the kth observation.

For local polynomial modelling, one can compute $CV(\hat{f})$ in a more efficient way:

$$CV(\hat{f}) = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{y(k) - \hat{f}(\varphi(k))}{1 - \mathrm{infl}(\varphi(k))}\right)^2 \tag{2.51}$$

Generalized Cross-Validation

The *generalized cross-validation* criterion is a simplified version of the one leave-out cross-validation, where the values $\mathrm{infl}(\varphi(k))$ in (2.51) are replaced by their mean value $\mathrm{tr}(H)/N$. The resulting criterion can also be interpreted as a penalizing approach, namely

$$GCV(\hat{f}) = \hat{P}(\hat{f})\frac{1}{\left(1 - \frac{tr(H)}{N}\right)^2} \tag{2.52}$$

Akaike's Criteria

Other penalizing approaches can be obtained by modifying Akaike's criteria for parametric methods [1]. These criteria contain $\dim\theta$ as a measure of the degrees of freedom. For nonparametric methods, this quantity could be replaced by $\mathrm{tr}(H)$ (see [146]). Akaike's information criterion then becomes[*]

$$AIC(\hat{f}) = \hat{P}(\hat{f})e^{2\frac{\mathrm{tr}(H)}{N}} \tag{2.53}$$

It has turned out that the AIC criterion often results in a rather small penalty. An enhanced variant is the *corrected* AIC [68]

$$AIC_C(\hat{f}) = \hat{P}(\hat{f})e^{1+\frac{2(\mathrm{tr}(H)+1)}{N-tr(H)-2}} \tag{2.54}$$

Yet another related criterion is Akaike's *Final Prediction Error*

$$FPE(\hat{f}) = \hat{P}(\hat{f})\frac{1+\frac{\mathrm{tr}(H)}{N}}{1-\frac{\mathrm{tr}(H)}{N}} \tag{2.55}$$

[*]In its original form, Akaike's criterion actually corresponds to the logarithm of (2.53).

Mallows' C_p Criterion

Mallows' C_p criterion [102] for parametric methods is an unbiased criterion of the risk function (2.45), which was extended to local regression by Cleveland and Devlin [33]. The latter version can be written

$$CP(\hat{f}) = \frac{1}{\sigma^2}\hat{P}(\hat{f}) + 2\frac{\operatorname{tr}(H)}{N} - 1 \tag{2.56}$$

Localized Versions

All criteria mentioned in this subsection are global. However, they can be *localized* (or made *adaptive*) by weighting the sums, so that the influence from data lying close to φ_0 becomes more important. For local polynomial methods, the weights used for this are those given by the kernel, i.e., the same weights as in the weighted least-squares problem (2.19). Instead of using $\hat{f}(\varphi(k))$, e.g., as in (2.50), the local polynomial model estimated in φ_0 is used (i.e., the polynomial with the coefficient $\hat{\beta}$ from (2.19)). Denoting this polynomial by $\bar{f}^{\varphi_0}(\varphi)$, we can for instance write the localized cross-validation as

$$LCV(\hat{f}(\varphi_0)) \triangleq \frac{\sum_{k=1}^{N} K_h(\widetilde{\varphi}(k))\left(y(k) - \bar{f}^{\varphi_0}_{-k}(\varphi(k))\right)^2}{\sum_{k=1}^{N} K_h(\widetilde{\varphi}(k))} \tag{2.57}$$

Analogously to the global case, $\bar{f}^{\varphi_0}_{-k}$ means the estimate we get when leaving out the kth observation. For details about localization of the other criteria, see [146].

2.4.4 Direct Plug-In Methods

The main alternative to the classical methods occurring in the literature is the class of *direct plug-in methods* [47, 155]. These methods use the formulas (2.41) and (2.43) for the asymptotically optimal bandwidths, and “plug in” estimates of the unknown quantities, e.g., $r(f'')$ and σ^2 for h_{AMISE}. To get the estimates, a preliminary model (e.g., a global, higher-order model) is used to obtain so-called *pilot estimates*, for instance of $f(\varphi(k))$ and $f''(\varphi(k))$. Then $r(f'')$ can be estimated by

$$\hat{r}(f'') = \frac{1}{N}\sum_{k=1}^{N}\left(\hat{\hat{f}}''(\varphi(k))\right)^2 \tag{2.58}$$

and σ^2 by

$$\hat{\sigma}^2 = \frac{1}{N - 2\operatorname{tr}(H) + \operatorname{tr}(H^T H)}\sum_{k=1}^{N}\left(y(k) - \hat{\hat{f}}(\varphi(k))\right)^2 \tag{2.59}$$

where $\hat{\hat{f}}''(\varphi(k))$ and $\hat{\hat{f}}(\varphi(k))$ are pilot estimates.

Plug-in methods have been popular during the last decade, and they have by some been regarded as superior to classical methods. However, Loader [95, 97] challenges this, and points out that they very much depend on the choice of pilot bandwidth. If this is not appropriately chosen, something which may be hard to do *ad hoc*, the final selected bandwidth may also be bad.

2.4.5 The Worst-Case MSE Criterion

An alternative to approximating a noncomputable criterion is to find a computable upper bound. This approach has the advantage of having a certain degree of built-in robustness, in the sense that we can give a guarantee that the fit will not be worse than a certain limit (if all assumptions about the function, noise etc., are correct). This limit value is then what is optimized.

As mentioned previously, a common pointwise criterion is the *worst-case MSE* which is the supremum of the MSE over a given function class $\mathcal{F}$, i.e.,

$$\mathit{WMSE}(\hat{f}(\varphi_0), \mathcal{F}) = \sup_{f \in \mathcal{F}} E[(\hat{f}(\varphi_0) - f(\varphi_0))^2 | \varphi_1^N] \tag{2.60}$$

This criterion will often be used in this thesis. Kernel estimators that minimize the worst-case MSE over a function class are sometimes called *minimax* estimators [85, 88, 137, 138].

An interesting quantity in this context is the *linear minimax risk* (see [47]), defined by

$$\mathcal{R}(\mathcal{F}) = \inf_{\hat{f}(\varphi_0)} \sup_{f \in \mathcal{F}} E[(\hat{f}(\varphi_0) - f(\varphi_0))^2 | \varphi_1^N] \tag{2.61}$$

where the infimum is taken over all linear estimators. This can be interpreted as the best possible performance (in terms of worst-case MSE) one can get using a linear estimator.

2.5 Kernel Functions

The performance of the kernel estimators and local polynomial estimators naturally depends on what kernel function is used. As mentioned in Section 2.2, the kernel function is normally chosen as a symmetric probability density function, which means that it satisfies the conditions

$$\int K(u)du = 1, \qquad \int uK(u)du = 0 \tag{2.62}$$

It also means that $K(u) \geq 0$ for all u. However, sometimes it is useful to consider kernel functions that also take negative values. The kernel functions known as *high-order kernels* are examples of such kernels.

A popular family of kernels is the "symmetric Beta family" [47], defined by

$$\frac{1}{\mathrm{Beta}(\frac{1}{2}, \gamma + 1)} \max\{1 - u^2, 0\}^\gamma, \qquad \gamma \in \mathbb{N} \tag{2.63}$$

Here, the Beta function just acts as a normalizing factor, making the kernel satisfy (2.62). When $\gamma = 0$, we get the *uniform* kernel. The choice $\gamma = 1$ leads to the Epanechnikov kernel (2.12). The kernels obtained for $\gamma = 2$ and $\gamma = 3$ are called *biweight* and *triweight* kernels, respectively [47]. A common property for the "symmetric Beta family" is that the kernels have finite support, which is desirable from a computational point of view [34].

In some applications where kernels that smoothly descend to zero are desirable (e.g., in signal processing and spectral analysis), a common kernel is the *tricube* kernel

$$K(u) = \frac{70}{81} \max\{1 - |u|^3, 0\}^3 \tag{2.64}$$

This kernel is the default choice in the robust Lowess and Loess estimators [32, 33], and Locfit [96] estimators. Also other kernels have been proposed; see, e.g., [74].

When estimating multivariate functions, one also needs multivariate kernel functions. A common method is to construct a multivariate kernel function $K(u)$ from a univariate function $K^1(u)$, either via a *product* construction

$$K(u) = \prod_{i=1}^{n} K^1(u_i) \tag{2.65}$$

or using a *radial* construction

$$K(u) = C_K K^1(\|u\|) \tag{2.66}$$

where C_K is a normalizing constant. Common choices of multivariate kernels are the Gaussian kernel, which can be regarded as being constructed by either of the product or radial constructions, and the *spherical Epanechnikov kernel* [47]

$$K(u) = C_K \max\{1 - \|u\|^2, 0\} \tag{2.67}$$

which is a radial construction.

To specify the width of the kernel a *bandwidth matrix* $\bar{H}$ is introduced, and the kernel

$$K_{\bar{H}}(u) = \det(\bar{H})^{-1} K(\bar{H}^{-1}u) \tag{2.68}$$

is used when calculating the weights. To avoid having too many design parameters, one often does not let $\bar{H}$ be fully parameterized, but uses a diagonal matrix or, even simpler, $\bar{H} = hI$, where I is the identity matrix.

2.5.1 Optimal Kernels

In the statistical literature, there exist many optimality results about kernels. A number of them consider the asymptotic mean squared error.

For kernel estimators, Legostaeva and Shiryaev [85] consider the class of functions $\mathcal{G}_{p+1}(L)$ described by (2.9). The kernel estimator they consider is a continuous version of (2.3), namely

$$\hat{f}(\varphi_0) = \int_{-\infty}^{\infty} K_h(\widetilde{\varphi}) y(\varphi) d\varphi$$

where $y(\varphi) = f(\varphi) + \eta(\varphi)$, and $\eta(\varphi)$ is a "white noise" process with $E[\eta(\varphi)] = 0$, $E[\eta(\phi)\eta(\varphi)] = \sigma^2\delta(\phi - \varphi)$. With some additional assumptions, one may interpret this as an asymptotic approximation to the fixed, equally spaced design. For this setup, a necessary and a sufficient condition for a weight function to minimize the worst-case MSE (2.6) is given. It turns out that the asymptotically optimal kernel functions have the form

$$K(u) = \operatorname{sgn} P_p(u) \max\{|P_p(u)| - \psi_0|t|^{p+1}, 0\} \tag{2.69}$$

where $P_p(u)$ is a pth order polynomial, whose coefficients together with $\psi_0 > 0$ should be determined from

$$\int_{-\infty}^{\infty} K(u)du = 1, \qquad \int_{-\infty}^{\infty} K^2(u)du = q, \qquad \int_{-\infty}^{\infty} u^k K(u)du = 0 \tag{2.70}$$

where q is a positive constant and $k = 1, \dots, p$. In particular, for $p = 1$, the optimal kernel is the Epanechnikov kernel. The minimax problem for $\mathcal{G}_{p+1}(L)$ has also been considered, e.g., in [137].

Kernel estimators that minimize the asymptotic worst-case MSE for the Hölder class $\Sigma(\beta, L)$ defined by (2.8) are given in [88], where a similar kernel estimator as in [85] is considered. For the special case $\beta = 2$ (i.e., for $\mathcal{F}_2(L)$), which was also considered in [162], the kernel function is a symmetric quadratic spline, having an infinite number of knots (breakpoints) in a finite interval. Furthermore, its local extrema have alternating signs and form a geometric series. Originally, this function was given as a solution of a related problem in [50].

In [54], asymptotic minimum variance kernels and asymptotically optimal kernels in the AMISE sense are given for estimation of a p times differentiable function and its derivatives. Also here, the Epanechnikov kernel turns out to be optimal in a special case.

For local polynomial modelling with random design, the Epanechnikov kernel has been shown to be optimal among the nonnegative kernels in the AMSE and AMISE sense [47]. In [47], it is also shown that the local linear estimator asymptotically achieves the linear minimax risk for a class of functions satisfying (2.9), if the Epanechnikov kernel is used with a bandwidth given by

$$h = \left(\frac{15\sigma^2}{p_\varphi(\varphi_0)L^2N} \right)^{1/5} \tag{2.71}$$

where p_φ is the probability density of the observations $\varphi(k)$ (cf. (2.41)).

2.6 Gaussian Processes

A slightly different approach to local modelling is the modelling by *Gaussian processes* (see, e.g., [56, 125]). An excellent introduction to the subject can be found in [101]. A Gaussian process can informally be described as a Gaussian probability distribution on a space $\mathcal{F}$ of functions $f(\varphi)$. More formally, a Gaussian process is a collection of random variables $\{f(\varphi)\}_{\varphi\in\Phi}$, indexed by a set Φ, such that for any finite subset $\{\varphi(1),\dots,\varphi(N)\}\subseteq\Phi$, the corresponding set $\{f(\varphi(1)),\dots,f(\varphi(N))\}$ has a joint multivariate Gaussian distribution.

A Gaussian process is fully specified by its *mean function*

$$\mu(\varphi)=E[f(\varphi)]$$

and *covariance function*

$$C(\varphi(i),\varphi(j))=E[(f(\varphi(i))-\mu(\varphi(i)))(f(\varphi(j))-\mu(\varphi(j)))]$$

Often $\mu(\varphi)$ is assumed to be identically equal to zero.

The system model used is the general model (2.1), where the noise terms $e(t)$ are assumed to be independent, identically distributed Gaussian variables with variance σ^2. This implies that $\{y(t)\}_{t=1}^{N+1}$ are jointly Gaussian, with zero mean (if $\mu(\varphi)\equiv 0$) and a covariance matrix

$$C_{N+1}=Q+\sigma^2 I$$

where

$$Q=\begin{pmatrix} C(\varphi(1),\varphi(1)) & C(\varphi(1),\varphi(2)) & \cdots & C(\varphi(1),\varphi(N+1)) \\ C(\varphi(2),\varphi(1)) & C(\varphi(2),\varphi(2)) & \cdots & C(\varphi(2),\varphi(N+1)) \\ \vdots & \vdots & \ddots & \vdots \\ C(\varphi(N+1),\varphi(1)) & C(\varphi(N+1),\varphi(2)) & \cdots & C(\varphi(N+1),\varphi(N+1)) \end{pmatrix}$$

For calculation of the one-step-ahead prediction, we will need to partition the covariance matrix C_{N+1} as follows

$$C_{N+1}=\begin{pmatrix} C_N & k_N \\ k_N^T & \kappa \end{pmatrix}$$

where $C_N\in\mathbb{R}^{N\times N}$, $k_N\in\mathbb{R}^N$, and $\kappa\in\mathbb{R}$. Also introduce

$$Y=\begin{pmatrix} y(1) \\ \vdots \\ y(N) \end{pmatrix}$$

Given a Gaussian process with $\mu(\varphi)\equiv 0$ and $C(\varphi(i),\varphi(j))$ given, the prediction of $y(N+1)$ can now be performed using

$$P(y(N+1)|y_1^N)=\frac{P(y_1^{N+1})}{P(y_1^N)}$$

This conditional probability distribution is Gaussian, with mean and variance given by

$$\hat{y}(N+1)\triangleq E[y(N+1)|y_1^N]=k_N^T C_N^{-1} Y$$
$$E[(y(N+1)-\hat{y}(N+1))^2|y_1^N]=\kappa-k_N^T C_N^{-1} k_N$$

Notice that the prediction $\hat{y}(N+1)$ is a linear function of the previous outputs, i.e., it can be written in the form (2.3). In general, however, the conditions (2.29) are not satisfied, which (as we will see in Chapter 3) means that the worst-case MSE over $\mathcal{F}_2(L)$ is infinite. Another thing to note is that neither C_N nor Y depend on $\varphi(N+1)$, so $\hat{y}(N+1)$ can be written in the form

$$\hat{y}(N+1) = \sum_{t=1}^{N} v_t C(\varphi(t), \varphi(N+1))$$

where

$$v = \begin{pmatrix} v_1 \\ \vdots \\ v_N \end{pmatrix} = C_N^{-1} Y$$

In words, the prediction $\hat{y}(N+1)$ as a function of $\varphi(N+1)$ is a weighted sum of the covariance function between $\varphi(N+1)$ and all the previous regression vectors $\varphi(t)$. Hence, the basic shape of the predicted function is completely determined by the shape of the covariance function $C(\varphi(i), \varphi(j))$. Therefore, an important problem should be to find the "true" Gaussian process, i.e., the "true" covariance function.

Finding the form of the covariance is a problem related to finding the proper model structure in parametric system identification. Assuming a certain structure, a common approach is to use a parameterized covariance function, and estimate the parameters θ by maximizing $P(\theta|Z_1^N)$ (*Evidence maximization*, see [100]). This is in general a nonconvex optimization problem. Another approach is to compute

$$P(y(N+1)|\varphi(N+1), Z_1^N) = \int P(y(N+1)|\varphi(N+1), Z_1^N, \theta) P(\theta|Z_1^N) d\theta$$

numerically using Monte Carlo methods [116, 158].

There are several approaches which could be interpreted as Gaussian processes [100]. One such example is [67], which uses a parameterized piecewise affine model with Gaussian *a priori* distributions for the parameters. Inference is done via Monte Carlo methods.

2.7 A Direct Weight Optimization Approach

Most of the different methods given in this chapter for choosing the weights w_k in the linear estimator (2.3) were all justified using asymptotic arguments, as $N \to \infty$. However, in reality only a finite number of data is given. Furthermore, these data may be sparsely and nonuniformly spread. This might deteriorate the performance of the estimation methods, as pointed out in [34]. In that paper, the suggested solution is to use either a nearest neighbor bandwidth selection approach, perhaps combined with an adaptive bandwidth selection using a localized C_p criterion.

In the following chapters, we will instead present a nonasymptotic approach for determining the weights, based on a uniform (over $\mathcal{F}_{p+1}(L)$) upper bound on the MSE (and hence on the worst-case MSE and linear minimax risk). Unlike, e.g., the local polynomial modelling methods, the approach directly optimizes the criterion with respect to the weights w_k, without using any kernel function or local polynomial model. Hence, it will be called a *direct weight optimization (DWO)* approach. Similar ideas were also given in [138].

Chapter 3 considers the case when f is univariate ($f : \mathbb{R} \to \mathbb{R}$) and belongs to $\mathcal{F}_2(L)$. The extension to multivariate functions and a Lipschitz condition on the pth derivative (i.e., $f \in \mathcal{F}_{p+1}(L)$) is then treated in Chapter 4.

If there are *a priori* known bounds on $f(\varphi_0)$ and/or $\nabla f(\varphi_0)$, this can easily be incorporated into the DWO approach. This is described in Chapter 5. Chapter 6 presents other possible extensions, such as estimating the derivative, while conclusions are given in Chapter 7.

3 Local DWO Modelling of Univariate Functions

As could be seen in Chapter 2 and Example 1.4, many methods for local modelling end up in a linear estimator in the form (2.3). The direct weight optimization (DWO) approach presented in this and following chapters is based on the idea of optimizing the linear estimator directly with respect to the weights, without taking the detour of fitting a local model to the data or deciding the weights from an asymptotically optimal kernel function. The criterion which is optimized is an upper bound on the (worst-case) MSE.

An early contribution following these lines is [138], where the authors consider the minimax estimation problem for an extension of the class $\mathcal{G}_{p+1}(L)$. The idea of obtaining the weights via optimization was also proposed in [146], where an estimate of the MSE is minimized. In the latter approach a bandwidth and an estimate of the Hessian have to be provided. This is solved via iterative search over different bandwidths, and by using a localized version of the criteria given in Section 2.4.3 to choose an appropriate bandwidth. The estimate of the Hessian is given by first estimating a local cubic model. In contrast, the method given in this thesis just needs the ratio L/σ in advance, where L is the Lipschitz constant in (2.7) and σ^2 is the noise variance. In the following chapters, we will assume that this ratio is given. However, if it is not known, a procedure similar to that in [146] could be used. This is further discussed in Chapter 7.

The present chapter focuses on the estimation of univariate functions with a Lipschitz condition on the first derivative, and presents the basic ideas behind the

DWO approach. Section 3.2 also shows some of the properties of the approach and its solutions.

3.1 The DWO Approach

Let us, just as in Chapter 2, consider the problem of estimating the value $f(\varphi_0)$ of an unknown univariate function $f : \mathbb{R} \to \mathbb{R}$ at a given point φ_0, given a set of input-output pairs $\{(\varphi(k), y(k))\}_{k=1}^N$, coming from the relation

$$y(k) = f(\varphi(k)) + e(k) \tag{3.1}$$

Here, we will assume that the function f is continuously differentiable, and that there is a Lipschitz constant L such that

$$|f'(\varphi(1)) - f'(\varphi(2))| \leq L|\varphi(1) - \varphi(2)| \quad \forall\, \varphi(1), \varphi(2) \in \mathbb{R} \tag{3.2}$$

We use the notation $\mathcal{F}_2(L)$, introduced in Section 2.1, for this class of functions.

As before, the noise terms $e(k)$ are independent random variables with $E[e(k)] = 0$ and $E[e^2(k)] = \sigma^2$, and are assumed to be independent of $\varphi(i)$ for all i. Both L and σ are assumed to be positive constants, given *a priori*. We also use the previously introduced notation

$$\widetilde{\varphi}(k) = \varphi(k) - \varphi_0 \tag{3.3}$$

To estimate $f(\varphi_0)$, we will use a linear estimator in the form (2.3), i.e.,

$$\hat{f}(\varphi_0) = \sum_{k=1}^N w_k y(k) \tag{3.4}$$

The criterion we consider is the worst-case MSE, defined by (2.60). For the estimator (3.4) and the function class $\mathcal{F}_2(L)$, the worst-case MSE becomes

$$\begin{aligned}
WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L)) &= \sup_{f \in \mathcal{F}_2(L)} E[\left(\sum_{k=1}^N w_k y(k) - f(\varphi_0)\right)^2 |\varphi_1^N] \\
&= \sup_{f \in \mathcal{F}_2(L)} E[\left(\sum_{k=1}^N w_k (f(\varphi(k)) + e(k)) - f(\varphi_0)\right)^2 |\varphi_1^N] \\
&= \sup_{f \in \mathcal{F}_2(L)} \left(\sum_{k=1}^N w_k f(\varphi(k)) - f(\varphi_0)\right)^2 + \sigma^2 \sum_{k=1}^N w_k^2 \qquad (3.5) \\
&= \sup_{f \in \mathcal{F}_2(L)} \Bigg(f(\varphi_0) \left(\sum_{k=1}^N w_k - 1\right) + f'(\varphi_0) \sum_{k=1}^N w_k \widetilde{\varphi}(k) \\
&\qquad + \sum_{k=1}^N w_k \left(f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k)\right)\Bigg)^2 + \sigma^2 \sum_{k=1}^N w_k^2
\end{aligned}$$

By Lemma A.3 in the Appendix, the last sum of the bias is bounded according to

$$|f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k)| \leq \frac{L}{2}\widetilde{\varphi}^2(k) \tag{3.6}$$

However, $f(\varphi)$ and $f'(\varphi)$ may be arbitrarily large. Hence, for the worst-case bias (and hence the worst-case MSE) in (3.5) to be finite, some requirements on the weights w_k are needed, namely

$$\sum_{k=1}^{N} w_k = 1 \tag{3.7a}$$

$$\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 0 \tag{3.7b}$$

These are exactly the same relations as (2.29), which were satisfied by the local linear estimator. If (3.7) does not hold, the bias is unbounded over $\mathcal{F}_2(L)$. Moreover, under these restrictions, any linear function is estimated with zero bias, since also the last term of the bias (the left hand side in (3.6)) is zero in this case.

Unfortunately, even though it turns out that the worst-case MSE (3.5) is a *convex* function of the weights w_k (this will be discussed further in Section 6.2), it is relatively difficult to minimize directly with respect to w. Instead, an upper bound on (3.5) will be derived and minimized in the following section. Section 6.2 analyzes the problem of minimizing the exact worst-case MSE. However, if considering the function class $\mathcal{G}_2(L)$ given by (2.9), we will see that the upper bound will be tight, and therefore it will be the exact worst-case MSE which is minimized.

3.1.1 Minimizing an Upper Bound on the Worst-Case MSE

Under the restrictions (3.7), the worst-case MSE becomes

$$\begin{aligned} &\mathit{WMSE}(\hat{f}(\varphi_0), \mathcal{F}_2(L)) \qquad (3.8)\\ &= \sup_{f \in \mathcal{F}_2(L)} \left(\sum_{k=1}^{N} w_k \left(f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k) \right) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned}$$

Using (3.6), we can get an upper bound on this expression

$$\mathit{WMSE}(\hat{f}(\varphi_0), \mathcal{F}_2(L)) \leq \frac{L^2}{4} \left(\sum_{k=1}^{N} \widetilde{\varphi}^2(k)|w_k| \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \tag{3.9}$$

Note that this bound is tight and attained by a parabola with $f''(\varphi) = L$ if the weights w_k are nonnegative. Also, if the function class $\mathcal{G}_2(L)$ is considered instead of $\mathcal{F}_2(L)$, we have

$$\sup_{f \in \mathcal{G}_2(L)} \left(\sum_{k=1}^{N} w_k \left(f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k) \right) \right)^2 = \frac{L^2}{4} \left(\sum_{k=1}^{N} \widetilde{\varphi}^2(k)|w_k| \right)^2$$

where the supremum is attained, e.g., if

$$f(\varphi_0) = f'(\varphi_0) = 0, \quad f(\varphi(k)) = \frac{L}{2}\operatorname{sgn}(w_k)\widetilde{\varphi}^2(k), \; k = 1, \dots, N$$

This means that the worst-case MSE in this case is given by the right-hand side of (3.9), and that we nonasymptotically reach the linear minimax risk.

It is now natural to minimize the upper bound in (3.9). Hence, we would like to choose the values of w_k that minimize the following convex optimization problem:

$$\begin{aligned} \min_{w} \quad & \frac{L^2}{4}\left(\sum_{k=1}^{N} \widetilde{\varphi}^2(k)|w_k|\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \\ \text{subj. to} \quad & \sum_{k=1}^{N} w_k = 1 \\ & \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 0 \end{aligned} \tag{3.10}$$

It turns out that this problem can be rewritten as a *convex quadratic program (QP)* [19], as the following theorem shows.

Theorem 3.1
Consider the optimization problem (3.10)*. The weight vector* $w^* = (w_1^* \; \dots \; w_N^*)^T$ *is a minimizer of* (3.10) *if and only if there is a vector* s^**, such that* (w^*, s^*) *is a minimizer of the following QP:*

$$\begin{aligned} \min_{w,s} \quad & \frac{L^2}{4}\left(\sum_{k=1}^{N} \widetilde{\varphi}^2(k)s_k\right)^2 + \sigma^2 \sum_{k=1}^{N} s_k^2 \\ \text{subj. to} \quad & s_k \geq w_k \\ & s_k \geq -w_k \\ & \sum_{k=1}^{N} w_k = 1 \\ & \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 0 \end{aligned} \tag{3.11}$$

Furthermore, $s_k^* = |w_k^*|$, $k = 1, \dots, N$.

Proof Given a feasible solution w to (3.10), we can get a feasible solution to (3.11) with the same value of the objective function by using the same w and $s_k = |w_k|$. Hence (3.11) is a relaxation of (3.10), and it suffices to show that when the optimal value of (3.11) is reached, then $s_k = |w_k|$ for all $k = 1, \dots, N$. Suppose, e.g., that $s_1 > |w_1|$. Then, without changing any other variables, the value of the objective function can be reduced by decreasing s_1. This can be seen by observing that the coefficient before s_1 in the first sum of the objective function is nonnegative, and the coefficient before s_1^2 in the second sum is positive, so decreasing s_1 will decrease the objective function. Therefore, when the objective function will reach its minimum, then $s_k = |w_k|$, and the equivalence is shown. □

Looking at the QP (3.11), if we divide the objective function by σ^2, we can note that only the ratio L/σ is important for determining the weights w_k^*, and not the actual values of L and σ. Intuitively, the larger ratio L/σ, the smaller number of data should be taken into account, since the bias term becomes more dominating with a larger ratio. Example 3.1 will illustrate this.

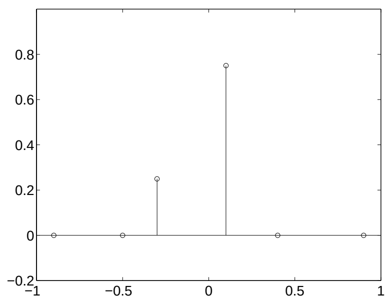
It should be pointed out, that the fact that the upper bound in (3.9) is tight for nonnegative weights w_k (considering the function class $\mathcal{F}_2(L)$) does not necessarily mean that minimizing (3.11) yields the weights that minimize the worst-case MSE, even if the resulting weights are positive. The reason for this is that a subset of the weights that really minimize the worst-case MSE may be negative, and so the upper bound is not tight for these weights. This is discussed in greater detail in Section 6.2.

Solving the QP (3.11) can be done very efficiently using standard solvers, e.g., Cplex [70].

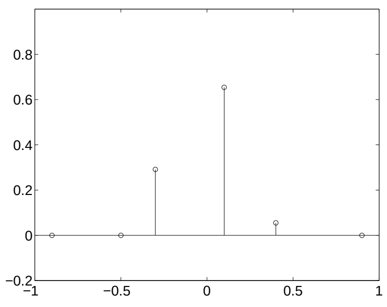
Example 3.1 *Consider once again the data set from Examples 1.4 and 2.1. For a given ratio L/σ, we can compute the weights w_k of the DWO approach by solving (3.11). The QP we need to solve can be written*

$$\begin{aligned}\min_{w,s} \quad & \frac{L^2}{4}\left((-0.9)^2 s_1 + (-0.5)^2 s_2 + (-0.3)^2 s_3 + 0.1^2 s_4 + 0.4^2 s_5 + 0.9^2 s_6\right)^2 \\ & + \sigma^2 \sum_{k=1}^{6} s_k^2 \\ \text{subj. to} \quad & s_k \geq w_k \\ & s_k \geq -w_k \\ & \sum_{k=1}^{6} w_k = 1 \\ & -0.9w_1 - 0.5w_2 - 0.3w_3 + 0.1w_4 + 0.4w_5 + 0.9w_6 = 0\end{aligned} \tag{3.12}$$

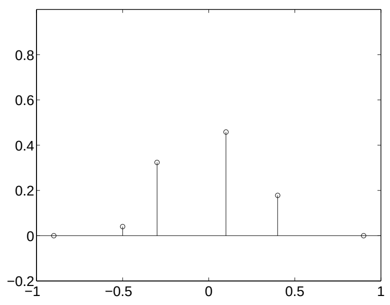
Figure 3.1 shows the weights for some different values of L. Note that for high values of L, most of the weights are zero. We will see in Section 3.2 that this is a general property. For very large L values, the weights equal those of the linear interpolation in Example 1.4. The smaller the value of L, the more nonzero weights we get. If $L = 0$, we obtain the same weights as for a global linear regression, as we will see later in Theorem 3.5.



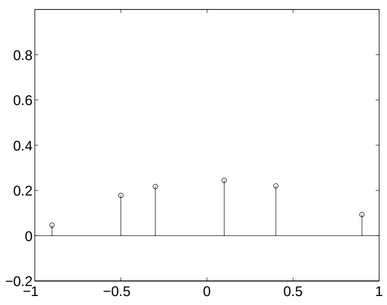
(a) $L/\sigma = 30$.



(b) $L/\sigma = 20$.



(c) $L/\sigma = 10$.



(d) $L/\sigma = 2$.

Figure 3.1: The weights obtained using DWO on the data set in Example 3.1 for some different values of L/σ.

3.2 Some Properties of the Approach

Having found the weights minimizing (3.10), let us analyze the solution and see what different properties it has.

3.2.1 Automatic Finite Bandwidth and Boundary Adaptation

An interesting feature of the DWO approach is that in most cases, the weights w_k corresponding to $\varphi(k)$ lying beyond a certain distance from φ_0 will be zero (see, e.g., Example 3.1). This means that we automatically get a finite bandwidth, and that the user does not have to bother about how many of the samples should be included in the estimator (in contrast to, e.g., the optimization approach in [146], where the bandwidth has to be specified). As pointed out, e.g., in [34], the fact that

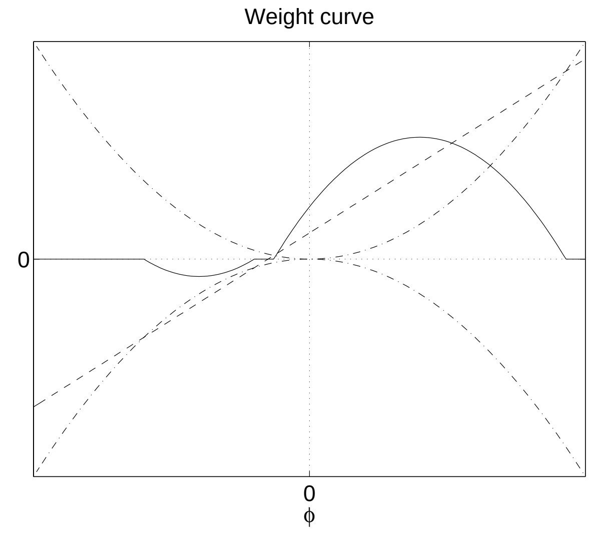


Figure 3.2: Basic shape of the weight curve (solid curve). The dash-dotted parabolas are $\pm g\widetilde{\varphi}^2$, and the dashed line is $\mu_1 + \mu_2\widetilde{\varphi}$. (The weight curve is scaled by a factor 4 to make the figure more clear.)

the bandwidth is finite is desirable from a computational point of view, since only a fraction of the data will be involved in the estimation. The following theorem holds for the DWO approach (a similar theorem, in a slightly different setting, was also shown in [138]; however, here we give an independent proof):

Theorem 3.2
Suppose that the problem (3.11) *is feasible, and* $\sigma > 0$. *Then there exist three numbers* $\mu_1 > 0$, μ_2, *and* $g \geq 0$, *such that for an optimal solution* (w^*, s^*), *we have*

$$w_k^* = \begin{cases} \mu_1 + \mu_2\widetilde{\varphi}(k) - g\widetilde{\varphi}^2(k), & g\widetilde{\varphi}^2(k) \leq \mu_1 + \mu_2\widetilde{\varphi}(k) \\ 0, & |\mu_1 + \mu_2\widetilde{\varphi}(k)| \leq g\widetilde{\varphi}^2(k) \\ \mu_1 + \mu_2\widetilde{\varphi}(k) + g\widetilde{\varphi}^2(k), & \mu_1 + \mu_2\widetilde{\varphi}(k) \leq -g\widetilde{\varphi}^2(k) \end{cases} \tag{3.13}$$

Moreover, if there are two indices k_1 *and* k_2, *such that* $0 \neq \widetilde{\varphi}(k_1) \neq \widetilde{\varphi}(k_2) \neq 0$, *then* $g > 0$.

Remark 3.1 *Figure 3.2 shows the basic shape of the curve along which the weights* w_k^* *are placed. When* $g\widetilde{\varphi}^2(k) \leq \mu_1 + \mu_2\widetilde{\varphi}(k)$ *(which in the figure corresponds to the dashed line being above the upper dash-dotted parabola), the weights will be positive. When* $\mu_1 + \mu_2\widetilde{\varphi}(k) \leq -g\widetilde{\varphi}^2(k)$ *(the dashed line is below the lower dash-dotted parabola), the weights are negative, and otherwise they are zero.*

In Figure 3.3, the weights are shown for some different cases. We can see clearly that the bandwidth increases with a decreasing ratio L/σ, and that negative weights occur if the data samples are asymmetrically placed around the point of interest.

More explicit expressions for the weights are given in Section 3.2.2.

Proof The proof uses the *Karush-Kuhn-Tucker (KKT) conditions* (see, e.g., [117]). Since the QP (3.11) is a convex optimization problem with linear constraints, the KKT conditions are necessary and sufficient conditions for optimality of a solution (see, e.g., [19] for details).

The Lagrangian function of (3.11) can be written

$$\begin{aligned}\mathcal{L}(w, s; \mu, \lambda) = \frac{L^2}{4}\left(\sum_{k=1}^{N} \widetilde{\varphi}^2(k) s_k\right)^2 + \sigma^2 \sum_{k=1}^{N} s_k^2 - 2\sigma^2 \mu_1 \left(\sum_{k=1}^{N} w_k - 1\right) \\ - 2\sigma^2 \mu_2 \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 2\sigma^2 \sum_{k=1}^{N} (\lambda_k^+ (s_k - w_k) + \lambda_k^- (s_k + w_k))\end{aligned} \tag{3.14}$$

where $\lambda_k^\pm \geq 0$, $k = 1, \dots, N$, and μ are the Lagrangian multipliers, scaled by a factor $1/2\sigma^2$. Since $s_k^* = |w_k^*|$ for an optimal solution (w^*, s^*), the KKT conditions are equivalent to the following relations:

$$\mu_1 + \mu_2 \widetilde{\varphi}(k) = \lambda_k^+ - \lambda_k^- \tag{3.15a}$$

$$\frac{L^2}{4\sigma^2}\left(\sum_{t=1}^{N} \widetilde{\varphi}^2(t) |w_t^*|\right) \widetilde{\varphi}^2(k) + |w_k^*| = \lambda_k^+ + \lambda_k^- \tag{3.15b}$$

$$\sum_{k=1}^{N} w_k^* = 1 \tag{3.15c}$$

$$\sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) = 0 \tag{3.15d}$$

$$s_k^* = |w_k^*| \tag{3.15e}$$

$$\lambda_k^+ (|w_k^*| - w_k^*) = 0 \tag{3.15f}$$

$$\lambda_k^- (|w_k^*| + w_k^*) = 0 \tag{3.15g}$$

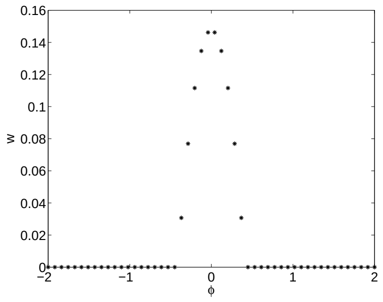
$$\lambda_k^\pm \geq 0, \quad k = 1, \dots, N \tag{3.15h}$$

Let

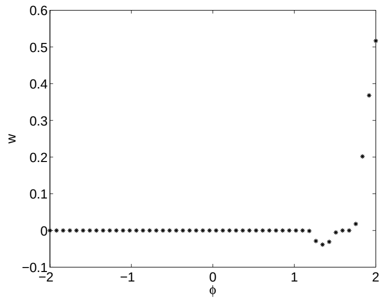
$$g = \frac{L^2}{4\sigma^2}\left(\sum_{t=1}^{N} \widetilde{\varphi}^2(t) |w_t^*|\right) \tag{3.16}$$

From (3.15f) and (3.15g), we can see that $w_k^* > 0$ implies $\lambda_k^- = 0$, and that $w_k^* < 0$ implies $\lambda_k^+ = 0$. Hence, we can eliminate $\lambda_k^\pm$ from the KKT conditions in these cases, getting

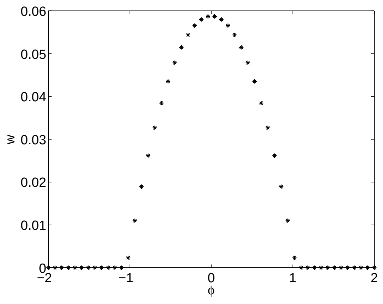
$$w_k^* = \mu_1 + \mu_2 \widetilde{\varphi}(k) - \operatorname{sgn}(w_k^*) g \widetilde{\varphi}^2(k), \quad w_k^* \neq 0 \tag{3.17}$$



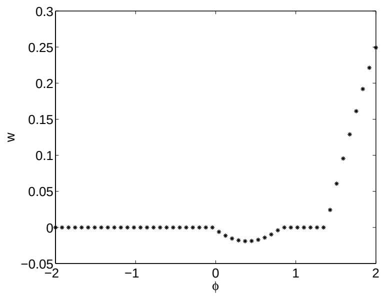
(a) $\varphi_0 = 0$, $L/\sigma = 10$.



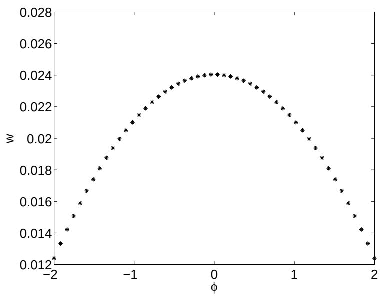
(b) $\varphi_0 = 2$, $L/\sigma = 10$.



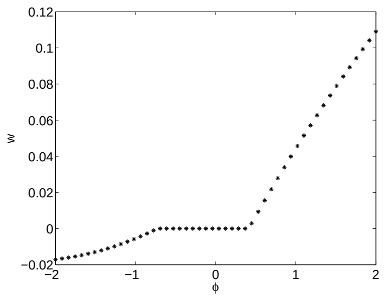
(c) $\varphi_0 = 0$, $L/\sigma = 1$.



(d) $\varphi_0 = 2$, $L/\sigma = 1$.



(e) $\varphi_0 = 0$, $L/\sigma = 0.1$.



(f) $\varphi_0 = 2$, $L/\sigma = 0.1$.

Figure 3.3: Weights for equally spaced data samples, for different values of L/σ and φ_0.

We can see that

$$\begin{aligned} w_k^* > 0 \quad &\Rightarrow \quad \mu_1 + \mu_2\widetilde{\varphi}(k) > g\widetilde{\varphi}^2(k) \\ w_k^* < 0 \quad &\Rightarrow \quad \mu_1 + \mu_2\widetilde{\varphi}(k) < -g\widetilde{\varphi}^2(k) \end{aligned}$$

Finally, if $w_k^* = 0$, we get from (3.15a), (3.15b), and (3.15h) that

$$\begin{aligned} 2\lambda_k^+ &= \mu_1 + \mu_2\widetilde{\varphi}(k) + g\widetilde{\varphi}^2(k) \geq 0 \\ 2\lambda_k^- &= -\mu_1 - \mu_2\widetilde{\varphi}(k) + g\widetilde{\varphi}^2(k) \geq 0 \end{aligned}$$

which implies

$$-g\widetilde{\varphi}^2(k) \leq \mu_1 + \mu_2\widetilde{\varphi}(k) \leq g\widetilde{\varphi}^2(k)$$

From these expressions, (3.13) is readily obtained.

Obviously, we have $g \geq 0$ from (3.16). Now suppose that there are two indices k_1 and k_2, such that $0 \neq \widetilde{\varphi}(k_1) \neq \widetilde{\varphi}(k_2) \neq 0$, and suppose that $g = 0$. Then (3.13) implies that $w_k^* = \mu_1 + \mu_2\widetilde{\varphi}(k)$ for all $k = 1, \ldots, N$. Furthermore, (3.16) implies that $w_{k_1}^* = w_{k_2}^* = 0$. But this means that $\mu_1 = \mu_2 = 0$, which makes $w_k^* = 0$ for all $k = 1, \ldots, N$. But this contradicts (3.15c), so $g > 0$.

To show that $\mu_1 > 0$, assume the opposite, i.e., $\mu_1 \leq 0$. We also assume $\mu_2 \geq 0$ (if μ_2 is negative, we can instead consider the problem we get by replacing $\widetilde{\varphi}(k)$ by $-\widetilde{\varphi}(k)$). Let S^+ be the set of indices k such that $w_k^* > 0$. Obviously, S^+ is nonempty; otherwise, (3.15c) will not be satisfied. Similarly, let S^- be the set of indices k such that $w_k^* < 0$.

First assume that $g > 0$. For all $k \in S^+$, it holds that

$$\mu_1 + \mu_2\widetilde{\varphi}(k) > g\widetilde{\varphi}^2(k) \qquad \Rightarrow \qquad \frac{\mu_2 - \sqrt{\mu_2^2 + 4\mu_1 g}}{2g} < \widetilde{\varphi}(k) < \frac{\mu_2 + \sqrt{\mu_2^2 + 4\mu_1 g}}{2g}$$

Similarly, for all $k \in S^-$, it holds that

$$\mu_1 + \mu_2\widetilde{\varphi}(k) < -g\widetilde{\varphi}^2(k) \quad \Rightarrow \quad \frac{-\mu_2 - \sqrt{\mu_2^2 - 4\mu_1 g}}{2g} < \widetilde{\varphi}(k) < \frac{-\mu_2 + \sqrt{\mu_2^2 - 4\mu_1 g}}{2g}$$

Since straightforward calculations show that

$$0 \leq \frac{-\mu_2 + \sqrt{\mu_2^2 - 4\mu_1 g}}{2g} \leq \frac{\mu_2 - \sqrt{\mu_2^2 + 4\mu_1 g}}{2g}$$

we have that

$$\max_{k \in S^-} \widetilde{\varphi}(k) < \min_{k \in S^+} \widetilde{\varphi}(k), \qquad 0 < \min_{k \in S^+} \widetilde{\varphi}(k) \tag{3.18}$$

If instead $g = 0$, it holds for all $k \in S^+$ that

$$\mu_1 + \mu_2\widetilde{\varphi}(k) > 0 \quad \Rightarrow \quad \mu_2 > 0, \ \widetilde{\varphi}(k) > 0$$

(this follows from the assumptions $\mu_1 \leq 0$ and $\mu_2 \geq 0$), and since for all $k \in S^-$

$$\mu_1 + \mu_2 \widetilde{\varphi}(k) < 0$$

we can conclude that (3.18) holds also in this case.

Now, from (3.15c), (3.15d), and (3.18) we get

$$\begin{aligned}
0 &= \sum_{k \in S^+} w_k^* \widetilde{\varphi}(k) + \sum_{k \in S^-} w_k^* \widetilde{\varphi}(k) \\
&\geq \min_{k \in S^+} \widetilde{\varphi}(k) \sum_{k \in S^+} w_k^* + \max_{k \in S^-} \widetilde{\varphi}(k) \sum_{k \in S^-} w_k^* \\
&\geq \min_{k \in S^+} \widetilde{\varphi}(k) \left(\sum_{k \in S^+} w_k^* + \sum_{k \in S^-} w_k^* \right) \\
&= \min_{k \in S^+} \widetilde{\varphi}(k) > 0
\end{aligned}$$

and we have a contradiction. It follows that $\mu_1 > 0$, and the theorem is proved. □

By examining the proof of Theorem 3.2, we can make an additional observation. Suppose that we have an optimal solution w^* to (3.11), with corresponding Lagrangian multipliers $\lambda^{\pm}$ and μ, and constant g given by (3.16). Now, let us extend the data set $\widetilde{\varphi}(k)$, $k = 1, \ldots, N$ by adding a point $\widetilde{\varphi}(N+1)$ that satisfies

$$-g\widetilde{\varphi}^2(N+1) \leq \mu_1 + \mu_2 \widetilde{\varphi}(N+1) \leq g\widetilde{\varphi}^2(N+1) \tag{3.19}$$

and consider the optimization problem (3.11) for the new data set. It turns out that the KKT conditions (3.15) will be satisfied by the same weights w^* as before, together with $w_{N+1}^* = s_{N+1}^* = 0$ and the Lagrangian multipliers μ, $\lambda^{\pm}$, and $\lambda_{N+1}^{\pm}$, where $\lambda_{N+1}^{\pm}$ are given by (3.15a) and (3.15b). In other words, the solution does not change, except that $w_{N+1}^* = 0$ and $\lambda_{N+1}^{\pm}$ are appended. We summarize this in a corollary:

Corollary 3.1
Suppose that $\varphi(k)$, $k = 1, \ldots, N$, L, and σ are given, and let (w^*, s^*) be an optimal solution to (3.11). Let $\lambda^{\pm}$ and μ be the corresponding Lagrangian multipliers, and g the constant defined by (3.16). If the data set is extended by $\widetilde{\varphi}(N+1)$ satisfying (3.19), then an optimal solution to (3.11) using the extended data set is given by

$$\begin{pmatrix} s^* \\ 0 \end{pmatrix}, \quad \begin{pmatrix} w^* \\ 0 \end{pmatrix}$$

with the corresponding Lagrangian multipliers

$$\begin{pmatrix} \lambda^+ \\ \lambda_{N+1}^+ \end{pmatrix}, \quad \begin{pmatrix} \lambda^- \\ \lambda_{N+1}^- \end{pmatrix}, \quad \mu$$

where

$$\lambda^+_{N+1} = \frac{1}{2}\left(\frac{L^2}{4\sigma^2}\left(\sum_{t=1}^{N}\widetilde{\varphi}^2(t)|w^*_t|\right)\widetilde{\varphi}^2(N+1) + \mu_1 + \mu_2\widetilde{\varphi}(N+1)\right)$$
$$\lambda^-_{N+1} = \frac{1}{2}\left(\frac{L^2}{4\sigma^2}\left(\sum_{t=1}^{N}\widetilde{\varphi}^2(t)|w^*_t|\right)\widetilde{\varphi}^2(N+1) - \mu_1 - \mu_2\widetilde{\varphi}(N+1)\right)$$

The result opens up for a possible reduction of the computational complexity: Since many of the weights w_k will be zero, we can already beforehand exclude data that will most likely correspond to zero weights, thus making the QP (3.11) considerably smaller. Having solved (3.11), one can easily check whether or not the excluded weights really should be zero, by checking if the excluded data points satisfy $|\mu_1 + \mu_2\widetilde{\varphi}(k)| \leq g\widetilde{\varphi}^2(k)$ (the middle case of (3.13)). In Section 7.4, this is discussed in some more detail.

For the case when the data samples $\widetilde{\varphi}(k)$ are lying symmetrically, the following theorem shows that the weights are nonnegative, and lie along the positive part of a parabola symmetrically around φ_0 (compare this with the local linear estimator (2.28)).

Theorem 3.3
Assume that the data samples $\widetilde{\varphi}(k)$ in (3.11) are lying symmetrically, i.e., that the nonzero $\widetilde{\varphi}(k)$ can be paired so that for each pair $(\widetilde{\varphi}(i), \widetilde{\varphi}(j))$ we have $\widetilde{\varphi}(i) = -\widetilde{\varphi}(j)$. Also assume that $\widetilde{\varphi}(k) \neq 0$ for at least one (hence two) indices k. Then $\mu_2 = 0$, i.e., the weights satisfy

$$w^*_k = \max\{\mu_1 - g\widetilde{\varphi}^2(k), 0\} \tag{3.20}$$

with $\mu_1 > 0$, $g > 0$.

Proof The positiveness of μ_1 and g follows directly from Theorem 3.2 and the fact that we have two distinctive, nonzero $\widetilde{\varphi}(k)$. For the proof we now need to consider the explicit expressions of the weights and Lagrangian multipliers that will be given in Section 3.2.2. Provided that the weights w^*_k are all nonnegative, we get from (3.32) that $\mu_2 = 0$. Since the KKT conditions are sufficient for optimality, it then suffices to show that there exists a solution with nonnegative weights w^*_k to (3.15). This will be done by induction. We assume that the data samples $\widetilde{\varphi}(k)$ are ordered by ascending magnitude. Let us also use the notation $w^{[N_0]}_k$ and $\mu^{[N_0]}$ for the solution to (3.15), using the set of data samples $\widetilde{\varphi}(1), \ldots, \widetilde{\varphi}(N_0)$, and $g^{[N_0]}$ for the corresponding g, given by (3.16).

For the case of just two nonzero data samples, the theorem is easily checked – (3.32)-(3.33) is obviously a solution to (3.15).

Now suppose that the theorem holds for all cases of at most $N_0 - 1$ samples, and consider the case of N_0 samples. The optimal weights $w^{[N_0-2]}_k$ using $N_0 - 2$

samples are given by (3.33), and the corresponding $\mu^{[N_0-2]}$ is given by (3.32). For $\widetilde{\varphi}(N_0)$ (and $\widetilde{\varphi}(N_0-1) = -\widetilde{\varphi}(N_0)$), there are now two possibilities: Either

$$\mu_1^{[N_0-2]} \leq g^{[N_0-2]}\widetilde{\varphi}^2(N_0) \tag{3.21}$$

or

$$g^{[N_0-2]}\widetilde{\varphi}^2(N_0) < \mu_1^{[N_0-2]} \tag{3.22}$$

In the first case we can apply Corollary 3.1, and we get a solution by letting $w_k^{[N_0]} = w_k^{[N_0-2]}$ for $k = 1, \ldots, N_0-2$, and $w_{N_0-1}^{[N_0]} = w_{N_0}^{[N_0]} = 0$. If on the other hand (3.22) holds, we need to show that there is a nonnegative solution to (3.15). For this, it is sufficient to show that the weights we get using (3.33) are all positive, because then the KKT conditions will be satisfied using the corresponding $\mu^{[N_0]}$ from (3.32). But from (3.33), (3.20), and (3.22) we get that

$$\begin{aligned}
w_i^{[N_0]} &= \frac{2}{8N_0\frac{\sigma^2}{L^2} + \sum_{j=1}^{N_0}\sum_{k=1}^{N_0}(\widetilde{\varphi}^2(j) - \widetilde{\varphi}^2(k))^2} \\
&\qquad \cdot \left(4\frac{\sigma^2}{L^2} + \sum_{k=1}^{N_0}\widetilde{\varphi}^4(k) - \left(\sum_{k=1}^{N_0}\widetilde{\varphi}^2(k)\right)\widetilde{\varphi}^2(i)\right) \\
&= \frac{2}{8N_0\frac{\sigma^2}{L^2} + \sum_{j=1}^{N_0}\sum_{k=1}^{N_0}(\widetilde{\varphi}^2(j) - \widetilde{\varphi}^2(k))^2} \\
&\quad \cdot \left(4\frac{\sigma^2}{L^2} + \sum_{k=1}^{N_0-2}\widetilde{\varphi}^4(k) - \left(\sum_{k=1}^{N_0-2}\widetilde{\varphi}^2(k)\right)\widetilde{\varphi}^2(i) + 2\widetilde{\varphi}^2(N_0)(\widetilde{\varphi}^2(N_0) - \widetilde{\varphi}^2(i))\right) \\
&= \frac{2}{8N_0\frac{\sigma^2}{L^2} + \sum_{j=1}^{N_0}\sum_{k=1}^{N_0}(\widetilde{\varphi}^2(j) - \widetilde{\varphi}^2(k))^2} \\
&\quad \cdot \Bigg(\left(4(N_0-2)\frac{\sigma^2}{L^2} + \frac{1}{2}\sum_{j=1}^{N_0-2}\sum_{k=1}^{N_0-2}(\widetilde{\varphi}^2(j) - \widetilde{\varphi}^2(k))^2\right)\left(\mu_1^{[N_0-2]} - g^{[N_0-2]}\widetilde{\varphi}^2(i)\right) \\
&\qquad\qquad + 2\widetilde{\varphi}^2(N_0)(\widetilde{\varphi}^2(N_0) - \widetilde{\varphi}^2(i))\Bigg) \\
&> 0, \qquad i = 1, \ldots, N_0
\end{aligned}$$

where in the last inequality we have used the fact that

$$g^{[N_0-2]}\widetilde{\varphi}^2(i) \leq g^{[N_0-2]}\widetilde{\varphi}^2(N) \leq \mu_1^{[N_0-2]}, \qquad i = 1, \ldots, N_0$$

by (3.22). Hence, all weights (for all $i = 1, \ldots, N_0$) given by (3.33) are positive, and the theorem is proven. □

Since the DWO method automatically decides which data samples should be taken into account by minimizing (3.10), it also automatically adapts to the case when the data samples are lying asymmetrically, e.g., when φ_0 is a boundary point, or when only few data are available. The estimate will always be optimal in the worst-case MSE sense if considering $\mathcal{G}_2(L)$, and in the sense of (3.10) for $\mathcal{F}_2(L)$. This is a great advantage, especially compared to the kernel estimators, but also to the local polynomial modelling approach.

3.2.2 Explicit Expressions for the Optimal Weights

Let us take a closer look at (3.11), and derive explicit expressions for the weights. Assume that L and σ are both strictly positive. Let S be the set of indices k such that $w_k^* \neq 0$. Let also

$$r_k = \begin{cases} 1 & w_k^* > 0 \\ 0 & w_k^* = 0 \\ -1 & w_k^* < 0 \end{cases} \tag{3.23}$$

For the nonzero weights (i.e., for $k \in S$), (3.16) and (3.17) give

$$w_k^* = \mu_1 + \mu_2 \widetilde{\varphi}(k) - r_k \frac{L^2}{4\sigma^2} \left(\sum_{t \in S} r_t \widetilde{\varphi}^2(t) w_t^* \right) \widetilde{\varphi}^2(k) \tag{3.24}$$

Reorder the data samples so that $w_1^*, \ldots, w_n^*$ are the nonzero weights. Using the notation

$$1_n = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n, \qquad \widetilde{\varphi}_{1:n} = \begin{pmatrix} \widetilde{\varphi}(1) \\ \vdots \\ \widetilde{\varphi}(n) \end{pmatrix}, \qquad z = \begin{pmatrix} r_1 \widetilde{\varphi}^2(1) \\ \vdots \\ r_n \widetilde{\varphi}^2(n) \end{pmatrix}, \qquad w^* = \begin{pmatrix} w_1^* \\ \vdots \\ w_n^* \end{pmatrix}$$

we can write the expression (3.24), together with the constraints (3.7), in matrix form as follows:

$$\begin{pmatrix} -\frac{L^2}{4\sigma^2} z z^T - I & 1_n & \widetilde{\varphi}_{1:n} \\ 1_n^T & 0 & 0 \\ \widetilde{\varphi}_{1:n}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} w^* \\ \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{3.25}$$

This system is closely related to the in optimization well-known KKT system (see [117]). According to Lemma A.1, the system is nonsingular if $L/\sigma > 0$[*] and there are at least two distinct values of $\widetilde{\varphi}(k)$, $k = 1, \ldots, n$. Furthermore, we get the

[*]In fact, the system is nonsingular even if $L = 0$, as long as the other requirement holds (see, e.g., [117]). However, for simplicity we only consider $L > 0$ here. The case $L = 0$ will be considered in Section 3.2.7.

solution

$$
\begin{aligned}
w_i^* = \frac{1}{\zeta}\Bigg(& \left(\gamma \sum_{k=1}^{n} \widetilde{\varphi}^2(k) - \left(\sum_{k=1}^{n} r_k \widetilde{\varphi}^3(k)\right)^2\right) \\
& + \left(\sum_{k=1}^{n} r_k \widetilde{\varphi}^2(k) \sum_{k=1}^{n} r_k \widetilde{\varphi}^3(k) - \gamma \sum_{k=1}^{n} \widetilde{\varphi}(k)\right) \widetilde{\varphi}(i) \\
& + \left(\sum_{k=1}^{n} \widetilde{\varphi}(k) \sum_{k=1}^{n} r_k \widetilde{\varphi}^3(k) - \sum_{k=1}^{n} \widetilde{\varphi}^2(k) \sum_{k=1}^{n} r_k \widetilde{\varphi}^2(k)\right) r_i \widetilde{\varphi}^2(i) \Bigg)
\end{aligned} \tag{3.26}
$$

$$
\mu = \frac{1}{\zeta} \begin{pmatrix} \gamma \sum_{k=1}^{n} \widetilde{\varphi}^2(k) - \left(\sum_{k=1}^{n} r_k \widetilde{\varphi}^3(k)\right)^2 \\ -\gamma \sum_{k=1}^{n} \widetilde{\varphi}(k) + \sum_{k=1}^{n} r_k \widetilde{\varphi}^2(k) \sum_{k=1}^{n} r_k \widetilde{\varphi}^3(k) \end{pmatrix} \tag{3.27}
$$

where

$$
\begin{aligned}
\gamma &= 4\frac{\sigma^2}{L^2} + z^T z \\
\zeta &= \frac{1}{6} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \Big(\widetilde{\varphi}(i)(r_k \widetilde{\varphi}^2(k) - r_j \widetilde{\varphi}^2(j)) + \widetilde{\varphi}(j)(r_i \widetilde{\varphi}^2(i) - r_k \widetilde{\varphi}^2(k)) \\
&\quad + \widetilde{\varphi}(k)(r_j \widetilde{\varphi}^2(j) - r_i \widetilde{\varphi}^2(i)) \Big)^2 + 2\frac{\sigma^2}{L^2} \sum_{i=1}^{n} \sum_{k=1}^{n} (\widetilde{\varphi}(i) - \widetilde{\varphi}(k))^2
\end{aligned}
$$

As mentioned in the remark after Lemma A.1, we can note that γ and ζ are both strictly positive.

3.2.3 Expressions for Nonnegative Weights

Suppose now that all nonzero weights are positive, i.e., that $r_k = 1$, $k = 1, \ldots, n$. In this case, the expressions can be somewhat simplified.

Note that ζ can also be written as

$$
\begin{aligned}
\zeta &= \frac{1}{6} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} (\widetilde{\varphi}(i) - \widetilde{\varphi}(j))^2 (\widetilde{\varphi}(j) - \widetilde{\varphi}(k))^2 (\widetilde{\varphi}(k) - \widetilde{\varphi}(i))^2 \\
&\quad + 2\frac{\sigma^2}{L^2} \sum_{i=1}^{n} \sum_{k=1}^{n} (\widetilde{\varphi}(i) - \widetilde{\varphi}(k))^2 \\
&= \frac{1}{6} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} (\varphi(i) - \varphi(j))^2 (\varphi(j) - \varphi(k))^2 (\varphi(k) - \varphi(i))^2 \\
&\quad + 2\frac{\sigma^2}{L^2} \sum_{i=1}^{n} \sum_{k=1}^{n} (\varphi(i) - \varphi(k))^2
\end{aligned} \tag{3.28}
$$

since

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(\widetilde{\varphi}(i)-\widetilde{\varphi}(j))^2(\widetilde{\varphi}(j)-\widetilde{\varphi}(k))^2(\widetilde{\varphi}(k)-\widetilde{\varphi}(i))^2\\
&\quad=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\Big(\widetilde{\varphi}(i)\widetilde{\varphi}(j)\widetilde{\varphi}(k)-\widetilde{\varphi}^2(i)\widetilde{\varphi}(j)-\widetilde{\varphi}(i)\widetilde{\varphi}^2(k)+\widetilde{\varphi}^2(i)\widetilde{\varphi}(k)\\
&\qquad\quad-\widetilde{\varphi}^2(j)\widetilde{\varphi}(k)+\widetilde{\varphi}(i)\widetilde{\varphi}^2(j)+\widetilde{\varphi}(j)\widetilde{\varphi}^2(k)-\widetilde{\varphi}(i)\widetilde{\varphi}(j)\widetilde{\varphi}(k)\Big)^2\\
&\quad=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\Big(\widetilde{\varphi}(i)(\widetilde{\varphi}^2(j)-\widetilde{\varphi}^2(k))+\widetilde{\varphi}(j)(\widetilde{\varphi}^2(k)-\widetilde{\varphi}^2(i))\\
&\qquad\quad+\widetilde{\varphi}(k)(\widetilde{\varphi}^2(i)-\widetilde{\varphi}^2(j))\Big)^2
\end{aligned}$$

This means that ζ is independent of φ_0 as long as the same weights are positive. After simple but tedious calculations, we get the following expressions for μ and w^*:

$$\begin{aligned}
\mu&=\frac{1}{\zeta}\begin{pmatrix}\gamma\sum_{k=1}^{n}\widetilde{\varphi}^2(k)-\left(\sum_{k=1}^{n}\widetilde{\varphi}^3(k)\right)^2\\ \sum_{k=1}^{n}\widetilde{\varphi}^2(k)\sum_{k=1}^{n}\widetilde{\varphi}^3(k)-\gamma\sum_{k=1}^{n}\widetilde{\varphi}(k)\end{pmatrix}\\
&=\frac{1}{\zeta}\begin{pmatrix}\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\widetilde{\varphi}^2(j)\widetilde{\varphi}^2(k)(\widetilde{\varphi}(j)-\widetilde{\varphi}(k))^2+4\frac{\sigma^2}{L^2}\sum_{k=1}^{n}\widetilde{\varphi}^2(k)\\ -\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\widetilde{\varphi}(j)\widetilde{\varphi}(k)(\widetilde{\varphi}(j)+\widetilde{\varphi}(k))(\widetilde{\varphi}(j)-\widetilde{\varphi}(k))^2-4\frac{\sigma^2}{L^2}\sum_{k=1}^{n}\widetilde{\varphi}(k)\end{pmatrix}
\end{aligned}\tag{3.29}$$

and

$$\begin{aligned}
w_i^*&=\frac{1}{\zeta}\Bigg(\left(\gamma\sum_{k=1}^{n}\widetilde{\varphi}^2(k)-\left(\sum_{k=1}^{n}\widetilde{\varphi}^3(k)\right)^2\right)\\
&\quad+\left(\sum_{k=1}^{n}\widetilde{\varphi}^2(k)\sum_{k=1}^{n}\widetilde{\varphi}^3(k)-\gamma\sum_{k=1}^{n}\widetilde{\varphi}(k)\right)\widetilde{\varphi}(i)\\
&\quad+\left(\sum_{k=1}^{n}\widetilde{\varphi}(k)\sum_{k=1}^{n}\widetilde{\varphi}^3(k)-\left(\sum_{k=1}^{n}\widetilde{\varphi}^2(k)\right)^2\right)\widetilde{\varphi}^2(i)\Bigg)\\
&=\frac{1}{2\zeta}\Bigg(\left(\sum_{j=1}^{n}\sum_{k=1}^{n}\widetilde{\varphi}^2(j)\widetilde{\varphi}^2(k)(\widetilde{\varphi}(j)-\widetilde{\varphi}(k))^2+8\frac{\sigma^2}{L^2}\sum_{k=1}^{n}\widetilde{\varphi}^2(k)\right)\\
&\quad-\left(\sum_{j=1}^{n}\sum_{k=1}^{n}\widetilde{\varphi}(j)\widetilde{\varphi}(k)(\widetilde{\varphi}(j)+\widetilde{\varphi}(k))(\widetilde{\varphi}(j)-\widetilde{\varphi}(k))^2+8\frac{\sigma^2}{L^2}\sum_{k=1}^{n}\widetilde{\varphi}(k)\right)\widetilde{\varphi}(i)
\end{aligned}\tag{3.30}$$

$$
\begin{aligned}
&+ \left(\sum_{j=1}^{n} \sum_{k=1}^{n} \widetilde{\varphi}(j)\widetilde{\varphi}(k)(\widetilde{\varphi}(j) - \widetilde{\varphi}(k))^2 \right) \widetilde{\varphi}^2(i) \Bigg) \\
&= \frac{1}{2\zeta} \left(\sum_{j=1}^{n} \sum_{k=1}^{n} (\varphi(i) - \varphi(j))(\varphi(i) - \varphi(k))(\varphi(j) - \varphi(k))^2 (\varphi(j) - \varphi_0)(\varphi(k) - \varphi_0) \right. \\
&\left. + 8\frac{\sigma^2}{L^2} \sum_{k=1}^{n} (\varphi(k) - \varphi(i))(\varphi(k) - \varphi_0) \right), \qquad i = 1, \ldots, n \qquad (3.31)
\end{aligned}
$$

In particular, when $\widetilde{\varphi}(k)$ are lying symmetrically around 0, all odd terms will disappear from the expressions above, and we get (after some calculations)

$$
\zeta = \frac{1}{2} \sum_{k=1}^{n} \widetilde{\varphi}^2(k) \left(\sum_{j=1}^{n} \sum_{k=1}^{n} (\widetilde{\varphi}^2(j) - \widetilde{\varphi}^2(k))^2 + 8n\frac{\sigma^2}{L^2} \right)
$$

which gives

$$
\mu_1 = \frac{2\gamma}{\sum_{j=1}^{n} \sum_{k=1}^{n} (\widetilde{\varphi}^2(j) - \widetilde{\varphi}^2(k))^2 + 8n\frac{\sigma^2}{L^2}}, \qquad \mu_2 = 0 \qquad (3.32)
$$

and

$$
w_i^* = \frac{2}{\sum_{j=1}^{n} \sum_{k=1}^{n} (\widetilde{\varphi}^2(j) - \widetilde{\varphi}^2(k))^2 + 8n\frac{\sigma^2}{L^2}} \left(\gamma - \left(\sum_{k=1}^{n} \widetilde{\varphi}^2(k) \right) \widetilde{\varphi}^2(i) \right) \qquad (3.33)
$$

3.2.4 Relation to Local Polynomial Modelling

We can compare the DWO approach with the local linear estimator in the following way. Use the notation of Section 3.2.2. Let

$$
\Phi_n = \begin{pmatrix} 1_n & \widetilde{\varphi}_{1:n} \end{pmatrix} \qquad (3.34)
$$

$$
G = \frac{L^2}{4\sigma^2} z z^T + I \qquad (3.35)
$$

where z is defined as in Section 3.2.2. Comparing (3.25) to the proof of Lemma A.1 in the Appendix shows that the weights of the DWO approach can be written as

$$
w^* = G^{-1}\Phi_n(\Phi_n^T G^{-1}\Phi_n)^{-1}\mathbf{e}_1 \qquad (3.36)
$$

This can now be compared with the equivalent weights (2.26) of local polynomial modelling. A word of caution is needed, though: In (3.35) and (3.36), only the data samples corresponding to nonzero weights are included. On the other hand, in the presentation of the local polynomial modelling, Φ includes all data samples, also those for which we get zero weights. Analogously, $\bar{K}_h$ generally contains many zero entries in the diagonal. However, the solution of the local polynomial modelling problem (2.19) will not be affected by deleting all terms for which $K_h(\widetilde{\varphi}(k)) = 0$, and so we may without restriction assume that that we only consider the data samples for which the equivalent weights (2.26) are nonzero (which also implies that $\bar{K}_h$ is invertible). We can now see that the weights from the DWO approach equal those obtained from a local polynomial estimator, with $\bar{K}_h$ from (2.21) replaced by G^{-1} from (3.35). Hence, if we only knew z, i.e., which weights w_k^* should be positive and which should be negative, we could obtain w^* by solving a weighted least-squares problem like (2.23). However, this is of course not known beforehand.

3.2.5 Asymptotic Behavior

In [85], it was shown that using the Epanechnikov kernel would yield an asymptotically optimal (continuous) kernel estimator with respect to the worst-case MSE over the function class $\mathcal{G}_2(L)$. Since, as we have seen, the optimal weights from (3.11) minimize the worst-case MSE over $\mathcal{G}_2(L)$, one would expect that the weights w_k of the DWO approach would asymptotically converge to the weights using the Epanechnikov kernel with a bandwidth given by (2.71). This is also the case under certain assumptions, as the following theorem shows.

Theorem 3.4
Consider the problem of estimating an unknown function $f \in \mathcal{F}_2(L)$ at a given internal point $\varphi_0 \in (-1/2, 1/2)$ under an equally spaced fixed design model

$$\varphi(k) = \frac{k-1}{N-1} - \frac{1}{2}, \quad k = 1, \dots, N \tag{3.37}$$

and with $\sigma > 0$. Let w^* be the minimizer of (3.10). *Then asymptotically, as $N \to \infty$,*

$$w_k^* \approx \frac{3}{4} C_N \max\{1 - \left(\frac{\widetilde{\varphi}(k)}{h_N}\right)^2, 0\}, \quad k = 1, \dots, N \tag{3.38}$$

where

$$C_N \asymp \frac{1}{N h_N}, \quad h_N \asymp \left(\frac{15\sigma^2}{L^2 N}\right)^{1/5} \quad \text{as } N \to \infty \tag{3.39}$$

Hence, the optimal weights (3.38) *approximately coincide with related asymptotically optimal weights and bandwidth* (2.71) *of the local polynomial estimator for the worst-case function in $\mathcal{F}_2(L)$.*

Here $a_N \asymp b_N$ means asymptotic equivalence of two real sequences (a_N) and (b_N), that is $a_N/b_N \to 1$ as $N \to \infty$.

Remark 3.2 *When the data are lying symmetrically around φ_0, e.g., when $\varphi_0 = 0$, it follows directly from Theorem 3.3 that the relation* (3.38) *will hold exactly also for finite N, i.e.,*

$$w_k^* = \frac{3}{4} C_N \max\{1 - \left(\frac{\widetilde{\varphi}(k)}{h_N}\right)^2, 0\}, \quad k = 1, \dots, N \tag{3.40}$$

where C_N and h_N are given by (3.39).

Proof Let us apply Theorem 3.2, from which it follows that there are three numbers $\mu_1 > 0$, μ_2, and $g > 0$, such that

$$w_k^* = \max\{\mu_1 + \mu_2\widetilde{\varphi}(k) - g\widetilde{\varphi}^2(k), 0\}, \quad k = 1, \dots, N \tag{3.41}$$

if and only if $\mu_1 + \mu_2\widetilde{\varphi}(k) + g\widetilde{\varphi}^2(k) \geq 0$ for all $k = 1, \dots, N$, which is the case if

$$\mu_2^2 \leq 4g\mu_1 \tag{3.42}$$

Also recall that the KKT conditions (3.15) applied in the proof of Theorem 3.2 represent necessary and sufficient conditions for optimality of the solution to the considered QP problem. Thus, in order to prove the first part of the theorem, it suffices to demonstrate that

$$\lim_{N\to\infty} \frac{\mu_2^2}{g\mu_1} = 0 \tag{3.43}$$

for the three parameters μ_1, μ_2, and g satisfying (3.15c), (3.15d), and (3.16), with the weights w_k^* given by (3.41). Denote the support of the function $w(\widetilde{\varphi}) = \max\{\mu_1 + \mu_2\widetilde{\varphi} - g\widetilde{\varphi}^2, 0\}$ by $[a, b]$, that is

$$\mu_1 + \mu_2 a - ga^2 = 0, \quad \mu_1 + \mu_2 b - gb^2 = 0, \quad a < b \tag{3.44}$$

and suppose that $[a, b] \in [-0.5 - \varphi_0, 0.5 - \varphi_0]$. If we find a solution to the system of the three equations (3.15c), (3.15d), and (3.16) with respect to $\mu_1 > 0$, μ_2, and $g > 0$, and (3.42) is satisfied, then we have proved (3.41). The following asymptotic relation for nonnegative weights (3.41) holds true as $N \to \infty$:

$$\frac{1}{N}\sum_{k=1}^{N} w_k\widetilde{\varphi}^m(k) = \int_a^b \left(\mu_1 + \mu_2\widetilde{\varphi} - g\widetilde{\varphi}^2\right)\widetilde{\varphi}^m d\widetilde{\varphi} + O(h/N)\left(\mu_1 + |\mu_2| + g\right) \tag{3.45}$$

for any $m = 0, 1, 2$, where

$$h = \frac{b - a}{2}$$

Thus, the equations (3.15c), (3.15d), and (3.16) may be written as follows:

$$\frac{1}{N} = \int_a^b \left(\mu_1 + \mu_2\widetilde{\varphi} - g\widetilde{\varphi}^2\right) d\widetilde{\varphi} + O(h/N)\left(\mu_1 + |\mu_2| + g\right) \tag{3.46}$$

$$0 = \int_a^b \left(\mu_1 + \mu_2\widetilde{\varphi} - g\widetilde{\varphi}^2\right)\widetilde{\varphi} d\widetilde{\varphi} + O(h/N)\left(\mu_1 + |\mu_2| + g\right) \tag{3.47}$$

$$\frac{4\sigma^2}{L^2}\frac{g}{N} = \int_a^b \left(\mu_1 + \mu_2\widetilde{\varphi} - g\widetilde{\varphi}^2\right)\widetilde{\varphi}^2 d\widetilde{\varphi} + O(h/N)\left(\mu_1 + |\mu_2| + g\right) \tag{3.48}$$

with

$$a = \frac{\mu_2 - \sqrt{\mu_2^2 + 4g\mu_1}}{2g}, \quad b = \frac{\mu_2 + \sqrt{\mu_2^2 + 4g\mu_1}}{2g}, \quad h = \frac{\sqrt{\mu_2^2 + 4g\mu_1}}{2g} \tag{3.49}$$

Note that the terms $O(h/N)$ in (3.46)–(3.48) do not depend on (μ_1, μ_2, g). Consequently, $O(h/N)|\mu_2|$ is uniformly bounded over μ_2 as $N \to \infty$.

Now, one might verify by direct substitution (see Section 3.4 for a detailed proof) that the solution to (3.46)–(3.48) has the following asymptotics:

$$\mu_1 \asymp \frac{3}{4Nh_N}, \quad \mu_2 = O(N^{-1}), \quad g \asymp \frac{\mu_1}{h_N^2} \tag{3.50}$$

with

$$h = \frac{\sqrt{\mu_2^2 + 4g\mu_1}}{2g} \asymp h_N \asymp \left(\frac{15\sigma^2}{L^2 N}\right)^{1/5} \tag{3.51}$$

Thus, we obtain

$$\lim_{N\to\infty} \frac{\mu_2^2}{g\mu_1} = \lim_{N\to\infty} \frac{\mu_1}{g}\left(\frac{\mu_2}{\mu_1}\right)^2 = 0 \tag{3.52}$$

and relation (3.43) is proved.

Since $\mu_2 = o(\mu_1)$, the relation (3.38) follows directly from (3.50). This proves the theorem. $\square$

3.2.6 The Estimated Function

Another interesting aspect to consider is how the estimate $\hat{f}(\varphi_0)$ behaves as a function of φ_0. For this purpose we use the explicit expressions from Section 3.2.2. To begin with, let us consider the case when w_k are nonnegative for all k. In this case, we know from (3.28) that ζ does not depend on φ_0, and from (3.30), we can see that φ_0 enters quadratically in the numerator of the expression for the weights. Furthermore, since $\hat{f}(\varphi_0)$ is a linear combination of the weights, the quadratic characteristics are inherited by $\hat{f}(\varphi_0)$ from w_k. Altogether, this means that $\hat{f}(\varphi_0)$ is a *piecewise quadratic* function in the intervals of φ_0 for which the weights w_k are all nonnegative.

When there are negative weights, ζ is no longer independent of φ_0. It can be shown that ζ is a fourth order polynomial in φ_0. Since φ enters also the numerator polynomially, the result is that $\hat{f}(\varphi_0)$ is a *piecewise rational* function in the intervals of φ_0 where some weights w_k are negative.

One should note, however, that $\hat{f}(\varphi_0)$ does not necessarily satisfy the Lipschitz condition (3.2), since the noise in the observations may be arbitrarily large, and hence may scale terms in $\hat{f}(\varphi_0)$ to an arbitrary degree. This phenomenon is also pointed out in [137].

Example 3.2 *Let us consider a simple example, where $N = 11$ data samples $\varphi(1), \dots, \varphi(11)$ are equally spaced on the interval $[-2, 2]$. Let $L/\sigma = 1$. Suppose that the function $f(\varphi)$ is to be estimated in several different points φ_0. In Figure 3.4, the values of the weights $w_1, \dots, w_6$ are plotted as functions of φ_0. (The plots for the remaining weights $w_7, \dots, w_{11}$ will be a mirrored version of $w_1, \dots, w_5$ due to the symmetry.) Also the (numerically computed) derivatives of the weights with respect to φ_0 are plotted. As we can see, when $\varphi_0 \in [-2, -1.5]$ or $\varphi_0 \in [1.5, 2]$ (approximately), some weights are negative. For the remaining interval, i.e., for $\varphi_0 \in [-1.5, 1.5]$, the weights are clearly piecewise quadratic functions of φ_0.*

When the weights are piecewise quadratic, so is also $\hat{f}(\varphi_0)$, since it is a linear combination of the weights. In Figure 3.5, $\hat{f}(\varphi_0)$ is shown for a particular set of observations.

3.2.7 Global Models

What happens when $L = 0$, i.e., when the underlying function is linear? What one would expect is that the resulting estimator would coincide with the usual global linear regression, i.e., that it uses all data samples to estimate a linear function which minimizes an unweighted least-squares criterion. The following theorem shows that this is exactly what happens.

Theorem 3.5
Suppose that there are at least two indices k_1 and k_2, such that $\varphi(k_1) \neq \varphi(k_1)$. Then, when $L = 0$, the DWO approach gives the same result as a local linear model with $K_h(\varphi) \equiv 1$.

Proof When $L = 0$, (3.10) becomes

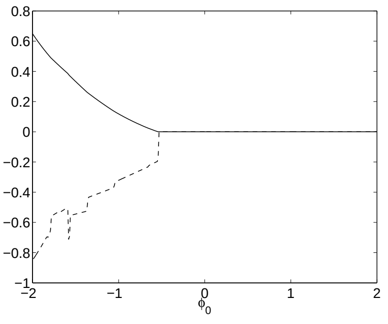
$$\begin{aligned} \min_w \quad & \sigma^2 \sum_{k=1}^N w_k^2 \\ \text{subj. to} \quad & \sum_{k=1}^N w_k = 1 \\ & \sum_{k=1}^N w_k \widetilde{\varphi}(k) = 0 \end{aligned} \tag{3.53}$$

Letting

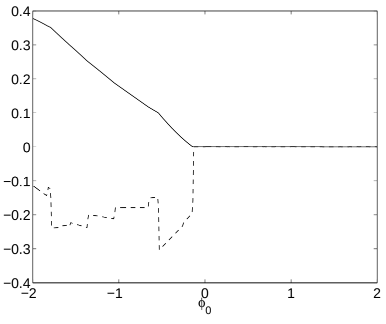
$$\Phi^T = \begin{pmatrix} 1 & \dots & 1 \\ \varphi(1) & \dots & \varphi(N) \end{pmatrix}$$

the problem (3.53) is equivalent to

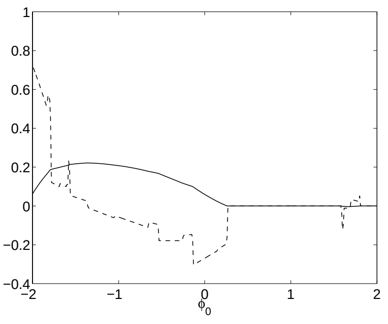
$$\begin{aligned} \min_w \quad & \|w\|^2 \\ \text{subj. to} \quad & \Phi^T w = \mathbf{e}_1 \end{aligned}$$



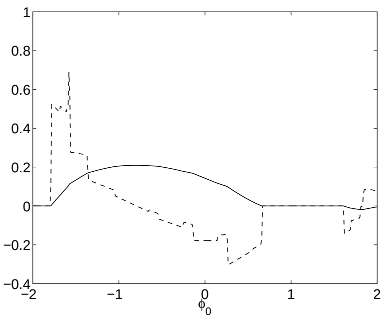
(a) w_1 ($\varphi(1) = -2$).



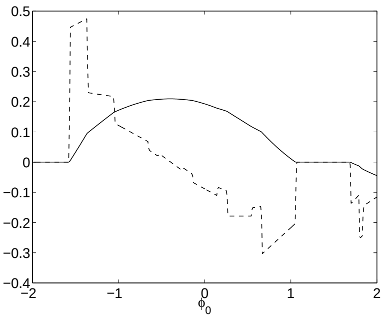
(b) w_2 ($\varphi(2) = -1.6$).



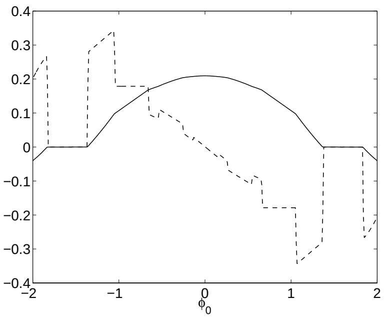
(c) w_3 ($\varphi(3) = -1.2$).



(d) w_4 ($\varphi(4) = -0.8$).



(e) w_5 ($\varphi(5) = -0.4$).



(f) w_6 ($\varphi(6) = 0$).

Figure 3.4: The weights w_k from Example 3.2 (solid) and their numerical derivatives (dashed), plotted as functions of φ_0.

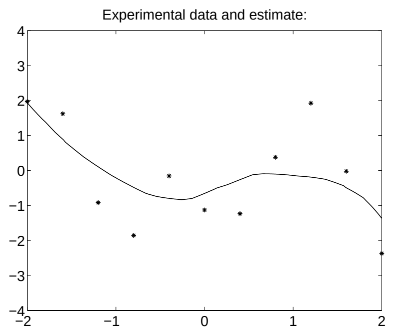


Figure 3.5: The estimated function $\hat{f}(\varphi_0)$ in Example 3.2.

Thus, what we would like to do is to find the least-norm solution to an underdetermined system of linear equations. But this is just what we get by using the pseudoinverse of Φ:

$$w^* = (\Phi^T)^\dagger \mathbf{e}_1 = \Phi(\Phi^T\Phi)^{-1}\mathbf{e}_1$$

which is the same as (2.26) with $\bar{K}_h = I$. □

Remark 3.3 *Instead of referring to the pseudoinverse, the solution of* (3.2.7) *can be obtained using the KKT conditions analogously to Lemma A.1 (with $\rho = 0$).*

Note that fitting a local linear model with $K_h(\varphi) \equiv 1$ to the data is not the same as fitting an ARX model, i.e., the weights corresponding to a local linear model (2.26) are not the same as the ones obtained in (1.10) for the ARX model. The reason for this is that the ARX model assumes that $f(0) = 0$ (it is a linear model) while the local linear model, in spite of its name, is an affine model, and also contains a parameter corresponding to a constant term.

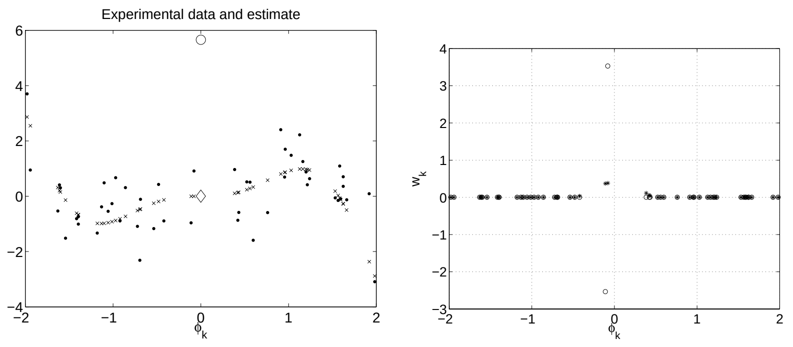
3.3 Examples

To illustrate a possible situation where the DWO approach described in Section 3.1 might be superior to the local linear estimator, we consider the following example.

Example 3.3 *Consider the function*

$$f(\varphi) = \varphi^2 \sin(2\varphi) \tag{3.54}$$

and let $\varphi(k)$, $k = 1, \ldots, 50$, be taken from a uniform distribution on $[-2, 2]$. *We let $\sigma^2 = 1$ and $L = 13$. Suppose that we would like to estimate $f(0)$. One example is given in Figure 3.6. The prediction errors, worst-case bias, variance, and MSE for this particular set of data $\{(\varphi(k), y(k))\}_{k=1}^{50}$ are listed in Table 3.1 for the two*



(a) Noiseless data ($\times$), data with noise ($\cdot$), and estimates of $f(0)$ using a local linear estimator ($\circ$) and QP ($\diamond$).

(b) Weights w_k from the local linear estimator ($\circ$) and QP ($*$).

Figure 3.6: Comparison of the local linear estimator, using the Epanechnikov kernel with bandwidth given by (2.71), and the DWO approach.

Table 3.1: Comparison of the asymptotically optimal local linear estimator and the DWO approach for the example shown in Figure 3.6.

	Local linear	DWO
Prediction error	-5.6567	-0.0016
(Bias)^2 (upper bound)	0.1148	0.0942
Variance	18.8892	0.3044
MSE (upper bound)	19.0040	0.3987

different approaches. Note that all values, except for the prediction errors, only depend on the values of $\varphi(k)$, not on $y(k)$.

As we can see in Figure 3.6(b), most weights are zero, both in the local linear and DWO approach. In the local linear estimator, the kernel function together with the bandwidth decides which weights should be zero. In the DWO approach, however, this is taken care of by the automatic finite bandwidth property mentioned in Section 3.2, which allows for a greater flexibility, such that the minimal upper bound on the MSE can be achieved. In this specific example, the effect is that the local linear estimator just takes the two data points closest to the point of interest (which both happen to be negative) into account. This makes the estimate very sensitive to the actual noise realization. The DWO approach, on the other hand, takes three additional points into account, of which two are positive, thereby making the estimate much less noise sensitive.

Table 3.2 shows the resulting estimates of the actual MSE from four Monte-Carlo simulations (with 10000 experiments each), where $\varphi(k)$ are taken from a *uniform distribution on $[-2,2]$ and $\sigma^2 = 1$. $f(\varphi_0)$ is estimated for $\varphi_0 = 0$ and $\varphi_0 = 1.5$, using $N = 20$ or $N = 50$ observations. As can be seen, the DWO approach performs better than the local linear approach in all four cases, and the difference is accentuated when the number of data is small, just as expected.*

Table 3.2: Comparison of the asymptotically optimal local linear estimator and the DWO approach; results from Monte-Carlo simulations.

φ_0	N	MSE (Local linear)	MSE (DWO)
0	50	0.2240	0.1424
1.5	50	0.2289	0.1785
0	20	26.9244	0.3182
1.5	20	9.5489	0.5087

The objective function of (3.10) gives an upper bound on the MSE, which is then minimized. To give an idea of how tight the upper bound is, it can be compared to a lower bound. A trivial lower bound is given by the variance (2.35) of the estimator, which only depends on σ and w. This is considered in the following example, which also shows the weight curves for some different cases.

Example 3.4 *Consider the same setup as in Example 3.3, but with $N = 10$. Figure 3.7 shows the weights for one realization of data samples when estimating $f(0)$, $f(1)$, and $f(2)$, respectively. Note that in each case, the weights lie along a curve as given by (3.13). Note also that the weights automatically adapt to data lying nonsymmetrically, e.g., at the boundary.*

Figure 3.8 shows the upper bound (3.9) on the worst-case MSE as a function of φ_0. The lower bound given by the variance is also plotted, like the actual MSE for the function given by (3.54). As we can see, the upper bound is reasonably tight.

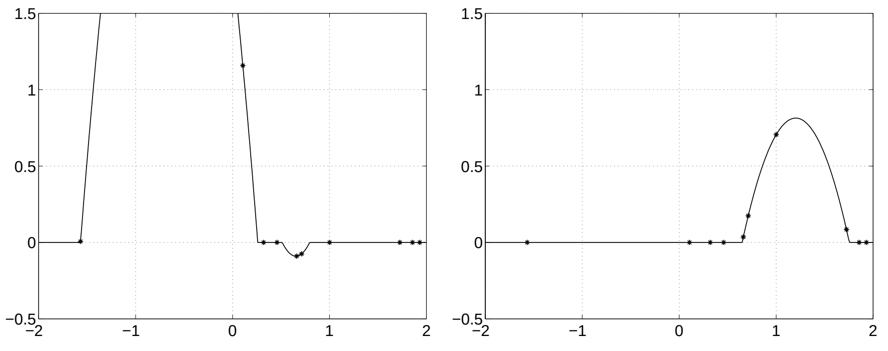
3.4 Proof of the Asymptotic Expressions in Theorem 3.4

We finish this chapter by giving some additional details for the proof of Theorem 3.4. Let us seek the solution to the system of equations given by (3.46)–(3.49) in the following asymptotic form (as $N \to \infty$)

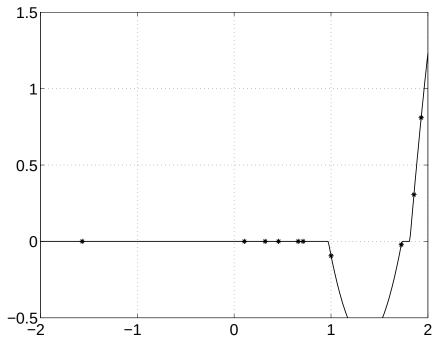
$$\mu_1 = \frac{\nu_1}{Nh_N}(1+o(1)), \quad \mu_2 = \frac{\nu_2}{N}(1+o(1)), \quad g = \frac{\nu_3\mu_1}{h_N^2}(1+o(1)) \tag{3.55}$$

with indefinite finite constants ν_i, $i = 1,2,3$, and with h_N as defined by (3.39). The goal here is to evaluate ν_i, $i = 1,2,3$. Note that

$$\frac{\mu_1}{g} = \frac{h_N^2}{\nu_3}(1+o(1)), \quad \frac{\mu_2}{g} = \frac{\nu_2 h_N^3}{\nu_3\nu_1}(1+o(1)) \tag{3.56}$$



(a) $\varphi_0 = 0$.   (b) $\varphi_0 = 1$.



(c) $\varphi_0 = 2$.

Figure 3.7: Weights and weight curves for Example 3.4, with $\varphi_0 = 0$, $\varphi_0 = 1$, and $\varphi_0 = 2$.

and

$$\widetilde{\varphi}_0 \triangleq \frac{a+b}{2} = \frac{\mu_2}{2g} \asymp \frac{\nu_2}{2\nu_3\nu_1} h_N^3 \tag{3.57}$$

Hence

$$h = \frac{\sqrt{\mu_2^2 + 4g\mu_1}}{2g} \asymp \sqrt{\frac{\mu_1}{g}} \asymp \frac{1}{\sqrt{\nu_3}} h_N, \quad a \asymp -h, \quad b \asymp h \tag{3.58}$$

Now evaluate the following integrals, occurring in (3.46)–(3.48):

$$\int_a^b d\widetilde{\varphi} = 2h \tag{3.59}$$

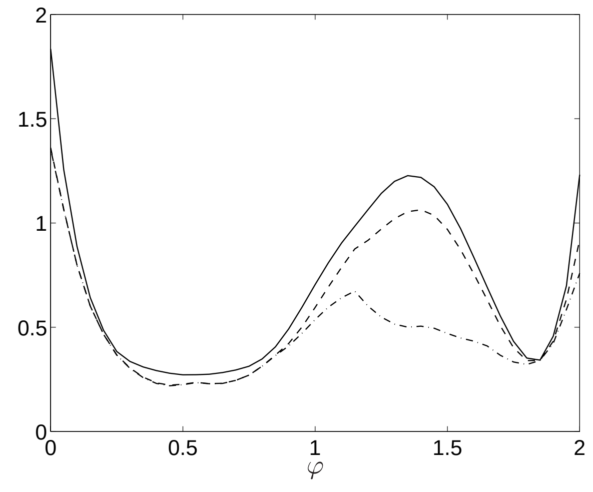


Figure 3.8: The upper bound (3.9) on the MSE (solid), the variance of the estimate (dash-dotted) and the exact MSE for (3.54) (dashed), using the data sample in Example 3.4.

$$\int_a^b \widetilde{\varphi} d\widetilde{\varphi} = 2h\widetilde{\varphi}_0 = O(h^4) \tag{3.60}$$

$$\int_a^b \widetilde{\varphi}^2 d\widetilde{\varphi} \asymp \frac{2}{3}h^3 \tag{3.61}$$

$$\int_a^b \widetilde{\varphi}^3 d\widetilde{\varphi} \asymp 2h^3\widetilde{\varphi}_0 = O(h^6) \tag{3.62}$$

$$\int_a^b \widetilde{\varphi}^4 d\widetilde{\varphi} \asymp \frac{2}{5}h^5 \tag{3.63}$$

Thus, taking representation (3.55) into account, and letting $N \to \infty$, we reduce the equations (3.46)–(3.48) to that of

$$1 = 2\nu_1 - \frac{2}{3}\nu_1\nu_3 \tag{3.64}$$

$$0 = \nu_1 O(1) + \frac{2}{3}\nu_2 - \nu_1\nu_3 O(1) \tag{3.65}$$

$$0 = \frac{2}{3}\nu_1 - \nu_1\nu_3\left(\frac{2}{5} + \frac{4}{15}\nu_3^{5/2}\right) \tag{3.66}$$

from what the unique nontrivial solution

$$\nu_1 = \frac{3}{4}, \quad \nu_2 = O(1), \quad \nu_3 = 1 \tag{3.67}$$

follows directly. This proves the asymptotic relations (3.50).

4

Local DWO Modelling of Multivariate Functions

In Chapter 3, the DWO approach was presented for the case of univariate functions. In this chapter, the approach is extended to multivariate functions, which is useful in many applications, particularly when considering dynamic systems. Furthermore, the functions in Chapter 3 were assumed to be once continuously differentiable, and the first derivatives should satisfy a Lipschitz condition. This can be extended to any degree of assumed differentiability. As we will see, many of the properties shown in Chapter 3 will also carry over to the multivariate case.

4.1 Problem Formulation

The problem we consider is the same as in Chapter 3, except that the function f is multivariate, $f : \mathbb{R}^n \to \mathbb{R}$, and that the assumptions are somewhat different. We assume that $f \in \mathcal{F}_{p+1}(L)$, which means that f is p times continuously differentiable ($p \geq 1$), and that the pth derivative satisfies a Lipschitz condition. By the latter will be meant that there is a constant L, such that

$$\max_{\|\xi\|=1} \left| \sum_{i_1=1}^{n} \cdots \sum_{i_p=1}^{n} \left(f^{(p)}_{i_1 \ldots i_p}(\varphi + h) - f^{(p)}_{i_1 \ldots i_p}(\varphi) \right) \xi_{i_1} \ldots \xi_{i_p} \right| \leq L\|h\| \tag{4.1}$$

for all $\varphi, h \in \mathbb{R}^n$. Here $f^{(p)}_{i_1 \dots i_p}$ means

$$f^{(p)}_{i_1 \dots i_p} = \frac{\partial^p f}{\partial \varphi_{i_1} \dots \partial \varphi_{i_p}} \tag{4.2}$$

One can show that, for $p = 1$, (4.1) is equivalent to (2.7), i.e., to

$$\|\nabla f(\varphi + h) - \nabla f(\varphi)\| \leq L\|h\|$$

for all $\varphi, h \in \mathbb{R}^n$.

The noise assumptions are the same as in Chapter 3. As before, f will be estimated using a linear estimator (3.4), where the weights are chosen to minimize an upper bound on the worst-case MSE (2.60). Remember, however, that in order for the worst-case MSE to be finite in the univariate case, the weights of (3.4) had to satisfy the additional requirements (3.7). The corresponding requirements for the multivariate case are given in the following theorem.

Theorem 4.1
Consider $f : \mathbb{R}^n \to \mathbb{R}$, $f \in \mathcal{F}_{p+1}(L)$. Then, for a linear estimator (3.4), *the worst-case MSE is finite if and only if*

$$\begin{gathered}
\sum_{k=1}^{N} w_k = 1 \\
\sum_{k=1}^{N} w_k \widetilde{\varphi}_{i_1}(k) = 0 \qquad i_1 = 1, \dots, n \\
\vdots \\
\sum_{k=1}^{N} w_k \widetilde{\varphi}_{i_1}(k) \cdot \dots \cdot \widetilde{\varphi}_{i_p}(k) = 0 \qquad i_j = 1, \dots, n,\ j = 1, \dots, p
\end{gathered} \tag{4.3}$$

On this subspace, the following upper bound can be given for the worst-case MSE:

$$WMSE(\hat{f}(\varphi_0), \mathcal{F}_{p+1}(L)) \leq \left(\frac{L}{(p+1)!} \sum_{k=1}^{N} |w_k| \|\widetilde{\varphi}(k)\|^{p+1} \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \tag{4.4}$$

Proof Analogously to (3.5), the worst-case MSE can be written as

$$\begin{aligned}
WMSE(\hat{f}(\varphi_0), \mathcal{F}_{p+1}(L)) &= \sup_{f \in \mathcal{F}_{p+1}(L)} \left(\sum_{k=1}^{N} w_k f(\varphi(k)) - f(\varphi_0) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \\
&= \sup_{f \in \mathcal{F}_{p+1}(L)} \Bigg(f(\varphi_0) \left(\sum_{k=1}^{N} w_k - 1 \right) + \sum_{i_1=1}^{n} f'_{i_1}(\varphi_0) \sum_{k=1}^{N} w_k \widetilde{\varphi}_{i_1}(k) + \dots \qquad (4.5) \\
&\quad + \frac{1}{p!} \sum_{i_1=1}^{n} \dots \sum_{i_p=1}^{n} f^{(p)}_{i_1 \dots i_p}(\varphi_0) \sum_{k=1}^{N} w_k \widetilde{\varphi}_{i_1}(k) \dots \widetilde{\varphi}_{i_p}(k)
\end{aligned}$$

$$+\sum_{k=1}^{N} w_k \Bigg(f(\varphi(k)) - f(\varphi_0) - \sum_{i_1=1}^{n} f'_{i_1}(\varphi_0)\widetilde{\varphi}_{i_1}(k) - \dots$$
$$- \frac{1}{p!}\sum_{i_1=1}^{n}\dots\sum_{i_p=1}^{n} f^{(p)}_{i_1\dots i_p}(\varphi_0)\widetilde{\varphi}_{i_1}(k)\dots\widetilde{\varphi}_{i_p}(k)\Bigg)\Bigg)^2 + \sigma^2\sum_{k=1}^{N} w_k^2$$

In this expression, all the terms of the bias, except for the last sum, are unbounded, and so, to guarantee a finite worst-case MSE, the requirements (4.3) need to be imposed. The last sum, on the other hand, is bounded according to Lemma A.3. Using this lemma and imposing (4.3), we can see that (4.4) is an upper bound on (4.5). □

We can note that instead of considering $\mathcal{F}_{p+1}(L)$, we could have considered a multivariate extension of the function class $\mathcal{G}_{p+1}(L)$ given by

$$f(\varphi) = f(\varphi_0) + \sum_{i_1=1}^{n} f'_{i_1}(\varphi_0)\widetilde{\varphi}_{i_1} + \dots + \frac{1}{p!}\sum_{i_1=1}^{n}\dots\sum_{i_p=1}^{n} f^{(p)}_{i_1\dots i_p}(\varphi_0)\widetilde{\varphi}_{i_1}\dots\widetilde{\varphi}_{i_p} \qquad (4.6)$$
$$+ c(\widetilde{\varphi})\|\widetilde{\varphi}\|^{p+1}$$

where

$$|c(\widetilde{\varphi})| \le \frac{L}{(p+1)!}$$

In this case, the bound on the worst-case MSE in (4.4) is tight and is attained if

$$c(\widetilde{\varphi}(k)) = \frac{L}{(p+1)!}\,\mathrm{sgn}(w_k)$$

Now, (4.4) can be minimized with respect to the weights w_k. This can be done using a QP, as the following theorem shows.

Theorem 4.2
Consider the following optimization problem:

$$\begin{aligned}
\min_{w} \quad & \left(\frac{L}{(p+1)!}\sum_{k=1}^{N}|w_k|\|\widetilde{\varphi}(k)\|^{p+1}\right)^2 + \sigma^2\sum_{k=1}^{N} w_k^2 \\
\text{subj. to} \quad & \sum_{k=1}^{N} w_k = 1 \\
& \sum_{k=1}^{N} w_k\widetilde{\varphi}_{i_1}(k) = 0 \qquad i_1 = 1,\dots,n \\
& \vdots \\
& \sum_{k=1}^{N} w_k\widetilde{\varphi}_{i_1}(k)\cdot\ldots\cdot\widetilde{\varphi}_{i_p}(k) = 0 \qquad i_j = 1,\dots,n,\ j = 1,\dots,p
\end{aligned} \qquad (4.7)$$

The weight vector $w^* = (w_1^* \ \dots \ w_N^*)^T$ is a minimizer of (4.7) if and only if there is a vector s^*, such that (w^*, s^*) is a minimizer of the following QP:

$$\begin{aligned}
\min_{w,s} \quad & \left(\frac{L}{(p+1)!} \sum_{k=1}^{N} s_k \|\widetilde{\varphi}(k)\|^{p+1} \right)^2 + \sigma^2 \sum_{k=1}^{N} s_k^2 \\
\text{subj. to} \quad & s_k \geq \pm w_k \\
& \sum_{k=1}^{N} w_k = 1 \\
& \sum_{k=1}^{N} w_k \widetilde{\varphi}_{i_1}(k) = 0 \qquad i_1 = 1, \dots, n \\
& \vdots \\
& \sum_{k=1}^{N} w_k \widetilde{\varphi}_{i_1}(k) \cdot \ldots \cdot \widetilde{\varphi}_{i_p}(k) = 0 \qquad i_j = 1, \dots, n, \ j = 1, \dots, p
\end{aligned} \tag{4.8}$$

Furthermore, $s_k^* = |w_k^*|$, $k = 1, \dots, N$.

Proof Completely analogous to Theorem 3.1. □

4.2 Properties

Several of the properties of the univariate case hold also for multivariate functions. For instance, when $L = 0$, the resulting weights equal those obtained by globally fitting a pth polynomial to the data. The proof is simple and a straightforward extension of Theorem 3.5.

Also the finite bandwidth property and boundary adaptation occur in the multivariate case. We can prove the following theorem, which corresponds to Theorem 3.2, and is also related to the asymptotic results of [85].

Theorem 4.3
Suppose that the problem (4.8) is feasible. Then there exist numbers $\mu^{(0)}$, $\mu_{i_1}^{(1)}, \dots, \mu_{i_1 \dots i_p}^{(p)}$, $i_j = 1, \dots, n$, $j = 1, \dots, p$, and $g \geq 0$, such that for an optimal solution (w^*, s^*), we have

$$w_k^* = \begin{cases} P(\widetilde{\varphi}(k)) - g\|\widetilde{\varphi}(k)\|^{p+1}, & g\|\widetilde{\varphi}(k)\|^{p+1} \leq P(\widetilde{\varphi}(k)) \\ 0, & -g\|\widetilde{\varphi}(k)\|^{p+1} \leq P(\widetilde{\varphi}(k)) \leq g\|\widetilde{\varphi}(k)\|^{p+1} \\ P(\widetilde{\varphi}(k)) + g\|\widetilde{\varphi}(k)\|^{p+1}, & P(\widetilde{\varphi}(k)) \leq -g\|\widetilde{\varphi}(k)\|^{p+1} \end{cases} \tag{4.9}$$

where

$$P(\widetilde{\varphi}(k)) = \mu^{(0)} + \sum_{i_1=1}^{n} \mu_{i_1}^{(1)} \widetilde{\varphi}_{i_1}(k) + \ldots + \sum_{i_1=1}^{n} \cdots \sum_{i_1=1}^{n} \mu_{i_1 \dots i_p}^{(p)} \widetilde{\varphi}_{i_1}(k) \cdot \ldots \cdot \widetilde{\varphi}_{i_p}(k) \tag{4.10}$$

Remark 4.1 *Note that the notation $\mu^{(j)}_{i_1\dots i_j}$ is used only to associate these numbers with the corresponding derivatives of f. The numbers $\mu^{(j)}_{i_1\dots i_j}$ are just constants, no derivatives or functions.*

Proof The proof uses the KKT conditions, which, since the QP (4.8) is a convex optimization problem with linear constraints, are necessary and sufficient conditions for optimality of a solution.

The Lagrangian function of (4.8) can be written

$$\begin{aligned}\mathcal{L}(w,s;\mu,\lambda) = &\left(\frac{L}{(p+1)!}\sum_{k=1}^{N} s_k\|\widetilde{\varphi}(k)\|^{p+1}\right)^2 + \sigma^2\sum_{k=1}^{N} s_k^2\\ &- 2\sigma^2\mu^{(0)}\left(\sum_{k=1}^{N} w_k - 1\right) - 2\sigma^2\sum_{i_1=1}^{n}\mu^{(1)}_{i_1}\sum_{k=1}^{N} w_k\widetilde{\varphi}_{i_1}(k) - \dots \qquad (4.11)\\ &- 2\sigma^2\sum_{i_1=1}^{n}\dots\sum_{i_p=1}^{n}\mu^{(p)}_{i_1\dots i_p}\sum_{k=1}^{N} w_k\widetilde{\varphi}_{i_1}(k)\dots\widetilde{\varphi}_{i_p}(k)\\ &- 2\sigma^2\sum_{k=1}^{N}(\lambda_k^+(s_k - w_k) + \lambda_k^-(s_k + w_k))\end{aligned}$$

where $\lambda_k^\pm \geq 0$, $k = 1,\dots,N$, and $\mu^{(0)}$, $\mu^{(j)}_{i_1\dots i_j}$, $i = 1,\dots,n$, $j = 1,\dots,p$ are the Lagrangian multipliers, scaled by a factor $1/2\sigma^2$. Since $s_k^* = |w_k^*|$ for an optimal solution (w^*, s^*), the KKT conditions are equivalent to the following relations:

$$P(\widetilde{\varphi}(k)) = \lambda_k^+ - \lambda_k^- \qquad (4.12a)$$

$$\frac{L^2}{((p+1)!)^2\sigma^2}\left(\sum_{t=1}^{N}|w_t^*|\|\widetilde{\varphi}(t)\|^{p+1}\right)\|\widetilde{\varphi}(k)\|^{p+1} + |w_k^*| = \lambda_k^+ + \lambda_k^- \qquad (4.12b)$$

$$\left.\begin{aligned}&\sum_{k=1}^{N} w_k^* = 1\\ &\sum_{k=1}^{N} w_k\widetilde{\varphi}_{i_1}(k) = 0 \qquad i_1 = 1,\dots,n\\ &\vdots\\ &\sum_{k=1}^{N} w_k\widetilde{\varphi}_{i_1}(k)\cdot\ldots\cdot\widetilde{\varphi}_{i_p}(k) = 0 \qquad i_j = 1,\dots,n,\ j = 1,\dots,p\end{aligned}\right. \qquad (4.12c)$$

$$s_k^* = |w_k^*| \qquad (4.12d)$$

$$\lambda_k^+(|w_k^*| - w_k^*) = 0 \qquad (4.12e)$$

$$\lambda_k^-(|w_k^*| + w_k^*) = 0 \tag{4.12f}$$

$$\lambda_k^\pm \geq 0, \quad k = 1, \ldots, N \tag{4.12g}$$

Let

$$g = \frac{L^2}{((p+1)!)^2\sigma^2}\left(\sum_{t=1}^{N} |w_t^*| \|\widetilde{\varphi}(t)\|^{p+1}\right) \tag{4.13}$$

From (4.12e) and (4.12f), we can see that $w_k^* > 0$ implies $\lambda_k^- = 0$, and that $w_k^* < 0$ implies $\lambda_k^+ = 0$. Hence, we can eliminate $\lambda_k^\pm$ from the KKT conditions in these cases, getting

$$w_k^* = P(\widetilde{\varphi}(k)) - \operatorname{sgn}(w_k^*) g \|\widetilde{\varphi}(k)\|^{p+1}, \quad w_k^* \neq 0 \tag{4.14}$$

We can see that

$$\begin{aligned} w_k^* > 0 \quad &\Rightarrow \quad P(\widetilde{\varphi}(k)) > g\|\widetilde{\varphi}(k)\|^{p+1} \\ w_k^* < 0 \quad &\Rightarrow \quad P(\widetilde{\varphi}(k)) < -g\|\widetilde{\varphi}(k)\|^{p+1} \end{aligned}$$

Finally, if $w_k^* = 0$, we get from (4.12a), (4.12b), and (4.12g) that

$$\begin{aligned} 2\lambda_k^+ &= P(\widetilde{\varphi}(k)) + g\|\widetilde{\varphi}(k)\|^{p+1} \geq 0 \\ 2\lambda_k^- &= -P(\widetilde{\varphi}(k)) + g\|\widetilde{\varphi}(k)\|^{p+1} \geq 0 \end{aligned}$$

which implies

$$-g\|\widetilde{\varphi}(k)\|^{p+1} \leq P(\widetilde{\varphi}(k)) \leq g\|\widetilde{\varphi}(k)\|^{p+1}$$

From these expressions, (4.9) is readily obtained.

□

4.3 Examples

The extension of the DWO approach to the multivariate case makes it applicable to many applications, e.g., to the prediction problem for dynamic systems. In this section, we will see two examples of this. As pointed out in Section 2.1, having a regression vector that depends on old values of y means that $\varphi(i)$ and $e(j)$ will not be independent for all values of i and j. For the time being, we will simply neglect this and assume that the effects will be small. The issue is discussed further in Section 7.1.

Example 4.1 *Consider the following extended version of the linear system known as the* Åström system *[94]:*

$$\begin{aligned} y(t) = {} & \\ & 1.5y(t-1) - 0.7y(t-2) + u(t-1) + 0.5u(t-2) \\ & + \alpha + L_0(\cos y(t-1) + 0.5u^2(t-1)) + e(t) \end{aligned} \tag{4.15}$$

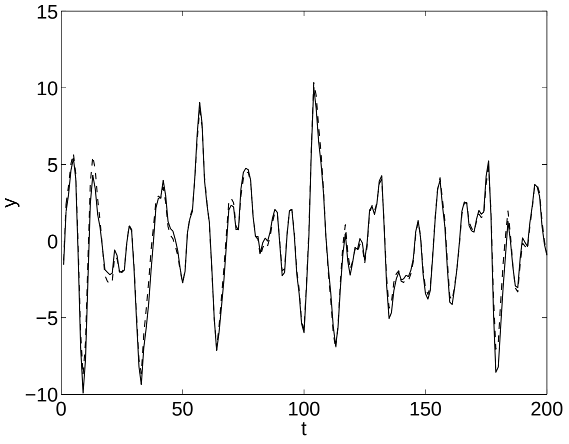


Figure 4.1: Simulated (solid) and true (dashed) output for system (4.15) with $L_0 = 0$, modelled using the DWO approach with $L = 0.01$.

First, set α and L_0 to zero. To get estimation data, $u(t)$ and $e(t)$ are both selected as random Gaussian sequences of length 500*, with unit variance. As validation data,* 200 *samples of noiseless data are selected, with $u(t)$ generated in the same way as for the estimation data. The simulated output for $L = 0.01$ is shown in Figure 4.1. As can be seen, the simulated output follows the true output well (*84.9% *fit). For $L = 0$, the result is the same as fitting an affine model using a least-squares criterion, according to Theorem 3.5, giving* 85.9% *fit. As comparison, a linear ARX model was also estimated, and performed slightly better compared to the other approaches (*91.8% *fit), as expected, since the true system was linear.*

Choosing $L_0 = L = 1$, $\alpha = 1$ and using an estimation data sequence of 50000 samples (generated as above), yielded a fit of 52.4% *for the simulated output, as compared to* 41.8% *for a linear ARX model (estimated after removing means from the data). The corresponding numbers for one-step-ahead predictions were* 94.8% *and* 77.5%*, respectively.*

In the latter experiment, the input was generally in the interval $[-4, 4]$*, while the output was generally in the interval* $[-10, 30]$*. This means that the linear and nonlinear terms of the input in* (4.15) *are of about the same order, while the linear terms of the output dominate over the nonlinear term. Thus, the nonlinear effects might not be overwhelming, which may explain that the ARX model performed fairly well in comparison.*

Example 4.2 *As another example, a nonlinear benchmark system proposed by [115] is considered. The system is defined in state-space form by*

$$x_1(t+1) = \left(\frac{x_1(t)}{1 + x_1^2(t)} + 1\right) \sin x_2(t)$$

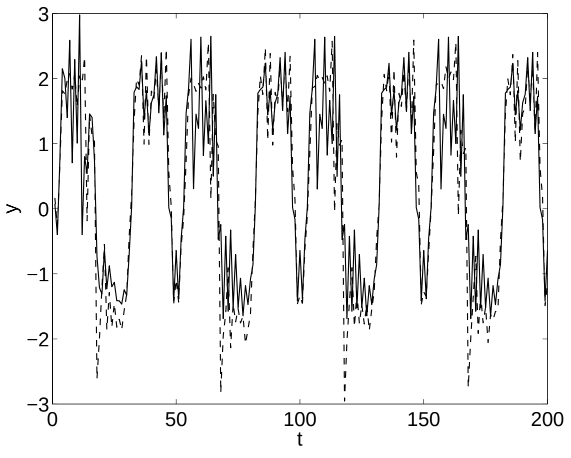


Figure 4.2: Simulated (solid) and true (dashed) output for system (4.16), modelled using the DWO approach with $L = 0.1$.

$$\begin{aligned} x_2(t+1) &= x_2(t)\cos x_2(t) + x_1(t)e^{-\frac{x_1^2(t)+x_2^2(t)}{8}} \\ &\quad + \frac{u^3(t)}{1+u^2(t)+0.5\cos(x_1(t)+x_2(t))} \\ y(t) &= \frac{x_1(t)}{1+0.5\sin x_2(t)} + \frac{x_2(t)}{1+0.5\sin x_1(t)} + e(t) \end{aligned} \tag{4.16}$$

The noise term $e(t)$ is added in accordance with [146] and has a variance of 0.1. The states are assumed not to be measurable, and following the discussion in [146], a NARX331 structure is used to model the system, i.e.,

$$\varphi(t) = (y(t-1)\ y(t-2)\ y(t-3)\ u(t-1)\ u(t-2)\ u(t-3))^T$$

As estimation data, $N = 50000$ samples were generated using a uniformly distributed random input $u(t) \in [-2.5, 2.5]$. To validate the model, the input signal

$$u(t) = \sin\frac{2\pi t}{10} + \sin\frac{2\pi t}{25}, \quad t = 1, \dots, 200$$

was used. Figure 4.2 shows the simulated output when L was chosen to be 0.1. The results are reasonable (49.7% fit), although it should be noted that the Lipschitz constant is not known a priori, and is chosen ad hoc to be constant over the entire state-space. In fact, since the real system is not of NARX structure, there might not even exist such a Lipschitz constant. Therefore, combining the approach with a local estimation of L using an algorithm similar to the bandwidth selection methods in, e.g., [146], would probably improve the results.

5

Using Prior Knowledge

One can easily imagine situations when something is known in advance about the function to estimate. For instance, we might have a rough idea of what would be reasonable values of the function from earlier experience, or physical limits might tell us, e.g., that the function only can take positive values. Furthermore, we might know that the rate of change is limited by some known constant.

If such bounds are known, it would be tempting to try to use them to improve the estimate. One could then remove the constraints (3.7) (which were imposed to get rid of the influence of the values $f(\varphi_0)$ and $f'(\varphi_0)$ on the estimate), and instead consider a restricted family of functions with some prior knowledge of the function value and its derivative:

$$|f(\varphi_0) - a| \leq \delta, \qquad |f'(\varphi_0) - b| \leq \Delta \tag{5.1}$$

It is easy to see that, provided that the assumptions (5.1) are correct, this will not make the worst-case MSE worse (we could always use weights satisfying (3.7), even if it is not required). On the other hand, if the bounds are very wide, they will probably not be of much help, either. Such issues will be discussed in this chapter.

Section 5.1 starts with the univariate case. For simplicity, we only consider the class $\mathcal{F}_2(L)$. In Section 5.2, the extension to multivariate functions is considered.

5.1 The Univariate Case

The basic problem is still the same: estimate the value $f(\varphi_0)$ of an unknown function $f : \mathbb{R} \to \mathbb{R}$ at a given point φ_0, given a set of input-output pairs $\{(\varphi(k), y(k))\}_{k=1}^{N}$, coming from

$$y(k) = f(\varphi(k)) + e(k) \tag{5.2}$$

We assume that f is continuously differentiable, and that there are known positive constants L, δ, Δ, and known constants a, b such that

$$|f'(\varphi(1)) - f'(\varphi(2))| \le L|\varphi(1) - \varphi(2)| \quad \forall\, \varphi(1), \varphi(2) \in \mathbb{R} \tag{5.3}$$

$$|f(\varphi_0) - a| \le \delta \tag{5.4}$$

$$|f'(\varphi_0) - b| \le \Delta \tag{5.5}$$

Following the notation of Chapter 2, this class of functions is denoted by $\mathcal{F}_2(L, \delta, \Delta)$. The noise terms are as previously independent, identically distributed random variables with zero mean and known variance σ^2.

There are some particular cases that deserve special attention:

- If $\delta \to \infty$ then the limit class

  $$\mathcal{F}_2(L, \Delta) \triangleq \mathcal{F}_2(L, \Delta, \infty) = \bigcup_{t=1}^{\infty} \mathcal{F}_2(L, \delta, \Delta)|_{\delta=t} \tag{5.6}$$

  describes the situation where we have no direct *a priori* information on the function value $f(\varphi_0)$.

- If in addition $\Delta \to 0^+$, then the limit class $\mathcal{F}_2(L, 0)$ represents a set of functions meeting condition (5.3) and having a given derivative $f'(\varphi_0) = b$.

- If both $\delta \to \infty$ and $\Delta \to \infty$, we are back to the function class $\mathcal{F}_2(L)$ studied in Chapter 3.

In the previous chapters, we have used a linear estimator (2.3). For the current function classes, it turns out to be useful to consider the slightly more general class of *affine estimators*:

$$\hat{f}(\varphi_0) = w_0 + \sum_{k=1}^{N} w_k y(k) \tag{5.7}$$

As for the linear estimators in previous chapters, the weights of the affine estimator will depend on φ_0, $\varphi(k)$, L, and σ, but not on $y(k)$. Furthermore, they may also depend on a, b, δ, and Δ.

The performance of an estimator $\hat{f}(\varphi_0)$ will, as before, be evaluated by an upper bound on the worst-case MSE. In the following, such bounds are given for the different function classes defined above. As it turns out, these upper bounds can then be minimized using quadratic programming, yielding optimal (in this sense) estimators.

5.1.1 Class $\mathcal{F} = \mathcal{F}_2(L, \delta, \Delta)$

As suggested, let us consider the affine estimator (5.7) and the function class $\mathcal{F}_2(L, \delta, \Delta)$ for finite δ, Δ. For this estimator and class, the worst-case MSE has the following upper bound:

$$\begin{aligned} &\mathit{WMSE}(\hat{f}(\varphi_0), \mathcal{F}_2(L, \delta, \Delta)) \le U_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w) \\ &\quad= \left(\left| w_0 + a \left(\sum_{k=1}^{N} w_k - 1 \right) + b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right| + \delta \left| \sum_{k=1}^{N} w_k - 1 \right| \right. \\ &\qquad \left. + \Delta \left| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{5.8}$$

This is true, since for any function $f \in \mathcal{F}_2(L, \delta, \Delta)$ the estimation error may be represented as follows

$$\begin{aligned} \hat{f}(\varphi_0) - f(\varphi_0) &= w_0 + \sum_{k=1}^{N} w_k y(k) - f(\varphi_0) \\ &= w_0 + \sum_{k=1}^{N} w_k (f(\varphi(k)) + e(k)) - f(\varphi_0) \\ &= w_0 + a \left(\sum_{k=1}^{N} w_k - 1 \right) + b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \\ &\quad + (f(\varphi_0) - a) \left(\sum_{k=1}^{N} w_k - 1 \right) + (f'(\varphi_0) - b) \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \\ &\quad + \sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0) \widetilde{\varphi}(k)) + \sum_{k=1}^{N} w_k e(k) \end{aligned} \tag{5.9}$$

Due to Lemma A.3, the inequality

$$|f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0) \widetilde{\varphi}(k)| \le \frac{L}{2} \widetilde{\varphi}^2(k) \tag{5.10}$$

follows from (5.3). Now, the MSE satisfies

$$\begin{aligned} \mathit{MSE}(\hat{f}(\varphi_0)) = &\left(w_0 + a \left(\sum_{k=1}^{N} w_k - 1 \right) + b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right. \\ &+ (f(\varphi_0) - a) \left(\sum_{k=1}^{N} w_k - 1 \right) + (f'(\varphi_0) - b) \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \\ &\left. + \sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0) \widetilde{\varphi}(k)) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{5.11}$$

$$\leq \Bigg(\left| w_0 + a\left(\sum_{k=1}^N w_k - 1\right) + b\sum_{k=1}^N w_k \widetilde{\varphi}(k) \right| \tag{5.12}$$
$$+ |f(\varphi_0) - a| \cdot \left| \sum_{k=1}^N w_k - 1 \right| + |f'(\varphi_0) - b| \cdot \left| \sum_{k=1}^N w_k \widetilde{\varphi}(k) \right|$$
$$+ \sum_{k=1}^N |w_k| \cdot |f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k)| \Bigg)^2 + \sigma^2 \sum_{k=1}^N w_k^2$$

from which the upper bound (5.8) follows directly.

Note that the upper bound $U_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)$ is easily minimized with respect to w_0 for any $w \in \mathbb{R}^N$. Indeed,

$$\operatorname*{argmin}_{w_0} U_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w) = -a\left(\sum_{k=1}^N w_k - 1\right) - b\sum_{k=1}^N w_k \widetilde{\varphi}(k) \tag{5.13}$$

Thus, we arrive at the following theorem:

Theorem 5.1
For the function class $\mathcal{F}_2(L, \delta, \Delta)$, the affine estimator minimizing $U_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)$ is found among the estimators satisfying

$$\hat{f}(\varphi_0) = \sum_{k=1}^N w_k y(k) - a\left(\sum_{k=1}^N w_k - 1\right) - b\sum_{k=1}^N w_k \widetilde{\varphi}(k) \tag{5.14}$$
$$= a + \sum_{k=1}^N w_k \left(y(k) - a - b\widetilde{\varphi}(k)\right), \qquad w \in \mathbb{R}^N$$

For this kind of estimators, the worst-case MSE has the following upper bound:

$$WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L, \delta, \Delta)) \leq U_{\mathcal{F}_2(L,\delta,\Delta)}(w) \tag{5.15}$$
$$= \left(\delta \left| \sum_{k=1}^N w_k - 1 \right| + \Delta \left| \sum_{k=1}^N w_k \widetilde{\varphi}(k) \right| + \frac{L}{2} \sum_{k=1}^N |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^N w_k^2$$

Proof Follows directly from (5.8) and (5.13). □

The second expression of (5.14) can be interpreted as if we use the *a priori* given values a and b and form an affine nominal model around φ_0:

$$\hat{y}(k) = a + b\widetilde{\varphi}(k)$$

Then the residuals $y(k) - \hat{y}(k)$ are used to improve the estimated function value a from the nominal model. How large effect the residuals should have depends on the weights obtained by minimizing (5.15), which in turn depends on Δ and δ, i.e., how uncertain the nominal values a and b are.

Let us take a closer look at (5.11). It is easy to see, that if $\delta \to \infty$, the MSE might be arbitrarily large unless $\sum_{k=1}^{N} w_k = 1$, since the term

$$(f(\varphi_0) - a)\left(\sum_{k=1}^{N} w_k - 1\right)$$

is unbounded. In fact, we can show the following interesting theorem:

Theorem 5.2
Assume that $\widetilde{\varphi}(k) \neq 0$, $k = 1, \ldots, N$. Given $a, b \in \mathbb{R}$ and $L, \Delta \in (0, \infty)$, there exists a $\delta_0 \in (0, \infty)$ such that for any $\delta > \delta_0$, the minimum of the upper bound $U_{\mathcal{F}_2(L,\delta,\Delta)}(w)$ given by (5.15) with respect to $w \in \mathbb{R}^N$ is attained on the subspace

$$\sum_{k=1}^{N} w_k = 1 \tag{5.16}$$

and does not depend on a or δ. In other words, given a sufficiently large δ, the affine estimator (5.7) minimizing $U_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)$ can be found in the form

$$\hat{f}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k)\,, \quad \sum_{k=1}^{N} w_k = 1 \tag{5.17}$$

with the weights w_k minimizing the simpler upper bound

$$U_{\mathcal{F}_2(L,\Delta)}(w) = \left(\Delta \left|\sum_{k=1}^{N} w_k \widetilde{\varphi}(k)\right| + \frac{L}{2}\sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k)\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \tag{5.18}$$

subject to the constraint (5.16).

Proof We defer the proof until Section 5.1.6. □

Theorem 5.2 implies that there is a limit (given by δ_0) for the uncertainty in the information about $f(\varphi_0)$, above which this information does not improve the quality of the estimate at all. However, the estimate naturally does not get worse, either.

5.1.2 Class $\mathcal{F} = \mathcal{F}_2(L, \Delta)$

Let us now turn to class $\mathcal{F}_2(L, \Delta)$, i.e., the case $\delta \to \infty$. From the remark just before Theorem 5.2, it follows that the MSE cannot be bounded above unless $\sum_{k=1}^{N} w_k = 1$. On the other hand, if this requirement is satisfied, we get the following upper bound on the worst-case MSE, which can be shown analogously to (5.8):

$$\begin{aligned} WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L,\Delta)) &\leq U_{\mathcal{F}_2(L,\Delta)}(w_0, w) \qquad (5.19) \\ &= \left(\left| w_0 + b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right| + \Delta \left| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned}$$

By minimizing the upper bound with respect to w_0, it can be seen that the minimizing estimator will be in the form (5.17), and that w can be found by minimizing (5.18) under assumption (5.16). We summarize this in the following theorem:

Theorem 5.3
For the function class $\mathcal{F}_2(L,\Delta)$, the affine estimator minimizing $U_{\mathcal{F}_2(L,\Delta)}(w_0, w)$ is found among the estimators satisfying

$$\hat{f}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = \sum_{k=1}^{N} w_k \left(y(k) - b\widetilde{\varphi}(k) \right) , \quad \sum_{k=1}^{N} w_k = 1 \qquad (5.20)$$

For this kind of estimators, the worst-case MSE has the following upper bound:

$$\begin{aligned} WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L,\Delta)) &\leq U_{\mathcal{F}_2(L,\Delta)}(w) \qquad (5.21) \\ &= \left(\Delta \left| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned}$$

Theorem 5.3 can be interpreted in a similar way as Theorem 5.1: We form a nominal affine model around φ_0, and use the residuals $y(k) - \hat{y}(k)$ to adjust the estimate $\hat{f}(\varphi_0)$. However, since we do not know anything about a nominal function value a, we should select the weights w_k in such a way that the effect of a is cancelled in the expression of $\hat{f}(\varphi_0)$. This is done as in (5.20), i.e., by choosing weights that satisfy $\sum_{k=1}^{N} w_k = 1$.

Analogously to Theorem 5.2, we can study what happens when Δ is large.

Theorem 5.4
Suppose that $\widetilde{\varphi}(k) \neq 0$, $k = 1, \ldots, N$, and that there are two indices k_1 and k_2 such that $\widetilde{\varphi}(k_1) \neq \widetilde{\varphi}(k_2)$. Given $b \in \mathbb{R}$ and $L \in (0, \infty)$, there exists a $\Delta_0 \in (0, \infty)$ such that for any $\Delta > \Delta_0$, the minimum of the upper bound $U_{\mathcal{F}_2(L,\Delta)}(w)$ given by (5.21)*, subject to the constraint* (5.16)*, is attained on the subspace*

$$\sum_{k=1}^{N} w_k = 1 \, , \quad \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 0 \qquad (5.22)$$

and does not depend on b or Δ. In other words, given a sufficiently large Δ, the affine estimator (5.7) *minimizing $U_{\mathcal{F}_2(L,\Delta)}(w_0, w)$ can be found in the form*

$$\hat{f}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) \qquad (5.23)$$

by minimizing the upper bound

$$U_{\mathcal{F}_2(L)}(w) = \left(\frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \tag{5.24}$$

subject to constraints (5.22).

Proof See Section 5.1.6. □

Remark 5.1 *Note that Theorem 5.4 does not hold in general if $\widetilde{\varphi}(k) = 0$ for some k. For example, if $\widetilde{\varphi}(1) > 0$ and $\widetilde{\varphi}(k) = 0$ for $k = 2, \ldots, N$, we get the following optimal solution:*

$$\begin{aligned} w_1^* &= \frac{\sigma^2}{(N-1)\widetilde{\varphi}^2(1)\left(\Delta + \frac{L}{2}\widetilde{\varphi}(1)\right)^2 + N\sigma^2} \\ w_k^* &= \frac{\widetilde{\varphi}^2(1)\left(\Delta + \frac{L}{2}\widetilde{\varphi}(1)\right)^2 + \sigma^2}{(N-1)\widetilde{\varphi}^2(1)\left(\Delta + \frac{L}{2}\widetilde{\varphi}(1)\right)^2 + N\sigma^2}, \qquad k = 2, \ldots, N \end{aligned} \tag{5.25}$$

We can see that $w_1^* \to 0$ as $\Delta \to \infty$, but for any finite Δ, w_1^* is positive. Furthermore,

$$\sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) = w_1^* \widetilde{\varphi}(1) > 0$$

5.1.3 Class $\mathcal{F} = \mathcal{F}_2(L, 0)$

We may now formally let $\Delta \to 0^+$ in the previous subsection, which means that the derivative $f'(\varphi_0) = b$ is *a priori* known. We obtain the following results as a direct consequence of what was stated there.

Corollary 5.1
For the function class $\mathcal{F}_2(L, 0)$, a finite worst-case MSE can be guaranteed only if the requirement (5.16) *is satisfied. Under this constraint, we get the following upper bound on the worst-case MSE:*

$$\begin{aligned} WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L,0)) &\le U_{\mathcal{F}_2(L,0)}(w_0, w) \\ &= \left(\left| w_0 + b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{5.26}$$

Hence, the affine estimator minimizing $U_{\mathcal{F}_2(L,0)}(w_0, w)$ is found among the estimators satisfying

$$\hat{f}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - b \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = \sum_{k=1}^{N} w_k \left(y(k) - b\widetilde{\varphi}(k) \right), \quad \sum_{k=1}^{N} w_k = 1 \tag{5.27}$$

For this kind of estimators, the worst-case MSE has the following upper bound:

$$\begin{aligned} \mathit{WMSE}(\hat{f}(\varphi_0), \mathcal{F}_2(L,0)) &\le U_{\mathcal{F}_2(L)}(w) \\ &= \left(\frac{L}{2}\sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k)\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{5.28}$$

Proof Follows directly from (5.19) and Theorem 5.3. □

As we can see from (5.27), the benefit of knowing the derivative $f'(\varphi_0)$ depends on the value of $\sum_{k=1}^{N} w_k \widetilde{\varphi}(k)$. If this value is small, the effect from knowing the derivative is small. For instance, if the data samples are lying symmetrically around φ_0, it turns out that the weights w_k will also be symmetric, and thus

$$\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 0$$

In this case, knowing $f'(\varphi_0)$ will not affect the function estimate at all. If, on the other hand, all $\widetilde{\varphi}(k) > 0$, the value of this sum will be relatively large, since all terms $w_k \widetilde{\varphi}(k)$ will be nonnegative (it is easy to see that the weights w_k that minimize (5.28) under the constraint (5.16) will be nonnegative, regardless of the values of $\widetilde{\varphi}(k)$). Hence, the knowledge of $f'(\varphi_0)$ seems to be more valuable the more asymmetrically spread the data samples are.

5.1.4 Class $\mathcal{F} = \mathcal{F}_2(L)$

For the class $\mathcal{F}_2(L)$, following a similar line of argument as for $\mathcal{F}_2(L, \Delta)$, we can see that a finite MSE can be guaranteed only if the weights w satisfy (5.22). Under this requirement, on the other hand, we get the following upper bound on the worst-case MSE:

$$\begin{aligned} \mathit{WMSE}(\hat{f}(\varphi_0), \mathcal{F}_2(L)) &\le U_{\mathcal{F}_2(L)}(w_0, w) \\ &= \left(|w_0| + \frac{L}{2}\sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k)\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{5.29}$$

Hence, by minimizing the upper bound with respect to w_0 (which of course will yield $w_0 = 0$), we will obtain a minimizing estimator in the form (5.23), where w can be found by minimizing (5.24) subject to the constraints (5.22). We are thus back to the situation in Chapter 3.

5.1.5 QP Formulations

In the Sections 5.1.1-5.1.4, it was pointed out how the weights w_0 and w_k of the affine estimator (5.7) could be chosen by minimizing different expressions, in order to get estimators with guaranteed upper bounds on the worst-case MSE. In this section we will show that these minimization problems can be formulated as convex quadratic programs.

To begin with, let us consider the function class $\mathcal{F}_2(L,\delta,\Delta)$ and the problem of finding the affine estimator minimizing (5.15).

Theorem 5.5
Given the positive numbers δ, Δ, L, consider the following minimization problem:

$$\begin{aligned}
\min_{w,s} \quad & \left(\delta s_a + \Delta s_b + \frac{L}{2}\sum_{k=1}^{N}\widetilde{\varphi}^2(k)s_k\right)^2 + \sigma^2\sum_{k=1}^{N}s_k^2 \\
\text{subj. to} \quad & s_a \geq \pm\left(\sum_{k=1}^{N}w_k - 1\right) \\
& s_b \geq \pm\sum_{k=1}^{N}w_k\widetilde{\varphi}(k) \\
& s_k \geq \pm w_k, \qquad k=1,\dots,N
\end{aligned} \tag{5.30}$$

where $s=(s_a,s_b,s_1,\dots,s_N)$. Then w^* is a minimizer of $U_{\mathcal{F}_2(L,\delta,\Delta)}(w)$ as defined in (5.15) if and only if there is a vector s^* such that (w^*,s^*) is a minimizer of (5.30). Furthermore, the following relations hold:

$$\begin{aligned}
s_a^* &= \left|\sum_{k=1}^{N}w_k^* - 1\right| \\
s_b^* &= \left|\sum_{k=1}^{N}w_k^*\widetilde{\varphi}(k)\right| \\
s_k^* &= |w_k^*|, \qquad k=1,\dots,N
\end{aligned} \tag{5.31}$$

Proof Given a feasible solution w to (5.15), we can get a feasible solution to (5.30) with the same value of the objective function by using the same w and

$$\begin{aligned}
s_a &= \left|\sum_{k=1}^{N}w_k - 1\right| \\
s_b &= \left|\sum_{k=1}^{N}w_k\widetilde{\varphi}(k)\right| \\
s_k &= |w_k|, \qquad k=1,\dots,N
\end{aligned} \tag{5.32}$$

Hence (5.30) is a relaxation of (5.15), and it suffices to show that for a minimizer (w^*,s^*) of (5.30), (5.32) will hold. Suppose, e.g., that $s_1^* > |w_1^*|$. Then, without changing any other variables, the value of the objective function can be reduced by decreasing s_1^*. This can be seen by observing that the coefficient before s_1^* is nonnegative in the first sum of the objective function, and positive in the second sum of the objective function, so decreasing s_1^* will decrease at least one of these sums, and hence the objective function. Hence, $s_1^* = |w_1^*|$. By similar arguments, one can show that the other equalities of (5.32) will also hold at the optimum, which proves the theorem. □

Note that (5.30) is a convex QP and can therefore be solved efficiently.

Starting from Theorem 5.5, we can now formulate QP:s for all the other cases mentioned in Section 5.1. Since the constraints (5.16) and (5.22) are all linear in w, they can just be added to the QP. For the function class $\mathcal{F}_2(L)$, the corresponding QP was given in Theorem 3.1. For $\mathcal{F}_2(L, \Delta)$, the resulting theorem is listed below.

Theorem 5.6
Given the positive numbers Δ, L, consider the following minimization problems:

$$\begin{aligned} \min_{w} \quad & \left(\Delta \left| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \\ \text{subj. to} \quad & \sum_{k=1}^{N} w_k = 1 \end{aligned} \tag{5.33}$$

and

$$\begin{aligned} \min_{w,s} \quad & \left(\Delta s_b + \frac{L}{2} \sum_{k=1}^{N} \widetilde{\varphi}^2(k) s_k \right)^2 + \sigma^2 \sum_{k=1}^{N} s_k^2 \\ \text{subj. to} \quad & s_b \geq \pm \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \\ & s_k \geq \pm w_k, \qquad k = 1, \dots, N \\ & \sum_{k=1}^{N} w_k = 1 \end{aligned} \tag{5.34}$$

where $w = (w_1, \dots, w_N)$ and $s = (s_b, s_1, \dots, s_N)$. Then w^* is a minimizer of (5.33) *if and only if there is a vector s^* such that (w^*, s^*) is a minimizer of* (5.34)*. Furthermore, the following relations hold:*

$$\begin{aligned} s_b^* &= \left| \sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) \right| \\ s_k^* &= |w_k^*|, \qquad k = 1, \dots, N \end{aligned} \tag{5.35}$$

5.1.6 Proofs of Theorems 5.2 and 5.4

Now, when the minimization problems have been reformulated as QP:s, we are ready to prove Theorems 5.2 and 5.4.

Proof (Proof of Theorem 5.4) From Theorem 5.6 we know that minimizing (5.21) subject to the constraint (5.16) is equivalent to solving (5.34). We also note that the optimization problem we get by adding

$$\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 0$$

as a constraint to (5.34) is nothing else than (3.11), and will yield a (finite) optimal value if we have at least two distinctive points $\widetilde{\varphi}(k)$. Call this value d.

The Lagrangian function of (5.34) can be written

$$\begin{aligned}\mathcal{L}(w, s; \mu, \lambda) = &\left(\Delta s_b + \frac{L}{2}\sum_{k=1}^{N} \widetilde{\varphi}^2(k) s_k\right)^2 + \sigma^2 \sum_{k=1}^{N} s_k^2 - \mu\left(\sum_{k=1}^{N} w_k - 1\right) \\ &- \lambda_b^+ \left(s_b - \sum_{k=1}^{N} w_k \widetilde{\varphi}(k)\right) - \lambda_b^- \left(s_b + \sum_{k=1}^{N} w_k \widetilde{\varphi}(k)\right) \\ &- \sum_{k=1}^{N} (\lambda_k^+ (s_k - w_k) + \lambda_k^- (s_k + w_k))\end{aligned} \tag{5.36}$$

where $\lambda_k^\pm \geq 0$, $k = 1, \ldots, N$, $\lambda_b^\pm \geq 0$, and μ are the Lagrangian multipliers. Let us also introduce

$$g(w, \Delta) = 2\left(\Delta \left|\sum_{k=1}^{N} w_k \widetilde{\varphi}(k)\right| + \frac{L}{2}\sum_{k=1}^{N} \widetilde{\varphi}^2(k)|w_k|\right) \tag{5.37}$$

Note that

$$g(w, \Delta) \geq L \min_k \widetilde{\varphi}^2(k) > 0 \tag{5.38}$$

Let (w^*, s_b^*, s^*) be an optimal solution of (5.34). Using the relations

$$s_b^* = \left|\sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k)\right|, \qquad s_k^* = |w_k^*|$$

together with the necessary KKT conditions, we get

$$\begin{aligned}&g(w^*, \Delta)\Delta = \lambda_b^+ + \lambda_b^- \\ &\mu + (\lambda_b^- - \lambda_b^+)\widetilde{\varphi}(k) = \lambda_k^+ - \lambda_k^- \\ &g(w^*, \Delta)\frac{L}{2}\widetilde{\varphi}^2(k) + 2\sigma^2|w_k^*| = \lambda_k^+ + \lambda_k^-\end{aligned}$$

$$\begin{aligned}
&\sum_{k=1}^{N} w_k^* = 1\\
&\lambda_b^+ \left(\left| \sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) \right| - \sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) \right) = 0\\
&\lambda_b^- \left(\left| \sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) \right| + \sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) \right) = 0\\
&\lambda_k^+ (|w_k^*| - w_k^*) = 0\\
&\lambda_k^- (|w_k^*| + w_k^*) = 0\\
&\lambda_b^\pm \geq 0, \quad \lambda_k^\pm \geq 0, \quad k = 1, \dots, N
\end{aligned} \tag{5.39}$$

Now assume, e.g., that $\sum_{k=1}^{N} w_k^* \widetilde{\varphi}(k) > 0$. This means that $\lambda_b^- = 0$. If also $w_k^* > 0$ for some k, this implies that $\lambda_k^- = 0$. By elimination of λ_b^+ and λ_k^+, we then get

$$2\sigma^2 w_k^* = \mu - g(w^*, \Delta) \left(\frac{L}{2} \widetilde{\varphi}^2(k) + \Delta \widetilde{\varphi}(k) \right)$$

If on the other hand $w_k^* < 0$, we get $\lambda_k^+ = 0$ and

$$2\sigma^2 w_k^* = \mu - g(w^*, \Delta) \left(-\frac{L}{2} \widetilde{\varphi}^2(k) + \Delta \widetilde{\varphi}(k) \right)$$

Finally, if $w_k^* = 0$, this yields

$$\begin{cases} 2\lambda_k^+ = \mu - g(w^*, \Delta) \left(-\frac{L}{2} \widetilde{\varphi}^2(k) + \Delta \widetilde{\varphi}(k) \right) \\ 2\lambda_k^- = -\mu + g(w^*, \Delta) \left(\frac{L}{2} \widetilde{\varphi}^2(k) + \Delta \widetilde{\varphi}(k) \right) \end{cases}$$

which, since $\lambda_k^\pm \geq 0$, implies

$$-g(w^*, \Delta) \frac{L}{2} \widetilde{\varphi}^2(k) \leq \mu - g(w^*, \Delta) \Delta \widetilde{\varphi}(k) \leq g(w^*, \Delta) \frac{L}{2} \widetilde{\varphi}^2(k)$$

We can summarize the last four expressions in the following form (where we define $\widetilde{\varphi} = (\widetilde{\varphi}(1) \ \dots \ \widetilde{\varphi}(N))^T$):

$$2\sigma^2 \begin{pmatrix} w_1^* \\ \vdots \\ w_N^* \end{pmatrix} = \mu \underbrace{\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}}_{v_1} - g(w^*, \Delta) \left(\frac{L}{2} \underbrace{\begin{pmatrix} \widetilde{\varphi}^2(1) \operatorname{sgn}_{p_1} w_1^* \\ \vdots \\ \widetilde{\varphi}^2(N) \operatorname{sgn}_{p_N} w_N^* \end{pmatrix}}_{v_2} + \Delta \widetilde{\varphi} \right) \tag{5.40}$$

for some $p_k \in [-1, 1]$, $k = 1, \dots, N$, where sgn_p is defined as

$$\operatorname{sgn}_p(t) = \begin{cases} 1 & t > 0 \\ p & t = 0 \\ -1 & t < 0 \end{cases} \tag{5.41}$$

Note that the Euclidean norm of the vector v_2 in (5.40) is bounded by

$$\|v_2\| \leq \left\| \begin{pmatrix} \widetilde{\varphi}^2(1) \\ \vdots \\ \widetilde{\varphi}^2(N) \end{pmatrix} \right\| \triangleq M$$

Let

$$\Delta_0 = \frac{\frac{L}{2}M + \frac{2\sigma\sqrt{d}}{L \min_k \widetilde{\varphi}^2(k)}}{\sqrt{\sum_{k=1}^N \left(\widetilde{\varphi}(k) - \frac{1}{N}\sum_{i=1}^N \widetilde{\varphi}(i)\right)^2}} \tag{5.42}$$

Then, for $\Delta > \Delta_0$,

$$\begin{aligned} \|w^*\| &= \frac{g(w^*,\Delta)}{2\sigma^2} \left\| \frac{\mu}{g(w^*,\Delta)} v_1 - \frac{L}{2} v_2 - \Delta\widetilde{\varphi} \right\| \\ &\geq \frac{L \min_k \widetilde{\varphi}^2(k)}{2\sigma^2} \left(\left\| \frac{\mu}{g(w^*,\Delta)} v_1 - \Delta\widetilde{\varphi} \right\| - \frac{L}{2}\|v_2\| \right) \\ &\geq \frac{L \min_k \widetilde{\varphi}^2(k)}{2\sigma^2} \left(\Delta \left\| \frac{\widetilde{\varphi}^T v_1}{\|v_1\|^2} v_1 - \widetilde{\varphi} \right\| - \frac{L}{2}M \right) \\ &= \frac{L \min_k \widetilde{\varphi}^2(k)}{2\sigma^2} \left(\Delta \left\| \widetilde{\varphi} - \frac{1}{N}\sum_{k=1}^N \widetilde{\varphi}(k) v_1 \right\| - \frac{L}{2}M \right) \\ &= \frac{L \min_k \widetilde{\varphi}^2(k)}{2\sigma^2} \left(\Delta \sqrt{\sum_{k=1}^N \left(\widetilde{\varphi}(k) - \frac{1}{N}\sum_{i=1}^N \widetilde{\varphi}(i) \right)^2} - \frac{L}{2}M \right) > \frac{\sqrt{d}}{\sigma} \end{aligned} \tag{5.43}$$

In the third row of (5.43), we have used the fact that the shortest distance from $\widetilde{\varphi}$ and the line spanned by v_1 is between $\widetilde{\varphi}$ and its orthogonal projection onto v_1 (see Figure 5.1). Comparing (5.43) with the objective function in (5.33), we can see that the last term alone of the objective function (and hence of the objective function in (5.34)) will be larger than d. This leads to a contradiction, and we conclude that $\sum_{k=1}^N w_k^* \widetilde{\varphi}(k) \leq 0$. The case $\sum_{k=1}^N w_k^* \widetilde{\varphi}(k) < 0$ is treated analogously, which implies that if $\Delta > \Delta_0$ as given by (5.42), then $\sum_{k=1}^N w_k^* \widetilde{\varphi}(k) = 0$. The proof is complete. □

Proof (Proof of Theorem 5.2) From Theorem 5.5 we know, that minimizing (5.15) is equivalent to solving (5.30). We also note that the optimization problem we get by adding

$$\sum_{k=1}^N w_k = 1$$

as a constraint to (5.30) is nothing else than (5.34), and will yield a finite optimal value. Call this value d.

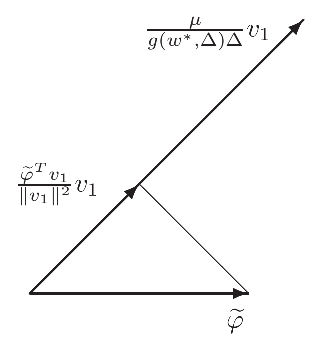

$\frac{\mu}{g(w^*,\Delta)\Delta} v_1$
$\frac{\widetilde{\varphi}^T v_1}{\|v_1\|^2} v_1$
$\widetilde{\varphi}$

Figure 5.1: Illustration of one of the inequalities in (5.43).

The Lagrangian function of (5.30) can be written

$$\begin{aligned}\mathcal{L}(w, s; \mu, \lambda) = &\left(\delta s_a + \Delta s_b + \frac{L}{2}\sum_{k=1}^{N}\widetilde{\varphi}^2(k)s_k\right)^2 + \sigma^2\sum_{k=1}^{N}s_k^2\\ &- \lambda_a^+\left(s_a - \sum_{k=1}^{N}w_k + 1\right) - \lambda_a^-\left(s_a + \sum_{k=1}^{N}w_k - 1\right)\\ &- \lambda_b^+\left(s_b - \sum_{k=1}^{N}w_k\widetilde{\varphi}(k)\right) - \lambda_b^-\left(s_b + \sum_{k=1}^{N}w_k\widetilde{\varphi}(k)\right)\\ &- \sum_{k=1}^{N}(\lambda_k^+(s_k - w_k) + \lambda_k^-(s_k + w_k))\end{aligned} \tag{5.44}$$

where $\lambda_k^\pm \geq 0$, $k = 1, \ldots, N$, $\lambda_a^\pm \geq 0$, and $\lambda_b^\pm \geq 0$ are the Lagrangian multipliers. Let us also introduce

$$g(w, \delta, \Delta) = 2\left(\delta\left|\sum_{k=1}^{N}w_k - 1\right| + \Delta\left|\sum_{k=1}^{N}w_k\widetilde{\varphi}(k)\right| + \frac{L}{2}\sum_{k=1}^{N}\widetilde{\varphi}^2(k)|w_k|\right) \tag{5.45}$$

Let (w^*, s_a^*, s_b^*, s^*) be an optimal solution of (5.30). Using the relations $s_a^* = |\sum_{k=1}^{N} w_k^* - 1|$, $s_b^* = |\sum_{k=1}^{N} w_k^*\widetilde{\varphi}(k)|$, and $s_k^* = |w_k^*|$ together with the necessary KKT conditions, we get

$$\begin{aligned}&g(w^*, \delta, \Delta)\delta = \lambda_a^+ + \lambda_a^-\\ &g(w^*, \delta, \Delta)\Delta = \lambda_b^+ + \lambda_b^-\\ &g(w^*, \delta, \Delta)\frac{L}{2}\widetilde{\varphi}^2(k) + 2\sigma^2|w_k^*| = \lambda_k^+ + \lambda_k^-\\ &\lambda_a^+ - \lambda_a^- + (\lambda_b^+ - \lambda_b^-)\widetilde{\varphi}(k) + \lambda_k^+ - \lambda_k^- = 0\end{aligned}$$

$$\begin{aligned}
&\lambda_a^+ \left(\left| \sum_{k=1}^N w_k^* - 1 \right| - \sum_{k=1}^N w_k^* + 1 \right) = 0 \\
&\lambda_a^- \left(\left| \sum_{k=1}^N w_k^* - 1 \right| + \sum_{k=1}^N w_k^* - 1 \right) = 0 \\
&\lambda_b^+ \left(\left| \sum_{k=1}^N w_k^* \widetilde{\varphi}(k) \right| - \sum_{k=1}^N w_k^* \widetilde{\varphi}(k) \right) = 0 \\
&\lambda_b^- \left(\left| \sum_{k=1}^N w_k^* \widetilde{\varphi}(k) \right| + \sum_{k=1}^N w_k^* \widetilde{\varphi}(k) \right) = 0 \\
&\lambda_k^+ (|w_k^*| - w_k^*) = 0 \\
&\lambda_k^- (|w_k^*| + w_k^*) = 0 \\
&\lambda_a^\pm \geq 0, \ \lambda_b^\pm \geq 0, \ \lambda_k^\pm \geq 0, \quad k = 1, \ldots, N
\end{aligned} \tag{5.46}$$

Now assume, e.g., that $\sum_{k=1}^N w_k^* > 1$. This means that $\lambda_a^- = 0$. By computations similar to the ones leading to (5.40), we arrive at

$$2\sigma^2 w^* = -g(w^*, \delta, \Delta) \left(\delta v_1 + \Delta \operatorname{sgn}_q \left(\sum_{k=1}^N w_k^* \widetilde{\varphi}(k) \right) \widetilde{\varphi} + \frac{L}{2} v_2 \right) \tag{5.47}$$

for some $q \in [-1, 1]$, and where v_1 and v_2 are defined as in (5.40). Note also that

$$g(w, \delta, \Delta) \geq L \min_k \widetilde{\varphi}^2(k) > 0 \tag{5.48}$$

Now, let

$$\delta_0' = \frac{1}{\sqrt{N}} \left(\Delta \|\widetilde{\varphi}\| + \frac{L}{2} M + \frac{2\sigma\sqrt{d}}{L \min_k \widetilde{\varphi}^2(k)} \right) \tag{5.49}$$

Then, for $\delta > \delta_0'$,

$$\begin{aligned}
\|w^*\| &= \frac{g(w^*, \delta, \Delta)}{2\sigma^2} \left\| \delta v_1 + \Delta \operatorname{sgn}_q \left(\sum_{k=1}^N w_k^* \widetilde{\varphi}(k) \right) \widetilde{\varphi} + \frac{L}{2} v_2 \right\| \\
&\geq \frac{L \min_k \widetilde{\varphi}^2(k)}{2\sigma^2} \left(\delta \|v_1\| - \Delta \|\widetilde{\varphi}\| - \frac{L}{2} \|v_2\| \right) \\
&\geq \frac{L \min_k \widetilde{\varphi}^2(k)}{2\sigma^2} \left(\delta \sqrt{N} - \Delta \|\widetilde{\varphi}\| - \frac{L}{2} M \right) > \frac{\sqrt{d}}{\sigma}
\end{aligned} \tag{5.50}$$

which means that the last term alone of (5.30) will be larger than d. This leads to a contradiction, and we conclude that $\sum_{k=1}^N w_k^* \leq 1$. The case $\sum_{k=1}^N w_k^* < 1$ is treated analogously. However, there is one technical difficulty in this case, namely to establish a positive lower bound on $g(w, \delta, \Delta)$. If we assume, e.g., that $\delta \geq 1$, we get

$$g(w, \delta, \Delta) \geq 2 \left| \sum_{k=1}^{N} w_k - 1 \right| + L \sum_{k=1}^{N} \widetilde{\varphi}^2(k) |w_k|$$

It is easy to see that the minimum of this expression with respect to the weights w_k is obtained for weights satisfying $w_k \geq 0$, $\sum_{k=1}^{N} w_k \leq 1$. Hence, we get

$$\begin{aligned} g(w, \delta, \Delta) &\geq \min_{\substack{w_k \geq 0 \\ \sum_{k=1}^{N} w_k \leq 1}} 2 - 2\sum_{k=1}^{N} w_k + L \sum_{k=1}^{N} \widetilde{\varphi}^2(k) w_k \\ &= \min_{\substack{w_k \geq 0 \\ \sum_{k=1}^{N} w_k \leq 1}} 2 + \sum_{k=1}^{N} w_k \left(L\widetilde{\varphi}^2(k) - 2 \right) \\ &= \min\{L \min_k \widetilde{\varphi}^2(k), 2\} > 0 \end{aligned}$$

With

$$\delta_0 = \max\{1, \frac{1}{\sqrt{N}} \left(\Delta \|\widetilde{\varphi}\| + \frac{L}{2} M + \frac{2\sigma\sqrt{d}}{\min\{L \min_k \widetilde{\varphi}^2(k), 2\}} \right) \} \tag{5.51}$$

we can now perform similar calculations as for the case $\sum_{k=1}^{N} w_k > 1$, which implies that if $\delta > \delta_0$ as given by (5.51), then $\sum_{k=1}^{N} w_k^* = 1$. The proof is complete. $\square$

5.1.7 Adjusting the Estimate

For the function class $\mathcal{F}_2(L, \delta, \Delta)$, we assume that we know a bound on $f(\varphi_0)$. However, even if this was taken into account when deriving the QP (5.30), using the optimal weights from (5.30) when computing the estimate $\hat{f}(\varphi_0)$ does not guarantee that the latter falls within the bounds, i.e., that

$$|\hat{f}(\varphi_0) - a| \leq \delta \tag{5.52}$$

The reason for this is that the weights w_k do not depend on $y(k)$, and the actual noise realization might therefore in unlucky cases deteriorate $\hat{f}(\varphi_0)$ to such a degree that it falls outside the bounds.

Fortunately, this is easily handled by adjusting the estimate afterwards according to

$$\hat{f}^{ADJ}(\varphi_0) = \min\{\max\{\hat{f}(\varphi_0), a - \delta\}, a + \delta\} \tag{5.53}$$

Using this adjustment, $\hat{f}^{ADJ}(\varphi_0)$ is guaranteed to satisfy (5.52). Note that the worst-case MSE will not increase by making this alteration, if the true function $f \in \mathcal{F}_2(L, \delta, \Delta)$. A similar adjustment was done in [137] for a problem of estimating the expected value of a random variable, when some of the observations are biased.

5.2 Estimating Multivariate Functions

So far in this chapter, we have assumed that the function to be estimated has a scalar argument. However, as mentioned previously, the regressors will have a higher dimension in many applications, in particular to dynamic systems. The extension to this case is immediate. In this section, we will describe some of the aspects of this kind of extension.

We now consider the problem of estimating the value $f(\varphi_0)$ of an unknown multivariate, continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at a given point φ_0, given a set of input-output pairs $\{(\varphi(k), y(k))\}_{k=1}^N$, coming from the relation

$$y(k) = f(\varphi(k)) + e(k) \tag{5.54}$$

Instead of the assumptions (5.3)-(5.5), we make the following assumptions:

$$\begin{aligned} \|\nabla f(\varphi(1)) - \nabla f(\varphi(2))\| &\le L\|\varphi(1) - \varphi(2)\| \quad \forall\, \varphi(1), \varphi(2) \in \mathbb{R}^n & (5.55)\\ |f(\varphi_0) - a| &\le \delta & (5.56)\\ \|\nabla f(\varphi_0) - b\| &\le \Delta & (5.57) \end{aligned}$$

With some abuse of notation, we let $\mathcal{F}_2(L, \delta, \Delta)$, $\mathcal{F}_2(L, \Delta)$, and $\mathcal{F}_2(L)$ denote also their multivariate counterparts. For $\mathcal{F}_2(L, \delta, \Delta)$ and an affine estimator (5.7), the worst-case MSE has the following upper bound, which is similar to the univariate case:

$$\begin{aligned} &WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L, \delta, \Delta)) \le U_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)\\ &\quad = \left(\left| w_0 + a\left(\sum_{k=1}^N w_k - 1\right) + b^T \sum_{k=1}^N w_k \widetilde{\varphi}(k) \right| + \delta \left| \sum_{k=1}^N w_k - 1 \right| \right.\\ &\qquad \left. + \Delta \left\| \sum_{k=1}^N w_k \widetilde{\varphi}(k) \right\| + \frac{L}{2} \sum_{k=1}^N |w_k| \|\widetilde{\varphi}(k)\|^2 \right)^2 + \sigma^2 \sum_{k=1}^N w_k^2 \end{aligned} \tag{5.58}$$

As in the univariate case, we can immediately eliminate w_0 by minimizing (5.58) for an arbitrary w, giving the following theorem:

Theorem 5.7
For multivariate functions of the function class $\mathcal{F}_2(L, \delta, \Delta)$, the affine estimator minimizing $U_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)$ is found among the estimators satisfying

$$\begin{aligned} \hat{f}(\varphi_0) &= \sum_{k=1}^N w_k y(k) - a\left(\sum_{k=1}^N w_k - 1\right) - b^T \sum_{k=1}^N w_k \widetilde{\varphi}(k)\\ &= a + \sum_{k=1}^N w_k \left(y(k) - a - b^T \widetilde{\varphi}(k)\right), \qquad w \in \mathbb{R}^N \end{aligned} \tag{5.59}$$

For this kind of estimators, the worst-case MSE has the following upper bound:

$$\begin{aligned} WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L, \delta, \Delta)) &\leq U_{\mathcal{F}_2(L,\delta,\Delta)}(w) \\ &= \left(\delta \left| \sum_{k=1}^{N} w_k - 1 \right| + \Delta \left\| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right\| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \|\widetilde{\varphi}(k)\|^2 \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{5.60}$$

Proof Analogous to the univariate case. □

So far, the differences to the univariate case have been small and obvious. However, when trying to transform the problem of minimizing (5.60) into a standard convex optimization problem, it turns out that it is impossible to formulate it as a QP problem. What prohibits this is the term

$$\Delta \left\| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right\| \tag{5.61}$$

which is the norm of a linear combination of vectors. Instead, we can formulate the problem as a *second-order cone program (SOCP)*, which is another standard class of convex optimization problems (see [19]). To do this, we introduce some slack variables $s = (s_1 \ \ldots \ s_N)^T$ and $t = (t_a \ t_b \ t_c)^T$, and get

$$\begin{aligned} \min_{w,s,t} \quad & t_c \\ \text{subj. to} \quad & \left(\delta t_a + \Delta t_b + \frac{L}{2} \sum_{k=1}^{N} \|\widetilde{\varphi}(k)\|^2 s_k \right)^2 + \sigma^2 \sum_{k=1}^{N} s_k^2 \leq t_c \\ & \left| \sum_{k=1}^{N} w_k - 1 \right| \leq t_a \\ & \left\| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \right\| \leq t_b \\ & |w_k| \leq s_k, \qquad k = 1, \ldots, N \end{aligned} \tag{5.62}$$

This problem is in standard SOCP form, except for the first, quadratic constraint. However, straightforward calculations show that this constraint is equivalent to

$$\left\| \begin{pmatrix} 2 \left(\delta t_a + \Delta t_b + \frac{L}{2} \sum_{k=1}^{N} \|\widetilde{\varphi}(k)\|^2 s_k \right) \\ 2\sigma s \\ 1 - t_c \end{pmatrix} \right\| \leq 1 + t_c \tag{5.63}$$

thus completing the problem reformulation.

For the other function classes, $\mathcal{F}_2(L, \Delta)$ and $\mathcal{F}_2(L)$, the extension to the multivariate case is done completely similarly. The minimization problem for $\mathcal{F}_2(L, \Delta)$ will also yield a SOCP, while the minimization problem for $\mathcal{F}_2(L)$ will still be possible to express as a QP (as we saw in Chapter 4), since the term (5.61) vanishes.

5.3 An Example

The following simple example illustrates the fact that the information about bounds on the function value and derivatives can be useful, but only if the bounds are tight enough.

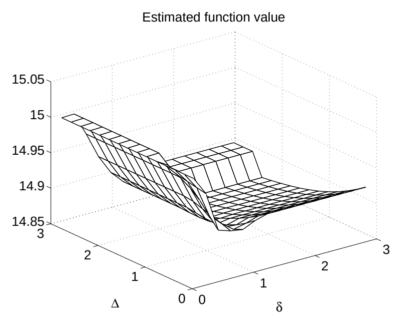
Example 5.1 *Let us consider the simple nonlinear system (with $f : \mathbb{R}^2 \to \mathbb{R}$)*

$$\begin{aligned} y(t) &= f(\varphi(t)) + e(t) \\ f(\varphi) &= 5(\varphi_1^2 + \varphi_2^2) + 5\varphi_1 + 10\varphi_2 + 15 \end{aligned} \tag{5.64}$$

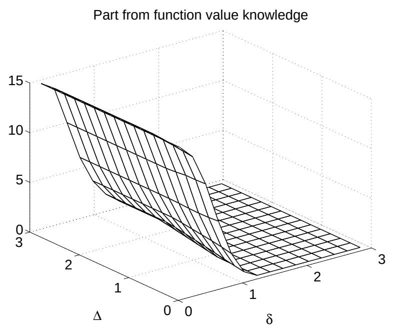
where $e(t) \in N(0, 1)$. Suppose that we would like to estimate y for $\varphi_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$, given only $N = 20$ data samples taken from the distribution $\varphi(t) \in N(0, I)$. We assume that we know that (5.55) is satisfied for $L = 10$ (this is, of course, the best possible L), and that we also know some approximate values a and b of the function $f(\varphi_0)$ and $\nabla f(\varphi_0)$, respectively, and the bounds δ and Δ according to (5.56) and (5.57). For simplicity, let a and b be the true values, $a = f(\varphi_0)$ and $b = \nabla f(\varphi_0)$.

Now we can use the estimator in (5.59), for which the appropriate weights are obtained by solving the SOCP given by (5.62) and (5.63). Solving the SOCP can be done using Yalmip *[98] and* SeDuMi *[149]. Naturally, different δ and Δ values should give different estimates. Figure 5.2(a) shows $\hat{f}(\varphi_0)$ for some different values of δ and Δ. In Figure 5.2(b) the part of the estimate coming from the* a priori *knowledge of the function value (i.e., the second term of (5.59)) is plotted. As we can see, for small values of δ, the estimate is based entirely on this information, while for large values, the* a priori *knowledge is not used at all, in agreement with Theorem 5.2. In Figure 5.2(c), the part of the $\hat{f}(\varphi_0)$ coming from the knowledge of $\nabla f(\varphi_0)$ (the third term of (5.59)) is used. For this example, we can see that this information is not used very much, but that it gives a certain contribution for small values of Δ (as long as δ is large enough, so that we do not only use the information about $f(\varphi_0)$).*

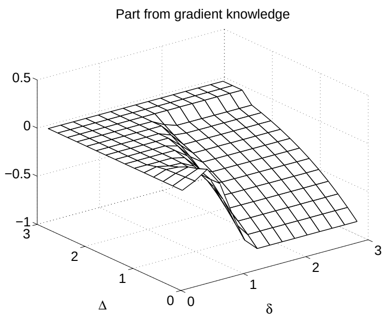
Finally, Figure 5.3 shows the optimal value of the criterion function, which decreases both with δ and Δ, just as should be expected.



(a) The estimate $\hat{f}(\varphi_0)$.



(b) The part of $\hat{f}(\varphi_0)$ coming from prior knowledge of $f(\varphi_0)$.



(c) The part of $\hat{f}(\varphi_0)$ coming from prior knowledge of $\nabla f(\varphi_0)$.

Figure 5.2: Estimates of $f(\varphi_0)$ in Example 5.1 for different values of δ and Δ.

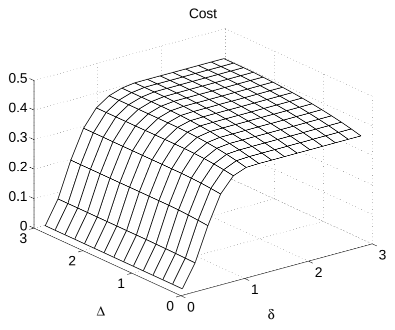


Figure 5.3: The optimal value of the criterion function in Example 5.1.

6

Other Extensions

This chapter presents some extensions that can be made to the proposed DWO approach. These extensions include estimation of the derivative, which may be useful in many applications. Section 6.2 considers the minimization of the exact worst-case MSE over $\mathcal{F}_2(L)$. Other extensions are the case of nonconstant variance, and the usage of more general function classes. These issues are described in Section 6.3.

6.1 Estimating the Derivative

In Chapter 5, we saw that prior information about the function value $f(\varphi_0)$ and its derivatives ($f'(\varphi_0)$ in the univariate case) can be of use if they are not too uncertain. The question is, given no prior information about $f(\varphi_0)$ and $f'(\varphi_0)$, if we can improve our estimate of $f(\varphi_0)$ by first estimating the derivative, and then use this estimate when estimating $f(\varphi_0)$. As we will see, the answer to this question is no for any reasonable linear estimate of $f'(\varphi_0)$, using the same given data set as for estimating $f(\varphi_0)$. However, since estimating the derivative is interesting in itself, we will first consider this topic in some more detail. We will treat the problem for the univariate case, for the three classes of functions that have been previously discussed, and use affine estimators according to (5.7).

Let us denote a general affine derivative estimator by $\widehat{f'}(\varphi_0)$. The MSE for

$\widehat{f}'(\varphi_0)$ is

$$MSE(\widehat{f}'(\varphi_0)) = E[\Big(\widehat{f}'(\varphi_0) - f'(\varphi_0)\Big)^2 |\varphi_1^N] \tag{6.1}$$

and the worst-case MSE becomes

$$WMSE(\widehat{f}'(\varphi_0), \mathcal{F}) = \sup_{f \in \mathcal{F}} MSE(\widehat{f}'(\varphi_0)) \tag{6.2}$$

Recall that for an affine estimator (5.7), its worst-case MSE (6.2) is a function of the estimator parameters w_0 and w. It also depends on the given class of functions $\mathcal{F}$. As for the function estimators, w_0 and w will be computed by minimizing an upper bound on the worst-case MSE.

6.1.1 Class $\mathcal{F} = \mathcal{F}_2(L, \delta, \Delta)$

Let us begin with the function class $\mathcal{F}_2(L, \delta, \Delta)$. Using an affine estimator (5.7), we get the following upper bound on the worst-case MSE:

$$\begin{aligned}
WMSE(\widehat{f}'(\varphi_0), \mathcal{F}_2(L,\delta,\Delta)) &\le U^1_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w) \\
= \Bigg(\Bigg|w_0 + a\sum_{k=1}^N w_k &+ b\left(\sum_{k=1}^N w_k\widetilde{\varphi}(k) - 1\right)\Bigg| + \delta\left|\sum_{k=1}^N w_k\right| \\
+ \Delta\left|\sum_{k=1}^N w_k\widetilde{\varphi}(k) - 1\right| &+ \frac{L}{2}\sum_{k=1}^N |w_k|\widetilde{\varphi}^2(k)\Bigg)^2 + \sigma^2\sum_{k=1}^N w_k^2
\end{aligned} \tag{6.3}$$

This can be seen by first considering the estimation error, which for any function $f \in \mathcal{F}_2(L, \delta, \Delta)$ may be represented as follows

$$\begin{aligned}
\widehat{f}'(\varphi_0) - f'(\varphi_0) &= w_0 + \sum_{k=1}^N w_k(f(\varphi(k)) + e(k)) - f'(\varphi_0) \\
&= w_0 + a\sum_{k=1}^N w_k + b\left(\sum_{k=1}^N w_k\widetilde{\varphi}(k) - 1\right) \\
&\quad + (f(\varphi_0) - a)\sum_{k=1}^N w_k + (f'(\varphi_0) - b)\left(\sum_{k=1}^N w_k\widetilde{\varphi}(k) - 1\right) \\
&\quad + \sum_{k=1}^N w_k(f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k)) + \sum_{k=1}^N w_k e(k)
\end{aligned} \tag{6.4}$$

Using Lemma A.3, the MSE (6.1) becomes

$$MSE(\widehat{f}'(\varphi_0)) = \left(w_0 + a\sum_{k=1}^N w_k + b\left(\sum_{k=1}^N w_k\widetilde{\varphi}(k) - 1\right)\right.$$

$$+ (f(\varphi_0) - a) \sum_{k=1}^{N} w_k + (f'(\varphi_0) - b) \left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1 \right) \tag{6.5}$$
$$+ \sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k)) \Bigg)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2$$
$$\leq \left(\left| w_0 + a \sum_{k=1}^{N} w_k + b \left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1 \right) \right| + |f(\varphi_0) - a| \cdot \left| \sum_{k=1}^{N} w_k \right| \right. \tag{6.6}$$
$$\left. + |f'(\varphi_0) - b| \cdot \left| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1 \right| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2$$

from which the upper bound (6.3) follows directly.

Similarly to the function estimation case, the upper bound $U^1_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)$ is easily minimized with respect to w_0 for any $w \in \mathbb{R}^N$, resulting in

$$\operatorname*{argmin}_{w_0} U^1_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w) = -a \sum_{k=1}^{N} w_k - b \left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1 \right) \tag{6.7}$$

Thus, we arrive at the following theorem:

Theorem 6.1
For the function class $\mathcal{F}_2(L, \delta, \Delta)$, the affine estimator minimizing $U^1_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)$ is found among the estimators satisfying

$$\widehat{f}'(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - a \sum_{k=1}^{N} w_k - b \left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1 \right) \tag{6.8}$$
$$= b + \sum_{k=1}^{N} w_k \left(y(k) - a - b\widetilde{\varphi}(k) \right)$$

For this kind of estimators, the worst-case MSE (6.2) *over the class $\mathcal{F}_2(L, \delta, \Delta)$ has the following upper bound:*

$$WMSE(\widehat{f}'(\varphi_0), \mathcal{F}_2(L, \delta, \Delta)) \leq U^1_{\mathcal{F}_2(L,\delta,\Delta)}(w) \tag{6.9}$$
$$= \left(\delta \left| \sum_{k=1}^{N} w_k \right| + \Delta \left| \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1 \right| + \frac{L}{2} \sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2$$

The interpretation of Theorem 6.1 is similar to the one of Theorem 5.1. From the *a priori* given values a and b we form an affine nominal model

$$\hat{y}(k) = a + b\widetilde{\varphi}(k)$$

The residuals of this model $y(k) - \hat{y}(k)$ are then used to adjust the given nominal value b of $f'(\varphi_0)$, and the effect of the adjustment depends on the weights, which are determined from (6.9).

When δ is large, it turns out that there is a theorem corresponding to Theorem 5.2 also for derivative estimators:

Theorem 6.2
Assume that $\widetilde{\varphi}(k) \neq 0$, $k = 1, \dots, N$. Given $a, b \in \mathbb{R}$ and $L, \Delta \in (0, \infty)$, there exists a $\delta_0 \in (0, \infty)$, such that for any $\delta > \delta_0$, the minimum of the upper bound $U^1_{\mathcal{F}_2(L,\delta,\Delta)}(w)$ given by (6.9) *with respect to $w \in \mathbb{R}^N$ is attained on the subspace*

$$\sum_{k=1}^{N} w_k = 0 \tag{6.10}$$

and does not depend on a or δ. In other words, given a sufficiently large δ, the affine derivative estimator (5.7) *minimizing $U^1_{\mathcal{F}_2(L,\delta,\Delta)}(w_0, w)$ can be found in the form*

$$\widehat{f'}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - b\left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1\right), \quad \sum_{k=1}^{N} w_k = 0 \tag{6.11}$$

with the weights w_k minimizing the simpler upper bound

$$U^1_{\mathcal{F}_2(L,\Delta)}(w) = \left(\Delta\left|\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1\right| + \frac{L}{2}\sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k)\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \tag{6.12}$$

subject to the constraint (6.10).

Proof Analogous to the proof of Theorem 5.2. □

In words this means that if the value a is too uncertain, it will not have any effect at all on the estimate $\widehat{f'}(\varphi_0)$.

6.1.2 Class $\mathcal{F} = \mathcal{F}_2(L, \Delta)$

For the class $\mathcal{F}_2(L, \Delta)$, when $\delta \to \infty$, it is easy to see that the MSE cannot have a finite upper bound unless (6.10) holds. With this constraint satisfied, the upper bound on the worst-case MSE can be written

$$\begin{aligned} WMSE(\widehat{f'}(\varphi_0), \mathcal{F}_2(L, \Delta)) &\leq U^1_{\mathcal{F}_2(L,\Delta)}(w_0, w) \\ &= \left(\left|w_0 + b\left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1\right)\right|\right. \\ &\quad \left. + \Delta\left|\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1\right| + \frac{L}{2}\sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k)\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{6.13}$$

We can minimize this bound with respect to w_0, and the resulting minimizing estimator will be in the form (6.11). It also follows that w can be computed by minimizing (6.12) subject to the constraint (6.10). This is summarized in the following theorem:

Theorem 6.3
For the function class $\mathcal{F}_2(L,\Delta)$, the affine estimator minimizing $U^1_{\mathcal{F}_2(L,\Delta)}(w_0,w)$ is found among the estimators satisfying

$$\widehat{f'}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - b\left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1\right) = b + \sum_{k=1}^{N} w_k \left(y(k) - b\widetilde{\varphi}(k)\right) \qquad (6.14)$$

$$\sum_{k=1}^{N} w_k = 0$$

For this kind of estimators, the worst-case MSE has the following upper bound:

$$\begin{aligned} WMSE(\widehat{f'}(\varphi_0), \mathcal{F}_2(L,\Delta)) &\le U^1_{\mathcal{F}_2(L,\Delta)}(w) \qquad (6.15)\\ &= \left(\Delta\left|\sum_{k=1}^{N} w_k\widetilde{\varphi}(k) - 1\right| + \frac{L}{2}\sum_{k=1}^{N} |w_k|\widetilde{\varphi}^2(k)\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned}$$

Proof Analogous to Theorem 5.3. □

Just as for the similar previous theorems, (6.15) can be interpreted as using the residuals from a nominal model to estimate $\widehat{f'}(\varphi_0)$. The constraint (6.10) is needed to make sure that any nominal function value (which we do not know anything about) will have no effect on the derivative estimate.

Let us also study what happens when Δ is large.

Theorem 6.4
Suppose that $\widetilde{\varphi}(k) \neq 0$, $k = 1,\dots,N$, and that there are two indices k_1 and k_2 such that $\widetilde{\varphi}(k_1) \neq \widetilde{\varphi}(k_2)$. Given $b \in \mathbb{R}$ and $L \in (0,\infty)$, there exists a $\Delta_0 \in (0,\infty)$, such that for any $\Delta > \Delta_0$, the minimum of the upper bound $U^1_{\mathcal{F}_2(L,\Delta)}(w)$ given by (6.15), *subject to the constraint* (6.10), *is attained on the subspace*

$$\sum_{k=1}^{N} w_k = 0, \quad \sum_{k=1}^{N} w_k\widetilde{\varphi}(k) = 1 \qquad (6.16)$$

and does not depend on Δ. In other words, given a sufficiently large Δ, the affine derivative estimator (5.7) *minimizing $U^1_{\mathcal{F}_2(L,\Delta)}(w_0,w)$ can be found in the form*

$$\widehat{f'}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) \qquad (6.17)$$

by minimizing the upper bound

$$U^1_{\mathcal{F}_2(L)}(w) = \left(\frac{L}{2}\sum_{k=1}^{N} |w_k|\widetilde{\varphi}^2(k)\right)^2 + \sigma^2\sum_{k=1}^{N} w_k^2 \qquad (6.18)$$

subject to constraints (6.16).

Proof Analogous to the proof of Theorem 5.4. □

Again, the conclusion is that a too uncertain *a priori* given value of $f'(\varphi_0)$ will have no effect on the estimate $\widehat{f}'(\varphi_0)$, but will be cancelled out by an appropriate choice of weights.

6.1.3 Class $\mathcal{F} = \mathcal{F}_2(L)$

Taking Theorems 6.3 and 6.4 into account, we directly arrive at the following:

Theorem 6.5
For the function class $\mathcal{F}_2(L)$, the affine estimator minimizing $U^1_{\mathcal{F}_2(L,\Delta)}(w_0, w)$ is found among the estimators satisfying

$$\widehat{f}'(\varphi_0) = \sum_{k=1}^{N} w_k y(k), \qquad \sum_{k=1}^{N} w_k = 0, \quad \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) = 1 \tag{6.19}$$

For this kind of estimators, the worst-case MSE has the following upper bound:

$$WMSE(\widehat{f}'(\varphi_0), \mathcal{F}_2(L)) \leq U^1_{\mathcal{F}_2(L)}(w) = \left(\frac{L}{2}\sum_{k=1}^{N} |w_k| \widetilde{\varphi}^2(k)\right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \tag{6.20}$$

We can compare the estimator (6.19) to the local polynomial derivative estimator (2.30). For $j = p = 1$, it turns out that the local linear derivative estimator will belong to the class of estimators described by (6.19), and will thus have a finite worst-case MSE. However, in general it will of course not minimize (6.20).

6.1.4 QP Formulations

Just as for the function estimation problems, it is straightforward to write the problems of minimizing the upper bounds on the worst-case MSE given in Theorems 6.1, 6.3, and 6.5 as convex quadratic programs. For instance, the weights minimizing (6.9) can be found by solving

$$\begin{aligned}
\min_{w,s} \quad & \left(\delta s_a + \Delta s_b + \frac{L}{2}\sum_{k=1}^{N} \widetilde{\varphi}^2(k) s_k\right)^2 + \sigma^2 \sum_{k=1}^{N} s_k^2 \\
\text{subj. to} \quad & s_a \geq \pm \sum_{k=1}^{N} w_k \\
& s_b \geq \pm \left(\sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - 1\right) \\
& s_k \geq \pm w_k, \qquad k = 1, \ldots, N
\end{aligned} \tag{6.21}$$

The other QP:s are obtained analogously.

6.1.5 Can We Use the Derivative Estimate to Improve the Function Estimate?

Suppose we are to estimate a function $f \in \mathcal{F}_2(L)$ at a given point φ_0, given no *a priori* information about $f(\varphi_0)$ and $f'(\varphi_0)$. An alternative to using the QP formulation (3.11) might be to first estimate a derivative according to Theorem 6.5, and then use this derivative when estimating the function according to Corollary 5.1. However, as we will now show, we will not gain anything by this procedure.

Let us denote the derivative estimator by

$$\widehat{f'}(\varphi_0) = \sum_{k=1}^{N} v_k y(k) \tag{6.22}$$

where v_k should satisfy the constraints

$$\sum_{k=1}^{N} v_k = 0, \quad \sum_{k=1}^{N} v_k \widetilde{\varphi}(k) = 1$$

similar to (6.16). The function estimator would then become

$$\hat{f}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - \widehat{f'}(\varphi_0) \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \tag{6.23}$$

where the weights w_k satisfy the constraint (5.16). Thus, based on Corollary 5.1, we obtain a *plug-in estimator* in the form

$$\hat{f}(\varphi_0) = \sum_{k=1}^{N} w_k y(k) - \left(\sum_{t=1}^{N} v_t y(t) \right) \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) \tag{6.24}$$

satisfying the constraints

$$\sum_{k=1}^{N} w_k = 1, \quad \sum_{k=1}^{N} v_k = 0, \quad \sum_{k=1}^{N} v_k \widetilde{\varphi}(k) = 1 \tag{6.25}$$

Now the following theorem holds:

Theorem 6.6
The plug-in estimator (6.24) *belongs to the family of estimators defined by* (5.23)*, satisfying the restrictions* (5.22)*. Hence, for the class $\mathcal{F}_2(L)$, the performance in terms of minimizing $U_{\mathcal{F}_2(L)}(w)$ in* (5.24) *cannot be better for the plug-in estimator than for the estimator given by* (3.11).

Proof The plug-in estimator (6.24) can easily be rewritten in the form (5.23):

$$\begin{aligned} \hat{f}(\varphi_0) &= \sum_{k=1}^{N} w_k y(k) - \sum_{k=1}^{N} v_k y(k) \sum_{t=1}^{N} w_t \widetilde{\varphi}(t) \\ &= \sum_{k=1}^{N} \left(w_k - v_k \sum_{t=1}^{N} w_t \widetilde{\varphi}(t) \right) y(k) = \sum_{k=1}^{N} \widetilde{w}_k y(k) \end{aligned} \tag{6.26}$$

where

$$\widetilde{w}_k = w_k - v_k \sum_{t=1}^{N} w_t \widetilde{\varphi}(t) \tag{6.27}$$

Hence, due to the conditions (6.25), $\widetilde{w}_k$ will satisfy the following constraints:

$$\sum_{k=1}^{N} \widetilde{w}_k = \sum_{k=1}^{N} w_k - \left(\sum_{k=1}^{N} v_k\right) \sum_{t=1}^{N} w_t \widetilde{\varphi}(t) = 1 \tag{6.28}$$

and

$$\sum_{k=1}^{N} \widetilde{w}_k \widetilde{\varphi}(k) = \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) - \left(\sum_{k=1}^{N} v_k \widetilde{\varphi}(k)\right) \sum_{t=1}^{N} w_t \widetilde{\varphi}(t) = 0 \tag{6.29}$$

Thus, the plug-in estimator is a linear estimator satisfying (5.22), among which the estimator minimizing the upper bound (3.9) on the worst-case MSE is given by the solution to (3.11). □

The consequence of Theorem 6.6 is that for any reasonable (where reasonable means having a finite worst-case MSE) estimate of the derivative based on the given data samples, we will get a plug-in estimator which satisfies the constraints (5.22). But for these estimators we already know how to obtain the best one (in terms of the upper bound on the worst-case MSE), namely by solving the QP (3.11). In other words, we have not gained anything at all. Intuitively, this should be quite natural, since we do not add any information by estimating the derivative. The only information we have are the values of L, σ, and $(\varphi(k), y(k))$, both before and after the estimation of the derivative.

6.2 Minimizing the Exact Worst-Case MSE

In Section 3.1.1, it was pointed out that the proposed DWO estimator achieves the linear minimax risk also nonasymptotically, if considering the function class $\mathcal{G}_2(L)$ (or, more generally, $\mathcal{G}_{p+1}(L)$). The reason for this was that the upper bound on the worst-case MSE was tight for all possible weights. However, this does not hold when considering the function class $\mathcal{F}_{p+1}(L)$, since the upper bound is not tight if any of the weights are negative. It then becomes interesting to investigate whether it is possible to minimize the exact worst-case MSE over $\mathcal{F}_{p+1}(L)$ instead of minimizing the upper bound. As it turns out, it is actually possible, at least in the univariate case.

In this section, we will discuss the minimization of the exact worst-case MSE, and point out how this can be done. For simplicity, we only consider univariate functions, belonging to the class $\mathcal{F}_2(L)$. The extension to any other degree of differentiability should be straightforward. However, extending the results to multivariate functions has not been considered.

For the function class $\mathcal{F}_2(L)$ and a linear estimator (3.4), the worst-case MSE can be written

$$\begin{aligned} WMSE(\hat{f}(\varphi_0), \mathcal{F}_2(L)) &= \sup_{f \in \mathcal{F}_2(L)} E[\left(\sum_{k=1}^{N} w_k y_k - f(\varphi_0) \right)^2 | \varphi_1^N] \\ &= \sup_{f \in \mathcal{F}_2(L)} \left(\sum_{k=1}^{N} w_k f(\varphi(k)) - f(\varphi_0) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \end{aligned} \tag{6.30}$$

As we have already seen in (3.5), to get a finite worst-case MSE the weights must satisfy (3.7). Hence, the problem to solve is

$$\begin{aligned} \min_{w} \sup_{f \in \mathcal{F}_2(L)} & \left(\sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0)) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2 \\ \text{subj. to} \quad & \sum_{k=1}^{N} w_k = 1 \\ & \sum_{k=1}^{N} w_k \varphi(k) = 0 \end{aligned} \tag{6.31}$$

Note that, for a fixed given function f, the function

$$J_f(w) = \left(\sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0)) \right)^2 + \sigma^2 \sum_{k=1}^{N} w_k^2$$

is a positive definite quadratic function of w (and hence convex). Therefore, since the objective function of (6.31) is the supremum of such convex functions, i.e.,

$$\sup_{f \in \mathcal{F}_2(L)} J_f(w)$$

problem (6.31) is a *convex* optimization problem. This is of course a great advantage, since it means that any local optimum will be a global optimum, and also since there exist efficient algorithms for solving such problems (see, e.g., [117]).

The greatest difficulty with solving (6.31) is that the evaluation of the objective function for given weights w_k is computationally relatively expensive. For this evaluation, a subalgorithm, sketched in the next section, is used to find the value of the bias part of (6.30), i.e.,

$$\max_{f \in \mathcal{F}_2(L)} \left(\sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0)) \right)^2 \tag{6.32}$$

for given weights w_k. This can be used together with a standard optimization algorithm (see, e.g., [117]) to find the solution to the entire problem. Note that since the supremum of (6.30) is attained, as we will see in Section 6.2.1, we can use maximum instead of supremum in (6.32).

6.2.1 Maximizing the MSE for Given Weights

Instead of solving (6.32), we can try to find a function that maximizes the bias, i.e.,

$$f^{max} \in \operatorname*{argmax}_{f \in \mathcal{F}_2(L)} \sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0)) \tag{6.33}$$

This function will then also maximize the MSE. Note that f^{max} by no means is unique (as we will see below, we can for instance find functions with any function value and derivative at φ_0, which will maximize the bias). In some texts, the notation Argmax is used instead of argmax to show that the result is a set of arguments, instead of a single argument. Note also that we equally well could have used argmin instead of argmax in (6.33), since $-f^{max}$ would minimize the bias, and what we are really interested in is the squared bias (6.32).

To find the value of (6.33), we first note that every function in $\mathcal{F}_2(L)$ can be written as

$$f(\varphi) = f(\varphi_0) + f'(\varphi_0)\widetilde{\varphi} + f_0(\varphi)$$

where $f_0 \in \mathcal{F}_2(L)$, $f_0(\varphi_0) = f_0'(\varphi_0) = 0$. This gives us

$$\sum_{k=1}^{N} w_k (f(\varphi(k)) - f(\varphi_0)) = f'(\varphi_0) \sum_{k=1}^{N} w_k \widetilde{\varphi}(k) + \sum_{k=1}^{N} w_k f_0(\varphi) = \sum_{k=1}^{N} w_k f_0(\varphi)$$

which means that we, without restrictions, can assume that $f(\varphi_0) = f'(\varphi_0) = 0$.

Secondly, we can assume that the maximal value of (6.33) is obtained for a piecewise quadratic function with a finite number of breakpoints, and with $|f''(\varphi)| = L$ (where the second derivative exists). The reason for this is that the only interesting values of the function are $f(\varphi(k))$ and $f'(\varphi(k))$ for each data point, and it is quite easy to show that for a given set of admissible values for $f(\varphi(k))$ and $f'(\varphi(k))$, there is a piecewise quadratic function with $|f''(\varphi)| = L$ (and a finite number of breakpoints) such that these values are obtained.

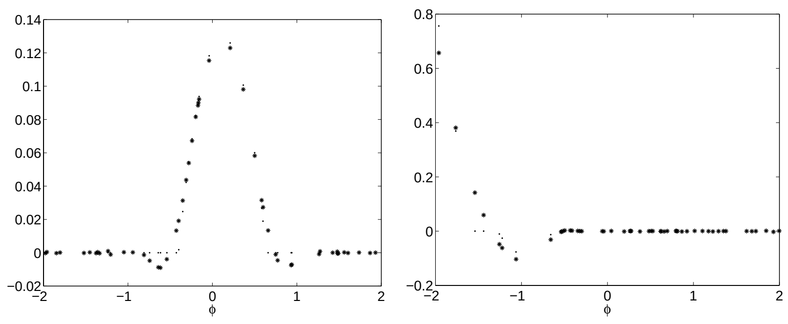
Hence we can parameterize f as follows:

$$f(\varphi) = \pm L \left(\frac{\widetilde{\varphi}^2}{2} + \sum_{j=1}^{p^+} (-1)^j \max\{\widetilde{\varphi} - b_j^+, 0\}^2 + \sum_{j=1}^{p^-} (-1)^j \max\{b_j^- - \widetilde{\varphi}, 0\}^2 \right)$$

where $b_{p^-}^- < \cdots < b_1^- < 0 \leq b_1^+ < \cdots < b_{p^+}^+$ are the breakpoints. Now we can differentiate the bias with respect to the breakpoints:

$$\frac{\partial}{\partial b_j^+} \sum_{k=1}^{N} w_k f(\varphi(k)) = \pm 2L(-1)^{j+1} \sum_{k=1}^{N} w_k \max\{\widetilde{\varphi}(k) - b_j^+, 0\}$$

$$\frac{\partial}{\partial b_j^-} \sum_{k=1}^{N} w_k f(\varphi(k)) = \pm 2L(-1)^{j} \sum_{k=1}^{N} w_k \max\{b_j^- - \widetilde{\varphi}(k), 0\}$$



(a) Interior point, $\varphi_0 = 0$. (b) Boundary point, $\varphi_0 = -2$.

Figure 6.1: Optimal weights with respect to the exact worst-case MSE ($*$) and weights computed with the DWO approach ($\cdot$), for two different values of φ_0.

Note that each derivative is a (piecewise affine) function of only one variable, namely the variable with respect to which the bias was differentiated. This means that we only have to consider the function

$$g^+(b) = \sum_{k=1}^{N} w_k \max\{\widetilde{\varphi}(k) - b, 0\}$$

and its zeros and sign changes to find the locations of the positive breakpoints, and a corresponding function $g^-(b)$ to find the negative breakpoints. The details of this are theoretically quite simple, although a bit tricky to implement, partly since it is difficult to know how many breakpoints are needed. It is easy to see, though, that at most N breakpoints are needed, since the derivative is piecewise affine and can have at most one zero in each interval between two consecutive data points $\varphi(k)$. Further investigations show that the number of breakpoints needed are at most the number of sign changes of w_k.

6.2.2 Properties of the Exact Worst-Case MSE Solution

We illustrate the properties of the resulting weights by an example.

Example 6.1 *Let us consider the case with $N = 50$ data samples, where $\varphi(k)$ are taken from a uniform distribution on the interval $[-2, 2]$. Let $L/\sigma = 5$. Figure 6.1 shows both the weights minimizing the worst-case MSE and the weights from the DWO approach* (3.11) *for two different values of φ_0, namely $\varphi_0 = 0$ (which is an interior point) and $\varphi_0 = -2$ (a boundary point). The weights minimizing the worst-case MSE were computed using the* Matlab *function* `fmincon`*.*

An interesting thing to note is that the optimal weights have several sign changes. This can be compared to the solution in [88], where the asymptotically optimal minimax estimator was computed. It would be reasonable to believe that, as $N \to \infty$, these estimators would converge to each other.

In Table 6.1, the worst-case MSE values are listed for the minimization of the exact worst-case MSE and for the DWO approach, as well as the optimal upper bound on the worst-case MSE given by the DWO approach. We can note that

Table 6.1: Worst-case MSE for the DWO approach and for the optimal weights.

	Upper bound, DWO	WMSE, DWO	Optimal WMSE
Interior point	0.1233	0.1233	0.1208
Boundary point	0.8265	0.7659	0.7198

the upper bound is tight for the interior point, since the weights are all positive, but that the optimal solution contains negative weights and can therefore reach a slightly lower worst-case MSE. However, the difference is not very large.

For the boundary point, the difference between the two approaches is larger, but is still not more than around 6%. Preliminary experiments indicate that these cases are quite typical.

6.3 Other Extensions

Apart from the previously mentioned extensions, several other extensions can be imagined. In this section, we briefly outline some of them.

Nonconstant Variance

The variance of the errors $e(k)$ have been assumed to be constant and equal to σ^2. There is nothing that prevents us from considering the case of each error having its own variance σ_k^2. The changes to the QP formulations are minor and immediate.

One could also assume that the noise terms are correlated (with a known covariance matrix). In this case, the variance term will contain cross-terms between the different weights. However, as long as the correlation is known, the problem to solve will still be a convex QP.

Varying L

As well as letting the variance depend on k, we can also let the L be sample-dependent. For instance, instead of the Lipschitz condition (3.2), we could use the assumption that f satisfies

$$|f(\varphi(k)) - f(\varphi_0) - f'(\varphi_0)\widetilde{\varphi}(k)| \leq \frac{L_k}{2}\widetilde{\varphi}^2(k)$$

for all $k = 1, \ldots, N$. The problem will still be possible to formulate as a QP.

A special case of varying L is to consider a modified version of class $\mathcal{G}_{p+1}(L)$, which is obtained if the Lipschitz condition (4.1) is replaced by

$$\left| f(\varphi) - f(\varphi_0) - \sum_{i_1=1}^{n} f'_{i_1}(\varphi)\widetilde{\varphi}_{i_1} - \ldots - \frac{1}{p!} \sum_{i_1=1}^{n} \cdots \sum_{i_p=1}^{n} f^{(p)}_{i_1 \ldots i_p}(\varphi)\widetilde{\varphi}_{i_1} \ldots \widetilde{\varphi}_{i_p} \right| \\ \leq \frac{1}{(p+1)!} \|Q\widetilde{\varphi}\|^{p+1}$$

where $Q \in \mathbb{R}^{n \times n}$ is a quadratic matrix. This means that the upper bound depends on in what direction we are looking.

More General Function Classes

Following the presentation in [138], we can consider an extension of the function class $\mathcal{G}_{p+1}(L)$, specified by

$$\left| f(\varphi_0) - \sum_{k=0}^{p} a_k f_k(\widetilde{\varphi}) \right| \leq M(\widetilde{\varphi})$$

where f_k and M are given functions. Letting $f_k(\widetilde{\varphi}) = \widetilde{\varphi}^k$ and $M(x) = M|\widetilde{\varphi}|^{p+1}$ brings us back to the definition (2.9) of $\mathcal{G}_{p+1}(L)$. The changes to the problem formulations will be straightforward.

7 Conclusions

In this part of the thesis, a direct weight optimization approach to the function estimation problem has been proposed and studied. As have been pointed out, the approach has a number of interesting features, including the following:

- The problem is phrased without any reference to bandwidth. The formulation offers the possibility to use all observations. The weights resulting from the DWO approach however show that there is a bandwidth feature even for a finite number of measurements: Observations outside a certain band carry weights that are exactly zero.

- Although the DWO approach does not give strictly better estimates (in the MSE sense) than, say, the local polynomial approach in all cases, the important point is that the delivered guaranteed MSE bound is better than what other approaches can offer. In practice, it is of course only this guaranteed bound that can be used for confidence intervals etc., since the actual MSE depends on the unknown function.

- The improvement over asymptotically optimal estimates is more pronounced (naturally enough) for fewer data, and more nonuniformly spread observation points $\varphi(k)$. For applications to higher regressor dimensions and dynamical systems, this is a very valuable property.

- The approach is easily extended to the case when *a priori* bounds on the function and its derivative are known, and improvements can be made by making use of this information. Theorems 5.2 and 5.4 however showed that when the bounds are very wide, the extra information may not be enough to improve the solution.

- For the class of affine functions ($L = 0$), the solution equals the one obtained by globally fitting an affine model to the data using the least-squares criterion, as should be expected.

- Asymptotically, for univariate functions and equally spaced design, and when minimizing the worst-case MSE over $\mathcal{G}_2(L)$ (or, equivalently, the upper bound (3.9) on the worst-case MSE over $\mathcal{F}_2(L)$), we obtain the weights of the local linear estimator using an asymptotically optimal Epanechnikov kernel. It is conjectured that these results could be extended also to more general designs and other degrees of differentiability. Furthermore, it is conjectured that minimizing the exact worst-case MSE over $\mathcal{F}_{p+1}(L)$, as was done in Section 6.2, asymptotically leads to the optimal kernels given by [88] under appropriate design assumptions.

Local modelling has found a wide range of applications, like for instance robot control, computer graphics, fermentation processes, econometrics, etc. A good survey is given in [5]. In all these domains, the DWO might naturally be an alternative.

In Section 4.3, some examples were given on how local modelling in general, and the DWO approach in particular, can be used in connection with system identification and prediction of dynamic systems. The idea of storing old data from a dynamic system in a database, and only retrieve the data needed each time a prediction is to be formed, has been called a *Model On Demand (MOD)* approach, and is considered in [146]. As described there, the predictions can be used, e.g., for *model predictive control (MPC)*. An early contribution employing similar ideas is [7], where a simple MOD-like method is used to control robots. “Experiences” are stored in a database, and when predictions are to be made, the database is searched for similar situations. Related ideas can also be found, e.g., in [17].

In addition, frequency domain applications are given in [146], such as automatic smoothing of periodograms and empirical transfer function estimates (ETFE) [94].

This part of the thesis has been focused on the main ideas of the DWO approach, and a number of properties have been pointed out and shown. Still, however, there are a number of questions that need to be discussed. One such question is what happens when there is a correlation between the noise and regression variables, when using the approach for dynamic systems. Another issue is how to find appropriate values of the design parameters, such as the structure of φ and the ratio L/σ. These problems will now be considered.

7.1 Using the DWO Approach for Dynamic Systems

As pointed out in Chapter 2, many of the local modelling approaches, including the DWO approach, assume that the regression vectors $\varphi(i)$ and noise contributions $e(i)$ are independent. However, when the data come from a dynamic system, such that $\varphi(i)$ will contain old values $y(j)$, $j < i$, this independence will not hold. The question is whether or not the DWO approach can be used for dynamic systems anyway. A simple answer is yes: instead of considering the minimization of (3.11) as minimizing the upper bound on the worst-case MSE, it can be viewed as a reasonable heuristic criterion to minimize. As could be seen in Examples 4.1 and 4.2, it can also work well in practice.

Let us have a closer look at the problem. For simplicity we consider the function class $\mathcal{F}_{p+1}(L)$. Suppose that the regression vector $\varphi(k)$ contains $y(k-1)$. This would mean that if $\varphi(k)$, $k = 1, \ldots, N$ are given, then also the corresponding $y(k-1)$ are given. Hence, the MSE becomes

$$\begin{aligned}
MSE(\hat{f}(\varphi_0)) &= E[(\hat{f}(\varphi_0) - f(\varphi_0))^2 | \varphi_1^N] \\
&= E[\left(\sum_{k=1}^{N} w_k y(k) - f(\varphi_0)\right)^2 |\varphi_1^N] \\
&= E[\left(\sum_{k=1}^{N-1} w_k y(k) + w_N f(\varphi(N)) - f(\varphi_0) + w_N e_N\right)^2 |\varphi_1^N] \qquad (7.1) \\
&= \left(\sum_{k=1}^{N-1} w_k y(k) + w_N f(\varphi(N)) - f(\varphi_0)\right)^2 + \sigma^2 w_N^2 \\
&= \left(\sum_{k=1}^{N} w_k f(\varphi(k)) - f(\varphi_0) + \sum_{k=1}^{N-1} w_k \big(y(k) - f(\varphi(k))\big)\right)^2 + \sigma^2 w_N^2 \\
&= \left(\sum_{k=1}^{N} w_k f(\varphi(k)) - f(\varphi_0)\right)^2 \\
&\quad + 2\left(\sum_{k=1}^{N} w_k f(\varphi(k)) - f(\varphi_0)\right)\left(\sum_{k=1}^{N-1} w_k \big(y(k) - f(\varphi(k))\big)\right) \\
&\quad + 2\sum_{k=1}^{N-2}\sum_{j=k+1}^{N-1} w_k w_j \big(y(k) - f(\varphi(k))\big)\big(y(j) - f(\varphi(j))\big) \\
&\quad + \sum_{k=1}^{N-1} w_k^2 \big(y(k) - f(\varphi(k))\big)^2 + \sigma^2 w_N^2
\end{aligned}$$

Since f is unknown, there is generally no way to evaluate this expression, and we cannot get an upper bound either. However, for large N, the second and third terms of the last expression should generally be much smaller than the squared terms, since $y(k) - f(\varphi(k)) = e(k)$ is the noise contribution, which is averaged in these sums. Furthermore, the fourth and fifth terms should be well approximated by the variance term (2.35). Hence, it seems reasonable to approximate the second and third terms in this expression by 0, and the fourth term by

$$\sigma^2 \sum_{k=1}^{N-1} w_k^2$$

and we are back to the MSE expression (2.33) together with (2.34) and (2.35). The only difference is that this is not the true MSE anymore, but an approximation.

One should also note that, as $N \to \infty$, the weights from the DWO approach will be nonzero only in a very small neighborhood of φ_0. For most reasonable dynamic systems, unless φ is an equilibrium point, this means that the regression vectors corresponding to the nonzero weights will have very different indices k, and so they will in general only be weakly correlated. This means that the approximation of the worst-case MSE used by the DWO approach will be asymptotically correct.

7.2 Adaptive Bandwidth Selection

Throughout the presentation of the DWO approach, the ratio L/σ has been assumed to be known. However, in practice this is often not the case[*]. Instead, the ratio would have to be estimated. One option is to use the methods known as classical methods (e.g., cross-validation techniques), described in Section 2.4.3. If a global estimate is desired, the adaptation of these methods to the DWO approach is immediate. However, in many applications a local estimate of the ratio would be to prefer. In this case, an appealing alternative is to take the view of Section 3.2.4, i.e., to regard the DWO weights as coming from a local polynomial model, where the least-squares problem (2.23) is weighted by the matrix G^{-1} from (3.35) instead of $\bar{K}_h$. Using this view, it is now straightforward to extend the localized adaptive bandwidth selection procedures as described in Section 2.4.3 (and in more detail in [146]) to the DWO approach, where, e.g., the sum $\sum_{k=1}^{N} w_k$ is replaced by $\mathrm{tr}(G^{-1})$. For instance, the localized cross-validation criterion (2.57) would take the form

$$LCV(\hat{f}(\varphi_0)) \triangleq \frac{(Y_{1:n} - F_{1:n})^T G^{-1} (Y_{1:n} - F_{1:n})}{\mathrm{tr}(G^{-1})} \tag{7.2}$$

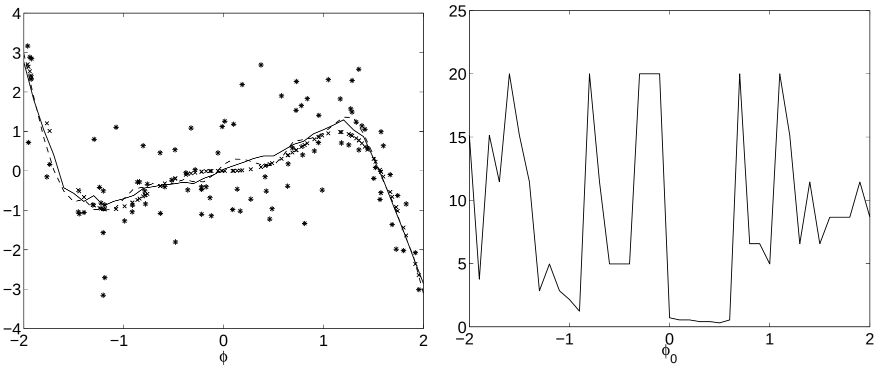
[*]Note however, that as long as the true function belongs to the function class $\mathcal{F}_{p+1}(L_0)$ for any L_0, and the noise variance is σ_0, we could get an upper bound on the worst-case MSE by solving (3.10) with any $L > L_0$ and $\sigma > \sigma_0$. Hence, any ratio L/σ will correspond to an upper bound on the worst-case MSE. However, it will certainly not be the best bound in general.

where $Y_{1:n}$ consists of the outputs corresponding to the nonzero weights, and $F_{1:n}$ is a vector consisting of the corresponding one leave-out estimates, i.e.,

$$Y_{1:n} = \begin{pmatrix} y(1) \\ \vdots \\ y(n) \end{pmatrix}, \qquad F_{1:n} = \begin{pmatrix} \bar{f}^{\varphi_0}_{-1}(\varphi(1)) \\ \vdots \\ \bar{f}^{\varphi_0}_{-n}(\varphi(n)) \end{pmatrix}$$

where the samples are renumbered as in Section 3.2.2.

Example 7.1 *Consider once again the setup of Example 3.3, with $N = 100$ data samples. The function has been estimated using a fixed ratio of $L/\sigma = 13$, and using a localized cross-validation (LCV) criterion as in (7.2). The different function estimates are shown in Figure 7.1(a). As can be seen, the performance of the estimate using the LCV criterion is comparable (maybe even slightly better) than the estimate using a fixed ratio. Figure 7.1(b) shows the estimated ratio L/σ for the both approaches. We can see that the estimate fluctuates quite much, which is natural, since the number of data is rather small.*



(a) Data samples ($*$), the true function values ($\times$), and the estimated functions.

(b) The estimated ratio L/σ, using the LCV criterion.

Figure 7.1: The function estimates in Example 7.1, using a fixed ratio $L/\sigma = 13$ (dashed) and a LCV criterion (solid).

It should be emphasized that the criterion and algorithm used for the computations in Example 7.1 should be seen as just a first step in the topic of adaptive choices of L/σ for the DWO approach. There is plenty of room for improvement of the methods.

As an alternative to the classical methods, one might use a plug-in method similar to what was described in Section 2.4.4. However, to the author's knowledge, not much has been done to estimate a Lipschitz constant as the one used in the DWO approach. The main focus of the plug-in approaches has been to approximate the MSE, rather than the worst-case MSE. Thus, much can still be done in order to find a reliable estimate of L/σ, which is also easy to compute.

7.3 Structure of the Regression Vector

We have assumed that the correct choice of elements for the regression vectors is known. This choice is a structure problem which is common to most approaches in system identification, and there has been a great deal of research within this area. One example is [126], where a method using *false nearest neighbors* was proposed for dynamic systems. This technique builds on the philosophy that has been a theme of this part, namely that outputs corresponding to regression vectors close to each other should be similar (i.e., that the true underlying regression function is smooth). However, if the regression vector is chosen to be too small, two similar regression vectors may correspond to very different output data, since the regression variables that are not included in the chosen regression vector may have a significant effect on the output. These regression vectors are then *false neighbors*, i.e., they are neighbors in the chosen (too small) regressor space, but are far from each other in the true regressor space. In this way, if several false neighbors are found, one can deduce that the chosen regressor space is too small.

A recent contribution in the area of regressor selection is [92], where ANOVA (ANalysis Of VAriance) is used to determine a good regression vector.

7.4 Algorithmic and Implementation Issues

There are many aspects on how to find the solution, e.g., of (3.10). In the early paper [138], a Newton-like algorithm (very similar to the hinge-finding algorithm in [22], described in Section 10.1.2) is proposed. However, nothing is shown about convergence.

The field of convex optimization has developed rapidly during the last decade, and there exist efficient algorithms that can solve a large variety of problems, including large-scale convex QP and SOCP problems. For a good introduction, see [19]. However, for the specific problems presented in this part of the thesis, we can use the knowledge of the automatic finite bandwidth property (described in Section 3.2.1) to make the computations even faster, according to what was suggested in the discussion after Corollary 3.1. If we would be able to guess which of the weights that should be nonzero, only these data points need to be taken into account, which leads to a much smaller QP to be solved. The solution can then be easily checked, by investigating if all the excluded data samples lie in the areas where the weight function is zero. This is the case if all excluded data samples satisfy the middle case of (3.13), i.e., if

$$-g\widetilde{\varphi}^2(k) \leq \mu_1 + \mu_2\widetilde{\varphi}(k) \leq g\widetilde{\varphi}^2(k)$$

If this requirement is satisfied for all excluded data, the solution is globally optimal. Otherwise, we can include the data that do not satisfy the requirement and solve the new QP we get.

The main question that arises is how to find an initial guess of which weights should be nonzero. One immediate option would be to use the asymptotically optimal bandwidths (2.41) or (2.43) (or their multivariate counterparts). It turns out that a slight overestimation of these bandwidths is a better choice, since taking some too many data samples into consideration is to prefer before considering too few, and instead having to solve two QP:s.

A problem with this approach is that the design density has to be known. Furthermore, with sparsely and unevenly spread data samples (which is often the case particularly for higher dimensions) there is a risk that too few data samples falls within the guessed bandwidth. An alternative is therefore to use a k-nearest neighbor approach to find an initial guess of the set of nonzero weights. k should be chosen to be at least larger than the dimension, i.e., $k > n$, in order not to get an underdetermined problem.

If the guess of the set of nonzero weights was wrong, one way of choosing a new set would be to consider all data samples falling into the area where the computed weight function is nonzero. Another alternative is to simply increase the guessed bandwidth by a factor $\alpha > 1$.

Preliminary experiments, using a $2n$-nearest neighbor approach to get an initial guess of which weights should be nonzero, have shown that the computational speed can be considerably improved compared to solving one large QP. However, much can probably be done to further improve the algorithms, and to use the structure of the problem in a more refined way. This is of particular importance if using adaptive bandwidth selection, which makes the computational complexity significantly larger.

The problem of finding and retrieving relevant data samples from a database of samples is another relevant issue. For an overview of this, see, e.g., [5, 146].

Part II

Identification of Piecewise Affine Systems

8

Prediction Error Methods for System Identification

When linear models are not sufficient for accurately describing the dynamics of a system, there are several different possible approaches to choose between. In Part I, a nonlinear model was built "on the fly", by locally estimating the system dynamics for each time point. In this part a parametric identification approach is considered, namely using *piecewise affine (PWA)* models. This might be an attractive alternative, both when the real system actually is piecewise affine, but also sometimes for a general nonlinear system.

This chapter briefly describes a few general issues concerning *prediction error methods*, which is a commonly used family of parametric system identification methods. Chapter 9 then introduces different classes of piecewise affine systems, while Chapters 10 and 11 are devoted to the identification of piecewise affine systems.

8.1 Prediction Error Methods

Let us first once again formulate the system identification problem. We assume that we are given a set of inputs, $u(t)$, and outputs, $y(t)$, $t = 1, \dots, N$, coming from a real (physical) system of some structure. The structure considered here is the form given in (1.6), i.e.,

$$y(t) = f(Z_{-\infty}^{t-1}) + e(t) \tag{8.1}$$

where $E[e(t)] = 0$ and $e(t)$ is independent of $Z_{-\infty}^{t-1}$. Using the given set of data, we would like to find a model that describes the true system as well as possible in some sense.

As already mentioned, the prediction error methods are parametric methods, which means that we search for the best model within a family of models, parameterized by a set of parameters collected in a parameter vector θ. In our context, this means that we only consider a parameterized family of functions $f(Z_1^{t-1}, \theta)$ with a certain structure. The system identification problem then reduces to finding the parameter values θ^* that fit the model to the experimental data as well as possible. In the prediction error methods, this is done by considering the predicted output

$$\hat{y}(t|\theta) = f(Z_1^{t-1}, \theta)$$

and the residual

$$\varepsilon(t, \theta) = y(t) - \hat{y}(t|\theta)$$

Here, $\hat{y}(t|\theta)$ is our guess of the value of $y(t)$, given the information Z_1^{t-1}. Since $e(t)$ is independent of Z_1^{t-1}, the best guess we can obtain is what we get by replacing $e(t)$ in (8.1) by its expected value, which is 0. The residual $\varepsilon(t, \theta)$ can be seen as a measure of how close the guess was.

Then the model is fitted to the data by minimizing a *criterion function*

$$V(\theta, Z_1^N) = \frac{1}{N} \sum_{t=1}^{N} \ell(\varepsilon(t, \theta)) \tag{8.2}$$

that is,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, V(\theta, Z_1^N)$$

There are many possible choices for the function $\ell(\varepsilon)$. A very common choice is

$$\ell_2(\varepsilon) = \varepsilon^2$$

which gives the *least squares* criterion. We have already seen an example of using the least squares criterion in Example 1.3.

It can be noted, that given that the true system belongs to the parameterized model class, and given that $e(t)$ are independent with the probability distribution function $f_e(x)$, the $\theta(v_j)$ that minimizes (8.2) will be the *maximum likelihood estimate* if

$$\ell(\varepsilon) = -\log f_e(\varepsilon)$$

(see [94]). From this it follows, e.g., that ℓ_2 corresponds to the maximum likelihood estimator for a system where $e(t)$ has a Gaussian distribution with known variance.

8.2 Numerical Minimization

In Example 1.3, it was possible to minimize $V(\theta, Z_1^N)$ analytically. In general, however, $V(\theta, Z_1^N)$ can be a (possibly nonconvex) function which has to be minimized using numerical algorithms. Some of the most important numerical algorithms are the *Newton* and *quasi-Newton algorithms* [117]. These are iterative algorithms, that update the estimate of $\operatorname{argmin}_\theta V(\theta)$ according to

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} - \alpha^{(i)} \left(R(\hat{\theta}^{(i)})\right)^{-1} \nabla V(\hat{\theta}^{(i)}) \tag{8.3}$$

where $\nabla V(\hat{\theta}^{(i)})$ is the gradient of $V(\theta)$ evaluated in $\hat{\theta}^{(i)}$. The role played by $R(\hat{\theta}^{(i)})$ will soon be explained. $\alpha^{(i)}$ is a scalar that determines the *step size* of the update, i.e., how much the estimate $\hat{\theta}^{(i)}$ should change in each iteration.

If $V(\theta)$ is a positive definite quadratic function of θ,

$$V(\theta) = \frac{1}{2}(\theta - \theta_{\min})^T P(\theta - \theta_{\min})$$

we could get right to the minimum in one step using $\alpha^{(0)} = 1$ and $R(\theta) = \nabla^2 V(\theta)$ (the Hessian of V), since

$$\hat{\theta}^{(1)} = \hat{\theta}^{(0)} - \left(\nabla^2 V(\hat{\theta}^{(0)})\right)^{-1} \nabla V(\hat{\theta}^{(0)}) = \hat{\theta}^{(0)} - P^{-1} P(\hat{\theta}^{(0)} - \theta_{\min}) = \theta_{\min}$$

If $V(\theta)$ is not quadratic, we can use the Taylor expansion to approximate it with a quadratic function. However, since the approximation may be good only in a small region around $\hat{\theta}^{(i)}$, we might need to adjust the value of $\alpha^{(i)}$ to ensure that $V(\hat{\theta}^{(i)})$ will actually decrease for increasing i. Methods where $\alpha^{(i)} \neq 1$ are sometimes called *damped*.

The Newton algorithms use $R(\theta) = \nabla^2 V(\theta)$, as described above. However, it might be time-consuming to compute the Hessian $\nabla^2 V(\hat{\theta}^{(i)})$ in each step, compared to computing only the gradient. Moreover, the Hessian is not necessarily positive definite when being far away from local minima. An alternative is therefore to use quasi-Newton algorithms, such as *Gauss-Newton*, that instead form an estimate of $\nabla^2 V(\theta)$, and use that in (8.3). The estimates can be constructed in such a way that they are always positive semidefinite, which can be used to guarantee convergence to a stationary point. For more details about numerical minimization algorithms, see, e.g., [94, 117].

Using numerical minimization algorithms like the ones described above means that there is a risk of getting stuck in local minima. One way of getting around this problem is to try different initial values for the algorithm. In this way we might at least find several local minima, from which we can choose the one with the lowest value of $V(\theta, Z_1^N)$. In the presence of noise and model errors, it also might not be absolutely necessary to find the global minimum, since we do not know if it gives the correct description of the system anyway. Nevertheless, we would like to find a good model, that gives a low value for $V(\theta, Z_1^N)$, so the question arises: How do we find a good solution (or an initial value that leads to a good local minimum)? This question will be treated for piecewise affine systems in Chapters 10 and 11.

8.3 Regularization

In the Newton and quasi-Newton algorithms, the update equation (8.3) contains the inverse $(R(\hat{\theta}^{(i)}))^{-1}$. Sometimes $R(\theta)$ may be ill-conditioned or even singular. To make it better conditioned, a term λI can be added to $R(\theta)$. This is known as *regularization*. One can also add a term $\lambda\|\theta - \theta^{\#}\|^2$ to the criterion function $V(\theta, Z_1^N)$ in (8.2), which will also have the effect of adding $2\lambda I$ to $\nabla^2 V(\theta, Z_1^N)$. This can be interpreted as providing a guess $\theta^{\#}$ of the value of θ, and trying to keep θ close to the guess. The coefficient λ can be used to weigh the importance of staying close to $\theta^{\#}$ against minimizing the prediction error.

9

Piecewise Affine Systems

Piecewise affine systems are an important class of nonlinear systems, which in recent years have been subject to considerable research, due to their close relation to hybrid systems. In this chapter, some different classes and different descriptions of piecewise affine systems will be presented. For notational simplicity, we consider noiseless models. The noise will later be included as an additive term, as in (1.6).

9.1 Continuous Time Models

A general continuous time piecewise affine system could be described by

$$\begin{aligned} \dot{x} &= A(v)x + B(v)u + b(v) \\ y &= C(v)x + D(v)u + d(v) \end{aligned} \tag{9.1}$$

where x is our state vector, u is the input to the system, and y is the system output. The vector v is a *key vector*, describing in what *mode* (i.e., in what affine subsystem) the system is for the time being. v can be a function of x, t, u, or some other external input. We assume that v can only take a finite number of different values.

The systems described by (9.1) include systems which are not always called piecewise affine systems. A common restriction (see for example [72]) is that v only

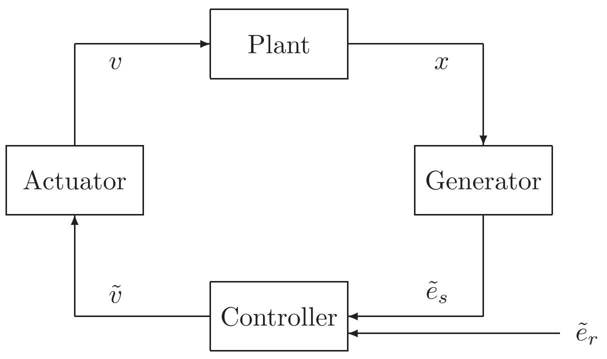


Figure 9.1: The structure of a switched system.

depends on x. Also in this thesis, the general form (9.1) will not be considered, but only subclasses of (9.1); mainly the class of *switched systems* described next.

9.1.1 Switched Systems

The *switched systems*, as defined, e.g., in [43], constitute an important subclass of the systems described by (9.1). The restrictions compared to (9.1) are that $u = 0$ and that the modes (and hence v) are determined by a finite automaton, driven by discrete signals generated when the state vector x reaches different hyperplanes, here called *switching surfaces (or hyperplanes)*, and by discrete external inputs.

The systems can be structured according to Figure 9.1. The continuous affine subsystems are located in the plant, for which the dynamics is given by

$$\dot{x} = A(v)x + b(v)$$

The controller is purely discrete and is modelled as a finite automaton. Its output symbols $\tilde{v}$ are translated by the actuator into key vector signals v for the plant.

The generator contains information about a set of hyperplanes

$$\mathcal{H}_i = \{x \mid C_i x = d_i\}$$

and generates logical input symbols for the controller according to

$$\tilde{e}_{si} = \begin{cases} \texttt{true} & x \in \mathcal{H}_i \\ \texttt{false} & x \notin \mathcal{H}_i \end{cases}$$

Altogether, the system will behave like an affine system as long as there is no external input and the state does not reach one of the switching hyperplanes $\mathcal{H}_i$. When doing so, or receiving an external input, the value of v changes and the system switches to another mode, i.e., to another affine subsystem.

In Chapter 12, we will, for notational purposes, mostly consider a special case of switched systems. However, it should be noted that most of the techniques in

Chapter 12 can be applied to the more general kind of switched systems described above. The systems considered will be in the form

$$\dot{x} = A(v)x + b(v) \ , \quad x \in X(v) \ , \quad v \in \{-1, 0, 1\}^M \tag{9.2}$$

where the state-space is partitioned into different regions $X(v)$ by M hyperplanes. The key vector v is a function of x, and has one element for each separating hyperplane. The elements of v are 1 or -1, depending on on which side of the hyperplanes x is situated, or 0 if x lies in a hyperplane. In this way, v has a one-to-one relationship to $X(v)$. This also implies that the dynamics of a trajectory $x(t)$ just depends on x, not on t or on any external input. The different regions $X(v)$ will be polyhedra (see Section A.3).

9.2 Discrete Time Models

The descriptions in Section 9.1 are continuous time models. It is also common to use discrete time models for the piecewise affine systems. A general form could be

$$\begin{aligned} x(t+1) &= A(v)x(t) + B(v)u(t) + b(v) \\ y(t) &= C(v)x(t) + D(v)u(t) + d(v) \end{aligned} \tag{9.3}$$

Other notable model structures for discrete time piecewise affine systems include *MLD (Mixed Logical Dynamical)* models [12], *MMPS (Max-Min-Plus-Scaling)* models, *LC (Linear Complementary)* models, and *ELC (Extended Linear Complementary)* models [62]. Essentially, these classes are the same as (9.3), where v is a piecewise constant function of x and u, and is constant over polyhedra in the combined state-space and input space (this is shown in [9] (for MLD), [62] (all classes)).

9.2.1 Models in Regression Form

In Chapters 10 and 11, we will use models in *regression form*, i.e.,

$$y(t) = \varphi^T(t)\theta(v) \tag{9.4}$$

where the key vector v only can take a finite number of values, and $\theta(v)$ is a function of v. The vector φ is our regression vector, which could for instance consist of old inputs and outputs, e.g.,

$$\varphi(t) = \begin{pmatrix} 1 & -y(t-1) & \dots & -y(t-n_a) & u(t-1) & \dots & u(t-n_b) \end{pmatrix}^T \tag{9.5}$$

Here we have included a constant 1 in the regression vector, to be able to express piecewise affine systems in the form (9.4). If we would not include the constant, (9.4) would be a piecewise linear system. Throughout this thesis, whenever we are dealing with piecewise affine systems in regression form, we will assume that the first element of $\varphi(t)$ equals 1. We will also assume that the key vector v is uniquely determined by the regression vector $\varphi(t)$, and that the regions of the different affine subsystems are polyhedral. When including additive white noise in the model (see Section 1.1.3), this kind of systems has been called *PWARX (PieceWise AutoRegressive eXogenous)* systems [13, 48]. These systems are a subclass of the systems described by (9.3), and can easily be transformed into that form by using the elements of $\varphi(t)$ (except the constant) as state variables.

However, there are also many other possible choices of φ. In general, $\varphi(t)$ could be any function of u and y, i.e.,

$$\varphi(t) = \begin{pmatrix} f_1(Z_1^{t-1}) & \dots & f_n(Z_1^{t-1}) \end{pmatrix}^T$$

In this case, a system in the form (9.4) could be called a *PWNARX (PieceWise Nonlinear AutoRegressive eXogenous)* system. This is of course not a piecewise affine system in general, but as we will see, most of the theory in Chapter 11 also applies to systems with this choice of $\varphi(t)$.

Just as in Part I, we assume that the form of the regression vectors is given, i.e., that we know which kind of data is relevant to predict $y(t)$. We refer to [92, 126] for ideas about how to determine this.

9.2.2 Chua's Canonical Representation and Hinging Hyperplane Models

A special class of piecewise affine functions are the functions that can be represented by the *canonical representation* introduced by Chua and Kang [28, 80]:

$$y(t) = \alpha_0^T \varphi(t) + \sum_{i=1}^{M} c_i |\alpha_i^T \varphi(t)| \tag{9.6}$$

where $\varphi(t)$ is given from (9.5). As can be seen, the functions described by Chua's canonical representation are continuous in $\varphi(t)$. It turns out (see [28]) that every scalar, continuous, piecewise affine function of one variable can be uniquely expressed in this form with $\alpha_{i2} = 1$ (where α_{i2} is the second element of α_i), i.e., as

$$y(t) = \alpha_{01} + \alpha_{02} x(t) + \sum_{i=1}^{M} c_i |\alpha_{i1} + x(t)|$$

(here we let $\varphi(t) = \begin{pmatrix} 1 & x(t) \end{pmatrix}^T$). However, not every multivariate, continuous, piecewise affine function can be expressed on the canonical representation (see Example 9.2). A necessary condition for this is that the regions of the different affine subsystems must be separated by hyperplanes. In [27], a necessary and sufficient condition for a function to be expressible in the canonical form is given. There are also other canonical representations covering larger classes of systems. One of them is Güzeliş' model class [59], where two nested absolute values are allowed:

$$y(t) = \alpha_0^T \varphi(t) + \sum_{i=1}^{M} b_i |\alpha_i^T \varphi(t)| + \sum_{j=1}^{K} c_j \left| \gamma_j^T \varphi(t) + \sum_{i=1}^{M} d_{ij} |\alpha_i^T \varphi(t)| \right| \tag{9.7}$$

In [91], it is shown that every continuous piecewise affine function can be written using (possibly deeply) nested absolute values. This result is extended in [89], where it is shown how deep the nesting needs to be in different situations. It turns out that for an n-dimensional continuous piecewise affine function, the nesting level k satisfies $k \leq n$.

Hinging hyperplane functions were introduced by Breiman [22] as a model class for function approximation and classification. They are composed of a sum of *hinge functions*, which are two half-planes joined continuously together at the *hinge*. The hinging hyperplane functions can be written as

$$y(t) = \sum_{i=1}^{M} \pm \max\{\beta_i^{+T} \varphi(t), \beta_i^{-T} \varphi(t)\} \tag{9.8}$$

The $\pm$ signs before the max functions play essentially the same role as c_i in (9.6), and are needed to allow for both convex and nonconvex functions. Since we can write

$$\begin{aligned}
&\max\{\beta_i^{+T} \varphi(t),\, \beta_i^{-T} \varphi(t)\} \\
&\quad = \frac{1}{2}(\beta_i^+ + \beta_i^-)^T \varphi(t) \\
&\qquad + \max\Big\{\big(\beta_i^+ - \frac{1}{2}(\beta_i^+ + \beta_i^-)\big)^T \varphi(t),\, \big(\beta_i^- - \frac{1}{2}(\beta_i^+ + \beta_i^-)\big)^T \varphi(t)\Big\} \\
&\quad = \frac{1}{2}(\beta_i^+ + \beta_i^-)^T \varphi(t) + \frac{1}{2}\max\Big\{(\beta_i^+ - \beta_i^-)^T \varphi(t),\, -(\beta_i^+ - \beta_i^-)^T \varphi(t)\Big\} \\
&\quad = \frac{1}{2}(\beta_i^+ + \beta_i^-)^T \varphi(t) + \frac{1}{2}\Big|(\beta_i^+ - \beta_i^-)^T \varphi(t)\Big|
\end{aligned}$$

it is easy to see that (9.6) and (9.8) define equivalent classes of functions. Another representation for the same class, which will be used here, and which can be transformed to Chua's or Breiman's form in a similar way, is

$$y(t) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M^+} \max\{\varphi^T(t)\theta_i, 0\} - \sum_{i=M^+ +1}^{M} \max\{\varphi^T(t)\theta_i, 0\} \tag{9.9}$$

Here the piecewise affine function is written as a sum of an affine term and M (positive and negative) max functions. Instead of the $\pm$ signs of (9.8), the positive max functions have been collected in one sum, and the negative into another. However, we will use the shorthand notation

$$y(t) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i, 0\} \tag{9.10}$$

thereby meaning exactly the same as in (9.9).

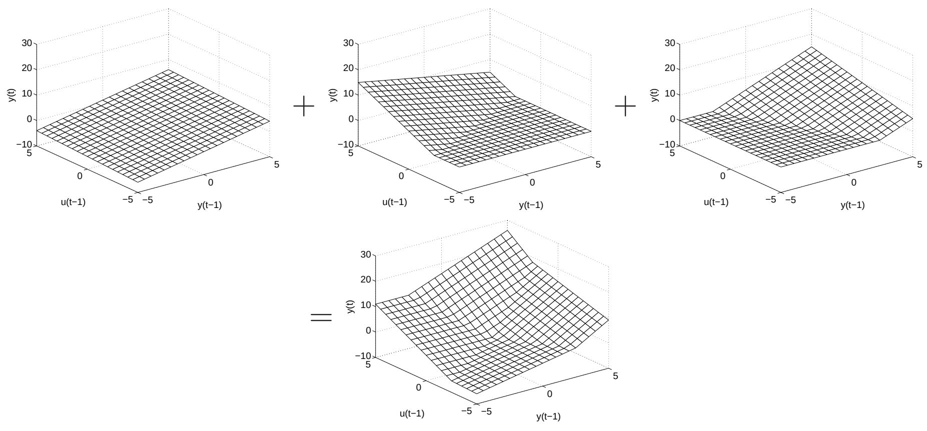


Figure 9.2: Example of a hinging hyperplane function. The upper leftmost function is the linear function in the first row of (9.11), and the middle and right diagrams correspond to the two max functions. The resulting function is shown in the lower diagram.

Example 9.1 (Hinging hyperplane model) *Consider Figure 9.2, where the following hinging hyperplane model is illustrated:*

$$\begin{aligned} y(t) &= y(t-1) + 0.2u(t-1) \\ &\quad + \max\{-y(t-1) + 2u(t-1), 0\} \\ &\quad + \max\{2y(t-1) + u(t-1), 0\} \end{aligned} \tag{9.11}$$

If we express the function in Chua's canonical representation, it could take the following form:

$$\begin{aligned} y(t) &= 1.5y(t-1) + 1.7u(t-1) \\ &\quad + |-0.5y(t-1) + u(t-1)| \\ &\quad + |y(t-1) + 0.5u(t-1)| \end{aligned}$$

In Breiman's form the same function can be written:

$$\begin{aligned} y(t) &= \max\{2.2u(t-1), y(t-1) + 0.2u(t-1)\} \\ &\quad + \max\{2y(t-1) + u(t-1), 0\} \end{aligned}$$

Example 9.2 *The following system (see Figure 9.3) cannot be expressed in the hinging hyperplane form:*

$$y(t) = \begin{cases} y(t-1) & \text{if } y(t-1) \geq 0,\ y(t-1) + u(t-1) \geq 0 \\ -y(t-1) & \text{if } y(t-1) < 0,\ -y(t-1) + u(t-1) \geq 0 \\ u(t-1) & \text{if } y(t-1) + u(t-1) < 0,\ -y(t-1) + u(t-1) < 0 \end{cases}$$

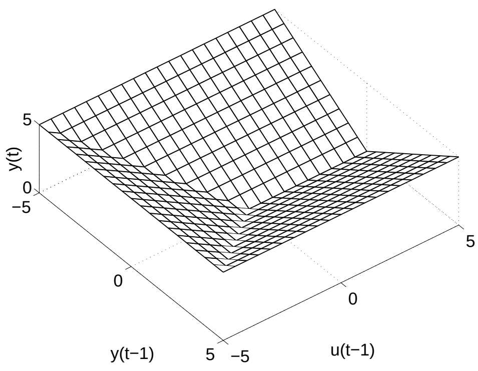


Figure 9.3: A piecewise affine system not expressible in the hinging hyperplane form.

This is due to the fact that the regions of the different subsystems are not separated by hyperplanes (but half-hyperplanes), so there is no way to place the hinges in the hinging hyperplane representation to get this partition of the state-space.

A property of the class described by (9.6), (9.8), and (9.9), which makes it suitable for function approximation, is that it has universal approximation properties [90]. Roughly speaking, this means that we can approximate any function arbitrarily well by using sufficiently many hinge functions (i.e., by letting $M \to \infty$).

We can note that the parameters of (9.9) are not uniquely determined, since for example

$$\varphi^T\theta_0 + \max\{\varphi^T\theta_1, 0\} = \varphi^T(\theta_0 + \theta_1) + \max\{-\varphi^T\theta_1, 0\}$$

This means that we can replace $\max\{\varphi^T\theta_1, 0\}$ by $\max\{-\varphi^T\theta_1, 0\}$, include the difference in the linear term, and still have the same system. Another reason for the ambiguity of (9.9) is that we can reorder the max functions. One way to at least partially avoid this ambiguity is to require that $0 \leq w^T\theta_1 \leq \cdots \leq w^T\theta_{M^+}$ and $0 \leq w^T\theta_{M^++1} \leq \cdots \leq w^T\theta_M$ for some constant (nonzero) vector w.

9.2.3 Representations with Fixed Regions

In function approximation applications, it is quite common to use piecewise affine models where the state-space is partitioned into a regular pattern of regions, with an affine function for each region. For example, in [76, 78], a representation called *HL CPWL (High Level Canonical PieceWise Linear)* representation is used, which is based on a partition of the state space into simplices (polytopes with $n + 1$ corners, where n is the dimension).

9.3 State Jumps

In this thesis, the state is assumed to move continuously in the continuous time models. However, there are situations when it is natural to allow state jumps, like in the following example.

Example 9.3 (Docking spacecraft) *Consider two spacecraft with unit mass, flying along a line in free space. The control signals are the forces acting on them. The system can be described by the following equations, where x_1 is the position and x_2 the velocity of the first spacecraft, and x_3 and x_4 the position and velocity of the second spacecraft, respectively:*

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} u \tag{9.12}$$

However, if the two spacecraft dock with each other, the new dynamics becomes

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 & 0 \\ 1/2 & 1/2 \\ 0 & 0 \\ 1/2 & 1/2 \end{pmatrix} u \tag{9.13}$$

Furthermore, we must ensure that $x_2 = x_4$ after the docking, i.e., that the velocities of the two spacecraft are equal. Denoting the values just before the docking by x_2^-, x_4^-, and the values just after the docking by x_2^+, x_4^+, preservation of the momentum gives $x_2^+ = x_4^+ = (x_2^- + x_4^-)/2$. This means that the state may jump when the mode switching occurs. (Note also that the dimension of the state-space can be reduced in the second mode, which would lead to state jumps between different state-spaces.)

9.4 Control and Analysis of Piecewise Affine Systems

Since piecewise affine systems, like other hybrid systems, are highly nonlinear, they might be difficult to analyze. Several approximating methods for analysis and control design have therefore been developed for different classes of hybrid systems.

An excellent overview of different approaches to modelling and control using multiple models (e.g., piecewise affine systems) is given in [112]. Some approaches for optimal control of hybrid systems can be found in [12, 21, 99, 118]. Other sub-optimal approaches have also been proposed, such as [108, 121]. In [83], supervisory control based on a discrete approximation of the system is studied. Controllability problems have been considered, e.g., in [9, 145].

In recent years, some stability analysis results using Lyapunov theory have appeared [20, 38, 72, 73, 87, 120, 135].

10 Identification of Piecewise Affine Systems

To be able to control and analyze a system, one will (in one way or another) need a model of the system that shows the relation between input and output signals, and the sensitivity to disturbances. Depending on what one would like to achieve, the model might be more or less accurate. In many cases, a linear model around the working point will be sufficient. However, when the nonlinearities are not negligible within the operating region, a piecewise affine model might be an attractive alternative. This can be the case both when the real system actually is piecewise affine, and for a general nonlinear system. Obtaining such a model from experimental data can be done in several different ways.

In this chapter, some existing approaches are investigated and compared. Some of the approaches are quite general nonlinear identification methods, for which special cases lead to piecewise affine models, while other approaches are more specialized on piecewise affine systems.

The system models considered in this and the next chapter will mostly be in regression form as described in Section 9.2.1, i.e.,

$$y(t) = \varphi^T(t)\theta(v) + e(t) \tag{10.1}$$

where $\theta(v)$ is a function of the key vector v, which in turn is a piecewise constant function $v = v(\varphi)$ of the regression vector φ, taking only a finite number of values. As described in Section 1.1.3, the noise $e(t)$ should satisfy $E[e(t)] = 0$, and should be independent of $Z_{-\infty}^{t-1}$. The regression vector could consist of old inputs and outputs, e.g.,

$$\varphi(t) = \begin{pmatrix} 1 & -y(t-1) & \dots & -y(t-n_a) & u(t-1) & \dots & u(t-n_b) \end{pmatrix}^T$$

The modes are consequently determined by v. If we let n_v be the number of different modes of the system, and let $\theta = \begin{pmatrix} \theta(v_1) & \dots & \theta(v_{n_v}) \end{pmatrix}^T$, the criterion function to be minimized (cf. Section 8.1) for this kind of models can be written as

$$V(\theta, Z_1^N) = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{n_v} \ell\left(y(t) - \varphi^T(t)\theta(v_j)\right) \chi_{v_j}(\varphi(t)) \tag{10.2}$$

where

$$\chi_{v_j}(\varphi) = \begin{cases} 1 & v(\varphi) = v_j \\ 0 & \text{otherwise} \end{cases}$$

The two most common choices for the function ℓ in (10.2), and the ones that will be considered here, are the squared Euclidean norm (2-norm)

$$\ell_2(\varepsilon) = \varepsilon^2$$

and the 1-norm

$$\ell_1(\varepsilon) = |\varepsilon|$$

Obviously, for a given φ, exactly one of the functions $\chi_{v_j}(\varphi)$ equals 1. This means that all the terms in the inner sum in (10.2) except for one are zero. Hence, if the partitioning χ_{v_j} of the state-space is known, i.e., if we know what experimental data belongs to which affine subsystem, then using 2-norm will reduce the problem of identifying piecewise affine systems of this kind to a least-squares problem like in Example 1.3, which can easily be solved using standard techniques. However, when the partitioning is unknown, and consequently both χ_{v_j} (or, equivalently, $v(\varphi)$) and θ should be estimated, the problem becomes much more difficult. There are two fundamental approaches to take. The first possibility is to define an *a priori* grid (like in Section 9.2.3), and estimate one affine subsystem per cell in the grid (see, e.g., [78]). This approach leads to a simple estimation process, but will in a sense suffer from the curse of dimensionality, since the number of cells (and hence the number of parameters) will grow exponentially with the number of dimensions, which also leads to a need for very large datasets. The other possible approach is to estimate both the partitioning and the subsystems simultaneously or iteratively (like in, e.g., [22, 48, 77]). This means that we will in general need a smaller number of subsystems, but that the estimation process will be more complex, with potentially many local minima, that will complicate the use of local search minimization algorithms.

Using piecewise affine systems for modelling and identification can be seen as a special case of multiple model approaches. In [112], an excellent introduction to nonlinear modelling using multiple models is given together with a survey of the field.

Some of the main issues in piecewise affine system identification are

- the ability to find a good model of the real system, preferably requiring as few parameters as possible,
- the ability to avoid getting stuck in local minima during the search for a good model, and
- the computational complexity of the identification process.

In the next chapter, an identification method based on *mixed-integer linear or quadratic programming* (MILP/MIQP; see Section A.4) will be presented in detail. As we will see, the mixed-integer programming approach guarantees that the global optimum is reached, at the cost of increased complexity. However, using a "truncated mixed-integer optimization algorithm" or a change detection algorithm might in some cases be a feasible alternative, as will be discussed in Section 11.5.

10.1 Existing Approaches

Different approaches to piecewise affine system identification can be found in many fields in the literature, as mentioned in Section 1.2.2. However, many of them are related. One of the crucial points that separate different groups of approaches from each other is how the partition into different regions is done. As we have seen from (10.2), as soon as we know the different regions, it is easy to identify the different affine subsystems one by one, using standard linear system identification techniques. From this viewpoint, the different approaches to identification of piecewise affine systems can be categorized as follows:

- All parameters, both the parameters determining the partition and the parameters of the submodels, are identified simultaneously. Such approaches can be found, e.g., in [8, 51, 77, 124].
- All parameters are identified simultaneously for a model class with a very simple partition, and new submodels/regions are added when needed. Examples of this kind of approach are found in [22, 46, 64, 69, 77, 124].
- The regions and submodels are identified iteratively or in several steps, each step considering either the regions or the models. The methods in [48, 65, 119, 143, 144] belong to this category. This is also the main approach in [112].
- The regions are predetermined or determined using only information about the distribution of the regression vectors, and the local submodels are identified after that. The approaches described in [26, 76, 78, 82, 103, 148, 152, 154] fall into this category.

The following sections describe the different categories more extensively, and some advantages and drawbacks are pointed out. It should also be mentioned that there are of course alternative ways of categorizing the different approaches. For example, one can distinguish between online and offline algorithms.

10.1.1 Identifying All Parameters Simultaneously

Perhaps the most straightforward way of attacking the piecewise affine system identification problem is to formulate the criterion function (10.2), parameterized according to a certain model structure, and then minimize it directly, using a numerical method, e.g., a Gauss-Newton search (see Section 8.2). In that way both the partitioning and the submodels are estimated simultaneously. The greatest advantage with such an approach might be its simplicity, and a reasonable computational complexity (depending on what numerical method is used).

The greatest drawback with this approach is that the optimization algorithm might get trapped in a local minimum. This drawback is shared with many other of the categories.

The result also depends on how the model is parameterized. In general, it is desirable to have as few free parameters as possible, both for numerical and complexity reasons.

One example of this approach is given by **Pucar and Sjöberg [124]** where a hinging hyperplane model, as defined in Section 9.2.2, is used. Another example is found in **Julián et al. [77]**, where Chua's canonical representation (9.6) is used to approximate nonlinear functions. In both cases, the number of hinge functions is assumed to be known. Pucar and Sjöberg present a damped Newton method for estimating all parameters of the model simultaneously, while Julián et al. use a Gauss-Newton algorithm. Local convergence of the parameters is proven. A similar approach, but for one-dimensional functions, is described in [57].

In an earlier article by **Batruni [8]**, a multilayer neural network with piecewise affine functions in Chua's canonical representation (as described in Section 9.2.2) in each layer is proposed. The parameters would be estimated using backpropagation, which basically is a kind of gradient search.

An approach using a 1-norm criterion function can be found in **Gad et al. [51]**. Since the criterion function itself becomes a piecewise affine function of the parameters, they can use a simplex-like optimization algorithm to efficiently find a local minimum. However, as for the other approaches in this category, convergence to a global minimum cannot be guaranteed.

There are several general numerical optimization algorithms that try to reduce the risk of getting stuck in a local minimum. These include simulated annealing, genetic algorithms, etc. (see, e.g., [37] for both simulated annealing and genetic algorithms). The basic idea of both simulated annealing and genetic algorithms is to modify a standard numerical optimization algorithm, such as the ones described in Section 8.2, by repeatedly using random updates of the parameter estimates. These kinds of algorithms can be used for the piecewise affine system identification problem, at the cost of increased computational complexity.

10.1.2 Adding One Partition at a Time

Instead of solving the entire optimization problem at once, it can be divided into several different steps, to get a number of easier optimization problems that can be solved one at a time. One way of achieving this is to start with a simple model, for instance a hinging hyperplane model with only one hinge function, and fit it to the data. Let us call this hinge function $k_1(\varphi, \theta_1)$. The criterion to be minimized is

$$V^1(\theta_1, Z_1^N) = \frac{1}{N}\sum_{t=1}^{N} \ell\left(y(t) - k_1(\varphi(t), \theta_1)\right)$$

If the resulting model $k_1(\varphi, \hat{\theta}_1)$ is not satisfactory, one can add a new hinge function by fitting it to the residual $\varepsilon_1(t) = y(t) - k_1(\varphi(t), \hat{\theta}_1)$; that is, we minimize

$$V^2(\theta_2, Z_1^N) = \frac{1}{N}\sum_{t=1}^{N} \ell\left(\varepsilon_1(t) - k_2(\varphi(t), \theta_2)\right)$$

The sum of the two models, $k_1(\varphi(t), \hat{\theta}_1) + k_2(\varphi(t), \hat{\theta}_2)$, will then be a better approximation of the system. This procedure is repeated until the value of the criterion function (10.2) does not decrease significantly anymore, or until the resulting model is satisfactory. One also has to take into consideration the risk of *overfitting*, which means that the model might start to adjust to the particular noise realization, if given too many degrees of freedom (e.g., too many hinge functions).

In addition to this algorithm, one can include a *refitting* procedure: When having added the hth hinge function, the previously added hinge functions are refitted one by one, so that the ith function $k_i(\varphi, \theta_i)$, $i \leq h$, is fitted to the residual $y(t) - \sum_{j \neq i} k_j(\varphi(t), \hat{\theta}_j)$. This procedure can also be iterated until no further improvement can be observed. Note that the criterion function will not increase by the refitting, so there is no risk of getting stuck in limit cycles.

The advantage of this approach of adding one, e.g., hinge function at a time, compared to the previous one of identifying all parameters simultaneously, is that each optimization problem is easier to solve. On the other hand, one has to solve several optimization problems instead of just one. When the number of regions/hinge functions of the real system is unknown, this approach may also have an advantage compared to the previous one. Just as for the first category, there is a risk of getting stuck in a local minimum.

Breiman [22] introduced the hinging hyperplane models and the algorithms described above for use in function approximation. The algorithm for estimating one single hinge function is called the *hinge-finding algorithm.* It starts by assuming that the dataset is partitioned into two subsets, and then estimates local affine models to each of the two sets. These affine models constitute the two submodels of a hinge function. From the fact that the hinge function should be continuous, the hinge of the hinge function can be determined. However, this new hinge may very well partition the data into two subsets which are different from the original subsets. Therefore, the procedure can be repeated using the new partition, and in this way we iterate until the same partitions are obtained in two consecutive iterations. Then a local minimum of $V^1(\theta_1, Z_1^N)$ is found.

The problem with this algorithm is that there is no guarantee for convergence. One can get stuck into limit cycles, or one can get least-squares problems that do not have a unique solution, which corresponds to a partition where one of the regions does not contain enough data points.

The identification algorithms in [22] were analyzed further in **Pucar and Sjöberg [124]**, and it was shown that the hinge-finding algorithm is a special case of a (nondamped) Newton algorithm, as presented in Section 8.2. It was also modified to guarantee convergence, yielding a damped Newton algorithm. Pucar and Sjöberg also give a greedy algorithm, where – just as in the previously described algorithms – one hinge function at a time is added. The model is written as a *weighted* sum of the hinge functions, and when a new function is added, its parameters are estimated together with the weights of the other hinge functions, while all other parameters are kept fixed. Pucar and Sjöberg compare the performance of this greedy function with the performance of the algorithm mentioned in the previous section for a noiseless example, and it turns out that the simultaneous estimation of the parameters performs better when the true model is within the model class (for this specific example).

Julián et al. [77] use a similar algorithm, but for Chua's canonical representation. They also mention the possibility of adding several hinge functions at a time.

Hush and Horne [69] use what they call *hinging sigmoid (HS)* functions for function approximation. A HS function is defined by

$$\sigma(\varphi) = \begin{cases} \theta_+ & \varphi^T\theta \geq \theta_+ \\ \varphi^T\theta & \theta_- \leq \varphi^T\theta \leq \theta_+ \\ \theta_- & \varphi^T\theta \leq \theta_- \end{cases}$$

Like in [22], one HS function is added at a time, and is fitted to the residual from the previous estimations. A refitting procedure similar to the one in [22] is also suggested. For the fitting of a single HS function three algorithms are proposed, of which the first is due to Breiman and Friedman [23]. All the algorithms make use of the fact that once the partition is known, finding the parameters θ, θ_+ and θ_- is a least-squares problem with linear constraints (corresponding to the requirement that the partition should not change). A similar approach is also presented in Section 11.7.

The Breiman/Friedman algorithm is analogous to the hinge-finding algorithm in [22] mentioned above, only for a different model class. It is a Newton algorithm, where in each iteration the updated parameter estimate is directly placed at the minimum of the local quadratic function. As for the hinge-finding algorithm, there is no guarantee for convergence for this algorithm. To avoid convergence problems, one should be able to modify the step size of the parameter update in a way similar to what is being done with the hinge-finding algorithm in [124].

The second algorithm in [69] also starts by assuming a partition. Given the partition, it solves the obtained QP problem including the linear constraints, and then detects if there are any data points at the borders of the partition. In that case, these data points are assigned to the other region (as long as this does not destroy the positive definiteness of the quadratic objective function), and the new QP problem is solved. This is repeated until the objective function does not decrease anymore.

This algorithm is also related to the Newton algorithms. Here the minimum is found within the region where the criterion function is quadratic (which corresponds to preserving the current partition of data points), and then, if this minimum is situated at a border of the region, a new step is taken into one of the adjacent regions. This means that each iteration explores exactly one new partition, and that the parameter estimates only move between adjacent regions of the piecewise quadratic criterion function.

The third algorithm described in [69] is an extension of the second, and can basically be seen as a way to give good initial values to the second algorithm. It first fits an affine function $\varphi^T\theta$ to the data. This affine function is used as the initial middle (nonconstant) part of the HS function. The data points are ordered along the direction of the gradient of the affine function, and the hinges are initially placed at the two extreme points on each side. From this initial partition, algorithm 2 is used to find a local minimum. Then one data point at a time is taken from the middle region and assigned to the upper region (where $\varphi^T\theta \geq \theta_+$) to see if improvements can be made. After this the other hinge location is optimized, and finally algorithm 2 is performed again.

In the article [46] by **Ernst**, hinging hyperplanes are used together with a tree structure, where a hinge function is added to the subregion with the worst fit. This approach is related both to the tree structures in [55] and to the model trees in [148], and is explained in more detail in Section 10.1.4.

A similar approach is taken by **Johansen and Foss [71]**. The models that are used resemble the hinging hyperplane trees, in the sense that the different subregions are partitioned hierarchically: The state-space is split by a hyperplane, and each of the two new regions are independently partitioned etc. To determine the position of the first hyperplane, their algorithm starts by performing an exhaustive search over models with $2+k$ regions. The best of these models determines where the first hyperplane should be placed. Having fixed that, the algorithm determines the second partitioning hyperplane by an exhaustive search over models with $3+k$ regions etc. Since the authors use interpolation between the different submodels, the resulting model is not a piecewise affine system, but the method could be used also for piecewise affine systems.

Heredia and Arce [64] use continuous piecewise affine systems with rectangular subregions with the borders parallel to the coordinate axes, and a somewhat different representation called *Continuous Threshold Decomposition*. Each border is defined by a threshold, and the authors use the same set of thresholds for all variables. They propose a method where one threshold at a time is added using a numerical method. An Akaike criterion [94] is used to decide how many thresholds should be added.

10.1.3 Finding Regions and Models in Several Steps

The last two categories are related in the sense that identifying the different regions and the different subsystems are done separately, in different steps. Many of the methods found in the literature are constructed in a similar way:

- Local affine models are used to find the partition of the data.
- After the partition is done, the parameters of the affine subsystems can easily be identified.

The local affine models in the first step can either be used to cluster the data points into different clusters, in which all data points have similar model parameters, or they can be used together with a change detection algorithm, in order to find the mode switchings.

One example of such an approach is found in [48] by **Ferrari-Trecate et al.** They consider PWARX systems, as defined in Section 9.2.1. Their identification algorithm consists of five steps. The first step is to estimate local affine models around each data point. This is done using the c data points that are closest to the point for which the local model is estimated, where c is a constant specified by the user. Then a measure of the "confidence level" for each local estimation is computed, based on how scattered the points used for the estimation are, and on the empirical covariance matrix for the estimation. After this, a K-means algorithm is used to cluster the data points (see [16], or [41] for an overview of different clustering techniques), based both on their position and on the parameters of the local model. The two last steps are to estimate an affine submodel for each cluster, and to try to find hyperplanes that separate the clusters from each other.

Another approach is proposed by **Bemporad et al. [10]**. The authors consider general PWARX systems, but with unknown but bounded (UBB) noise (see Section 1.1.3). The number of submodels is not fixed. The approach consists of three main steps. To get the initial assignment of data samples to different submodels and an initial estimate of $\theta(v_j)$ (and the number of submodels), an algorithm is used to approximately find the largest subset of inequalities of the form $|\varphi(t)^T\theta(v_1)-y(t)| \leq \delta$ that has a feasible solution. Having got an estimate of $\theta(v_1)$, the process is repeated on the remaining data samples to get $\theta(v_2)$, etc.

In the second phase the points are reassigned to different submodels according to the following scheme:

- Points that satisfy $|\varphi(t)^T\theta(v_i) - y(t)| \leq \delta$ for only one i are assigned to the corresponding class.
- Points that satisfy $|\varphi(t)^T\theta(v_i) - y(t)| \leq \delta$ for several values of i are called *undecidable*, and are not assigned to any class.
- Points that do not satisfy $|\varphi(t)^T\theta(v_i) - y(t)| \leq \delta$ for any value of i are called *infeasible*, and are not assigned to any class.

Given the new classification, $\theta(v_1), \ldots, \theta(v_{n_v})$ are reestimated. This is iterated until convergence. During this procedure, submodels that are too similar to each other are merged. Also, submodels containing too few points are removed. In this way, the number of submodels can be kept low.

In the last step, an algorithm similar to the first step is used to find the separating hyperplanes, giving the least possible number of misclassified points. Optionally, support vector machines (SVM) can then be used to optimize the location of the separating hyperplanes.

Hoffmann and Engell [65, 66] consider a hybrid system model with several modes, where the dynamics of each mode is affine. When the state reaches one of several switching regions, the system switches to another mode according to a table. State jumps (see Section 9.3) are also allowed, and are modelled by affine functions, one for each switching region.

The identification process is divided into several steps: First the mode switches are detected with a standard change detection algorithm, where a model

$$x(k+1) = Ax(k) + Bu(k) + d(k)H(k-K)$$

is fitted to a window of the time series. When $\|d(k)\|$ exceeds a given threshold, this indicates that a mode switching has occurred.

In the second step, the switching points are clustered and classified as belonging to different switching regions, and the regions are approximated by hyperspheres. Then a possible mode sequence is found. From this a discrete model for the different mode switchings is built.

In the last steps, the affine functions for the state jumps and the continuous dynamics of each mode are estimated using standard linear identification techniques.

One problem with the approach by Hoffmann and Engell is that the system model is only valid along the trajectories of the experimental data. This is due to the very flexible hybrid model, which does not allow any conclusions about unexplored parts of the state-space to be drawn, since there might for instance be an undiscovered switching region anywhere in these parts.

In **Münz and Krebs [109]**, the authors propose to, as a first step, use separating hyperplanes which are parallel to the coordinate axes. The position of these are optimized one by one in an iterative fashion. Having done this, each rectangular subregion is divided into simplices for which affine subsystems are identified.

In **Skeppstedt et al. [143, 144]**, an online approach is considered. The models taken into consideration are of PWARX type. A multiple-model recursive parameter estimation algorithm called XAFFM is used to estimate the current parameter value. XAFFM does not only give estimates of the parameters, but also of the covariance matrix and posterior probability of the parameter estimate.

By applying Bayes' rule, using information about the operating point and the output of the sample, it is determined to which of the affine submodels that the current sample most likely belongs. If the posterior probability of the current estimated parameter value is high enough, the parameter value of the submodel is updated. However, if the estimate of the current parameter value and the previously estimated submodel parameter value differ too much, no updating is done. Instead, if this happens repeatedly, a new submodel is added to the total model.

After the update of the dynamics, the separating hyperplanes are updated by a standard procedure for linear discrimination.

The *Switching Dynamical Systems* considered in **Petridis and Kehagias [119]** do not really belong to the model classes considered so far in this chapter. They consist of a number of subsystems, between which switchings occur with a certain probability for each time step. The dependence between the current state and mode, which is inherent in the models considered previously, is not considered in [119].

For these systems, an iterative identification algorithm related to the algorithm by Skeppstedt et al. is presented. Initially, the number of submodels K is set to one, $t = 1$ and a threshold ϵ is chosen. Then the following steps are executed iteratively:

1. For $k = 1, \dots, K$, compute predictions $\hat{y}_k(t)$ of $y(t)$ using the current models.
2. For $k = 1, \dots, K$, if $|y(t) - \hat{y}_k(t)| < \epsilon$, let $y(t)$ be assigned to model k and go directly to step 3. If no model satisfies the requirement, let $K = K + 1$, add one model and assign $y(t)$ to this model.
3. Estimate all submodels using the data assigned to them. Let $t = t + 1$ and go to 1.

Some convergence results are given, in the sense that the ratio between the falsely and the correctly classified data points will asymptotically approach 0 with probability 1.

In **Gelfand and Ravishankar [55]**, a tree structure is used to partition the state-space. Each node represents a subregion, and to each node the best affine model of the particular subregion is associated. The structure will be considered in some more detail in Section 10.1.4. To update the parameters and to prune the tree, an online, gradient-based, two-step updating procedure is used.

Medeiros et al. [104, 105] use an algorithm called GRASP (Greedy Randomized Adaptive Search Procedure) to iteratively and heuristically find good partitions of the state-space. When a partition is given, the remaining parameters are estimated using least squares, and the mean squared error is taken as a measure of how good the partition was.

Related to this category are the *Local Learning* methods of [110, 111] by **Murray-Smith and Gollee**, also considered in [112]. They use local linear models, which are interpolated to get the total system model. The interpolation implies that the resulting systems are not piecewise affine systems. The identification process follows an iterative algorithm, where the model structure is first determined using *a priori* knowledge. Then the model parameters are identified locally for each submodel. After this, the algorithm determines where the model structure needs improvement, and the structure is adjusted. The last two steps are then iterated until the result is satisfactory.

There are also some relationships to the local modelling methods in Part I, and similar methods [122, 141]. In fact, if estimating a local linear model from a given set of estimation data, using a weighted least squares criterion and a piecewise constant weight function, the resulting global model will be a piecewise affine system (even if it consists of a very large number of submodels). The shapes of the different regions and the submodels are determined by the choice of weight function, which (as described earlier) becomes a trade-off between incorporating as much data as possible to decrease the variance of the estimate, and building a model which is as local as possible to avoid a biased model.

10.1.4 Using Predetermined Regions

One possible approach is to partition the state-space in a regular grid, with one affine subsystem for each region in the grid. Since both the partitioning of the state-space and the identification of the subsystems become very easy (given enough experimental data in each region), this approach might work well for low-dimensional systems. However, the number of regions grows exponentially with the dimensions, so both the computational complexity and the need for experimental data are exponential in the number of dimensions, making the approach infeasible for higher-dimensional systems.

One example of this kind of approach is given by **Billings and Voon [15]**, where the state-space is partitioned into rectangular regions, with sides parallel to the coordinate axes. Another example is found in [78] by **Julián** (see also [76]). Here the author considers nonlinear function approximation using the HL CPWL class (see Section 9.2.3). Also **Simani et al. [140]** use a given simplicial partition and identify affine models for each subregion. Some of the noise characteristics is also identified.

Instead of deciding on the partition completely independently of the experimental data, another option might be to take the distribution of the regression vectors into account. In this way, the state-space can be partitioned in such a way that each region gets a proper amount of experimental data to allow for the estimation of an affine subsystem, and the exponential complexity can be avoided. The approach in [26] and the model trees proposed in [148] fall into this category.

In [26] by **Choi and Choi**, a method is proposed, where the state-space is first partitioned into hypertriangles, with experimental data points in each corner. For each hypertriangle an affine subsystem is fitted. Since only $n+1$ points are used to fit each n-dimensional subsystem, this method is quite sensitive to noise. Therefore, the authors also suggest to use an unsupervised learning algorithm with fewer cells to get the partition. Then the problem of estimating the affine subsystems becomes a standard identification problem.

In the neural network literature, one can find several other examples of methods that start by clustering the data, and then build local linear models for each cluster; see, e.g., [82, 103, 152, 154].

The model trees proposed by **Strömberg et al. [148]** stem from the Classification and Regression Trees (CART) [24]. Here, a binary tree is built, where each node represents a region of the state-space, and the regions of the children nodes form a partition of the region of the parent node. Then one linear subsystem is associated with each leaf of the tree.

The obvious disadvantage with just considering the distribution of the regression vectors, and not the corresponding y values, is that a set of data which really should belong to the same subsystem might be split arbitrarily. To somewhat make up for this drawback, one option is to make the partition rather fine (with many regions), and then afterwards merge adjacent regions with similar subsystems (cf. the pruning strategy in [148]). A minor problem with this approach is that the merged regions may be of varying shapes and sizes, and it might be difficult to represent them efficiently. For models represented as a tree structure, this is no problem as long as the adjacent regions are also adjacent leaves in the tree. Then, all that needs to be done is to erase the leaves and let their parent node become a new leaf representing the merged region. However, if the regions belong to completely different branches of the tree, it may even be hard to determine whether they are adjacent or not.

One idea would be to use Breiman's Hinge Finding Algorithm (or the modified version in [124]) to find good partitions for the model trees. This would reduce the risk that two adjacent regions with similar dynamics would end up in completely different branches of the tree. This is what is done by Ernst in [46].

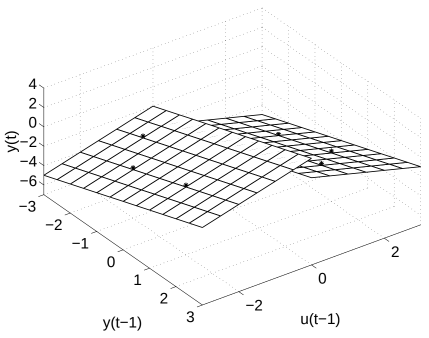
In Gelfand and Ravishankar [55], the splitting of a region is decided based on the best affine model for that region. A threshold is chosen as the mean of the experimental outputs in that region. If the predicted output from the best affine model of the region exceeds this threshold, the data sample is assigned to one of the children nodes; otherwise it is assigned to the other. In this way, the nodes are assigned approximately the same number of data samples, but the problem of adjacent regions with similar dynamics being split into different branches is not necessarily avoided.

10.2 Discussion

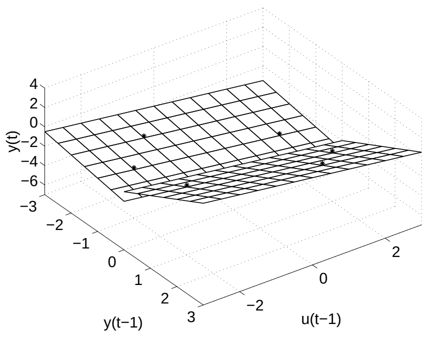
The recurring problems with piecewise affine system identification concern how to find good solutions (a global minimum or a good local minimum) to the identification problem given a model structure, and how to keep the computational complexity and model complexity (number of model parameters) low. The computational complexity for a given algorithm is a function both of the number of regions/subsystems, the number of experimental data, and of the number of dimensions of the regression vector. Hence, there are many trade-off situations when comparing different methods and tuning parameters for given algorithms.

One such issue was mentioned already in Section 1.2.2: The choice between choosing an *a priori* grid or estimating the partitioning together with the subsystems. The first option leads to a simple estimation process, but to get a good result, the numbers of regions and data needed will increase exponentially with the number of dimensions. Hence, this approach may be good for low-dimensional systems, but for systems of higher dimensions another approach might be preferred.

Let us take a closer look at the requirements on the data. Assume that the true system is piecewise affine. To be able to identify a system in a satisfactory way, several or all modes (depending on the system structure) have to be excited by the data, and furthermore, each mode has to be excited for a sufficiently long time. A set of data that, given the model structure and a criterion function, uniquely determines the system will here be called *persistently exciting* (cf. [94]). For example, for the case of predetermined regions and mutually independent affine subsystems (as in a PWARX system), we need at least as many linearly independent data samples in each mode as the dimension of the regression vector. When the regions are to be estimated as well, we might even need more data, as the following example shows.



(a) True system.



(b) False system.

Figure 10.1: The system that is identified in Example 10.1.

Example 10.1 (Insufficient data) *Consider the system*

$$y(t) = \begin{cases} \begin{pmatrix} 1 & -1 & 1 \end{pmatrix} \varphi(t) & \varphi_3(t) < 0 \\ \begin{pmatrix} -1 & -1 & -1 \end{pmatrix} \varphi(t) & \varphi_3(t) \geq 0 \end{cases}$$

where $\varphi(t) = \begin{pmatrix} 1 & -y(t-1) & u(t-1) \end{pmatrix}^T$ *(and hence* $\varphi_3(t) = u(t-1)$*). Suppose that we are given the data samples* $(\varphi(t), y(t))$*:*

$$\Big\{ (\begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}, -2),\ (\begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}, -4),\ (\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}, 0),$$

$$(\begin{pmatrix}1\\-1\\-2\end{pmatrix},0),\ (\begin{pmatrix}1\\2\\-1\end{pmatrix},-2),\ (\begin{pmatrix}1\\1\\-2\end{pmatrix},-2)\Big\}$$

Given the true partition of the state-space, these points determine the system uniquely, since there are three linearly independent samples in each region (see Figure 10.1(a)). However, we could alternatively choose another, incorrect partition and get a completely different system. An example of this is shown in Figure 10.1(b), where the system

$$y(t) = \begin{cases} \begin{pmatrix}-2.5 & -1.5 & -0.5\end{pmatrix}\varphi(t) & \varphi_2(t) < 0 \\ \begin{pmatrix}-3.5 & 0.5 & -0.5\end{pmatrix}\varphi(t) & \varphi_2(t) \geq 0 \end{cases}$$

is shown to match the experimental data perfectly.

The need for enough data from each subsystem limits the number of subsystems that would be feasible to identify. Therefore, for larger examples it becomes advantageous to allow the regions to be formed according to the data, to allow for fewer regions and parameters.

Another trade-off issue is the one concerning the choice between the three first categories of the previous sections. The first category – simultaneous identification of all parameters – is perhaps the most straightforward and general option, since any numerical optimization algorithm can be applied. The other categories are more heuristic, and the problem is split up into several subproblems, each of which might be easier to solve than tackling the entire problem directly. Generally, it is hard to say which strategy should be preferred, regarding the quality of the solutions and the computational complexity. Some comparisons exist, e.g., the one in [124] mentioned previously, but no extensive study comparing several methods and their performance has been found by the author.

There is also a trade-off between model complexity and the result quality. The more degrees of freedom that are built into the model structure, the closer the model can approximate the experimental data. However, since in practice data are more or less corrupted by noise, a too large degree of model flexibility might cause the model to adjust to the actual noise realization, thereby causing overfitting. This is a general problem of system identification, occurring not only for piecewise affine systems.

Finally, there is a trade-off between the computational complexity of the algorithm chosen, and the quality of the resulting model. This is most obvious in the algorithms where all parameters are identified simultaneously. The risk of ending up in a local minimum varies, depending on what numerical optimization algorithm is chosen. For example, having chosen a Gauss-Newton method, a simple way of increasing the chance of finding at least a good suboptimal model is to run the Gauss-Newton method several times, starting from different initial values. This will (on average) increase the quality of the solution, but at the same time increase the computational complexity. In the next chapter we will see another example of improving the result at the cost of increased complexity.

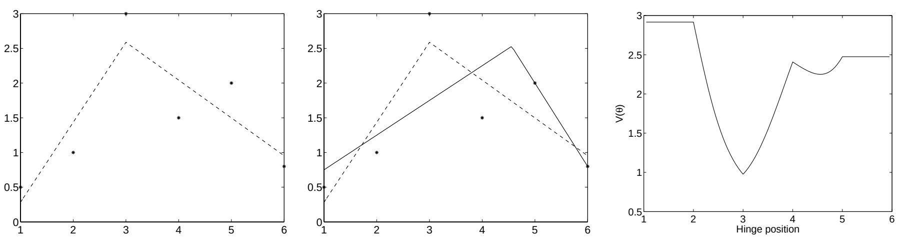
11

PWA Identification Using MILP/MIQP

The algorithms described in Chapter 10 all have one property in common: They cannot guarantee that the optimal solution, i.e., the global minimum of (10.2), is found. In this chapter, a type of algorithms that reach the globally optimal solution in a finite number of steps is presented. The algorithms are based on *mixed-integer linear/quadratic programming (MILP/MIQP)*, which is described in Section A.4 in the Appendix. The basic algorithms are described in Section 11.1, and some extensions are presented in Section 11.2.

Although it was shown, e.g., that the hinge-finding algorithm, proposed in [22] and described in Section 10.1.2, converges to the global minimum for a noiseless system consisting of one single hinge, local minima can lead to problems even in very simple cases for noisy systems or systems with multiple hinges. The following example illustrates this.

Example 11.1 *Consider the problem of fitting a hinge function to the six data samples given in Figure 11.1(a), using a 2-norm criterion. Figure 11.1(a) also shows the corresponding globally optimal function, with the optimal cost* 0.98*. In Figure 11.1(c) the cost is plotted as a function of the position of the hinge (recall that once the hinge position is known, the identification problem is simple), and we can see that there is a local minimum between* 4 *and* 5 *with the cost* 2.25*. The corresponding function is plotted in Figure 11.1(b). Furthermore, simple*



(a) Data samples (∗) and globally optimal model (dashed).

(b) Locally (solid) and globally (dashed) optimal model.

(c) Cost as a function of hinge position for Example 11.1.

Figure 11.1: Identification of a single hinge function.

calculations show that Breiman's method [22] will not converge to the optimal solution (regardless of the initial value), but will in most cases converge to the local minimum. The modified method provided in [124] will converge to the global optimum if starting sufficiently close to it, but will converge to the local optimum if the hinge is originally placed between 4 *and* 5.

It should be said from the beginning that the computational complexity of the globally optimal algorithms presented in this chapter is such that they are not a realistic alternative for identifying larger piecewise affine systems, or using large sets of experimental data. This subject is treated in more detail in Section 11.3. Another question is if it is really necessary to find the global optimum in practice, or if a good suboptimal solution will do. This issue becomes even more relevant when using piecewise affine systems as approximations to general nonlinear systems, and when the experimental data are corrupted by noise. Nevertheless, algorithms that guarantee convergence to the global optimum in finite time may still be of theoretical interest.

Furthermore, there are special cases where it might be feasible to use identification algorithms based on MILP/MIQP. One example is the problem of identifying a piecewise affine Wiener model, which is described in Section 11.4. Another case is when a system only seldom changes modes, which allows for a change detection algorithm to be used to reduce the complexity. This case is considered in Section 11.6. Finally, one idea is to use an MILP/MIQP solver to get a decent, suboptimal solution, which can be used as an initial value for a conventional optimization algorithm. This is studied in Section 11.5.

The way we formulate the identification problem can also be used together with different methods similar to some of the approaches in Chapter 10. We will consider two Newton-like algorithms in Section 11.7.

11.1 Formulating the Identification Problem as an MILP/MIQP Problem

The system class we will consider in this section is the class of hinging hyperplane systems given by (9.10) and repeated here for convenience:

$$y(t) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i, 0\} + e(t) \tag{11.1}$$

Recall that this is shorthand notation for

$$y(t) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M^+} \max\{\varphi^T(t)\theta_i, 0\} - \sum_{i=M^+ +1}^{M} \max\{\varphi^T(t)\theta_i, 0\} + e(t)$$

where we assume M^+ and M to be known. As mentioned in Section 9.2.2, two properties of this class are that the piecewise affine function is continuous, and that the affine subsystems are separated by the hyperplanes $\varphi^T(t)\theta_i = 0$.

Letting $\theta = \begin{pmatrix}\theta_0^T & \dots & \theta_M^T\end{pmatrix}^T$, the predicted output $\hat{y}(t|\theta)$ becomes

$$\hat{y}(t|\theta) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i, 0\} \tag{11.2}$$

As in Chapter 8, we are given $y(t)$, $\varphi(t)$, $t = 1, \dots, N$, and would like to minimize the prediction error according to some criterion function, i.e.,

$$\min_{\theta} \quad V(\theta, Z_1^N) = \sum_{t=1}^{N} \ell(\varepsilon(t, \theta)) \tag{11.3}$$

where

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t|\theta) = y(t) - \left(\varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i, 0\}\right)$$

In this chapter, we will use

$$\ell_2(\varepsilon) = \varepsilon^2$$

and

$$\ell_1(\varepsilon) = |\varepsilon|$$

The main question of this section is: How can we rewrite these problems to fit into the mixed-integer programming (MILP/MIQP) framework? If we would be able to formulate the piecewise affine system identification problem as an MILP or MIQP, then the optimal solution could be found in a finite number of steps, e.g., by using the algorithm described in Section A.4. There are two steps to accomplish that: To rewrite (11.3) as a linear/quadratic function of the unknowns by introducing auxiliary variables, and to rewrite the constraints of the auxiliary variables as linear inequalities plus the constraint on some of the variables to be discrete.

To be able to make the reformulation into a MILP or MIQP, we will need bounds on the products $\varphi^T(t)\theta_i$. Therefore, we will make the assumption that there are known bounds for the parameters θ_i. These bounds could be in any norm without affecting the later results, but for simplicity the Euclidean norm is chosen here, i.e.,

$$\|\theta_i\| \leq U_{\theta_i}, \quad i = 1, \ldots, M$$

This implies that $\varphi^T(t)\theta_i$ are bounded:

$$L_i(t) \triangleq -\|\varphi(t)\|U_{\theta_i} \leq \varphi^T(t)\theta_i \leq \|\varphi(t)\|U_{\theta_i} \triangleq U_i(t)$$

This is not a serious restriction, since the bounds can be chosen arbitrarily large. However, the tighter the bounds, the more efficiently the optimization can be performed. Other types of bounds that could be imagined include componentwise bounds on θ_i, i.e.,

$$L_{\theta_i} \preccurlyeq \theta_i \preccurlyeq U_{\theta_i}$$

where $\preccurlyeq$ denotes componentwise inequalities. If this type of bounds is used, the definitions of $L_i(t)$ and $U_i(t)$ should be changed accordingly.

11.1.1 Reformulating the Criterion Function

Let us start with the criterion function. The problem here is that $\hat{y}(t|\theta)$ is a nonlinear function of θ. To eliminate that problem, introduce the auxiliary variables

$$z_i(t) = \max\{\varphi^T(t)\theta_i, 0\}, \quad i = 1, \ldots, M, \ t = 1, \ldots, N \tag{11.4}$$

and regard these as unknown variables in our minimization. The criterion function for $\ell_2(\varepsilon)$ now takes the form

$$V_2 = \sum_{t=1}^{N} \left(y(t) - \left(\varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm z_i(t) \right) \right)^2 \tag{11.5}$$

which is quadratic in the unknowns θ_0 and $z_i(t)$. The corresponding function for $\ell_1(\varepsilon)$ is

$$V_1 = \sum_{t=1}^{N} \left| y(t) - \left(\varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm z_i(t) \right) \right| \tag{11.6}$$

This function is unfortunately neither linear, nor quadratic. However, there is a standard trick for rewriting functions like this as a linear function with linear constraints. Introduce the *slack variables* $s_1, \ldots, s_N$. Then, minimizing V_1 is the same as solving

$$\begin{aligned} \min_{\theta,z,s} \quad & \sum_{t=1}^{N} s_t \\ \text{subj. to} \quad & s_t \geq y(t) - \varphi^T(t)\theta_0 - \sum_{i=1}^{M} \pm z_i(t) \\ & s_t \geq -\left(y(t) - \varphi^T(t)\theta_0 - \sum_{i=1}^{M} \pm z_i(t) \right) \\ & z_i(t) = \max\{\varphi^T(t)\theta_i, 0\} \end{aligned} \tag{11.7}$$

Now the criterion function is linear at the price of increasing the number of variables by N and the number of linear constraints by $2N$. However, in the context this is not a very serious complexity increase, since it is generally the discrete variables introduced in the next section that determine the complexity.

11.1.2 Reformulating the Constraints

By introducing the auxiliary variables $z_i(t)$, we have also introduced the additional nonlinear constraints (11.4). The next step in our reformulation process is to rewrite these constraints as linear constraints, with the help of additional discrete variables. To that end, define

$$\delta_i(t) = \begin{cases} 0 & \varphi^T(t)\theta_i < 0 \\ 1 & \varphi^T(t)\theta_i \geq 0 \end{cases} \qquad i = 1, \ldots, M \tag{11.8}$$

This means that we can write $z_i(t)$ as

$$z_i(t) = \delta_i(t)\varphi^T(t)\theta_i$$

Notice also that geometrically, $\delta_i(t)$ tells us on what side of the ith hinge $\varphi(t)$ is positioned. Now we can use the method described in [12] to transform the constraints (11.4) and (11.8) to linear inequalities. However, there is a more efficient way, which will now be described. This transformation method is related to the method of transforming boolean expressions into linear inequalities, described in [106], and should be possible to use in many applications, where similar transformations are needed.

Consider $z_i(t)$ and $\delta_i(t)$ for given i and t. In Figure 11.2, the feasible region for the two constraints (definitions) (11.4) and (11.8) is drawn as two solid lines. What we would like is to be able to express that region as the intersection between a polyhedron (the linear constraints) and the set $\delta_i(t) = \{0, 1\}$. The simplest way of doing this is to construct the convex hull (see Section A.3) of the feasible region (marked with dashed lines in Figure 11.2). This will both give the smallest polyhedron including the original feasible region and the smallest set of linear

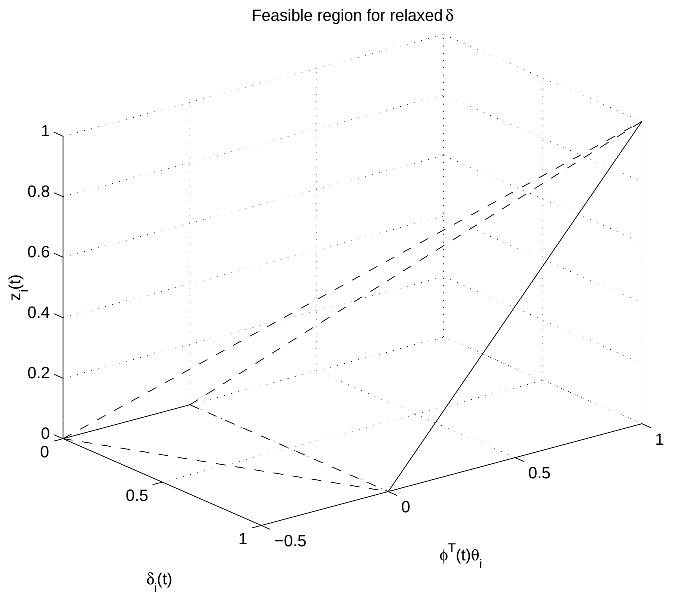


Figure 11.2: The feasible region for the constraints (11.4) and (11.8) in the $(\delta_i(t), \varphi^T(t)\theta_i, z_i(t))$ space (solid lines), and its convex hull (the dashed tetrahedron). Here we have assumed that $L_i(t) = -0.5$, $U_i(t) = 1$.

inequalities[*]. The fact that we get the smallest polyhedron is a general property of convex hulls, while the second property follows since four linear inequalities (together with the requirement $\delta \in \{0, 1\}$) are needed to define each of the two line segments of the feasible region. By choosing the linear inequalities to the ones defining the complex hull, both line segments are defined by the same inequalities.

Example 11.2 (Tightness of the linear constraints) *Consider the case*

$$\varphi(1) = \begin{pmatrix}1\\1\end{pmatrix}, \quad \varphi(2) = \begin{pmatrix}1\\0\end{pmatrix}, \quad \varphi(3) = \begin{pmatrix}1\\-1\end{pmatrix}$$
$$L_i(1) = -2, \quad L_i(2) = -1.25, \quad L_i(3) = -2$$
$$U_i(1) = 1, \quad U_i(2) = 2, \quad U_i(3) = 1$$

which gives the feasible region for θ_i shown in Figure 11.3. Here the upper bound $U_i(2) = 2$ for $\varphi^T(2)\theta_i$ will never be reached, since the bounds on $\varphi^T(1)\theta_i$ and

[*]Note, though, that this only holds as long as we neglect the influence that variables from different timepoints have on each other, and as long as the bounds $L_i(t)$ and $U_i(t)$ are as tight as possible. Otherwise, the bounds on $\varphi^T(t_0)\theta_i$ may be reduced if bounds on $\varphi^T(t)\theta_i$ for other timepoints limit the possible values of θ_i. This would make the feasible region smaller. Example 11.2 illustrates this.

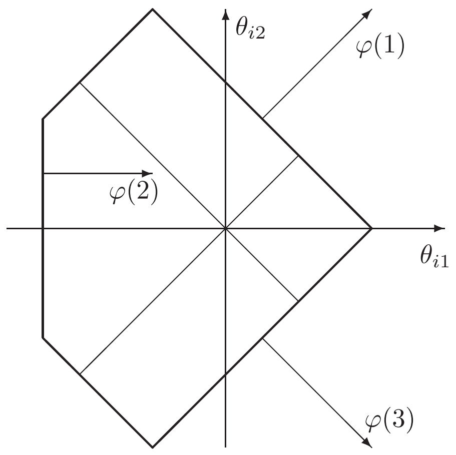


Figure 11.3: The feasible region (bounded by the thick outer lines) in Example 11.2. The thin lines show $\varphi^T(1)\theta_i = 0$ and $\varphi^T(3)\theta_i = 0$.

$\varphi^T(3)\theta_i$ also give an implicit bound on $\varphi^T(2)\theta_i$ (corresponding to the rightmost vertex of the feasible region). Furthermore, if $\varphi^T(1)\theta_i \geq 0$ (i.e., $\delta_i(1) = 1$), the lower bound $L_i(2) = -1.25$ for $\varphi^T(2)\theta_i$ (corresponding to the leftmost vertical line in Figure 11.3) will not be reached either. Hence, whether or not $L_i(2)$ is a tight lower bound on $\varphi^T(2)\theta_i$ depends on the value of $\delta_i(1)$.

At first sight, the convex hull in Figure 11.2 seems to be defined by

$$\begin{cases} z_i(t) \geq 0 \\ z_i(t) \leq U_i(t)\delta_i(t) \\ \varphi^T(t)\theta_i \leq z_i(t) \\ (1-\delta_i(t))L_i(t) + z_i(t) \leq \varphi^T(t)\theta_i \end{cases} \tag{11.9}$$

However, there is one problem in using these inequalities. According to (11.8), the point $(\delta_i(t), \varphi^T(t)\theta_i, z_i(t)) = (0, 0, 0)$ should not be part of the feasible region. This means that the convex hull is described by the union of all regions obtained by different positive μ in the following inequalities

$$\begin{cases} z_i(t) \geq 0 \\ z_i(t) \leq U_i(t)\delta_i(t) \\ (1-\delta_i(t))\mu + \varphi^T(t)\theta_i \leq z_i(t) \\ (1-\delta_i(t))L_i(t) + z_i(t) \leq \varphi^T(t)\theta_i \end{cases} \qquad \mu > 0 \tag{11.10}$$

which is a description that might be awkward to work with, due to the nonfixed parameter μ. Actually, this is more of a theoretical than a practical problem. The reason for this is best understood geometrically: When $\varphi^T(t)\theta_i = 0$, it means that

$\varphi(t)$ is lying on the ith hinge, and then it does not really matter to what side we assign it, i.e., we can let $\delta_i(t)$ be 0 or 1. Therefore, we can let $\mu = 0$.

Putting (11.5) and (11.9) together now gives us the following problem:

$$\begin{aligned} \min_{\theta,z,\delta} \quad & V_2 = \sum_{t=1}^{N} \left(y(t) - \left(\varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm z_i(t) \right) \right)^2 \\ \text{subj. to} \quad & z_i(t) \geq 0 \\ & z_i(t) \leq U_i(t)\delta_i(t) \\ & \varphi^T(t)\theta_i \leq z_i(t) \\ & (1-\delta_i(t))L_i(t) + z_i(t) \leq \varphi^T(t)\theta_i \\ & \delta_i(t) \in \{0,1\} \end{aligned} \tag{11.11}$$

This is an MIQP problem, and we are done with the reformulation.

In the same way, minimizing V_1 can be reformulated as an MILP problem using (11.7) and (11.9):

$$\begin{aligned} \min_{\theta,z,\delta,s} \quad & \sum_{t=1}^{N} s_t \\ \text{subj. to} \quad & s_t \geq y(t) - \varphi^T(t)\theta_0 - \sum_{i=1}^{M} \pm z_i(t) \\ & s_t \geq \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm z_i(t) - y(t) \\ & z_i(t) \geq 0 \\ & z_i(t) \leq U_i(t)\delta_i(t) \\ & \varphi^T(t)\theta_i \leq z_i(t) \\ & (1-\delta_i(t))L_i(t) + z_i(t) \leq \varphi^T(t)\theta_i \\ & \delta_i(t) \in \{0,1\} \end{aligned} \tag{11.12}$$

11.1.3 Restricting the Search Space

We have now reached the MILP/MIQP formulations, so in principle, we are ready to apply an MILP/MIQP solver to find the optimal parameter values. However, as noted in Section 9.2.2, the parameter values are not uniquely determined in the expressions above, which means that there will be several optimal solutions, all describing the same system. To avoid this, we can restrict the search space by requiring that $0 \leq w^T\theta_1 \leq \cdots \leq w^T\theta_{M^+}$ and $0 \leq w^T\theta_{M^++1} \leq \cdots \leq w^T\theta_M$ for a given constant (nonzero) vector w, just as in Section 9.2.2.

An interesting choice of w is $w = -\varphi(t)$ for some t, say, $t = 1$. This gives the inequalities

$$\begin{aligned}\varphi^T(1)\theta_{M^+} \le \cdots \le \varphi^T(1)\theta_1 \le 0\\ \varphi^T(1)\theta_M \le \cdots \le \varphi^T(1)\theta_{M^++1} \le 0\end{aligned} \tag{11.13}$$

that should be included in (11.11) and (11.12). In particular, we can set

$$\begin{cases}\delta_i(1) = 0\\ z_i(1) = 0\end{cases} \qquad i = 1, \dots, M$$

which means that we can exclude these $2M$ variables from the optimization, thus getting a somewhat smaller optimization problem.

It should be emphasized that this reduction of the search space not necessarily leads to a faster computation time, even if it often helps. This is something that needs to be investigated more thoroughly. See [159] for a discussion of the relations between number of variables, number of constraints, and computation time.

11.1.4 Some Examples

Let us now look at some examples of using MILP/MIQP for piecewise affine system identification.

Example 11.3 *In the following example, we consider a noiseless system described by*

$$\begin{aligned}y(t) = y(t-1) &- u(t-1) + 2\\ &+ \max\{y(t-1) + u(t-1) - 1, 0\}\\ &- \max\{u(t-1) - 1, 0\}\end{aligned} \tag{11.14}$$

which we can rewrite as

$$y(t) = \varphi^T(t)\theta_0 + \max\{\varphi^T(t)\theta_1, 0\} - \max\{\varphi^T(t)\theta_2, 0\} \tag{11.15}$$

with

$$\varphi(t) = \begin{pmatrix} 1\\ -y(t-1)\\ u(t-1)\end{pmatrix} \tag{11.16}$$

and

$$\theta_0 = \begin{pmatrix} 2\\ -1\\ -1\end{pmatrix}, \quad \theta_1 = \begin{pmatrix} -1\\ -1\\ 1\end{pmatrix}, \quad \theta_2 = \begin{pmatrix} -1\\ 0\\ 1\end{pmatrix} \tag{11.17}$$

which are the parameters to be identified. The set of experimental data consists of $(u(t), y(t))$, $t = 0, \dots, 12$, and can be shown to be persistently exciting (see Section 10.2). In Figure 11.4, the function to be identified and the data are plotted.

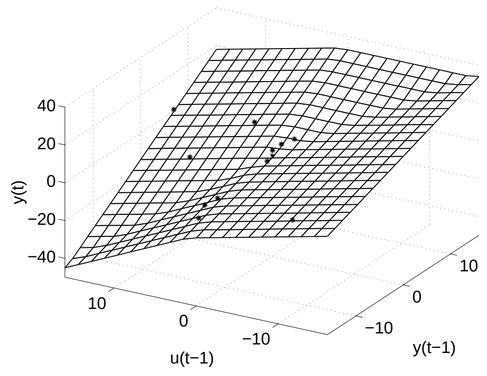


Figure 11.4: The function defined in (11.14) and the experimental data.

Table 11.1: The experimental data, and the correct values of $\delta_i(t)$.

t	$u(t-1)$	$y(t-1)$	$\delta_1(t)$	$\delta_2(t)$
1	0	0	0	0
2	0	2	1	0
3	5	5	1	1
4	16	7	1	1
5	10	0	1	1
6	2	-8	0	1
7	3	-9	0	1
8	2	-12	0	1
9	-10	-13	0	0
10	0	-1	0	0
11	0.5	1	1	0
12	-1	3	1	0

Let us now formulate this system in a form suitable for the MILP/MIQP identification algorithms. Following (11.4) *and* (11.8), *we start by introducing*

$$z_i(t) = \max\{\varphi^T(t)\theta_i, 0\}, \quad i = 1, 2$$

and

$$\delta_i(t) = \begin{cases} 0 & \varphi^T(t)\theta_i < 0 \\ 1 & \varphi^T(t)\theta_i \geq 0 \end{cases}, \quad i = 1, 2$$

Table 11.1 shows the correct values of $\delta_i(t)$, which show that the data are well distributed over the different modes. Now, depending on what criterion function we would like to use, we can formulate the identification problem exactly as in (11.11) *or* (11.12). *We can also add the restrictions* (11.13). *Then a conventional MILP/MIQP solver can be used to identify the parameters. The computation time* *for this experiment was a few seconds, and a few hundred nodes in the MILP/MIQP search trees were visited before the correct parameter values were found (100-800 nodes, depending on implementation and strategy of the MILP/MIQP solvers).*

The following example is taken from [13].

Example 11.4 *Consider the system*

$$\begin{aligned} y(t) = 0.8y(t-1) + 0.4u(t-1) - 0.1 \\ + \max\{-0.3y(t-1) + 0.6u(t-1) + 0.3, 0\} \end{aligned} \tag{11.18}$$

The model is identified on the data reported in Figure 11.5(a), by solving an MILP with 66 variables (of which 20 integers) and 168 constraints. For this example, the translation into inequalities given by [12] was used instead of the method in (11.9). *The problem was solved by using* Cplex *[70] (1014 LP solved in 0.68 s on a Sun Ultra 10 running* Matlab *5.3), and, for comparison, was solved again using BARON [139] (73 LP solved in 3.00 s, same machine), which results in a zero output prediction error (Figure 11.5(b)). The fitted hinging hyperplane model is shown in Figure 11.6. By adding a random Gaussian noise with zero mean and variance 0.1 on the measured output $y(t)$, the following model*

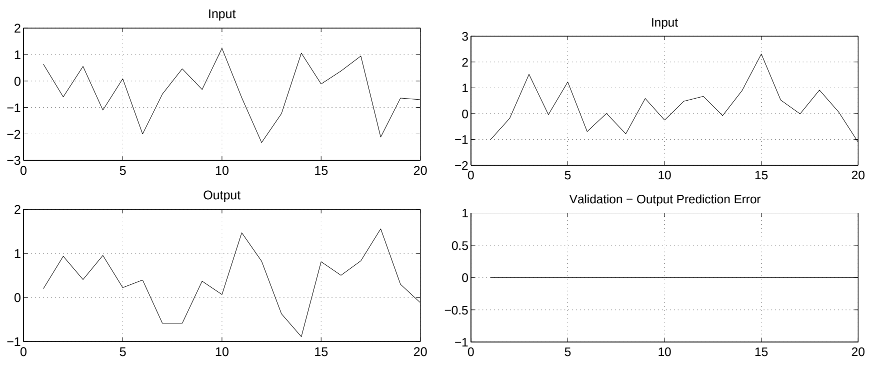
$$\begin{aligned} y(t) = 0.8315y(t-1) + 0.3431u(t-1) - 0.2014 \\ + \max\{-0.3391y(t-1) + 0.6205u(t-1) + 0.3977, 0\} \end{aligned} \tag{11.19}$$

was identified in 1.39 s (3873 LP solved) using Cplex *(7.86 s, 284 LP using BARON) on the estimation set reported in Figure 11.7(a), and produces the validation data reported in Figure 11.7(b). For comparison, a linear ARX model was identified on the same estimation data, with the following result*

$$y(t) = 0.8250y(t-1) + 0.7217u(t-1) \tag{11.20}$$

The results for validation data are found in Figure 11.8 (higher order ARX models did not produce significant improvements). Clearly, the error generated by driving the ARX model in open-loop with the validation input $u(t)$ is much larger than for the hinging hyperplane model, and would not make (11.20) *suitable for instance for formal verification tools, where a good performance of open-loop prediction is a critical requirement.*

In Figure 11.9 we compare the performance in terms of LP/QPs and total computation time of the linear criterion (11.6) *versus the quadratic criterion* (11.5). *The reported numbers are computed on a Sun Ultra 60 (2×360 MHz) running* Matlab *5.3 and the solver BARON [139], by averaging the result of ten estimation data sets generated by feeding random Gaussian inputs $u(t)$ and zero output noise to system* (11.18).



(a) Estimation data. (b) Validation data.

Figure 11.5: Identification of model (11.18) – noiseless case.

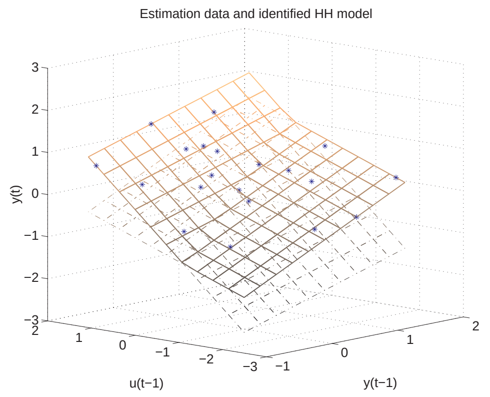
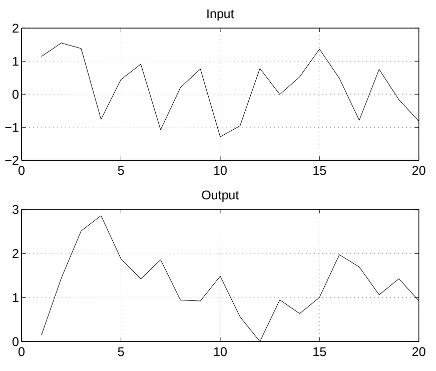


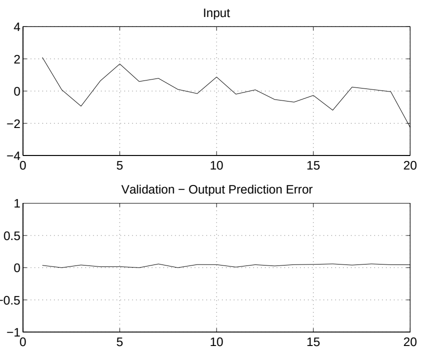
Figure 11.6: Identification of model (11.18) – noiseless case. Identified hinging hyperplane model.

11.2 Extensions

The MILP/MIQP formulations in Section 11.1 assumed that the system could be described by hinging hyperplane functions with a known number of max functions (M known), and known signs of the max functions (M^+ known). In this section, several extensions will be considered. Firstly, the case when M^+ is not known will be discussed in Section 11.2.1. Secondly, in Section 11.2.2 we will extend the hinging hyperplane models by allowing various forms of discontinuities. We will also consider a kind of robust hinging hyperplane models in Section 11.2.3. In the last extension (in Section 11.2.4), the identification problem for general PWARX systems (described in Section 9.2.1) will be considered. The extensions in Sections



(a) Estimation data.



(b) Validation data.

Figure 11.7: Identification of model (11.18) – noisy case, $y(t)$ is perturbed by a random Gaussian noise with zero mean and variance 0.1.

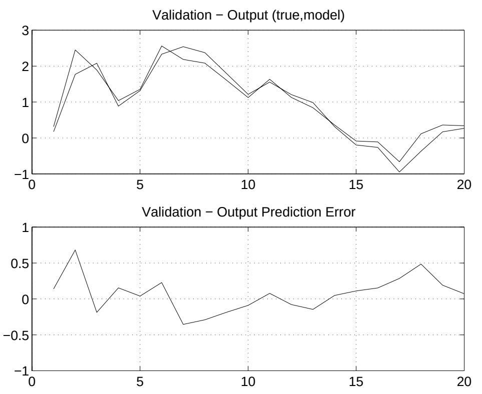


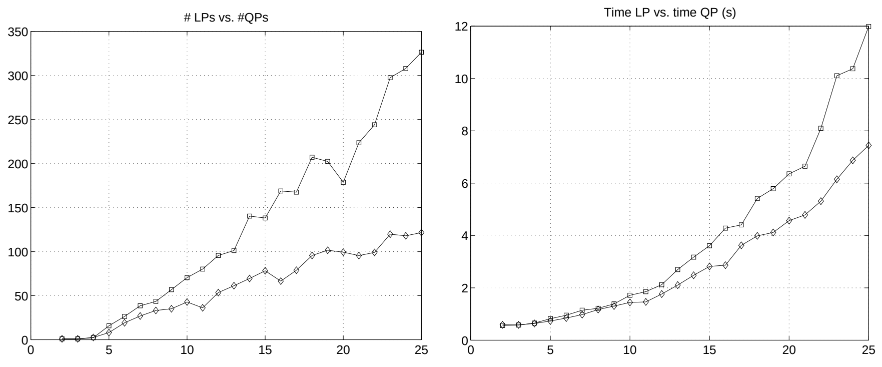
Figure 11.8: Result of identification of a linear ARX model – same estimation and validation data as in Figure 11.7.

11.2.2 and 11.2.3 are taken from [13].

Only the quadratic criterion function V_2 will be considered. The reformulation of V_1 follows immediately, by using slack variables as in (11.7).

11.2.1 Unknown Number of Positive Max Functions

When M^+ is unknown, the sign of each max function has to be determined. Here, two ways of doing this will be described. In the first method, we introduce M



(a) Average number of LPs and QPs.

(b) Average computation time (Sun Ultra 60 (2×360 MHz) running Matlab 5.3 and the solver BARON).

Figure 11.9: Identification of model (11.18) – MILP vs. MIQP (results are averaged on ten estimation data sets generated by random Gaussian inputs $u(t)$ and zero output noise). The horizontal axes show the number of estimation data samples. The results from MILP are marked with diamonds, and the results from MIQP are marked with squares.

discrete variables $\sigma_i \in \{0, 1\}$, and rewrite the predicted output as

$$\hat{y}(t, \theta, \sigma) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M}(2\sigma_i - 1)\max\{\varphi^T(t)\theta_i, 0\} \tag{11.21}$$

Using the same definition of $z_i(t)$ as before (see (11.4)) yields

$$\hat{y}(t, \theta, \sigma) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M}(2\sigma_i - 1)z_i(t) \tag{11.22}$$

This expression is still not linear in the unknowns, which means that the criterion functions will not be so either. Therefore we define

$$\zeta_i(t) = \sigma_i z_i(t), \quad i = 1, \dots, M \tag{11.23}$$

and get

$$\hat{y}(t, \theta, \sigma) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M}(2\zeta_i(t) - z_i(t)) \tag{11.24}$$

The criterion function V_2 becomes

$$V_2 = \sum_{t=1}^{N}\left(y(t) - \left(\varphi^T(t)\theta_0 + \sum_{i=1}^{M}(2\zeta_i(t) - z_i(t))\right)\right)^2 \tag{11.25}$$

which is quadratic in θ_0, $\zeta_i(t)$ and $z_i(t)$.

What remains now is to formulate the definitions of σ_i, $z_i(t)$, and $\zeta_i(t)$ as linear inequalities. For this we need the auxiliary discrete variables $\delta_i(t)$, defined as in (11.8). Following a similar line of reasoning as in Section 11.1, one arrives at the inequalities

$$\begin{cases} \zeta_i(t) \geq 0 \\ \zeta_i(t) \leq z_i(t) \\ z_i(t) \leq U_i(t)\delta_i(t) \\ \zeta_i(t) \leq U_i(t)\sigma_i \\ (1-\delta_i(t))L_i(t) + z_i(t) \leq \varphi^T(t)\theta_i \\ \varphi^T(t)\theta_i \leq z_i(t) \\ z_i(t) - \zeta_i(t) \leq U_i(t)(1-\sigma_i) \end{cases} \tag{11.26}$$

which can easily be checked by testing all possible values of $(\delta_i(t), \sigma_i)$.

Since M^+ is unknown, we cannot include the inequalities (11.13) right away. However, we can restrict the search space to some extent by including

$$\varphi^T(1)\theta_i \leq 0, \quad i = 1, \ldots, M \tag{11.27}$$

Furthermore, we can let

$$\sigma_{i+1} \leq \sigma_i, \quad i = 1, \ldots, M-1 \tag{11.28}$$

to order the max functions, so that the positive max functions come first.

An alternative way of dealing with the problem of unknown M^+ is to write the predicted output as

$$\hat{y}(t|\theta) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M}(\max\{\varphi^T(t)\theta_i^+, 0\} - \max\{\varphi^T(t)\theta_i^-, 0\}) \tag{11.29}$$

where we require that

$$\varphi^T(t)\theta_i^+ \geq 0 \quad \Leftrightarrow \quad \varphi^T(t)\theta_i^- \geq 0 \tag{11.30}$$

This can be regulated with the help of the discrete variables $\delta_i(t)$, where

$$\delta_i(t) = 1 \quad \Leftrightarrow \quad \varphi^T(t)\theta_i^+ \geq 0 \quad \Leftrightarrow \quad \varphi^T(t)\theta_i^- \geq 0 \tag{11.31}$$

Furthermore, we introduce the auxiliary variables

$$z_i(t) = \max\{\varphi^T(t)\theta_i^+, 0\} - \max\{\varphi^T(t)\theta_i^-, 0\} = \delta_i(t)\varphi^T(t)(\theta_i^+ - \theta_i^-) \tag{11.32}$$

We also need the lower and upper bounds

$$\begin{aligned} L_i^+(t) &\leq \varphi^T(t)\theta_i^+ \leq U_i^+(t) \\ L_i^-(t) &\leq \varphi^T(t)\theta_i^- \leq U_i^-(t) \\ L_i(t) &\leq \varphi^T(t)(\theta_i^+ - \theta_i^-) \leq U_i(t) \end{aligned}$$

The problem can now be written as

$$\begin{aligned}
\min_{\theta^{\pm},z,\delta} \quad & V_2 = \sum_{t=1}^{N} \left(y(t) - \left(\varphi^T(t)\theta_0 + \sum_{i=1}^{M} z_i(t) \right) \right)^2 \\
\text{subj. to} \quad & \varphi^T(t)\theta_i^+ \geq L_i^+(t)(1-\delta_i(t)) \\
& \varphi^T(t)\theta_i^+ \leq U_i^+(t)\delta_i(t) \\
& \varphi^T(t)\theta_i^- \geq L_i^-(t)(1-\delta_i(t)) \\
& \varphi^T(t)\theta_i^- \leq U_i^-(t)\delta_i(t) \\
& z_i(t) \geq L_i(t)\delta_i(t) \\
& z_i(t) \leq U_i(t)\delta_i(t) \\
& \varphi^T(t)(\theta_i^+ - \theta_i^-) \geq z_i(t) + L_i(t)(1-\delta_i(t)) \\
& \varphi^T(t)(\theta_i^+ - \theta_i^-) \leq z_i(t) + U_i(t)(1-\delta_i(t)) \\
& \delta_i(t) \in \{0,1\}
\end{aligned} \tag{11.33}$$

Here the method from [12] is used to transform the definitions of $z_i(t)$ and $\delta_i(t)$ into inequalities. The first four inequalities establish the relation between θ_i^+, θ_i^-, and $\delta_i(t)$, while the last four inequalities define $z_i(t)$.

The advantage of this method compared to the first alternative is that we need fewer variables. A drawback is that θ_i^+ and θ_i^- cannot be uniquely determined; if $({\theta_i^+}^*, {\theta_i^-}^*)$ is a solution, so is, e.g., $({\theta_i^+}^* + \lambda{\theta_i^+}^*, {\theta_i^-}^* + \lambda{\theta_i^+}^*)$ for $\lambda \geq 0$, as long as the upper and lower bounds are not reached. Another drawback is that the two hinging hyperplanes given by θ_i^+ and θ_i^- will not always correspond to one single hinging hyperplane giving the same partition of the estimation dataset.

To restrict the search space, we can include the following inequalities that correspond to (11.13):

$$\begin{cases} \varphi^T(1)\theta_i^+ \leq 0 & i = 1,\dots,M \\ \varphi^T(1)(\theta_1^+ - \theta_1^-) \leq \cdots \leq \varphi^T(1)(\theta_M^+ - \theta_M^-) \end{cases}$$

which makes the variables $\delta_i(1)$, $z_i(1)$ redundant.

11.2.2 Discontinuous Hinging Hyperplane Models

In hinging hyperplane models, the output $y(t)$ is a continuous function of the regressor $\varphi(t)$. On the other hand, hybrid systems often consist of piecewise affine discontinuous mappings. In order to tackle discontinuities, we can modify the hinging hyperplane model (11.2) in the form

$$\hat{y}(t|\theta) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M} (\varphi^T(t)\theta_i + a_i)\delta_i(t) \tag{11.34a}$$

$$\delta_i(t) = \begin{cases} 0 & \varphi^T(t)\theta_i < 0 \\ 1 & \varphi^T(t)\theta_i \geq 0 \end{cases} \tag{11.34b}$$

where a_i, $i = 1, \ldots, M$ are additional free parameters, $a_i^- \leq a_i \leq a_i^+$. This modification allows a discontinuity of constant size along each hinge. A more general class is obtained by using

$$\hat{y}(t|\theta) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \varphi^T(t)\theta_i\delta_i(t) \tag{11.35a}$$

$$\delta_i(t) = \begin{cases} 0 & C_i\varphi(t) < 0 \\ 1 & C_i\varphi(t) \geq 0 \end{cases} \tag{11.35b}$$

where C_i, $i = 1, \ldots, M$ are additional free (row) vectors of parameters, controlling the partitioning of the state-space independently of θ_i. By introducing new continuous variables $z_i(t)$

$$z_i(t) = (\varphi^T(t)\theta_i + a_i)\delta_i(t)$$

or

$$z_i(t) = \varphi^T(t)\theta_i\delta_i(t)$$

respectively, both the problems of identifying (11.34) and (11.35) can again be recast as an MILP/MIQP. Using the transformations from [12], from (11.35) we get

$$\begin{aligned}
\min_{\theta,z,\delta,C} \quad & V_2 = \sum_{t=1}^{N} \left(y(t) - \left(\varphi^T(t)\theta_0 + \sum_{i=1}^{M} z_i(t) \right) \right)^2 \\
\text{subj. to} \quad & C_i\varphi(t) \geq L_{C_i}(t)(1 - \delta_i(t)) \\
& C_i\varphi(t) \leq U_{C_i}(t)\delta_i(t) - (1 - \delta_i(t))\mu \\
& z_i(t) \geq L_i(t)\delta_i(t) \\
& z_i(t) \leq U_i(t)\delta_i(t) \\
& \varphi^T(t)\theta_i \geq z_i(t) + L_i(t)(1 - \delta_i(t)) \\
& \varphi^T(t)\theta_i \leq z_i(t) + U_i(t)(1 - \delta_i(t)) \\
& \delta_i(t) \in \{0, 1\}
\end{aligned} \tag{11.36}$$

where μ is a small number (just as in (11.10)), which we in practice can choose, e.g., to the machine precision†; and $L_{C_i}(t)$ and $U_{C_i}(t)$ are lower and upper bounds on $C_i\varphi(t)$.

Note that the problem (11.36) mostly does not have a unique solution. For instance, once θ_i, $z_i(t)$, and $\delta_i(t)$ have been fixed, in general there exist infinitely many vectors C_i satisfying the constraints (11.35b) (cf. the general PWARX systems in Section 11.2.4).

†Here μ cannot be set to zero, since $z_i(t)$ is discontinuous where $\delta_i(t)$ switches between 0 and 1 (compare with the discussion just after (11.10)).

11.2.3 Robust Hinging Hyperplane Models

As will be discussed in Chapter 12, in formal verification methods, model uncertainty needs to be handled in order to provide safety guarantees. Typically, the model is associated with a bounded uncertainty. In the context of symbolic solvers for timed automata, differential inclusions $a \leq \dot{x} \leq b$ are handled, for instance by the solver HyTech [63]. As another example, for piecewise affine and MLD systems (see Section 9.2), the verification algorithm proposed in [14] handles model uncertainty as additive input disturbances entering a nominal model.

In the present context of hinging hyperplane models, we wish to find an uncertainty description of the form

$$\varphi^T(t)\theta_0^- + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i^-, 0\} \leq y(t) \leq \varphi^T(t)\theta_0^+ + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i^+, 0\} \tag{11.37}$$

(that should hold for all t) for an inclusion-type of description, or the form

$$y(t) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i, 0\} + e(t), \quad e^- \leq e(t) \leq e^+ \tag{11.38}$$

for an additive-disturbance-type of description. With a finite number of data samples, it is naturally impossible to guarantee the inclusions of (11.37) or (11.38) to hold for any kind of input and initial conditions, so what will be proposed here are methods for making the inclusions hold at least for the experimental data.

If we consider the inclusion-type of uncertainty description (11.37), a pair of extreme models with parameters θ^-, θ^+ can be obtained by solving (11.11) or (11.12) with the additional linear constraints

$$y(t) \geq \varphi^T(t)\theta_0^- + \sum_{i=1}^{M} \pm z_i^-(t), \quad t = 1, \ldots, N \tag{11.39}$$

for estimating θ^-, and

$$y(t) \leq \varphi^T(t)\theta_0^+ + \sum_{i=1}^{M} \pm z_i^+(t), \quad t = 1, \ldots, N \tag{11.40}$$

for estimating θ^+, respectively. Turning to the additive-disturbance description of the form (11.38), it can computed in two alternative ways:

1. First, identify the model parameters $\hat{\theta}$ by solving (11.11) or (11.12), and then compute

$$\begin{aligned} e^+ &\triangleq \max_{t=1,\ldots,N} y(t) - \hat{y}(t, \hat{\theta}) \\ e^- &\triangleq \min_{t=1,\ldots,N} y(t) - \hat{y}(t, \hat{\theta}) \end{aligned} \tag{11.41}$$

2. Modify the MILP (11.12) by using only one slack variable s, i.e., replace all entries of s_t in (11.12) by s. The objective function that should be minimized simply becomes s. The corresponding optimum then provides a nominal model such that the bound on the norm of the additive disturbance $e(t)$ is minimized.

11.2.4 General PWARX Systems

In Section 9.2.1, PWARX systems were defined as being of the following type:

$$y(t) = \varphi^T(t)\theta(v) + e(t) \tag{11.42}$$

where the key vector v is a function of $\varphi(t)$, $\varphi(t)$ consists of old inputs and outputs plus a constant element, and the regions of the different subsystems are polyhedral. We assume that the regions are separated by M hyperplanes $\{\varphi \in \mathbb{R}^n \mid C_i\varphi = 0\}$, $i = 1, \ldots, M$, where $C_i \in \mathbb{R}^{1\times n}$ are unknown normal vectors of the hyperplanes, and n is the dimension of φ. In order to better fit into the framework of this chapter, v will be defined slightly different compared to Section 9.1.1:

$$v_i = \begin{cases} 0 & \text{if } C_i\varphi < 0 \\ 1 & \text{if } C_i\varphi \geq 0 \end{cases}$$

We can notice directly that in most cases, it will be impossible to determine C_i uniquely from the experimental data, since there is a continuum of hyperplanes which will split the set of vectors $\varphi(t)$ in the same partitions.

Let us now consider the special case of $M = 2$ for the remainder of the section. The reformulation for other values of M is completely analogous. First we can write $\hat{y}(t|\theta)$ as

$$\begin{aligned} \hat{y}(t|\theta) &= \varphi^T(t)\theta(v) \\ &= \varphi^T(t)\left[\theta(\begin{pmatrix}0\\0\end{pmatrix})(1 - v_1(t))(1 - v_2(t)) + \theta(\begin{pmatrix}1\\0\end{pmatrix})v_1(t)(1 - v_2(t))\right. \\ &\qquad \left. +\theta(\begin{pmatrix}0\\1\end{pmatrix})(1 - v_1(t))v_2(t) + \theta(\begin{pmatrix}1\\1\end{pmatrix})v_1(t)v_2(t)\right] \end{aligned}$$

Rearranging the terms gives

$$\begin{aligned} \hat{y}(t|\theta) &= \varphi^T(t)\left[\theta(\begin{pmatrix}0\\0\end{pmatrix}) + v_1(t)\left(-\theta(\begin{pmatrix}0\\0\end{pmatrix}) + \theta(\begin{pmatrix}1\\0\end{pmatrix})\right)\right. \\ &\qquad + v_2(t)\left(-\theta(\begin{pmatrix}0\\0\end{pmatrix}) + \theta(\begin{pmatrix}0\\1\end{pmatrix})\right) \\ &\qquad \left. +v_1(t)v_2(t)\left(\theta(\begin{pmatrix}0\\0\end{pmatrix}) - \theta(\begin{pmatrix}1\\0\end{pmatrix}) - \theta(\begin{pmatrix}0\\1\end{pmatrix}) + \theta(\begin{pmatrix}1\\1\end{pmatrix})\right)\right] \\ &\triangleq \varphi^T(t)\left[\tilde{\theta}_0 + v_1(t)\tilde{\theta}_1 + v_2(t)\tilde{\theta}_2 + v_1(t)v_2(t)\tilde{\theta}_3\right] \end{aligned}$$

where the new variables $\tilde{\theta}_0, \dots, \tilde{\theta}_3$ have been introduced as different linear combinations of $\theta(v)$ for the different values of v. Now we can identify $\tilde{\theta}_0, \dots, \tilde{\theta}_3$, and then solve for $\theta(v)$, but first the expression for $\hat{y}$ must be linear in the unknown variables. To accomplish that, introduce

$$\begin{aligned}
z_1(t) &= v_1(t)\varphi^T(t)\tilde{\theta}_1 \\
z_2(t) &= v_2(t)\varphi^T(t)\tilde{\theta}_2 \\
z_3(t) &= v_2(t)\varphi^T(t)\tilde{\theta}_3 \\
z_4(t) &= v_1(t)z_3(t)
\end{aligned}$$

to get

$$\hat{y}(t|\theta) = \varphi^T(t)\tilde{\theta}_0 + z_1(t) + z_2(t) + z_4(t)$$

and the quadratic criterion function

$$V_2 = \sum_{t=1}^{N} \Big(y(t) - \big(\varphi^T(t)\tilde{\theta}_0 + z_1(t) + z_2(t) + z_4(t) \big) \Big)^2$$

What is left now is to rewrite the definitions of $z_j(t)$ as linear inequalities. Since the values of $v_j(t)$ do not depend on $\tilde{\theta}_k$, but on C_i, we cannot use the same procedure as in Section 11.1.2. Instead, the method in [12] is used again, which yields

$$\begin{cases} C_i\varphi(t) \geq L_{C_i}(t)(1 - v_i(t)) \\ C_i\varphi(t) \leq U_{C_i}(t)v_i(t) - (1 - v_i(t))\mu \end{cases} \qquad i = 1, 2$$

$$\begin{cases} z_j(t) \leq U_{\tilde{\theta}_j}(t)v_j(t) \\ z_j(t) \geq L_{\tilde{\theta}_j}(t)v_j(t) \\ \varphi^T(t)\tilde{\theta}_j \geq z_j(t) + L_{\tilde{\theta}_j}(t)(1 - v_j(t)) \\ \varphi^T(t)\tilde{\theta}_j \leq z_j(t) + U_{\tilde{\theta}_j}(t)(1 - v_j(t)) \end{cases} \qquad j = 1, 2$$

$$\begin{cases} z_3(t) \leq U_{\tilde{\theta}_3}(t)v_2(t) \\ z_3(t) \geq L_{\tilde{\theta}_3}(t)v_2(t) \\ \varphi^T(t)\tilde{\theta}_3 \geq z_3(t) + L_{\tilde{\theta}_3}(t)(1 - v_2(t)) \\ \varphi^T(t)\tilde{\theta}_3 \leq z_3(t) + U_{\tilde{\theta}_3}(t)(1 - v_2(t)) \end{cases}$$

$$\begin{cases} z_4(t) \leq U_{\tilde{\theta}_3}(t)v_1(t) \\ z_4(t) \geq L_{\tilde{\theta}_3}(t)v_1(t) \\ z_3(t) \geq z_4(t) + L_{\tilde{\theta}_3}(t)(1 - v_1(t)) \\ z_3(t) \leq z_4(t) + U_{\tilde{\theta}_3}(t)(1 - v_1(t)) \end{cases}$$

Here, μ is a small number, e.g., the machine precision. $U_{C_i}(t)$, $L_{C_i}(t)$, $U_{\tilde{\theta}_i}(t)$, and $L_{\tilde{\theta}_i}(t)$ are upper and lower bounds on $C_i\varphi(t)$ and $\varphi^T(t)\tilde{\theta}_i$, respectively, which can be derived from bounds on C_i and $\theta(v)$ given a priori.

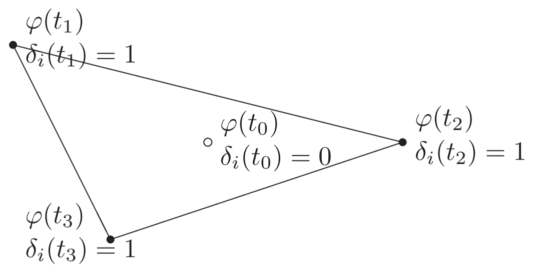


Figure 11.10: An infeasible assignment of $\delta_i(t)$ values. No hyperplane can separate $\varphi(t_0)$ from $\varphi(t_1)$, $\varphi(t_2)$, and $\varphi(t_3)$, so if $\delta_i(t_1) = \delta_i(t_2) = \delta_i(t_3) = 1$, then we must also have $\delta_i(t_0) = 1$.

To restrict the search space a bit more, we can also, e.g., require that

$$C_i\varphi(1) = -1, \quad i = 1, 2$$

which also implies that $v_1(1) = v_2(1) = 0$ and $z_j(1) = 0$, $j = 1, \ldots, 4$.

11.3 Computational Complexity and Theoretical Aspects

As mentioned in Section A.4, the general MILP or MIQP problem is $\mathcal{NP}$-hard [49, 151], so that, in the worst case, every combination of δ values has to be tested. However, in practice not all combinations of δ values are feasible, which improves the complexity a bit. This is the subject of the following section. We will only consider the systems of Section 11.1, although some of the discussion is relevant also to the extensions in Section 11.2. No rigorous complexity analysis will be made; instead some aspects of it will be discussed.

11.3.1 Number of Feasible δ Combinations

Since the identification problem is easily solved once we know the correct partitioning of the data (it reduces to an LP or a convex QP problem), we can look at the optimization process as trying to find the best way of partitioning the set of vectors $\varphi(t)$, i.e., to partition the regression vector space into different regions. As we have also seen, $\delta_i(t)$ describes on what side of the ith hinge that $\varphi(t)$ is positioned. In other words, for each t, $\delta_1(t), \ldots, \delta_M(t)$ together work as a key vector, telling us in what region $\varphi(t)$ is situated. Each leaf in the MILP/MIQP search trees (see Section A.4 in the Appendix) corresponds to a certain combination of δ values that is to be tested. However, in all cases of interest, many combinations of values of the $\delta_i(t)$ variables will be infeasible, i.e., they will not correspond to a possible partitioning of the state-space. For example, if a regression vector $\varphi(t_0)$ belongs to the convex hull of some other regression vectors $\varphi(t_1), \ldots, \varphi(t_k)$ (see Figure 11.10), and we assign the values $\delta_i(t_1) = \delta_i(t_2) = \cdots = \delta_i(t_k) = 1$, then $\delta_i(t_0) = 1$ by necessity, since no hyperplane $\varphi^T\theta_i = 0$ can separate $\varphi(t_0)$ from $\varphi(t_1), \dots, \varphi(t_k)$. The natural question arises: How many feasible combinations of $\delta_i(t)$ values are there?

If the dimension of $\varphi(t)$ equals $n+1$, the question can also be formulated as: In how many ways can we group N points in $\mathbb{R}^n$ (that is, the points composed by elements number 2 to $n+1$ of $\varphi(t)$, $t = 1, \dots, N$) by separating them with M hyperplanes? This question is answered in Corollary A.2. Note that $\varphi(t)$ corresponds exactly to the vectors used in the proof of Corollary A.1, since the first element of $\varphi(t)$ always equals 1.

To be able to use Corollary A.2, we need the data to be in general position, but since – according to Section A.5 – for a randomly chosen set of points in $\mathbb{R}^n$, this condition should be satisfied with probability one, the restriction should not be too serious, especially if we assume white noise in our experimental data.

We assume that M and M^+ are known, and that the data set is persistently exciting. This implies that trivial partitions like having one hinge going outside the data set, or several hinges positioned at the same place, are – albeit feasible – unnecessary to try.

We also have to remember that some of the hinge functions should be positive and some negative. This will increase the number of δ combinations that have to be tested, since at each possible position for a hinge, both a positive and a negative hinge function has to be tested.

We summarize the previous discussion in the following corollary:

Corollary 11.1
Assume that N samples of data in general position are given, and that a hinging hyperplane model with M^+ positive and $M^- = M - M^+$ negative hinge functions should be fitted to it, using (11.11) *or* (11.12). *Then the number of combinations of values of $\delta_i(t)$ that have to be tested equals*

$$f_\delta(n, N, M, M^+) = \binom{f(n+1, N)/2 - 1}{M} \cdot \binom{M}{M^+} \tag{11.43}$$

where

$$f(n, N) = 2\sum_{k=0}^{n-1} \binom{N-1}{k}$$

Proof According to Corollary A.2, there are $\binom{f(n+1,N)/2-1}{M}$ nontrivial ways in which a set of N points in $\mathbb{R}^n$ can be partitioned by M hyperplanes. For each partition, we should decide which of the M hyperplanes that should be hinges of the positive hinge functions. In other words, out of M hinge functions, we should choose M^+ to be positive. □

A MILP/MIQP search tree with only the feasible, nontrivial solutions as leaves, and the other cut away, would therefore have $f_\delta(n, N, M, M^+)$ leaves, and consequently contain $2f_\delta(n, N, M, M^+) - 1$ nodes altogether. Table 11.2 lists some values

Table 11.2: Number of feasible δ combinations for some different values of n, M, and N. Here we assume $M^+ = M$.

N	(n, M)						
	$(1,1)$	$(1,2)$	$(1,3)$	$(2,1)$	$(2,2)$	$(2,3)$	$(3,1)$
20	19	171	969	190	17955	1125180	1159
50	49	1176	18424	1225	749700	$3.1 \cdot 10^8$	19649
100	99	4851	156849	4950	$1.2 \cdot 10^7$	$2.0 \cdot 10^{10}$	161799
200	199	19701	1293699	19900	$2.0 \cdot 10^8$	$1.3 \cdot 10^{12}$	1313599
500	499	124251	$2.1 \cdot 10^7$	124750	$7.8 \cdot 10^9$	$3.2 \cdot 10^{14}$	$2.1 \cdot 10^7$

of $f_\delta(n, N, M, M^+)$. However, one should keep in mind that the computation of $f_\delta(n, N, M, M^+)$ just considers the feasibility of different δ combinations. Since the standard MILP/MIQP algorithms (such as branch-and-bound) can throw away obviously bad solutions, much fewer nodes will probably be tested in practice. The following examples illustrate this. It should be noted that in the MILP cases in these examples – like all MILP examples in this chapter – the Cplex MILP solver [70] was used. This is a general-purpose MILP solver, and does not (to the author's knowledge) explicitly make use of the feasibility properties special to this problem.

Example 11.5 *For the system in Example 11.3, the number of feasible combinations of δ values is $f_\delta(2, 12, 2, 1) = 4290$, so an MILP/MIQP search tree could contain $2 \cdot 4290 - 1 = 8579$ nodes. However, as previously mentioned only a few hundred nodes were actually visited. The total number of δ combinations (feasible and infeasible) is $2^{MN} = 2^{24} = 1.7 \cdot 10^7$.*

Example 11.6 *A noiseless hinging hyperplane system described by*

$$\begin{aligned} y(t) = 0.6312 &+ 0.0518y(t-1) - 0.2881u(t-1) \\ &+ \max\{-0.6019 + 0.8529y(t-1) + 0.3759u(t-1), 0\} \\ &- \max\{0.1238 - 0.5255y(t-1) + 1.8081u(t-1), 0\} \end{aligned} \tag{11.44}$$

was identified from 300 data samples, using a 1-norm criterion function and the Cplex MILP solver. Although the number of feasible δ combinations equals $f_\delta(2, 300, 2, 1) = 2.0 \cdot 10^9$, the problem was solved after having visited about 1000 nodes, taking about 5 minutes on a SunBlade100 machine. The total number of δ combinations is $2^{600} = 4.1 \cdot 10^{180}$.

Although the results of the previous examples are encouraging, the systems are rather simple, and the algorithms might not be feasible to use for larger examples. Another problem is that the algorithms seem to slow down when trying to identify systems with noise. However, some systems have a structure which can be exploited

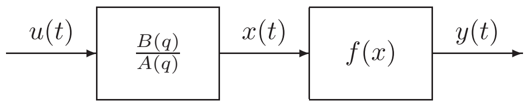


Figure 11.11: Wiener process with piecewise affine static output mapping.

to restrict the search space further, and to speed up the algorithm. In Section 11.4, piecewise affine Wiener models are considered.

Another option is to interrupt the algorithm before it is completed, and use the best solution found so far, or use it as an initial value for, e.g., a Gauss-Newton algorithm. This is described in Section 11.5. One could also use a change detection algorithm to detect timepoints when it seems more probable that a switching between different modes has occurred, and in this way reduce the number of combinations needed to consider. This will be considered in Section 11.6.

11.4 Piecewise Affine Wiener Models

As already mentioned, using MILP/MIQP directly on the problem formulations in Section 11.1 might be too complex to be used in realistic examples. However, there are some model classes where the special model structure allows us to simplify the computations considerably. Let us now turn to the class of *piecewise affine Wiener models*. These models consist of a linear system, followed by a static nonlinearity. The model structure is shown in Figure 11.11, and can be described by the relations

$$\begin{aligned} A(q)x(t) &= B(q)u(t) \\ y(t) &= f(x(t)) \end{aligned} \tag{11.45}$$

where $A(q) = 1 + \sum_{h=1}^{n_a} a_h q^{-h}$, $B(q) = \sum_{k=1}^{n_b} b_k q^{-k}$, and q^{-1} is the delay operator, $q^{-1}x(t) = x(t-1)$. For simplicity of notation, we assume that $n_a \geq n_b$ (this has no effect on the results). We assume that $f(x)$ is a piecewise affine, continuous, invertible function (without restrictions we can assume that f is strictly increasing), and parameterize its inverse as

$$x(t) = y(t) - \alpha_0 + \sum_{i=1}^{M} \pm \max\{\beta_i y(t) - \alpha_i, 0\} \tag{11.46}$$

The $\pm$ signs before the max functions are allowed in order to be able to represent nonconvex functions, and we use the shorthand notation introduced for hinging hyperplane models in (9.10); i.e., (11.46) is equivalent to

$$x(t) = y(t) - \alpha_0 + \sum_{i=1}^{M^+} \max\{\beta_i y(t) - \alpha_i, 0\} - \sum_{i=M^+ +1}^{M} \max\{\beta_i y(t) - \alpha_i, 0\}$$

where M^+ and M are assumed to be known. To avoid overparameterization, we assume that the coefficient of the linear part in (11.46) equals 1 (this can be compensated for in the $B(q)$ polynomial). As $\max\{-p, 0\} = -p + \max\{p, 0\}$ for all $p \in \mathbb{R}$, without loss of generality we can also assume $\beta_i \geq 0$.

Note that, in general, the piecewise affine Wiener models do not belong to the class of hinging hyperplane models, and therefore we cannot directly use the same kind of problem reformulations as in Section 11.1. To see this, we can rewrite the system using

$$x(t) = (1 - A(q))x(t) + B(q)u(t) \tag{11.47}$$

to get

$$\begin{aligned} y(t) &= f\left([1 - A(q)]\, x(t) + B(q)u(t)\right) \\ &= f\left([1 - A(q)]\, f^{-1}(y(t)) + B(q)u(t)\right) \end{aligned}$$

From Lemma A.4 (see Section A.6 in the appendix), we get an expression for f given (11.46). We can then see that the expression for $y(t)$ contains nested max functions. In fact, the model class is not a subset of the class of hinging hyperplane models, but of Güzeliş' model class (see (9.7)).

Like before, we would like to minimize a criterion function as in (11.3), with $\ell = \ell_2$ or $\ell = \ell_1$. We assume that there are known bounds for the values of the unknown parameters a_h, b_k, α_i, and β_i. We also introduce

$$\theta = (a_1 \ \ldots \ a_{n_a} \ b_1 \ \ldots \ b_{n_b} \ \alpha_0 \ \ldots \ \alpha_M \ \beta_1 \ \ldots \ \beta_M)^T$$

In the case of noisy systems, we assume that the noise enters the system according to the following equation

$$\begin{aligned} A(q)x(t) &= B(q)u(t) + e(t) \\ y(t) &= f(x(t)) \end{aligned} \tag{11.48}$$

where $E[e(t)] = 0$ and $e(t)$ is independent of $\{u_1^{t-1}, x_1^{t-1}, y_1^{t-1}\}$. However, the algorithm may work well also when the noise enters the system differently, as illustrated in Example 11.7.

The problem of identification of Wiener models has been subject to considerable research, and there are many papers published (see, e.g., [25, 60, 153, 157]).

11.4.1 Reformulating the Identification Problem

Like in the hinging hyperplane formulation, the first thing to do is to get rid of the max functions. This is done by introducing the discrete variables

$$\delta_i(t) = \begin{cases} 0 & \beta_i y(t) - \alpha_i < 0 \\ 1 & \beta_i y(t) - \alpha_i \geq 0 \end{cases} \quad i = 1, \ldots, M \tag{11.49}$$

An additional problem for this model structure is that we will get products between the coefficients a_h of the $A(q)$ polynomial and the coefficients inside the max functions, β_i and α_i. Furthermore, since a_h may very well be negative, the inequalities in the definition (11.49) of $\delta_i(t)$ may change directions if we multiply by a_h. This is not desirable, so to get rid of these problems, first define $a_h = a_h^+ - a_h^-$, where $a_h^+, a_h^- \geq \gamma$, and $\gamma > 0$ is any positive scalar. Here we let $\gamma = 1$. Then

$$\begin{aligned}
&a_h \max\{\beta_i y(t-h) - \alpha_i, 0\} \\
&\quad = (a_h^+ - a_h^-) \max\{\beta_i y(t-h) - \alpha_i, 0\} \\
&\quad = \max\{a_h^+ \beta_i y(t-h) - a_h^+ \alpha_i, 0\} - \max\{a_h^- \beta_i y(t-h) - a_h^- \alpha_i, 0\} \\
&\quad = \max\{c_{ih}^+ y(t-h) - d_{ih}^+, 0\} - \max\{c_{ih}^- y(t-h) - d_{ih}^-, 0\}
\end{aligned}$$

where

$$\begin{aligned}
c_{ih}^{\pm} &\triangleq a_h^{\pm} \beta_i, \quad i = 1, \ldots, M,\ h = 1, \ldots, n_a \\
d_{ih}^{\pm} &\triangleq a_h^{\pm} \alpha_i, \quad i = 1, \ldots, M,\ h = 1, \ldots, n_a
\end{aligned}$$

Let also

$$\begin{aligned}
c_{i0} &= c_{i0}^+ = c_{i0}^- \triangleq \beta_i, \quad i = 1, \ldots, M \\
d_{i0} &= d_{i0}^+ = d_{i0}^- \triangleq \alpha_i, \quad i = 1, \ldots, M \\
d_{0h} &\triangleq a_h \alpha_0, \quad h = 1, \ldots, n_a \\
d_{00} &\triangleq \alpha_0,
\end{aligned}$$

and

$$\bar{d}_0 \triangleq \sum_{h=0}^{n_a} d_{0h} = \left(1 + \sum_{h=1}^{n_a} a_h\right) \alpha_0$$

As $a_h^+, a_h^- > 0$, from (11.49) it now follows

$$\delta_i(t) = \begin{cases} 0 & a_h^{\pm} \beta_i y(t) - a_h^{\pm} \alpha_i < 0 \\ 1 & a_h^{\pm} \beta_i y(t) - a_h^{\pm} \alpha_i \geq 0 \end{cases} \quad i = 1, \ldots, M,\ h = 1, \ldots, n_a$$

and hence

$$\delta_i(t) = \begin{cases} 0 & c_{ih}^{\pm} y(t) - d_{ih}^{\pm} < 0 \\ 1 & c_{ih}^{\pm} y(t) - d_{ih}^{\pm} \geq 0 \end{cases} \quad i = 1, \ldots, M,\ h = 0, \ldots, n_a \tag{11.50}$$

Introduce the auxiliary continuous variables

$$\begin{aligned}
z_{ih}(t) &= \delta_i(t-h) \left((c_{ih}^+ - c_{ih}^-) y(t-h) - (d_{ih}^+ - d_{ih}^-) \right), \ h = 1, \ldots, n_a \\
z_{i0}(t) &= \delta_i(t)(c_{i0} y(t) - d_{i0})
\end{aligned} \tag{11.51}$$

for $i = 1, \ldots, M$, $t = n_a + 1, \ldots, N$.

To rewrite the criterion function (11.3), we first try to express $y(t)$ as a linear function of our unknowns. First we note that, by (11.46) and (11.51),

$$\begin{aligned} x(t) &= y(t) - d_{00} + \sum_{i=1}^{M} \pm z_{i0}(t) \\ a_1 x(t-1) &= a_1 y(t-1) - d_{01} + \sum_{i=1}^{M} \pm z_{i1}(t) \\ &\vdots \\ a_{n_a} x(t-n_a) &= a_{n_a} y(t-n_a) - d_{0n_a} + \sum_{i=1}^{M} \pm z_{in_a}(t) \end{aligned} \tag{11.52}$$

Now, using (11.48) and (11.52), we get

$$\begin{aligned} x(t) &= y(t) - d_{00} + \sum_{i=1}^{M} \pm z_{i0}(t) \\ &= -\sum_{h=1}^{n_a} \left(a_h y(t-h) - d_{0h} + \sum_{i=1}^{M} \pm z_{ih}(t) \right) + \sum_{k=1}^{n_b} b_k u(t-k) + e(t) \end{aligned}$$

which provides the relation

$$y(t) = -\sum_{h=1}^{n_a} a_h y(t-h) + \sum_{k=1}^{n_b} b_k u(t-k) + \bar{d}_0 - \sum_{i=1}^{M} \sum_{h=0}^{n_a} \pm z_{ih}(t) + e(t) \tag{11.53}$$

and therefore

$$\hat{y}(t|\theta) = -\sum_{h=1}^{n_a} a_h y(t-h) + \sum_{k=1}^{n_b} b_k u(t-k) + \bar{d}_0 - \sum_{i=1}^{M} \sum_{h=0}^{n_a} \pm z_{ih}(t) \tag{11.54}$$

This means that the criterion function (11.3) using the ℓ_2 norm can be written as

$$\begin{aligned} V_2(\theta, Z_1^N) = \sum_{t=n_a+1}^{N} \Bigg(& y(t) + \sum_{h=1}^{n_a} a_h y(t-h) - \sum_{k=1}^{n_b} b_k u(t-k) \\ & - \bar{d}_0 + \sum_{i=1}^{M} \sum_{h=0}^{n_a} \pm z_{ih}(t) \Bigg)^2 \end{aligned} \tag{11.55}$$

If 1-norm is used, we can rewrite the criterion function similarly and then introduce slack variables, analogously to what was done in (11.7).

The next step is to translate the definitions (11.51) of $z_{ih}(t)$ and (11.50) of $\delta_i(t)$ into linear inequalities. This is done using the method in [12]. The resulting inequalities are

$$
\begin{cases}
c_{ih}^{\pm} y(t) - d_{ih}^{\pm} \geq L_{ih}^{\pm}(t)(1-\delta_i(t)) \\
c_{ih}^{\pm} y(t) - d_{ih}^{\pm} \leq U_{ih}^{\pm}(t)\delta_i(t)
\end{cases}
\quad i = 1,\dots,M,\ h = 0,\dots,n_a,\ t = 1,\dots,N
$$

$$
\begin{cases}
z_{i0}(t) \geq L_{i0}(t)\delta_i(t) \\
z_{i0}(t) \leq U_{i0}(t)\delta_i(t) \\
c_{i0} y(t) - d_{i0} \geq z_{i0}(t) + L_{i0}(t)(1-\delta_i(t)) \\
c_{i0} y(t) - d_{i0} \leq z_{i0}(t) + U_{i0}(t)(1-\delta_i(t))
\end{cases}
\quad i = 1,\dots,M,\ t = n_a+1,\dots,N
$$

$$
\begin{cases}
z_{ih}(t) \geq L_{ih}(t)\delta_i(t-h) \\
z_{ih}(t) \leq U_{ih}(t)\delta_i(t-h) \\
(c_{ih}^{+} - c_{ih}^{-})y(t-h) - (d_{ih}^{+} - d_{ih}^{-}) \geq z_{ih}(t) + L_{ih}(t)(1-\delta_i(t-h)) \\
(c_{ih}^{+} - c_{ih}^{-})y(t-h) - (d_{ih}^{+} - d_{ih}^{-}) \leq z_{ih}(t) + U_{ih}(t)(1-\delta_i(t-h))
\end{cases}
\quad h = 1,\dots,n_a
\tag{11.56}
$$

where $U_{ih}^{\pm}(t)$, $L_{ih}^{\pm}(t)$, $U_{ih}(t)$, and $L_{ih}(t)$ are upper and lower bounds on $c_{ih}^{\pm}y(t) - d_{ih}^{\pm}$ and $(c_{ih}^{+} - c_{ih}^{-})y(t-h) - (d_{ih}^{+} - d_{ih}^{-})$, respectively, computed from the known upper and lower bounds on a_h, b_k, α_i, and β_i. Note also that if $c_{ih}^{\pm}y(t) - d_{ih}^{\pm} = 0$, $\delta_i(t)$ is allowed to be either 0 or 1. This is analogous to what was discussed in Section 11.1.2, and corresponds to a data sample lying on a breakpoint of the nonlinearity.

Beside the previously mentioned reformulations, that are very similar to the ones in the hinging hyperplane case, we can also introduce some extra constraints that will help the MILP/MIQP solver to discard infeasible solutions, thereby reducing the complexity. Without loss of generality, we can assume that the breakpoints for the positive max functions in the piecewise affine output nonlinearity are ordered, and similarly for the negative max functions. Therefore,

$$\delta_i(t) = 1 \quad \Rightarrow \quad \delta_j(t) = 1 \tag{11.57}$$

should hold for all $i, j \leq M^+$ such that $j < i$, and for all $i, j > M^+$ such that $j < i$. Each constraint (11.57) is translated into

$$\delta_i(t) - \delta_j(t) \leq 0, \quad j < i \tag{11.58}$$

where $i, j \leq M^+$ or $i, j > M^+$. Now, since

$$\delta_i(t) - \delta_j(t) \leq 0,\ \delta_j(t) - \delta_k(t) \leq 0 \quad \Rightarrow \quad \delta_i(t) - \delta_k(t) \leq 0$$

we only need to include the inequalities (11.58) for pairs of consecutive indices i, j.

Not only the breakpoints, but also the output data $y(t)$ can easily be ordered. This means that we can also get additional relations on $\delta_i(t)$ by using (11.49). In fact, if $\delta_i(t_0) = 1$ and $y(t_1) \geq y(t_0)$, it must follow that $\delta_i(t_1) = 1$. Like above, we can translate these relations into

$$\delta_i(t_0) - \delta_i(t_1) \leq 0, \quad \forall t_1 \neq t_0 : y(t_1) \geq y(t_0), \quad t_0, t_1 = 1,\dots,N \tag{11.59}$$

Table 11.3: Estimation results.

Parameter	True value	Noiseless data	$\lvert e(t)\rvert < 0.01$	$\lvert e(t)\rvert < 0.1$
a_1	-0.5	-0.5000	-0.4990	-0.5360
b_1	2	2.0000	2.0024	2.0003
α_0	-2	-2.0000	-2.0001	-1.7748
α_1	0.5	0.5000	0.5095	0.5509
α_2	-1.5	-1.5000	-1.4924	-1.4999
β_1	0.5	0.5000	0.5016	0.5028
β_2	0.5	0.5000	0.4988	0.4876

The final MIQP problem that we need to solve is

$$\min \sum_{t=n_a+1}^{N} \left(y(t) + \sum_{h=1}^{n_a} a_h y(t-h) - \sum_{k=1}^{n_b} b_k u(t-k) \right. \tag{11.60a}$$

$$\left. - \bar{d}_0 + \sum_{i=1}^{M} \sum_{h=0}^{n_a} \pm z_{ih}(t) \right)^2$$

$$\text{subj. to constraints (11.56), (11.58), (11.59)} \tag{11.60b}$$

with respect to the continuous variables $\{a_h, b_k, c_{i0}, d_{i0}, \bar{d}_0, c_{ih}^{\pm}, d_{ih}^{\pm}, z_{ih}(t)\}$ and the binary variables $\{\delta_i(t)\}$; $h = 1, \ldots, n_a$, $i = 0, \ldots, M$, $k = 1, \ldots, n_b$, $t = 1, \ldots, N$. The solution to (11.60) provides the optimal parameter estimates $\hat{a}_h$, $\hat{b}_k$, and

$$\hat{\alpha}_0 = \frac{\hat{\bar{d}}_0}{1 + \sum_{h=1}^{n_a} \hat{a}_h} \tag{11.61}$$

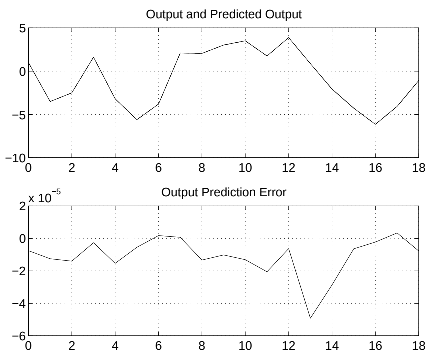
$$\hat{\alpha}_i = \hat{d}_{i0} \tag{11.62}$$

$$\hat{\beta}_i = \hat{c}_{i0} \tag{11.63}$$

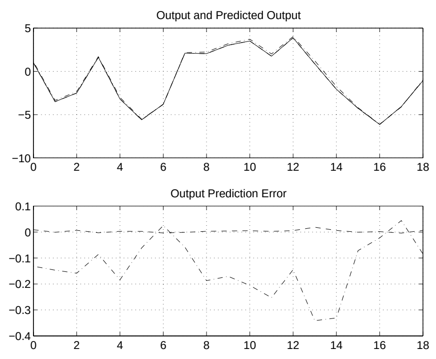
Finally, since the nonlinearity $f(x)$ is invertible, we can obtain the estimate $\hat{f}(x)$ by using the result of Lemma A.4.

Example 11.7 *The following example is taken from [13]. A Wiener model, consisting of a first-order linear system and a piecewise affine nonlinearity with two breakpoints, is identified using $N = 20$ estimation data points. The system is first identified using noiseless data, and then using noisy measurements $\tilde{y}(t) = y(t) + e(t)$, where $e(t)$ are independent random variables, uniformly distributed on a symmetric interval around zero. The resulting estimated parameters are shown in Table 11.3. Apparently, the estimated parameters are overall very close to the true values, the closer the lower the intensity of the output noise, as should be expected.*

The estimated model was also tested on a set of validation data, and in Figure 11.12 the resulting one-step-ahead predicted output and output error are plot-



(a) System estimated with noiseless data.



(b) System estimated with output noise $|e(t)| \leq 0.01$ (dashed line), and $|e(t)| \leq 0.1$ (dot-dashed line).

Figure 11.12: Validation results.

ted. Note that such a good performance cannot be achieved by using standard linear identification techniques.

11.4.2 Complexity Analysis

The complexity of the problem is tightly connected to the number of discrete variables, which equals MN. However, by imposing the constraints expressed by (11.58) and (11.59), the degrees of freedom for the integer variables are reduced considerably. If we would only have positive (or only negative) max functions, instead of having to test 2^{MN} different cases (the number of feasible and infeasible combinations of δ values) in the worst case, only $\binom{M+N}{M}$ combinations would have to be tested. Furthermore, of these combinations only $\binom{N-1}{M}$ are nontrivial (so that every affine region of the nonlinearity contains at least one data sample). If we have M^+ positive and $M^- = M - M^+$ negative max functions, this number must be multiplied by a factor $\binom{M}{M^+}$ to account for the fact that the positive and negative max functions could be ordered in different ways. For example, for $N = 20$ and $M = 2$ this means that the number of possible combinations of integer variables decreases from approximately 10^{12} to 231 (in the case of only positive max functions) or 462 (for one positive and one negative max function). Of these, 171 combinations (or 342, respectively) are nontrivial. In general, for a fixed M, the worst-case complexity grows as N^M. Note that this simplification is possible since the nonlinearity is one-dimensional, which allows an ordering of the breakpoints and of the output data. In fact, the number of nontrivial combinations equals $f_\delta(1, N, M, M^+)$, calculated in Corollary 11.1.

11.5 Using Suboptimal MILP/MIQP Solutions

One advantage with the MILP/MIQP algorithms is that they give intermediate results, which are suboptimal, but get better and better the longer the algorithm runs. This feature could be used for complex MILP/MIQP problems, where the time to find the optimal solution (and then prove that it really is optimal) might be extremely long. Instead of waiting for the optimal solution, one can abort the computations and use the best solution found so far.

In the case of piecewise affine system identification, one possibility is to use the suboptimal solutions obtained from the MILP/MIQP solver as initial values for a standard local minimization algorithm, e.g., Gauss-Newton (see Section 8.2). Finding good initial values for minimization algorithms is an important issue, and hopefully, using the intermediate solutions from MILP/MIQP as initial values gives better results than just using randomly picked initial values. Experiments show that this is often the case. However, it might sometimes take a little while until the MILP/MIQP solver finds suboptimal solutions that are worth using. The following examples illustrate this.

Example 11.8 *Consider once again the system in Example 11.3 and the 2-norm criterion function. The intermediate results from a simple, straightforward* Matlab *implementation of an MIQP solver are shown in Table 11.4. The parameters of these suboptimal solutions were then used as initial values of the* Matlab *optimization functions* `fminsearch` *and* `lsqnonlin`*, and the resulting criterion function values are also found in Table 11.4.*

As comparison, a number of random values were given as initial parameters to `fminsearch` *and* `lsqnonlin`*. In Figure 11.13, the values obtained by using the intermediate results from MIQP as initial values are compared to the mean result of using random initial values. We can see that already after about 250 visited nodes, the combined MIQP/*`fminsearch` *and MIQP/*`lsqnonlin` *algorithms perform better than using only* `fminsearch` *and* `lsqnonlin`*, respectively.*

Example 11.9 *Consider the system in Example 11.6, but with white noise added to $y(t)$ in (11.44). The variance of the noise is 0.1, while the input is white noise with unit variance. The system was identified using 100 such data samples and the 1-norm criterion function. The problem was reformulated according to (11.12) and given to the* Cplex *MILP solver. The intermediate solutions were used as initial values to the* Matlab *function* `fminsearch`*. Just as in Example 11.8, a number of random initial values to* `fminsearch` *were also used as comparison. Table 11.5 shows the resulting values of the criterion function. In Figure 11.14, the values of the criterion function are shown as a function of the approximative computation time for the MILP (on a SunBlade 100). Here, the combined MILP/*`fminsearch` *performs better than using random initial values from the very first intermediate result produced by the MILP solver.*

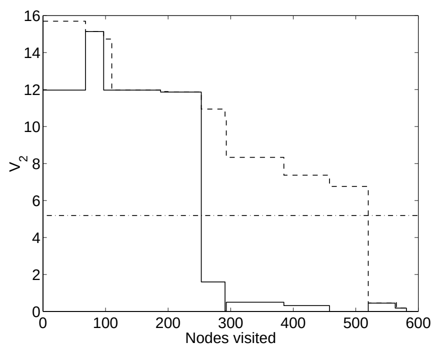
Node	MIQP	`fminsearch`	`lsqnonlin`
23	15.6967	11.9709	15.6967
68	15.1356	15.1356	15.1356
97	14.7320	11.9709	14.5412
100	14.7320	11.9709	14.5412
108	14.5007	11.9709	11.9132
110	11.9711	11.9709	11.8683
144	11.9711	11.9709	11.8683
188	11.8921	11.8629	11.8640
200	11.8632	11.8631	11.8632
253	10.9457	1.5977	0.3908
291	10.4377	0.0145	0.0564
293	8.3370	0.5000	0.0564
300	8.3370	0.5000	0.0564
385	7.3687	0.3188	0.0564
391	7.3687	0.3188	0.0564
458	6.7620	0.0000	0.0564
520	0.4540	0.4496	0.0000
523	0.4540	0.4496	0.0000
560	0.4514	0.4496	0.0000
563	0.4796	0.1817	0.0000
565	0.1834	0.1817	0.1669
581	0.1122	0.0000	0.0000
583	0.0044	0.0000	0.0000

Table 11.4: Intermediate results for the system in Examples 11.3 and 11.8. The first column shows the number of the node where the result was found (e.g., the first result was found in the 23rd node visited). The second column contains the values of the criterion function V_2 for the different solutions obtained by the MIQP solver. The third and fourth columns contain the values of V_2 after the solutions have been improved by running the Matlab routines `fminsearch` and `lsqnonlin`, respectively.

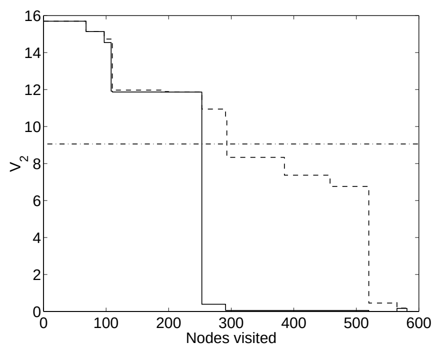
Example 11.10 *The noiseless system*

$$\begin{aligned} y(t) = 1.1515 - 0.5557y(t-1) - 0.3370y(t-2) + 1.3795u(t-1) \\ + \max\{-0.4898 + 0.0672y(t-1) + 1.9245y(t-2) - 0.3428u(t-1), 0\} \\ + \max\{-0.0336 + 0.4396y(t-1) + 0.1561y(t-2) - 1.3756u(t-1), 0\} \\ - \max\{-0.3871 - 1.6273y(t-1) + 0.3843y(t-2) + 1.6882u(t-1), 0\} \\ - \max\{1.0491 - 0.0629y(t-1) + 0.9997y(t-2) - 1.3911u(t-1), 0\} \end{aligned}$$

was identified using 100 data samples and the 1-norm criterion function. The Cplex *MILP solver was used to solve the MILP problem obtained, and the intermediate solutions were used as initial values to the* Matlab *function* `fminsearch`*.*



(a) Using `fminsearch`.



(b) Using `lsqnonlin`.

Figure 11.13: The intermediate results of Example 11.8 plotted against node number. The dashed lines show the V_2 values of the results from MIQP, the solid lines show the V_2 values after improving the solutions with `fminsearch` (a) and `lsqnonlin` (b), and the dash-dotted lines show the mean values of V_2 after using `fminsearch` (a) and `lsqnonlin` (b) on random initial values.

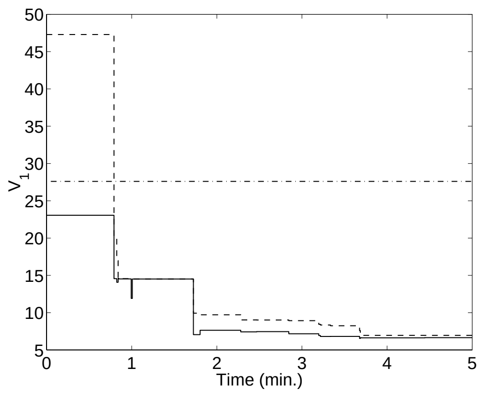


Figure 11.14: The criterion function values of the different solutions in Table 11.5 (Example 11.9) as a function of the computation time: The criterion function values after using only MILP (dashed line), after improving the results by using `fminsearch` (solid line), and the mean result from using `fminsearch` with random initial values (dash-dotted line).

	Node	Time	MILP	`fminsearch`
1	0	0.1	47.2966	23.0627
2	2310	0.8	19.9396	14.5632
3	2439	0.8	16.9958	14.0697
4	2500	0.8	14.7300	14.6152
5	2526	0.8	14.5641	14.5168
6	3100	1.0	14.5217	11.9073
7	3150	1.0	14.5137	14.5137
8	5940	1.7	9.9354	7.0514
9	6230	1.8	9.7185	7.6424
10	8010	2.3	9.0228	7.4318
11	8750	2.5	9.0206	7.4650
12	10220	2.8	8.9276	7.1749
13	11640	3.2	8.4372	6.9383
14	11717	3.2	8.3458	6.8154
15	12190	3.3	8.2480	6.8169
16	13590	3.7	7.6014	6.5391
17	13628	3.7	6.9740	6.6377
18	13700	3.7	6.9599	6.6272
19	16840	4.4	6.9572	6.6519

Table 11.5: Identification of the system in Example 11.9. Results of using intermediate solutions from an MILP algorithm as input to `fminsearch`: The number of visited nodes, the computation time (in minutes, using a SunBlade 100 machine), and the values of V_2 for the solutions before and after using `fminsearch`.

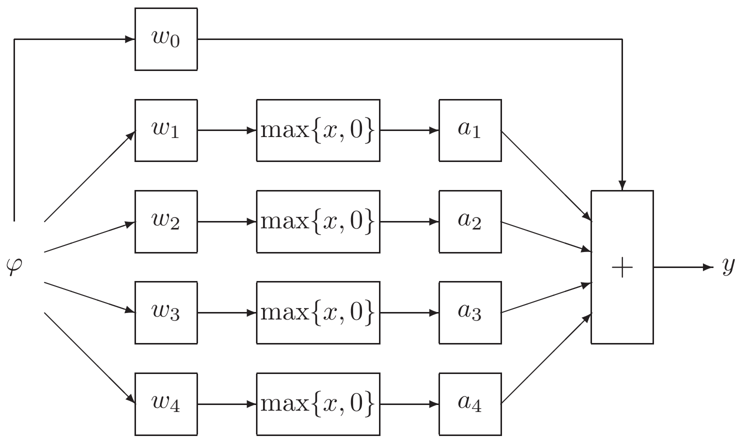


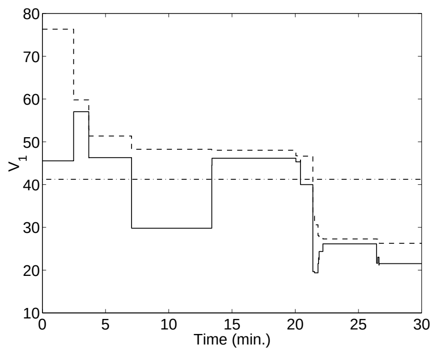
Figure 11.15: The structure of the neural network used in Example 11.10.

	Node	Time	MILP	`fminsearch`	NN
1	0	0.1	76.3178	45.5419	19.3366
2	2270	2.5	59.7858	57.0291	11.1362
3	3510	3.7	52.9559	46.2431	19.0352
4	3520	3.7	51.3513	46.2964	10.6999
5	7040	7.1	48.2320	29.7991	18.8824
6	13780	13.4	48.0376	44.5297	18.6923
7	13796	13.4	48.0274	46.1497	10.6867
8	21690	20.1	46.7244	45.3083	19.2758
9	22110	20.4	46.6540	45.7513	19.2948
10	22120	20.4	46.6491	39.9945	18.6721
11	23330	21.4	32.6644	19.6507	0.1239
12	23490	21.5	30.5823	19.3618	0.2066
13	23820	21.8	28.0669	21.5232	0.6236
14	23880	21.8	28.0649	22.8920	16.8954
15	23910	21.9	27.8096	22.4603	0.2265
16	24930	21.9	27.7681	24.3475	14.7952
17	24390	22.2	27.2844	26.1267	0.3424
18	29930	26.4	27.2509	21.6040	0.5424
19	30060	26.5	27.0350	23.0246	11.4028
20	30210	26.6	26.3121	21.1705	19.0251
21	30230	26.6	26.2591	21.5190	0.5986

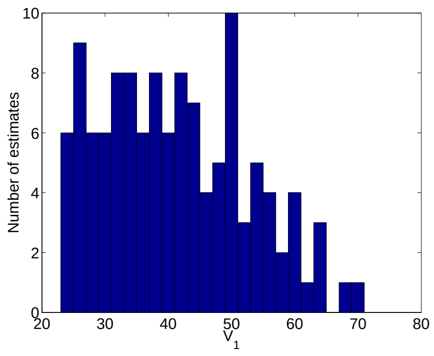
Table 11.6: Identification of the system in Example 11.10. Results of using intermediate solutions of an MILP algorithm as input to `fminsearch` and the neural network in Figure 11.15: The number of visited nodes, the computation time (in minutes, using a SunBlade 100 machine), the values of V_2 for the MILP solutions, and the values of V_2 after improving the MILP solutions using `fminsearch` and the neural network (NN), respectively.

They were also used as initial values to a neural network with the structure shown in Figure 11.15. The network was then trained using a backpropagation algorithm with variable learning rate (`traingdx`* in the Neural Network Toolbox in* Matlab*).*

Table 11.6 shows the resulting values of the criterion function. The computation time shown in the table (and in subsequent figures) is the time for finding the intermediate solution with the MILP solver (on a SunBlade 100 machine). Running `fminsearch` *once took about 15 seconds on average, while the time for training the neural network was approximately 3-25 minutes. The results from the combined MILP/*`fminsearch` *solver are visualized in Figure 11.16(a), where the values of the criterion function are shown as a function of the computation time, and are compared to the mean result using random initial values. After about 20 minutes, the combined solver works better than just using* `fminsearch` *with random initial values. An interesting thing to note here is that the* 11*th intermediate result – where the combined solver starts working better – is the first solution where both positive*



(a) The criterion function values after using only MILP (dashed line), after improving the results by using `fminsearch` (solid line), and the mean result from using `fminsearch` with random initial values (dash-dotted line).



(b) The distribution of results using `fminsearch` with random initial values for 30 minutes.

Figure 11.16: (a) The criterion function values of the different solutions in Table 11.6 (Example 11.10) as a function of the computation time (using `fminsearch`), and (b) the results from using `fminsearch` during 30 minutes.

and negative hinge functions are used. Figure 11.16(b) shows the distribution of results when `fminsearch` *was run repeatedly during 30 minutes with random initial values. As we can see, the combined MILP/*`fminsearch` *solver performed better (the best result when using* `fminsearch` *with random initial values was* 23.3*, while the best result from the combined solver was* 19.4*).*

The results from the neural network are shown in Figure 11.17, also here with the criterion function value as a function of the computation time. We can see that also in this case, using the intermediate results from the MILP solver as initial values for the network training mostly gives better results than the average result for using random initial values.

11.6 Using Change Detection to Reduce Complexity

Some PWA systems of interest may not switch so frequently between the different dynamics of the different submodels. For such systems, it should be possible to use a change detection algorithm to roughly find the timepoints when switches occur, and use this information to reduce the complexity of (11.12) or (11.11) by forcing several samples, lying in the same interval between two switches, to belong to the same subsystem. Here we propose to use an MILP algorithm over a sliding window

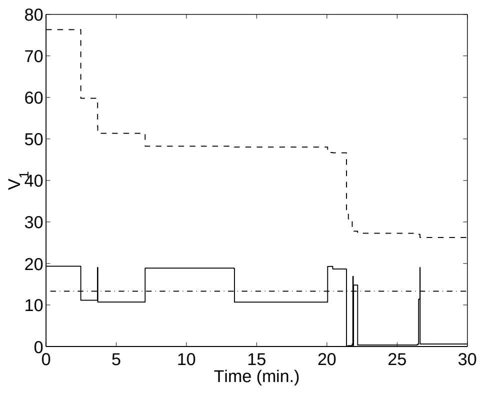


Figure 11.17: The criterion function values of the different solutions in Table 11.6 (Example 11.10) as a function of the computation time (using the neural network in Figure 11.15): The criterion function values after using only MILP (dashed line), after improving the results by using a neural network (solid line), and the mean result from training the neural network with random initial values (dash-dotted line).

as a change detection algorithm. The formulation (11.12) is used, taking only data from time $t_0, \ldots, t_0 + L - 1$ into account, where L is the length of the window. Furthermore, only one switch is allowed in each window. Hence, the MILP solved takes the form

$$\begin{aligned} &\min_{s,\theta,z,\delta} \sum_{t=t_0}^{t_0+L-1} s_t \\ \text{subj. to}\quad & s_t \geq y(t) - \varphi^T(t)\theta_0 - z_1(t) + z_2(t) \\ & s_t \geq \varphi^T(t)\theta_0 + z_1(t) - z_2(t) - y(t) \\ & \delta(t_0) \leq \cdots \leq \delta(t_0 + L - 1) \\ & \text{inequalities (11.9) with } \delta_1(t) = \delta_2(t) = \delta(t) \end{aligned} \tag{11.64}$$

Note that we only need two hinges (one positive and one negative) and L discrete variables since only one switch is allowed, compared to the MN discrete variables needed in (11.12). (If the PWARX structure we would like to identify just contains positive hinges, we would only need one (positive) hinge in (11.64).) Furthermore, the inequalities $\delta(t_0) \leq \cdots \leq \delta(t_0 + L - 1)$ also help to reduce the complexity drastically. As an alternative to (11.64) we could also use one of the methods in Section 11.2.1 with $M = 1$.

In each position t_0 of the window, the fit of the local hinging hyperplane model (i.e., the optimal value of the cost function in (11.64)) is compared to the fit of a linear model over the same window. The value of the relative improvement of the cost function,

$$k_{t_0} = 1 - \frac{V^*_{HHARX}}{V^*_{ARX}} \tag{11.65}$$

is assigned to the time point of the change, and as the window is moving, these values are summed up (for each time point). If the sum of the relative improvements for a certain time point exceeds a threshold K_0, chosen by the user, this time point will be considered as a possible switch time.

The advantage of using (11.64) instead of a standard change detection algorithm, e.g., Brandt's GLR method (see, e.g., [58]), is that the latter does not require linear separability between the classes; nor does it take the continuity of the PWA function into account.

After having obtained the estimated possible time points of the switches as described above, we solve (11.12) or (11.11), but using the same δ variable for all samples lying in the same time interval between two consecutive possible switches. This will force the samples to belong to the same submodel, and will reduce the complexity considerably. To summarize, the algorithm consists of two phases:

1. Use a sliding window with a local MILP algorithm to detect possible switches and divide the time series into segments.

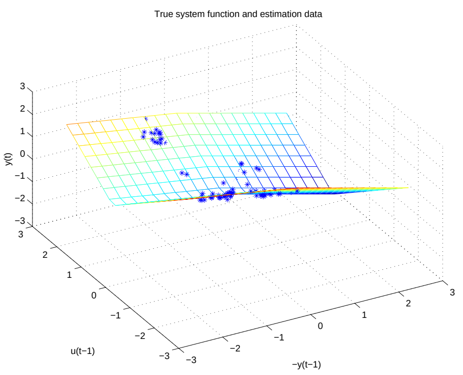
2. Use an MILP to simultaneously assign the different segments to different submodels and estimate the parameters of the submodels.

Once again, note that in the first step, the MILP solved just uses two hinges, independently of how many hinge functions the final global model contains.

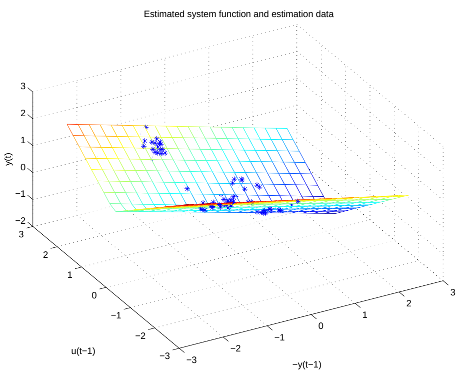
Example 11.11 *The system*

$$\begin{aligned} y(t) = &-0.3 + 1.2y(t-1) - u(t-1) \\ &+ \max\{-1.2 + 2u(t-1), 0\} \\ &- \max\{-0.2y(t-1), 0\} + e(t) \end{aligned} \tag{11.66}$$

where $e(t)$ is white Gaussian noise with variance 0.01, is identified using 100 data samples. The true system function and the data samples are shown in Figure 11.18(a). The proposed sliding window algorithm was used with $L = 15$ and $K_0 = 1$. This resulted in the system shown in Figure 11.18(b). Table 11.7 shows the values of the objective function V_1 (11.6) for the true system and the identified model, for the estimation data and a set of validation data. As can be seen, the identified model shows a good performance. The computation time running CPLEX *on a 333 MHz Pentium II laptop (128 MB RAM) was 144 s (42 s for the sliding windows and 102 s for the final large MILP). This should be compared to solving the MILP* (11.12) *directly, which did not return a solution within a maximum allotted time of 3 hours on the same computer.*



(a) System function (11.66) and estimation data.



(b) Identified model.

Figure 11.18: Identification of (11.66).

Table 11.7: Identification of (11.66) – values of the objective function V_1.

	True system	Identified model
Estimation data	7.8213	7.4833
Validation data	7.8668	8.6777

11.6.1 Complexity

The advantage of the described sliding window algorithm, compared to solving (11.12) or (11.11) directly, lies in the reduction of the computational complexity. In the sliding window phase, the complexity is linear in the number of data N when using a window of fixed size, as opposed to the exponential complexity of (11.12) and (11.11).

For the second phase, the complexity is closely related to the number of possible switches. Here, the thresholding procedure makes it possible to explicitly trade off between complexity of the algorithm and optimality: The higher the threshold value, the fewer possible switch times will be considered. If it is high enough, no switches will be allowed, which means that all samples will be forced to belong to the same submodel, and we will end up with a linear model. If, on the other hand, the threshold value is chosen to be zero, every time point will be considered as a possible switch time, and we will again get the globally optimal solution.

As previously mentioned, the described algorithm requires the system to switch only seldom, staying in each submodel for a period at least in the order of the window length, L. The general issue of designing input signals having the desired properties of sufficiently exciting the modes of the system and letting the system switch seldom is a subject for future research.

11.6.2 Approximating General Nonlinear Systems

To give another example of the described sliding window algorithm, the problem of approximating a simple nonlinear system is considered. The capability of approximating arbitrary nonlinear systems is an interesting issue. Since hinging hyperplane functions have the universal approximation property, as mentioned in Section 9.2.2, they can (under mild conditions) approximate any function arbitrarily well, given a large enough number of hinges. As a very simple illustration, a quadratic NARX (nonlinear ARX) system is approximated by a hinging hyperplane model in the following example.

Example 11.12 *Consider the system*

$$y(t) = -0.5y(t-1)^2 + 0.7u(t-1) + e(t) \tag{11.67}$$

where $e(t)$ is white Gaussian noise with variance 0.01, is identified using 100 data samples. The input is designed to make the output change sign only seldom (about every 25 samples). The true system function and the data samples are shown in Figure 11.19(a). Using the sliding window algorithm with one hinge, $L = 10$, and a threshold $K_0 = 1$, resulted in the system shown in Figure 11.19(b). We can see that the parabola is approximated by the hinge in a natural way. The computation time running Cplex *on a 333 MHz Pentium II laptop (128 MB RAM) was 19.7 s (17.7 s for the sliding windows and 2 s for the final large MILP). Solving the MILP* (11.12) *directly required about 1300 s of computations.*

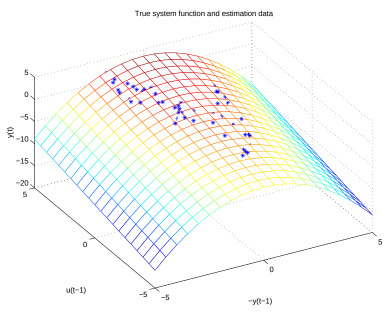
If we instead use three hinges to approximate the true system function, we get the result shown in Figure 11.19(c). The computation time was 152 s (20 s for the sliding windows and 132 s for the final large MILP).

11.7 Related Approaches

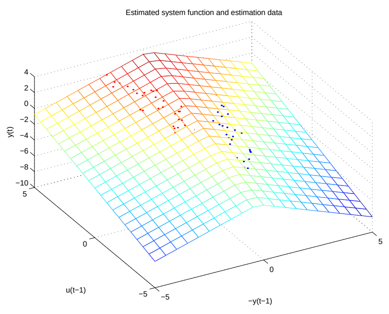
The formulation of the piecewise affine system identification problem using discrete variables $\delta_i(t)$ can also be used as a basis for other algorithms than those described in this chapter. To highlight the relation between the MILP/MIQP approach and other, previously proposed algorithms, two Newton-like methods using $\delta_i(t)$ will be discussed here. The second algorithm has, to the author's knowledge, not appeared previously in the literature.

Consider once more the predicted output (11.2), repeated here for convenience:

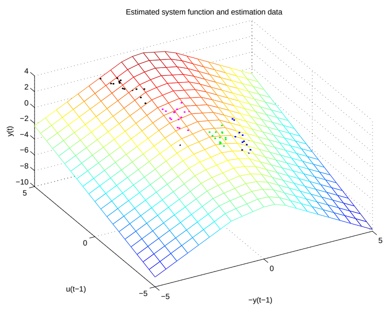
$$\hat{y}(t|\theta) = \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm \max\{\varphi^T(t)\theta_i, 0\} \tag{11.68}$$



(a) System function (11.67) and estimation data.



(b) Identified model using one hinge.



(c) Identified model using three hinges.

Figure 11.19: Identification of (11.67).

Using the definition (11.8) of $\delta_i(t)$, we can rewrite (11.68) as

$$\begin{aligned}\hat{y}(t|\theta) &= \varphi^T(t)\theta_0 + \sum_{i=1}^{M} \pm\delta_i(t)\varphi^T(t)\theta_i \qquad (11.69)\\ &= \underbrace{\begin{pmatrix}\varphi^T(t) & \pm\delta_1(t)\varphi^T(t) & \dots & \pm\delta_M(t)\varphi^T(t)\end{pmatrix}}_{\Phi^T(t,\delta(t))}\underbrace{\begin{pmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_M\end{pmatrix}}_{\theta}\end{aligned}$$

where we also introduce $\delta(t) = \begin{pmatrix}\delta_1(t) & \dots & \delta_M(t)\end{pmatrix}$ for notational convenience. ($\Phi^T(t, \delta(t))$ is not to be confused with Φ in (2.20) in Part I.) The criterion function

V_2 from (11.5) can now be written

$$V_2 = \sum_{t=1}^{N} (y(t) - \Phi^T(t, \delta(t))\theta)^2 \tag{11.70}$$

This form can be used as the starting point for several different identification algorithms not using MILP/MIQP, often related to the approaches presented in Chapter 10. Here, some of the possible approaches will be discussed.

For a given set of $\delta(t)$, (11.70) would be minimized by

$$\hat{\theta} = \left(\sum_{t=1}^{N} \Phi(t, \delta(t))\Phi^T(t, \delta(t)) \right)^{-1} \sum_{t=1}^{N} \Phi(t, \delta(t))y(t)$$

However, the fact that $\delta(t)$ is determined by θ and $\varphi(t)$ has been neglected here. What one could do is to compute the value of $\delta(t)$ corresponding to $\hat{\theta}$. Let us call this value $\hat{\delta}^{(0)}(t)$. Now we can iterate, letting

$$\hat{\theta}^{(k)} = \left(\sum_{t=1}^{N} \Phi(t, \hat{\delta}^{(k-1)}(t))\Phi^T(t, \hat{\delta}^{(k-1)}(t)) \right)^{-1} \sum_{t=1}^{N} \Phi(t, \hat{\delta}^{(k-1)}(t))y(t) \tag{11.71a}$$

and

$$\hat{\delta}_i^{(k)}(t) = \begin{cases} 0 & \varphi^T(t)\hat{\theta}_i^{(k)} < 0 \\ 1 & \varphi^T(t)\hat{\theta}_i^{(k)} \geq 0 \end{cases} \tag{11.71b}$$

When, for some k, $\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)}$, we have reached a local minimum.

This algorithm is a variant of the hinge-finding algorithm due to Breiman, described in [22]. The difference between them is that this version considers all hinge functions simultaneously, instead of fitting one hinge function at a time. The last property makes it belong to the first category of approaches listed in Chapter 10. Like the hinge-finding algorithm, there is no guarantee for convergence, but the algorithm can be modified to guarantee convergence. In this case it becomes the same algorithm as in [124].

Another option, inspired by one of the algorithms proposed in [69], is to solve the convex QP

$$\begin{aligned} \min_{\theta} \quad & V_2 = \sum_{t=1}^{N} (y(t) - \Phi^T(t, \delta(t))\theta)^2 \\ \text{subj. to} \quad & \begin{cases} \varphi^T(t)\theta_i \leq 0 & \text{if } \delta_i(t) = 0 \\ \varphi^T(t)\theta_i \geq 0 & \text{if } \delta_i(t) = 1 \end{cases} \end{aligned} \tag{11.72}$$

for given $\delta_i(t)$. The algorithm is as follows:

Algorithm 11.1

1. *Assign initial values to* $\delta_i(t)$ *and solve the corresponding QP* (11.72).
2. *There are two possibilities for the resulting* $\hat{\theta}$*:*
   - *If* $\hat{\theta}$ *does not satisfy* $\varphi^T(t)\hat{\theta}_i = 0$ *for any* $t = 1, \dots, N$, $i = 1, \dots, M$, *this means that* $\hat{\theta}$ *lies in the interior of the feasible region of* (11.72). *In other words, no data sample* $\varphi(t)$ *lies on a hinge. This means that a local minimum for the original problem* (11.70) *is found, and we stop.*
   - *Otherwise, we collect all pairs* (t, i) *such that* $\varphi^T(t)\hat{\theta}_i = 0$, *and change the values of the corresponding* $\delta_i(t)$ *(or a subset of them). This is the same as assigning some of the data points* $\varphi(t)$ *lying on a hinge to the region on the other side of the hinge. Then go to 3.*
3. *Solve* (11.72) *for the new set of* $\delta_i(t)$ *values. If* $\varphi^T(t)\hat{\theta}_i = 0$ *for the same pairs* (t, i) *as before, go to 4. Otherwise, go to 2.*
4. *If not all possible assignments of values to* $\delta_i(t)$ *for the pairs* (t, i) *with* $\varphi^T(t)\hat{\theta}_i = 0$ *are tested, choose a new assignment and go to 3. Otherwise, we have found a local minimum, and stop.*

Note that the value of V_2 will never increase, since the optimal solution of the previous problem is a feasible solution to the new problem as well. This means that the algorithm will converge.

This algorithm is almost a damped Newton algorithm, as explained in Section 10.1.2 for the corresponding algorithm in [69]. The differences compared to the algorithm in [69] are that hinging hyperplanes are used instead of hinging sigmoids, and that all parameters are updated simultaneously.

11.8 Conclusions

In this chapter, an approach to piecewise affine system identification using mixed-integer programming has been considered. The theoretical advantage of this approach, compared to, e.g., using standard local minimization tools, is that we are guaranteed to find the global optimum in a finite number of steps. Furthermore, an upper bound on the number of steps can be given. In practice, however, the computational complexity is too large to make the approach applicable to medium-sized or large identification problems of general piecewise affine systems.

In spite of this, using MILP/MIQP for identification might be a feasible option in some cases. As described in Section 11.4, some model structures, like the piecewise affine Wiener models, allow a significant reduction of the complexity, due to the fact that the nonlinearity is one-dimensional. Another, potentially attractive, way of utilizing the MILP/MIQP formulation was described in Section 11.5, where intermediate solutions from the MILP/MIQP solver were used as initial values for a local minimization algorithm. One problem with the MILP/MIQP formulations is that the computational complexity increases quite rapidly with the number of experimental data. However, if we are only interested in obtaining good initial values for another algorithm, we might very well choose a subset of the experimental dataset, consisting of, say, a few hundred data samples, that are well distributed over the state-space, and use these for the MILP/MIQP algorithm.

For the case when the estimation data not so frequently switch between different modes, a change detection approach was proposed. However, one should keep in mind that the task of designing a seldom-switching input, which at the same time is persistently exciting, might not be an easy task. Furthermore, much could probably be done to develop and improve the proposed algorithm.

Finally, the relation between the mixed-integer programming approach and some Newton-like methods was pointed out, and an extension of an algorithm proposed in [69] was given.

There are still issues that need to be investigated. One interesting topic is what conditions can be given to ensure that the experimental data is persistently exciting, given different model structures. For continuous-time piecewise affine systems in Chua's canonical form, some work has been done on this [75]. The MILP/MIQP solving strategies could probably also be specialized to exploit the structure of this kind of problems. More experiments could be done to compare the performance of different algorithms. In particular, the algorithms of Section 11.5 should only be seen as a first step in exploring the approach of using intermediate results from an MILP/MIQP solver. These algorithms could probably be developed substantially.

Part III

Robust Verification

12

Robust Verification

As mentioned in Section 1.3, the problem of verification is to prove that a set of bad states is always avoided, or alternatively that a set of target states is actually reached. These types of questions often arise in applications in connection with safety issues. Verification of discrete systems is a well-established research field, with a broad range of applications. With the growing interest of hybrid systems in the last decade, verification of hybrid systems has also found a number of applications in control. One application area where verification methods for hybrid systems have been applied is the chemical industry, where safety is an important issue [44]. Another application example is the verification of some properties of the landing gear system of a certain Swedish aircraft [43, 114]. Verification has also been used for special cases of a collision avoidance system for air-traffic around airports [93].

In this chapter, the main question will be the one of proving that some states are never reached. As most system models contain uncertainties in one way or the other, an important question is how large uncertainties can be tolerated before the verified properties can no longer be guaranteed to hold. The system may also be disturbed by noise, which must be taken into account in the verification. These two problems will be addressed.

We will consider systems, where the state-space is partitioned into several polyhedral regions, with one affine subsystem for each region. The mathematical definition of the system class is done in Section 12.1. As in Example 1.5, we will study the behavior of the trajectories at the boundaries of the regions. From this information, a finite automaton which is an outer approximation of the original system can be constructed, analogously to what was done in Example 1.5. The concept of inner and outer approximations was explained in Section 1.3. We will not consider how the discrete verification problem is performed, given the approximating automaton. This is described in more detail in, e.g., [43]. Instead we will assume that we are given a set of transitions, for which we should prove if they will possibly occur or not. In other words, we expect to be given a guarantee of the kind: "If these transitions never occur, then the set of bad states will never be reached". In the example in Section 12.8, we will see that it can sometimes be quite easy to find such sets of transitions by direct inspection. The problems solved in this chapter are formulated in Section 12.2 and Section 12.3.

The methods can also be extended to find an inner approximation of the system behavior, and hence to show that a set of states will really be reached in finite time. How this can be done is studied in Section 12.7.1. In Section 12.7.2, we will show that the methods work for switched systems as well.

It would also be nice to be able to combine the verification with other objectives. Section 12.8 shows an example of how this could be done.

12.1 Some Notation and Assumptions

The systems considered in this chapter are in the form (9.2), that is,

$$\dot{x} = A(v)x + b(v), \quad x \in X(v) \ , \quad v \in \{-1, 0, 1\}^M \tag{12.1}$$

where $A(v) \in \mathbb{R}^{n\times n}$, $b(v) \in \mathbb{R}^n$ are given for all $v \in \{-1, 1\}^M$ such that $X(v)$ is nonempty (this will be explained in more detail below). The vector v is a key vector, which is a piecewise constant function of x and can be seen as a label for each region $X(v)$. The regions $X(v) \subset \mathbb{R}^n$ are polyhedra, separated from each other by M hyperplanes. These are defined by

$$\{x \in \mathbb{R}^n \mid C_i x = d_i\}, \quad i = 1, \dots, M \tag{12.2}$$

where $C_i \in \mathbb{R}^{1\times n}$ and $d_i \in \mathbb{R}$ are given. To allow for a compact representation, we let C_i form the rows in an $M \times n$ matrix C, and collect d_i into the vector d.

Now v is defined according to the following rule:

$$v_i = \begin{cases} -1 & \text{if } C_i x < d_i \\ 0 & \text{if } C_i x = d_i \\ 1 & \text{if } C_i x > d_i \end{cases} \tag{12.3}$$

In this way there is a one-to-one relationship between v and $X(v)$. Since $v_i = 0$ implies that $X(v) \subseteq \{x \in \mathbb{R}^n \mid C_i x = d_i\}$, we notice that a nonempty $X(v)$ will be contained in an $(n-1)$-dimensional affine subspace of $\mathbb{R}^n$ if and only if there is at least one zero entry in v. Nonempty polyhedra $X(v)$ corresponding to vectors

v with no zero entries will be called *full-dimensional* polyhedra. The corners of a full-dimensional polyhedron will be regions $X(v^c)$, where v^c contains (at least) n zeros. Here we will only treat the case when v^j contains exactly n zeros; other cases can be regarded as degenerate special cases of this, with several corners in the same point.

As mentioned, we assume that $A(v)$ and $b(v)$ are given for all full-dimensional polyhedra $X(v)$. For the faces, corners etc., we assume that the dynamics is given as a convex combination (whose weights might be unknown) of the dynamics of all adjacent full-dimensional polyhedra. The following example clarifies this assumption.

Example 12.1 *Consider two adjacent full-dimensional polyhedra $X(v^1), X(v^2) \in \mathbb{R}^2$, the face $X(v^f)$ between them, and the corners $X(v^{c_1})$, $X(v^{c_2})$ at the ends of the face. Figure 12.1 shows three different ways in which the trajectories could be directed, and the corresponding outer approximating automata.*

In Figure 12.1(a), the trajectories from $X(v^1)$ are pointing towards $X(v^f)$, while the trajectories in $X(v^2)$ are going away from $X(v^f)$. Since the dynamics of $X(v^f)$ is a convex combination of the two, this implies that the trajectories will go through $X(v^f)$ in the direction from $X(v^1)$ to $X(v^2)$. If the transition from $X(v^1)$ to $X(v^f)$ takes place at the right corner $X(v^{c_2})$, we might get a transition to the corner, since the trajectories at $X(v^f)$ always have a component pointing to the right.

In Figure 12.1(b), the trajectories from both regions $X(v^1)$ and $X(v^2)$ are all going inwards towards $X(v^f)$. This means that we will have a chattering *along the line. According to our assumption, the state could go either to $X(v^1)$ or $X(v^2)$ from $X(v^f)$, but as soon as it leaves $X(v^f)$, it will be immediately pushed back. In the automaton we approximate this behavior by stating that there cannot be any transitions at all from $X(v^f)$ to $X(v^1)$ or $X(v^2)$. However, since all trajectories are pointing to the right, we know that the state will eventually end up in $X(v^{c_2})$.*

Figure 12.1(c) is quite similar to Figure 12.1(b) in the sense that we will get a chattering behavior along $X(v^f)$, but here we do not know in which corner (if any) we will end up, $X(v^{c_1})$ or $X(v^{c_2})$.

12.2 Problem Formulation for Known Systems

Now consider one of the full-dimensional polyhedra $X(v)$, and how the trajectories $x(t)$ behave inside it. Let $X(v')$ be one of the faces of the polyhedron (which means that $v'_i = 0$ for some i, and $v'_j = v_j$, $j \neq i$). Basically, there are three different options for the qualitative behavior of the trajectories near $X(v')$ (see Figure 12.2):

(a) They are all exiting $X(v)$ (some may be parallel to $X(v')$).

(b) They are all entering $X(v)$ (some may be parallel to $X(v')$).

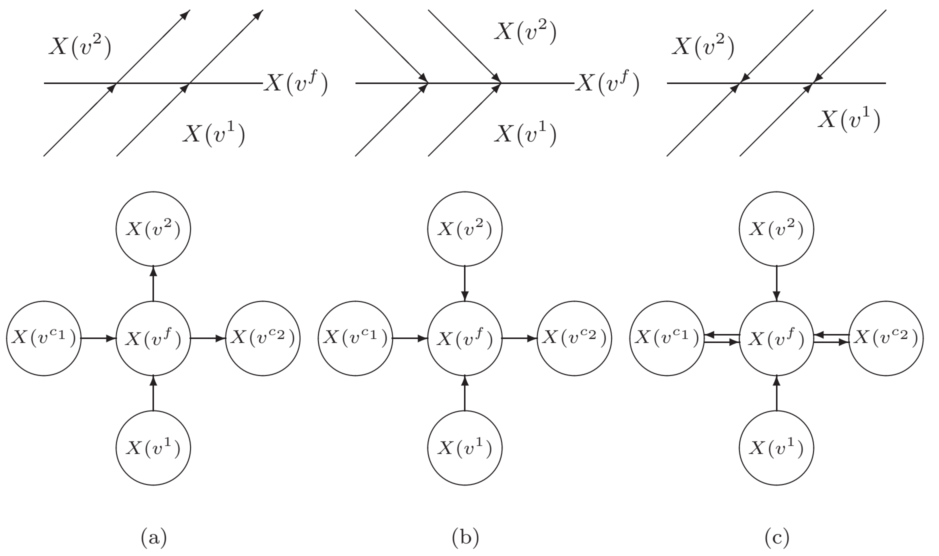


Figure 12.1: Three different cases for the dynamics at a boundary $X(v^f)$, and the corresponding outer approximating automata. $X(v^{c_1})$ is the left corner of $X(v^f)$, and $X(v^{c_2})$ is the right corner.

(c) Some trajectories are entering and some are exiting $X(v)$.

Since the system is affine inside the polyhedron, the trajectories are smooth, and therefore these three cases are the only possible options. The special case of all trajectories being parallel to $X(v')$ is included in both the first cases, but this is of little practical importance.

The three different cases will lead to different approximating automata (see Figure 12.3). What we need to find out is therefore which of the cases we get, given $A(v)$, $b(v)$. From (12.3), it is not hard to see that C_i is a normal vector of the polyhedron face $X(v')$. Using this together with (12.1), we can rewrite the three different cases as

(a) $C_i(A(v)x + b(v)) \geq 0$ for all $x \in X(v')$*.

(b) $C_i(A(v)x + b(v)) \leq 0$ for all $x \in X(v')$.

(c) $C_i(A(v)x^1 + b(v)) > 0$ and $C_i(A(v)x^2 + b(v)) < 0$ for some $x^1, x^2 \in X(v')$.

*Note that this condition corresponds to the case "All trajectories are exiting $X(v)$" only if C_i is pointing out of $X(v)$, that is, if $v_i = -1$. Otherwise conditions (a) and (b) are switched. To avoid this, one could replace condition (a) by "$v_i C_i(A(v)x + b(v)) \leq 0$ for all $x \in X(v')$", and similarly for condition (b). However, this is mainly a matter of taste.

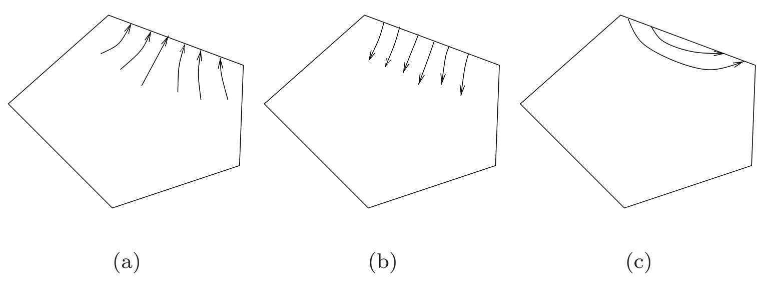


Figure 12.2: Three options for the behavior of the trajectories in the vicinity of a polyhedron face.

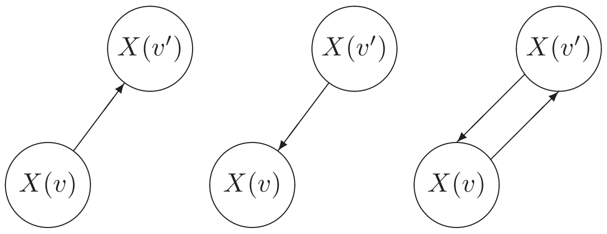


Figure 12.3: Parts of the automata corresponding to the three cases in Figure 12.2.

These conditions can all be easily checked for given $A(v)$, $b(v)$, C and d, for example by finding the maximum and minimum values on $X(v')$ of $C_i(A(v)x+b(v))$, yielding two LP problems.

12.3 The Problem of Robust Verification

In Section 12.2, we assumed that the dynamics of the system was completely known. In practice, however, there will almost always be model errors, and the systems might be disturbed by noise. To that end, let us introduce some uncertainty in our system model:

$$\dot{x} = (A(v) - \Delta(v))x + b(v) - \delta(v), \quad x \in X(v) \ , \quad v \in \{-1, 0, 1\}^M \tag{12.4}$$

where v is defined by

$$v_i = \begin{cases} -1 & \text{if } C_i x < d_i + \gamma_i \\ 0 & \text{if } C_i x = d_i + \gamma_i \\ 1 & \text{if } C_i x > d_i + \gamma_i \end{cases} \tag{12.5}$$

Depending on the applications, the matrices $\Delta(v) \in \mathbb{R}^{n\times n}$, $\delta(v) \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}^M$ can be viewed either as uncertainties in the model, or as matrices of our choice (the latter case can occur, e.g., in the control design process). We can see that the different matrices affect the system dynamics in different ways: $\Delta(v)$ and $\delta(v)$ affect the dynamics of the different subsystems, while γ affects the partitioning of the state-space.

We will only allow values of γ that do not affect the topology of the state-space partition compared to the case $\gamma = 0$. In other words, the regions $X(v)$ will always have the same number of faces and the same neighbors as they would have if the separating hyperplanes were defined by $Cx = d$. This also means that we will not allow regions to disappear or new regions to be created when changing γ. An equivalent way of stating this assumption is to say that for each corner $X(v^c)$, v^c should remain constant, and not be changed because a γ value causes the corner to cross another hyperplane. The following example illustrates this.

Example 12.2 (Preserving topology) *Consider the polyhedral partition of the state-space in Figure 12.4 (which is the same as in Example A.1). The partition can be described using the matrices*

$$C = \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix}, \quad d = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

The open polytope corresponding to P in Example A.1 can now be written as $X(v)$, where $v = \begin{pmatrix} -1 & -1 & 1 \end{pmatrix}^T$ (P is the closure of this open polytope).

Now assume that we do not know where the real location of the hyperplane corresponding to the third row of C and d (so far modelled by $x_1 = -1$) is. We can include this uncertainty in the model by writing $x_1 = -1 + \gamma_3$. Now, if $\gamma_3 > 1.5$, we can see that the hyperplane is moved to the right of the corner where the other two hyperplanes meet. In this way, the polytope P disappears, and a new polytope is created. To avoid these kinds of phenomena, we can require that $\gamma_3 < 1.5$.

Assuming (12.4), the three cases from the previous section now become

(a) $C_i[(A(v) - \Delta(v))x + b(v) - \delta(v)] \geq 0$ for all $x \in X(v')$.

(b) $C_i[(A(v) - \Delta(v))x + b(v) - \delta(v)] \leq 0$ for all $x \in X(v')$.

(c) $C_i[(A(v) - \Delta(v))x^1 + b(v) - \delta(v)] > 0$ and $C_i[(A(v) - \Delta(v))x^2 + b(v) - \delta(v)] < 0$ for some $x^1, x^2 \in X(v')$.

An interesting question is: How much could $\Delta(v)$, $\delta(v)$ and γ change, without changing the qualitative behavior at each face of the polyhedron, i.e., without one face switching from, say, case (a) to case (c)? In other words, for what values of $\Delta(v)$, $\delta(v)$ and γ do the different cases occur? This is the main question of this chapter.

Depending on how many of the parameters are varied simultaneously, this question may be more or less difficult to answer. If all of them are varied, the problem is

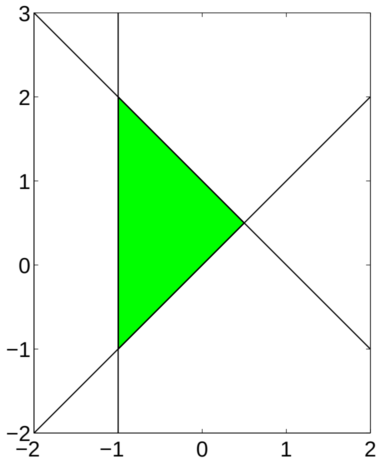


Figure 12.4: The polytope in Example 12.2.

a nonconvex quadratic problem, and will just be mentioned shortly. In the following section the question is answered for some different combinations of parameter variations, and an example is given in Section 12.8. We will mostly consider the two first cases (all trajectories exiting/entering the polytope), since the solution sets of these problems will turn out to be the easiest to describe. The solution sets for the third case can be obtained, either as the complement of the union of the other solution sets, or in some problems as a union of hyperplanes (see [127]).

12.4 Solutions to the Robust Verification Problems

Let us now consider the problem corresponding to case (a) of the previous sections. Rearranging the terms, we can write it as

$$C_i(A(v)x + b(v)) \geq C_i(\Delta(v)x + \delta(v)) \quad \text{for all } x \in X(v') \tag{12.6}$$

This form has a natural interpretation: On the left hand side we have the nominal flow through the face of the polyhedron, and the right hand side is the part of the flow that is affected by the variable matrices $\Delta(v)$ and $\delta(v)$. What (12.6) tells us is that the variable flow (the one caused by $\Delta(v)$ and $\delta(v)$) must be made small enough, or that $X(v')$ must lie in a region where the variable flow is small enough; otherwise we will get a total flow in the other direction from what was specified.

In the following subsections we solve this problem for some special cases, when at least one of $\Delta(v)$, $\delta(v)$ and γ is zero. Problem (b) can be solved completely analogously. Problem (c) is considered in a special case in Section 12.4.1.

12.4.1 $\Delta(v) = 0$, $\gamma = 0$, $\delta(v)$ is Varied

The simplest problem arises when we only allow $\delta(v)$ in (12.4) to vary, and let $\Delta(v)$ and γ equal 0. Equation (12.6) then takes the form

$$C_i(A(v)x + b(v)) \geq C_i\delta(v) \quad \text{for all } x \in X(v') \tag{12.7}$$

This problem can be solved immediately by solving the LP problem

$$\begin{aligned} \min_x \quad & C_i(A(v)x + b(v)) \\ \text{subj. to} \quad & x \in \overline{X(v')} \end{aligned} \tag{12.8}$$

where $\overline{X(v')}$ is the closure of $X(v')$ (see Section A.3). Denoting the solution of the LP problem by x_*, the solution set to (12.7) is

$$S_\delta = \{\delta \in \mathbb{R}^n \bigm| C_i\delta \leq C_i(A(v)x_* + b(v))\} \tag{12.9}$$

Problem (b) is solved by maximizing instead of minimizing $C_i(A(v)x+b(v))$ to get the limit x^*, and the solution set consists of all δ satisfying $C_i\delta \geq C_i(A(v)x^* + b(v))$. The values of δ for which $C_i\delta$ lies in between x_* and x^* are solutions to problem (c). Thus, in this case all solution sets will be convex.

12.4.2 $\gamma = 0$, $\Delta(v)$ and $\delta(v)$ are Varied

When letting both $\Delta(v)$ and $\delta(v)$ vary, we need to find a direct representation (see Section A.3) of $X(v')$ to get a solution to (12.6). For the case when $X(v')$ is bounded, the direct representation will be on the following form:

$$X(v') = \{\sum_{j=1}^{r} \lambda_j x^j \bigm| \lambda_j \in \mathbb{R},\ \lambda_j > 0,\ \sum_{j=1}^{r} \lambda_j = 1\} \tag{12.10}$$

Here $x^j \in \mathbb{R}^n$, $j = 1, \dots r$ are the corners of $X(v')$.

According to Equation (A.11), when $X(v')$ is unbounded, we must include in the direct representation some vectors, $x^{r+1}, \dots, x^{r+h}$, which are parallel to the unbounded edges of $X(v')$. A direct representation of $X(v')$ would then be:

$$X(v') = \{\sum_{j=1}^{r+h} \lambda_j x^j \bigm| \lambda_j \in \mathbb{R},\ \lambda_j > 0,\ \sum_{j=1}^{r} \lambda_j = 1\} \tag{12.11}$$

Note that, as in (A.11), $\lambda_{r+1}, \dots, \lambda_{r+h}$ are not included in the set of λ_j that should sum up to one; they can be arbitrarily large. In some cases there are no corners, e.g., when $X(v')$ consists of an entire hyperplane. As mentioned in Section A.3, in such cases, a "dummy" corner must be introduced somewhere in $X(v')$.

For notational simplicity, let us drop the argument v of $A(v)$ etc. for a while. Now the following theorem gives the solutions to our problem.

Theorem 12.1
Consider the system given by (12.4) *and* (12.5). *Assume that* $\gamma = 0$. *Then the set of solutions to the problem* (12.6) *is given by*

$$\begin{aligned} S_{\Delta\delta} = \{(\Delta, \delta) \;\big|\; & C_i(Ax^j + b) \geq C_i(\Delta x^j + \delta), \quad j = 1, \ldots, r; \\ & C_i A x^{r+j} \geq C_i \Delta x^{r+j}, \quad j = 1, \ldots, h\} \end{aligned} \tag{12.12}$$

Proof To show that the inequality (12.6) is satisfied for an arbitrary point $x \in X(v')$, we use the direct representation (12.11):

$$\begin{aligned} C_i(Ax + b) &= C_i(A \sum_{j=1}^{r+h} \lambda_j x^j + b) \\ &= \sum_{j=1}^{r} \lambda_j C_i(Ax^j + b) + \sum_{j=r+1}^{r+h} \lambda_j C_i A x^j \\ &\geq \sum_{j=1}^{r} \lambda_j C_i(\Delta x^j + \delta) + \sum_{j=r+1}^{r+h} \lambda_j C_i \Delta x^j \\ &= C_i(\Delta \sum_{j=1}^{r+h} \lambda_j x^j + \delta) \\ &= C_i(\Delta x + \delta) \end{aligned}$$

To see that the first r conditions of (12.12) are necessary, we just need to notice that they are precisely (12.6) for the corners of $X(v')$. If a corner would not satisfy (12.6), then for continuity reasons, there would be points in $X(v')$ not satisfying (12.6) either. To show the necessity of the last h conditions, fix Δ and δ, suppose that $C_i A x^{r+j} < C_i \Delta x^{r+j}$ for some j, and study $x = x^1 + \lambda x^{r+j} \in X(v')$, where $\lambda > \frac{C_i(Ax^1 + b - \Delta x^1 - \delta)}{C_i(\Delta x^{r+j} - Ax^{r+j})} > 0$. We then have

$$\begin{aligned} C_i(Ax + b) &= C_i(Ax^1 + b) + \lambda C_i A x^{r+j} \\ &= C_i(Ax^1 + b) + \lambda C_i(Ax^{r+j} - \Delta x^{r+j}) + \lambda C_i \Delta x^{r+j} \\ &< C_i(Ax^1 + b) - C_i(Ax^1 + b - \Delta x^1 - \delta) + \lambda C_i \Delta x^{r+j} \\ &= C_i(\Delta x^1 + \delta) + \lambda C_i \Delta x^{r+j} \\ &= C_i(\Delta x + \delta) \end{aligned}$$

where the direction of the inequality follows since λ is multiplied by a negative number. This means that (Δ, δ) is outside the solution set, and the necessity is shown. □

When $X(v')$ is a polytope, Theorem 12.1 can interpreted as follows: The inequality (12.6) holds for all $x \in X(v')$ if and only if it holds for the corners of $X(v')$. This property follows directly from the inequality being linear in x.

Note that the solution set $S_{\Delta\delta}$ is a polyhedron in the space $\mathbb{R}^{n \times n} \times \mathbb{R}^n$, and therefore convex.

12.4.3 Multiple Requirements

So far, we have only been looking at one single polyhedron face. In most cases, the requirements may stipulate that several transitions of an approximating automaton should remain invariant. This case is easily handled by partitioning the problem into subproblems of the form treated above, and then taking the intersection of the solution sets as the solution set for the entire problem. How many transitions we need to consider will depend on the system and what we want to verify. For example, if all we are interested in is keeping the state on one side of a hyperplane, we only need to consider transitions through this hyperplane. It should be noted that considering fewer transitions will lead to a larger – and therefore less conservative – solution set, and will also require less computations.

12.4.4 $\Delta = 0$, δ and γ are Varied

The problem gets more complicated as soon as γ is not fixed anymore. We immediately notice from (12.4) and (12.5) that the regions $X(v)$ will no longer be fixed, but vary with γ. We also have to consider several regions $X(v)$ simultaneously, since moving a hyperplane will affect all regions adjacent to it. As mentioned in Section 12.3, we will only allow values of γ that keep the topology of the state-space regions invariant. If we do not make this requirement, new regions – for which we do not know the system dynamics – can be created by moving the hyperplanes defined by $Cx = d + \gamma$.

With this requirement, and with $\Delta(v) = 0$ for all v, the problem is still convex, as will be shown in the following. What we need to do first is to express the corners as functions of γ. Having done that, we can use these functions to express the topology preservation requirements and the properties to be verified.

First, remember that each corner x^j of the full-dimensional polyhedron $X(v)$ is itself a region $X(v^j)$, where v^j contains n zeros. The zeros of v^j correspond to the equations $C_i x^j = d_i + \gamma_i$ that x^j satisfies. To be able to pick out the corresponding rows of C_i, d_i, and γ_i, we need to introduce some new matrices.

Let $D_{[v^j]} = \operatorname{diag}(v^j)$. From $D_{[v^j]}$ we then construct $Q_{[v^j]} \in \mathbb{R}^{(M-n)\times M}$ by deleting all rows containing only zeros. Similarly, we define $P_{[v^j]} \in \mathbb{R}^{n\times M}$ by deleting all rows in $I - D^2_{[v^j]}$ containing only zeros.

Now $P_{[v^j]}$ has the following property: When multiplying another matrix from the left by $P_{[v^j]}$, it picks out the rows corresponding to the zero entries of v^j. $Q_{[v_j]}$, on the other hand, picks out the rows not picked out by $P_{[v^j]}$, and furthermore multiplies the rows corresponding to the -1 entries of v^j by -1.

With this notation, we can pick out the equalities in (12.5) by writing

$$P_{[v^j]} C x^j = P_{[v^j]}(d + \gamma)$$

Since x^j is uniquely determined by the equalities in (12.5), $P_{[v^j]}C$ will always be invertible. Hence, we can write

$$x^j = (P_{[v^j]}C)^{-1} P_{[v^j]}(d + \gamma) \tag{12.13}$$

Example 12.3 *Consider once again the partition in Examples A.1 and 12.2. The corner* $x^j = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$ *corresponds to the key vector* $v^j = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$. *Hence,*

$$D_{[v^j]} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad Q_{[v^j]} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \quad P_{[v^j]} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Assume now that the positions of the hyperplanes are not certain, but that they could be translated by γ. *By* (12.13), *we can write the corner* x^j *as*

$$\begin{aligned} x^j &= (P_{[v^j]}C)^{-1}P_{[v^j]}(d+\gamma) \\ &= \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix}\right]^{-1}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}\left(\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} + \gamma\right) \\ &= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{-1}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix}\right) \\ &= (1+\gamma_1)\begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} + \gamma_2\begin{pmatrix} 1/2 \\ -1/2 \end{pmatrix} \end{aligned}$$

Note that if $\gamma = 0$, *we retain the original point.*

The requirement that the topology should be preserved is the same as saying that it should always be possible to express each corner x^j as a region $X(v^j)$, where v^j must be constant. In other words, the equalities and inequalities (12.5) that define v^j should remain invariant. Of course, the equalities are satisfied (x^j is constructed from them in (12.13)). Thus, we get the following set of inequalities for each corner $X(v^j)$ and all $i = 1, \ldots, M$:

$$\begin{aligned} C_i(P_{[v^j]}C)^{-1}P_{[v^j]}(d+\gamma) &< d_i + \gamma_i \quad \text{if} \quad v_i^j = -1 \\ C_i(P_{[v^j]}C)^{-1}P_{[v^j]}(d+\gamma) &> d_i + \gamma_i \quad \text{if} \quad v_i^j = 1 \end{aligned} \tag{12.14}$$

or more compactly

$$Q_{[v^j]}\left(C(P_{[v^j]}C)^{-1}P_{[v^j]} - I\right)(d+\gamma) \succ 0 \tag{12.15}$$

where $\succ$ denotes componentwise inequality.

Example 12.4 *For the point x^j in Example 12.3, the inequality* (12.15) *becomes*

$$
\begin{aligned}
0 &< \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \cdot \\
&\cdot \left[\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix} \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right] \cdot \\
&\cdot \left(\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} + \gamma \right) \\
&= \frac{3}{2} + \frac{1}{2}\gamma_1 + \frac{1}{2}\gamma_2 - \gamma_3
\end{aligned}
$$

Here we can note that, if $\gamma_1 = \gamma_2 = 0$, we get back the requirement that was imposed on γ_3 in Example 12.2.

What we need to do now is to take care of the requirements on the flow through the surfaces. This can be done completely analogously to what was done in Section 12.4.2, but with $\Delta = 0$. However, we need to plug in the expression for the corners into the inequalities of (12.12), to get, e.g.,

$$
\begin{aligned}
&C_i(A(P_{[v^j]}C)^{-1}P_{[v^j]}(d+\gamma)+b) \geq C_i\delta, \quad j = 1, \ldots, r \qquad (12.16) \\
&C_iAx^{r+j} \geq 0, \quad j = 1, \ldots, h
\end{aligned}
$$

for problem (a). Since the surfaces are only translated, the directions of the unbounded edges do not change, so x^{r+j} are not affected by γ. Inequalities like these, together with (12.15), give the final solution set. As can be seen, all inequalities are linear in δ and γ, and the resulting solution set is therefore a polyhedron.

12.4.5 Δ, δ and γ are Varied

The final case, when all parameters are allowed to vary, is quite similar to the one when only Δ is fixed. Like in Section 12.4.4 we get the inequalities (12.15). We also get the requirements on the flow through the surfaces of a region by plugging in the expression (12.13) for x^j into (12.12):

$$
\begin{aligned}
&C_i\left((A-\Delta)(P_{[v^j]}C)^{-1}P_{[v^j]}(d+\gamma)+b-\delta\right) \geq 0, \quad j = 1, \ldots, r \quad (12.17) \\
&C_iAx^{r+j} \geq C_i\Delta x^{r+j}, \quad j = 1, \ldots, h
\end{aligned}
$$

However, here the inequalities become quadratic, and the solution set is nonconvex. This makes it harder to efficiently represent and work with the solution set, and therefore this case will not be further discussed.

12.5 Interpretations

Perhaps the most obvious interpretation is to view $\Delta(v)$, $\delta(v)$ and γ in (12.4) and (12.5) as uncertainties due to model errors and/or noise. The methods in Section 12.4 then provide bounds for the uncertainties for the requirements of the approximating automata to hold. For natural reasons, the bounds may be very asymmetric, indicating that the system is more sensitive to certain types of model errors than to others.

The problem formulation is quite general in that no structure of $\Delta(v)$, $\delta(v)$ or γ is assumed. It should be stressed that these uncertainties by no means need to be constant over time; we have showed that as long as they are kept within the computed bounds, the verification will hold, no matter how the uncertainties vary. The only structure that is assumed is the topology of the different polyhedral regions $X(v)$. If the uncertainty has some further structure, we can parameterize $\Delta(v)$ and $\delta(v)$ accordingly, thereby reducing the dimensionality and simplifying the problem. For example, if only the upper left element of $A(v)$ is uncertain, we can write $\Delta(v)$ as

$$\Delta(v) = \begin{pmatrix} \Delta_1 & 0 \\ 0 & 0 \end{pmatrix}$$

Another example is to set $\Delta(v) = 0$ as in Section 12.4.1, which can be interpreted as a model with additive noise:

$$\dot{x} = A(v)x + b(v) - \delta(v), \quad x \in X(v)\ , \quad i = 1, \ldots, N \tag{12.18}$$

A nonzero $\Delta(v)$ can also be interpreted as multiplicative noise.

Another type of parameterization is used in the example in Section 12.8. In this parameterization, some of the elements of $\Delta(v)$ and $\delta(v)$ are common to several polyhedra.

An alternative interpretation is to consider $\Delta(v)$, $\delta(v)$ and/or γ as parameters of our choice, to be used for control design. One natural parameterization would then be $\gamma = 0$, $\delta(v) = 0$, $\Delta(v) = B(v)L(v)$, where $B(v)$ are fixed vectors that depend on the system, while we can choose $L(v)$ freely. In this way we get (piecewise) linear state feedback control, and the problem becomes that of finding the linear state feedback vectors $L(v)$ that make our system fulfill the requirements on the approximating automata. Another parameterization would be the one in Section 12.4.4, where we can regard γ as a vector that lets us place the switching surfaces of a controller in an optimal manner.

12.6 Computational Complexity

In this section, some aspects on the computational complexity will be discussed. However, no rigorous analysis is made.

The computational complexity depends on what parameters we let vary. As we could see in Section 12.4.1, when only δ is varied we just need to solve one or two LP problems for each requirement. This can be done efficiently (see, e.g., [39]).

When only γ is fixed, the complexity gets a bit worse. From Section 12.4.2 we see, that once we know a direct representation of $X(v')$, it is trivial to divide $\mathbb{R}^{n \times n} \times \mathbb{R}^n$ into the three solution sets corresponding to problem (a), (b) and (c). Conversely, if we want the solutions to be written as intersections of half-spaces (as in (12.12)), we need to know the direct representation of $X(v')$. Therefore the computational complexity for this problem is essentially identical to that of finding the direct representation. Unfortunately, the number of vectors needed in such a representation grows very quickly with the size of the problem. An upper bound for the number of corners in a polyhedron can be calculated in the following way: In a corner, n linearly independent faces meet (where n is still the dimension of the state-space). Since the polyhedron has m faces, the number of corners cannot be larger than $\binom{m}{n}$ (i.e., the binomial coefficient).

However, if we restrict ourselves to the case where the polyhedra are formed by the state-space being divided by hyperplanes (as we have done in this chapter), it is fairly easy (but still quite time-consuming) to calculate the direct representation of all the polyhedra once and for all. The total number of corners is then bounded above by $\binom{M}{n}$, where M is the number of separating hyperplanes.

Finally, letting δ and γ vary leads to a similar set of inequalities as in the previous case. However, we cannot find the corners directly, since they are functions of γ. Instead we can store the inverses $(P_{[v^j]}C)^{-1}$ for each corner. Thus, we will need n times as large storage, and about the same amount of computations as when δ and Δ vary.

12.7 Extensions

The considered methods can be extended in different ways. We will consider two kinds of extensions: Using inner instead of outer approximations (that can be used to prove that a certain region of the state-space really is reached in finite time), and extending the model class to switched systems.

12.7.1 Inner Approximations

So far, we have only dealt with the behavior of the trajectories at the border of each region. As we have seen, using the analysis of this behavior we can construct outer approximating finite automata, with the help of which we may be able to guarantee that some states are never reached. Another kind of verification, that one might be interested in, is to prove that a certain region really is reached. This can be harder, since it is not enough to consider only what happens at the borders of a region. The reason for this is that the state, being in a region $X(v)$, might never get to the border of $X(v)$ if the system has a stable equilibrium or a limit cycle inside $X(v)$. Another difficulty with this kind of verification is that even if we know that the state trajectory leaves the region $X(v)$, we might not know through which face it will exit, since this information may be lost when approximating the system with a discrete automaton. We only know that one out of a set of transitions will occur. Following the terminology of [43], we say that the transitions are *guaranteed to occur nondeterministically.*

In this thesis, we will not go into detail with how the analysis of the resulting nondeterministic approximating automata is performed. For more details about this, see [43]. Instead, a sufficient condition for the trajectories to be guaranteed to exit $X(v)$ will be given. This condition also comes from [43]:

Proposition 12.1
A sufficient condition for the state trajectory to exit a polyhedron, $X(v)$, is that the normal vector, C_i, of one of its faces, $X(v')$, satisfies

$$v_i C_i \dot{x} < 0, \quad \forall x \in \overline{X(v)} \tag{12.19}$$

Proof See [43]. □

Intuitively, the condition in Proposition 12.1 means that the velocity vector is always pointing towards the hyperplane $C_i x = d_i$ (or $C_i x = d_i + \gamma_i$ for a system with uncertainties). Now, given the face $X(v')$, we can include this case in our framework with the following theorem:

Theorem 12.2
Consider the system given by (12.4) *and* (12.5)*. Let a direct representation of $X(v)$ be given by*

$$X(v) = \{\sum_{j=1}^{s+k} \lambda_j x^j \mid \lambda_j \in \mathbb{R},\ \lambda_j > 0,\ \sum_{j=1}^{s} \lambda_j = 1\} \tag{12.20}$$

Assume that $\gamma = 0$. Then, for a given $X(v')$, the set of solutions to (12.19) *is given by*

$$\begin{aligned} S_{\Delta\delta} = \{(\Delta, \delta) \quad \Big| \quad & v_i C_i (A x^j + b) < v_i C_i (\Delta x^j + \delta), \quad j = 1, \ldots, s; \\ & v_i C_i A x^{s+j} < v_i C_i \Delta x^{s+j}, \quad j = 1, \ldots, k\} \end{aligned} \tag{12.21}$$

If we assume instead that $\Delta = 0$, the set of solutions is given by

$$\begin{aligned} S_{\delta\gamma} = \{(\delta, \gamma) \quad \Big| \quad & C_i (A (P_{[v^j]} C)^{-1} P_{[v^j]} (d + \gamma) + b) \geq C_i \delta, \quad j = 1, \ldots, s; \\ & C_i A x^{s+j} \geq 0, \quad j = 1, \ldots, k \\ & Q_{[v^c]} \left(C (P_{[v^c]} C)^{-1} P_{[v^c]} - I \right) (d + \gamma) \succ 0, \quad \forall \text{ corners } v^c\} \end{aligned} \tag{12.22}$$

Proof Analogous to the proof of Theorem 12.1 and the results in Section 12.4.4. □

Note that since in Theorem 12.2 a face $X(v')$ must be given, the condition is somewhat more restrictive than in Proposition 12.1, where it is sufficient that any of the faces satisfies (12.19).

12.7.2 Switched Systems

The methods described in this chapter can also be extended to switched systems, as defined in Section 9.1.1. Looking back at the equations, we can see that what is required is that we have a polyhedral partition of the state-space, and affine systems within each region. We should also be able to approximate the system by inner and outer approximating finite automata.

In the case of switched systems, there may be several possible systems in each region, depending on the input v to the plant. This means that the approximating automata will also need to have several states for each region, namely one state for each possible affine subsystem in that region. Switchings between these states can only occur if a border of a polyhedral region is reached, or if an external input is received. If there is no external input, essentially nothing is changed in our analysis methods above. However, if the system can receive an external input signal at any time, which causes switchings between a set of different affine subsystems, the situation is a bit different. The properties to be verified then need to be checked for all the subsystems reachable by only giving external input signals.

In the following section, an example of a switched system will be considered.

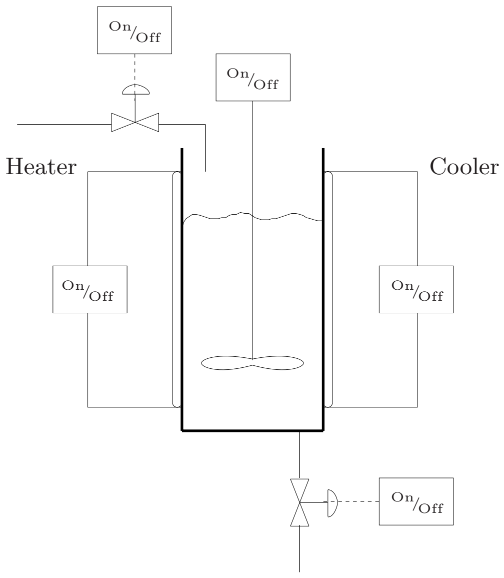


Figure 12.5: A schematic picture of the chemical reactor.

12.8 An Example: A Chemical Reactor

To demonstrate the properties of this kind of problems and solutions, we can look at a simple example. In [42], a (fictional) chemical reactor is modelled, and a control strategy is proposed, after which some properties are verified. Here we assume that some of the parameter values are uncertain, and try to determine how large errors can be tolerated before the verification is not valid any more.

12.8.1 System Model

A picture of the chemical reactor is shown in Figure 12.5. It consists of a tank containing a mixture of two fluids. When a certain temperature is reached, an exothermal reaction between the two fluids starts, giving the desired product. The temperature can be controlled by a heater and a cooler. There is also a blender helping to mix the fluid. The mixture is provided through an inflow valve. There is also a draining valve. The valves can be either open or closed.

The system model derived in [42] has two continuous state variables: the fluid level x_1 and the temperature x_2. Furthermore, there are six control signals, each one taking a value in $\{0,1\}$. They are described in Table 12.1. It could be worth mentioning that u_r is an artificial, uncontrollable signal that indicates whether or not the reaction is in progress.

Table 12.1: Inputs to the chemical reactor.

Signal	Interpretation
u_b	blender signal
u_i	inflow valve signal
u_d	draining valve signal
u_h	heater signal
u_c	cooler signal
u_r	reaction signal

The plant dynamics is described by

$$\dot{x} = A(u)x + b(u) \tag{12.23}$$

where

$$A(u) = \begin{pmatrix} -a_\mathrm{h} u_\mathrm{d} & 0 \\ 0 & -(a_{\mathrm{T}_1}(1-u_\mathrm{b}) + a_{\mathrm{T}_2} u_\mathrm{b}) \end{pmatrix} \tag{12.24}$$

$$b(u) = \begin{pmatrix} b_\mathrm{h} u_\mathrm{i} \\ b_\mathrm{heat} u_\mathrm{h} + b_\mathrm{cool} u_\mathrm{c} + b_\mathrm{reac} u_\mathrm{r} \end{pmatrix} \tag{12.25}$$

To begin with, we will assume here that the coefficients in $A(u)$ and $b(u)$ are uncertain, and that they are given by

$$\begin{pmatrix} a_\mathrm{h} \\ a_{\mathrm{T}_1} \\ a_{\mathrm{T}_2} \\ b_\mathrm{h} \\ b_\mathrm{heat} \\ b_\mathrm{cool} \\ b_\mathrm{reac} \end{pmatrix} = \begin{pmatrix} 1.23 \cdot 10^{-3} \\ 0.15 \cdot 10^{-3} \\ 0.22 \cdot 10^{-3} \\ 9.838 \\ 29.43 \cdot 10^{-3} \\ -44.15 \cdot 10^{-3} \\ 44.15 \cdot 10^{-3} \end{pmatrix} - \begin{pmatrix} \delta_\mathrm{ah} \\ \delta_{\mathrm{T}_1} \\ \delta_{\mathrm{T}_2} \\ \delta_\mathrm{bh} \\ \delta_\mathrm{heat} \\ \delta_\mathrm{cool} \\ \delta_\mathrm{reac} \end{pmatrix} \tag{12.26}$$

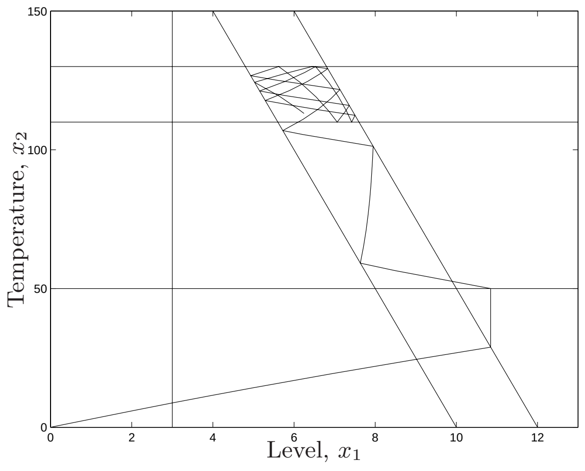


Figure 12.6: The switching hyperplanes and an example trajectory.

where the numerical values are the nominal parameter values used in [42].

The controller is designed such that the control signals are switched on or off when the state reaches certain hyperplanes. The rules are listed below (the last rule is given by the physical properties of the system):

1. $u_{\mathrm{b}} = 0$ when $x_1 < 3$, $u_{\mathrm{b}} = 1$ otherwise.
2. u_{i} is set to 0 when $25x_1 + x_2 \geq 300$, and is set to 1 when $25x_1 + x_2 \leq 250$.
3. $u_{\mathrm{d}} = 0$ when $x_2 < 50$, $u_{\mathrm{d}} = 1$ otherwise.
4. $u_{\mathrm{h}} = 1$ when $x_2 < 50$, $u_{\mathrm{h}} = 0$ otherwise.
5. u_{c} is set to 0 when $x_2 \leq 110$, and is set to 1 when $x_2 \geq 130$.
6. $u_{\mathrm{r}} = 0$ when $x_2 < 50$, $u_{\mathrm{r}} = 1$ otherwise.

Note that the system contains hysteresis in u_{i} and u_{c}. This is handled by considering each polyhedron where the hysteresis occurs as two polyhedra with two different subsystems.

In Section 12.8.4, we will consider changing the thresholds for different control actions.

The switching hyperplanes and an example trajectory are shown in Figure 12.6. For further details concerning the system model, see [42].

12.8.2 What to Verify

There are certain requirements on the controller, which are verified in [42]. These are:

1. The temperature should stay between 0 and 150.
2. The tank must not be empty, and it must not overflow. The maximum level is 13.
3. There should be an operating region with moderate temperature and fluid level which should be invariant. In [42], this region is chosen to be

$$\{x \mid 250 \leq 25x_1 + x_2 \leq 300,\ 110 \leq x_2 \leq 130\} \tag{12.27}$$

4. The operating region should always be reached from the initial states in finite time. For simplicity, and to avoid introducing additional conservatism, we will not consider this requirement in this thesis.

The requirements can be translated to mathematical formulas:

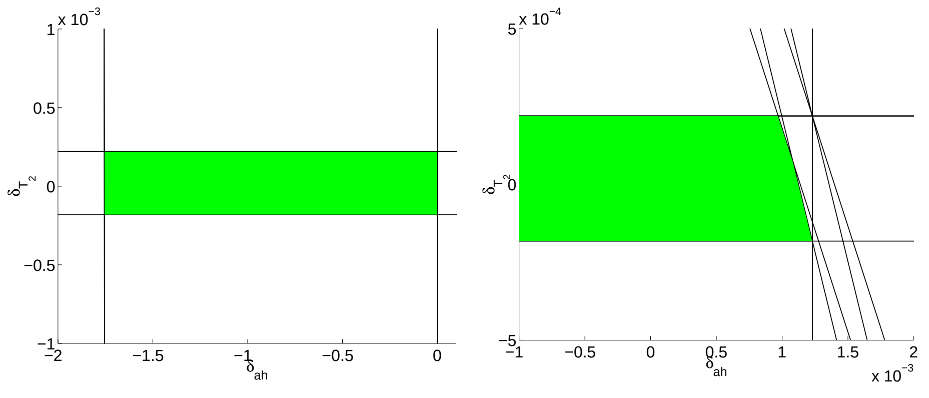
1. (a) $\dot{x}_2 \geq 0$ when $0 \leq x_1 \leq 13$, $x_2 = 0$.
   (b) $\dot{x}_2 \leq 0$ when $0 \leq x_1 \leq 13$, $x_2 = 150$.
2. (a) $\dot{x}_1 \geq 0$ when $x_1 = 0$, $0 \leq x_2 \leq 150$.
   (b) $\dot{x}_1 \leq 0$ when $x_1 = 13$, $0 \leq x_2 \leq 150$.
3. (a) $\dot{x}_2 \geq 0$ when $250 \leq 25x_1 + x_2 \leq 300$, $x_2 = 110$, and $u_c = 0$.
   (b) $\dot{x}_2 \leq 0$ when $250 \leq 25x_1 + x_2 \leq 300$, $x_2 = 130$, and $u_c = 1$.
   (c) $\begin{pmatrix} 25 & 1 \end{pmatrix} \dot{x} \geq 0$ when $25x_1 + x_2 = 250$, $110 \leq x_2 \leq 130$, and $u_i = 1$.
   (d) $\begin{pmatrix} 25 & 1 \end{pmatrix} \dot{x} \leq 0$ when $25x_1 + x_2 = 300$, $110 \leq x_2 \leq 130$, and $u_i = 0$.

We also have to know what affine subsystems $\dot{x}$ will satisfy in the different cases. We get those by considering the control rules.

12.8.3 Deriving Bounds for Parameter Uncertainties

Since we assume that the parameter values are uncertain, the question is how large the errors can get before the requirements are violated. Following the procedure in Section 12.4.2, we first need to find the direct representation to all polytope sides involved in the mathematical formulations of the requirements, which is basically the same as finding all the corners. Then, by using (12.12) we can get an exact answer to our question: The errors have to lie in a polyhedron described by

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 150 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 150 & 0 & 0 & -1 & -1 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 \\ -140 & 0 & -110 & 25 & 0 & 0 & 1 \\ -120 & 0 & -130 & 25 & 0 & 0 & 1 \\ 0 & 0 & -110 & 0 & 0 & 0 & 1 \\ -140 & 0 & -110 & 25 & 0 & 1 & 1 \\ -120 & 0 & -130 & 25 & 0 & 1 & 1 \\ 0 & 0 & 130 & 0 & 0 & -1 & -1 \\ 190 & 0 & 110 & 0 & 0 & 0 & -1 \\ 170 & 0 & 130 & 0 & 0 & 0 & -1 \\ 190 & 0 & 110 & 0 & 0 & -1 & -1 \\ 170 & 0 & 130 & 0 & 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} \delta_{\mathrm{ah}} \\ \delta_{\mathrm{T_1}} \\ \delta_{\mathrm{T_2}} \\ \delta_{\mathrm{bh}} \\ \delta_{\mathrm{heat}} \\ \delta_{\mathrm{cool}} \\ \delta_{\mathrm{reac}} \end{pmatrix} \preccurlyeq \begin{pmatrix} 9.8380 \\ 0.0294 \\ 0.0225 \\ 0.0330 \\ 0.0160 \\ 245.7977 \\ 245.8179 \\ 0.0200 \\ 245.7536 \\ 245.7738 \\ 0.0286 \\ 0.2137 \\ 0.1935 \\ 0.2579 \\ 0.2377 \end{pmatrix} \tag{12.28}$$



(a) The entire allowed region. (b) Close-up near the origin.

Figure 12.7: The allowed region for δ_{h} and δ_{T_2}, assuming that the other errors are equal to zero. Figure 12.7(b) shows a close-up of the region near the origin.

The expression above has been simplified, in that redundant inequalities have been removed. We notice immediately that the polyhedron contains the origin, which means that the nominal system satisfies the requirements. To get a more intuitive feeling for the bounds, one can also consider a subset of the errors and set the other errors to zero. For example, suppose that only a_{h} and a_{T_2} are uncertain. In order not to violate the requirements, their deviations from the nominal values have to be contained in the polyhedron shown in Figure 12.7. As we can see, the basic effect of the requirements (12.28) in this case is that δ_{T_2} has to lie in the interval $[-0.18 \cdot 10^{-3}, 0.22 \cdot 10^{-3}]$, while δ_{h} approximately can vary between -1.76 and $1 \cdot 10^{-3}$.

12.8.4 Adjusting Control Rules

Let us now turn to the question of finding bounds, inside which we can move the thresholds of the controller rules without affecting the properties to verify. Here we assume that $A(u)$ is known exactly, but that the parameters of $b(u)$ are still unknown. The hyperplanes we are going to move are

$$\begin{aligned} x_1 &= 3 + \gamma_1 \\ 25x_1 + x_2 &= 250 + \gamma_2 \\ 25x_1 + x_2 &= 300 + \gamma_3 \\ x_2 &= 110 + \gamma_4 \\ x_2 &= 130 + \gamma_5 \end{aligned}$$

These are the thresholds for the blender, the inflow valve, and the cooler signals. The other hyperplanes are assumed to be fixed (by physical properties of the plant). We would like to know how much we can change the values of $\gamma_1, \dots, \gamma_5$.

First we should make sure that we preserve the topology. Applying (12.15) to all corners yields the condition (after removing several redundant inequalities)

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ 25 & -1 & 0 & 1 & 0 \\ 25 & -1 & 0 & 0 & 1 \\ 25 & -1 & 0 & 0 & 0 \\ 25 & 0 & -1 & 1 & 0 \\ 25 & 0 & -1 & 0 & 1 \\ 25 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \gamma \prec \begin{pmatrix} 3 \\ 10 \\ 100 \\ 75 \\ 150 \\ 25 \\ 60 \\ 40 \\ 80 \\ 20 \\ 65 \\ 45 \\ 25 \\ 115 \\ 95 \\ 75 \\ 50 \\ 140 \\ 185 \\ 120 \\ 205 \\ 190 \\ 135 \\ 170 \\ 155 \\ 20 \end{pmatrix} \tag{12.29}$$

The first ten inequalities make sure that the moving hyperplanes do not pass any of the corners of the fixed hyperplanes (compare with Figure 12.6). The remaining inequalities state the relations between the moving hyperplanes (for example, the last inequality tells us that the cooler should be turned off at a lower temperature than when it is turned on).

In addition to the topological requirement, the system also has to satisfy the properties to verify. This is achieved if δ_{bh}, δ_{heat}, δ_{cool}, δ_{reac}, and γ satisfy the following set of inequalities, obtained from (12.16):

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 25 & 0 & 0 & 1 & 0 & 0.0012 & 0 & -0.001 & 0 \\ 25 & 0 & 0 & 1 & 0 & 0.0012 & 0 & 0 & -0.001 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.0002 & 0 \\ 25 & 0 & 1 & 1 & 0 & 0.0012 & 0 & -0.001 & 0 \\ 25 & 0 & 1 & 1 & 0 & 0.0012 & 0 & 0 & -0.001 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -0.0002 \\ 0 & 0 & 0 & -1 & 0 & 0 & -0.0012 & 0.001 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -0.0012 & 0 & 0.001 \\ 0 & 0 & -1 & -1 & 0 & 0 & -0.0012 & 0.001 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & -0.0012 & 0 & 0.001 \end{pmatrix} \begin{pmatrix} \delta_{\mathrm{bh}} \\ \delta_{\mathrm{heat}} \\ \delta_{\mathrm{cool}} \\ \delta_{\mathrm{reac}} \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \end{pmatrix} \preccurlyeq \begin{pmatrix} 9.8380 \\ 0.0294 \\ 0.0225 \\ 245.7977 \\ 245.8179 \\ 0.0200 \\ 245.7536 \\ 245.7738 \\ 0.0286 \\ 0.2138 \\ 0.1935 \\ 0.2579 \\ 0.2377 \end{pmatrix} \tag{12.30}$$

Together with (12.29), (12.30) now describes the region, in which the values of δ_{bh}, δ_{heat}, δ_{cool}, δ_{reac}, and γ are allowed to vary. We can also note that setting $\delta_{\mathrm{ah}} = \delta_{\mathrm{T}_1} = \delta_{\mathrm{T}_2} = 0$ in (12.28) and $\gamma = 0$ in (12.30) both give the same restrictions for the remaining uncertainties δ_{bh}, δ_{heat}, δ_{cool}, and δ_{reac}.

From (12.30), one can go further and, e.g., find the values of γ that allow as large uncertainties as possible in the b parameters. This gives in a sense the maximally robust controller satisfying the verification, and is calculated similarly to the computation of the *Chebychev center* of a polyhedron (the center of the largest sphere inscribed in a polyhedron; see [19]). Here we would like to find the largest disc, parallel to the γ coordinate axes and with its center somewhere along the subspace $\delta = 0$, inscribed in the polyhedron defined by (12.29) and (12.30). Let us represent the disc as

$$\mathcal{D} = \{\begin{pmatrix}0\\ \gamma\end{pmatrix} + \begin{pmatrix}\delta\\ 0\end{pmatrix} \mid \|\delta\| \le R\} \tag{12.31}$$

where the vector $\left(\begin{smallmatrix}0\\ \gamma\end{smallmatrix}\right)$ is the center of the disc and R is the radius. We would like to maximize R subject to the constraint that $\mathcal{D}$ satisfies (12.29) and (12.30). Let us stack (12.29) and (12.30) on top of each other, and denote the resulting matrices according to

$$F\begin{pmatrix}\delta\\ \gamma\end{pmatrix} \preccurlyeq g \tag{12.32}$$

For each row i in (12.32), the constraint that $\mathcal{D}$ should satisfy $F_i\left(\begin{smallmatrix}\delta\\ \gamma\end{smallmatrix}\right) \le g_i$ can be expressed

$$F_i\begin{pmatrix}0\\ \gamma\end{pmatrix} + \sup_{\|\delta\|\le R} F_i\begin{pmatrix}\delta\\ 0\end{pmatrix} \le g_i$$

But since

$$\sup_{\|\delta\|\le R} F_i\begin{pmatrix}\delta\\ 0\end{pmatrix} = \sup_{\|\delta\|\le R} F_i\begin{pmatrix}I\\ 0\end{pmatrix}\delta = R\left\|F_i\begin{pmatrix}I\\ 0\end{pmatrix}\right\|$$

(where I is an identity matrix with the same number of rows as δ), we get

$$F_i\begin{pmatrix}0\\ \gamma\end{pmatrix} + R\left\|F_i\begin{pmatrix}I\\ 0\end{pmatrix}\right\| \le g_i$$

This is a linear inequality in γ and R, and the desired value of γ can be computed from the LP

$$\begin{aligned} \max_{\gamma,R} \quad & R \\ \text{subj. to} \quad & F_i\begin{pmatrix}0\\ \gamma\end{pmatrix} + R\left\|F_i\begin{pmatrix}I\\ 0\end{pmatrix}\right\| \le g_i, \quad i = 1,\dots,M_F \end{aligned} \tag{12.33}$$

where M_F is the number of rows of F.

12.9 Conclusions

We have suggested an approach to investigate how sensitive approximating automata for piecewise affine systems might be to changes in the underlying subsystems, to noise, and to translations of the switching surfaces. Section 12.4 provided the sets of system matrices that satisfy certain requirements on the behavior of the system. As pointed out, these can either be seen as giving a measure of how robust the approximating automata are to uncertainties in the system, or as giving limits for how much the system can be changed, e.g., in a control design process, without altering the overall behavior described by the approximating automata. The methods can also be extended to switched systems, which was mentioned in Section 12.7.2.

It would be natural to combine these requirements with other objectives. One example of this was given in Section 12.8.4. To give another example, when using the state feedback parameterization described in Section 12.5, one would probably want to find $L(v)$ that are optimal in a certain respect. Since the solution sets of the two first problems in Section 12.4.2 and Section 12.4.4 are convex, we can form all sorts of convex optimization problems, which can be solved very efficiently once we know the direct representations of the polyhedra (see for example [19]).

Part IV

Appendices

A

Mathematical Preliminaries

In this appendix, some general concepts and results needed in previous chapters are introduced. These include some material about convex sets and polyhedra, system identification, mixed-integer programming, and some combinatorial aspects in classification. References for further reading are given in each section.

A.1 Explicit Solution of a System of Linear Equations

In Part I, we need to be able to solve a certain form of systems of linear equations. This is accomplished using the following lemma.

Lemma A.1
Let $\rho > 0$, $x, z \in \mathbb{R}^n$ where $n \geq 2$, and let x have at least two elements that are not equal. Then

$$\begin{pmatrix} -\rho z z^T - I & 1_n & x \\ 1_n^T & 0 & 0 \\ x^T & 0 & 0 \end{pmatrix} \begin{pmatrix} w \\ \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{A.1}$$

(with $w \in \mathbb{R}^n$ and $\mu_1, \mu_2 \in \mathbb{R}$) is nonsingular, and has the solution

$$w_i = \frac{1}{\zeta}\left(\left(\gamma\sum_{k=1}^n x_k^2 - \left(\sum_{k=1}^n x_k z_k\right)^2\right) + \left(\sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k - \gamma\sum_{k=1}^n x_k\right) x_i \right. \tag{A.2}$$
$$\left. + \left(\sum_{k=1}^n x_k \sum_{k=1}^n x_k z_k - \sum_{k=1}^n x_k^2 \sum_{k=1}^n z_k\right) z_i\right)$$

$$\mu = \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix} = \frac{1}{\zeta}\begin{pmatrix} \gamma\sum_{k=1}^n x_k^2 - \left(\sum_{k=1}^n x_k z_k\right)^2 \\ -\gamma\sum_{k=1}^n x_k + \sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k \end{pmatrix} \tag{A.3}$$

where

$$\gamma = \frac{1}{\rho} + z^T z$$
$$\zeta = \frac{1}{6}\sum_{i=1}^n\sum_{j=1}^n\sum_{k=1}^n \left(x_i(z_k - z_j) + x_j(z_i - z_k) + x_k(z_j - z_i)\right)^2 + \frac{1}{2\rho}\sum_{i=1}^n\sum_{k=1}^n (x_i - x_k)^2$$

Remark A.1 *Note that both γ and ζ are strictly positive.*

Proof Let

$$G = -\rho z z^T - I, \qquad A = \begin{pmatrix}1_n^T\\ x^T\end{pmatrix}$$

First we need to show that (A.1) is nonsingular. It is easy to show (see, e.g., [161, Theorem 2.2]) that

$$\det\begin{pmatrix}G & A^T\\ A & 0\end{pmatrix} = \det(G)\det(-AG^{-1}A^T)$$

Since A has full rank and G is negative definite, it follows that $AG^{-1}A^T$ is negative definite, and hence the system (A.1) is nonsingular.

Now,

$$\begin{pmatrix}w\\ \mu\end{pmatrix} = \begin{pmatrix}G & A^T\\ A & 0\end{pmatrix}^{-1}\begin{pmatrix}0\\ \begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}G^{-1}A^T(AG^{-1}A^T)^{-1}\\ -(AG^{-1}A^T)^{-1}\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} \tag{A.4}$$

according to well-known matrix inversion formulas (see, e.g., [161, Theorem 2.4]). Furthermore,

$$G^{-1} = -I + \frac{1}{\gamma} z z^T$$

$$\begin{aligned}
G^{-1}A^T &= -\begin{pmatrix} 1_n & x \end{pmatrix} + \frac{1}{\gamma} z \begin{pmatrix} \sum_{k=1}^n z_k & \sum_{k=1}^n x_k z_k \end{pmatrix} \\
AG^{-1}A^T &= \frac{1}{\gamma} \\
&\cdot \begin{pmatrix} -\gamma n + \left(\sum_{k=1}^n z_k\right)^2 & -\gamma \sum_{k=1}^n x_k + \sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k \\ -\gamma \sum_{k=1}^n x_k + \sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k & -\gamma \sum_{k=1}^n x_k^2 + \left(\sum_{k=1}^n x_k z_k\right)^2 \end{pmatrix} \\
(AG^{-1}A^T)^{-1} &= \frac{1}{\zeta} \\
&\cdot \begin{pmatrix} -\gamma \sum_{k=1}^n x_k^2 + \left(\sum_{k=1}^n x_k z_k\right)^2 & \gamma \sum_{k=1}^n x_k - \sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k \\ \gamma \sum_{k=1}^n x_k - \sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k & -\gamma n + \left(\sum_{k=1}^n z_k\right)^2 \end{pmatrix}
\end{aligned}$$

The first equality can easily be checked. The second and third follow directly from the first and the definition of A. To see that the last equality holds, we need to show that $\det(AG^{-1}A^T) = \zeta/\gamma$. Then the equality holds by the well-known inversion formula for 2-by-2 matrices. But

$$\begin{aligned}
\zeta &= \frac{1}{6} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \left(x_i(z_k - z_j) + x_j(z_i - z_k) + x_k(z_j - z_i)\right)^2 + \frac{1}{2\rho} \sum_{i=1}^n \sum_{k=1}^n (x_i - x_k)^2 \\
&= \frac{1}{6} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \left(x_i z_k - x_i z_j + x_j z_i - x_j z_k + x_k z_j - x_k z_i\right)^2 \\
&\quad + \frac{1}{2\rho} \sum_{i=1}^n \sum_{k=1}^n (x_i^2 - 2x_i x_k + x_k^2) \\
&= \frac{1}{6} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n (6x_k^2 z_i^2 - 6x_i x_k z_j^2 - 6x_i x_k z_i z_k - 6x_k^2 z_i z_j + 12 x_i x_k z_j z_k) \\
&\quad + \frac{1}{2\rho} \sum_{i=1}^n \sum_{k=1}^n (2x_k^2 - 2x_i x_k) \\
&= n \sum_{k=1}^n x_k^2 \sum_{k=1}^n z_k^2 - \left(\sum_{k=1}^n x_k\right)^2 \sum_{k=1}^n z_k^2 - n \left(\sum_{k=1}^n x_k z_k\right)^2 - \sum_{k=1}^n x_k^2 \left(\sum_{k=1}^n z_k\right)^2 \\
&\quad + 2 \sum_{k=1}^n x_k \sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k + \frac{1}{\rho} \left(n \sum_{k=1}^n x_k^2 - \left(\sum_{k=1}^n x_k\right)^2 \right) \\
&= \left(\frac{1}{\rho} + \sum_{k=1}^n z_k^2\right) \left(n \sum_{k=1}^n x_k^2 - \left(\sum_{k=1}^n x_k\right)^2 \right) - n \left(\sum_{k=1}^n x_k z_k\right)^2 - \sum_{k=1}^n x_k^2 \left(\sum_{k=1}^n z_k\right)^2 \\
&\quad + 2 \sum_{k=1}^n x_k \sum_{k=1}^n z_k \sum_{k=1}^n x_k z_k
\end{aligned}$$

$$
\begin{aligned}
&= \gamma \frac{1}{\gamma^2} \left(\gamma^2 n \sum_{k=1}^{n} x_k^2 - \gamma n \left(\sum_{k=1}^{n} x_k z_k \right)^2 - \gamma \left(\sum_{k=1}^{n} z_k \right)^2 \sum_{k=1}^{n} x_k^2 \right. \\
&\quad + \left(\sum_{k=1}^{n} z_k \right)^2 \left(\sum_{k=1}^{n} x_k z_k \right)^2 - \gamma^2 \left(\sum_{k=1}^{n} x_k \right)^2 + 2\gamma \sum_{k=1}^{n} x_k \sum_{k=1}^{n} z_k \sum_{k=1}^{n} x_k z_k \\
&\quad \left. - \left(\sum_{k=1}^{n} z_k \right)^2 \left(\sum_{k=1}^{n} x_k z_k \right)^2 \right) \\
&= \gamma \det(AG^{-1}A^T)
\end{aligned}
$$

where, in the third and fourth steps, the indices of several terms are switched. This shows the expression for $(AG^{-1}A^T)^{-1}$.

From (A.4), we can now see that μ is given by the first column of $(AG^{-1}A^T)^{-1}$, multiplied by -1, which gives the desired expression (A.3). For w, (A.4) gives that multiplying the expression for $G^{-1}A^T$ by the first column of $(AG^{-1}A^T)^{-1}$ yields the desired expression, which is just a vector version of (A.2). $\square$

A.2 Lipschitz Conditions

When studying multivariate functions in Chapter 4, some assumptions are made about the pth derivative satisfying a Lipschitz condition defined in (4.1). Under this assumption, one can show a useful inequality, given by Lemma A.3. However, to be able to carry out the proof, the following lemma is needed. We use the notation $f^{(p)}_{i_1 \ldots i_p}$ for

$$
f^{(p)}_{i_1 \ldots i_p} = \frac{\partial^p f}{\partial x_{i_1} \ldots \partial x_{i_p}} \tag{A.5}
$$

Lemma A.2
Let $f : \mathbb{R}^n \to \mathbb{R}$ be p times differentiable. Then

$$
\begin{aligned}
&f(x + t_0 h) - f(x) - t_0 \sum_{i_1=1}^{n} f'_{i_1}(x) h_{i_1} - \ldots - \frac{t_0^p}{p!} \sum_{i_1=1}^{n} \cdots \sum_{i_p=1}^{n} f^{(p)}_{i_1 \ldots i_p}(x) h_{i_1} \ldots h_{i_p} \\
&= \int_0^{t_0} \int_0^{t_1} \cdots \int_0^{t_{p-1}} \\
&\qquad \sum_{i_1=1}^{n} \cdots \sum_{i_p=1}^{n} \left(f^{(p)}_{i_1 \ldots i_p}(x + t_p h) - f^{(p)}_{i_1 \ldots i_p}(x) \right) h_{i_1} \ldots h_{i_p} dt_p \ldots dt_2 dt_1
\end{aligned}
$$

Proof The proof is carried out using induction. For $p = 1$, the lemma is trivial. Now suppose that the lemma is shown for $p = q - 1$, and consider the case $p = q$.

Then

$$
\begin{aligned}
&f(x+t_0h)-f(x)-t_0\sum_{i_1=1}^n f'_{i_1}(x)h_{i_1}-\ldots-\frac{t_0^q}{q!}\sum_{i_1=1}^n\cdots\sum_{i_q=1}^n f^{(q)}_{i_1\ldots i_q}(x)h_{i_1}\ldots h_{i_q}\\
&=\int_0^{t_0}\sum_{i_1=1}^n f'_{i_1}(x+t_1h)h_{i_1}-\sum_{i_1=1}^n f'_{i_1}(x)h_{i_1}-\ldots\\
&\qquad-\frac{t_1^{q-1}}{(q-1)!}\sum_{i_1=1}^n\cdots\sum_{i_q=1}^n f^{(q)}_{i_1\ldots i_q}(x)h_{i_1}\ldots h_{i_q}\,dt_1\\
&=\int_0^{t_0}\sum_{i_1=1}^n h_{i_1}\Bigg(f'_{i_1}(x+t_1h)-f'_{i_1}(x)-\ldots\\
&\qquad-\frac{t_1^{q-1}}{(q-1)!}\sum_{i_2=1}^n\cdots\sum_{i_q=1}^n f^{(q)}_{i_1\ldots i_q}(x)h_{i_2}\ldots h_{i_q}\Bigg)dt_1\\
&=\int_0^{t_0}\sum_{i_1=1}^n h_{i_1}\Bigg(\int_0^{t_1}\cdots\int_0^{t_{q-1}}\\
&\qquad\sum_{i_2=1}^n\cdots\sum_{i_q=1}^n\left(f^{(q)}_{i_1\ldots i_q}(x+t_qh)-f^{(q)}_{i_1\ldots i_q}(x)\right)h_{i_2}\ldots h_{i_q}dt_q\ldots dt_2\Bigg)dt_1\\
&=\int_0^{t_0}\int_0^{t_1}\cdots\int_0^{t_{q-1}}\\
&\qquad\sum_{i_1=1}^n\cdots\sum_{i_q=1}^n\left(f^{(q)}_{i_1\ldots i_q}(x+t_qh)-f^{(q)}_{i_1\ldots i_q}(x)\right)h_{i_1}\ldots h_{i_q}dt_q\ldots dt_2dt_1
\end{aligned}
$$

where the induction hypothesis was used in the third step, with f replaced by f'_{i_1}. The lemma is shown. □

The following lemma is a generalization of [40, Lemmas 4.1.12, 4.1.14].

Lemma A.3
Let $f:\mathbb{R}^n\to\mathbb{R}$ be p times differentiable and satisfy the Lipschitz condition

$$
\max_{\|\xi\|=1}\left|\sum_{i_1=1}^n\cdots\sum_{i_p=1}^n\left(f^{(p)}_{i_1\ldots i_p}(x+h)-f^{(p)}_{i_1\ldots i_p}(x)\right)\xi_{i_1}\ldots\xi_{i_p}\right|\le L\|h\| \tag{A.6}
$$

for all $h\in\mathbb{R}^n$. Then

$$
\begin{aligned}
&\left|f(x+h)-f(x)-\sum_{i_1=1}^n f'_{i_1}(x)h_{i_1}-\ldots-\frac{1}{p!}\sum_{i_1=1}^n\cdots\sum_{i_p=1}^n f^{(p)}_{i_1\ldots i_p}(x)h_{i_1}\ldots h_{i_p}\right|\\
&\qquad\le\frac{L}{(p+1)!}\|h\|^{p+1}
\end{aligned} \tag{A.7}
$$

Remark A.2 *Note that for $p = 1$, the Lipschitz condition* (A.6) *reduces to*

$$\|\nabla f(x+h) - \nabla f(x)\| \leq L\|h\|$$

and (A.7) *becomes*

$$\left|f(x+h) - f(x) - \nabla^T f(x)h\right| \leq \frac{L}{2}\|h\|^2$$

For $p = 2$, (A.6) *is equivalent to*

$$\|\nabla^2 f(x+h) - \nabla^2 f(x)\| \leq L\|h\|$$

where $\nabla^2 f$ is the Hessian of f, and where we use the usual induced 2-norm for matrices. Also, (A.7) *reduces to*

$$\left|f(x+h) - f(x) - \nabla^T f(x)h - \frac{1}{2}h^T\nabla^2 f(x)h\right| \leq \frac{L}{6}\|h\|^3$$

Proof For $h = 0$, equality holds trivially. Now consider $h \neq 0$, and let $\bar{h} = h/\|h\|$. By using Lemma A.2 we get

$$\begin{aligned}
&\left|f(x+h) - f(x) - \sum_{i_1=1}^{n} f'_{i_1}(x)h_{i_1} - \ldots - \frac{1}{p!}\sum_{i_1=1}^{n}\cdots\sum_{i_p=1}^{n} f^{(p)}_{i_1\ldots i_p}(x)h_{i_1}\ldots h_{i_p}\right| \\
&= \left|\int_0^1\int_0^{t_1}\cdots\int_0^{t_{p-1}}\right. \\
&\qquad \left.\sum_{i_1=1}^{n}\cdots\sum_{i_p=1}^{n}\left(f^{(p)}_{i_1\ldots i_p}(x+t_p h) - f^{(p)}_{i_1\ldots i_p}(x)\right)h_{i_1}\ldots h_{i_p}\,dt_p\ldots dt_2 dt_1\right| \\
&\leq \int_0^1\int_0^{t_1}\cdots\int_0^{t_{p-1}} \\
&\qquad \left|\sum_{i_1=1}^{n}\cdots\sum_{i_p=1}^{n}\left(f^{(p)}_{i_1\ldots i_p}(x+t_p h) - f^{(p)}_{i_1\ldots i_p}(x)\right)h_{i_1}\ldots h_{i_p}\right| dt_p\ldots dt_2 dt_1 \\
&= \|h\|^p\int_0^1\int_0^{t_1}\cdots\int_0^{t_{p-1}} \\
&\qquad \left|\sum_{i_1=1}^{n}\cdots\sum_{i_p=1}^{n}\left(f^{(p)}_{i_1\ldots i_p}(x+t_p h) - f^{(p)}_{i_1\ldots i_p}(x)\right)\bar{h}_{i_1}\ldots \bar{h}_{i_p}\right| dt_p\ldots dt_2 dt_1 \\
&\leq \|h\|^p\int_0^1\int_0^{t_1}\cdots\int_0^{t_{p-1}} \\
&\qquad \max_{\|\xi\|=1}\left|\sum_{i_1=1}^{n}\cdots\sum_{i_p=1}^{n}\left(f^{(p)}_{i_1\ldots i_p}(x+t_p h) - f^{(p)}_{i_1\ldots i_p}(x)\right)\xi_{i_1}\ldots \xi_{i_p}\right| dt_p\ldots dt_2 dt_1
\end{aligned}$$

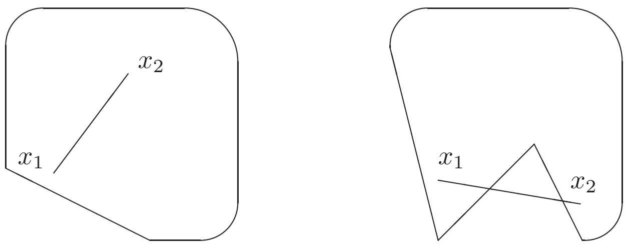


Figure A.1: A convex set (left) and a nonconvex set (right).

$$
\begin{aligned}
&\le \|h\|^p \int_0^1 \int_0^{t_1} \dots \int_0^{t_{p-1}} L\|t_p h\| dt_p \dots dt_2 dt_1 \\
&= L\|h\|^{p+1} \int_0^1 \int_0^{t_1} \dots \int_0^{t_{p-1}} t_p dt_p \dots dt_2 dt_1 \\
&= \frac{L}{(p+1)!}\|h\|^{p+1}
\end{aligned}
$$

and the lemma is proven. □

A.3 Convex Sets and Polyhedra

For the piecewise affine system models used in this thesis, some concepts concerning convex sets and polyhedra will be needed. These are presented in this section.

A line passing through two different points $x_1, x_2 \in \mathbb{R}^n$, $x_1 \neq x_2$, can mathematically be described as the set

$$\{\lambda x_1 + (1-\lambda)x_2 \mid \lambda \in \mathbb{R}\}$$

The (closed) *line segment* between x_1 and x_2 is the part of the line that lies between the points (including x_1 and x_2). This corresponds to

$$\{\lambda x_1 + (1-\lambda)x_2 \mid 0 \le \lambda \le 1\}$$

If $x_1 = x_2$, the line segment consists of the single point x_1.

A set $P \subseteq \mathbb{R}^n$ is *convex* if the line segment between any two points in P lies entirely in P (see Figure A.1). This condition can be written as

$$\forall x_1, x_2 \in P, \lambda \in [0,1] : \lambda x_1 + (1-\lambda)x_2 \in P$$

For any set S, the smallest convex set P that contains S (i.e., $S \subseteq P$) is called the *convex hull* of S.

We can extend the concept of line segment between two points to consider a weighted mean of a number of points $x_1, \dots, x_r$:

$$x_{co} = \sum_{i=1}^{r} \lambda_i x_i, \quad \lambda_i \ge 0, \ \sum_{i=1}^{r} \lambda_i = 1 \tag{A.8}$$

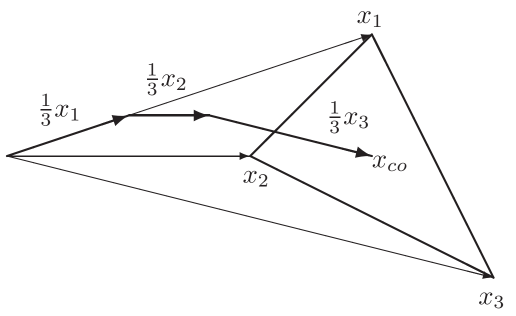


Figure A.2: An example of a convex combination. Any point in the convex hull of $\{x_1, x_2, x_3\}$ can be written as a weighted mean (in the form (A.8)) of x_1, x_2 and x_3.

This kind of weighted mean is called a *convex combination* of $x_1, \ldots, x_r$ (see Figure A.2). It turns out that the convex hull of a set S is the set of all convex combinations of points in S (or alternatively, this could be used as the definition of convex hull).

An important example of convex sets are the *polyhedra*. Polyhedra are sets defined by linear inequalities, and can be written in the form

$$\{x \in \mathbb{R}^n \mid Cx \preccurlyeq d\} \tag{A.9}$$

where $C \in \mathbb{R}^{M \times n}$, $d \in \mathbb{R}^M$, and $\preccurlyeq$ denotes componentwise inequality. A bounded polyhedron is called a *polytope*.

Example A.1 (Polytope) *The set $P \subseteq \mathbb{R}^2$ of points satisfying*

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_1 - x_2 &\leq 0 \\ -x_1 &\leq 1 \end{aligned}$$

is a polyhedron (see Figure A.3). Furthermore, since P is bounded, it is a polytope.

Polytopes can also be represented in an alternative way, namely as the set of convex combinations of the corners. This type of representation is called *direct representation.*

Example A.2 (Direct representation) *The polytope in Example A.1 can also be represented as*

$$\left\{ \lambda_1 \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} + \lambda_2 \begin{pmatrix} -1 \\ 2 \end{pmatrix} + \lambda_3 \begin{pmatrix} -1 \\ -1 \end{pmatrix} \;\middle|\; \lambda_i \geq 0, \sum_{i=1}^{3} \lambda_i = 1 \right\}$$

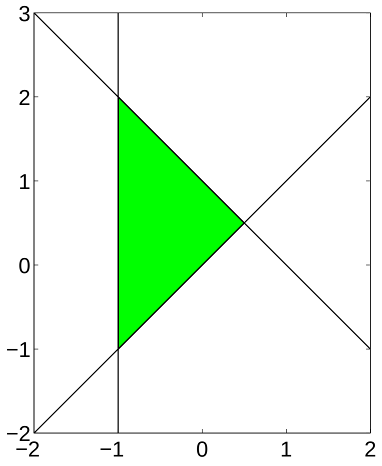


Figure A.3: The polytope in Example A.1.

The direct representation for unbounded polyhedra is a bit trickier. Apart from the corners, we must also include in the representation some vectors $x_{r+1}, \ldots, x_{r+h}$, which are parallel to the unbounded edges of the polyhedron:

$$\{\sum_{i=1}^{r+h} \lambda_i x_i \mid \lambda_i \geq 0, \sum_{i=1}^{r} \lambda_i = 1\} \tag{A.10}$$

Note here that $\lambda_{r+1}, \ldots, \lambda_{r+h}$ are not included in the set of λ_i that should sum up to one; they can be arbitrarily large.

Example A.3 (Unbounded polyhedron) *If we would remove the constraint $-x_1 \leq 1$ from the polytope in Example A.1, we would get an unbounded polyhedron defined by*

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_1 - x_2 &\leq 0 \end{aligned}$$

This polyhedron can be represented as

$$\left\{ 1 \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} + \lambda_2 \begin{pmatrix} -1 \\ -1 \end{pmatrix} + \lambda_3 \begin{pmatrix} -1 \\ 1 \end{pmatrix} \;\middle|\; \lambda_i \geq 0 \right\}$$

Some special polyhedra (e.g., a hyperplane) do not have any corners. In that case, a “dummy” corner must be introduced somewhere in the polyhedron to allow for a direct representation.

Unfortunately, going from direct representation (A.10) to the form (A.9) is in general computationally quite complex.

In Chapter 12, we are going to use *open polyhedra*. By this will be meant a set, whose direct representation is

$$\{\sum_{i=1}^{r+h} \lambda_i x_i \mid \lambda_i > 0,\ \sum_{i=1}^{r} \lambda_i = 1\} \tag{A.11}$$

i.e., where all λ_i are strictly positive. Note that this does not necessarily mean that the set is open. However, relative to the smallest affine subspace containing the open polyhedron, it is an open set.

Example A.4 (Open polyhedron) *The (open) line segment*

$$\left\{\lambda_1 \begin{pmatrix}1\\2\end{pmatrix} + \lambda_2 \begin{pmatrix}2\\3\end{pmatrix} \;\middle|\; \lambda_i > 0, \lambda_1 + \lambda_2 = 1\right\}$$

is an open polyhedron (in fact an open polytope), but not an open subset of $\mathbb{R}^2$. *However, seen as a subset of the line through the points* $\left(\begin{smallmatrix}1\\2\end{smallmatrix}\right)$ *and* $\left(\begin{smallmatrix}2\\3\end{smallmatrix}\right)$, *it is open.*

An excellent introduction to convex sets can be found in [19].

A.4 MILP/MIQP

In Chapter 11, some different piecewise affine system identification problems are formulated as *Mixed-Integer Linear Programs (MILP)* and *Mixed-Integer Quadratic Programs (MIQP)*. These are two types of optimization problems that involve both discrete and continuous variables. In general, an MILP problem has the following structure:

$$\begin{aligned} \min_{x,\delta} \quad & p^T \begin{pmatrix}x\\ \delta\end{pmatrix} \\ \text{subj. to} \quad & C \begin{pmatrix}x\\ \delta\end{pmatrix} \le d \\ & x \in \mathbb{R}^n,\ \delta \in \{0,1\}^m \end{aligned} \tag{A.12}$$

The MIQP problem is almost identical, except that the objective function is quadratic:

$$\begin{aligned} \min_{x,\delta} \quad & \begin{pmatrix}x^T & \delta^T\end{pmatrix} Q \begin{pmatrix}x\\ \delta\end{pmatrix} + p^T \begin{pmatrix}x\\ \delta\end{pmatrix} \\ \text{subj. to} \quad & C \begin{pmatrix}x\\ \delta\end{pmatrix} \le d \\ & x \in \mathbb{R}^n,\ \delta \in \{0,1\}^m \end{aligned} \tag{A.13}$$

We assume that Q is positive definite or semidefinite. If it is only positive semidefinite, one could use regularization techniques as in Section 8.3 to make it positive definite.

As we can see, the only difference compared to an ordinary LP or QP (Linear or Quadratic Program) is that the δ variables are discrete. However, although LP and QP problems can be solved efficiently using standard algorithms, it has been shown that solving a general MILP/MIQP problem is $\mathcal{NP}$-hard [49, 151]. This means that there is no known algorithm solving the MILP/MIQP problems with polynomial worst-case complexity, and that if such an algorithm is found, several other hard problems would also be solvable in polynomial time. It is quite easy to find an algorithm, for which we in the worst case have to test all combinations of δ values, so the worst-case complexity is at most 2^m times the complexity of an LP/QP problem in n variables. However, there are algorithms that work better for many practical cases. Some solvers for the MILP/MIQP problems can be found in, e.g., [11, 36, 70, 139].

A.4.1 A Branch-and-Bound Algorithm

Let us take a closer look at (A.12) and (A.13). If we would replace the requirement $\delta \in \{0,1\}^m$ by $0 \le \delta \le 1$, the problems would become ordinary LP or QP problems. Similarly, if we fix some of the elements of δ to either 0 or 1 and relax the other elements, we would also get LP/QP problems. These combined relaxations and restrictions can be used to form a branch-and-bound algorithm, which will be illustrated by an example:

Example A.5 (Branch-and-bound, depth first) *We would like to solve the problem*

$$\begin{aligned} \min_{x,\delta} \quad & J = x - 2\delta_1 - \delta_2 \\ \text{subj. to} \quad & x + 5\delta_1 + 6\delta_2 \le 9 \\ & 0 \le x \le 1 \\ & \delta_1, \delta_2 \in \{0,1\} \end{aligned} \tag{A.14}$$

Figure A.4 shows the feasible region for this problem. Let us start by solving the relaxed problem we get by replacing $\delta_1, \delta_2 \in \{0,1\}$ by $0 \le \delta_1 \le 1$, $0 \le \delta_2 \le 1$. This is an LP problem with the solution

$$\begin{cases} J^* = -8/3 \\ x = 0 \\ \delta_1 = 1 \\ \delta_2 = 2/3 \end{cases}$$

In Figure A.4, the feasible region for the relaxed problem is the three-dimensional region bounded by the cube and the plane $x + 5\delta_1 + 6\delta_2 = 9$. Let us now fix one

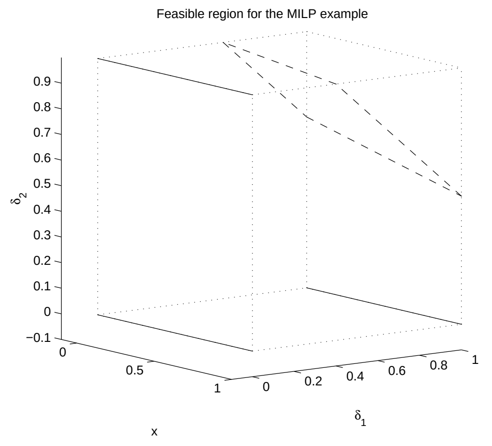


Figure A.4: The feasible region for Example A.5. The solid lines define the feasible region, the dotted lines show the bounds $0 \leq x \leq 1$, $0 \leq \delta \leq 1$, and the dashed lines show the hyperplane defined by $x + 5\delta_1 + 6\delta_2 = 9$.

of the δ variables and solve the new, more restricted LP problem we get. From looking at the optimal solution to the relaxed problem, it seems natural to start by fixing δ_1 to 1. See Figure A.5, where the relaxed problem corresponds to the root node, and the new problem corresponds to its right child. Note that, in general, since we restrict the domain over which we are minimizing J, we can never get a lower value of J than the optimum we get for the relaxed case. In this special case, since the optimal value for δ_1 was 1 already in the relaxed case, we already know that $J^* = -8/3$ and $\delta_2 = 2/3$ is the optimal solution when δ_1 is set to 1. Hence we can continue directly by fixing δ_2, e.g., to 1. The problem has now become

$$\begin{aligned} \min_{x,\delta} \quad & J = x - 2 - 1 \\ \text{subj. to} \quad & x + 5 + 6 \leq 9 \\ & 0 \leq x \leq 1 \end{aligned}$$

This problem is obviously infeasible, so instead we let $\delta_2 = 0$ and get

$$\begin{aligned} \min_{x,\delta} \quad & J = x - 2 \\ \text{subj. to} \quad & x + 5 \leq 9 \\ & 0 \leq x \leq 1 \end{aligned}$$

which has the optimal solution

$$\begin{cases} J^* = -2 \\ x = 0 \\ (\delta_1 = 1,\ \delta_2 = 0) \end{cases}$$

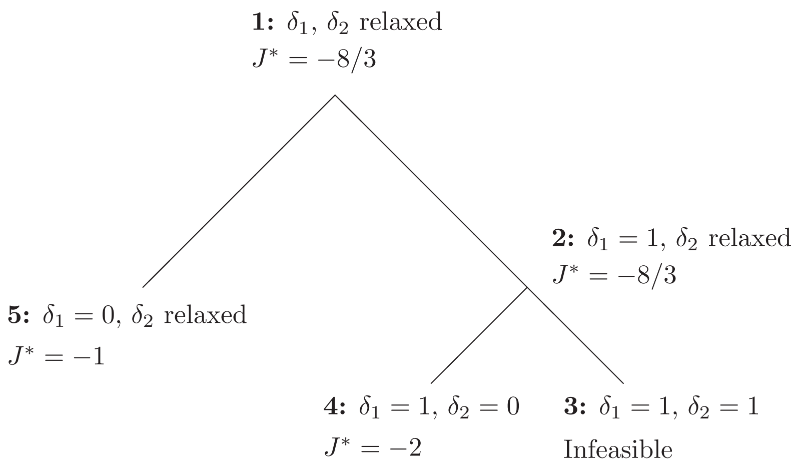

1: δ_1, δ_2 relaxed
$J^* = -8/3$
2: $\delta_1 = 1$, δ_2 relaxed
$J^* = -8/3$
5: $\delta_1 = 0$, δ_2 relaxed
$J^* = -1$
4: $\delta_1 = 1$, $\delta_2 = 0$
$J^* = -2$
3: $\delta_1 = 1$, $\delta_2 = 1$
Infeasible

Figure A.5: The search tree for Example A.5.

This optimum is our first feasible candidate so far for an optimum to the original problem. However, we have not yet excluded all possible combinations of values for δ. Therefore, we now let $\delta_1 = 0$, while δ_2 is relaxed. The problem to solve is

$$\begin{aligned} \min_{x,\delta} \quad & J = x - \delta_2 \\ \text{subj. to} \quad & x + 6\delta_2 \leq 9 \\ & 0 \leq x \leq 1 \\ & 0 \leq \delta_2 \leq 1 \end{aligned}$$

with the optimal solution

$$\begin{cases} J^* = -1 \\ x = 0 \\ \delta_2 = 1 \\ (\delta_1 = 0) \end{cases}$$

Apparently, this optimal point is worse than the feasible candidate point we got previously. Since we cannot get better values of J by restricting the domain further (i.e., fixing δ_2), we can conclude that our candidate point really is the optimum for the original problem, without having to test the remaining possibilities.

The procedure used in the previous example is summarized in the following algorithm, which is a version of a standard algorithm that can be found, e.g., in [160].

Algorithm A.1 (Branch-and-bound, depth first)

Given: *A problem like* (A.12) *or* (A.13).
Initialization: *Set* $LB = -\infty$ *and* $UB = \infty$. *Relax the problem by allowing all elements in* δ *to take any value in the interval* $[0, 1]$. *Start building on a tree structure by creating a root node. (Each level in the tree structure will correspond to one more element in* δ *being fixed; cf. Figure A.5.)*

1. *Solve the current LP/QP problem. If it is infeasible, go to 4. If it is feasible, denote the optimal point by* x^*, δ^*, *and the optimal value by* J^*. *If* $\delta^* \in \{0, 1\}^m$, *go to 3. Otherwise, go to 2.*
2. *If* $J^* > UB$, *the optimal solution will not be in this branch of the tree, so go to 4. Else, fix one of the relaxed elements in* δ *to* 0 *or* 1, *according to some strategy. Add one child to the current node in the tree and move to the child node. If all of the currently fixed elements of* δ *have already previously been fixed to their other possible values, and if* $J^* > LB$, *set* $LB = J^*$. *Go to 1.*
3. *If* $J^* < UB$, *set* $UB = J^*$, *and let* $x_{min} = x^*$ *and* $\delta_{min} = \delta^*$. *Go to 4.*
4. *Go up one level in the tree, i.e., relax the* δ *element that was last fixed. If the other possible value of that element has not yet been tested, fix the element to that value. (In other words, if the element was fixed to* 0 *and has not yet been fixed to* 1, *set it to* 1.*) Otherwise, go up one level at a time in the tree until an untested value of a* δ *element is found. Then go to 1. If the root node is reached and no untested value is found, we are done. Then set* $LB = UB$.

When these iterations are done, the optimal value of the objective function equals $LB = UB$. *If* $LB = UB = \infty$, *the problem is infeasible. Otherwise, the optimal point is given by* x_{min} *and* δ_{min}.

There are some remarks to be made. Firstly, the algorithm does not specify how to choose the element to be fixed in step 2. There are several possible strategies to use for this. In Example A.5, the element of δ that was closest to an integer value was chosen. The thought behind this is to quickly reach a good feasible point for the original problem. With its help, entire subtrees can be ruled out as being obviously bad (or at least worse than the best value found so far), which may save a large amount of computations, just as in the example.

During the iterations of Algorithm A.1, the variables UB and LB give an upper and a lower bound on the optimal value. This could be useful, e.g., if the computation is stopped before the search for the optimal value is completed. Then the best suboptimal point so far might be usable, together with the lower bound, that shows how far from the optimum we can be at most.

There are variants of this algorithm. For example, one can also traverse the search tree in a best-first manner, where the nodes with the best optimal values J^* are traversed first, or in a breadth-first manner, or in a combination of the three. There are also several other algorithms besides the branch-and-bound algorithms for solving the MILP/MIQP problems. A good reference is [160].

A.5 Separating Points with Hyperplanes

When investigating the computational complexity of the algorithms in Chapter 11, a question that will arise is: In how many ways can we group N points in $\mathbb{R}^n$ by separating them with M hyperplanes? To be able to answer this question unambiguously, we need to restrict ourselves to only consider a certain kind of point sets. The terminology here partly follows [35].

Definition A.1
A set of points $X \in \mathbb{R}^n$ is in general position, *if for every subset of points $x_1, \dots, x_k \in X$, where $k \le n+1$, the vectors*

$$\begin{pmatrix} 1 \\ x_1 \end{pmatrix}, \dots, \begin{pmatrix} 1 \\ x_k \end{pmatrix}$$

are linearly independent.

This definition is equivalent to specifying that no subset of k points should be contained in an affine subspace of a lower dimension than $\min\{k-1, n\}$. For example, if we consider points in $\mathbb{R}^2$, no points should coincide, and three points should never be on a line. Note that, nonstrictly speaking, for an arbitrarily chosen set of points in $\mathbb{R}^n$, this condition is satisfied with probability one.

Let us begin by determining in how many ways a set of points can be split by one hyperplane. This result will follow as a corollary to the following theorem, which is also found in, e.g., [35].

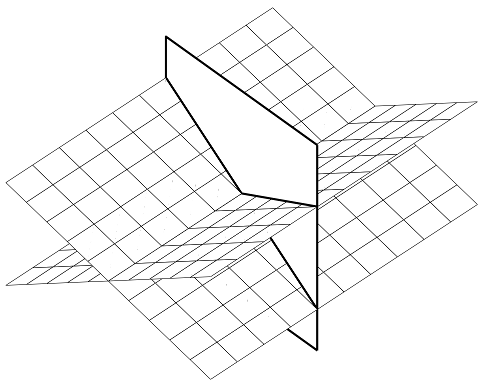
Theorem A.1
Consider N hyperplanes in $\mathbb{R}^n$ going through the origin, such that the normal vectors of any subset consisting of n or fewer hyperplanes are linearly independent. Let $f(n, N)$ be the number of regions into which $\mathbb{R}^n$ is divided by the hyperplanes. Then $f(n, N)$ can be written as a sum of binomial terms

$$f(n, N) = 2 \sum_{k=0}^{n-1} \binom{N-1}{k} \tag{A.15}$$

for $n \ge 1$, $N \ge 1$ (we let $\binom{0}{0} = 1$ and $\binom{N}{k} = 0$ for $k > N$).

Proof We prove the theorem by induction. Clearly, $f(n,1) = 2 = 2\sum_{k=0}^{n-1}\binom{0}{k}$ for $n \ge 1$, since one hyperplane divides the space into two half-spaces. We also have $f(1, N) = 2 = 2\binom{N-1}{0}$ for $N \ge 1$ (a one-dimensional line can only be divided into two half-lines, no matter how many points you put in the origin).

Now suppose that the theorem has been shown for all $\{(n, N) \mid n \le n_0, N \le N_0 - 1\}$, and suppose that we have placed $N_0 - 1$ hyperplanes in $\mathbb{R}^{n_0}$, and are about to place one more (see Figure A.6(a)). The N_0th hyperplane is an $(n_0 - 1)$-dimensional subspace of $\mathbb{R}^{n_0}$, and will be cut by all the other $N_0 - 1$ hyperplanes.


(a) Two planes crossing the origin, and a third to be placed.

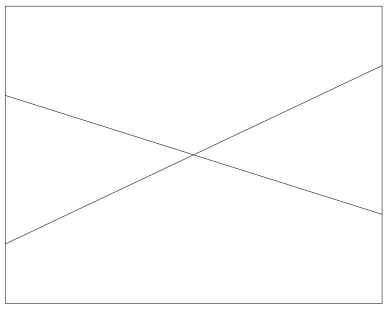

(b) The new hyperplane will be partitioned into several pieces by the first planes. Each of these pieces will divide one of the old regions in Figure A.6(a) into two new regions.

Figure A.6: Illustration of Theorem A.1.

It will therefore be partitioned into $f(n_0 - 1, N_0 - 1)$ pieces (Figure A.6(b)). But each of these pieces will be a border that divides one of the original regions into two new regions. Hence, the original $f(n_0, N_0 - 1)$ regions will increase in number by $f(n_0 - 1, N_0 - 1)$ when the N_0th hyperplane is placed in $\mathbb{R}^{n_0}$. This means that the new number of regions will be

$$f(n_0, N_0) = f(n_0, N_0 - 1) + f(n_0 - 1, N_0 - 1)$$

which means, according to the induction assumption,

$$\begin{aligned} f(n_0, N_0) &= 2\sum_{k=0}^{n_0-1}\binom{N_0-2}{k} + 2\sum_{k=0}^{n_0-2}\binom{N_0-2}{k} \\ &= 2 + 2\sum_{k=0}^{n_0-2}\left[\binom{N_0-2}{k+1} + \binom{N_0-2}{k}\right] \\ &= 2 + 2\sum_{k=0}^{n_0-2}\binom{N_0-1}{k+1} \\ &= 2\sum_{k=0}^{n_0-1}\binom{N_0-1}{k} \end{aligned}$$

and the theorem is proven. □

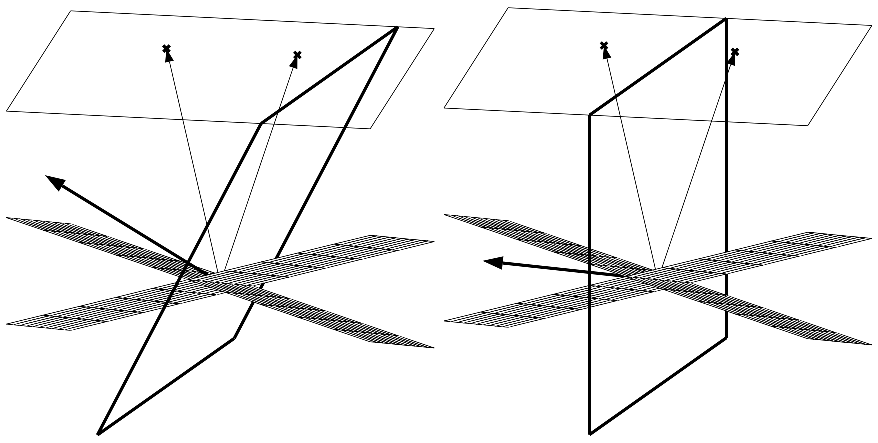


Figure A.7: Illustration of the one-to-one correspondence in Corollary A.1. The upper hyperplane is the affine subspace containing the given point set. Their associated vectors are normal vectors of hyperplanes dividing the space into different regions. The hyperplane drawn with thick lines partitions the point set into two subsets. We will get different partitions depending on in which region the normal vector of the thick-line hyperplane is.

Corollary A.1
A set of N points in general position in $\mathbb{R}^n$ can be partitioned in $f(n+1, N)/2$ ways by a hyperplane.

Proof By considering the vectors

$$\begin{pmatrix} 1 \\ x_1 \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_N \end{pmatrix}$$

we can regard the points as being contained in an n-dimensional affine subspace of $\mathbb{R}^{n+1}$ (see Figure A.7). The vectors are normal vectors to N hyperplanes, which by Theorem A.1 divide $\mathbb{R}^{n+1}$ into $f(n+1, N)$ regions. Now consider an arbitrary hyperplane H going through the origin of $\mathbb{R}^{n+1}$. Its normal vector can lie in one of two opposite regions (depending on how it is directed) among the $f(n+1, N)$ regions. There is a one-to-one correspondence between the set of opposite regions in which the normal vector of this hyperplane lies and how it* partitions the given points. To see this, note that each time the normal vector of H passes a boundary and enters a new region, the hyperplane also passes a point and gives rise to a new partition. This one-to-one correspondence proves the theorem. □

Now when we know in how many ways one hyperplane can divide a set of points, it is easy to extend the result to several hyperplanes. We make the restriction that no pair of hyperplanes should divide the point set in the same way; in that case,

*Or rather its intersection with the affine subspace in which the given points are contained.

they could be replaced by one hyperplane. Since the trivial partitions obtained by letting all points lie on the same side of a hyperplane will be uninteresting in Chapter 11, we also exclude these cases. The remaining partitions are called *nontrivial*.

Corollary A.2
A set of N points in general position in $\mathbb{R}^n$ can be partitioned in $\binom{f(n+1,N)/2-1}{M}$ nontrivial ways by M hyperplanes.

Proof From $f(n+1,N)/2-1$ nontrivial options of placing one hyperplane, we should choose M different ways to place the hyperplanes. $\square$

A.6 Inverting a Univariate Piecewise Affine Function

In Chapter 11, we will also need to invert a univariate, continuous, piecewise affine function. The following lemma can be used for this purpose.

Lemma A.4
Consider the function

$$\begin{aligned} f(x) &= x - \alpha_0 + \sum_{i=1}^{M} \sigma_i \max\{\beta_i x - \alpha_i, 0\} \\ \beta_i &> 0, \quad \frac{\alpha_1}{\beta_1} < \frac{\alpha_2}{\beta_2} < \cdots < \frac{\alpha_M}{\beta_M} \end{aligned} \tag{A.16}$$

where $\sigma_i \in \{-1, 1\}$, and assume that f is strictly increasing, i.e.[†],

$$0 < \sum_{m=0}^{k} \sigma_m \beta_m < \infty \quad \forall k = 1, \ldots, M \tag{A.17}$$

Then the inverse of $f(x)$ is

$$\begin{aligned} f^{-1}(y) &= y + \alpha_0 - \sum_{k=1}^{M} \sigma_k \max\{\tilde{\beta_k} y - \tilde{\alpha_k}, 0\} \\ \tilde{\beta_k} &= \frac{\beta_k}{\sum_{m=0}^{k-1} \sigma_m \beta_m \sum_{m=0}^{k} \sigma_m \beta_m} \\ \tilde{\alpha_k} &= \frac{\sum_{m=0}^{k-1} \sigma_m (\beta_m \alpha_k - \beta_k \alpha_m)}{\sum_{m=0}^{k-1} \sigma_m \beta_m \sum_{m=0}^{k} \sigma_m \beta_m} \end{aligned} \tag{A.18}$$

[†]For the sake of a more compact notation, let $\sigma_0 = \beta_0 = 1$.

Proof We prove the statement by induction over M. For $M = 0$, we have $f(x) = x - \alpha_0$, so $f^{-1}(y) = y + \alpha_0$. Now suppose that the statement holds for a certain $M-1 \geq 0$, and consider a function f with M breakpoints. For $x < \alpha_M/\beta_M$, f has only $M-1$ breakpoints, so in this region f^{-1} can be written as in (A.18), according to the induction assumption. For $x \geq \alpha_M/\beta_M$, we get

$$f(x) = \left(\sum_{m=0}^{M} \sigma_m \beta_m\right) x - \sum_{m=0}^{M} \sigma_m \alpha_m \tag{A.19}$$

so

$$f^{-1}(y) = \frac{y + \sum_{m=0}^{M} \sigma_m \alpha_m}{\sum_{m=0}^{M} \sigma_m \beta_m} \tag{A.20}$$

Since $x = f^{-1}(y)$, we can rewrite the condition $x \geq \alpha_M/\beta_M$ as

$$\begin{aligned}
& \frac{y + \sum_{m=0}^{M} \sigma_m \alpha_m}{\sum_{m=0}^{M} \sigma_m \beta_m} \geq \frac{\alpha_M}{\beta_M} \\
\Leftrightarrow\quad & \frac{1}{\sum_{m=0}^{M} \sigma_m \beta_m} \left(y - \frac{1}{\beta_M} \sum_{m=0}^{M-1} \sigma_m (\beta_m \alpha_M - \beta_M \alpha_m) \right) \geq 0 \\
\Leftrightarrow\quad & y \geq \frac{\tilde{\alpha}_M}{\tilde{\beta}_M}
\end{aligned}$$

Now, for $y \geq \tilde{\alpha}_M/\tilde{\beta}_M$, by using simple but tedious calculations we can rewrite (A.20) as

$$\begin{aligned}
f^{-1}(y) = {} & \frac{y + \sum_{m=0}^{M-1} \sigma_m \alpha_m}{\sum_{m=0}^{M-1} \sigma_m \beta_m} \\
& + \frac{-\sigma_M \beta_M}{\sum_{m=0}^{M-1} \sigma_m \beta_m \sum_{m=0}^{M} \sigma_m \beta_m} \left(y - \frac{1}{\beta_M} \sum_{m=0}^{M-1} \sigma_m (\beta_m \alpha_M - \beta_M \alpha_m) \right) \\
= {} & f_{M-1}^{-1}(y) - \sigma_M (\tilde{\beta}_M y - \tilde{\alpha}_M)
\end{aligned}$$

where $f_{M-1}^{-1}(y)$ is the inverse of the given function without the Mth max function. Since this function equals $f(x)$ for $x < \alpha_M/\beta_M$, we get $f_{M-1}^{-1}(y) = f^{-1}(y)$ for $y < \tilde{\alpha}_M/\tilde{\beta}_M$. Furthermore, by using a max function and by the induction assumption we now can write

$$\begin{aligned}
f^{-1}(y) &= f_{M-1}^{-1}(y) - \sigma_M \max\{\tilde{\beta}_M y - \tilde{\alpha}_M, 0\} \\
&= y + \alpha_0 - \sum_{k=1}^{M-1} \sigma_k \max\{\tilde{\beta}_k y - \tilde{\alpha}_k, 0\} - \sigma_M \max\{\tilde{\beta}_M y - \tilde{\alpha}_M, 0\} \\
&= y + \alpha_0 - \sum_{k=1}^{M} \sigma_k \max\{\tilde{\beta}_k y - \tilde{\alpha}_k, 0\}
\end{aligned}$$

which is just what we wanted to prove. □

Bibliography

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, pages 267–281, 1973.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[3] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.

[4] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[5] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, Feb. 1997.

[6] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11(1-5):75–113, Feb. 1997.

[7] C. G. Atkeson and D. J. Reinkensmeyer. Using associative content-addressable memories to control robots. In *The 27th IEEE Conference on Decision and Control*, pages 792–797, Dec. 1988.

[8] R. Batruni. A multilayer neural network with piecewise-linear structure and back-propagation learning. *IEEE Transactions on Neural Networks*, 2(3): 395–403, May 1991.

[9] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876, Oct. 2000.

[10] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A greedy approach to identification of piecewise affine models. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[11] A. Bemporad and D. Mignone. *MIQP.M: A Matlab function for solving mixed integer quadratic programs*. ETH Zurich, 2000. Code available at `http://control.ethz.ch/~hybrid/miqp`.

[12] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.

[13] A. Bemporad, J. Roll, and L. Ljung. Identification of hybrid systems via mixed-integer programming. In *The 40th IEEE Conference on Decision and Control*, pages 786–792, Dec. 2001.

[14] A. Bemporad, F. D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 45–58. Springer-Verlag, 2000.

[15] S. A. Billings and W. S. F. Voon. Piecewise linear identification of non-linear systems. *International Journal of Control*, 46(1):215–235, 1987.

[16] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[17] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for local modelling and control design. *International Journal of Control*, 72(7-8):643–658, May 1999.

[18] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, Nov. 1992.

[19] S. Boyd and L. Vandenberghe. Convex optimization. Course Reader for EE364, Introduction to Convex Optimization with Engineering Applications, Stanford University, May 3, 1999.

[20] M. S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43 (4):475–482, Apr. 1998.

[21] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[22] L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, May 1993.

[23] L. Breiman and J. H. Friedman. Function approximation using ramps. In *Snowbird Workshop. Machines that Learn*, 1994.

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification And Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[25] J. Bruls, C. T. Chou, B. R. J. Haverkamp, and M. Verhaegen. Linear and non-linear system identification using separable least-squares. *European Journal of Control*, 5(1):116–128, 1999.

[26] C.-H. Choi and J. Y. Choi. Constructive neural networks with piecewise interpolation capabilities for function approximations. *IEEE Transactions on Neural Networks*, 5(6):936–944, Nov. 1994.

[27] L. O. Chua and A.-C. Deng. Canonical piecewise-linear representation. *IEEE Transactions on Circuits and Systems*, 35(1):101–111, Jan. 1988.

[28] L. O. Chua and S. M. Kang. Section-wise piecewise-linear functions: Canonical representation, properties and applications. *Proceedings of the IEEE*, 65: 915–929, 1977.

[29] A. Chutinan. *Hybrid System Verification Using Discrete Model Approximations*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, May 1999.

[30] A. Chutinan and B. H. Krogh. Computing approximating automata for a class of linear hybrid systems. In *Hybrid Systems V*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[31] A. Chutinan and B. H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *The 37th IEEE Conference on Decision and Control: Session on Synthesis and Verification of Hybrid Control Laws (TM-01)*, 1998.

[32] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, Dec. 1979.

[33] W. S. Cleveland and S. J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, Sept. 1988.

[34] W. S. Cleveland and C. L. Loader. Smoothing by local regression: Principles and methods. In W. Haerdle and M. G. Schimek, editors, *Statistical Theory and Computational Aspects of Smoothing*, pages 10–49. Springer, New York, 1996.

[35] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14(3):326–334, June 1965.

[36] Dash Associates. *XPRESS-MP User Guide*, 1999.

[37] L. Davis. *Genetic Algorithms and Simulated Annealing.* London: Pitman; Los Altos, Calif.: Kaufmann, 1987.

[38] R. A. DeCarlo, M. S. Branicky, S. Pettersson, and B. Lennartsson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, July 2000.

[39] D. den Hertog. *Interior Point Approach to Linear, Quadratic and Convex Programming: Algorithms and Complexity.* Kluwer Academic, 1994.

[40] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, 1983.

[41] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[42] V. Einarsson. On verification of switched systems using abstractions. Licentiate Thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden. Thesis No. 705, 1998.

[43] V. Einarsson. *Model Checking Methods for Mode Switching Systems.* PhD thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, 2000.

[44] S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg. Continuous-discrete interactions in chemical processing plants. *Proceedings of the IEEE*, 88(7): 1050–1068, July 2000.

[45] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability and its Applications*, 14:153–158, 1969.

[46] S. Ernst. Hinging hyperplane trees for approximation and identification. In *The 37th IEEE Conference on Decision and Control*, volume 2, pages 1266–1271, 1998.

[47] J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications.* Chapman & Hall, 1996.

[48] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39 (2):205–217, Feb. 2003.

[49] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications.* Oxford University Press, 1995.

[50] A. T. Fuller. Relay control systems optimized for various performance criteria. In *1st World Congress IFAC*, pages 584–607, 1960.

[51] E. F. Gad, A. F. Atiya, S. Shaheen, and A. El-Dessouki. A new algorithm for learning in piecewise-linear neural networks. *Neural Networks*, 13(4-5): 485–505, 2000.

[52] A. Garulli, A. Tesi, and A. Vicino, editors. *Robustness in Identification and Control.* Lecture Notes in Control and Information Sciences. Springer-Verlag, 1999.

[53] T. Gasser and H.-G. Müller. Kernel estimation of regression functions. In T. Gasser and M. Rosenblatt, editors, *Smoothing Techniques for Curve Estimation*, Lecture Notes in Mathematics, pages 23–68. Springer-Verlag, 1979.

[54] T. Gasser, H.-G. Müller, and V. Mammitzsch. Kernels for nonparametric curve estimation. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 47(2):238–252, 1985.

[55] S. B. Gelfand and C. S. Ravishankar. A tree-structured piecewise linear adaptive filter. *IEEE Transactions on Information Theory*, 39(6):1907–1922, Nov. 1993.

[56] M. N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification.* PhD thesis, University of Cambridge, 1997.

[57] R. E. Groff, D. E. Koditschek, and P. P. Khargonekar. Piecewise linear homeomorphisms: The scalar case. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 259–264, 2000.

[58] F. Gustafsson. *Adaptive Filtering and Change Detection.* John Wiley & Sons, 2000.

[59] C. Güzeliş and İ. C. Göknar. A canonical representation for piecewise-affine maps and its applications to circuit analysis. *IEEE Transactions on Circuits and Systems*, 38(11):1342–1354, Nov. 1991.

[60] A. Hagenblad. Aspects of the identification of Wiener models. Licentiate Thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden. Thesis No. 793, 1999.

[61] W. Härdle. *Applied Nonparametric Regression.* Number 19 in Econometric Society Monographs. Cambridge University Press, 1990.

[62] W. P. M. H. Heemels, B. D. Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[63] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.

[64] E. A. Heredia and G. R. Arce. Piecewise linear system modeling based on a continuous threshold decomposition. *IEEE Transactions on Signal Processing*, 44(6):1440–1453, June 1996.

[65] I. Hoffmann. *Identifikation hybrider dynamischer Systeme.* PhD thesis, Universität Dortmund, 1999.

[66] I. Hoffmann and S. Engell. Identification of hybrid systems. In *American Control Conference*, pages 711–712, 1998.

[67] C. C. Holmes and B. K. Mallick. Bayesian regression with multivariate linear splines. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 63(1):3–17, 2001.

[68] C. M. Hurvich, J. S. Simonoff, and C.-L. Tsai. Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 60 (2):271–293, 1998.

[69] D. R. Hush and B. Horne. Efficient algorithms for function approximation with piecewise linear sigmoidal networks. *IEEE Transactions on Neural Networks*, 9(6):1129–1141, Nov. 1998.

[70] ILOG, Inc. *CPLEX 7.0 User's Manual.* Gentilly, France, 2000.

[71] T. A. Johansen and B. A. Foss. Identification of non-linear system structure and parameters using regime decomposition. *Automatica*, 31(2):321–326, Feb. 1995.

[72] M. Johansson. *Piecewise Linear Control Systems.* PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund University, Box 118, SE-221 00 Lund, Sweden, 1999.

[73] M. Johansson and A. Rantzer. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, 1998.

[74] M. C. Jones and M. P. Wand. Asymptotic effectiveness of some higher order kernels. *Journal of Statistical Planning and Inference*, 31(1):15–21, Apr. 1992.

[75] M. A. Jordán and O. A. A. Orqueda. Persistency of excitation and richness in continuous-time piecewise linear systems. In *1st International Conference on Control of Oscillations and Chaos*, volume 2, pages 234–237, Aug. 1997.

[76] P. Julián, A. Desages, and O. Agamennoni. High level canonical piecewise linear representation using a simplicial partition. *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, 46(4):463–480, 1999.

[77] P. Julián, M. Jordán, and A. Desages. Canonical piecewise-linear approximation of smooth functions. *IEEE Transactions on Circuits and Systems*, 45 (5):567–571, May 1998.

[78] P. M. Julián. *A High Level Canonical Piecewise Linear Representation: Theory and Applications*. PhD thesis, Departamento de Ingeniería Eléctrica, Universidad Nacional del Sur, 1999.

[79] C. Kahlert and L. O. Chua. The complete canonical piecewise-linear representation — part i: The geometry of the domain space. *IEEE Transactions on Circuits and Systems*, 39:222–236, 1992.

[80] A. M. Kang and L. O. Chua. A global representation of multidimensional piecewise-linear functions. *IEEE Transactions on Circuits and Systems*, CAS-25:938–940, Nov. 1978.

[81] G. Kerschen, J.-C. Golinval, and K. Worden. Theoretical and experimental identification of a non-linear beam. *Journal of Sound and Vibration*, 244(4): 597–613, July 2001.

[82] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Recurrent SOM with local linear models in time series prediction. In *6th European Symposium on Artificial Neural Networks*, pages 167–172, Apr. 1998.

[83] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88(7):1026–1049, July 2000.

[84] S. Kowalewski, O. Stursberg, M. Fritz, H. Graf, I. Hoffmann, J. Preußig, M. Remelhe, S. Simon, and H. Treseler. A case study in tool-aided analysis of discretely controlled continuous systems: the two tanks problem. In *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*, pages 163–185. Springer-Verlag, 1999.

[85] I. L. Legostaeva and A. N. Shiryaev. Minimax weights in a trend detection problem of a random process. *Theory of Probability and its Applications*, 16 (2):344–349, 1971.

[86] C.-A. Lehalle and R. Azencott. Piecewise affine neural networks and non-linear control. In *International Conference on Artificial Neural Networks*, volume 2, pages 633–638, Sept. 1998.

[87] C.-A. Lehalle and R. Azencott. Piecewise affine neural networks and non-linear control: Stability results. In *International Conference on Artificial Neural Networks*, pages 608–612, Sept. 1999.

[88] S. L. Leonov. Remarks on extremal problems in nonparametric curve estimation. *Statistics & Probability Letters*, 43(2):169–178, 1999.

[89] X. Li, S. Wang, and W. Yin. A canonical representation of high-dimensional continuous piecewise-linear functions. *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, 48(11):1347–1351, Nov. 2001.

[90] J.-N. Lin and R. Unbehauen. Canonical piecewise-linear approximations. *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, 39(8):697–699, Aug. 1992.

[91] J.-N. Lin, H.-Q. Xu, and R. Unbehauen. A generalization of canonical piecewise-linear functions. *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, 41(4):345–347, Apr. 1994.

[92] I. Lind. Regressor selection in system identification using ANOVA. Licentiate Thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden. Thesis No. 921, 2001.

[93] C. Livadas, J. Lygeros, and N. A. Lynch. High-level modeling and analysis of the traffic alert and collision avoidance system (TCAS). *Proceedings of the IEEE*, 88(7):926–948, July 2000.

[94] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, 2nd edition, 1999.

[95] C. R. Loader. Old Faithful erupts: Bandwidth selection reviewed. Technical report, AT&T Bell Laboratories, 1995.

[96] C. R. Loader. Locfit: An introduction. Technical report, AT&T Bell Laboratories, 1997.

[97] C. R. Loader. Bandwidth selection: Classical or plug-in? *The Annals of Statistics*, 27(2):415–438, Apr. 1999.

[98] J. Löfberg. *YALMIP 2.2 – Yet Another LMI Parser*. Linköping University, SE-581 83 Linköping, Sweden, 2002. `http://www.control.isy.liu.se/~johanl/yalmip.html`.

[99] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.

[100] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[101] D. J. C. MacKay. Gaussian processes: A replacement for supervised neural networks? Lecture notes for a tutorial at NIPS 1997, 1997.

[102] C. L. Mallows. Some comments on C_p. *Technometrics*, 15:661–676, 1973.

[103] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, July 1993.

[104] M. C. Medeiros, M. G. C. Resende, and A. Veiga. Piecewise linear time series estimation with GRASP. Technical report, AT&T Labs Research, Florham Park, NJ 07932 USA, 1999.

[105] M. C. Medeiros, A. Veiga, and M. G. C. Resende. A combinatorial approach to piecewise linear time series analysis. Technical report, AT&T Labs Research, Florham Park, NJ 07932 USA, 1999.

[106] D. Mignone, A. Bemporad, and M. Morari. A framework for control, fault detection, state estimation, and verification of hybrid systems. In *American Control Conference*, pages 134–138, June 1999.

[107] M. Milanese and A. Vicino. Optimal estimation theory for dynamic systems with set membership uncertainty: An overview. *Automatica*, 27(6):997–1009, Nov. 1991.

[108] T. Moor and J. Raisch. Discrete control of switched linear systems. In *European Control Conference*, Aug.–Sept. 1999.

[109] E. Münz and V. Krebs. Identification of hybrid systems using a priori knowledge. In *15th IFAC World Congress on Automatic Control*, July 2002.

[110] R. Murray-Smith. Local model networks and local learning. In *Fuzzy Duisburg*, pages 404–409, Feb. 1994.

[111] R. Murray-Smith and H. Gollee. A constructive learning algorithm for local model networks. In *IEEE Workshop on Computer-intensive methods in control and signal processing*, pages 21–29, 1994.

[112] R. Murray-Smith and T. A. Johansen, editors. *Multiple Model Approaches to Modelling and Control.* Taylor & Francis, 1997.

[113] E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 10:186–190, 1964.

[114] S. Nadjm-Tehrani and J.-E. Strömberg. Formal verification of dynamic properties in an aerospace application. *Formal Methods in System Design*, 14(2): 135–169, Mar. 1999.

[115] K. S. Narendra and S.-M. Li. Neural networks in control systems. In P. Smolensky, M. C. Mozer, and D. E. Rumelhart, editors, *Mathematical Perspectives on Neural Networks*, chapter 11, pages 347–394. Lawrence Erlbaum Associates, 1996.

[116] R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report 9702, Department of Statistics, University of Toronto, Jan. 1997.

[117] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer-Verlag, 1999.

[118] D. L. Pepyne and C. G. Cassandras. Optimal control of hybrid systems in manufacturing. *Proceedings of the IEEE*, 88(7):1108–1123, July 2000.

[119] V. Petridis and A. Kehagias. Identification of switching dynamical systems using multiple models. In *The 37th IEEE Conference on Decision and Control: Session on System Identification I (WA-07)*, 1998.

[120] S. Pettersson. *Analysis and Design of Hybrid Systems.* PhD thesis, Control Engineering Laboratory, Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, 1999.

[121] P. Philips, M. Weiss, and H. A. Preisig. Control based on discrete-event models of continuous systems. In *European Control Conference*, Aug.–Sept. 1999.

[122] C. Pizzi and P. Pellizzari. Adaptive local linear models for financial time series. *Journal of Computational Intelligence in Finance*, Jan. 1998.

[123] M. B. Priestley and M. T. Chao. Non-parametric function fitting. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 34, 385-392 1972.

[124] P. Pucar and J. Sjöberg. On the hinge-finding algorithm for hinging hyperplanes. *IEEE Transactions on Information Theory*, 44(3):1310–1319, May 1998.

[125] C. E. Rasmussen. *Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression.* PhD thesis, University of Toronto, 1996.

[126] C. Rhodes and M. Morari. The false nearest neighbors algorithm: An overview. *Computers & Chemical Engineering*, 21(Suppl. 1):S1149–S1154, May 1997.

[127] J. Roll. Invariance of approximating automata for piecewise linear systems. Technical Report LiTH-ISY-R-2178, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, 1999.

[128] J. Roll. Invariance of approximating automata for piecewise linear systems with uncertainties. In *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 396–406. Springer-Verlag, 2000.

[129] J. Roll. Robust verification of piecewise affine systems. In *15th IFAC World Congress on Automatic Control, Session T-We-A21*, July 2002.

[130] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. Provisionally accepted for Automatica, 2003.

[131] J. Roll, A. Nazin, and L. Ljung. A non-asymptotic approach to local modelling. In *The 41st IEEE Conference on Decision and Control*, pages 638–643, Dec. 2002.

[132] J. Roll, A. Nazin, and L. Ljung. Direct weight optimization for nonparametric estimation of a regression function at a given point. Submitted to Scandinavian Journal of Statistics, 2003.

[133] J. Roll, A. Nazin, and L. Ljung. Local modelling of nonlinear dynamic systems using direct weight optimization. Accepted for the 13th IFAC Symposium on System Identification, Rotterdam, Aug. 2003.

[134] J. Roll, A. Nazin, and L. Ljung. Local modelling with a priori known bounds using direct weight optimization. Submitted to the European Control Conference, Cambridge, Sept. 2003.

[135] M. Rubensson. Discrete-time stability analysis of hybrid systems. Licentiate Thesis, Department of Signals and Systems, Chalmers University of Technology, Sweden, 2000.

[136] D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *The Annals of Statistics*, 22(3):1346–1370, Sept. 1994.

[137] J. Sacks and W. Strawderman. Improvements on linear minimax estimates. In *Statistical decision theory and related topics III*, volume 2, pages 287–304. Academic Press, 1982.

[138] J. Sacks and D. Ylvisaker. Linear estimation for approximately linear models. *The Annals of Statistics*, 6(5):1122–1137, 1978.

[139] N. V. Sahinidis. BARON – Branch And Reduce Optimization Navigator. Technical report, University of Illinois at Urbana-Champaign, Dept. of Chemical Engineering, Urbana, IL, USA, 2000.

[140] S. Simani, C. Fantuzzi, R. Rovatti, and S. Beghelli. Parameter identification for piecewise-affine fuzzy models in noisy environment. *International Journal of Approximate Reasoning*, 22(1-2):149–167, Sept.–Oct. 1999.

[141] A. C. Singer, G. W. Wornell, and A. V. Oppenheim. Codebook prediction: A nonlinear signal modeling paradigm. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 325–328, 1992.

[142] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

[143] A. Skeppstedt. Construction of composite models from large data-sets. Licentiate Thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden. Thesis No. 149, 1988.

[144] A. Skeppstedt, L. Ljung, and M. Millnert. Construction of composite models from observed data. *International Journal of Control*, 55(1):141–152, 1992.

[145] E. D. Sontag. Interconnected automata and linear systems: A theoretical framework in discrete-time. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III - Verification and Control*, number 1066 in Lecture Notes in Computer Science, pages 436–448. Springer-Verlag, 1996.

[146] A. Stenman. *Model on Demand: Algorithms, Analysis and Applications*. PhD thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, 1999.

[147] C. J. Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5(4):595–645, 1977.

[148] J.-E. Strömberg, F. Gustafsson, and L. Ljung. Trees as black-box model structures for dynamical systems. In *European Control Conference*, pages 1175–1180, Grenoble, France, July 1991.

[149] J. F. Sturm. *SeDuMi 1.05*, 2002. `http://fewcal.kub.nl/sturm/software/sedumi.html`.

[150] O. Stursberg and S. Kowalewski. Approximating switched continuous systems by rectangular automata. In *European Control Conference*, Aug.–Sept. 1999.

[151] S. A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, 1991.

[152] J. Vesanto. Using the SOM and local models in time-series prediction. In *Workshop on Self-Organizing Maps*, pages 209–214, Espoo, Finland, June 1997.

[153] J. Vörös. Parameter identification of Wiener systems with discontinuous nonlinearities. *Systems & Control Letters*, 44(5):363–372, Dec. 2001.

[154] J. Walter, H. Ritter, and K. Schulten. Non-linear prediction with self-organizing maps. In *International Joint Conference on Neural Networks*, volume 2, pages 747–752, 1990.

[155] M. P. Wand and M. C. Jones. *Kernel Smoothing.* Chapman & Hall, 1995.

[156] G. S. Watson. Smooth regression analysis. *Sankhyā, Series A*, 26:359–372, 1964.

[157] T. Wigren. Recursive prediction error identification using the nonlinear Wiener model. *Automatica*, 29(4):1011–1025, 1993.

[158] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, 1996.

[159] H. P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons, 4th edition, 1999.

[160] L. A. Wolsey. *Integer Programming.* Wiley, 1998.

[161] F. Zhang. *Matrix Theory: Basic Results and Techniques.* Springer-Verlag, 1999.

[162] L. H. Zhao. Minimax linear estimation in a white noise problem. *The Annals of Statistics*, 25(2):745–755, 1997.

PhD Dissertations, Division of Automatic Control, Linköping University

M. Millnert: Identification and control of systems subject to abrupt changes. Thesis no. 82, 1982. ISBN 91-7372-542-0.

A.J.M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis no. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis no. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis no. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis no. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis no. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis no. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis no. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis no. 194, 1988. ISBN 91-7870-380-8.

A. Isaksson: On system identification in one and two dimensions with signal processing applications. Thesis no. 196, 1988. ISBN 91-7870-383-2.

M. Viberg: Subspace fitting concepts in sensor array processing. Thesis no. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis no. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis no. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis no. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis no. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis no. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis no. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis no. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis no. 353, 1994. ISBN 91-7871-430-3.