Linköping Studies in Science and Technology. Dissertations
No. 571

Model on Demand: Algorithms, Analysis and Applications

Anders Stenman

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

Linköping 1999

Model on Demand: Algorithms, Analysis and Applications

`stenman@isy.liu.se`
`www.control.isy.liu.se/~stenman`
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 91-7219-450-2 ISSN 0345-7524

Printed in Sweden by Linus & Linnea AB

This page is intentionally left blank.

Abstract

System identification deals with the problem of estimating models of dynamical systems from observed data. In this thesis, we focus on the identification of nonlinear models, and, in particular, on the situation that occurs when a very large amount of data is available.

Traditional treatments of the estimation problem in statistics and system identification have mainly focused on global modeling approaches, i.e., the model has been optimized on basis of the entire data set. However, when the number of observations grows very large, this approach becomes less attractive to deal with because of the difficulties in specifying model structure and the complexity of the associated optimization problem. Inspired by ideas from local modeling and database systems technology, we have taken a conceptually different point of view. We assume that all available data are stored in a database, and that models are built “on demand” as the actual need arises. When doing so, the bias/variance trade-off inherent to all modeling is optimized locally by adapting the number of data and their relative weighting. For this concept, the name *model-on-demand* has been adopted.

In this thesis we have adopted a weighted regression approach for the modeling part, where a weight sequence is introduced to localize the estimation problem. Two conceptually different approaches for weight selection are discussed, where the first is based on traditional kernel assumptions and the other relies on an explicit optimization stage. Furthermore, two algorithms corresponding to these approaches are presented and their asymptotic properties are analyzed. It is concluded that the optimization approach might produce more accurate predictions, but that it at the same time is more demanding in terms of computational efforts.

Compared to global methods, and advantage with the model-on-demand concept is that the models are optimized locally, which might decrease the modeling error. A potential drawback is the computational complexity, both since we have to search for neighborhoods in a multidimensional regressor space, and since the derived estimators are quite demanding in terms of computational resources.

Three important applications for the concept are presented. The first one addresses the problem of nonlinear time-domain identification. A number of nonlinear model structures are evaluated from a model-on-demand perspective and it is concluded that the method is useful for predicting and simulating nonlinear systems provided sufficiently large datasets are available. It is demonstrated through simulations that the prediction errors are in order of magnitude directly comparable to more established modeling tools such as artificial neural nets and fuzzy identification.

The second application addresses the frequency-domain identification problems that occur when estimating spectra of time series or frequency responses of linear systems. We show that the model-on-demand approach provides a very good way of estimating such quantities using automatic, adaptive and frequency-dependent choices of frequency resolution. This gives several advantages over traditional spectral analysis techniques.

The third application, which is closely related to the first one, is control of nonlinear processes. Here we utilize the predictive power of the model-on-demand estimator for on-line optimization of control actions. A particular method, model-free predictive control, is presented, that combines model-on-demand estimation with established model predictive control techniques.

Acknowledgments

I am very grateful to all the people that have supported me during the work with the thesis. First of all, I would like to thank my supervisors, Professors Fredrik Gustafsson and Lennart Ljung, for their guidance through the work. Especially Fredrik deserves my deepest gratitude for his infinite patience and for putting up with all my stupid questions. I am also indebted to Dr Daniel E. Rivera (the "sparky" guy) who invited me to visit ASU last year. Thank you for taking care of me while being there, for insightful discussions about my work and for giving interesting new ideas and proposals for future research.

Fredrik Tjärnström have read the thesis thoroughly and have given valuable comments and suggestions for improvements. For this I am very grateful. I also want to thank Johan Löfberg for initial discussions on model predictive control.

Now some words (in Swedish) for my private sponsors:

Först av allt skulle jag vilja tacka morsan och farsan för att ni alltid ställer upp i alla väder. Tack även till syrran och brorsan med familjer, för glada tillrop när allt kändes trist och ledsamt. Jag vill också passa på att rikta ett tack till mina fina och välartade kollegor i reglerteknikgruppen. Må era fotriktiga sandaler bära er långt i livet och glöm inte bort att alltid trampa i nerförsbackarna. Förlåt mig även för att jag alltid sabbar era angelägna fikarumskonversationer med sura uppstötningar. ;-)

This work was supported by the Swedish Research Council for Engineering Sciences (TFR), which is gratefully acknowledged.

Linköping, April 1999
Anders Stenman

Contents

1 Introduction

The problem considered in this thesis is how to derive relationships between inputs and outputs of a dynamical system when very little a priori knowledge is available. In traditional system identification literature, this is usually known as black-box modeling. Very rich and well established theory for black-box modeling of linear systems exists, see for example Ljung (1999) and Söderström and Stoica (1989). In recent years the interest for nonlinear system identification has been growing, and the attention has been focused on a number of nonlinear black-box structures; neural networks, wavelets as to mention some of them (Sjöberg *et al.*, 1995). However, nonlinear identification and function approximation have been studied for a long time within the statistical community, as instances of the more general regression problem (Härdle, 1990).

1.1 The Regression Problem

Within many areas of science one often wishes to study the relationship between a number of variables. The purpose could for example be to predict the outcome of one of the variables, on the basis of information provided by the others. In the statistical theory this is usually referred to as the *regression problem*. The objective is to determine a functional relationship between a predictor variable (or regression variable) $X \in \mathbb{R}^d$ and a response variable $Y \in \mathbb{R}$ (or $Y \in \mathbb{C}$), given a set of observations $\{(X_1, Y_1), \dots, (X_N, Y_N)\}$. Mathematically, the problem is to find a function of X, $m(X)$, so that the difference

$$Y - m(X) \tag{1.1}$$

becomes small in some sense. It is a well-known fact that the function m that minimizes

$$\mathrm{E}(Y - m(X))^2 \tag{1.2}$$

for a fixed point $X = x$, is the conditional expectation of Y given X. That is,

$$m(x) = \mathrm{E}(Y|X = x). \tag{1.3}$$

This function, the best mean square predictor of Y given X, is often referred to as the *regression function* or the regression of Y on X (Wand and Jones, 1995). Hence it follows that the observed data usually are modeled as

$$\boxed{Y_i = m(X_i) + \epsilon_i, \qquad i = 1, \ldots, N} \tag{1.4}$$

where $\{\epsilon_i\}$ are identically distributed random variables with zero means, which are independent of the predictor data $\{X_i\}$. We will assume this to be our canonical model in the sequel of the thesis.

The task of estimating the regression function $m(x)$ from observations can be done in essentially two different ways. The quite commonly used *parametric* approach is to assume that the function $m(\cdot)$ has a pre-specified form, for instance a hyperplane with unknown slope and offset. As an alternative one could try to estimate $m(\cdot)$ *non-parametrically* without reference to a specific form.

1.1.1 Parametric Regression

Parametric estimation methods rely on the assumption that the true regression function m has a pre-specified functional form, which can be fully described by a finite dimensional parameter vector θ;

$$m(x) = m(x, \theta). \tag{1.5}$$

The structure of the model is chosen from families that are known to be flexible and which have been successful in previous applications. This means that the parameters are not necessarily required to have any physical meaning, they are just tuned to fit the observed data as well as possible.

It turns out that it is useful to interprete the parameterization (1.5) as an expansion in terms of basis functions $g_i(\cdot)$, i.e.,

$$m(x, \theta) = \sum_{i=1}^{r} \alpha_i \, g_i(x, \beta_i, \gamma_i). \tag{1.6}$$

This formulation allows that the dependence of m in some of the components of θ can be linear while it on others can be nonlinear. This is for instance the situation in some commonly used special cases of basis functions as summarized in Table 1.1, which will be more thoroughly described in Chapter 2.

Modeling Approach	$g_i(x, \beta_i, \gamma_i)$
Fourier series	$\sin(\beta_i x + \gamma_i)$
Feedforward neural networks	$1/(1+\exp(-\beta_i^T x - \gamma_i))$
Radial basis functions	$\exp(-\|x-\gamma_i\|_{\beta_i}^2)$
Linear regression	x_i

Table 1.1 *Basis functions used in some common modeling approaches.*

Once a particular model structure is chosen, the parameters can be estimated from the observations by an optimization procedure that minimizes the "size" of the prediction errors in a global fashion,

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} (Y_i - m(X_i, \theta))^2 , \tag{1.7a}$$

$$\hat{m}(x) = m(x, \hat{\theta}). \tag{1.7b}$$

This optimization problem usually has to be solved using a numeric search routine, except in the linear regression case where an explicit solution exists. Hence it will typically have numerous local minima which in general makes the search for the desired global minimum hard (Cybenko, 1996).

The greatest advantage with parametric models is that they give a very compact description of the dataset once the parameter vector estimate $\hat{\theta}$ is determined. A drawback, however, is the required assumption of the imposed parameterization. Sometimes the assumed function family (or model structure) might be too restrictive or too low-dimensional (i.e., too few parameters) to fit unexpected features in the data.

1.1.2 Nonparametric Regression

The problems with parametric regression methods can be overcome by removing the restriction that the regression function belongs to a parametric function family. This leads to an approach that usually is referred to as *nonparametric regression*. The basic idea behind nonparametric methods is that one should let the data decide which function that fits them best without the restrictions imposed by a parametric model.

There exist several methods for obtaining nonparametric estimates of the functional relationship, ranging from the simple nearest neighbor method to more advanced smoothing techniques. A fundamental assumption is that observations located close to each other are related, so that an estimate at a certain operation point x can be constructed from observations in a small neighborhood around x.

The simplest nonparametric method is perhaps the *nearest neighbor* approach. The estimate $\hat{m}(x)$ is taken as the response variable Y_i that corresponds to the predictor variable X_i that is the nearest neighbor of x, i.e.

$$\hat{m}(x) = Y_{i^*}, \qquad i^* = \arg\min_{i} \|X_i - x\| .$$

Hence the estimation problem is essentially reduced to a dataset searching problem, rather than a modeling problem.

Due to its simplicity, the nearest neighbor method suffers from a major drawback. The observations are almost always corrupted by measurement noise. Hence the nearest neighbor estimate is in general a very poor and noisy estimate of the true function value. Significant improvements can therefore be achieved using an interpolation or *smoothing* operation,

$$\hat{m}(x) = \sum_i W_i(x) Y_i \tag{1.8}$$

where $\{W_i(x)\}$ denotes a sequence of weights that may depend on both x and the predictor variables. The weights can of course be selected in many ways. A popular and traditionally used approach in statistics is to select the weights according to a *kernel function* (Härdle, 1990), which explicitly specifies the shape of the weight sequence. A similar approach has been considered in traditional signal processing applications where the so-called Hamming window is frequently used for smoothing (Blackman and Tukey, 1958).

In more sophisticated nonparametric techniques like *local polynomial smoothers* (Fan and Gijbels, 1996) the weights W_i are obtained implicitly by letting the kernel localize the functional approximation (1.7) around x. That is,

$$\hat{\theta}(x) = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} (Y_i - m(X_i, \theta))^2 \, w_i(x), \tag{1.9a}$$

$$\hat{m}(x) = m(x, \hat{\theta}(x)). \tag{1.9b}$$

For models that are linear in the parameters (e.g., polynomial models), this results in a linear smoothing operation analogous to (1.8).

The weights in (1.8) and (1.9) are typically tuned by a *smoothing parameter* which controls the degree of local averaging, i.e. the size of the neighborhood around x. A too large neighborhood will include observations located far away from x, whose expected values may differ significantly from $m(x)$, and as a result the estimator will produce an "over-smoothed" or *biased* estimate. When using a too small neighborhood, on the other hand, only a few number of observations will contribute to the estimate at x, hence making it "under-smoothed" or noisy. The basic problem in nonparametric methods is thus to find the optimal choice of the smoothing parameter that will balance the bias error against the variance error.

The advantage with nonparametric models is their *flexibility*, since they allow predictions to be computed without reference to a fixed parametric model. The price that has to be paid for that is the computational complexity. In general nonparametric methods require more computations than parametric ones. The convergence rate with respect to sample size N is also slower than for parametric methods.

1.2 System Identification

System identification is a special case of the regression problem presented in the previous section. It deals with the problem of determining mathematical models of dynamical systems on the basis of observed data from the systems, and can be divided into two main branches; *time-domain* and *frequency-domain* methods.

1.2.1 Time-Domain Identification

Having collected a dataset of paired inputs and outputs,

$$Z^N = \{(u(1), y(1)), \dots, (u(N), y(N))\},$$

from a system, the goal in *time-domain system identification* is to model future system outputs as a function of past data. It is normally assumed that the system dynamics can be modeled as

$$y(t) = m(\varphi(t)) + v(t), \qquad t = 1, \dots, N, \tag{1.10}$$

where $\varphi(t) = \varphi(Z^{t-1})$ is a so-called regression vector constructed from the elements of Z^{t-1}, and $v(t)$ is an error term which accounts for the fact that in general it is not possible to model $y(t)$ as an exact function of past observations. Nevertheless, a requirement must be that the error term is small or white, so that we can treat $\hat{m}(\varphi(t))$ as a good prediction of $y(t)$, that is,

$$\hat{y}(t) = \hat{y}(t|t-1) = \hat{m}(\varphi(t)). \tag{1.11}$$

The system identification problem is thus to find a "good" function $m(\cdot)$ such that the discrepancy between the true and the predicted outputs, i.e., the *prediction error*

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

is minimized.

The problem of determining $\hat{y}(t)$ from experimental data with poor or no a priori knowledge of the system is usually referred to as *black-box modeling* (Ljung, 1999). It has traditionally been solved using parametric linear models of different sophistication. However problems usually occur when encountering highly nonlinear systems that poorly allow themselves to be approximated by linear models. As a consequence of this, the interest for nonlinear modeling alternatives like neural networks and radial basis functions has been growing in recent years (Sjöberg *et al.*, 1995; Chen *et al.*, 1990).

In this thesis we will instead apply nonparametric methods of the type described in Section 1.1.2. This will result in predictors of the form

$$\hat{y}(t) = \sum_k W_k(\varphi(t))\, y(k), \tag{1.12}$$

where the weights typically are tuned so that they give output measurements associated with regressors located close to $\varphi(t)$ more influence than those located far away from it. It turns out that this is a promising approach for dealing with nonlinear systems modeling, as the following simple example also illustrates.

Example 1.1 **A simple laboratory-scale tank system**

Consider the laboratory-scale tank system shown in Figure 1.1 (a). It has earlier been investigated by Lindskog (1996) and Stenman (1997). Suppose the modeling aim is to describe how the water level $h(t)$ changes with the voltage $u(t)$ that controls the pump, given a dataset that consists of 1000 observations of $u(t)$ and $h(t)$. The dataset is plotted in Figure 1.1 (b).

A reasonable assumption is that the water level at the current time instant t can be expressed in terms of the water level and the pump voltage at the previous time instant $t-1$, i.e.,

$$\hat{h}(t) = m(h(t-1), u(t-1)).$$

Assuming that the function m can be described by a linear regression model,

$$m(h(t-1), u(t-1)) = \theta_1 h(t-1) + \theta_2 u(t-1) + \theta_0, \tag{1.13}$$

the parameters θ_i can easily be estimated using linear least squares, resulting in

$$\hat{\theta}_1 = 0.9063, \quad \hat{\theta}_2 = 1.2064, \quad \text{and} \quad \hat{\theta}_0 = -5.1611.$$

The result from a simulation using these values is shown in Figure 1.1 (c). The solid line represents the measured water level, and the dashed line corresponds to a simulation using the estimated parameters. As shown, the simulated water level follows the true level quite well except at levels close to zero, where the linear model produces negative levels. This indicates that the true system is nonlinear, and that better results could be achieved using a nonlinear or a nonparametric model. Figure 1.1 (d) shows a simulation using a nonparametric model obtained from the model structure (1.13) and a local modeling procedure of the type (1.9)[1]. The performance of the model is clearly much better at low water levels in this case. ❑

1.2.2 Frequency-Domain Identification

A traditional application of nonparametric methods in system identification arises in the *frequency domain* when estimating frequency functions of dynamical systems. If the system considered is linear, i.e., if the description (1.10) can be simplified as

$$y(t) = G(q)u(t) + v(t), \qquad t = 1, \ldots, N, \tag{1.14}$$

an estimate of the frequency function $G(e^{i\omega})$ can be formed as the ratio between the Fourier transforms of the output and input signals,

$$\hat{\hat{G}}(e^{i\omega_k}) = \frac{Y(\omega_k)}{U(\omega_k)} = \frac{\sum_{t=1}^{N} y(t)\, e^{-i\omega_k t}}{\sum_{t=1}^{N} u(t)\, e^{-i\omega_k t}}. \tag{1.15}$$

[1]The simulation has been performed using Algorithm 4.2 in Chapter 4. Consult Stenman (1997) for experimental details.

This estimate is often called the *empirical transfer function estimate*, ETFE (Ljung, 1999). It is well-known that the ETFE is a very crude and noisy estimate of the true frequency response. In particular, for sufficiently large N, the ETFE can be modeled as

$$\hat{\hat{G}}(e^{i\omega_k}) = G(e^{i\omega_k}) + \epsilon_k,$$

where ϵ_k is a complex-valued disturbance with zero mean and variance proportional to the noise to input signal ratio. Since this model fits into the general regression formulation (1.4), the frequency function value at the frequency ω can be estimated in a nonparametric fashion

$$\hat{G}(e^{i\omega}) = \sum_k W_k(\omega)\hat{\hat{G}}(e^{i\omega_k}), \tag{1.16}$$

where the weights again are selected so that a good trade-off between bias and variance is achieved.

Frequency response estimation is closely related to *spectral analysis*. This is easily realized by noting that (1.15) can be rewritten as

$$\hat{\hat{G}}(e^{i\omega_k}) = \frac{Y(\omega_k)}{U(\omega_k)} = \frac{\frac{1}{N}Y(\omega_k)U^*(\omega_k)}{\frac{1}{N}|U(\omega_k)|^2} = \frac{\hat{\hat{\Phi}}_{yu}(\omega_k)}{\hat{\hat{\Phi}}_u(\omega_k)}, \tag{1.17}$$

Here $\hat{\hat{\Phi}}_u(\omega)$ and $\hat{\hat{\Phi}}_{yu}(\omega)$ are referred to as the *periodogram* of $u(t)$ and *cross-periodogram* between $y(t)$ and $u(t)$ respectively (Brockwell and Davis, 1987). The periodograms can similar to the ETFE be modeled as

$$\hat{\hat{\Phi}}_u(\omega_k) = \Phi_u(\omega_k) + \epsilon'_k \qquad \text{and} \qquad \hat{\hat{\Phi}}_{yu}(\omega_k) = \Phi_{yu}(\omega_k) + \epsilon''_k.$$

where $\Phi_u(\omega_k)$ and $\Phi_{yu}(\omega_k)$ denote the true spectra, and ϵ'_k and ϵ''_k are zero mean disturbances. Thus the corresponding spectrum and cross-spectrum can be estimated in a nonparametric fashion according to

$$\hat{\Phi}_u(\omega) = \sum_k W_k(\omega)\hat{\hat{\Phi}}_u(\omega_k), \tag{1.18a}$$

$$\hat{\Phi}_{yu}(\omega) = \sum_k W_k(\omega)\hat{\hat{\Phi}}_{yu}(\omega_k). \tag{1.18b}$$

This implies that an alternate way of estimating the frequency response is by means of the ratio

$$\hat{G}(e^{i\omega}) = \frac{\hat{\Phi}_{yu}(\omega)}{\hat{\Phi}_u(\omega)}. \tag{1.19}$$

1.3 Model-on-Demand

The main contribution in this thesis is *model-on-demand*, MOD, which is another approach of obtaining nonparametric estimates of nonlinear regression functions on the basis of observed data. The concept was originally formulated for solving time-domain prediction problems like (1.12) given large datasets, but it turned out that it also worked out well on the frequency-domain problems of Section 1.2.2.

1.3.1 The Basic Ideas

The model-on-demand approach is a “data mining” technology that takes advantage of the increasing ability of computers to collect and manipulate large amounts of data. The basic idea behind the concept is that all available observations are stored in a database, and that models are built “on demand” as the actual need arises. This leads to an estimation procedure consisting of essentially two parts – a dataset searching procedure and a local modeling procedure.

For simplicity, it is as in (1.8) assumed that the modeling part is implemented as a weighted average of the response variables in a neighborhood around x,

$$\hat{m}(x) = \sum_i W_i Y_i, \tag{1.20}$$

where the weights W_i are tuned in such a way that the pointwise mean square error (MSE) measure is minimized. We will for this purpose study two different approaches for weight selection. The first one is based on local polynomial techniques similar to (1.9) and assumes that the weights are implicitly specified by a kernel. The second approach instead relies upon a direct *optimization scheme* that determines the weight sequence explicitly.

Compared to global methods, an advantage with the MOD approach is that the modeling is optimized locally, which might improve the performance. A potential drawback is the computational complexity, as we both have to search for a neighborhood of x in a multidimensional space, and as the derived estimator is quite computationally intensive. In this thesis, however, we will only investigate the properties of the modeling part of the problem. The searching problem is left as a topic for future research and is only briefly discussed in the thesis.

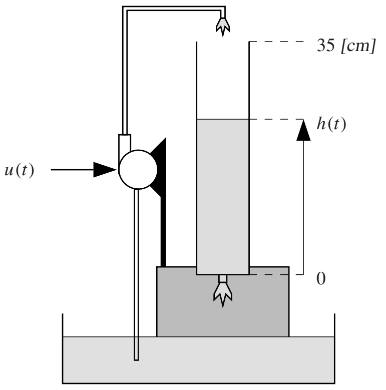
1.3.2 Applications

The main applications for the developed smoothing methods are the three identification problems (1.12), (1.16) and (1.18). There may of course be many reasons and needs for estimating such models. Some of them may be as follows:

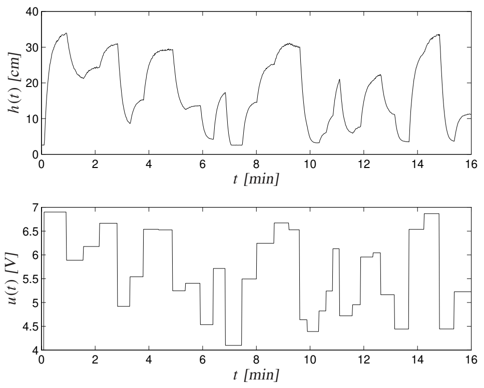
- Based on the observations we have collected so far, we will be able to forecast the future behavior of the system. This is usually referred to as *prediction*. Conceptually speaking, this can be formalized as in Figure 1.2. The predictor/estimator takes a dataset and an operation point x as inputs, and uses some suitable modeling approach, parametric or nonparametric, to produce an estimate $\hat{m}(x)$.

- System analysis and fault detection in general require investigation or monitoring of certain parameters which may not be directly available through measurements. We will therefore have to derive their values by means of a model of the system.

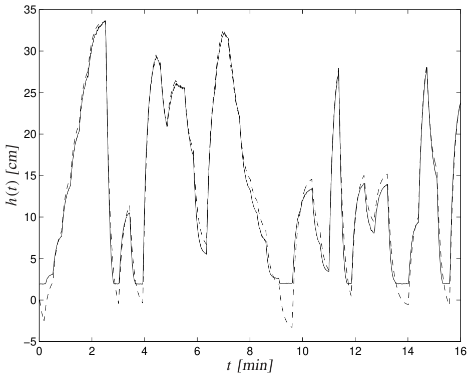
- Modern control theory usually requires a model of the process to be controlled. One example is *predictive control* where the control signal from the regulator is optimized on the basis of predictions of future outputs of the system.



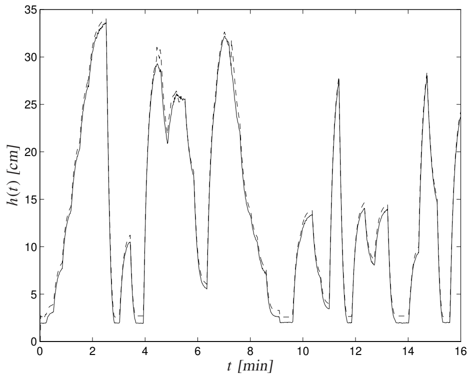
(a)



(b)



(c)



(d)

Figure 1.1 *(a) A simple tank system. (b) Experimental data. (c) The result of a simulation using a linear model. (d) The result of a simulation with a nonparametric model. The solid lines are the true level signals. The dashed lines are the simulated signals.*

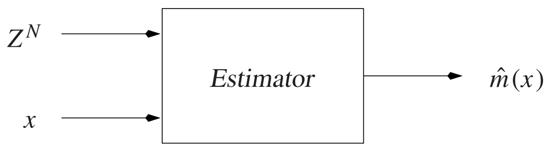


Figure 1.2 *A conceptual view of prediction.*

1.3.3 Related Approaches in the Literature

The idea of local modeling is not new. Various local approaches have been studied for a long time within the statistical community (Härdle, 1990), although there almost always seems to have been global assumptions and considerations in some step of the estimation procedure. However, adaptive methods have gained a significant interest in recent years, and the development of them still seems to be an open and active research area, see for instance Fan and Gijbels (1996) and the references therein.

In the domain of machine learning and artificial intelligence, the nonparametric ideas were rediscovered and developed by Atkeson and Reinkensmeyer (1988), Aha (1989) and Bottou and Vapnik (1992), and have successfully been used under the names *lazy learning* and *least commitment learning* for robot learning and control (Atkeson *et al.*, 1997*a*; Atkeson *et al.*, 1997*b*).

1.4 Thesis Outline

The thesis is divided into six chapters, in addition to the introductory and concluding chapters. The first two chapters give an overview of existing parametric and nonparametric methods that relate to the model-on-demand concept, and the last four chapters derive, analyze and exemplify the proposed methods.

The purpose of Chapter 2 is to give the reader a brief background on parametric estimation methods, especially in system identification applications. Examples of some commonly used linear and nonlinear black-box models are given, along with the two basic parameter estimation methods.

Chapter 3 serves as an introduction to nonparametric smoothing methods. The chapter is mainly focused on a special class of so-called kernel estimation methods which is widely used in statistics. The fundamental ideas and terminology are presented as well as the statistical and asymptotical properties that are associated with these methods.

Chapter 4 is the core chapter of the thesis. It presents the basic ideas behind the model-on-demand concept and discusses possible implementations. The chapter is concluded with a discussion regarding different properties and computational aspects of the methods.

In Chapter 5 the model-on-demand approach is applied to the time domain system identification problem (1.12). First two simulated examples are considered, and then three real data applications, a water heating system, a hydraulic robot and a buffer vessel are successfully simulated by the method.

Chapter 6 presents important applications for the model-on-demand method in the fields of spectral analysis and frequency response estimation. In particular we try to solve the smoothing problems (1.16) and (1.18). The chapter starts by a review of the traditional treatments of the topics, whereafter the model-on-demand modeling concept is modified to fit into this framework.

Chapter 7 discusses how the local modeling approach can utilized in a control context. The chapter starts with a brief review of classical approaches such as adaptive control, whereafter a more thoroughly treatment of different prediction-based control methods is given. Special attention is payed to the *model predictive control* approach, and it shown through simulations that the model-on-demand approach is a useful alternative to more traditional modeling approaches in this framework.

Finally, Chapter 8 gives a summary of the thesis and directions for future work.

1.5 Contributions

The contributions of this thesis are mainly the material contained in Chapters 4 to 7. They can be summarized as follows:

- The concept of model-on-demand models is advocated as a method for modeling system behaviors given large data volumes.
- Particular implementations of model-on-demand algorithms are proposed, that form estimates in a nonparametric fashion as a weighted average of the response variables. Two different approaches for weight selection are studied, where the first is based on traditional kernel functions and the other relies on an explicit optimization stage. The performance of the estimator is in both cases optimized locally by adapting the number of contributing data and their relative weighting.
- An analysis of the asymptotic properties of the optimization approach is outlined. It is shown that the method produces consistent estimates and that the convergence rate is in the same order as for nonparametric methods. A comparison to traditional local polynomial estimators is made, and it is concluded that the optimization approach may produce more accurate predictions.
- Simulation studies in the time-domain identification setting show that the model-on-demand method for some applications give smaller prediction errors than other proposed methods like neural nets and fuzzy identification.
- It is shown that the derived methods are quite efficient in matter of performance for smoothing frequency response estimates and periodograms. In particular we present methods that outperform traditional spectral analysis techniques by allowing adaptive and frequency-dependent choice of frequency resolution.
- It is demonstrated how the time-domain prediction methods can be utilized in a predictive control framework. By treating the local model obtained at each sample time as a local linearization, it is possible to reuse tools and concepts from the linear MPC/GPC framework. Three different variants of the idea, based on local linearization, linearization along a trajectory and nonlinear optimization respectively, are presented. They are all illustrated in numerical simulations.

1.6 Publications

The thesis is based on a number of publications, some of that have been, or will be, presented at different conferences. These are:

Stenman, A. (1997). *Just-in-Time Models with Applications to Dynamical Systems*. Licentiate thesis LIU-TEK-LIC-1997:02. Department of Electrical Engineering, Linköping University. S-581 83 Linköping, Sweden.

Stenman, A., A.V. Nazin and F. Gustafsson (1997). Asymptotic properties of Just-in-Time models. In: *Preprints of the 11th IFAC Symposium on System Identification, Kitakyushu, Japan* (Y. Sawaragi and S. Sagara, Eds.). pp. 1249–1254.

Stenman, A., F. Gustafsson and L. Ljung (1996). Just in time models for dynamical systems. In: *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan.* pp. 1115–1120.

Stenman, A., F. Gustafsson, D.E. Rivera, L. Ljung and T. McKelvey (1999). On adaptive smoothing of empirical transfer function estimates. To be presented at the IFAC World Congress, July 1999, in Beijing, China.

Rivera, D. E., S. Adusumilli, A. Stenman, F. Gustafsson, T. McKelvey and L. Ljung. (1997). Just-in-time models for improved identification and control. In: *Proc. American Inst. of Chemical Engineers (AIChE) Annual Meeting*.

Braun, M.W., D.E. Rivera, A. Stenman, W. Foslien and C. Hrenya (1999). Multi-level pseudo-random signal design and just-in-time estimation applied to nonlinear identification of a RTP wafer reactor. To be presented at the 1999 American Control Conference, San Diego, California.

Stenman, A. and F. Gustafsson (1999). Adaptive smoothing methods for frequency-domain identification. Submitted to Automatica.

Stenman, A. (1999). Model-free predictive control. Submitted to CDC '99.

2

Parametric Methods

This chapter gives a brief review of parametric estimation methods, which quite often are considered when solving the regression problem described in Chapter 1. The basic concept of parametric regression methods is given in Section 2.1. Section 2.2 gives some examples of common black-box models, both linear and nonlinear, that are frequently used in system identification. Section 2.3 describes the two basic parameter estimation methods used when a certain model class is chosen. Section 2.4, finally, briefly states the basic asymptotic properties associated with parametric models.

2.1 Parametric Regression Models

A very commonly used way of estimating the regression function m in a regression relationship

$$Y_i = m(X_i) + \epsilon_i, \tag{2.1}$$

on basis of observed data, is the *parametric* approach. The basic assumption is that m belongs to a family of functions with a pre-specified functional form, and that this family can be parameterized by a finite-dimensional parameter vector θ,

$$m(X_i, \theta). \tag{2.2}$$

One of the simplest examples, which is very commonly used, is the *linear regression*,

$$m(X_i, \theta) = m(X_i, \alpha, \beta) = \alpha + X_i^T \beta, \tag{2.3}$$

where it is assumed that the relation between the variables can be described by a hyperplane, whose offset and slope are controlled by the parameters α and β. In the general case, though, a wide range of different nonlinear model structures is possible. The choice of parameterization depends very much on the situation. Sometimes there are physical reasons for modeling Y as a particular function of X, while at other times the choice is based on previous experience with similar datasets.

Once a particular model structure is chosen, the parameter vector θ can naturally be assessed by means of the fit between the model and the data set

$$\sum_i \ell\left(Y_i - m(X_i, \theta)\right), \tag{2.4}$$

using a suitable scalar-valued and positive norm $\ell(\cdot)$. As will be described in Section 2.3, this fit can be performed in essentially two different ways, depending on which norm that is used and how the parameter vector appears in the parameterization. When the parameters enter linearly as in (2.3), they can be easily computed using simple least-squares methods. In general though, this optimization problem is non-convex and may have a number of local minima which makes its solution difficult.

An advantage with parametric models is that they give a very compact description of the data set once the parameter vector θ is estimated. In some applications, the data set may occupy several megabytes while the model is represented by only a handful of parameters. A major drawback, however, is the particular parameterization that must be imposed. Sometimes the assumed function family might be too restrictive or too low-dimensional to fit unexpected features in the data.

2.2 Parametric Models in System Identification

System identification is a special case of the regression relationship (2.1) where the measured variables are related to inputs $u(t)$ and outputs $y(t)$ of dynamical systems. The systems are almost exclusively considered to be represented in *discrete* time (i.e., sampled) since it is the most natural way of collecting data.

There are mainly two different approaches for modeling dynamical discrete-time systems:

The *state-space* approach assumes that it possible to model the plant as a system of first order difference equations,

$$x(t+1) = g(x(t), u(t), w(t)) \tag{2.5a}$$
$$y(t) = h(x(t), u(t)) + \epsilon(t) \tag{2.5b}$$

where $x(t)$ denotes the *state* vector that sums up all the dynamical information, and $w(t)$ and $\epsilon(t)$ denote process and measurement noise, respectively.

The *input-output* approach assumes that the system can be modeled as a nonlinear difference equation

$$y(t) = m(\varphi(t)) + v(t) \tag{2.6}$$

where the *regression vector*

$$\varphi(t) = \varphi(Z^{t-1}) \tag{2.7}$$

is constructed from past measurements,

$$Z^N \triangleq \{(u(1), y(1)), \ldots, (u(N), y(N))\},$$

and $v(t)$ is measurement noise (which not necessarily has to be white). The two approaches are closely related, since the existence of a input-output description like (2.6) follows from observability of (2.5a) and (2.5b) in a neighborhood of the origin of the state space (Narendra and Li, 1996). In this thesis, though, we shall mainly concentrate on the input-output description. Over the years, different names and concepts have been proposed, associated with different parameterizations of the function m and different noise models. We will in the following two subsections briefly describe some of the most commonly used ones.

2.2.1 Linear Black-box Models

Linear black-box models have been thoroughly discussed and analyzed in the system identification literature during the last decades, see for example Ljung (1999) and Söderström and Stoica (1989). We will here just give a very brief introduction to the subject.

A general linear discrete-time system with an additive noise term $v(t)$ can be described by the relation

$$y(t) = G(q)u(t) + v(t) \tag{2.8}$$

where $u(t)$ is the input signal and $G(q)$ is the *transfer function* from input to output $y(t)$. The noise term $v(t)$ can in general terms be characterized by its *spectrum*, i.e., a description of its frequency components, but it is usually more convenient to describe $v(t)$ as being generated as white noise $\{\epsilon(t)\}$ filtered through a linear filter $H(q)$,

$$v(t) = H(q)\epsilon(t). \tag{2.9}$$

Now, if the transfer functions $G(q)$ and $H(q)$ in (2.8) and (2.9) are unknown, it is natural to introduce parameters θ in their descriptions that reflects the lack of knowledge. A general such parameterization is the black-box family (Ljung, 1999),

$$\boxed{A(q^{-1})y(t) = q^{-n_k}\frac{B(q^{-1})}{F(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}\epsilon(t),} \tag{2.10}$$

where n_k is the time delay from $u(t)$ to $y(t)$, and

$$\begin{aligned}
A(q^{-1}) &= 1 + a_1q^{-1} + \ldots + a_{n_a}q^{-n_a} \\
B(q^{-1}) &= b_1 + b_2q^{-1} + \ldots + b_{n_b}q^{-n_b+1} \\
C(q^{-1}) &= 1 + c_1q^{-1} + \ldots + c_{n_c}q^{-n_c} \\
D(q^{-1}) &= 1 + d_1q^{-1} + \ldots + d_{n_d}q^{-n_d} \\
F(q^{-1}) &= 1 + f_1q^{-1} + \ldots + f_{n_f}q^{-n_f}
\end{aligned}$$

are polynomials in the backshift operator q^{-1} with property $q^{-1}y(t) = y(t-1)$. This results in the parameter vector

$$\theta = \left(a_1, \ldots, a_{n_a}, \ldots, f_1, \ldots, f_{n_f}\right)^T. \tag{2.11}$$

Well-known and commonly used special cases of the structure (2.10) include the FIR (Finite Impulse Response) model

$$y(t) = b_1 u(t-n_k) + \ldots + b_{n_b} u(t-n_b-n_k+1) + \epsilon(t), \tag{2.12}$$

which corresponds to $A(q^{-1}) = C(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$, and the ARX (Auto Regressive with eXogeneous input) model,

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) & \\ = b_1 u(t-n_k) + \ldots + b_{n_b} u(t-n_b-n_k+1) + \epsilon(t) & \end{aligned} \tag{2.13}$$

which corresponds to $C(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$. Since both these structures only depend on measured quantities, they have the nice property of being expressible in terms of a *linear regression*,

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

and hence the parameter vector θ can be determined using simple and powerful estimation methods (e.g., least squares), see Section 2.3.1.

Other special cases of the general structure (2.10) include the named structures

$$\begin{aligned} \text{AR:} \quad & B(q^{-1}) = 0, \quad C(q^{-1}) = D(q^{-1}) = 1 \\ \text{ARMA:} \quad & B(q^{-1}) = 0, \quad D(q^{-1}) = 1 \\ \text{ARMAX:} \quad & D(q^{-1}) = F(q^{-1}) = 1 \\ \text{OE:} \quad & A(q^{-1}) = C(q^{-1}) = D(q^{-1}) = 1 \\ \text{BJ:} \quad & A(q^{-1}) = 1 \end{aligned}$$

where the first two are used for time series modeling only. Here the acronyms denote Auto Regressive, Auto Regressive Moving Average, Auto Regressive Moving Average with eXogeneous input, Output-Error and Box-Jenkins, respectively. As shown by Ljung and Wahlberg (1992), the ARX model (2.13) is capable of describing any linear system provided sufficiently large model orders are allowed. The reason for considering other more complex models, like the ARMAX or BJ models, is that they provide more flexible parameterizations so that an equally good fit can be obtained using fewer parameters.

In general, the predictor associated with (2.10) can at least be written in *pseudo-linear regression* form

$$\hat{y}(t|\theta) = \varphi^T(t,\theta)\theta. \tag{2.14}$$

Some of the regressors, i.e., the components of $\varphi(t,\theta)$, may in this case be dependent of previous model outputs, and are given by

- Outputs, $y(t-k)$, associated with the A-polynomial.

- Inputs, $u(t-k)$, associated with the B-polynomial.
- Prediction errors, $\varepsilon(t-k) = y(t-k) - \hat{y}(t-k|\theta)$, associated with the C-polynomial.
- Simulated outputs from past inputs, $\hat{y}_u(t-k|\theta)$, associated with the F-polynomial.
- Simulation errors, $\varepsilon_u(t-k) = y(t-k) - \hat{y}_u(t-k|\theta)$, associated with the D-polynomial.

2.2.2 Nonlinear Black-box Models

When turning into nonlinear modeling, things in general become much more complicated. The reason for this is that almost nothing is excluded, and that a wide range of possible model parameterizations exists. To reduce the complexity though, it is natural to think of the parameterization (2.2) as a function expansion (Sjöberg *et al.*, 1995),

$$m(\varphi(t), \theta) = \sum_{k=1}^{r} \alpha_k g_k(\varphi(t), \beta_k, \gamma_k). \tag{2.15}$$

The functions $g_k(\cdot)$ are usually referred to as *basis functions*, because the role they play in (2.15) is very similar to that of a functional space basis in mathematical sense. Typically, the basis functions are constructed from a simple scalar "mother" basis function, $\kappa(\cdot)$, which is scaled and translated according to the parameters β_k and γ_k.

Using scalar basis functions, there are essentially three basic methods of expanding them into higher regressor dimensions:

Ridge construction. A ridge basis function has the form

$$g_k(\varphi(t), \beta_k, \gamma_k) = \kappa(\beta_k^T \varphi(t) + \gamma_k), \tag{2.16}$$

where $\kappa(\cdot)$ is a scalar basis function, $\beta_k \in \mathbb{R}^n$ and $\gamma_k \in \mathbb{R}$. The ridge function is constant for all $\varphi(t)$ in the direction where $\beta_k^T \varphi(t)$ is constant. Hence the basis functions will have unbounded support in this subspace, although the mother basis function $\kappa(\cdot)$ has local support. See Figure 2.1 (a).

Radial construction. In contrast to the ridge construction, the radial basis functions have true local support as is illustrated in Figure 2.1 (b). The radial support can be obtained using basis functions of the form

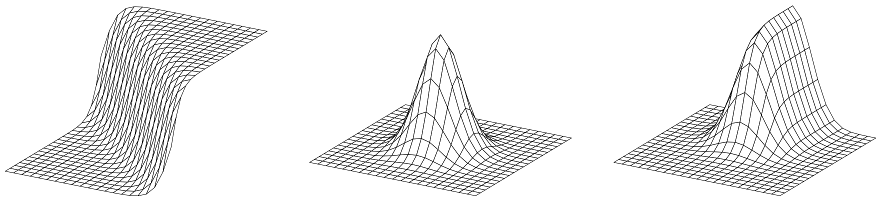
$$g_k(\varphi(t), \beta_k, \gamma_k) = \kappa(\|\varphi(t) - \gamma_k\|_{\beta_k}), \tag{2.17}$$

where $\gamma_k \in \mathbb{R}^n$ is a center point and $\|\cdot\|_{\beta_k}$ denotes a scaled vector norm on the regressor space. The scaling is often taken as a diagonal matrix.

Composition. A composition is obtained when the ridge and radial constructions are combined when forming the basis functions. A typical example is illustrated in Figure 2.1 (c). In general, the composition can be written as a tensor product

$$g_k(\varphi(t), \beta_k, \gamma_k) = g_{k,1}\big(\varphi_1(t), \beta_{k,1}, \gamma_{k,1}\big) \times \cdots \times g_{k,r}\big(\varphi_r(t), \beta_{k,r}, \gamma_{k,r}\big), \tag{2.18}$$

where each $g_{k,i}(\cdot)$ is either a ridge or a radial function.



(a) Ridge (b) Radial (c) Composition

Figure 2.1 *Three different methods of expanding into higher regressor dimensions.*

Using the function expansion (2.15) and the different basis function constructions (2.16)-(2.18), a number of well-known nonlinear model structures can be formed – for example *neural networks*, *radial basis function networks* and *wavelets*.

Neural Networks

The combination of (2.15), the ridge construction (2.16), and the so-called *sigmoid* mother basis function,

$$\kappa(x) = \sigma(x) \triangleq \frac{1}{1 + \exp(-x)}, \tag{2.19}$$

results in the celebrated *one hidden layer feedforward neural net*. See Figure 2.2. Many different generalizations of this basic structure is possible. If the outputs of the $\kappa(\cdot)$ blocks are weighted, summed and fed through a new layer of $\kappa(\cdot)$ blocks, one usually talks about *multi-layer* feedforward neural nets. So-called *recurrent* neural networks are obtained if instead some of the internal signals in the network are fed back to the input layer. See (Haykin, 1994) for further structural issues. Neural network models are highly nonlinear in the parameters, and have thus to be estimated through numerical optimization schemes as will be described in Section 2.3.2. Modeling of dynamical systems using neural nets have been extensively studied in for instance Sjöberg (1995) and Sjöberg *et al.* (1995).

Radial Basis Networks

A closely related concept to the neural net approach is the *radial basis function (RBF) network* (Chen *et al.*, 1990). It is constructed using the expansion (2.15) and the radial construction (2.17). The radial mother basis function $\kappa(\cdot)$ is often taken as a Gaussian function

$$\kappa(x) = \exp\left(-x^2\right).$$

Compared to neural networks, the RBF network has the advantage of being linear in the parameters (provided that the location parameters are fixed). This makes the estimation process easier.

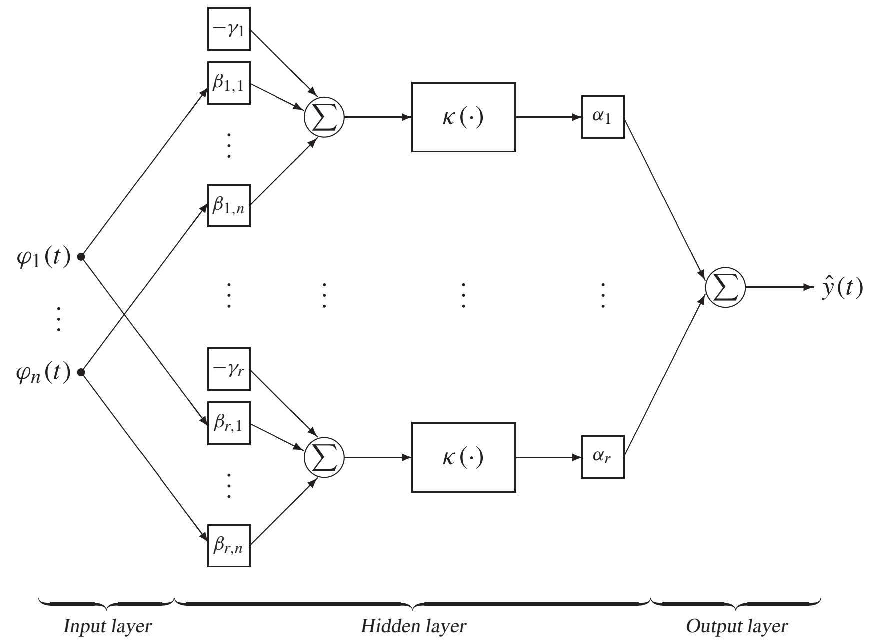


Figure 2.2 *A one hidden layer feedforward neural net.*

Wavelets

Wavelet decomposition of a function is another example of the parameterization (2.15) (Daubechies, 1990). A mother basis function (usually referred to as the *mother wavelet* and denoted by $\psi(\cdot)$ rather than $\kappa(\cdot)$) is scaled and translated to form a wavelet basis. The mother wavelet is usually a small wave (a pulse) with bounded support.

It is common to let the expansion (2.15) be double indexed according to scale and location. For the scalar case and the specific choices $\beta_j = 2^j$ and $\gamma_k = k$, the basis functions can be written as

$$g_{j,k} = 2^{j/2}\kappa(2^j\varphi(t) - k). \tag{2.20}$$

Multivariable wavelet functions can be constructed from scalar ones using the composition method (2.18).

Wavelets have *multi-resolution* capabilities. Several different scale parameters are used simultaneously and overlappingly. With a suitable chosen mother wavelet along with scaling and translation parameters, the wavelet basis can be made orthonormal, which makes it easy to compute the coordinates $\alpha_{j,k}$ in (2.15). See for example Sjöberg *et al.* (1995) for details.

Regressors for Nonlinear Black-box Models

As in the linear case we have the same freedom in selecting the regressors. Following the nomenclature for the linear black-box models, it is natural to use similar names for nonlinear model structures. According to Sjöberg *et al.* (1995), we can distinguish between:

- NAR models, which use $y(t-k)$ as regressors.
- NFIR models, which use $u(t-k)$ as regressors.
- NARX models, which use $u(t-k)$ and $y(t-k)$ as regressors.
- NARMAX models, which use $u(t-k)$, $y(t-k)$ and $\varepsilon(t-k|\theta)$ as regressors.
- NOE models, which use $u(t-k)$ and $\hat{y}_u(t-k|\theta)$ as regressors.
- NBJ models, which use $u(t-k)$, $\hat{y}(t-k|\theta)$, $\varepsilon(t-k|\theta)$ and $\varepsilon_u(t-k|\theta)$ as regressors.

The last three structures NARMAX, NOE and NBJ use regressors which consist of past outputs from the model. This is usually referred to as *recurrency* and can be interpreted as an extension of the pseudo-linear regression concept (2.14). Recurrent structures are in general much harder to work with than those structures whose regressors only consist of measured data.

2.3 Parameter Estimation

When a particular model structure, linear or nonlinear, is chosen, the next step is to estimate the parameters on the basis of the observations Z^N. This is usually done by minimizing the "size" of the prediction errors,

$$V_N(\theta, Z^N) \triangleq \frac{1}{N}\sum_{i=1}^{N} \ell(Y_i - m(X_i, \theta)), \tag{2.21}$$

where the norm typically is taken as the quadratic L_2 norm, i.e., $\ell(\varepsilon) = \varepsilon^2$. The parameter estimate is then given by the minimizing argument of (2.21). That is,

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta, Z^N). \tag{2.22}$$

This is a very well established problem formulation known as *prediction error minimization* (PEM) methods. Depending on how the parameters appear in the parameterization, this minimization can be performed either using a linear least squares approach or a nonlinear least squares approach.

2.3.1 Linear Least Squares

When the predictor is linear in the parameters and a quadratic norm is used, an explicit solution that minimizes (2.21) exists. The optimal parameter estimate is then simply given by the ordinary least squares solution

$$\hat{\theta}_N = \left(\frac{1}{N} \sum_{i=1}^{N} \mathcal{B}(X_i)\mathcal{B}^T(X_i) \right)^{-1} \frac{1}{N} \sum_{i=1}^{N} \mathcal{B}(X_i)Y_i, \tag{2.23}$$

provided that the inverse exists. Here, and in the sequel of the thesis, $\mathcal{B}(X_i)$ denotes the vector of basis functions associated with the parameterization. For instance, for the simple example (2.3) we have

$$\mathcal{B}(X_i) = \begin{pmatrix} 1 \\ X_i \end{pmatrix}.$$

For numerical reasons the inverse in (2.23) is rarely formed. Instead the estimate is computed using QR- or singular value decomposition (Björck, 1996).

2.3.2 Nonlinear Optimization

When the predictor is nonlinear in the parameters, the minimum of the loss function (2.21) can normally not be computed analytically. Instead one has to resort to methods that search for the minimum numerically. A commonly used such optimization method is *Newton's algorithm* (Dennis and Schnabel, 1983),

$$\hat{\theta}_N^{(k+1)} = \hat{\theta}_N^{(k)} - \mu \left[V_N''(\hat{\theta}_N^{(k)}, Z^N) \right]^{-1} V_N'(\hat{\theta}_N^{(k)}, Z^N), \tag{2.24}$$

in which $V_N'(\cdot)$ and $V_N''(\cdot)$ denote the gradient and the Hessian of the loss function (2.21) with respect to θ. This is an iterative local search procedure where the parameter vector estimate in each iteration is updated in the negative gradient direction with a step size related to the inverse of the Hessian.

A problem though with numerical search methods like (2.24) is that they usually run into problems for highly nonlinear structures whose loss functions have several local minima. There are in other words no guarantees that the parameter estimate converges to the global minimum of (2.21). There is no easy solution to this problem, so the choice of initial value $\hat{\theta}_N^{(0)}$, i.e., where to start the search is crucial. Otherwise only various global search strategies remain, such as random search, random restarts and genetic algorithms.

2.4 Asymptotic Properties of the Parametric Model

An interesting question is what properties the estimate resulting from (2.21) will have. These will naturally depend on the properties of the data set Z^N. In general it is a difficult problem to characterize the quality of $\hat{\theta}_N$ exactly. Instead one normally investigates the asymptotic properties of $\hat{\theta}_N$ as the number of data, N, tends to infinity.

It is an important aspect of the general parameter estimation method (2.21) that the asymptotic properties of the resulting estimate can be expressed in general terms for arbitrary model structures.

The first basic result is the following;

$$\hat{\theta}_N \to \theta^* \quad \text{as} \quad N \to \infty, \tag{2.25}$$

where

$$\theta^* = \arg\min_{\theta} \mathrm{E}(Y_i - m(X_i, \theta))^2. \tag{2.26}$$

That is, as more and more data become available, the estimate converges to the value θ^*, that would minimize the expected value of the squared prediction errors. This is in a sense *the best possible approximation* of the true regression function that is available within the model structure. The expectation E in (2.26) is taken with respect to all random disturbances that affect the data and it also includes averaging over the predictor variables.

The second basic result is the following one: If the prediction error

$$\varepsilon_i(\theta^*) = Y_i - m(X_i, \theta^*)$$

is approximately white noise, then the covariance matrix of $\hat{\theta}_N$ is approximately given by

$$\mathrm{E}(\hat{\theta}_N - \theta^*)(\hat{\theta}_N - \theta^*)^T \sim \frac{\sigma^2}{N}\left[\mathrm{E}\,\psi_i \psi_i^T\right]^{-1}, \tag{2.27}$$

where

$$\sigma^2 = \mathrm{E}\,\varepsilon_i^2(\theta^*) \tag{2.28}$$

and

$$\psi_i = \frac{d}{d\theta} m(X_i, \theta)\bigg|_{\theta=\theta^*}. \tag{2.29}$$

That is, the covariance of $\hat{\theta}_N$ decays as N^{-1}.

The results (2.25) through (2.29) are general and hold for all model structures, both linear and non-linear ones, subject only to some regularity and smoothness conditions. See Ljung (1999) for more details around this.

3

Nonparametric Methods

In Chapter 2, a brief review of parametric estimation methods was given. It was concluded that parametric methods offer good modeling alternatives, since a parametric model provides a high degree of data compression, i.e., the major features of the data are condensed into a few number of parameters. A problem with the approach, though, is the requirement of certain parameterizations. Those must be selected so that they match the properties of the underlying regression function. Otherwise quite poor results are often obtained.

The problem with parametric regression models mentioned above can be solved by removing the restriction that the regression function belongs to a parametric function family. This leads to an approach which is usually referred to as *nonparametric regression*. The basic idea behind nonparametric methods is that one should let the data decide which function fits them best without the restrictions imposed by a parametric model.

Local nonparametric regression models have been discussed and analyzed in the statistical literature for a long time. In the context of so-called kernel regression methods, traditional approaches have involved the *Nadaraya-Watson estimator* (Nadaraya, 1964; Watson, 1964) and some alternative kernel estimators, for example the *Priestly-Chao* estimator (Priestly and Chao, 1972) and the *Gasser-Müller* estimator (Gasser and Müller, 1979). In this chapter we give a brief introduction to a special class of such models, *local polynomial estimators* (Stone, 1977; Cleveland, 1979; Müller, 1987; Fan, 1992). These estimate the regression function at a certain point by locally fitting a polynomial of degree p to the data using weighted regression. The Nadaraya-Watson estimator can be seen as a special case in this framework since it corresponds to fitting a zero degree polynomial, i.e., a local constant, to data.

The presentation here is neither formal nor complete. The purpose is just to introduce concepts and notation used in the statistical literature. More comprehensive treatments of the topic are given in the books by Härdle (1990), Wand and Jones (1995) and Fan and Gijbels (1996) upon which this survey is based. Other interesting treatments of the subject (including descriptions of existing software packages) are given in Cleveland and Grosse (1991), Cleveland *et al.* (1992), Cleveland and Loader (1994), Loader (1995) and Loader (1997).

The outline of the chapter is as follows: Sections 3.1 and 3.2 define the basic smoothing problem and discuss how it can be solved using the concept of local polynomial estimation. Section 3.3 summarizes several statistical aspects of local polynomial estimators such as bias and variance errors as well as asymptotical properties. Section 3.4 reviews possible solutions to the important problem of bandwidth selection, and Section 3.5 extends the local polynomial ideas to the local likelihood framework. Sections 3.6 and 3.7 provide some notes on the selection of the polynomial degree and adaptive smoothing methods. Section 3.8, finally, extends and generalizes the local polynomial ideas to the multivariable case.

3.1 The Smoothing Problem

Smoothing of a noisy data set $\{(Y_i, X_i)\}_{i=1}^N$ concerns the problem of estimating the function m in the regression relationship

$$Y_i = m(X_i) + \epsilon_i, \qquad i = 1, \ldots, N, \tag{3.1}$$

without the imposition that m belongs to a parametric family of functions. Depending on how the data have been collected, several alternatives exist. If there are multiple observations at a certain point x, an estimate of $m(x)$ can be obtained by just taking the average of the corresponding Y-values. In most cases however, repeated observations at a given x are not available, and one has to resort to other solutions that deduce the value of $m(x)$ using observations at other positions than x. In the trivial case where the regression function $m(\cdot)$ is constant, estimation of $m(x)$ reduces to taking the average over the response variables Y. In general situations, though, it is unlikely that the true regression curve is constant. Rather, the assumed function is modeled as a smooth continuous function which is nearly constant in a small neighborhood around x. A natural approach is therefore some sort of mean of the response variables near the point x. This *local average* should then be constructed so that it is defined only from observations in a small neighborhood around x. This local averaging can be seen as the basic idea of *smoothing*.

Almost all smoothing methods can, at least asymptotically, be described as a weighted average of the Y_i's near x,

$$\hat{m}(x) = \sum_{i=1}^{N} W_i(x) Y_i, \tag{3.2}$$

where $\{W_i(x)\}$ is a sequence of weights that may depend on x and the predictor data $\{X_i\}$. An estimator of the type (3.2) is usually called a *linear smoother* and the result of a smoothing operation is sometimes called a *smooth* (Tukey, 1977). A simple estimate can be obtained by defining the weights as constant over adjacent intervals. In more sophisticated methods like kernel estimation and local polynomial regression, the weights are chosen according to a kernel function $K_h(\cdot)$ of fixed form. Such methods will be described more in detail in the following section.

The fact that smoothers, by definition, average over observations with considerably different expected values has been paid special attention in the statistical literature. The weights W_i are typically tuned by a *smoothing parameter* (bandwidth) that controls the degree of local averaging, i.e., the size of the neighborhood around x. A too large neighborhood will include observations located far away from x, whose expected values may differ considerably from $m(x)$, and as a result the estimator will produce an "over-smoothed" or *biased* estimate. On the other hand, when using a too small neighborhood, only a few number of observations will contribute to the estimate at x, hence making it "under-smoothed" or noisy. The basic problem in nonparametric methods is thus to find the optimal choice of smoothing parameter that will balance the bias error against the variance error.

Before going into details about local regression models, we will give some basic terminology and notation. Nonparametric regression is studied in both *fixed design* and *random design* contexts. In the fixed design case, the predictor variables consist of ordered non-random numbers. A special case is the *equally spaced fixed design* where the difference $X_{i+1} - X_i$ is constant for all i, for example $X_i = i/N$, $i = 1, \ldots, N$. The random design occurs when the predictor variables instead are independent, identically distributed random variables. The regression relationship is in both cases assumed to be modeled as in (3.1), where ϵ_i are independent random variables with zero means and variances σ^2, which are independent of $\{X_i\}$. The overview is concentrated on the scalar case, because of its simpler notation. However, the results are generalized to the multivariable case in Section 3.8.

3.2 Local Polynomial Models

Local polynomial estimators form a special class of nonparametric regression models which was among first discussed by Stone (1977), Cleveland (1979) and Katkovnik (1979). The basic idea is to estimate the regression function $m(\cdot)$ (or its derivatives) in (3.1) at a particular point x, by locally fitting a pth degree polynomial

$$\beta_0 + \beta_1(X_i - x) + \ldots + \beta_p(X_i - x)^p \tag{3.3}$$

to the data belonging to a neighborhood around x. The polynomial is fitted as a weighted regression problem: Minimize

$$V_x(\beta) = \sum_{i=1}^{N} \ell\left(Y_i - \sum_{j=0}^{p} \beta_j (X_i - x)^j \right) K_h(X_i - x), \tag{3.4}$$

where $\ell(\cdot)$ is a scalar-valued, positive norm function, h is a *bandwidth* parameter controlling the size of the local neighborhood, and

$$K_h(\cdot) \triangleq h^{-1} K(\cdot/h)\,, \tag{3.5}$$

with $K(\cdot)$ a *kernel* function assigning weights to each remote data point according to its distance from x. If $\hat{\beta}_j,\ j = 0, \dots, p$, denote the minimizers of (3.4), an estimate of the derivative $m^{(\nu)}(x)$ is given by

$$\hat{m}^{(\nu)}(x, h) = \nu!\hat{\beta}_\nu. \tag{3.6}$$

However, our main interest in this thesis will be to estimate the regression function itself, that is,

$$\hat{m}(x, h) = \hat{\beta}_0.$$

The norm $\ell(\cdot)$ is usually taken as the quadratic L_2 norm, $\ell(\varepsilon) = \varepsilon^2$, which is convenient both for computation and analysis. However, demands for robustness against outliers may sometimes warrant other choices. Unless otherwise stated though, we shall in the sequel of the thesis assume the quadratic norm as the default choice.

3.2.1 Kernel Functions

The kernel is normally chosen as a function with probability density properties satisfying

$$\int K(u)\,du = 1, \qquad \int uK(u)\,du = 0. \tag{3.7}$$

A wide range of different kernel functions is possible in general, but often both practical and performance theoretic considerations limit the choice. Commonly used kernels include the Gaussian kernel $K(u) = (\sqrt{2\pi})^{-1} \exp(-u^2/2)$ and the "symmetric Beta family" (Fan and Gijbels, 1996);

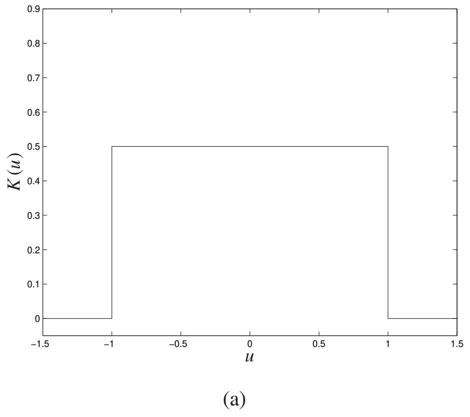
$$K(u) = \frac{1}{\mathrm{Beta}(1/2, \gamma + 1)} \left(1 - u^2\right)_+^\gamma, \qquad \gamma = 0, 1, 2, \dots$$

where $(\cdot)_+$ denotes the positive part. Here $\gamma = 0$ corresponds to the *uniform* (or boxcar) kernel, see Figure 3.1 (a). For the choice $\gamma = 1$ the kernel becomes

$$K(u) = \frac{3}{4}\left(1 - u^2\right)_+. \tag{3.8}$$

This so-called *Epanechnikov* kernel (Epanechnikov, 1969), which has been shown to be optimal in asymptotic mean square error sense (see Section 3.3.5), is depicted in Figure 3.1 (b). An undesirable property of the Beta family kernels in general and the uniform kernel in particular, though, is that their derivatives are discontinuous. This problem has been acknowledged for a long time in the domains of signal processing and spectral analysis (Ljung, 1999; Stoica and Moses, 1997). The uniform kernel has a Fourier transform with large side lobes, which leads to that high frequency components in the data, i.e., noise, can leak into the resulting estimate. A remedy is to use a kernel that smoothly descends to zero. One such example is the *tricube* kernel,

$$K(u) = \frac{70}{81}\left(1 - |u|^3\right)_+^3, \tag{3.9}$$



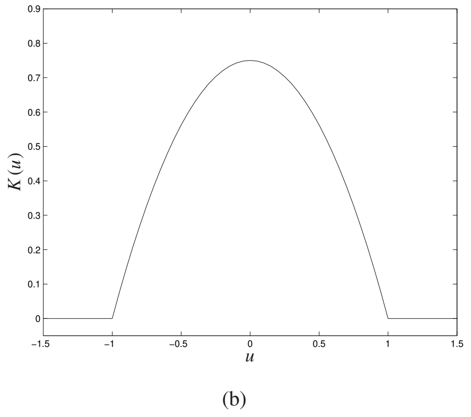


Figure 3.1 *(a) The uniform (boxcar) kernel. (b) The Epanechnikov kernel.*

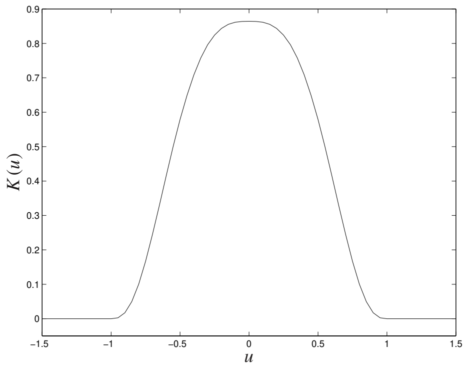


Figure 3.2 *The tricube kernel.*

which is the default choice in the LOWESS and LOESS robust fitting procedures (Cleveland, 1979; Cleveland and Devlin, 1988) and in the LOCFIT package (Loader, 1997). A plot of this kernel is shown in Figure 3.2.

It might be worth pointing out that the constant factors in the kernel functions do not actually affect the local fit – they are merely normalization constants so that the kernels become density functions, and can be ignored in practical implementations.

3.2.2 The Bandwidth

The selection of the bandwidth parameter is crucial for the performance of the estimator, since it governs a trade-off between the bias and variance errors. This fact is best illustrated with an example, which is adapted from Wand and Jones (1995).

Example 3.1 The role of the bandwidth

Consider the regression function

$$m(x) = 3\exp\left(-x^2/0.3^2\right) + 2\exp\left(-(x-1)^2/0.7^2\right), \qquad x \in [0, 1], \tag{3.10}$$

which is represented by the dashed curve in Figure 3.3 (a)–(c). A data set (represented by circles) was generated according to

$$Y_i = m(X_i) + \epsilon_i, \qquad i = 1, \dots, 100 \tag{3.11}$$

where $X_i = i/100$, and ϵ_i are independent Gaussian random variables with zero means and variances 0.16. The solid lines represent local polynomial estimates with $p = 1$ using Epanechnikov kernels and bandwidths 0.011, 1, and 0.15 respectively. ❑

As shown in Example 3.1, if the bandwidth h is small, the local polynomial fitting depends heavily on the measurements that are close to x, thus producing an estimate that is very noisy and wiggly. This is shown in Figure 3.3 (a) where the bandwidth $h = 0.011$ is used. A large bandwidth, on the other hand, tends to weight the measurements more equally, hence resulting in an estimate that approaches a straight line through the data, i.e., a global linear fit. This is illustrated in Figure 3.3 (b), where a bandwidth $h = 1$ is used. A good choice of bandwidth is a compromise between these two extremes, as shown in Figure 3.3 (c) where a bandwidth $h = 0.15$ is used.

In practice, it is undesirable to have to specify the bandwidth explicitly when performing the local polynomial fit. A better idea is to make use of the available data set, and on basis on these data *automatically* estimate a good bandwidth. This procedure is usually referred to as *(data-driven) bandwidth selection*, and will be described more detailed in Section 3.4.

3.2.3 Computing the Estimate

In order to simplify the notation in the sequel of the chapter, it is convenient to work with the more compact matrix notation. Let

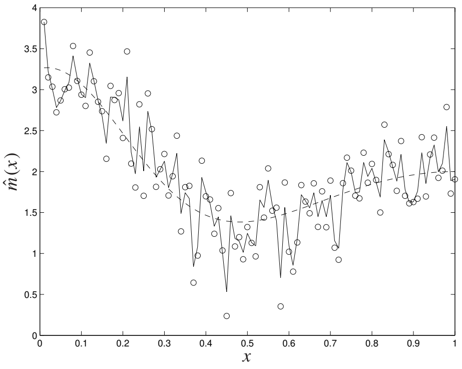
$$\mathbf{y} \triangleq \begin{pmatrix} Y_1 & \dots & Y_N \end{pmatrix}^T, \qquad \beta \triangleq \begin{pmatrix} \beta_0 & \beta_1 & \dots & \beta_p \end{pmatrix}^T,$$

and introduce the Vandermonde type *design matrix* of size $N \times (p+1)$,

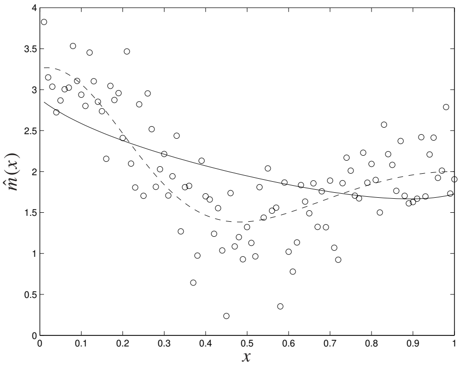
$$\mathbf{X} \triangleq \begin{pmatrix} 1 & (X_1 - x) & \cdots & (X_1 - x)^p \\ \vdots & \vdots & & \vdots \\ 1 & (X_N - x) & \cdots & (X_N - x)^p \end{pmatrix} = \begin{pmatrix} \mathcal{B}^T(X_1 - x) \\ \vdots \\ \mathcal{B}^T(X_N - x) \end{pmatrix}.$$

Further, let $\mathbf{W}$ denote the $N \times N$ weight matrix,

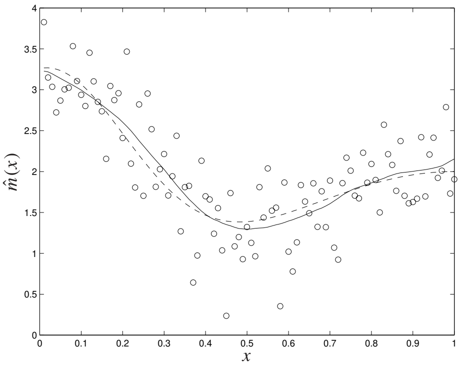
$$\mathbf{W} \triangleq \operatorname{diag}(K_h(X_1 - x), \dots, K_h(X_N - x)).$$



(a) $h = 0.011$



(b) $h = 1$



(c) $h = 0.15$

Figure 3.3 *Local linear estimates (solid) of the regression function (3.10) based on the dataset (3.11) (circles) using different bandwidths. The dashed curve represents the true regression function.*

Then, using the quadratic L_2-norm, $\ell(\varepsilon) = \varepsilon^2$, the local polynomial fit (3.4) can be rewritten in matrix form as

$$\hat{\beta} = \arg\min_{\beta} \, (\mathbf{y} - \mathbf{X}\beta)^T \mathbf{W} (\mathbf{y} - \mathbf{X}\beta). \tag{3.12}$$

Ordinary weighted least squares theory hence gives the solution

$$\hat{\beta} = \left(\mathbf{X}^T \mathbf{W} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}, \tag{3.13}$$

provided the information matrix $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is non-singular.

The expression for $\hat{m}(x, h) = \hat{\beta}_0$ in (3.13) is quite complicated in general, but for the special cases $p = 0$ and $p = 1$, simple explicit formulas exist. With $p = 0$, i.e., when fitting a local constant to data, the estimator becomes

$$\hat{m}(x, h) = \frac{\sum_{i=1}^{N} K_h(X_i - x)\, Y_i}{\sum_{i=1}^{N} K_h(X_i - x)}, \tag{3.14}$$

which is widely known as the *Nadaraya-Watson* kernel estimator (Nadaraya, 1964; Watson, 1964). With $p = 1$, the estimator is termed *local linear* estimator (Fan, 1992), and can be explicitly expressed as

$$\hat{m}(x, h) = \frac{1}{N} \sum_{i=1}^{N} \frac{\{s_2(x, h) - s_1(x, h)(X_i - x)\}\, K_h(X_i - x)\, Y_i}{s_2(x, h) s_0(x, h) - s_1^2(x, h)}, \tag{3.15}$$

where

$$s_j(x, h) \triangleq \frac{1}{N} \sum_{i=1}^{N} (X_i - x)^j K_h(X_i - x). \tag{3.16}$$

The Nadaraya-Watson estimator has been extensively used in the past. Nowadays it has to some extent fallen into disuse, mainly because of its (as we shall see later) bad boundary properties.

3.2.4 Nearest Neighbor Estimators

The *k-nearest neighbor (k-NN)* estimator is a variant of the kernel estimator for local constant fitting. The *k-NN* weight sequence has been introduced by Loftsgaarden and Quesenberry (1965) in the related density estimation problem, and has been used by Cover and Hart (1967) for classification purposes. The *k-NN* estimator is defined as

$$\hat{m}_k(x) = \sum_{i=1}^{N} W_{i,k}(x) Y_i, \tag{3.17}$$

where the weights are defined through the *index set*,

$$\Omega_k(x) = \{i : X_i \text{ is one of the } k \text{ nearest neighbors of } x\}, \tag{3.18}$$

such that

$$W_{i,k}(x) = \begin{cases} 1/k, & \text{if } i \in \Omega_k(x) \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

More generally, the nearest neighbor estimator can be thought of as a local polynomial fitting procedure as in (3.4), with weights being generated by a kernel function

$$W_{i,k}(x) = K_{h_k}(X_i - x), \tag{3.20}$$

where the bandwidth h_k is the distance between x and its kth nearest neighbor,

$$h_k = \max_i |X_i - x|, \qquad i \in \Omega_k(x).$$

In this context, h_k is usually referred to as *nearest neighbor bandwidth*.

Cleveland and Loader (1994) argue in favor of nearest neighbor bandwidths over fixed bandwidths. A fixed bandwidth can have large variance in regions with low data density, and, in extreme cases, empty neighborhoods will lead to undefined estimates. The nearest neighbor bandwidth approach has been taken in the commonly used LOESS package (Cleveland and Devlin, 1988; Cleveland and Grosse, 1991; Cleveland *et al.*, 1992). Its successor LOCFIT (Loader, 1997) allows a mixture of fixed and nearest neighbor components.

3.2.5 Equivalent Weights

As seen from (3.13) the estimate defined by (3.6) is linear in the responses Y_i, and can be written in the linear smoother form (3.2) with weights

$$\mathbf{w}_\nu^T(x) \triangleq (W_1^{(\nu)}(x), \dots, W_N^{(\nu)}(x)) = \nu! e_{\nu+1}^T \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{W}. \tag{3.21}$$

where e_k denotes the kth column of the identity matrix. The weights $W_i^{(\nu)}(x)$ are usually referred to as *equivalent weights* or *weight diagram* (Hastie and Loader, 1993; Fan and Gijbels, 1996). It is easy to show that they satisfy the discrete moment conditions,

$$\sum_{i=1}^{N} (X_i - x)^q W_i^{(\nu)}(x) = \delta_{\nu,q}, \qquad q = 0, \dots, p, \tag{3.22}$$

where $\delta_{i,j}$ denotes the Kronecker delta. (Recall that ν denotes the derivative order of $m(\cdot)$ that we want to determine, but we will usually suppress it as index or superscript in the case $\nu = 0$). Hence it follows that the finite sample bias when estimating polynomials up to order p is zero (Ruppert and Wand, 1994; Fan and Gijbels, 1996).

Figure 3.4 shows examples of equivalent weights for the cases $p = 0$, $p = 1$ and $p = 2$ when estimating the regression function at an interior point of the regressor space using the Epanechnikov kernel and bandwidths $h = 1$. Figure 3.5 illustrates their ability to adapt to the situation at the left boundary. For higher order fits the weights will depend on both the design points and the estimation location (see for instance equation (3.15)), and hence adapt automatically to various design distributions and boundary conditions. Note in particular in Figure 3.5 (b)–(c) how the weights W_i deform in order to reduce the bias. Traditional kernel estimators like the Nadaraya-Watson estimator (3.14), i.e., local constant fits, do not have this property and therefore get larger bias within boundary regions. See Figure 3.5 (a) where the weights W_i have exactly the same shape as for the interior case in Figure 3.4 (a). As a consequence of this bad property they are nowadays seldom used as the default choice for local regression.

The equivalent weights $\mathbf{w}_0(x) = \mathbf{w}(x)$ can be used to define the so-called *hat matrix*

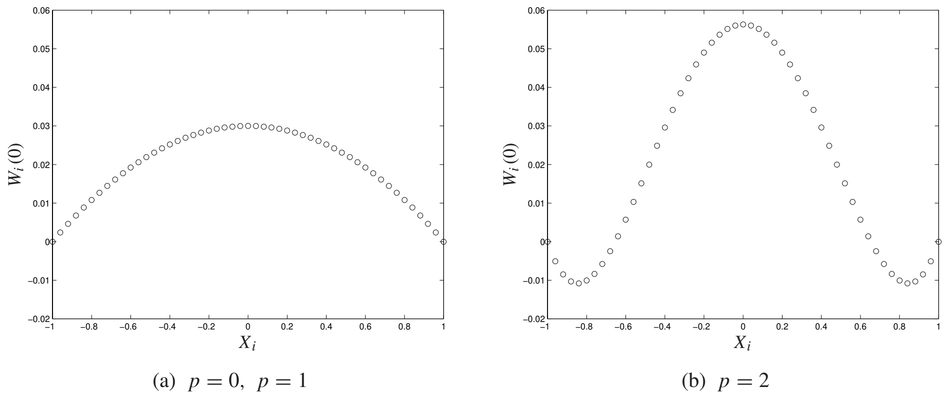


Figure 3.4 *Equivalent weights for different polynomial orders when estimating the regression function at an interior point ($x = 0$) using bandwidth $h = 1$. The local constant and local linear weight sequences coincide when the design is equally spaced and x is located on the grid.*

(or smoothing matrix),

$$\mathbf{H} \triangleq \begin{pmatrix} \mathbf{w}^T(X_1) \\ \vdots \\ \mathbf{w}^T(X_N) \end{pmatrix}, \tag{3.23}$$

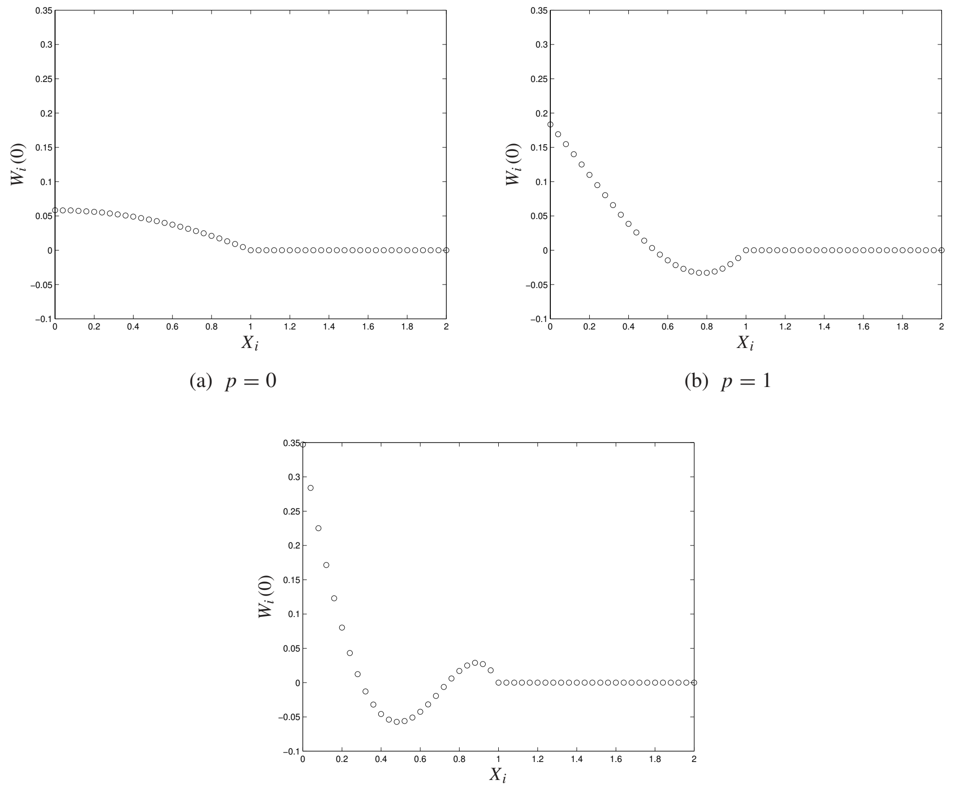
which maps data to fitted values according to

$$\begin{pmatrix} \hat{m}(X_1, h) \\ \vdots \\ \hat{m}(X_N, h) \end{pmatrix} = \begin{pmatrix} \mathbf{w}^T(X_1) \\ \vdots \\ \mathbf{w}^T(X_N) \end{pmatrix} \mathbf{y} = \mathbf{H}\mathbf{y}. \tag{3.24}$$

This definition will be needed later on in this chapter and in Chapter 4.

3.3 Statistical Properties of the Local Fit

It is in general of interest to investigate the performance and the statistical properties of local polynomial estimators. Typically, this concerns questions regarding *consistency*, i.e., whether or not the estimate converges to the true regression function m, and *convergence rate*, i.e., how fast the estimate tends to m with respect to the number of samples N.



(a) $p = 0$ (b) $p = 1$

(c) $p = 2$

Figure 3.5 *Equivalent weights for different polynomial orders when estimating the regression function at a left boundary point ($x = 0$) using bandwidth $h = 1$.*

3.3.1 The MSE and MISE Criteria

When analyzing the performance it is necessary to have some kind of measure that specifies the accuracy of the estimator. An often used pointwise error measure is the *mean square error (MSE)*,

$$\mathrm{MSE}(\hat{m}(x,h)) \overset{\Delta}{=} \mathrm{E}\big(\hat{m}(x,h) - m(x)\big)^2 . \tag{3.25}$$

The MSE has the nice feature of being decomposable into a squared bias part and a variance error part,

$$\mathrm{MSE}(\hat{m}(x,h)) = \underbrace{\big(\mathrm{E}\,\hat{m}(x,h) - m(x)\big)^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}\big(\hat{m}(x,h)\big)}_{\text{variance}} . \tag{3.26}$$

where the variance error typically decreases and the bias error increases with increasing h. As we shall see in Section 3.3.5, this implies that a good choice of bandwidth is one that balances the bias error versus the variance error.

If one instead is interested in a global error measure, it is natural to integrate the squared error over all x, and take expectation. This leads to the *mean integrated square error (MISE)* (Wand and Jones, 1995),

$$\mathrm{MISE}(\hat{m}(x,h)) \triangleq \mathrm{E} \int \{\hat{m}(x,h) - m(x)\}^2 \, dx. \tag{3.27}$$

By changing the order of expectation and integration, (3.27) can be rewritten

$$\mathrm{MISE}(\hat{m}(x,h)) = \int \mathrm{E}\{\hat{m}(x,h) - m(x)\}^2 \, dx = \int \mathrm{MSE}(\hat{m}(x,h)) \, dx, \tag{3.28}$$

i.e., the MISE can be obtained by integrating the MSE over all x.

3.3.2 Bias

The bias error of the linear estimate (3.6) is

$$\begin{aligned} b(x,h) &\triangleq \mathrm{E}\,\hat{m}(x,h) - m(x) \\ &= \sum_{i=1}^{N} W_i(x) m(X_i) - m(x) = \sum_{i=1}^{N} W_i(x) \left(m(X_i) - m(x) \right). \end{aligned} \tag{3.29}$$

Assuming $m(x)$ is $p+2$ times differentiable, a Taylor series expansion around x yields,

$$\begin{aligned} m(X_i) - m(x) &= m'(x)(X_i - x) + \ldots + \frac{m^{(p)}(x)}{p!}(X_i - x)^p \\ &\quad + \frac{m^{(p+1)}(x)}{(p+1)!}(X_i - x)^{p+1} + \frac{m^{(p+2)}(x)}{(p+2)!}(X_i - x)^{p+2} + \ldots \end{aligned}$$

According to (3.22), the moments up to order p vanish. It is thus possible to write

$$\begin{aligned} b(x,h) &= \sum_{i=1}^{N} W_i(x)(m(X_i) - m(x)) \\ &= \frac{m^{(p+1)}(x)}{(p+1)!} \sum_{i=1}^{N} W_i(x)(X_i - x)^{p+1} \\ &\quad + \frac{m^{(p+2)}(x)}{(p+2)!} \sum_{i=1}^{N} W_i(x)(X_i - x)^{p+2} + \ldots \end{aligned} \tag{3.30}$$

Under the assumption that the equivalent weights have bounded support, so that $W_i(x) = 0$ when $|X_i - x| > h$, we therefore see that the bias will be of size $O(h^{p+1})$.

3.3.3 Variance

The variance of the estimate is given by

$$\begin{aligned} v(x,h) &\triangleq \operatorname{Var}\hat{m}(x,h) = \mathrm{E}\left(\hat{m}(x,h) - \mathrm{E}\,\hat{m}(x,h)\right)^2 \\ &= \mathrm{E}\left(\sum_{i=1}^{N} W_i(x)\epsilon_i\right)^2 = \sigma^2 \sum_{i=1}^{N} W_i^2(x) = \sigma^2 \mathbf{w}^T(x)\mathbf{w}(x) \\ &= \sigma^2 e_1^T \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{W}^2\mathbf{X}\left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1} e_1. \end{aligned} \tag{3.31}$$

A related quantity is the *influence function* (Loader, 1997)

$$\operatorname{infl}(x) \triangleq e_1^T \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1} e_1 K_h(0). \tag{3.32}$$

It provides an upper bound for the variance,

$$\frac{1}{\sigma^2} v(x,h) \leq \operatorname{infl}(x),$$

since the elements of the diagonal matrix $\mathbf{W}^2$ is less than or equal to $K_h(0)\mathbf{W}$. Note that the diagonal elements of the hat matrix $\mathbf{H}$ in (3.23) is given by $W_i(X_i) = \operatorname{infl}(X_i)$.

3.3.4 Degrees-of-Freedom

The variance function (3.31) and the influence function (3.32) characterize the degree of pointwise smoothing. It is also useful to have a *global* measure of the amount of smoothing being performed which corresponds to the *degrees-of-freedom* measure in the parametric case (i.e., the number of parameters used). One such measure is

$$\operatorname{tr}(\mathbf{H}) = \sum_{i=1}^{N} \operatorname{infl}(X_i). \tag{3.33}$$

In case of parametric modeling, this measure reduces to the number of parameters in the model, since then

$$\operatorname{tr}(\mathbf{H}) = \operatorname{tr}(I) = \dim\theta,$$

where θ denotes the parameter vector. It is thus a natural measure of the degrees-of-freedom for a nonparametric smoother.

3.3.5 Asymptotic MSE Approximations: AMSE

A non-trivial problem with the MSE formula (3.26) is that the bias and variance terms depend on the bandwidth h in a complicated way, which makes it difficult to analyze the influence of the bandwidth on the performance of the kernel estimator. One way to overcome this problem is to use large sample approximations (i.e., large N) for the bias and variance terms. This leads to what is referred to as the *asymptotic mean square error*, AMSE (Wand and Jones, 1995).

The basic idea is to assume that the data is equispaced distributed on a bounded interval. Thus when N tends to infinity, the sums in (3.30) and (3.31) can be approximated with integrals. Derivation for a general p is quite messy in notational sense, since special care is required in case that p is even. We therefore choose to only concentrate on the case $p = 1$ corresponding to the estimator (3.15).

To simplify the notation it turns out that it is convenient to introduce the following notations;

$$\mu_k(K) \triangleq \int u^k K(u)\, du, \qquad \text{and} \qquad r(K) \triangleq \int K^2(u)\, du.$$

For the estimator (3.15), it then follows from (3.30) and some calculations that the bias error asymptotically is given by (Wand and Jones, 1995):

$$b(x,h) \simeq \tfrac{1}{2} h^2 m''(x) \mu_2(K) + o(h^2) + O(N^{-1}). \tag{3.34}$$

Equation (3.31) similarly provides the asymptotic variance formula:

$$v(x,h) \simeq \frac{1}{Nh} r(K)\sigma^2 + o((Nh)^{-1}). \tag{3.35}$$

These two expressions can be combined to form the AMSE,

$$\text{AMSE}\big(\hat{m}(x,h)\big) = \Big(\tfrac{1}{2} h^2 m''(x)\, \mu_2(K)\Big)^2 + \frac{1}{Nh} r(K)\sigma^2. \tag{3.36}$$

Here the trade-off between bias and variance becomes clear. Equation (3.36) shows that the squared bias error is asymptotically proportional to h^4, which means that in order to decrease the bias error, h has to be small. However, a small value of h yields that the variance error part becomes large, since it is asymptotically proportional to $(Nh)^{-1}$. The bandwidth must thus be chosen so that the bias is balanced against the variance.

Equation (3.36) also provides information of the convergence rate. Minimizing (3.36) w.r.t. h results in the lower bound

$$\inf_{h>0} \text{AMSE}\big(\hat{m}(x,h)\big) = \frac{5}{4} \Big(\mu_2(K)\, r^2(K)\, m''(x) \sigma^4\Big)^{2/5} N^{-4/5}, \tag{3.37}$$

which is attained for the asymptotic optimal bandwidth

$$h_{\text{AMSE}} = \left(\frac{r(K)\sigma^2}{(m''(x))^2 \mu_2^2(K)} \right)^{1/5} N^{-1/5}. \tag{3.38}$$

From these expressions we see that the mean square error tends to zero as the sample size N tends to infinity. This implies that the kernel estimator converges in probability to the true regression function $m(x)$. The best achievable rate of this convergence is of order $N^{-4/5}$, which is slower than the typical rate of order N^{-1} for parametric models as described in Section 2.4. To obtain the rate of $N^{-4/5}$, the bandwidth must be selected in order of $N^{-1/5}$.

Determining the corresponding result for the MISE measure (3.27) is straightforward. Integrating (3.36) over all x yields

$$\mathrm{AMISE}\big(\hat{m}(x, h_N)\big) = \frac{1}{4} h_N^4 r(m''(x))\, \mu_2^2(K) + \frac{1}{N h_N} r(K)\sigma^2. \tag{3.39}$$

Hence the asymptotic global optimal bandwidth is given by

$$h_{\mathrm{AMISE}} = \left(\frac{r(K)\sigma^2}{r(m''(x))\mu_2^2(K)} \right)^{1/5} N^{-1/5}. \tag{3.40}$$

Example 3.2 Asymptotically optimal bandwidth

Consider again the smoothing problem introduced in Example 3.1. Suppose we are interested in minimizing the MSE globally for all $x \in [0, 1]$, i.e., the MISE. The bandwidth that asymptotically minimizes the MISE is given by (3.40). Since

$$N = 100, \qquad \sigma^2 = 0.16,$$

$$r(K) = \frac{3}{5}, \qquad \mu_2(K) = \frac{1}{5},$$

and, since we know the true function (3.10),

$$r(m''(x)) = \int_0^1 (m''(x))^2\, dx = 605.916,$$

we get

$$h_{\mathrm{AMISE}} = 0.13,$$

which is close to the value $h = 0.15$ used in Figure 3.3 (c). ❑

Note that all results stated in this section are valid only if the point x is an interior point of the interval. At the boundary the situation in general gets degenerated, which results in slower convergence rates there.

3.4 Bandwidth Selection

Practical implementations of local polynomial estimators require that the bandwidth h is specified, and as was shown in Section 3.2, this choice is crucial for the performance of the estimator. A method that uses the available data set to automatically produce a bandwidth h is called a *bandwidth selector* (Wand and Jones, 1995). Several methods for doing this exist. They can be divided into two broad classes (Loader, 1995):

Classical Methods: Cross-validation, Generalized cross-validation, AIC, FPE, Mallows C_p and the like, which are straightforward extensions of methods used in parametric modeling, and which are aimed to minimize estimates of the MSE or similar measures.

Plug-in Methods: The unknown quantities in the AMSE and AMISE formulas are estimated from data and "plugged in" to compute an asymptotically optimal bandwidth.

3.4.1 Classical Methods

The accuracy of the estimator depends mainly on the bandwidth, the polynomial degree and the kernel. The classical methods rely on expressing the quality of the estimator as a function of the bandwidth, and to optimize this measure with respect to the bandwidth parameter. In order to do so, one needs a criterion with which to assess the performance of the fit. A natural choice here is the mean squared prediction error distance

$$P(h) = \frac{1}{N}\sum_{i=1}^{N}\big(m(X_i) - \hat{m}(X_i, h)\big)^2, \tag{3.41}$$

which consists of a squared bias component (increasing in h) and a variance component (decreasing in h). However, since the true regression function values in (3.41) are unknown, we have to replace them with measurements Y_i to form the so-called *resubstitution estimate* of the prediction error (Härdle, 1990),

$$\hat{P}(h) = \frac{1}{N}\sum_{i=1}^{N}\big(Y_i - \hat{m}(X_i, h)\big)^2. \tag{3.42}$$

Unfortunately, it turns out that this quantity is a biased estimate of $P(h)$, and $\hat{P}(h)$ is an increasing function of h, i.e., the optimal bandwidth will be the smallest one (see Figure 3.6 where $\hat{P}(h)$ has been computed for the dataset in Example 3.1). The intuitive reason for this is that observation Y_i is used in $\hat{m}(X_i, h)$ to predict itself, which will give the illusion of a perfect fit as the bandwidth approaches zero. The problem is closely related to the the concept of *overfitting* in parametric estimation theory. Solutions include to make use of *cross-validation* ideas similar to those of parametric modeling, or to multiply $\hat{P}(h)$ with a *penalizing function* $\Xi(h)$ so that it imitates the behavior of (3.41).

Cross-Validation

A straightforward solution to the bias problem of the estimate (3.42) is the *one leave-out cross-validation* method (Härdle, 1990), which is based on estimators, $\hat{m}_{-i}(X_i, h)$, where the ith observation is left out when computing the estimate. A *cross-validation* function (Stone, 1974),

$$CV(h) \triangleq \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{m}_{-i}(X_i, h))^2 \tag{3.43}$$

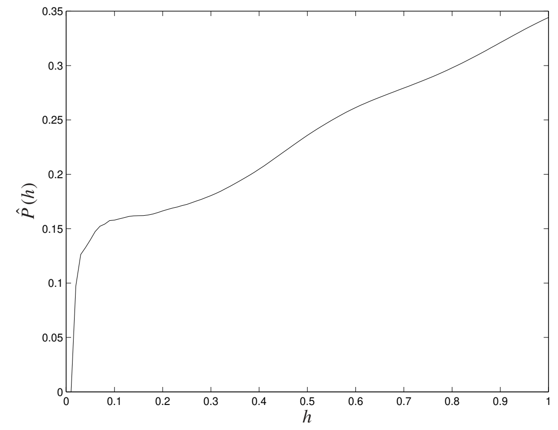


Figure 3.6 *$\hat{P}(h)$ for the dataset used in Example 3.1.*

is then formed, and the bandwidth h is taken as the minimizing argument of $CV(h)$.

At a first sight the CV score seems quite expensive to compute since the estimate has to be recomputed for each excluded observation. However, there is a short-cut,

$$CV(h) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{Y_i - \hat{m}(X_i, h)}{1 - \operatorname{infl}(X_i)}\right)^2. \tag{3.44}$$

Hence $CV(h)$ can be computed without recalculating the fit for each excluded point.

Generalized Cross-Validation

A simplified variant of the CV criterion (3.44) is *generalized cross-validation*, which was first introduced in the context of smoothing splines by Craven and Wahba (1979). Here the influence values $\operatorname{infl}(X_i)$ are replaced by their average values $\operatorname{tr}(\mathbf{H})/N$, where $\mathbf{H}$ is the hat matrix defined in (3.23). That is,

$$GCV(h) \triangleq N\frac{\sum_{i=1}^{N}(Y_i - \hat{m}(x, h))^2}{(N - \operatorname{tr}(\mathbf{H}))^2}. \tag{3.45}$$

The GCV criterion can thus be seen as a penalizing approach with

$$\Xi(h) = \left(\frac{1}{1 - \operatorname{tr}(\mathbf{H})/N}\right)^2.$$

Akaike Information Criteria: AIC, FPE

For parametric models, a commonly used penalizing approach for model structure selection is Akaike's information criterion (Akaike, 1973);

$$AIC(\theta) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{m}(X_i,\theta))^2 \cdot \exp\left(\frac{2\dim\theta}{N}\right), \tag{3.46}$$

which is an unbiased estimate of the expected Kullback-Leibler information (Kullback and Leibler, 1951). However, for nonparametric models it is not directly meaningful to think of number of parameters. A more useful approach is instead to replace this quantity with the degrees-of-freedom measure $\mathrm{tr}(\mathbf{H})$ introduced in (3.33). That is,

$$AIC(h) \triangleq \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{m}(X_i,h))^2 \cdot \exp\left(\frac{2\,\mathrm{tr}(\mathbf{H})}{N}\right). \tag{3.47}$$

Recall here that $\mathrm{tr}(\mathbf{H})$ increases with decreasing bandwidth. However, it is well known that AIC results in a quite small penalty. An enhanced version of (3.47) is the *corrected* AIC (Hurwich *et al.*, 1998);

$$AIC_C(h) \triangleq \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{m}(X_i,h))^2 \cdot \exp\left(1 + \frac{2(\mathrm{tr}(\mathbf{H})+1)}{N - \mathrm{tr}(\mathbf{H}) - 2}\right). \tag{3.48}$$

Another closely related variant is Akaike's *Final Prediction Error*,

$$FPE(h) \triangleq \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{m}(X_i,h))^2 \cdot \frac{1+\mathrm{tr}(\mathbf{H})/N}{1-\mathrm{tr}(\mathbf{H})/N}. \tag{3.49}$$

C_p Estimate of Risk

A slightly different approach compared to the above methods is to consider the *risk* function,

$$R(h) = \frac{1}{N\sigma^2}\sum_{i=1}^{N}\mathrm{E}\left(\hat{m}(X_i,h) - m(X_i)\right)^2, \tag{3.50}$$

which is a measure of the *estimation error*. A bias-variance decomposition of this quantity yields

$$\begin{aligned} R(h) &= \frac{1}{N\sigma^2}\sum_{i=1}^{N}b^2(X_i,h) + \frac{1}{N\sigma^2}\sum_{i=1}^{N}v(X_i,h) \\ &= \frac{1}{N\sigma^2}\sum_{i=1}^{N}b^2(X_i,h) + \frac{1}{N}\,\mathrm{tr}\left(\mathbf{H}\mathbf{H}^T\right). \end{aligned} \tag{3.51}$$

The expected value of (3.42) can be decomposed into bias and variance components according to

$$\begin{aligned} \mathrm{E}\,\frac{1}{N}\sum_{i=1}^{N}\big(Y_i-\hat{m}(X_i,h)\big)^2 &= \frac{1}{N}\sum_{i=1}^{N}b^2(X_i,h)+\frac{1}{N}\sum_{i=1}^{N}\mathrm{Var}\,\big(Y_i-\hat{m}(X_i,h)\big) \\ &= \frac{1}{N}\sum_{i=1}^{N}b^2(X_i,h)+\frac{\sigma^2}{N}\,\mathrm{tr}\left((I-\mathbf{H})(I-\mathbf{H})^T\right). \end{aligned} \tag{3.52}$$

Hence eliminating the squared bias terms from (3.51) and (3.52) and ignoring the expectation result in the unbiased estimate of (3.50),

$$\begin{aligned} CP(h) &\triangleq \frac{1}{N\sigma^2}\sum_{i=1}^{N}(Y_i-\hat{m}(X_i,h))^2-\frac{1}{N}\,\mathrm{tr}\left((I-\mathbf{H})(I-\mathbf{H})^T\right)+\frac{1}{N}\,\mathrm{tr}(\mathbf{H}\mathbf{H}^T) \\ &= \frac{1}{N\sigma^2}\sum_{i=1}^{N}(Y_i-\hat{m}(X_i,h))^2-1+\frac{2}{N}\,\mathrm{tr}(\mathbf{H}). \end{aligned} \tag{3.53}$$

For parametric methods, this is known as Mallows C_p criterion (Mallows, 1973). It was extended to local regression by Cleveland and Devlin (1988).

Example 3.3 **A Comparison**

In Figure 3.7, $CV(h)$, $GCV(h)$ and $CP(h)$ functions for the dataset used in Example 3.1 are depicted. All functions have their minimum at $h \approx 0.17$, which indicates that this would be a good choice of a global bandwidth for this particular example. ❑

3.4.2 Plug-in Methods

The other class of bandwidth selectors, so-called *direct plug-in methods* (Wand and Jones, 1995; Fan and Gijbels, 1996), is based on the simple idea of "plugging in" values of σ^2 and $r(m''(x))$ into the asymptotically optimal bandwidth formula (3.40). This was in fact what we did in Example 3.2, since we knew both the true regression function and the noise variance. In practice, however, the values of $r(m''(x))$ and σ^2 are unknown and have to be estimated from data. Quite often they are estimated on the basis of some preliminary smoothing stage, which then raises a secondary bandwidth selection problem. For example, an estimator for $r(m''(x))$ is given by

$$r\widehat{(m''(x))} = \frac{1}{N}\sum_{i=1}^{N}(\widehat{m''}(X_i,g))^2, \tag{3.54}$$

and an estimator for σ^2 by

$$\hat{\sigma}^2 = \frac{1}{N-2\,\mathrm{tr}(\mathbf{H})+\mathrm{tr}(\mathbf{H}^T\mathbf{H})}\sum_{i=1}^{N}(Y_i-\hat{m}(X_i,l))^2, \tag{3.55}$$

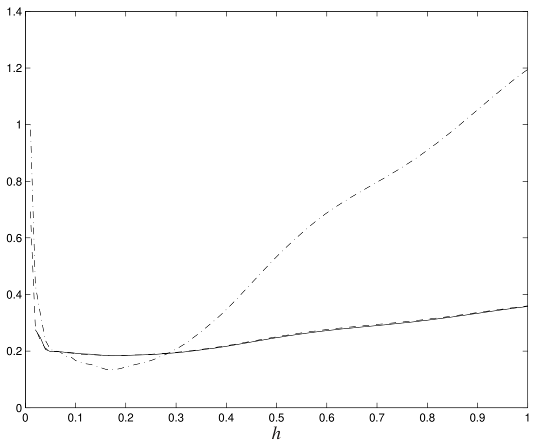


Figure 3.7 *The $CV(h)$ criterion (solid), the $GCV(h)$ criterion (dashed), and the $CP(h)$ criterion (dash-dotted) for the dataset used in Example 3.1.*

where the normalization is included in order to make the estimate unbiased (Wand and Jones, 1995). The unknown quantities in (3.40) have thus to be estimated using two additional estimators $\widehat{m''}(X_i, g)$ and $\hat{m}(X_i, l)$ with auxiliary so-called *pilot* bandwidths g and l, respectively.

Plug-in methods have been thoroughly studied within the statistical community during the last decade, and their superior performance in comparison to the classical methods has been claimed. However, Loader (1995) argues that this has found to be without foundation: "The plug-in approach is heavily dependent on arbitrary specification of pilot bandwidths; if these are wrong, the selection is wrong".

3.5 Local Maximum-Likelihood Models

When it is known that the observations Y_i in (3.1) has a non-Gaussian distribution, say $f(y, m)$, it might be useful that instead of (3.4) consider a nonparametric maximum-likelihood approach as introduced by Tibshirani and Hastie (1987). Let

$$l(y, m) \triangleq \log f(y, m). \tag{3.56}$$

Then forming the local log-likelihood function

$$\mathcal{L}_x(\beta) = \sum_{i=1}^{N} l\left(Y_i, \sum_{j=0}^{p} \beta_j (X_i - x)^j\right) K_h(X_i - x) \tag{3.57}$$

and maximizing over β_j will lead a local maximum-likelihood estimate of the same fashion as (3.6). Note that this is consistent with (3.4) assuming Gaussian distributed noise with standard deviation $\sigma = 1$. Maximization of the log-likelihood has typically to be done by means of numerical search, for instance by using Newton's algorithm (2.24). This makes local likelihood even more computationally intensive than the ordinary local polynomial approach. Nevertheless, an application for local likelihood estimation will be presented in Chapter 6.

3.5.1 Bandwidth Selectors for Local Likelihood

Bandwidth selections schemes can also be derived for the local likelihood case. However, in this case it is not so useful to assess the quality of the fit by means of the squared residuals. A more appropriate measure is the *deviance* (Loader, 1997) which is defined as

$$D(Y, \hat{m}) \triangleq 2 \left(\sup_m l(Y, m) - l(Y, \hat{m}) \right). \tag{3.58}$$

It is easy to show that the deviance reduces to the squared residual in the case of Gaussian distributed noise. The likelihood counterpart to (3.43) is

$$\begin{aligned} LCV(h) &\triangleq \sum_{i=1}^{N} D(Y_i, \hat{m}_{-i}(X_i, h)) \\ &\approx C_L - \frac{2}{N} \sum_{i=1}^{N} l(Y_i, \hat{m}(X_i, h)) + \frac{2}{N} \sum_{i=1}^{N} \text{infl}(X_i) \cdot \left[\dot{l}(Y_i, \hat{m}(X_i, h)) \right]^2, \end{aligned} \tag{3.59}$$

where

$$\text{infl}(x) = e_1^T \left(\mathbf{X}^T \mathbf{W} \mathbf{V} \mathbf{X} \right)^{-1} e_1 K_h(0), \tag{3.60}$$

is the likelihood influence function and

$$\mathbf{V} = \text{diag}(-\ddot{l}(Y_i, \hat{m}(X_i, h)), \ldots, -\ddot{l}(Y_N, \hat{m}(X_N, h))).$$

Here $\dot{l}(Y, m)$ and $\ddot{l}(Y, m)$ denote partial derivatives of $l(\cdot, \cdot)$ w.r.t. m. The corresponding likelihood AIC is obtained by replacing the squared derivative in (3.59) with its expected value. This is (apart from the constant C_L) consistent with the ordinary AIC definition (3.47).

3.6 The Degree of the Local Polynomial

An obvious question, which so far has not been paid any attention in this chapter, is how to choose the order p of the local polynomial. It is a common fact that fitting higher orders polynomials leads to a possible reduction of the bias, but at the same time gives an increase of the variability, since more parameters are introduced. Therefore the polynomial degree is, like the bandwidth, a bias-variance trade-off.

It can be shown that the asymptotic variance of the estimate only increases when moving from an odd order approximation to its consecutive even order approximation (Fan and

Gijbels, 1996). It has therefore been claimed within the statistical literature that estimators with $p-\nu$ odd will perform better than estimators with $p-\nu$ even, since the extra parameter gives a bias reduction, especially in the boundary regions, but does not cause an increase of variability. However, Cleveland and Loader (1994) do not agree on that. They argue that global parametric fitting can be seen as the limiting case of local regression. Therefore, ruling out fitting with even degrees is no more sensible than ruling out even degrees for global polynomial fitting.

In some situations, moving from a qth to a $(q+1)$th order approximation results in a too dramatic change in the estimate, and one wishes a compromise. A solution is to consider a linear combination of the two approximations,

$$\hat{m}_p(x,h) = (1-c)\hat{m}_q(x,h) + c\,\hat{m}_{q+1}(x,h),$$

where $0 \le c \le 1$. In such situations, $p = q + c$ is referred to as the *mixing degree* (Cleveland *et al.*, 1995). The mixing degree can, like the bandwidth, be determined from data using cross-validation techniques as described in Section 3.4.1.

3.7 Adaptive Methods

Until now we have only considered global bandwidth (and mixing degree) selectors, that select a *single* value of the parameter of interest. However, adaptive (or local) methods that select local values for each estimation point have gained a significant interest in recent years, and the development of them still seems to be an open and active research area.

Adaptive bandwidth selection will be part of the model-on-demand approach described in the following chapter, though, so we will postpone the treatment of such methods to this part of the thesis.

3.8 Extensions to the Multivariable Case

The theory for local polynomial estimation can rather easily be extended to the multivariable case where $X_i \in \mathbb{R}^d$ and $Y_i \in \mathbb{R}$. The multivariable local linear estimator is for instance given by

$$\hat{m}(x,H) = \hat{\beta}_0, \tag{3.61}$$

where $\hat{\beta} = (\hat{\beta}_0,\ \hat{\beta}_1^T)^T$ is the solution to the weighted least squares problem

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} \Big(Y_i - \beta_0 - \beta_1^T(X_i - x)\Big)^2 \mathbf{K}_H(X_i - x). \tag{3.62}$$

Here

$$\mathbf{K}_H(u) \triangleq |H|^{-1}\mathbf{K}\Big(H^{-1}u\Big), \tag{3.63}$$

and $\mathbf{K}(\cdot)$ is an d-dimensional kernel function satisfying

$$\int_{\mathbb{R}^d} \mathbf{K}(u)\, du = 1, \qquad \int_{\mathbb{R}^d} u\mathbf{K}(u)\, du = 0. \tag{3.64}$$

The symmetric positive definite $d \times d$ matrix H is called the *bandwidth matrix*, since it is the multivariable counterpart to the usual scalar bandwidth parameter.

The kernel function is often taken as an d-dimensional probability density function. There are two common methods for constructing multidimensional kernel functions from a scalar kernel $K(\cdot)$; the *product* construction,

$$\mathbf{K}(u) = \prod_{i=1}^{d} K(u_i) \tag{3.65}$$

and the *radial* construction

$$\mathbf{K}(u) = C_{\mathbf{K}} \cdot K\Big((u^T u)^{1/2}\Big), \tag{3.66}$$

where $C_{\mathbf{K}}$ is a normalizing constant so that $\mathbf{K}(\cdot)$ becomes a multivariable density function. These directly correspond to the parametric construction methods described in Section 2.2.2.

A problem with multidimensional kernel estimators, though, is the large number of parameters. In general, the bandwidth matrix H has $\frac{1}{2}d(d+1)$ independent entries (H is symmetric), which, even for a quite moderate value of d, gives a non trivial number of smoothing parameters to choose. Considerable simplifications can be obtained by restricting H to be a diagonal matrix $H = \operatorname{diag}(h_1, \ldots, h_d)$, which leads to a kernel of the type

$$\mathbf{K}_H(u) = \left(\prod_{k=1}^{d} h_k\right)^{-1} \mathbf{K}\left(\frac{u_1}{h_1}, \ldots, \frac{u_d}{h_d}\right), \tag{3.67}$$

or by letting H be specified by a single bandwidth parameter $H = hI$, which results in the symmetric kernel

$$\mathbf{K}_H(u) = h^{-d}\mathbf{K}(u/h). \tag{3.68}$$

3.8.1 Asymptotic MSE Approximations

The multivariable fixed design counterpart of the univariate asymptotic MSE in Section 3.3.5 for a local linear smoother is (see Ruppert and Wand (1994))

$$\operatorname{AMSE}(\hat{m}(x, H)) = \left(\frac{1}{2}\mu_2(\mathbf{K}) \operatorname{tr}\Big\{H^2 \mathcal{H}_m(x)\Big\}\right)^2 + N^{-1}\sigma^2 |H|^{-1} r(\mathbf{K}), \tag{3.69}$$

where $\mathcal{H}_m(x)$ denotes the Hessian (i.e., the second order derivative matrix) of m. Using the single bandwidth kernel (3.68), this simplifies to

$$\operatorname{AMSE}(\hat{m}(x, h)) = \left(\frac{1}{2}\mu_2(\mathbf{K}) h^2 \operatorname{tr}\{\mathcal{H}_m(x)\}\right)^2 + N^{-1}\sigma^2 h^{-d} r(\mathbf{K}). \tag{3.70}$$

Hence minimizing this expression w.r.t. h gives,

$$\inf_{h>0} \text{AMSE}(\hat{m}(x,h)) =$$
$$\left(\frac{1}{4}d^{4/(d+4)} + d^{-d/(d+4)}\right)\left(\mu_2^d(\mathbf{K})r^2(\mathbf{K})\,\text{tr}^d\,\mathcal{H}_m(x)\sigma^4\right)^{2/(d+4)} \cdot N^{-4/(d+4)}, \quad (3.71)$$

with optimal bandwidth

$$h = \left(\frac{d\,r(\mathbf{K})\sigma^2}{\mu_2^2(\mathbf{K})\,\text{tr}^2\{\mathcal{H}_m(x)\}}\right)^{1/(d+4)} N^{-1/(d+4)}. \quad (3.72)$$

As shown, the convergence rate $N^{-4/(d+4)}$ for the multivariable kernel estimator is slower than for the corresponding scalar estimator ($d = 1$). This is a manifestation of the so-called *curse-of-dimensionality* which follows from the sparseness of data in higher regressor dimensions. For example, even with the quite large component-wise distance of 0.1, it takes ten billions of data points to fill up a unit cube in $\mathbb{R}^{10}$.

3.8.2 Dimensionality Reduction Methods

The curse-of-dimensionality problem discussed above has motivated the derivation of methods that – in some way or another – try to reduce the dimensionality of the underlying smoothing problem. One such approach is *additive models* introduced by (Hastie and Tibshirani, 1990).

Additive models assume that the regression surface is an additive function of the components of the predictor. For instance, for $d = 2$ we can consider the model

$$Y_i = m_1(X_{i,1}) + m_2(X_{i,2}) + \epsilon_i, \qquad i = 1, \ldots, N. \quad (3.73)$$

Estimation of the submodels in (3.73) can be performed using the *backfitting algorithm* which alternates between estimating $m_1(x_1)$ from $X_{i,1}$ and $Y_i - m_2(X_{i,2})$, and $m_2(x_2)$ from $X_{i,2}$ and $Y_i - m_1(X_{i,1})$. If this procedure converges, the resulting estimate is obtained as

$$\hat{m}(x) = \hat{m}_1(x_1, h_1) + \hat{m}_2(x_2, h_2).$$

A variant of the above idea where the data is projected in certain specified directions is *projection pursuits models* (Friedman and Stuetzle, 1981). Another dimensionality reduction approach is to assume the the model is global in some directions. This is referred to as *conditionally parametric models* and will be discussed further in the next chapter.

4

Model-on-Demand

In this chapter we will present an alternative solution to the nonlinear regression/prediction problem presented in Chapter 1 which we believe might be useful in situations when very large datasets are available; *model-on-demand*. The basic idea is to store all observed data in a database, and to build local models "on demand" as the actual need arises.

Since the modeling is performed locally, it is in contrast to parametric modeling sufficient to consider rather simple model structures. In particular, it turns out that nonparametric models of the fashion described in Chapter 3 are well suited for this purpose. The main difference in this chapter, however, is that we will perform the model tuning locally, independently of the result obtained at other locations.

The organization of the chapter is as follows. The first two parts, Sections 4.1 and 4.2, give an introduction and describes the basic principles behind the model-on-demand philosophy. Section 4.3 discusses different approaches for how a particular estimator can be implemented. It turns out that two possible solutions are local polynomial regression models similar to those of Chapter 3, and a weight optimization approach. These are discussed further in Sections 4.4 and 4.5 respectively. Section 4.6 compares the two modeling approaches. Section 4.7 discusses how to obtain approximate confidence bands, Section 4.8 discusses several aspects associated with data management and storage, and Section 4.9, finally, provides some conclusions.

4.1 Introduction

Let us again return to the nonlinear modeling/prediction problem introduced in Chapter 1. Suppose that we have collected a large set of observations $\{(Y_i, X_i)\}_{i=1}^N$ which we assume can be modeled as

$$Y_i = m(X_i) + \epsilon_i, \qquad i = 1, \dots, N, \tag{4.1}$$

and that we want to compute an estimate (or prediction) of the nonlinear mapping $m(\cdot)$ at a certain operating point x. The observations may for instance relate to the input-output behavior of a nonlinear dynamic system whose future outputs we want to predict, but we will in this chapter not restrict ourselves to special classes of data, but rather, allow a more general function approximation framework. More specialized applications to dynamical systems will instead be treated later in Chapters 5, 6 and 7.

As outlined in Chapters 2 and 3 the prediction problem has been solved traditionally in system identification and statistics by *global* modeling methods like nonlinear black-box models and nonparametric methods. However, when the number of samples N grows very large, this approach becomes less attractive to deal with, both from a data management and strict computational point of view. Situations where one is faced with enormous datasets has become quite common today within the field of process control. In some applications it is for instance not unusual that the volumes of data may be in the order of Gigabytes.

As discussed in Chapter 2, the global (parametric) modeling problem is typically associated with an optimization process which aims at minimizing the "size" of the misfit between the model and the data;

$$\hat{\theta} = \arg\min_{\theta} \sum_i \ell(Y_i - m(X_i, \theta)), \tag{4.2a}$$

$$\hat{m}(x) = m(x, \hat{\theta}). \tag{4.2b}$$

Although this approach has the appealing feature of giving a high degree of data compression once the parameters are tuned to fit the data, it seems both inefficient and unnecessary to waste a large amount of calculations to optimize a model which is valid over the entire regressor space, while it in most cases, especially in higher regressor dimensions, is more likely that we will only visit a very restricted subset of it. Furthermore, the minimization problem (4.2) is for general model structures typically non-convex and will have a number of local minima which make the search for the global minimum hard.

4.2 The Model-on-Demand Idea

Inspired by ideas both from local modeling and database systems technology, we have taken a conceptually different point of view. We assume that all available observations are stored in a database, and that models are built dynamically *on demand* when the actual need arises. The main idea is as follows: When there is need for a model at (or around) a certain operating point x, relevant data is retrieved from the database, and a modeling operation is performed on that subset, see Figure 4.1. For this concept we have adopted the name *model-on-demand*, MOD.

The key issue here is of course what to be defined as relevant data. In accordance with the nonparametric methods in Chapter 3 the most natural choice is to consider the data belonging to a small neighborhood around the operating point, but it is clear that other

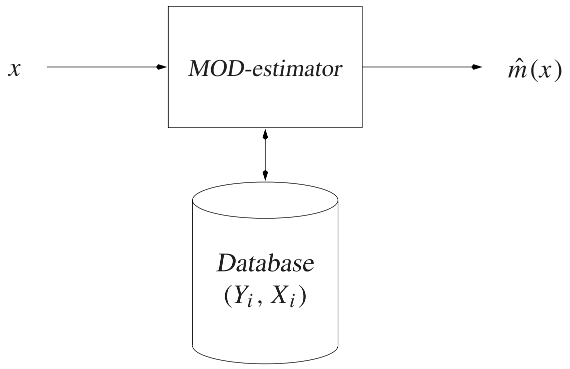


Figure 4.1 *The model-on-demand idea: A subset of relevant data is retrieved from the database, and is used to compute an estimate of $m(x)$.*

more arbitrary relevance measures also can be considered. Chapter 5 will give examples of this.

In earlier contributions (see, for instance, Stenman *et al.* (1996), Stenman (1997) and Stenman *et al.* (1997)) we used the name *just-in-time models*, a term coined by Cybenko (1996). However, to avoid confusion with Cybenko's slightly different approach, we decided to change the name.

Compared to global methods, an advantage with the model-on-demand idea is that the model tuning is optimized locally at the point x, and that this optimization is performed independently of the results obtained at other locations. This increases the flexibility and might improve the performance. In addition, since the estimation is deferred until query time, there are neither no problems with adding new observations to the database. This is in contrast to global modeling approaches, where the model in general has to be recomputed when new data arrives.

A potential drawback with the approach, though, is the computational complexity, both since it is required to search for a neighborhood of x in a multidimensional regressor space, and since the derived estimator is quite computationally intensive. This follows from the fact that each demanded prediction in practice gives rise to an independent estimation problem like (4.2) and that the obtained model information is discarded between subsequent queries. In this work though, we will mainly focus on the properties of the modeling part of the problem. Computational issues and data management and storage topics will be just briefly discussed in the end of the chapter.

Before commencing a more detailed treatment of the model-on-demand concept, we will introduce some basic assumptions and notations that will be used throughout the chapter. We assume that the data are generated according to (4.1), where $X_i \in \mathbb{R}^d$ and $Y_i \in \mathbb{R}$, and that ϵ_i denotes independent, identically distributed random variables with zero means and variances σ_i^2. We further assume that the true regression function $m(\cdot)$ is at least twice differentiable.

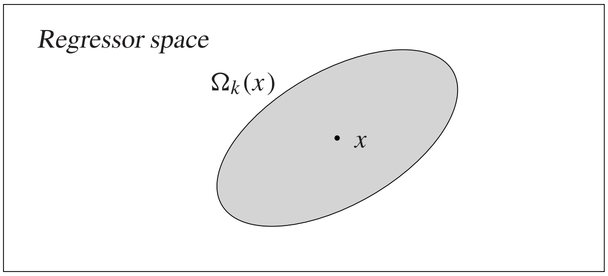


Figure 4.2 *The data belonging to a local neighborhood of x are used for estimation. The shape of the neighborhood $\Omega_k(x)$ is depending on which distance function is used.*

4.3 Model-on-Demand Estimators: Possibilities

When performing the localized modeling operation discussed in the preceeding section, it is clear that this can be done in a quite arbitrary way and that a wide range of solutions is possible. The simplest approach is perhaps to use a nearest neighbor estimator, but this normally becomes unsatisfactory because of the presence of noise in the observations. A natural remedy is to consider a more sophisticated modeling approach like (4.2) and incorporate a *weighting* in the criterion so as to localize the functional approximation around x. That is,

$$\hat{\beta} = \arg\min_{\beta} \sum_{i\in\Omega_k(x)} \ell(Y_i - m(X_i,\beta))w_i(x), \tag{4.3a}$$

$$\hat{m}(x) = m(x,\hat{\beta}). \tag{4.3b}$$

To be consistent with the notation in Chapter 3, we have here replaced θ with β. Moreover, $w_i(x)$ denotes the weights (which we soon shall return to), and the symbol $\Omega_k(x)$ is used to denote the index set for a neighborhood of x containing k samples, i.e.,

$$\Omega_k(x) \triangleq \{i_1,\dots,i_k\} = \{i : d(X_i,x) \le h\} \tag{4.4}$$

for some $h \in \mathbb{R}$ and some suitable scalar-valued distance function $d(\cdot,\cdot)$, see Figure 4.2. The parameter h can in this context be interpreted as a scalar smoothness or “bandwidth” parameter, which determines the radius of the neighborhood.

4.3.1 Choice of Local Model Structure

In general, there are no limits on what model structure can be used as local model in (4.3), and in principle it is possible to adapt all the global modeling approaches from Chapter 2 to the local framework. However, it is well known that models that are linear in the unknown parameters, such as local polynomials, are easier to estimate than more general model structures. Since the parameter estimation cost is a significant part of the total computational cost for each prediction, this is an important benefit.

In order to get an reasonable trade-off between approximation error and computational complexity, we shall therefore in the sequel of the chapter mainly concentrate on the local *linear* and local *quadratic* model structures. Then, and if the quadratic L_2 norm is used, the localized fitting criterion (4.3) reduces to

$$\hat{\beta} = \arg\min_{\beta} \sum_{i \in \Omega_k(x)} \Big(Y_i - \mathcal{B}^T(X_i - x)\beta \Big)^2 w_i(x), \tag{4.5a}$$

$$\hat{m}(x) = \hat{\beta}_0, \tag{4.5b}$$

which can be solved efficiently and explicitly by means of standard weighted least squares. Here, as in previous chapters,

$$\beta = \begin{pmatrix} \beta_0 & \beta_1 & \dots & \beta_p \end{pmatrix}^T$$

denotes the parameter vector, and $\mathcal{B}(X_i - x)$ will represent the vector of basis functions for the polynomial model. That is,

$$\mathcal{B}(X_i - x) = \begin{pmatrix} 1 & (X_i - x)^T \end{pmatrix}^T$$

for the local linear case where $p = 1 + d$, and

$$\mathcal{B}(X_i - x) = \begin{pmatrix} 1 & (X_i - x)^T & \operatorname{vech}^T\big((X_i - x)(X_i - x)^T\big) \end{pmatrix}^T,$$

for the local quadratic ditto where $p = 1 + d + d(d+1)/2$. In the latter case the notation $\operatorname{vech}(A)$ denotes the *vector-half* of the symmetric matrix A, i.e., the vector obtained by stacking the columns of the matrix A above each other, and eliminating the above-diagonal entries (Henderson and Searle, 1979).

If the distribution of the noise term ϵ_i is known, it is of course possible to replace the least-squares estimator (4.5) with a local likelihood counterpart. However, since the parameter estimation then has to be performed by numerical search methods, this approach is undesirable for general situations because of the computational complexity.

4.3.2 Selecting the Weights

A very important feature that completely distinguishes the local modeling (4.3) from its global counterpart (4.2) is the choice of weighting, $w_i(x)$. We shall in this chapter study two different approaches for weight selection. The first one, which will be treated in Section 4.4, is based on *local polynomial* techniques from Chapter 3 and assumes that the weights are implicitly specified by a kernel. The second approach, which will be studied in Section 4.5, instead relies upon a *direct optimization scheme* that determines the weight sequence explicitly.

4.3.3 Related Approaches in the Literature

As pointed out in Chapters 1 and 3, the idea of local modeling with polynomial models is not a new concept. Various local approaches have been studied for a long time within the statistical community, although there almost always seems to have been global assumptions and considerations in some step of the estimation procedure. However, adaptive methods have gained a significant interest in recent years, and the development of them still seems to be an open and active research area, see for instance Fan and Gijbels (1996) and the references therein.

In the fields of artificial intelligence and machine learning, similar nonparametric ideas were rediscovered and developed by Atkeson and Reinkensmeyer (1988), Aha (1989) and Bottou and Vapnik (1992), and have successfully been used under the names *lazy learning* and *least commitment learning* for robot learning and control (Atkeson *et al.*, 1997*a*; Atkeson *et al.*, 1997*b*).

4.4 MOD-estimation using Local Polynomial Regression

As a start of our modeling efforts we will in this section study how the local polynomial ideas from Chapter 3 can be utilized in the model-on-demand framework. By adapting the multivariable kernel methods to the local modeling problem, a natural approach is to let the smoothing weights in (4.5a) be explicitly defined by a multivariable kernel function, that is,

$$w_i(x) = \mathbf{K}_H(X_i - x) = \frac{1}{|H|}\mathbf{K}\Big(H^{-1}(X_i - x)\Big). \tag{4.6}$$

The main difference with this approach in the MOD framework as compared to the statistical setting reviewed in Chapter 3, however, is that we do not want to assign weights globally to the entire data set, rather just consider the local data belonging to the neighborhood $\Omega_k(x)$. In addition, we would like to perform the weight tuning locally at x without taking into account estimation results at other locations.

As discussed in Section 3.8, the general kernel approach (4.6) with a full bandwidth matrix will usually result in a quite large number of smoothing parameters to choose. In order to reduce the complexity, we will therefore concentrate on the single bandwidth case $H = h \cdot I$, and the radial construction (3.66) but with a more general vector norm. It is convenient to decompose the weighting scheme that thus follows into two separate mappings; one that maps the local data to distances,

$$d(X_i, x) = \|X_i - x\|_{\mathbf{M}}, \tag{4.7}$$

where $\|\cdot\|_{\mathbf{M}}$ denotes a scaled vector norm, and another that maps the scaled distances to weights,

$$w_i(x) = K\left(\frac{d(X_i, x)}{h}\right), \tag{4.8}$$

where $K(\cdot)$ is a scalar kernel function of the kind described in Section 3.2.1. (We have here ignored the normalizing constant since it in fact does not affect the fit). In terms of the general formulation (4.6), this effectively means that we obtain a bandwidth matrix of the type

$$H = h \cdot \mathbf{M}^{-1/2}. \tag{4.9}$$

It turns out that the choice of distance function is a very important design issue. Let us therefore discuss this topic further.

4.4.1 Distance Functions

Local polynomial regression depends in the multivariable setting critically on the distance function $d(X_i, x)$ through the weighting $w_i(x)$. The relative importance of the regressor components in generating the distance measurement depends on how the regressors are *scaled*, i.e., how much they are stretched or shrunk. (Note that we here will use the term *scaling* for this concept, since the term *weighting* in the local modeling setting is reserved for the contribution of different measurements in the loss function (4.5a)).

In general, there are many ways to define and use distance functions. However, commonly used choices in the context of local modeling typically include (Atkeson *et al.*, 1997*a*):

- **Global distance functions**: The same distance function is used over the entire regressor space and is chosen in advance of the queries.
- **Local distance functions**: Different distance functions are used in different parts of the regressor space. This type of distance function can be further divided into the following subclasses:
  - **Query-based distance functions**: The scaling that defines the distance function is set on each demanded prediction by optimizing some goodness-of-fit measure such as cross-validation. This approach is sometimes referred to as *uniform metric* and has been discussed in Stanfill and Waltz (1986).
  - **Point-based distance functions**: Each stored data point has associated with it a distance function $d_i(X_i, x)$. The optimization procedure (4.5a) uses different distance functions for each point X_i, and they are normally chosen in advance of the queries and stored together with the data. This is sometimes referred to as *variable metric* (Stanfill and Waltz, 1986).

In case that the distance function is defined according to (4.7), it is entirely determined by the scaling matrix $\mathbf{M}$. Then altering the elements in $\mathbf{M}$ will change the shape and the orientation of the neighborhood $\Omega_k(x)$, which might affect the accuracy of the prediction. Possible choices of distance functions for continuous, real-valued regressor variables include (Atkeson *et al.*, 1997*a*):

- **Euclidean distance**:

$$d(X_i, x) = \|X_i - x\| = \sqrt{(X_i - x)^T (X_i - x)}, \tag{4.10}$$

where $\mathbf{M}$ is the identity matrix.

- **Diagonally scaled Euclidean distance**:

$$d(X_i, x) = \|X_i - x\|_{\mathbf{M}} = \sqrt{(X_i - x)^T \mathbf{M}(X_i - x)}, \tag{4.11}$$

  where $\mathbf{M}$ is a diagonal matrix, $\mathbf{M} = \text{diag}(m_1, m_2, \ldots, m_d)$, with $m_i \geq 0$.

- **Fully scaled Euclidean distance**:

$$d(X_i, x) = \|X_i - x\|_{\mathbf{M}} = \sqrt{(X_i - x)^T \mathbf{M}(X_i - x)}, \tag{4.12}$$

  where the matrix $\mathbf{M}$ is full and positive semidefinite.

The diagonal distance function (4.11) transforms the radially symmetric distance function (4.10) into an axis parallel ellipse, see Figure 4.3 (a)–(b). By also allowing cross-terms as in (4.12) it is possible to make the ellipse arbritrary oriented. See Figure 4.3 (c).

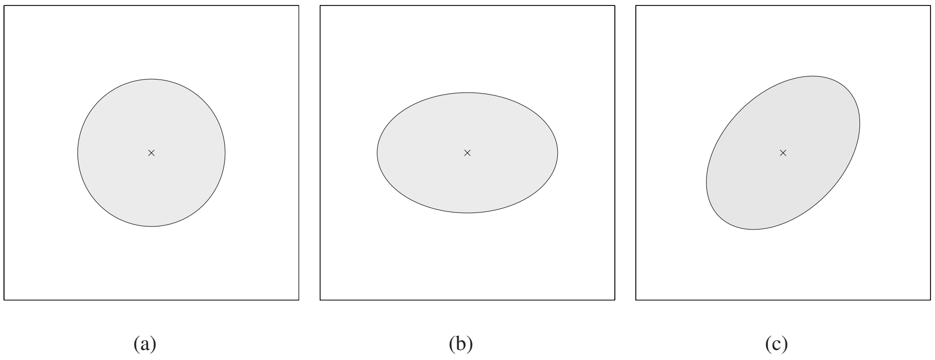


Figure 4.3 *Different distance functions. (a) Euclidean distance. (b) Diagonally scaled Euclidean distance. (c) Fully scaled Euclidean distance.*

Selecting the scaling matrix $\mathbf{M}$ is very important, especially when the regressor components have very different magnitudes. An obvious default choice for a global distance function (which will be adopted here) is to make it proportional to the inverse covariance of the regressors. This provides a natural normalization with respect to the standard deviation, so that each regressor component has roughly the same influence on the distance measure.

Cleveland and Grosse (1991) point out that zeroing out an entire column of the scaling matrix corresponds to making the model global in a certain direction. They refer to this as a *conditionally parametric* model. This approach can in some situations be useful to combat the *curse-of-dimensionality* introduced in Chapter 3, since the locality of the function approximation then is reduced.

4.4.2 Computing the Local Estimate

For notational simplicity in the remainder of the section, it is convenient to introduce local versions of the matrix quantities defined in Section 3.2.3. Let

$$\mathbf{y}_k \triangleq (Y_{i_1}, \ldots, Y_{i_k})^T, \tag{4.13}$$

and introduce the local design matrix of basis functions

$$\mathbf{X}_k \triangleq \begin{pmatrix} \mathcal{B}^T(X_{i_1} - x) \\ \vdots \\ \mathcal{B}^T(X_{i_k} - x) \end{pmatrix}. \tag{4.14}$$

Furthermore, let $\mathbf{W}_k$ denote the local weight matrix,

$$\mathbf{W}_k \triangleq \operatorname{diag}(w_{i_1}(x), \ldots, w_{i_k}(x)). \tag{4.15}$$

The solution to the local polynomial modeling problem (4.5a) is thus (analogous to (3.13)) obtained as the weighted least squares solution

$$\hat{\beta} = \left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k\right)^{-1} \mathbf{X}_k^T \mathbf{W}_k \mathbf{y}_k \tag{4.16}$$

i.e., the local estimate (4.5) can be written in the linear smoother form

$$\hat{m}(x, k) = \mathbf{w}_k^T \mathbf{y}_k, \tag{4.17}$$

with weights

$$\mathbf{w}_k^T \triangleq \begin{pmatrix} W_{i_1}(x) & \ldots & W_{i_k}(x) \end{pmatrix} = e_1^T \left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k\right)^{-1} \mathbf{X}_k^T \mathbf{W}_k. \tag{4.18}$$

The key problem here is of course to determine which data the local estimate should be based on. However, since we by support of the model-on-demand idea only are interested in optimizing the prediction *locally* at the current operating point x, it seems natural to aim at minimizing the pointwise MSE measure of the estimator (4.17) subject to the size (and possibly also shape) of the local neighborhood $\Omega_k(x)$. Recall from Chapter 3 that the MSE measure is defined by the expression

$$\operatorname{MSE}\left(\hat{m}(x, k)\right) = \operatorname{E}\left(\hat{m}(x, k) - m(x)\right)^2 \tag{4.19}$$

and that it can be decomposed into a squared bias term (increasing in k) and a variance term (decreasing in k). In a computational sense this raises demand for an "algorithm" of the following fashion:

> Start initially with a small neighborhood, just containing as much data as the least squares solution (4.16) is well conditioned, and expand it with more and more observations until a reasonable trade-off between bias and variance is obtained.

In practice, this can be achieved by means of localized versions of the bandwidth selection procedures in Section 3.4. This will be described more in detail in Section 4.4.5.

4.4.3 Regularization

A potential problem that may occur when solving for the weighted least squares estimate (4.16) is that the data is distributed in such a way that the information matrix $\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k$ is nearly singular. The most common situation involves a discrete regressor variable, so that the neighborhood only contains data points lying on one or two lines. If there are not enough neighboring point with non-zero contributions in all directions of the regressor space, there are not enough equations to solve for the parameters of the local polynomial. A possible solution is *regularization* or *ridge regression* (Draper and Nostrand, 1979; Draper and Smith, 1981), which instead of (4.16) uses

$$\hat{\beta} = \left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k + \mathbf{\Lambda}\right)^{-1} \left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{y}_k + \mathbf{\Lambda}\bar{\beta}\right) \tag{4.20}$$

when solving for the parameter vector β. Here $\mathbf{\Lambda}$ is a diagonal matrix with small positive elements λ_i^2;

$$\mathbf{\Lambda} = \operatorname{diag}(\lambda_0^2, \ \dots\ , \lambda_p^2), \tag{4.21}$$

and $\bar{\beta}$ is an expectation of the parameter estimate (often taken as a vector of zeros). Since the least squares solution (4.16) corresponds to solving the over-determined system of equations

$$\mathbf{W}_k^{1/2} \mathbf{X}_k \beta \cong \mathbf{W}_k^{1/2} \mathbf{y}_k.$$

This is equivalent to adding "fake" observations, i.e.,

$$\begin{pmatrix} \mathbf{W}_k^{1/2} \mathbf{X}_k \\ \mathbf{\Lambda}^{1/2} \end{pmatrix} \beta \cong \begin{pmatrix} \mathbf{W}_k^{1/2} \mathbf{y}_k \\ \mathbf{\Lambda}^{1/2} \bar{\beta} \end{pmatrix}$$

which, in the absence of informative real data, biases the estimate against $\bar{\beta}$ (Draper and Smith, 1981; Atkeson *et al.*, 1997*a*). From a Bayesian estimation perspective, this can justified if the extra terms added to (4.20) in some way or another reflect the assumed *a priori* distributions of the estimated parameters.

Cleveland and Grosse (1991) have considered another approach in their implementation of the LOESS package. They compute the solution to (4.16) using singular value decomposition (SVD), and set small singular values to zero in the computed inverse. This "pseudo-inverse" approach is equivalent to eliminating directions with limited data support from the local model.

4.4.4 Incorporating Variance Information

If the variances of the observations are known *a priori*, it is from a maximum-likelihood viewpoint natural to modify the least squares criterion (4.5) to incorporate the variance information as an extra weighting $v_i = 1/\sigma_i^2$. That is,

$$\hat{\beta} = \arg\min_{\beta} \sum_{i \in \Omega_k(x)} \left(Y_i - \mathcal{B}^T (X_i - x)\beta\right)^2 w_i(x) \cdot v_i, \tag{4.22a}$$

$$\hat{m}(x) = \hat{\beta}_0. \tag{4.22b}$$

The solution to the least squares problem (4.16) is changed accordingly;

$$\hat{\beta} = \left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{V}_k \mathbf{X}_k\right)^{-1} \mathbf{X}_k^T \mathbf{W}_k \mathbf{V}_k \mathbf{y}_k, \tag{4.23}$$

where

$$\mathbf{V}_k \triangleq \operatorname{diag}(v_{i_1}, \ldots, v_{i_k}) = \operatorname{diag}\left(1/\sigma_{i_1}^2, \ldots, 1/\sigma_{i_k}^2\right). \tag{4.24}$$

4.4.5 Local Model Assessment

The computational procedure outlined in the end of Section 4.4.2 raises the need for methods that estimate the MSE or similar measures. In Chapter 3, a number of classical goodness-of-fit criteria for selecting smoothing parameters, such as cross-validation, generalized cross-validation and C_p, were described. A problem with them in their original formulations, however, is that they are evaluated globally using estimates computed at different locations. In the model-on-demand framework it is more desirable to perform this assessment locally, since we would like to keep the estimate at x independent (in a computational sense) of the estimates at other points. Fortunately, it turns out that this can be achieved by means of *localized* versions of the above mentioned methods.

For a fixed x, consider a local polynomial fit using k neighbors. Let as usual $\hat{\beta} = \hat{\beta}(x)$ denote the fitted coefficients, and let

$$\bar{m}(X_i, k) \triangleq \mathcal{B}^T(X_i - x)\hat{\beta}, \qquad i \in \Omega_k(x), \tag{4.25}$$

denote the estimated local polynomial within the neighborhood $\Omega_k(x)$ centered at x. The idea is now to replace the separately fitted values $\hat{m}(X_i, k)$ in the global methods with the values of (4.25) when forming the goodness-of-fit functions, and weight their influences with the smoothing weights $w_i(x)$ in order to emphasize the locality (Cleveland and Loader, 1994; Fan *et al.*, 1996).

Let us for the sake of completeness now present localized versions of the methods in Section 3.4.1. Which is the best is much depending on the application, although some of them have more appealing properties than others. We will leave this choice as an user option in the algorithms that follow.

Localized Cross-validation

The localized weighted version of the cross-validation criteria (3.43) is,

$$CV(x, k) \triangleq \frac{\sum_{i \in \Omega_k(x)} w_i(x)\,(Y_i - \bar{m}_{-i}(X_i, k))^2}{\sum_{i \in \Omega_k(x)} w_i(x)}, \tag{4.26}$$

where $\bar{m}_{-i}(X_i, k)$ denotes the leave-i-out local model fitted at x and evaluated at X_i, and $w_i(x)$ are the smoothing weights. The weighting (4.26) guarantees that the residuals close to x receive larger attention in the criterion than those located far away. As in the global case, there is a short-cut so that the leave-out residuals

$$\varepsilon_i^{(-i)} \triangleq Y_i - \bar{m}_{-i}(X_i, k) \tag{4.27}$$

can be computed from the local residuals, $\varepsilon_i = Y_i - \bar{m}(X_i, k)$, without recalculating the regression parameters for each excluded point. This follows from the relation

$$\varepsilon_i^{(-i)} = \frac{\varepsilon_i}{1 - \operatorname{infl}(x, X_i)}, \tag{4.28}$$

where the *local influence function* (Loader, 1997) is defined according to

$$\operatorname{infl}(x, X_i) \triangleq \mathcal{B}^T(X_i - x)\left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k\right)^{-1} \mathcal{B}(X_i - x) w_i(x). \tag{4.29}$$

We thus end up with the expression

$$CV(x, k) = \frac{1}{\operatorname{tr}(\mathbf{W}_k)} \sum_{i \in \Omega_k(x)} w_i(x) \left(\frac{Y_i - \bar{m}(X_i, k)}{1 - \operatorname{infl}(x, X_i)}\right)^2. \tag{4.30}$$

Localized Generalized Cross-validation

Recall from Chapter 3 that generalized cross-validation was obtained from cross-validation by replacing the influence function with its mean value. The locally weighted average of the influence function in equation (4.29) is

$$\frac{\sum_{i \in \Omega_k(x)} w_i(x) \operatorname{infl}(x, X_i)}{\sum_{i \in \Omega_k(x)} w_i(x)} = \frac{1}{\operatorname{tr}(\mathbf{W}_k)} \operatorname{tr}\left(\left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k\right)^{-1} (\mathbf{X}_k^T \mathbf{W}_k^2 \mathbf{X}_k)\right). \tag{4.31}$$

By replacing the influence values in (4.30) with this quantity, one obtains a local generalized cross-validation statistics similar to the global case. That is,

$$GCV(x, k) \triangleq \operatorname{tr}(\mathbf{W}_k) \frac{\sum_{i \in \Omega_k(x)} w_i(x) \left(Y_i - \bar{m}(X_i, k)\right)^2}{\operatorname{tr}(\mathbf{W}_k) - \operatorname{tr}\left(\left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k\right)^{-1} \mathbf{X}_k^T \mathbf{W}_k^2 \mathbf{X}_k\right)^2}. \tag{4.32}$$

Localized C_p

A local version of Mallows C_p criterion (3.53) can be obtained by considering the localized risk function

$$R(x, k) = \frac{\sum_{i \in \Omega_k(x)} w_i(x) \operatorname{E}\left(\bar{m}(X_i, k) - m(X_i)\right)^2 / \sigma_i^2}{\sum_{i \in \Omega_k(x)} w_i(x)}. \tag{4.33}$$

Performing similar manipulations as in Section 3.4.1 and allowing an arbitrary penalty $\alpha \geq 2$ on the variance term to prevent the criterion to find spurious features for small neighborhoods, one obtains the local generalized Mallows C_p with variance penalty α (Cleveland and Loader, 1994);

$$CP(x, k) \triangleq \frac{1}{\operatorname{tr}(\mathbf{W}_k)} \Bigg(\sum_{i \in \Omega_k(x)} \frac{w_i(x)}{\sigma_i^2} (Y_i - \bar{m}(X_i, k))^2 - \operatorname{tr}(\mathbf{W}_k) + \alpha \operatorname{tr}\left((\mathbf{X}_k^T \mathbf{V}_k \mathbf{W}_k \mathbf{X}_k)^{-1} \mathbf{X}_k^T \mathbf{W}_k^2 \mathbf{V}_k \mathbf{X}_k \right) \Bigg), \tag{4.34}$$

where $\mathbf{V}_k$ as in (4.24) is a diagonal matrix with entries $1/\sigma_i^2$. Note that this criterion requires that the noise variance is known or estimated from data.

Localized AIC

A localized variant of the AIC criterion in Chapter 3 can be obtained by replacing tr($\mathbf{H}$) with a local degrees-of-freedom measure. The local weighted average of the influence values is given by (4.31). In accordance with (3.47) we hence obtain

$$AIC(x,k) \triangleq \frac{\sum_{i\in\Omega_k(x)} w_i(x)(Y_i - \bar{m}(X_i,k))^2}{\mathrm{tr}(\mathbf{W}_k)} \times \exp\left(\frac{\alpha\,\mathrm{tr}\left(\left(\mathbf{X}_k^T\mathbf{W}_k\mathbf{X}_k\right)^{-1}(\mathbf{X}_k^T\mathbf{W}_k^2\mathbf{X}_k)\right)}{\mathrm{tr}(\mathbf{W}_k)}\right), \qquad (4.35)$$

where $\alpha \geq 2$ similar to (4.34) provides an extra penalty on the variance term.

Localized FPE

As in the global case it is also possible estimate the prediction error using a localized *final prediction error* criterion

$$FPE(x,k) \triangleq \frac{\sum_{i\in\Omega_k(x)} w_i(x)(Y_i - \bar{m}(X_i,k))^2}{\mathrm{tr}(\mathbf{W}_k)} \times \frac{2\,\mathrm{tr}(\mathbf{W}_k) + \alpha\,\mathrm{tr}\left(\left(\mathbf{X}_k^T\mathbf{W}_k\mathbf{X}_k\right)^{-1}(\mathbf{X}_k^T\mathbf{W}_k^2\mathbf{X}_k)\right)}{2\,\mathrm{tr}(\mathbf{W}_k) - \alpha\,\mathrm{tr}\left(\left(\mathbf{X}_k^T\mathbf{W}_k\mathbf{X}_k\right)^{-1}(\mathbf{X}_k^T\mathbf{W}_k^2\mathbf{X}_k)\right)}. \qquad (4.36)$$

Here the quantity $\mathrm{tr}(\mathbf{H})/N$ in (3.49) has been replaced by (4.31).

Hypothesis Test

The methods described above are all variants of the cross-validation and penalizing ideas. Bontempi *et al.* (1998) instead consider a hypothesis test approach when determining the appropriate size of the local neighborhood. By introducing the vector of cross-validation residuals for a model of size k,

$$E_k^{cv} \triangleq \left(\varepsilon_{i_1}^{(-i_1)} \quad \ldots \quad \varepsilon_{i_k}^{(-i_k)}\right)^T, \qquad (4.37)$$

they formulate the null hypothesis H_0 that E_k^{cv} and E_{k+1}^{cv} belong to the same distribution. When H_0 is rejected is it assumed that the model of size $k+1$ is significantly worse than the model of size k, i.e., that a significant increase in modeling bias is detected.

The hypothesis is evaluated using a nonparametric permutation test, which assumes, for each i, that the residuals can be assigned to the two vectors E_k^{cv} and E_{k+1}^{cv} with the same probability. If H_0 is true, this is equivalent to the assumption that the difference Δ_i

between residuals at position i in the two vectors, is to be as likely positive as negative. The null hypothesis H_0 is tested against some alternative hypothesis H_1 by computing the value $D = \sum_i \Delta_i$, and assuming that D is an instance of a random variable D^*. The sampling distribution for D^* is estimated using a randomization procedure similar to the bootstrap principle (Efron and Tibshirani, 1993), and a one-tailed test determines whether or not the hypothesis has to be rejected.

In our opinion the hypothesis test seems to be too time-consuming to be a useful alternative for applications with timing constraints.

4.4.6 Automatic Neighborhood-Size Selection

The localized goodness-of-fit criteria derived in the preceeding subsection provide us with information of the quality of the fit for a given neighborhood size k. This enables *adaptive* and *automatic* determination of the most appropriate degree of smoothing in the following way: Fit locally at x using a number of different neighborhood sizes, k, and and choose the optimal k as the one that has the lowest cost for a given goodness-of-fit measure, i.e., the one that reflects a good balance between bias and variance. Let us for the sake of illustrational clarity demonstrate this approach with a scalar example.

Example 4.1 **Adaptive smoothing of the Dopler function**

To show how the adaptive neighborhood size selection scheme works in practice, we consider estimation of the so-called *Dopler function*,

$$m(x) = 20\sqrt{x(1-x)}\,\sin\left(2\pi\,\frac{1.05}{x+0.05}\right), \qquad x \in [0,\ 1],$$

which earlier has been used by Donoho and Johnstone (1994) in wavelet applications. It has also been studied by Fan and Gijbels (1996) and Loader (1997).

A dataset $\{(Y_i, X_i)\}$ consisting of $N = 2048$ points was generated according to the regression model (4.1). The predictor variables was chosen to be equidistantly distributed on [0, 1] and the corresponding Y's were distorted by additive Gaussian noise with unit variance. The resulting data are shown in Figure 4.4.

The regression curve was estimated on grid of x-values using a local quadratic estimator and the tricube kernel. The bandwidth was selected locally for each estimation point using the localized C_p criterion (4.34) with $\hat{\sigma}^2 = 1$. The estimated result along with the true regression curve shown in Figure 4.5 (a). The associated adaptively selected bandwidths are shown in Figure 4.5 (b).

As expected the localized C_p criterion selects smaller bandwidths where the variability of the true function is high, i.e., for small x. For large x the true curve becomes smooth which enables the possibility of using larger bandwidths there. ❑

A potential problem with the adaptive approach is clearly illustrated here: The selected bandwidths are usually quite rough regarded as a function of x. The curve in Figure 4.5

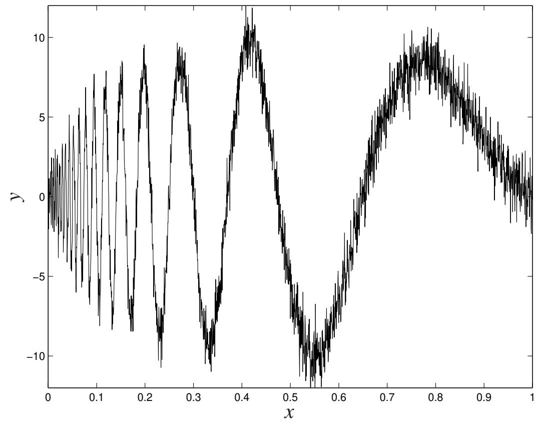


Figure 4.4 *The Dopler function dataset.*

(b) represents a quite typical behavior in this context. However, this is the price that has to be paid for the localization. A smoother, less variable bandwidth function $h(x)$ can be achieved by averaging the goodness-of-fit measure over neighboring points in a MISE-like way, but this will in general increase the computational burden. Another possibility is to increase the penalty α on the variance term in the bandwidth selection criteria as was demonstrated in Section 4.4.5.

4.4.7 Minimizing Goodness-of-Fit Measures

The adaptive method outlined in the preceeding subsection requires that local fits are computed for a number of different neighborhood sizes, and that the one with the lowest goodness-of-fit cost provides the final choice of k.

Computing estimates at a large number of neighborhood sizes using a novel implementation of (4.16) can be a very time consuming task, though. Considerable speedups can be obtained using updating ideas when computing the estimate or using exponential step length for the bandwidth as we shall see next.

Uniform or Exponentially Decaying Kernels: RLS

When using a *uniform* or *exponentially shaped* weighting function, i.e., when consecutive weights are related through the property

$$w_i(x) = \lambda_i \cdot w_{i-1}(x), \tag{4.38}$$

the parameter estimate $\hat{\beta}$ following from (4.16) can be efficiently computed for increasing neighborhood sizes by means of the recursive least squares (RLS) algorithm, see, e.g.,

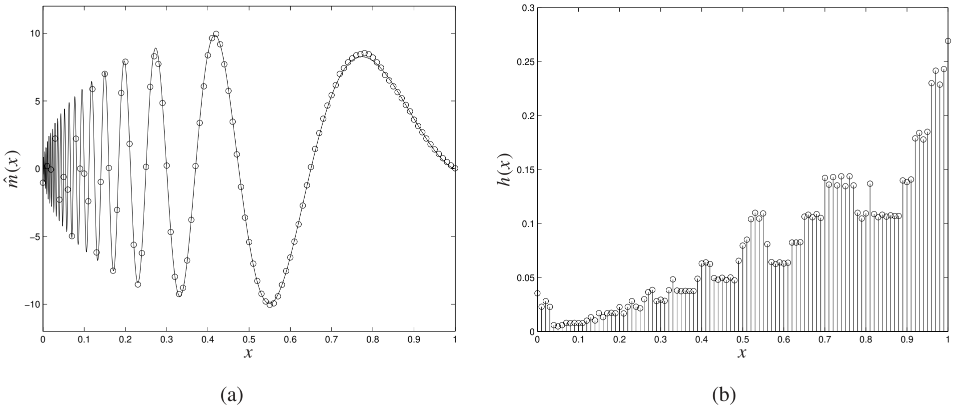


Figure 4.5 *(a) Estimated function values (circles) and true function (solid) for the Dopler example. (b) Locally selected bandwidths.*

Ljung (1999);

$$\hat{\beta}^{(k)} = \hat{\beta}^{(k-1)} + L_k\left(Y_k - \mathcal{B}^T(X_k - x)\hat{\beta}^{(k-1)}\right), \tag{4.39a}$$

$$L_k = \frac{P_k\mathcal{B}(X_k - x)}{\lambda_k + \mathcal{B}^T(X_k - x)P_{k-1}\mathcal{B}(X_k - x)}, \tag{4.39b}$$

$$P_k = \frac{1}{\lambda_k}\left(P_{k-1} - \frac{P_{k-1}\mathcal{B}(X_k - x)\mathcal{B}^T(X_k - x)P_{k-1}}{\lambda_k + \mathcal{B}^T(X_k - x)P_{k-1}\mathcal{B}(X_k - x)}\right), \tag{4.39c}$$

where $\hat{\beta}^{(k)}$ denotes the estimate based on k data. According to Gustafsson (1992, Appendix B.3), the sum of squared residuals can also be updated incrementally as the neighborhood grows, since,

$$\sum_{i=1}^{k}\left(Y_i - \mathcal{B}^T(X_i - x)\hat{\beta}^{(k)}\right)^2 + (\beta - \hat{\beta}^{(k)})^T P_0^{-1}(\beta - \hat{\beta}^{(k)}) = \sum_{i=1}^{k}\frac{\left(Y_i - \mathcal{B}^T(X_i - x)\hat{\beta}^{(i-1)}\right)^2}{\mathcal{B}^T(X_i - x)P_{i-1}\mathcal{B}(X_i - x) + \lambda_k}, \tag{4.40}$$

where P_0 denotes the covariance matrix for the initial estimate $\hat{\beta}^{(0)}$.

By utilizing these expressions, both the parameter estimate and the goodness-of-fit measures described in Section 4.4.5 can be computed rather efficiently for consecutive neighborhood sizes. However, as already pointed out in Section 3.2.1, the uniform weighting function suffers from some unsuitable properties that motivate avoiding it: As new observations are brought into the expanding neighborhood, they are assigned the same weight as the old, accumulated measurements. This normally results in large variability, both in the estimate and in the goodness-of-fit measure. However, this variability can be reduced by using weighting functions that smoothly decay to zero, such as the tricube kernel (3.9).

General Kernels: Exponential Updating

The updating methodology works when the weighting $w_i(x)$ is constant or decays exponentially. When using general kernels, the weights typically change with the location of data, and updating cannot be used. Brute force computation of estimates and goodness-of-fit for a large number of neighborhood sizes at each prediction point x using the weighted least squares formula (4.16) can be very expensive in terms of computational effort. In addition, the computational cost for this increases drastically as the neighborhood size grows. A remedy is to update the neighborhood size (or the bandwidth) exponentially during the minimization stage. In the LOCFIT implementation, Loader (1997) considers the following approach for local, adaptive bandwidth selection:

Step 1: Fit at a very small bandwidth h_0, close to the smallest bandwidth for which a well defined estimate is obtained.

Step 2: Increase the bandwidth exponentially according to

$$h_i = C_h \cdot h_{i-1},$$

where $C_h > 1$, and compute the corresponding fits. This procedure is repeated until a goodness-of-fit measure fails at a low significance level.

Step 3: At the bandwidth with lowest goodness-of-fit cost in Step 2, perform a finer search for the final bandwidth.

Loader (1997) suggests taking

$$C_h = 1 + \frac{0.3}{d}, \tag{4.41}$$

where d, as usual, denotes the dimension of the regressor space. This results in smaller updates of the bandwidth as the dimension grows, which seems reasonable.

Example 4.2

Consider again the Dopler dataset treated in Example 4.1. Since the tricube kernel was used as weighting scheme in this case, we have to resort to the exponential updating approach outlined above for speeding up the estimation. The C_p-values for $x = 0.3$ as function of the bandwidth h is shown in Figure 4.6.

The plot indicates that the minimizing bandwidth is located somewhere around $h = 0.023$ which corresponds to

$$k = 2 \cdot 0.023 \cdot 2048 \approx 90$$

neighboring data points. ❑

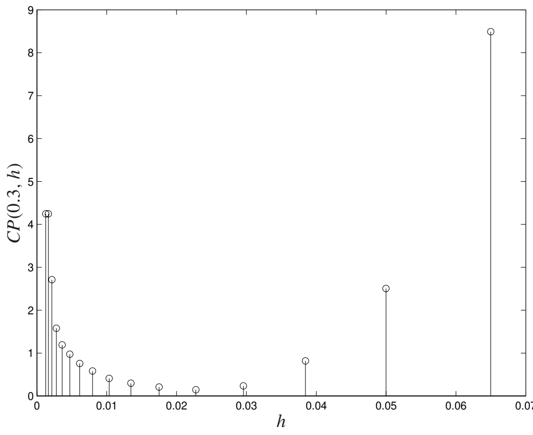


Figure 4.6 *Values of the C_p criterion when estimating the dopler function in Example 4.1 at $x = 0.3$.*

4.4.8 Noise Variance Estimation

In some situations it might be required to estimate the noise variance. A standard estimate of the noise variance in parametric modeling is (see, e.g., Ljung (1999, Lemma II.1));

$$\hat{\sigma}^2 = \frac{\sum_i (Y_i - m(X_i, \theta))^2}{N - \dim \theta} = \frac{\frac{1}{N} \sum_i (Y_i - m(X_i, \theta))^2}{1 - \frac{\dim \theta}{N}} \tag{4.42}$$

By replacing the average of the squared residuals and the degrees-of-freedom measure, $\dim \theta$, in (4.42) with their local counterparts, we obtain the local noise variance estimate

$$\hat{\sigma}^2(x) = \frac{\sum_i w_i(x)\,(Y_i - \bar{m}(X_i))^2}{\operatorname{tr}(\mathbf{W}_k) - \operatorname{tr}\left(\left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k\right)^{-1} (\mathbf{X}_k^T \mathbf{W}_k^2 \mathbf{X}_k)\right)}, \tag{4.43}$$

which is also consistent with (3.55). It is easily shown that (4.43) reduces to (4.42) in case of uniform weighting, i.e., when $w_i(x) = 1/N$. As noted by Fan and Gijbels (1996), the expression (4.43) is valid only under the assumption that $\bar{m}(x)$ is unbiased. This is rarely the case, so variance estimates are usually computed using small neighborhoods where the bias is small.

Estimates of the noise variance are useful when using the C_p criterion, and when computing approximate confidence bands as will be discussed in Section 4.7. Another application that needs this quantity will be presented in the next section.

4.4.9 Summary

To summarize all the different aspects of local polynomial estimation that have been discussed so far in this chapter, we now propose an algorithm that is possible to use as estimation procedure in the model-on-demand context.

ALGORITHM 4.1 Model-on-demand estimator based on local regression

Inputs: A database of observations $\{(Y_i, X_i)\}_{i=1}^N$, an estimation point x, and specifications in terms of polynomial order, kernel, distance function $d(X_i, x)$ on the regressor space, and goodness-of-fit measure.
Outputs: An estimate $\hat{m}(x)$, an estimate of its variance, and an estimate of the noise variance $\hat{\sigma}^2(x)$.

Step 1: Retrieve relevant data from the database and sort them in ascending order according to the distance from x.

Step 2: Perform model fits according to (4.5a) for increasing neighborhoods $\Omega_k(x)$ and record the parameter values $\hat{\beta}^{(k)}$ along with the corresponding goodness-of-fit cost. This procedure is repeated until the goodness-of-fit measure fails at a low level of significance or a maximum neighborhood size $k_{\max}$ is exceeded. For general kernels the neighborhood radius is updated in an exponential manner, otherwise the recursive updating (4.39) is used.

Step 3: Search for the neighborhood size k_{opt} with lowest cost according to the stored goodness-of-fit values from Step 2.

Step 4: From the corresponding parameter vector value $\hat{\beta}^{(k_{\text{opt}})}$, extract and return the estimate $\hat{m}(x)$ as shown in equation (4.5b). The noise variance $\hat{\sigma}^2(x)$ is obtained from (4.43) using a small neighborhood, and $\operatorname{Var}\hat{m}(x) = \hat{\sigma}^2(x)\mathbf{w}_k^T\mathbf{w}_k$.

Here the tricube kernel is used as default weighting. A default choice of distance function is to select the scaling matrix according to the inverse covariance matrix of the regressors. If variance information is available, a good choice for goodness-of-fit is the C_p criterion (4.34). Otherwise use the *AIC* or *FPE* measures defined in (4.35) and (4.36). Here the penalty α provides an extra degree of freedom in controlling the variability of the estimates. The default value is $\alpha = 2$. Increasing it normally results in a less variable estimate that however may miss detailed features in data. ❑

Since the estimator is formed using standard local polynomial assumptions, its asymptotic properties coincides with (3.70)–(3.72).

4.5 MOD-estimation from an Optimization Perspective

In the preceeding section we assumed that the weights $w_i = w_i(x)$ were explicitly determined by a kernel function of fixed form and that the fit was tuned by optimizing the size (and shape) of the local neighborhood. In this section we shall relax these assumptions and take a completely different point of view. We will instead allow the the weights to be unknown free variables, and optimize them in order to minimize the local MSE.

Consider again the local modeling procedure (4.3) with a local linear model structure and a quadratic norm. Using the notations (4.13)–(4.15), the solution to the associated weighted least squares problem (4.5a) is given by

$$\begin{aligned}\hat{\beta} &= \left(\mathbf{X}_k^T \mathbf{W}_k \mathbf{X}_k\right)^{-1} \mathbf{X}_k^T \mathbf{W}_k \mathbf{y}_k \\ &= \left(\sum_{i\in\Omega_k(x)} w_i \mathcal{B}(X_i - x)\mathcal{B}^T(X_i - x)\right)^{-1} \sum_{i\in\Omega_k(x)} w_i \mathcal{B}(X_i - x) Y_i \\ &= \left(\sum_{i\in\Omega_k(x)} w_i \begin{pmatrix} 1 & (X_i - x)^T \\ (X_i - x) & (X_i - x)(X_i - x)^T \end{pmatrix}\right)^{-1} \sum_{i\in\Omega_k(x)} w_i \begin{pmatrix} 1 \\ X_i - x \end{pmatrix} Y_i. \end{aligned} \tag{4.44}$$

If we for a moment assume that the weights satisfy

$$\sum_{i\in\Omega_k(x)} w_i = 1, \tag{4.45a}$$

$$\sum_{i\in\Omega_k(x)} w_i (X_i - x) = 0, \tag{4.45b}$$

this reduces to

$$\begin{aligned}\hat{\beta} &= \begin{pmatrix} 1 & 0 \\ 0 & \sum_{i\in\Omega_k(x)} w_i (X_i - x)(X_i - x)^T \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i\in\Omega_k(x)} w_i Y_i \\ \sum_{i\in\Omega_k(x)} w_i (X_i - x) Y_i \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & \left(\sum_{i\in\Omega_k(x)} w_i (X_i - x)(X_i - x)^T\right)^{-1} \end{pmatrix} \begin{pmatrix} \sum_{i\in\Omega_k(x)} w_i Y_i \\ \sum_{i\in\Omega_k(x)} w_i (X_i - x) Y_i \end{pmatrix}. \end{aligned} \tag{4.46}$$

That is, the estimate can be written in the linear smoother form

$$\hat{m}(x) = \hat{\beta}_0 = \sum_{i\in\Omega_k(x)} w_i Y_i = \mathbf{w}^T \mathbf{y}_k, \tag{4.47}$$

with

$$\mathbf{w} \overset{\Delta}{=} \begin{pmatrix} w_{i_1} & w_{i_2} & \ldots & w_{i_k} \end{pmatrix}^T. \tag{4.48}$$

The assumptions that were made in (4.45) can hence be justified, since they are consistent with the property (3.22) for the equivalent weights, i.e., that linearity of the underlying regression function is preserved.

4.5.1 Optimizing the Weights

As shown above, the local linear model reduces to a weighted average of the observations belonging to the neighborhood, provided the weight sequence satisfies the conditions (4.45). However, we have still not said anything about how to construct the actual weight sequence.

Since the prediction will be computed at one particular point x, it is natural to investigate the pointwise MSE measure (4.19) as a function of the weights, and on the basis of this determine how to choose the weights in order to minimize this measure. The MSE can as earlier shown, be decomposed into a squared bias term and a variance term according to

$$\mathrm{MSE}\left(\hat{m}(x,\mathbf{w})\right) = (b(x,\mathbf{w}))^2 + v(x,\mathbf{w}). \tag{4.49}$$

Here we have added $\mathbf{w}$ as argument to the estimator (instead of as earlier h), to emphasize the fact that the weights contained in $\mathbf{w}$ now are considered to be free variables.

Bias

Similar to the derivation in Section 3.3.2, the bias error of the linear smoother (4.47) is given by

$$\begin{aligned} b(x,\mathbf{w}) &\triangleq \mathrm{E}\,\hat{m}(x,\mathbf{w}) - m(x) = \mathrm{E}\sum_{i\in\Omega_k(x)} w_i Y_i - m(x) \\ &= \sum_{i\in\Omega_k(x)} w_i\big(m(X_i) - m(x)\big). \end{aligned} \tag{4.50}$$

By assuming that $m(\cdot)$ is at least two times differentiable everywhere, it can be expanded in a second order Taylor series expansion around x,

$$m(X_i) - m(x) = \mathcal{D}_m^T(x)(X_i - x) + \frac{1}{2}(X_i - x)^T\mathcal{H}_m(x)(X_i - x) + \dots, \tag{4.51}$$

where $\mathcal{D}_m(\cdot)$ and $\mathcal{H}_m(\cdot)$ denote the Jacobian (first order derivative vector) and the Hessian (second order derivative matrix of m) respectively. The first order moment of $X_i - x$ vanishes as a consequence of (4.45b). It thus follows that

$$b(x,\mathbf{w}) \approx \frac{1}{2}\sum_{i\in\Omega_k(x)} w_i(X_i - x)^T\mathcal{H}_m(x)(X_i - x) = \mathbf{b}^T\mathbf{w}, \tag{4.52}$$

where

$$\begin{aligned} \mathbf{b} &\triangleq \begin{pmatrix} B_{i_1} & \dots & B_{i_k}\end{pmatrix}^T \\ &= \frac{1}{2}\begin{pmatrix}(X_{i_1} - x)^T\mathcal{H}_m(x)(X_{i_1} - x) & \dots & (X_{i_k} - x)^T\mathcal{H}_m(x)(X_{i_k} - x)\end{pmatrix}^T. \end{aligned} \tag{4.53}$$

Variance

In analogy with the derivation i Section 3.3.3, the variance of the estimator is

$$v(x,\mathbf{w}) \triangleq \mathrm{Var}\,\hat{m}(x,\mathbf{w}) = \mathrm{E}\left(\sum_{i\in\Omega_k(x)} w_i\epsilon_i\right)^2 = \sigma^2\sum_{i\in\Omega_k(x)} w_i^2 = \sigma^2\mathbf{w}^T\mathbf{w}. \tag{4.54}$$

Optimizing the Weights

When knowing the MSE of the estimator as a function of the weights **w**,

$$\mathrm{MSE}\left(\hat{m}(x,\mathbf{w})\right) = \left(\mathbf{b}^T\mathbf{w}\right)^2 + \sigma^2\mathbf{w}^T\mathbf{w} = \mathbf{w}^T(\mathbf{b}\mathbf{b}^T + \sigma^2 I)\mathbf{w}, \tag{4.55}$$

it seems reasonable to try to minimize this expression w.r.t. **w** under the constraints (4.45).

By means of the notation (4.14) for the local design matrix, the constraints can expressed in compact form as

$$\mathbf{X}_k^T\mathbf{w} = e_1. \tag{4.56}$$

We thus have the following optimization problem for the weights;

$$\boxed{\begin{array}{rl} \displaystyle\min_{\mathbf{w}} & \mathbf{w}^T\left(\mathbf{b}\mathbf{b}^T + \sigma^2 I\right)\mathbf{w} \\ \text{subject to} & \mathbf{X}_k^T\mathbf{w} - e_1 = 0. \end{array}} \tag{4.57}$$

This is a quadratic optimization problem with linear constraints, and it can therefore easily be solved as a linear system of equations. From basic calculus, we have the result that in order to minimize an object function with respect to a constraint function, the gradients of the two functions have to be parallel. By introducing so-called *Lagrange multipliers*,

$$\boldsymbol{\mu} \triangleq \begin{pmatrix}\mu_0 & \mu_1 & \dots & \mu_p\end{pmatrix}^T,$$

the constrained minimization of the MSE expression (4.57) can be stated in matrix form as

$$\begin{pmatrix}(\mathbf{b}\mathbf{b}^T + \sigma^2 I) & \mathbf{X}_k \\ \mathbf{X}_k^T & 0\end{pmatrix}\begin{pmatrix}\mathbf{w} \\ \boldsymbol{\mu}\end{pmatrix} = \begin{pmatrix}0 \\ e_1\end{pmatrix}. \tag{4.58}$$

This is a linear equation system in $k+d+1$ equations and $k+d+1$ variables, from which the weights w_i can be uniquely solved. Solving such a large equation system by *brute force* computations might seem to be a quite desperate thing to do. However, since the matrix in (4.58) has a simple structure, the solution can fortunately be computed more efficiently. From (4.58) we have

$$\begin{pmatrix}\mathbf{w} \\ \boldsymbol{\mu}\end{pmatrix} = \begin{pmatrix}A & \mathbf{X}_k \\ \mathbf{X}_k^T & 0\end{pmatrix}^{-1}\begin{pmatrix}0 \\ e_1\end{pmatrix}, \tag{4.59}$$

where $A \triangleq (\mathbf{b}\mathbf{b}^T + \sigma^2 I)$. From the block matrix inversion formula (Kailath, 1980), it then follows that

$$\begin{pmatrix}A & \mathbf{X}_k \\ \mathbf{X}_k^T & 0\end{pmatrix}^{-1} = \begin{pmatrix}A^{-1} - A^{-1}\mathbf{X}_k D^{-1}\mathbf{X}_k^T A^{-1} & A^{-1}\mathbf{X}_k D^{-1} \\ D^{-1}\mathbf{X}_k^T A^{-1} & D^{-1}\end{pmatrix}, \tag{4.60}$$

where

$$D = \mathbf{X}_k^T A^{-1}\mathbf{X}_k. \tag{4.61}$$

Hence the weights can be computed as

$$\mathbf{w} = A^{-1}\mathbf{X}_k D^{-1} e_1 = A^{-1}\mathbf{X}_k \left(\mathbf{X}_k^T A^{-1}\mathbf{X}_k\right)^{-1} e_1, \tag{4.62}$$

where

$$A^{-1} = (\mathbf{b}\mathbf{b}^T + \sigma^2 I)^{-1} = \frac{1}{\sigma^2}\left(I - \frac{\mathbf{b}\mathbf{b}^T}{\sigma^2 + \mathbf{b}^T\mathbf{b}}\right) \tag{4.63}$$

follows from the matrix inversion lemma (Kailath, 1980):

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}.$$

4.5.2 Properties of the Optimized Weight Sequence

It may be of interest to investigate how the second order derivative (Hessian) of the regression function and the noise variance affect the shape of the optimized weights in (4.62), both when estimating at interior points and within the boundary regions of the regressor space. For illustrational ease we will only consider scalar examples. However, generalization to higher dimensions is straightforward and causes no conceptual difficulties.

Interior Point

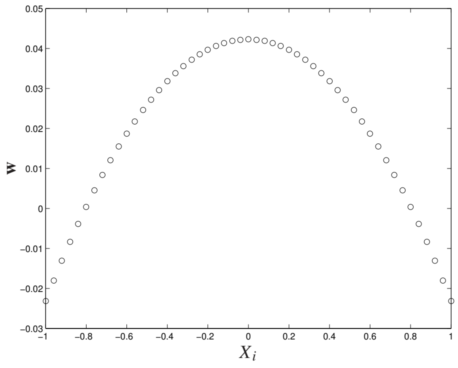
In Figure 4.7 (a)–(c), a scalar example is shown, where three weight sequences centered about $x = 0$ have been computed at 51 equidistantly distributed points, using a fixed noise variance but with different values of the second order derivative $m''(0)$.

For a large value of $m''(0)$, which corresponds to a regression function with a high degree of curvature at $x = 0$, we see that the weight sequence gets narrow (Figure 4.7 (a)), hence giving the measurements close to $x = 0$ a larger impact on the resulting estimate than those far away from this point. On the other hand, for a small value of $m''(0)$, which is equivalent to an approximately linear regression function, the weight sequence tends to weight all measurements almost equally (Figure 4.7 (c)).

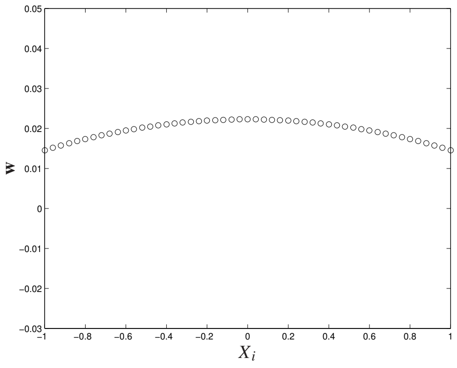
In Figure 4.8 (a)–(c), the same experiment as in Figure 4.7 is repeated using a fixed second order derivative but with different variances. As shown, a large variance, which corresponds to a high degree of uncertainty in the measurements, results in a weight sequence that pays almost equal attention to all measurements (Figure 4.8 (a)). A smaller variance, which means that the measurements are a little more reliable, again leads to a narrow weight sequence that will favour the local measurements (Figure 4.8 (c)).

It is interesting to note that the narrow weight sequences in Figure 4.7 (a) and Figure 4.8 (c) take negative values near the boundary. This is in contrast to the equivalent weights for the Nadaraya-Watson estimator and the local linear estimator, which always take non-negative values during estimation at interior points of the regressor space. However, by investigating the expression (4.62) for the optimized weights further, this can be explained as follows: Inserting (4.62) into (4.47) yields

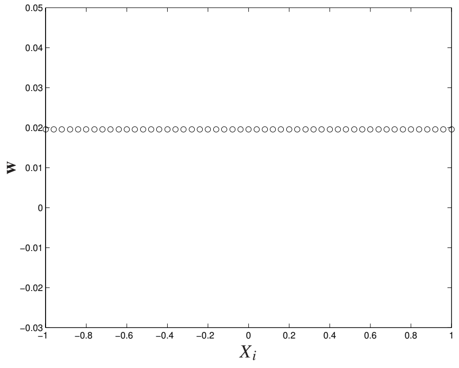
$$\hat{m}(x) = \mathbf{w}^T\mathbf{y}_k = e_1^T(\mathbf{X}_k^T A^{-1}\mathbf{X}_k)^{-1}\mathbf{X}_k^T A^{-1}\mathbf{y}_k, \tag{4.64}$$



(a) $m''(0) = 1,\ \sigma^2 = 0.1$



(b) $m''(0) = 0.1,\ \sigma^2 = 0.1$



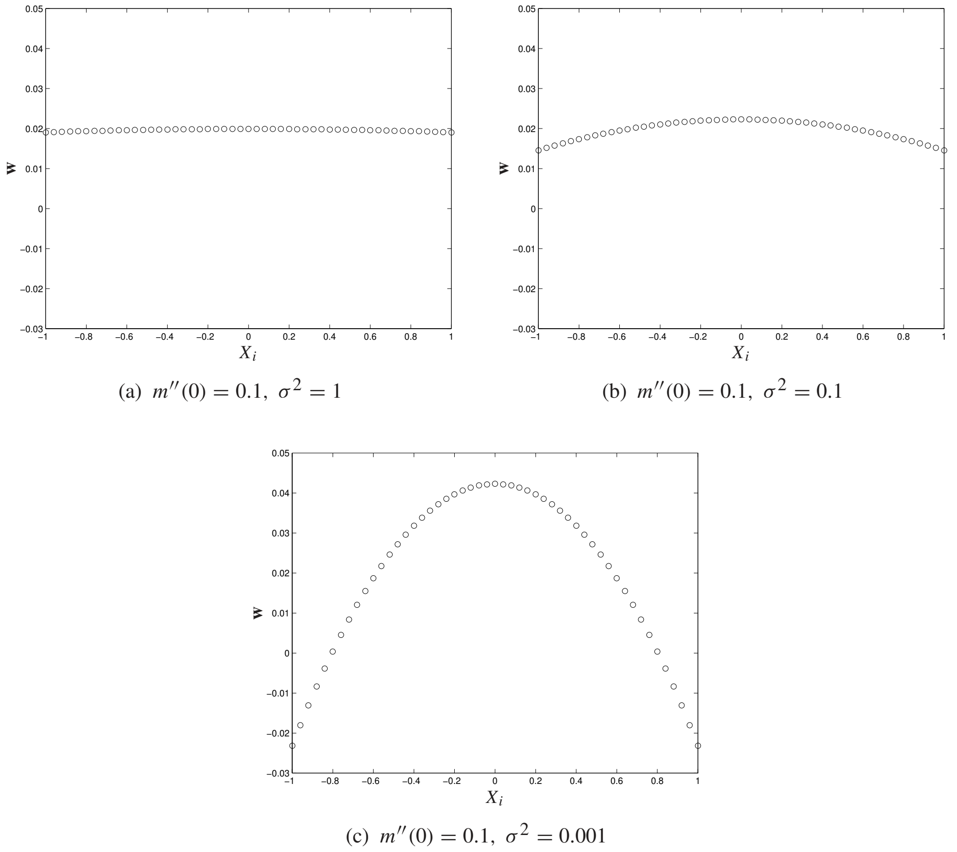
(c) $m''(0) = 0,\ \sigma^2 = 0.1$

Figure 4.7 *Optimized weights for fixed noise variance, but using different second order derivatives.*

i.e., the estimate (4.47) can be thought as being given by $\hat{\beta}_0$ in the weighted least squares fitting problem,

$$\hat{\beta} = \arg\min_{\beta}(\mathbf{y}_k - \mathbf{X}_k\beta)^T A^{-1}(\mathbf{y}_k - \mathbf{X}_k\beta). \tag{4.65}$$

The "weight" matrix A^{-1} given by (4.63) is symmetric and positive definite. Hence its



(a) $m''(0) = 0.1,\ \sigma^2 = 1$

(b) $m''(0) = 0.1,\ \sigma^2 = 0.1$

(c) $m''(0) = 0.1,\ \sigma^2 = 0.001$

Figure 4.8 *Optimized weights for fixed second order derivative, but using different noise variances.*

eigenvalues are real and positive, and its eigenvectors are orthogonal. Since

$$A^{-1}\mathbf{b} = \frac{1}{\sigma^2}\left(\mathbf{b} - \frac{\mathbf{b}\mathbf{b}^T\mathbf{b}}{\sigma^2 + \mathbf{b}^T\mathbf{b}}\right) = \frac{1}{\sigma^2}\left(1 - \frac{\mathbf{b}^T\mathbf{b}}{\sigma^2 + \mathbf{b}^T\mathbf{b}}\right)\mathbf{b}$$
$$= \frac{1}{\sigma^2}\left(\frac{\sigma^2}{\sigma^2 + \mathbf{b}^T\mathbf{b}}\right)\mathbf{b}, \tag{4.66}$$

it follows that $\mathbf{b}$ is an eigenvector of A^{-1} with eigenvalue

$$\lambda_{\mathbf{b}} = \frac{1}{\sigma^2}\left(\frac{\sigma^2}{\sigma^2 + \mathbf{b}^T\mathbf{b}}\right). \tag{4.67}$$

Let $u \perp \mathbf{b}$ denote one of the other eigenvectors of A^{-1}, and let λ_u be the corresponding eigenvalue. Due to the orthogonality we then have

$$A^{-1}u = \frac{1}{\sigma^2}u = \lambda_u u, \tag{4.68}$$

i.e., since $\mathbf{b}^T\mathbf{b} \geq 0$,

$$\lambda_u > \lambda_{\mathbf{b}}. \tag{4.69}$$

This can be interpreted as that residuals in the "direction" **b**, i.e., quadratic trends in the data, will be penalized less than those in other directions. Hence the estimator (4.47) can be seen as mixture between a local linear and a local quadratic smoother. By comparing the optimized weights in Figure 4.7 (a) and Figure 4.8 (c) with the equivalent weights shown in Figure 3.4 on page 32, this seems like a reasonable interpretation.

Boundary

In Figure 4.9 (a)–(c), the same experiment as in Figure 4.7 is repeated at the left boundary of the regressor space. Similarly to the equivalent weights for the local linear and local quadratic estimators displayed in Figures 3.5 (b) and 3.5 (c) on page 33, the optimized weights adapt to eliminate boundary bias effects.

4.5.3 Computing the Weights

Estimation using the optimized weights derived in Section 4.5.1, can be interpreted as a two-step procedure:

Step 1: Estimate the Hessian and the noise variance from the data belonging to the local neighborhood $\Omega_k(x)$.

Step 2: Compute the weights as the solution (4.62) to the constrained minimization problem (4.57), and form the estimate as a weighted average of the local responses.

In the framework of local polynomial regression given in Chapter 3, this is a variant of a plug-in method. However, in contrast to the methods proposed in the statistical literature, we do not rely on asymptotic expressions in the estimate.

The Hessian and noise variance estimation in Step 1 can be performed using local quadratic or local cubic regression in the same fashion as demonstrated in Section 4.4, and hence we face the same problem of determining the appropriate size for the local neighborhood $\Omega_k(x)$.

4.5.4 An Optimization-based Algorithm

In the optimization framework we are now able to propose a four-step algorithm which can be summarized as follows:

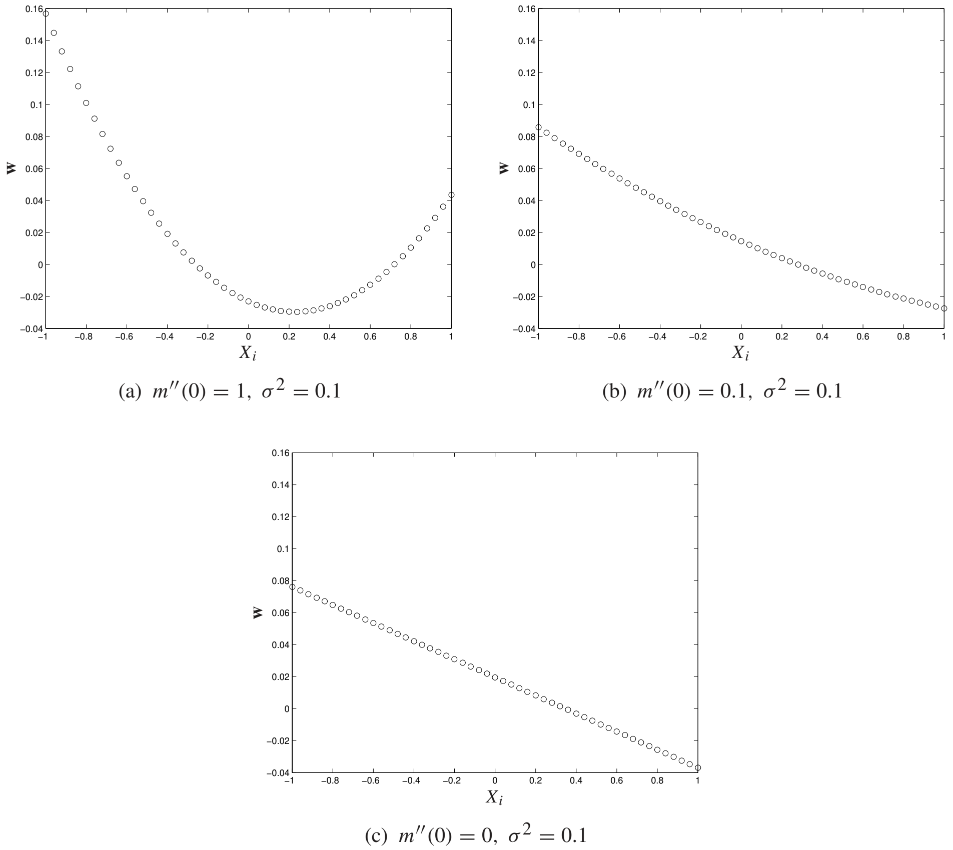


Figure 4.9 *Optimized weights at the left boundary for fixed noise variance, but when using different second order derivatives.*

ALGORITHM 4.2 Model-on-demand estimator based on optimization

Inputs: A database $\{(Y_i, X_i)\}_{i=1}^N$, an operating point x and a distance function $d(X_i, x)$ on the regressor space.
Outputs: An estimate $\hat{m}(x)$, and estimates of its bias and variance.

Step 1: Retrieve relevant data from the database and sort the regressors X_i in ascending order according to the distance from x.

Step 2: Estimate the Hessian $\mathcal{H}_m(x)$ and the local noise variance $\sigma^2(x)$ from the neighborhood $\Omega_k(x)$. This can be achieved by means of Algorithm 4.1 and a local quadratic model. The entries of $\mathcal{H}_m(x)$ is obtained from the last $d(d+1)/2$ elements of $\hat{\beta}^{(k_{\text{opt}})}$, and $\hat{\sigma}^2(x)$ follows from (4.43).

Step 3: Use the result from Step 2 to compute weights

$$\mathbf{w} = \mathbf{A}\mathbf{X}_k \left(\mathbf{X}_k^T \mathbf{A}\mathbf{X}_k\right)^{-1} e_1,$$

where

$$\mathbf{A} = \frac{1}{\sigma^2}\left(I - \frac{\mathbf{b}\mathbf{b}^T}{\sigma^2 + \mathbf{b}^T\mathbf{b}}\right).$$

Step 4: Form the resulting estimate as a weighted average of the corresponding response variables,

$$\hat{m}(x) = \mathbf{w}^T \mathbf{y}_k.$$

A bias estimate is obtained from (4.52) and the variance follows from (4.54).

The default values for tuning is the same as for Algorithm 4.1. ❏

In comparison with Algorithm 4.1, Algorithm 4.2 may increase the accuracy even when the function $m(\cdot)$ is not purely quadratic. This will be demonstrated in Section 4.6.

4.5.5 Asymptotic Properties of the Optimized Weights

In analogy with the kernel case in Chapter 3, it might be interesting to investigate the asymptotic properties of the optimization approach as the number of observations tends to infinity. It turns out that the derivation becomes quite messy in the multivariable case, so we choose to only perform the analysis for the scalar case. To be able to compare our result with the corresponding asymptotic result for kernel estimators presented in Section 3.3, we here consider the same design setting (i.e., fixed and equally spaced). The consistency of (4.47) and the rate of which the MSE tends to zero as a function of the sample size N are given in the following proposition, which was first stated in Stenman (1997).

Proposition 4.1 Consider the fixed equally spaced design regression model

$$Y_i = m(X_i) + \epsilon_i, \qquad i = 1, \dots, N$$

where $X_i = i/N$ and ϵ_i are i.i.d. random variables with zero mean and variance σ^2. Furthermore, assume that the following hold:

(i) The second order derivative $m''(x)$ is continuous on $[0,\ 1]$.

(ii) The neighborhood $\Omega_k(x)$ contains $k = k_N$ data and is defined as in (4.4) with $d(X_i, x) = |X_i - x|$.

(iii) The bandwidth parameter $h = h_N$ is a sequence satisfying $h_N \to 0$ and $Nh_N \to \infty$ as $N \to \infty$.

(iv) The estimation point x is located at the grid and is an interior point of the interval, i.e., $x = l/N$ for some integer l satisfying $h_N N \le l \le (1 - h_N)N$.

Let $\hat{m}(x)$ be a linear smoother according to (4.47) with optimized weights satisfying (4.45), and let the neighborhood size be chosen according to

$$k_N \simeq 2\left(\frac{15\,\sigma^2}{(m''(x))^2}\right)^{1/5} \cdot N^{4/5}. \tag{4.70}$$

Then

$$\mathrm{AMSE}\left(\hat{m}(x)\right) = \frac{3}{4}\left(\frac{(m''(x))^2\sigma^8}{15}\right)^{1/5} \cdot N^{-4/5}. \tag{4.71}$$

Proof Let h_N be the bandwidth, i.e., the distance from x that defines the neighborhood $\Omega_k(x)$. The number of data points belonging to $\Omega_k(x)$ is then $k = k_N = \lfloor 2h_N N \rfloor$.

The MSE was derived in Section 4.5.1 and is given by

$$\mathrm{MSE}\left(\hat{m}(x,\mathbf{w})\right) \simeq \left(\mathbf{b}^T\mathbf{w}\right)^2 + \sigma^2 \cdot \mathbf{w}^T\mathbf{w}. \tag{4.72}$$

Here the similarity follows as a consequence of the constraints (4.45) and a second order Taylor expansion of $m(\cdot)$ at x. However, the error made in the Taylor expansion vanishes asymptotically since $h_N \to 0$ as $N \to \infty$.

We now want to minimize the right hand side of (4.72) subject to the constraints (4.45). This can be done by introducing a Lagrange function $\mathcal{L}$ according to

$$\mathcal{L} = \frac{1}{2}\,\mathrm{MSE}\left(\hat{m}(x,\mathbf{w})\right) + \mu_0 \cdot (\mathbf{1}^T\mathbf{w} - 1) + \mu_1 \cdot (\boldsymbol{\alpha}^T\mathbf{w}) \tag{4.73}$$

where μ_i are Lagrange multipliers, $\mathbf{1}$ denotes a vector consisting of all ones and

$$\boldsymbol{\alpha} \triangleq \begin{pmatrix}\alpha_{i_1} & \dots & \alpha_{i_k}\end{pmatrix}^T = \begin{pmatrix}X_{i_1} - x & \dots & X_{i_k} - x\end{pmatrix}^T.$$

Taking partial derivatives of (4.73) results in

$$\frac{\partial \mathcal{L}}{\partial w_i} = \sigma^2 w_i + (\mathbf{b}^T\mathbf{w})B_i + \mu_0 + \alpha_i\mu_1 = 0, \qquad \forall i, \tag{4.74}$$

which is consistent with (4.58). Introduce the short-hand notation

$$\kappa = \mathbf{b}^T\mathbf{w} \tag{4.75}$$

for the bias error term in (4.72). Hence

$$w_i = -\frac{1}{\sigma^2}(\kappa B_i + \mu_0 + \alpha_i\mu_1), \tag{4.76}$$

and we get the equation system

$$\mathbf{1}^T\mathbf{w} = -\frac{1}{\sigma^2}(\kappa \cdot \mathbf{b}^T\mathbf{1} + \mu_0 \cdot k_N + \mu_1 \cdot \mathbf{1}^T\boldsymbol{\alpha}) = 1, \tag{4.77a}$$

$$\boldsymbol{\alpha}^T\mathbf{w} = -\frac{1}{\sigma^2}(\kappa \cdot \mathbf{b}^T\boldsymbol{\alpha} + \mu_0 \cdot \mathbf{1}^T\boldsymbol{\alpha} + \mu_1 \cdot \boldsymbol{\alpha}^T\boldsymbol{\alpha}) = 0, \tag{4.77b}$$

$$\mathbf{b}^T\mathbf{w} = -\frac{1}{\sigma^2}(\kappa \cdot \mathbf{b}^T\mathbf{b} + \mu_0 \cdot \mathbf{1}^T\mathbf{b} + \mu_1 \cdot \mathbf{b}^T\boldsymbol{\alpha}) = \kappa. \tag{4.77c}$$

for κ, μ_0 and μ_1. The odd moments of α_i in (4.77) vanish asymptotically, since

$$\mathbf{1}^T\boldsymbol{\alpha} = \sum_{i\in\Omega_k(x)} (X_i - x) = O(k_N/N) = O(h_N) \to 0, \tag{4.78}$$

and

$$\mathbf{b}^T\boldsymbol{\alpha} = \frac{m''(x)}{2} \sum_{i\in\Omega_k(x)} (X_i - x)^3 = O(k_N^3/N^3) = O(h_N^3) \to 0, \tag{4.79}$$

as $N \to \infty$. For the even moments of α_i we have

$$\mathbf{1}^T\mathbf{b} = \frac{m''(x)}{2} \sum_{i\in\Omega_k(x)} (X_i - x)^2 = m''(x) \sum_{k=1}^{k_N/2} \left(\frac{k}{N}\right)^2 \simeq \frac{m''(x)k_N^3}{24N^2}, \tag{4.80}$$

$$\boldsymbol{\alpha}^T\boldsymbol{\alpha} = \sum_{i\in\Omega_k(x)} (X_i - x)^2 \simeq \frac{k_N^3}{12N^2}, \tag{4.81}$$

and

$$\begin{aligned}\mathbf{b}^T\mathbf{b} &= \frac{(m''(x))^2}{4} \sum_{i\in\Omega_k(x)} (X_i - x)^4 = \frac{(m''(x))^2}{2} \sum_{k=1}^{k_N/2} \left(\frac{k}{N}\right)^4 \\ &\simeq \frac{(m''(x))^2 k_N^5}{320N^4}.\end{aligned} \tag{4.82}$$

See Appendix 4.A for the series formulas. Hence, when inserting equations (4.78)–(4.82) into (4.77), the following asymptotic solution is obtained

$$\kappa = \frac{30\, m''(x)k_N^2\sigma^2\, N^2}{720\,\sigma^2\, N^4 + (m''(x))^2 k_N^5} \tag{4.83a}$$

$$\mu_0 = -\frac{9\,\sigma^2\, \left((m''(x))^2 k_N^5 + 320\,\sigma^2\, N^4\right)}{4\,k_N\left(720\,\sigma^2\, N^4 + (m''(x))^2 k_N^5\right)} \tag{4.83b}$$

$$\mu_1 = 0 \tag{4.83c}$$

From (4.76) it now follows that

$$\mathbf{w} = -\sigma^{-2}(\kappa\cdot\mathbf{b} + \mu_0\cdot\mathbf{1} + \mu_1\cdot\boldsymbol{\alpha}) \simeq -\sigma^{-2}(\kappa\cdot\mathbf{b} + \mu_0\cdot\mathbf{1}). \tag{4.84}$$

The variance error is thus given by

$$\sigma^2\cdot\mathbf{w}^T\mathbf{w} \simeq -(\kappa\cdot\mathbf{w}^T\mathbf{b} + \mu_0\cdot\mathbf{w}^T\mathbf{1}) = -(\kappa^2 + \mu_0). \tag{4.85}$$

Hence (4.72) gives

$$\inf_{\mathbf{w}} \text{AMSE}\left(\hat{m}(x,\mathbf{w})\right) = -\mu_0 = \frac{9\,\sigma^2\,\left((m''(x))^2 k_N^5 + 320\,\sigma^2\,N^4\right)}{4\,k_N\left(720\,\sigma^2\,N^4 + (m''(x))^2 k_N^5\right)}. \tag{4.86}$$

By substituting k_N from (4.70) we thus finally arrive at

$$\inf_{\mathbf{w}} \text{AMSE}\left(\hat{m}(x,\mathbf{w})\right) \simeq \frac{3}{4}\left(\frac{(m''(x))^2\sigma^8}{15}\right)^{1/5} N^{-4/5}, \tag{4.87}$$

and (4.71) is proved. ■

Let us try to interprete the above result. The MSE formula in equation (4.86) is a decreasing function of k_N, but for the given value (4.70) it has a stationary point. This can be explained as follows: If $m(x)$ is a quadratic function, i.e., the second order derivative $m''(x)$ is constant for all x, the Taylor expansion will be valid over the entire interval. The optimal neighborhood should thus be chosen as $k_N = N$. However, the Taylor expansion is in most cases only valid locally, which requires that $k_N < N$ in order to guarantee that $h_N = \frac{k_N}{2N} \to 0$ as $N \to \infty$. Thus the stationary point will provide a good choice of k_N.

From this result we can make the following observations: Equation (4.71) indicates that the mean square error (MSE) tends to zero as the number of samples N tends to infinity, i.e., the optimized weight sequence is a consistent estimator for the true regression function. It also shows that the speed of which the MSE decays is in the order of $N^{-4/5}$, and that we thus achieve the same rate of convergence for the optimization approach, as using a local linear estimator with an Epanechnikov kernel. See Section 3.3.5 for comparison. From (4.70) we have that in order to obtain this convergence rate, the neighborhood size k_N has to be chosen in the order of $N^{4/5}$.

It is worth to point out that the result of the proposition, as in the kernel regression case, is valid only when x is an interior point of the interval. At the boundary the convergence rate usually gets slower.

Random Design

As discussed in Stenman (1997), it is also possible to motivate that Proposition 4.1 also holds for the random design case when the regressor variables X_i are uniformly distributed on the interval [0, 1]. Then the sums in equations (4.78) through (4.82) can be approximated with expectation according to

$$\sum_{i\in\Omega_k(x)} (X_i - x)^k \approx k_N \cdot \text{E}(X_i - x)^k,$$

where $X_i - x$ is uniformly distributed on $[-h_N,\ h_N]$. From Appendix 4.B, it thus follows that

$$\begin{aligned}
k_N \cdot \mathrm{E}(X_i - x) &= 0, \\
k_N \cdot \mathrm{E}(X_i - x)^2 &= k_N \frac{h_N^2}{3} = \frac{k_N^3}{12N^2}, \\
k_N \cdot \mathrm{E}(X_i - x)^3 &= 0, \\
k_N \cdot \mathrm{E}(X_i - x)^4 &= k_N \frac{h_N^4}{5} = \frac{k_N^5}{80N^4}.
\end{aligned}$$

Hence equations (4.78)–(4.82) still hold, which implies that the proposition is valid even for the uniform random design case. Since all other distributions, at least locally, can be viewed as uniform distributions, we have that the result to some extent also is applicable for general random design settings.

Asymptotic Shape of the Weighting

It may be of interest to investigate the shape of the optimal weight sequence in the asymptotic case. From (4.84) we have the expression

$$\begin{aligned}
w_i &\simeq -\frac{1}{\sigma^2}(\kappa\, B_i + \mu_0) = -\frac{\mu_0}{\sigma^2}\left(1 + \frac{\kappa}{\mu_0} B_i\right) \\
&= -\frac{\mu_0}{\sigma^2}\left(1 + \frac{\kappa\, m''(x)}{2\mu_0}(X_i - x)^2\right) \\
&= C_w \left(1 - \delta \left(\frac{X_i - x}{h_N}\right)^2\right). \qquad (4.88)
\end{aligned}$$

with

$$\delta = -\frac{\kappa\, m''(x)}{2\mu_0} = \frac{5}{3} \frac{(m''(x))^2 h_N^3 N}{(m''(x))^2 h_N^3 N + 10\sigma^2 / h_N^2}.$$

A plot of this function for the simple values $C_w = h_N = 1$, $\delta = 5/3$ (corresponding to $\sigma = 0$) and $x = 0$ is given in Figure 4.10 along with an unnormalized version of the quadratic Epanechnikov kernel. Both functions are of parabolic shape, but while the Epanechnikov kernel function is positive for all x in the interval, the optimized weight sequence takes negative values for

$$|x| \geq \sqrt{3/5} \approx 0.775.$$

This property is analogous to that of the equivalent weights for a local quadratic smoother as shown in Section 3.2.5. Kernel functions with such properties are usually referred to as *higher order kernels* in the statistical literature (Fan and Gijbels, 1996). From equations (4.88) and (3.8) we see that $\delta = 1$ results in an Epanechnikov-type weight sequence. In fact, this choice of δ directly corresponds to both equation (3.38) evaluated for the Epanechnikov

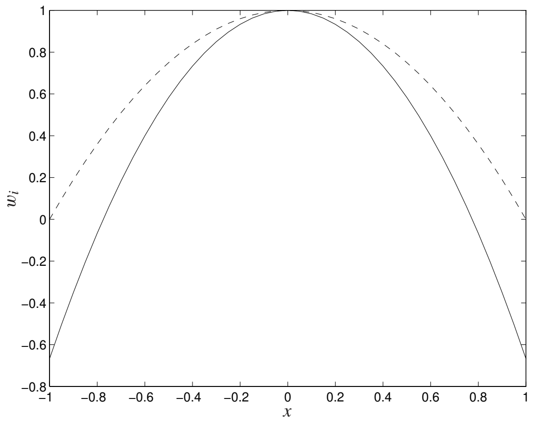


Figure 4.10 *Asymptotic shape of the optimized weight sequence (solid) and the Epanechnikov kernel (dashed).*

kernel, and the stationary point (4.70). The interpretation is that a requirement of positive weights results in a convergence rate corresponding to (3.37) and (4.71). A larger value of k_N gives $\delta > 1$, i.e., both positive and negative weights. This will decrease the bias even more and gives faster convergence. However, determining an exact expression for this rate will require that higher order derivatives are taken into account in the analysis.

4.5.6 Optimal Norm

As with the weights it is also possible to optimize the norm (or, rather, the scaling $\mathbf{M}$) that controls the distance function. Atkeson *et al.* (1997*a*) have after simulation studies found that an optimal choice of $\mathbf{M}$ for estimation at a point x is proportional to the Hessian of m:

$$\mathbf{M} \approx c \cdot \mathcal{H}_m(x). \tag{4.89}$$

That is, the norm adapts to the quadratic part of the true regression function. This is a quite natural result.

4.6 Polynomial Regression Versus Optimization

It may be interesting to compare the optimized weights used in Algorithm 4.2 with those obtained from Algorithm 4.1 or the local polynomial regression methods in Chapter 3. A very useful tool for doing such comparisons is the concept of equivalent weights introduced in Section 3.2.5.

Figure 4.11 illustrates the equivalent weights for the optimization approach and the local linear estimator (3.15) combined with an Epanechnikov kernel, when estimating the function

$$m(x) = \sin(2\pi x) \tag{4.90}$$

at $x = 0.27$ using simulated data generated according to the standard model (4.1) with $N = 100$, $X_i = i/N$ and ϵ_i chosen as Gaussian distributed noise with standard deviation $\sigma = 0.05$. The bandwidth is selected according to the AMISE measure (3.39) which results in $h = 0.1$. The optimized weights are represented by the dashed line and the equivalent local linear weights by the dashed-dotted line. The corresponding estimates are indicated with a circle and a cross respectively.

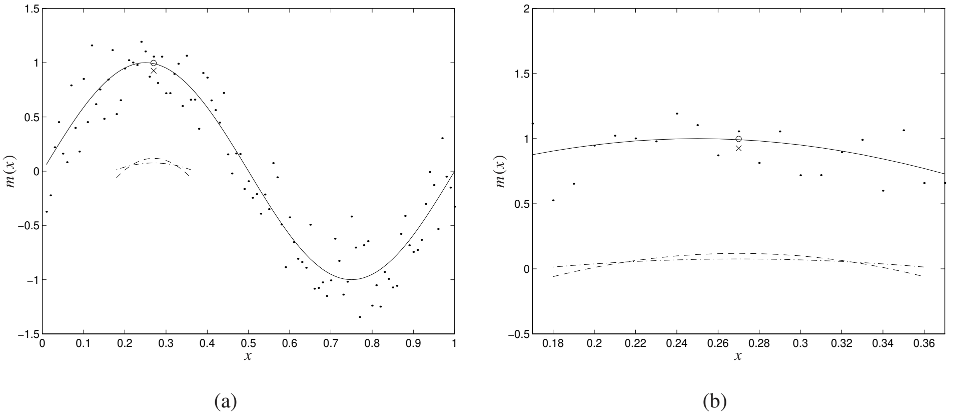


Figure 4.11 A comparison between the optimized weights and the equivalent local linear weights, when estimating the function (4.90) at $x = 0.27$. (a) True regression function (solid), simulated data (dots), optimized estimate (circle) and local linear estimate (cross). The equivalent weights for the optimized estimator (dashed) and the local linear estimator (dash-dotted) are also plotted. (b) Magnification of the plot in (a).

A Monte-Carlo simulation, where the estimates are averaged over 1000 different noise realizations, is shown in Table 4.1. For comparison a Nadaraya-Watson estimate of the type (3.14) has also been included. As indicated, the optimized weights give a slightly better result (about 10%) than the other two methods. We also see that the Nadaraya-Watson and the local linear estimates coincide. As pointed out in Chapter 3, this will always be the case when using fixed and equally spaced design, and when x is an interior point of the interval.

In Figure 4.12 the same experiment is repeated at the boundary point $x = 0$. As shown, the Nadaraya-Watson estimator results in a large bias error since the weights are always positive (recall the discussion in Section 3.2.5). The optimized and local linear weights which are allowed to take negative values give almost equivalent results.

The result of a Monte-Carlo simulation using 1000 iterations is summarized in Table 4.2. As indicated, the optimized and the local linear weights give essentially the same

Method	Absolute error
Optimized weights	0.064
Local linear estimator	0.070
Nadaraya-Watson estimator	0.070

Table 4.1 *Result of a Monte-Carlo simulation with 1000 iterations when estimating the function (4.90) from data at $x = 0.27$.*

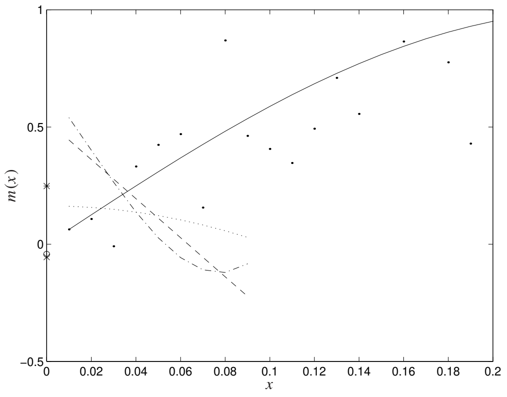


Figure 4.12 *A comparison between the optimized, local linear, and kernel weights, when estimating the function (4.90) at the boundary point $x = 0$. True regression function (solid), simulated data (dots), optimized estimate (circle), local linear estimate (cross), and Nadaraya-Watson estimate (star). The optimized weights (dashed), the local linear weights (dash-dotted), and the Nadaraya-Watson weights (dotted) are also plotted.*

Method	Absolute Error
Optimized weights	0.0091
Local linear estimator	0.0099
Nadaraya-Watson estimator	0.178

Table 4.2 *Result of a Monte-Carlo simulation with 1000 iterations when estimating the function (4.90) from data at the boundary point $x = 0$.*

prediction error near the boundary.

The conceptual difference between the optimization and kernel approaches can be illustrated as in Figure 4.13. Here the horizontal axis represents the number of data N

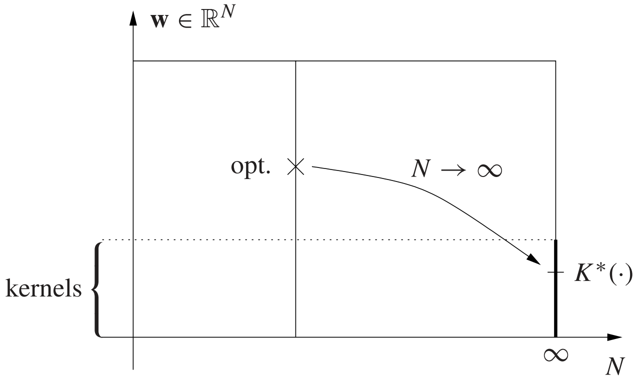


Figure 4.13 *The relation between weight optimization and optimal kernels.*

and the vertical axis all possible weight sequences. The Epanechnikov kernel represents a subset of these sequences (represented by the thick line in the figure) and is optimal under the assumption of an infinite number of data (Fan and Gijbels, 1996). The optimization approach is performed for a finite number of data (the cross in the figure). However, when the requirement of positiveness is put on the weights and N tends to infinity, the optimized weights approach the Epanechnikov kernel. This was shown in Section 4.5.5. The fact that the optimization is performed for a finite number of data might increase the accuracy. For instance, for the simple example above we gained about 10%. However, the optimization might in some situations be too time consumable to be usable in practice. Moreover, it relies on additional quantities like the Hessian and the noise variance which have to accurately estimated from data. This could be a problem in higher regressor dimensions since a quadratic model then implies many parameters to estimate. The conclusion is therefore that the optimization approach probably is to prefer only for a moderate number of dimensions.

4.7 Pointwise Confidence Intervals

All estimation methods described in this chapter provide pointwise estimates for each fitting point x. In certain situations, though, there might also be need for quantifying the errors associated with each such estimate. A useful diagnostic tool for such purposes is the concept of *confidence intervals*.

It is rather straightforward to derive approximate confidence interval for linear smoothers like (4.17) and (4.47). From Section 3.3.3 we have that the variance of smoother can be estimated as

$$\hat{v}(x) = \hat{\sigma}^2(x)\,\mathbf{w}^T\mathbf{w} = \hat{\sigma}^2(x)\|\mathbf{w}\|^2. \tag{4.91}$$

where $\hat{\sigma}^2(x)$ is an estimate of the noise variance obtained from for instance (4.43). Under normality assumptions it is thus possible to obtain approximate $(1-\alpha)100\%$ confidence bands as

$$I(x) = \left(\hat{m}(x) - \lambda_{\alpha/2}\,\hat{\sigma}(x)\|\mathbf{w}\|,\ \hat{m}(x) + \lambda_{\alpha/2}\,\hat{\sigma}(x)\|\mathbf{w}\|\right), \tag{4.92}$$

where $\lambda_{\alpha/2}$ denotes the $\alpha/2$ quantile of the normal distribution. For the commonly used requirement of 95% coverage it has the well-known value of $\lambda_{0.025} = 1.96$.

Note that equation (4.92) provides confidence intervals for the expected value $\mathrm{E}\,\hat{m}(x)$. To obtain true intervals for $m(x)$ some bias adjustment is required, see for example Fan and Gijbels (1996). However, it turns out that it is quite cumbersome to estimate the bias, so (4.92) provides a reasonable approximation. Consult Sun and Loader (1994) for more discussion around this.

Example 4.3

Recall the Dopler function introduced in Example 4.1. Figure 4.14 shows the estimated curve and 95% confidence intervals for a small part of the function. ❑

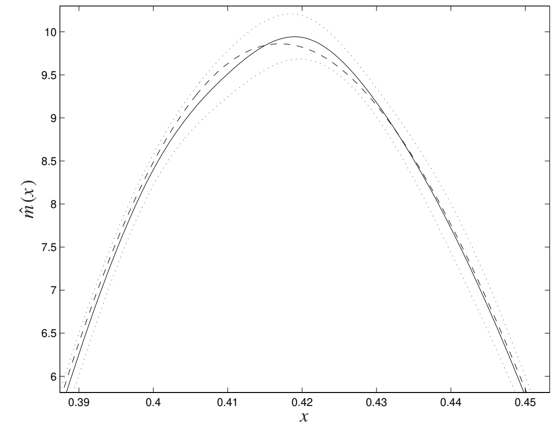


Figure 4.14 *Estimated Dopler function (solid line) and 95% confidence intervals (dotted lines). The dashed line is the true function.*

Another way of constructing confidence bands is based on the *bootstrap* principle (Efron and Tibshirani, 1993). The idea is to determine the distribution of $\hat{m}(x)$ empirically by utilizing a resampling procedure that generates "fake" datasets with approximately the same properties as the original data. An algorithm based on these ideas is given in Härdle (1990). A severe drawback with this approach, however, is that the resampling and associated re-estimation require large computational resources.

4.8 Data Management

So far, we have not paid much attention to the dataset searching problem. The model-on-demand idea relies on that neighboring data points can be retrieved in an easy and efficient way in order to compute the local prediction. Recent versions of the proposed algorithms are implemented in MATLAB (MathWorks, 1996) and use a sorting and searching routine that is essentially equivalent to linear search. For real industrial applications with large datasets, however, this is not acceptable.

Dataset searching is in general a problem since there are lack of good data structures that allow nearest neighbors to be retrieved fast and efficiently. The best known searching methods, such as the *k-d* tree approach described below, require a search that is exponential in the dimension of the regressor space, and they effectively reduce to *brute-force*, exhaustive searches for dimensions d greater than 20, on databases that consist of up to several million samples (Cybenko, 1996). However, research interest for methods that perform approximate neighborhood searching has been growing in recent years (Sadjadi, 1993).

4.8.1 k-d Trees

The *k-d tree* (Bentley, 1975; Samet, 1990) is a particular data structure for storing multidimensional data. The idea is to build up a tree structure by recursively dividing the regressor space into smaller cells or boxes.

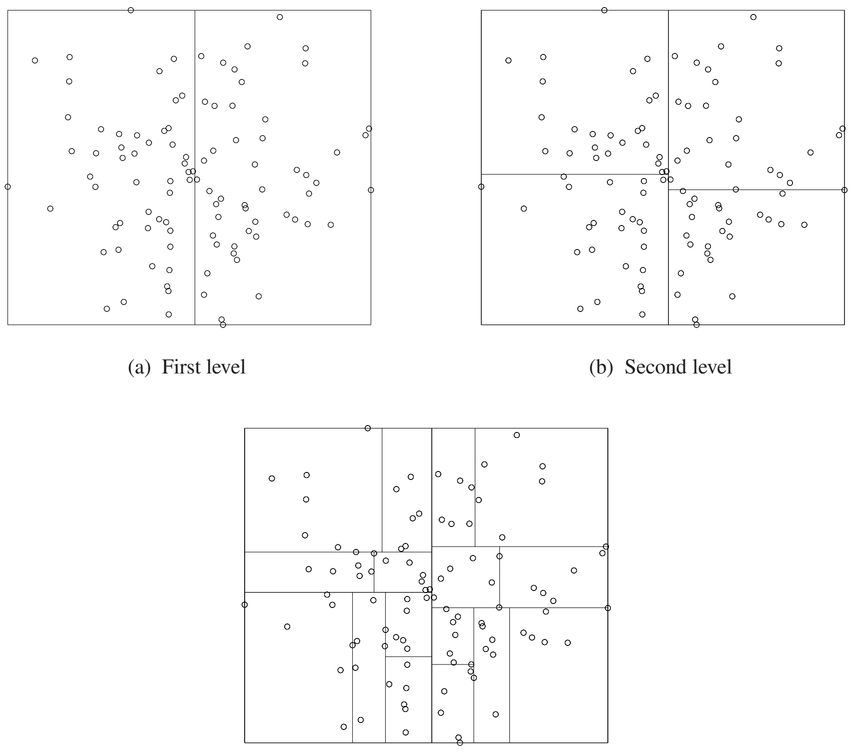
In the term *k-d tree*, the letter k denotes the dimensionality of the space being represented. In principle, it is a generalization of a binary search tree (Aho *et al.*, 1983), with the distinction that at each level in the tree, different regressor components are compared when determining in which direction a cut is to be made.

In the original *k-d* tree formulation (Bentley, 1975), each data point is represented as a node in the tree, and the discriminator, i.e., the regressor component to compare at a certain level in the tree, is chosen cyclically from the regression vector. For instance, in two dimensions (i.e., a *2-d* tree), the x_1 coordinates are compared at the even levels in the tree, and the x_2 coordinates at the odd levels.

An extension of the above idea is the *adaptive k-d tree* (Friedman *et al.*, 1977). Here the data points are stored only at the leaf nodes. Each interior node in the tree contains the median of the regressor component currently used as the discriminator. The discriminator is normally chosen to be the regressor component which currently has the maximum spread. All data points with values less than or equal to the discriminator are added to the left subtree, and all data point with values greater than the discriminator are added to the right subtree. This procedure is repeated recursively until only a few data points remain in each cell, at which point they are stored as linked lists.

Example 4.4 The construction of an adaptive 2-d tree

Figure 4.15 (a)–(c) shows the construction of an adaptive *2-d* tree for a two-dimensional dataset of 100 samples distributed according to the circles. Each dividing line is the median for the discriminator with the largest spread, and is represented as a node in the tree. The


(a) First level (b) Second level

(c) Fourth level

Figure 4.15 *The construction of a 2-d tree. The data are bounded recursively until a small number of data points remain in each box.*

rectangular boxes obtained after a split represent the contents of the corresponding left and right subtrees.

When a query for a neighborhood of a point x occurs, only a subset of the nodes in the tree need to be inspected, see Figure 4.16 which illustrates neighborhood search in a tree of depth four. This greatly reduces the look-up time compared to linear search. ❏

Wide-spread software packages for nonparametric regression, such as as LOESS and LOCFIT (Cleveland and Grosse, 1991; Loader, 1997), utilize *k-d* trees or similar tree structures for speeding up the estimation procedure. They essentially compute fits at the vertices (i.e., corners) of the bounding boxes and interpolate the result at intermediate locations. However, this approach assumes that the entire dataset is available and known *a priori* before estimation, and is therefore in a sense a global method.

Moore *et al.* (1997) have adopted a similar idea for computing predictions more efficiently. At each node in the *k-d* tree they also store unweighted versions of the matrix quantities needed for computing a fit (i.e., $\mathbf{X}_k^T\mathbf{X}_k$ and $\mathbf{X}_k^T\mathbf{y}_k$) using all the data below the node. By assuming that the weighting is approximately equal in some cells, they can then utilize these pre-computed quantities to approximate the general formula (4.16). As for the LOESS and LOCFIT implementations discussed above, this provides an efficient way of computing fits, but requires that the entire dataset is processed in advance of the queries. This requirement follows from the fact that the tree has to be re-balanced if new measurements are inserted.

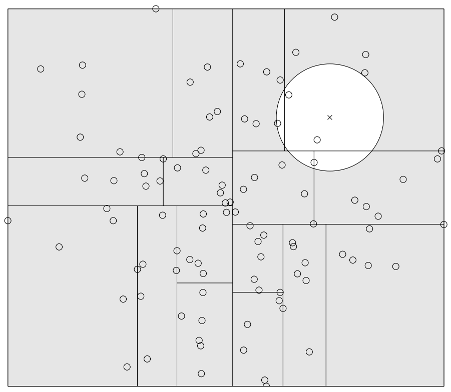


Figure 4.16 *Only a subset of the leaf nodes need to be inspected in order to find the nearest neighbors to the query point* x.

4.8.2 Database Systems and Query Languages

When working with huge data volumes, it would be more than desirable to make use of a *database management system*, DBMS. In recent years, research on database systems in the computer science area has resulted in powerful main memory database systems, which enable very large volumes of data to be stored in the main memory of the computer (Garcia-Molina and Salem, 1992). Provided underlying efficient data structures are available, parts of the data can then be very easily retrieved by queries to the database system, formulated in high level so-called *structured query languages*, SQL (Elmasri and Navathe, 1994).

We close this discussion by mentioning that recent versions of MATLAB provide interfaces to several commercial database managers in its so-called DATABASE TOOLBOX. This could be an interesting path to explore for future implementations of MOD-like approaches.

4.9 Conclusions

This chapter has presented the concept of *model-on-demand*. It is a "data mining" approach that takes advantage of the ability of modern computers to collect and manipulate large datasets. All observations are stored in a database and models are built "on demand" when needed. The bias/variance trade-off inherent to all modeling is optimized locally by adapting the number of data that contribute to the model and their relative weighting.

The model-on-demand idea provides a very flexible way of computing estimates, that to some extent reduces the bias problem associated with global parametric fitting. In addition, it provides the potential for performance rivaling that of global methods, while involving less a priori knowledge from the user. This will be demonstrated in the following chapters, where the algorithms are applied to several system identification problems. It is neither no problems with adding more data: When new observations arrive, just insert them into the database.

The price that has to be paid for the flexibility, however, is a more expensive lookup and estimation procedure. For each demanded prediction, relevant data must be retrieved from the database and a new local model must be formed. This of course limits the rate at which predictions can be computed. An additional drawback is that desirable properties like smoothness and statistical analyzability provided by parametric modeling are retained.

The local modeling associated with model-on-demand can be performed quite arbitrarily. However, in order to keep things simple we have here adopted a weighted regression approach similar to the methods described in Chapter 3. Two conceptually different approaches for weight selection have been discussed, where the first is based on traditional kernel functions and the other relies on an explicit optimization stage. Furthermore, two algorithms corresponding to these approaches have been presented and the asymptotic properties for the optimization approach have been analyzed. It has been demonstrated in simulations that the optimization approach might produce more accurate predictions. However, it is at the same time more demanding in terms of computational efforts. For applications that require fast predictions, the approach based on ordinary weighted regression might hence be to prefer.

It is clear the linear smoothing philosophy that the MOD approaches represent is related to the concept of *radial basis networks* described in Section 2.2.2. This is for instance easily seen by comparing the roles of $\kappa(\cdot)$ and w_i in equations (2.15), (4.17) and (4.47). The main conceptual difference however, is that the $\kappa(\cdot)$ function in the radial basis approach represents a true basis function that is used to build up the functional approximation. In the nonparametric regression case, the weights are instead used to localize the function approximation around the estimation point.

It may be discussed if the approach taken here to keep the data in quite "raw" form is the best solution. Fast methods like LOESS and the approach of Moore *et al.* (1997), that preprocess the dataset, might be to prefer in order provide faster evaluation of each prediction. However, this drastically reduces the flexibility since it is harder to include new measurements. There is no good answer to this question and we leave it open for future research.

4.A Some Power Series Formulas

The expressions for the sums

$$S_k = \sum_{i=1}^{n} i^k, \qquad k = 1,\ 2,\ 3,\ 4,$$

can be summarized as follows:

$$S_1 = \sum_{i=1}^{n} i \ = \frac{n^2}{2} + \frac{n}{2} \tag{4A.1}$$

$$S_2 = \sum_{i=1}^{n} i^2 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} \tag{4A.2}$$

$$S_3 = \sum_{i=1}^{n} i^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4} \tag{4A.3}$$

$$S_4 = \sum_{i=1}^{n} i^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30} \tag{4A.4}$$

The sums have been computed with the assistance of MAPLE (Char *et al.*, 1991).

4.B Moments of a Uniformly Distributed Random Variable

Suppose that the random variable X is uniformly distributed on the interval $[a,\ b]$. Then the moments are given by the following formula

$$EX^k = \frac{1}{k+1}(b^k + b^{k-1}a + \ldots + ba^{k-1} + a^k) \tag{4B.1}$$

See, e.g., Davenport (1975).

5

Applications to Time-domain Identification

In Chapters 3 and 4 we studied several nonparametric estimation methods from a quite general function approximation viewpoint. In this chapter we shall apply and specialize these concepts to the field of *time-domain system identification*, which can be seen as a special application of the general model (1.4), where the measured quantities relate to inputs and outputs of nonlinear dynamical systems. Identification can of course also be performed in the frequency domain. Nonparametric methods for such applications will be treated later in Chapter 6.

The organization of the chapter is as follows. We start in Section 5.1 by giving a brief review of the nonlinear system identification problem. Sections 5.2 and 5.3 discuss different model structures and how they can be used in the nonparametric setting. Some additional relevance aspects in the local framework, like coping with time variability and outliers, are discussed in Section 5.4. Some user guidelines like the determination of model orders and choice of input signals are provided in Section 5.5. Section 5.6 compares the local modeling with the global methods described in Chapter 2. The following two parts, Sections 5.7 and 5.8, illustrate the ideas in numerical examples, both on simulated and real data applications. Section 5.9, finally, provides some conclusions.

5.1 Introduction

As stressed several times earlier in the thesis, system identification is a special case of the general regression relationship (1.4), where $Y_t = y(t)$ represents the output from a dynamical system at time t, and $X_t = \varphi(t)$ relates to lagged input and output measurements of the system. We will now try to apply the model-on-demand ideas to this specialized estimation problem. From a strict statistical point of view, this might introduce a complication since

Y_t and X_t contrary to our earlier assumptions now become *dependent*. However, we shall initially adopt a more pragmatic standpoint and neglect this fact.

We will in this chapter consider nonlinear, multi-input, single-output (MISO) systems in discrete time, governed in input-output form by the dynamic relationship

$$y(t) = m(\varphi(t)) + v(t), \tag{5.1}$$

where, as before,

$$\varphi(t) = \varphi(Z^{t-1}) \tag{5.2}$$

is a regression vector constructed from past measurements;

$$Z^N = \{(u(1), y(1)), \dots, (u(N), y(N))\},$$

and $v(t)$ is a noise term (that does not necessarily need to be white). The goal of time-domain identification is to produce a good *predictor*, i.e., one that estimates

$$\mathrm{E}(y(t)|\varphi(t)) \tag{5.3}$$

as well as possible, utilizing all the available information contained in $\varphi(t)$. As discussed in Chapter 2, this problem has traditionally been solved by global, parametric black-box methods such as neural networks. In this chapter we shall instead study how the model-on-demand approach developed in Chapter 4 can be utilized for *prediction* and *simulation* purposes.

5.1.1 Prediction

The main motivation for our modeling aims in the time-domain is of course the *prediction* problem. That is, given past measurements of the system in question we will be able to predict or *forecast* the short-term behavior of it. This can conceptually be illustrated as in Figure 5.1. The predictor can be viewed as a device that takes measured inputs u and

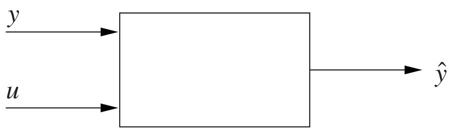


Figure 5.1 *Prediction: Measured inputs u and outputs y are used to predict future outputs $\hat{y}$.*

outputs y as input, and produces predictions $\hat{y}$ of future outputs.

In terms of the model-on-demand concept introduced in Chapter 4, the assumed scenario is that a large amount of measurements from the process has been recorded and stored in a database along with certain quality measures such as on-line estimates of the noise variance. The best and most flexible way of storing such input-output data is, as discussed earlier, an open question and is outside the scope of this thesis. However, in order to make the problem feasible and enable use of the methods derived in Chapter 4, we assume that the stored observations Z^N are searchable and retrievable from the database as output-regressor pairs,

$$\big(y(k),\ \varphi(k)\big), \qquad k = 1, \dots, N. \tag{5.4}$$

Now, if a prediction of the output is needed at time instant $t > N$, a subset of relevant pairs are retrieved from the database, and an estimate is computed based on their values. By reestablishing the nonparametric approach we have adopted earlier in this thesis, this effectively means that the prediction is formed as a weighted average of the outputs $y(k)$ belonging to the subset, i.e.,

$$\hat{y}(t) = \hat{y}(t|t-1) = \sum_{k=1}^{N} W_k(\varphi(t))y(k) \tag{5.5}$$

where the weights – in one form or another – are tuned with the aim at minimizing the mean square prediction error (MSE)

$$\mathrm{E}\big(y(t) - \hat{y}(t)\big)^2$$

in a pointwise sense. This typically results in that the measurements located close to the current regression vector (or *operating point*) $\varphi(t)$ are getting a larger influence on the prediction than those located far away from it. Furthermore, the size of the neighborhood around $\varphi(t)$ is determined from a bias/variance trade-off.

Again it is worth to emphasize that the optimization associated with the predictor (5.5) produces a single local estimate corresponding to the current regression vector. To obtain predictions at other locations in the regressor space, the weights change and new optimization problems have to be solved. That is, the MOD predictor is an *on-line* approach. However, since the estimator essentially is implemented as a local parametric fitting procedure, it is clear that it also provides a local model that describes the system dynamics in a small neighborhood around $\varphi(t)$. This feature will be utilized for the control applications that follow in Chapter 7, but is normally ignored in the prediction context.

A conceptually different but at the same time related *off-line* approach called "operating regime based identification" has been proposed by Johansen and Foss (1995). Their idea is to divide the regressor space into a number of smaller boxes or "regimes", and use simple local models to describe the system dynamics within each regime. A global description of the system dynamics is then obtained by interpolating the local models. A similar approach was also discussed in Skeppstedt *et al.* (1992).

The predictor (5.5) will here be referred to as a *one-step-ahead* predictor since it predicts the output one step forward in time. It is rather straightforward to generalize this concept. By shifting the components of the regression vector, adding the just predicted output and corresponding input signal value to it and predicting an additional time, a *two-step-ahead* predictor is obtained. Repeating this an arbitrary number of times results in a general *k-step-ahead* predictor. Here the integer k is usually referred to as the *prediction horizon*. The k-step-ahead predictor will be extensively used in the control applications treated in Chapter 7.

5.1.2 Simulation

Sometimes it may also be of interest to investigate the long-term behavior of the system under consideration. The tool for such investigations is *simulation*. Simulation can be performed by feeding an arbritrary input sequence

$$u^*(t), \qquad t = 1, \ldots, M$$

into the predictor device in Figure 5.1 and replacing the $y(t)$-measurements with past simulated outputs $y^*(t)$, see Figure 5.2. In the light of the just introduced concept of

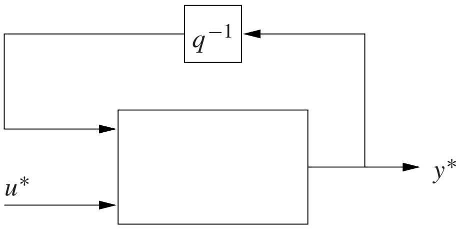


Figure 5.2 *Simulation: An arbritrary signal u^* is used as input and past predicted outputs are fed back to the prediction device.*

prediction, this can be interpreted as the limiting case of k-step-ahead prediction as k tends to infinity. A potential complication with simulation in the local modeling context, however, is that it is much harder to guarantee boundedness and stability properties of the simulated output, since the underlying one-step-ahead predictor will be re-estimated at each time step. This is in contrast to global models (especially linear ones), where stability rather easily can be checked.

It might perhaps sound strange that the simulated output can run unbounded since the associated predictor (as a consequence of (4.17) and (4.47)) is formed as an affine combination of lagged (and stored) outputs which normally should be bounded. However, a typical such situation occurs when the underlying system dynamics is unstable, which will cause the simulated output to grow without limits. Another critical situation, which is quite typical for the local modeling, occurs when the predictor at one time instant returns a "bad" estimate, which, when it is included at the following time instant, causes the regression vector to be located outside the support of the stored data. The estimation process has to perform *extrapolation* rather than interpolation, which is known to produce bad predictions.

5.2 Model Design Considerations

In order to obtain a local model like (5.5) that is suitable for prediction and/or simulation, there is a number of design questions which have to be taken into consideration:

1. Which quantities, constructed from past measurements Z^N, should be included in the regression vector $\varphi(t)$?

2. Which modeling approach should be used?

3. Which distance function (and other relevance criteria) should be used when determining relevant data?

The first question is the same for all modeling approaches – linear as well as nonlinear ones – and, as we soon shall see in Section 5.3, in most situations it turns out that the most useful choices of regression vectors are to let them be built up by lagged inputs and outputs.

The second question arises in nonlinear black-box modeling too, but then normally includes a wide range of possible parameterizations (different basis functions, different number of hidden layers etc). In a local modeling framework, we essentially just have to decide upon the polynomial degree and the norm for the fit.

The third question is rather unique for the nonparametric problem, because of its local properties. Nevertheless, it is an important one, since we can emphasize different features in data with different relevance measures and distance functions (recall the discussion in Section 4.4.1). We shall consider further aspects of this in Sections 5.4 and 5.5.

5.3 Model Structures for Nonparametric Methods

For parametric black-box models, it is well known that an extra regressor component enables an increased flexibility (since the degrees-of-freedom are increased), but that this comes to the price of an increased variance error of the estimate. It is thus often desirable to keep the dimension of the regression vector as small as possible.

This conclusion still holds and is perhaps even more important in the nonparametric case. We saw in Chapter 3 that the rate of convergence typically gets slower in higher regressor dimensions. This was primarily due to the so-called *curse-of-dimensionality* problem, which followed from the sparseness of data in the multidimensional regressor space, see Section 3.8. Another, but equally important reason is the dataset searching problem discussed in the end of Chapter 4. It was concluded that tree structures, like *k-d trees*, are most efficient for a moderate number of dimensions.

Now, keeping the number of regressors limited is normally a matter of choosing the most appropriate model structure for the actual application. However, the special properties (locality etc) associated with local modeling, here also impose additional restrictions that have to be taken into consideration. It turns out that both the estimation and associated dataset searching problems become *significantly easier* if the components of $\varphi(t)$ only relate to measured data.

As we shall see later, a possible alternative to restrict the size of the regression vector, is to impose restrictions on the mapping $m(\cdot)$. One such possibility is to assume that the regression function is global in certain regressor dimensions. This was referred to as *conditionally parametric* models in Chapter 4, and can be a useful approach to fight the curse-of-dimensionality. The local modeling approach also provides the flexibility of using different sets of regressors (subsets of the elements of $\varphi(t)$) for modeling at different parts of the regressor space. In the following though, we will assume that all available regressors are used and that the actual configuration of $\varphi(t)$ is fixed in advance.

Let us for the sake of completeness here again review some of the different choices on model structures that were introduced in Chapter 2, and discuss their ability to be adapted to the local polynomial framework. Although maybe a bit off-topic, we will start with the related time-series prediction problem in Subsection 5.3.1. However, the subject for our main interest, *input-output models*, will be studied more thoroughly in Subsection 5.3.2.

5.3.1 Time Series

Local polynomial models have been proven quite successful for time series modeling in previous applications (Weigend, 1996). Among the first to adopt local modeling ideas to time series prediction were Farmer and Sidorowich (1987) and Farmer and Sidorowich (1988), who applied local linear models to chaotic data.

NAR Structures

The nonlinear auto regressive, NAR, model for time series modeling is formed according to

$$\begin{aligned} y(t) &= m(y(t-1), \dots, y(t-n_a)) + \epsilon(t) \\ &= m(\varphi_y(t)) + \epsilon(t). \end{aligned} \tag{5.6}$$

It can be illustrated as in Figure 5.3 below. Its predictor $\hat{y}(t)$ is obtained by simply deleting

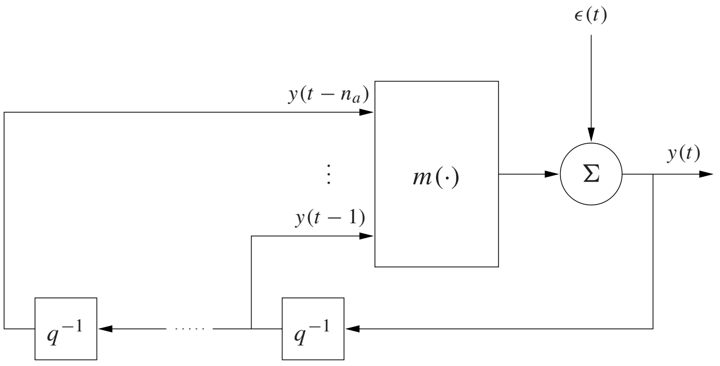


Figure 5.3 *The NAR model structure.*

the noise term $\epsilon(t)$ in (5.6). Since it entirely depends on measured data, it is straightforward to apply the local methods in Chapter 4 to this modeling problem. We will not pursue this subject any further here, though. We instead close this subsection by mentioning that the performance of the NAR model as a function of the two most crucial parameters – the number of neighbors in the local approximation and the model order n_a – has been explored by Casdagli (1991) and Casdagli and Weigend (1994).

5.3.2 Input-Output Models

Although time series modeling is an important and interesting application, our main concern in this thesis is modeling of dynamical systems. We shall therefore here again review the nonlinear input-output structures introduced in Chapter 2 and study how they can be utilized in the model-on-demand context. The overview will be made assuming single-input, single-output (SISO) systems. Generalization to the multiple-input, single-output (MISO) case is straightforward though; think of $u(t)$ as a vector of inputs.

NFIR Structures

In analogy with the linear case, the simplest nonlinear dynamical model is the nonlinear finite impulse response model, NFIR, which only uses past inputs as regressors,

$$\begin{aligned} y(t) &= m(u(t-n_k), \dots, u(t-n_b-n_k+1)) + \epsilon(t) \\ &= m(\varphi_u(t)) + \epsilon(t). \end{aligned} \tag{5.7}$$

Here n_b and n_k denote the number of past inputs and the time delay, respectively. It can be illustrated as in Figure 5.4. As for the NAR case, the corresponding predictor is obtained

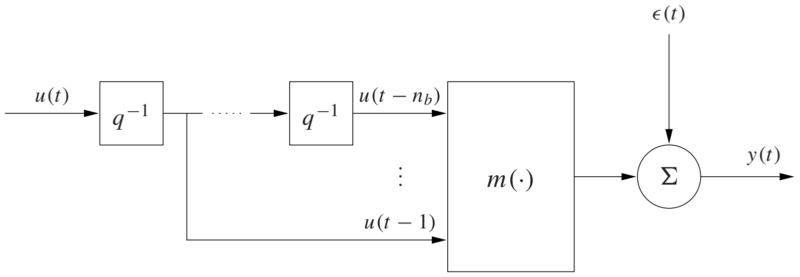


Figure 5.4 *The NFIR model structure with $n_k = 1$.*

by deleting the noise term. That is,

$$\hat{y}(t) = m(\varphi_u(t)). \tag{5.8}$$

A major advantage with this structure is that it always will generate bounded predictions (provided, of course, that $m(\cdot)$ does not have any singularities), since it only depends on past inputs which are assumed to be bounded. However, in order to achieve accurate predictions, the NFIR structure normally requires quite large model orders, n_b, which, as discussed earlier in the chapter, is unsuitable for the local modeling approach, both with respect to the curse-of-dimensionality problem and the dataset searching problem.

NARX Structures

Combining the NAR and the NFIR model structures results in the nonlinear, autoregressive with exogenous input, NARX, model. As stressed in Chapter 2, the linear FIR and ARX models are capable of approximating any linear system provided arbitrary high model orders are allowed. It is easy to believe that the NARX model will play a similar role for the nonlinear case. Since (in theory) all the measured data can be made available as inputs to the NARX model (i.e., an arbitrary sized regression vector), and this information can be used in the local modeling, it follows that this model structure can approximate any, sufficiently smooth, nonlinear system well. The most general NARX structure is

$$\begin{aligned} y(t) &= m(y(t-1),\ldots,y(t-n_a),u(t-n_k),\ldots,u(t-n_b-n_k+1))+\epsilon(t) \\ &= m(\varphi(t))+\epsilon(t), \end{aligned} \tag{5.9}$$

which was referred to as the NARX$_1$ structure by Sjöberg (1995). Here as before, n_a denotes the number of past outputs, n_b the number of past inputs, and n_k is the time delay between the input and output signal. In fact it is possible to have an arbitrary delay n_k^y for lagged outputs as well, but for simplicity we have here decided to let it be equal to one.

The structure (5.9) can be depicted schematically as in Figure 5.5. The associated

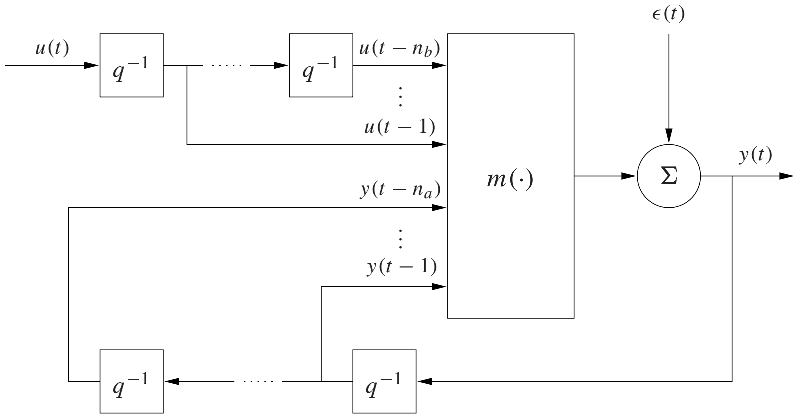


Figure 5.5 *The NARX model structure with* $n_k = 1$.

one-step-ahead predictor is obtained by deleting the noise term. That is,

$$\hat{y}(t) = m(\varphi(t)). \tag{5.10}$$

Similar to its linear counterpart, the NARX model has several nice properties which make it suitable to work with and hence a default choice for the applications that follow:

- It is general and can describe any nonlinear system well.
- It is not *recurrent*. That is, the regressors are independent of previous model-related quantities.

However, its shortcomings are also the same as for the linear ARX model:

- There is no specific noise model. Instead the noise dynamics are modeled together with the plant dynamics. This is easily seen from Figure 5.5.

- As a consequence of this, a good approximation of *both* the system and the noise dynamics may require a high dimension of the regression vector $\varphi(t)$.

A solution to these problems is to include separate noise dynamics in the model as will be discussed next. An equally good fit can then probably be achieved using fewer number of regressors.

Structures with Separate Noise Models: NARMAX and NBJ

We saw in Chapter 2 that structures with separate noise models can be constructed by incorporating past prediction errors $\varepsilon(t-k)$ into the regression vector. However, since it is quite difficult to model the noise in a general nonlinear way, it is natural to impose the restriction that the model is linear in the prediction errors, $\varepsilon(t-k) = y(t-k) - \hat{y}(t-k)$. This corresponds to the NARMAX-type structure

$$y(t) = m(\varphi(t)) + C(q^{-1})\epsilon(t), \tag{5.11}$$

where $\varphi(t)$ is constructed as in the NARX case and $C(q^{-1})$ is assumed to be monic. The corresponding predictor is

$$\hat{y}(t) = m(\varphi(t)) + (C(q^{-1}) - 1)\varepsilon(t), \tag{5.12}$$

which can be interpreted as a conditionally parametric model that is global in the $\varepsilon(t-k)$-directions of the regressor space.

A major problem with this structure in the model-on-demand context, is the recurrency, i.e., that parts of the extended regression vector now depend on past model outputs. Since a major component of our estimation procedure is dataset searching in the space spanned by the regressors, this implies that the estimation database has to be built *on-line* during estimation in order to record old prediction errors. Thus the NARMAX structure is not that useful in the nonparametric setting.

The same conclusion also holds for the nonlinear Box-Jenkins (NBJ) and output-error (NOE) structures, since some components of the regression vector in both cases then will rely on model dependent quantities.

NFIR Systems with Colored Noise

So far we have assumed that the noise term $\epsilon(t)$ is white. Let us see what happens if we relax this assumption. Consider an NFIR model with an additive but not necessarily white noise term $v(t)$,

$$y(t) = m(\varphi_u(t)) + v(t). \tag{5.13}$$

For the noise $v(t)$ we will in general only need a spectral description similar to the linear case in Section 2.2.1. It can thus be described by

$$v(t) = H(q)\,\epsilon(t) \tag{5.14}$$

where $\{\epsilon(t)\}$ is white noise. A predictor for (5.13) is

$$\begin{aligned}\hat{y}(t) &= m(\varphi_u(t)) + (H(q) - 1)\,\epsilon(t) \\ &= H^{-1}(q)\,m(\varphi_u(t)) + \Big(1 - H^{-1}(q)\Big)\,y(t). \qquad (5.15)\end{aligned}$$

In the first term, the linear filter $H^{-1}(q)$ can equally well be absorbed into the nonlinear mapping $m(\cdot)$. The structure (5.13) can thus be viewed as a NFIR structure, accomplished by a linear term consisting of past $y(t)$'s. That is,

$$\hat{y}(t) = -a_1 y(t-1) - \ldots - a_{n_a} y(t - n_a) + \tilde{m}(\varphi_u(t)). \qquad (5.16)$$

See also Figure 5.6. Since this structure has the same regressor configuration as the general

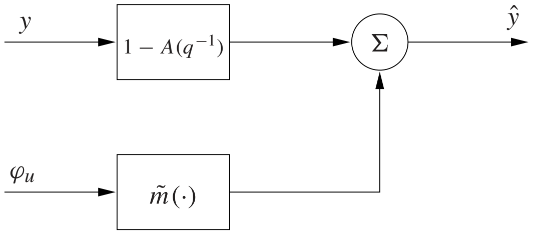


Figure 5.6 *A NARX model consisting of a linear part for the past outputs and a nonlinear part for the past inputs.*

NARX$_1$ model (5.9), it was referred to as a NARX$_2$ model by Sjöberg (1995). The fact that it is not recurrent makes it a very interesting and useful alternative to work with.

In the model-on-demand context, the NARX$_2$ model can similarly to the NARMAX structure above be interpreted as a conditionally parametric local linear model which is global in the $y(t-k)$-directions of the regressor space. As outlined in Section 4.4.1, estimation of such models can be performed by zeroing out the corresponding elements of the scaling matrix which controls the distance function. Another alternative is to start by fitting a global linear model to the $y(t-k)$-regressors, and use the residuals for nonparametric modeling in the $u(t-k)$-directions of the regressor space. This is usually referred to as *semi-parametric modeling*.

The NARX$_2$ concept can be generalized to the case where the predictor is a sum of two nonlinear terms. That is,

$$\hat{y}(t) = m_1(\varphi_y(t)) + m_2(\varphi_u(t)), \qquad (5.17)$$

where $\varphi_y(t)$ and $\varphi_u(t)$ are defined as in the NAR and NFIR cases. A model of this type was referred to as an additive model in Chapter 3, and can (although even more computational demanding) be estimated using the backfitting algorithm as discussed in Section 3.8.2. In the work by Sjöberg (1995), a model of the form (5.17) is called a NARX$_3$ model, and can be interpreted as a FIR model of the type (5.13) where the noise dynamics is state dependent, i.e.,

$$v(t) = H(q, \varphi_u(t))\, \epsilon(t). \tag{5.18}$$

If the mixing properties between past y's and u's are minor, the NARX$_3$ model could be a useful alternative to deal with the curse-of-dimensionality.

5.4 Other Relevance Aspects

Relaxing the assumptions (time-invariance, informative data etc) that were made in the beginning of the chapter may give rise to the need of other relevance measures, which will be treated and discussed in the following two subsections.

5.4.1 Coping with Time-varying Systems

We have hitherto in this chapter only considered *time-invariant* systems, i.e., systems whose properties are not changing with time. There might be situations, though, where this assumption does not hold. However, provided that the database is updated continously during operation, it turns out that coping with (slowly) time-varying systems is rather straightforward to do by just keeping track of the time indices.

The simplest solution is perhaps to include the sample time t as an extra regressor component in $\varphi(t)$. That is,

$$\varphi(t) = (y(t-1), y(t-2), \ldots, u(t-n_k), u(t-n_k-1), \ldots, t)^T . \tag{5.19}$$

Thus, when a prediction is required, the samples located closest in *both* space and time are retrieved, leading to that the most recent measurements are getting the largest influence on the estimate. Here the relative importance between spatial and time locality can be traded off by a proper choice of distance function.

An obvious question is of course whether or not the time component of the extended regression vector (5.19) should be included in the local modeling or not. If it is included, we will always perform estimation at the boundary of the t-part of the regressor space, which might cause problems with bias. A pragmatic approach is therefore to only utilize the extra time information in $\varphi(t)$ while searching for relevant data, and omit it during the estimation phase.

The above way of coping with time-varying systems can be compared to the traditional, recursive estimation approach (see, e.g., Ljung (1999)), where the time-variability is dealt with using an exponential forgetting similar to the RLS algorithm in equation (4.39) but with $k = t$. In that case, however, the forgetting factor λ only controls the locality in time, and spatial locality is neglected. An extension of this concept that fits in our framework, is to utilize the forgetting mechanism as an extra weighting in a modified least squares criterion similar to (4.22). That is,

$$v_i = \lambda^{t-t_i}, \tag{5.20}$$

where the forgetting factor $0 < \lambda \leq 1$ as in the RLS case has to be tuned by the user in order to trade off noise sensitivity against tracking capabilities.

A third approach is to assume that the dataset search routine provides the ability of retrieving data belonging to certain time ranges. Then, if we for instance know that the system properties changed significantly at time t_0, we can choose to only retrieve data $(y(\tau),\ \varphi(\tau))$ stored at times $\tau > t_0$.

5.4.2 Incorporating Other Quality Measures

Continuing the ideas introduced in the preceeding subsection, it is of course also possible to associate other attributes with the data than just only the time-stamps, and utilize this information in the estimation process. A quite natural extension here, is to mark the measurements with some kind of quality measure during data acquisition. Possible candidates for this will be discussed next.

Noise Variance

The most natural choice of quality measure in the local approach is of course the *noise variance*. For some applications, like *GPS* (global positioning system), it turns out that the variance is known *a priori*, while it in others can be estimated by means of different on-line techniques.

How to utilize this extra information opens up different possibilities. A novel and straightforward solution is to include the variance information in the regression vector, and only search for the data that have as low variance as possible.

Another, perhaps more natural approach (which can be justified in the maximum-likelihood framework) is to incorporate the variance information as an *inverse weighting* in the local fitting criterion. That is, the local estimate is computed in a fashion similar to the modified least squares criterion (4.22).

Both these approaches could be useful to combat the effects of *outliers*, that is, "bad" and irrelevant data that may affect the estimation result in a bad direction.

Excitation Requirements

Another problem may arise if the data acquisition is made during feedback control. Feedback in general introduces dependencies in data, and identifiability can be lost due to poor excitation. According to Ljung (1999) a necessary condition for excitation for linear systems is that the spectrum satisfies

$$\Phi_Z(\omega) > 0. \tag{5.21}$$

It would therefore be desirable to mark the data with an on-line estimate of (5.21). Such estimates can be obtained using FFT:s with forgetting factors or wavelet spectrogram. However, how to generalize this idea to the nonlinear setting is not completely obvious and will require further investigations.

5.5 User Guidelines

We will here present some user guidelines that might be useful when solving prediction/simulation problems using the model-on-demand approach. Some of them have already been emphasized in earlier sections, but it might anyway be useful to make a summary.

5.5.1 Choice of Model Structure

Different model structures and their advantages and disadvantages in the MOD framework were discussed at length in Section 5.3. It was concluded that NFIR and NARX models were the best suited candidates to deal with the properties of this modeling philosophy. Now the choice between these structures is mainly a matter of how many regressors we can afford and *a priori* knowledge of the application. It be guided by the following aspects:

- NFIR models require that the entire dynamics of the system is covered by lagged inputs. That is, if the response time from input to output is τ and the sampling interval is T_s, then the number of regressors should be at least $n_b = \tau / T_s$. Even for quite simple systems, this could be a large number.
- By using NARX models, it is possible to model the same dynamics with significantly fewer regressors. A disadvantage, though, is that past outputs bring in past disturbances into the model which increases the demands on the estimation procedure. The model output may also run unbounded during simulation.
- If it is known or suspected that the dynamics associated with lagged outputs and lagged inputs can be separated, it might be useful to try the NARX$_2$ and NARX$_3$ structures. Otherwise the NARX$_1$ model should be considered to be the default choice for the prediction/simulation problem.

5.5.2 Selecting the Order of the Model

So far we have only been dealing with different structural issues, but we have not addressed the choice of the model order. It is however inherent from the discussion above that this is a closely related problem, since a good choice of model structure saves parameters.

The choice of model order is normally very much depending on the particular application, and it is hard to give general guidelines. A natural methodology, however, is to try to get as far as possible with simple linear models, and if that still gives unsatisfactory results, stick to nonlinear black-box models with the same regressor configurations.

From parametric modeling it is known that a large number of parameters in the model provides an increased flexibility but also results in larger variance errors. On the other hand, a small number of parameters gives large bias errors. The main component of order selection procedure, is thus to handle this trade-off between bias and the variance. Since the local modeling in fact has associated with it a parametric estimation procedure that is repeated at each sampling instant, it is clear that this trade-off is applicable also here.

A very natural and commonly used approach in system identification is *cross-validation*, which is an approach that we in a slightly different form already have been dealing with (recall sections 3.4.1 and 4.4.5). The fundamental idea is the quite natural assumption that the best model is the one that – in some sense – gives the best performance when applied to data that are not used when computing the estimate. In practice this is achieved by splitting the available dataset into two parts, *estimation data*, which is used for computing estimates and *validation data*, for which the model is evaluated and some performance measure is computed. As pointed out by Ljung (1999), cross-validation has an intuitive appeal; we simple check if the current predictor configuration is capable of "reproducing" data it has not yet seen. If that works out well, we have some confidence in the current structure.

5.5.3 Choice of Distance Function

The choice of distance function was thoroughly discussed in the general estimation framework in Chapter 4, but it might be worth to reconsider this question also here. Recall from the definition of distance functions in Chapter 4 that by selecting the scaling matrix **M** properly it is possible to emphasize different features in data. The default choice used in the MOD algorithms was to let it be proportional to the inverse covariance of the regressors. However, there exist situations where other choices might be useful to consider. Imagine for instance a simple tank system like the one introduced in Example 1.1. It is well known that the dynamics of such a system will depend on the water level in the tank. By tuning the elements of the scaling matrix in an intelligent way so that the regressors associated with the level measurements get large influence in generating the distance measures, it is possible to make the locality in water level more important than locality in other regressors.

5.5.4 Choice of Input Signals

The fact that we are dealing with a combination of both nonlinear systems and nonparametric estimators requires some additional attention when designing data acquisition experiments. Pseudo-random binary sequences (PRBS) have been widely used in identification of linear systems and have the advantages of providing simple implementations, reproducibility, and autocorrelation functions similar to white noise.

The PRBS input, however, is not well suited for nonlinear applications. Since the PRBS consists of only two distinct levels, the resulting data obtained from the process may not provide sufficient information to reveal nonlinear features in its behavior. Additionally, the distribution of data on only two lines in the $u(t-k)$-parts of the regressor space might make the underlying least-squares problem ill conditioned. Regularization can help in some situations (recall the discussion in Section 4.4.3) but it is in general more desirable to choose an input signal that is continously changing over the domain of interest such as a Gaussian or a uniformly distributed signal. An alternative is multi-level pseudo-random sequences (m-level PRS) whose usefulness in the model-on-demand context is discussed in Braun *et al.* (1999).

5.6 Comparison with Global Methods

The adoption of local modeling techniques to the system identification field of course also motivates the need for comparisons with other well established approaches such as neural nets and wavelets. Since the methods represent quite different paradigms it is clear that they will have both advantages and disadvantages when evaluated from different viewpoints.

As discussed many times earlier, the greatest advantages with parametric black-box models is that the model is fitted to data once, which provides a very cheap and efficient way of obtaining predictions. Furthermore, global methods can deal with *recurrent* model structures, although the complexity of the estimation procedure then increases significantly (Sjöberg *et al.*, 1995).

On the other hand, local modeling methods such as model-on-demand are easy and straightforward to apply. Since the modeling is based on simple polynomial model assumptions, this approach gives the advantage of providing good fits using significantly simpler models, than the ones required in the global case. A complication, though, is that the local approach requires quite large datasets, relies heavily on data management routines, and that the estimation thus requires larger computational resources due to the underlying searching problem. This can of course be critical for real-time applications with timing constraints.

5.7 Applications Based on Simulated Data

We will now illustrate the MOD paradigm in a number of numerical simulations. We will start this section by considering simulated data. The treatment of applications based on real measured data is postponed to Section 5.8.

5.7.1 A Linear Discrete-Time System

As a start of our investigations, it may be interesting to study whether or not the local modeling approach degrades the performance compared to global methods if the true system is linear. In this section the model-on-demand approach is therefore applied to a simulated linear dynamic system. The system considered is the so-called Åström system, which has earlier been investigated by Ljung (1999). It is a second order system defined according to

$$y(t) - 1.5y(t-1) + 0.7y(t-2) = u(t-1) + 0.5u(t-2) + \epsilon(t), \tag{5.22}$$

that is, an ARX system with $n_a = n_b = 2$ and $n_k = 1$. (To simplify the notation, we will in the following abbreviate this as ARX 221).

In order to generate data, the system was simulated for $t = 1, 2, \ldots, 5000$ with $\{u(t)\}$ and $\{\epsilon(t)\}$ selected as independent and Gaussian distributed random sequences with zero means and unit variances. The data were used to create an estimation database of pairs $(y(t),\ \varphi(t))$ with

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

Using this database, a simulation was made using a third Gaussian random sequence as input. It turned out that MOD-algorithm 4.2 with default values performed best in this case. The resulting output is represented by the solid curve in Figure 5.7 (a). The dashed curve is the true noiseless output obtained from the true system (5.22). For comparison, a

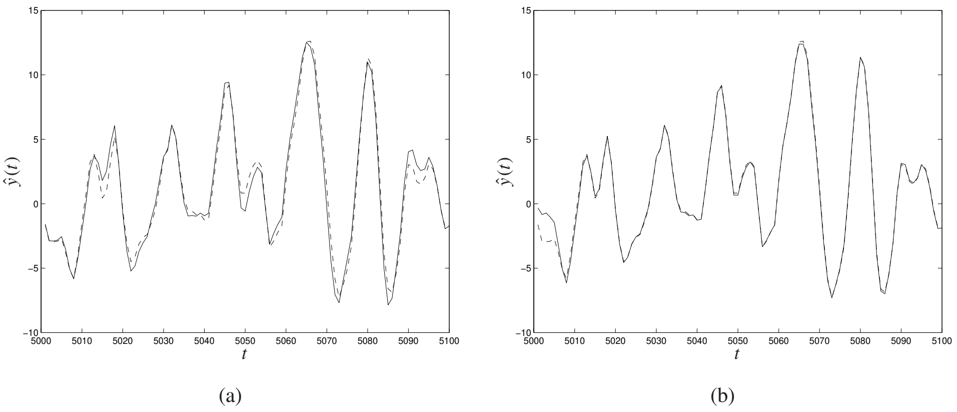


Figure 5.7 *(a) Result of a model-on-demand simulation using an estimation database consisting of 5000 samples generated from the Åström system (5.22). (b) Result of a simulation using a linear ARX model with the same regressors. The solid lines are the simulated outputs, the dashed lines the true output.*

linear ARX model with the same regressor configuration was also tried. The result of this experiment is shown in Figure 5.7 (b). As shown the linear ARX model performs slightly better, since it in contrast to the model-on-demand approach is optimized over the entire regressor space. The discrepancy in the model-on-demand simulation case is due to the fact that the model-on-demand estimator at some $\varphi(t)$-points adapts to the local properties of the noise sequence $\epsilon(t)$, i.e., that the localized goodness-of-fit criterion (FPE in this case) find spurious features in the noise. A possible solution to this problem is to increase the penalty α on the noise term for the FPE as demonstrated in Section 4.4.5. However, the prediction errors are in this case quite small, so the model-on-demand estimator provides an acceptable result.

5.7.2 A Nonlinear Discrete-Time System

In this section the model-on-demand approach has been applied to a nonlinear benchmark system proposed by Narendra and Li (1996). The system is defined in state-space form by

$$x_1(t+1) = \left(\frac{x_1(t)}{1+x_1^2(t)} + 1\right)\sin(x_2(t)) \tag{5.23a}$$

$$x_2(t+1) = x_2(t)\cos(x_2(t)) + x_1(t)\exp\left(-\frac{x_1^2(t)+x_2^2(t)}{8}\right) + \frac{u^3(t)}{1+u^2(t)+0.5\cos(x_1(t)+x_2(t))} \tag{5.23b}$$

$$y(t) = \frac{x_1(t)}{1+0.5\sin(x_2(t))} + \frac{x_2(t)}{1+0.5\sin(x_1(t))} + \varepsilon(t) \tag{5.23c}$$

Although not present in the original treatment, we have here added a measurement error term $\varepsilon(t)$. The plant (5.23) does not correspond to any real system, and is chosen to be sufficiently complex and nonlinear so that conventional linear methods will not give satisfactory performance. Since it has two state variables, $x_1(t)$ and $x_2(t)$, it is of order two. The input signal $u(t)$ affects $x_2(t+1)$ which in turn affects $y(t+1)$. Hence the plant has relative degree (or time delay) one. The objective for our modeling efforts is to model the system output, given measurements of $u(t)$ and $y(t)$.

We assume that the states are not measurable, so we have to rely entirely on an input-output model. Since the order of the system is two, a natural model choice is a NARX 221 structure. However, it turned out that in order to reach an acceptable result in the MOD case, it was required to increase the model order by one. That is, a NARX 331 structure;

$$\hat{y}(t) = m(y(t-1), y(t-2), y(t-3), u(t-1), u(t-2), u(t-3)).$$

A database consisting of $N = 50\,000$ samples was generated using a random input $u(t)$ which was chosen to be uniformly distributed on the interval $[-2.5,\ 2.5]$, and a white noise disturbance $\epsilon(t)$ with variance 0.1. The performance of a model-on-demand simulation was validated using a test input defined as

$$u(t) = \sin(2\pi t/10) + \sin(2\pi t/25), \qquad t = 1, \ldots, 200.$$

Note here that $|u(t)| \leq 2$ which guarantees that the regression vector will be located within the support of the stored data.

For this application, MOD-algorithm 4.1 with a local linear model structure performed best. The result of a simulation is shown in Figure 5.8 (a). The simulated output follows the true noise-free output reasonably well and the quality of the simulation is at least comparable with the result reported by Narendra and Li (1996) after using a neural net and a significantly larger dataset (500 000 samples). For comparison a linear ARX 331 model was also tried, see Figure 5.8 (b). But as expected the system is too nonlinear in this case to be modeled accurately by a simple linear model.

5.8 Applications Based on Real Measured Data

We saw in Section 5.7 that the model-on-demand approach turned out be useful in predicting/simulating the artificially constructed examples. Now we will apply the algorithms on datasets obtained from real physical systems (in fact we have already done so in Example 1.1). They all represent well-known applications that have been thoroughly studied in

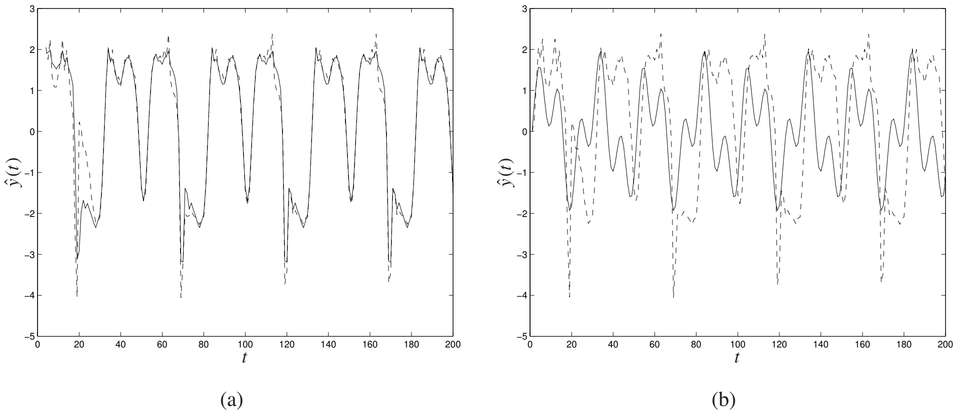


Figure 5.8 *(a) Result of a simulation using a NARX 331 model and the model-on-demand approach. Solid: Simulated output. Dashed: True output. (b) Result of a simulation using a linear ARX 331 model. Solid: Simulated output. Dashed: True output.*

previous contributions. This has the great advantage of enabling direct comparison with known and well established identification techniques.

A potential problem for all of them, which motivates some words of caution in advance, though, is that the sizes of the datasets are rather modest. This might be severe hurdles for success in the model-on-demand case.

5.8.1 Simulation of a Water Heating Process

In this subsection we consider simulation of a water heating process as depicted in Figure 5.9 (a)[1]. It has earlier been investigated by Koivisto (1995), Lindskog (1996) and Stenman (1997). The system setup can be described as follows: Cold water flows into to the heater with a flow rate of $Q_{\text{in}}(t)$, and is heated by a resistor element which is controlled by the voltage $u(t)$. At the outlet, the water temperature $T(t)$ is measured. The modeling problem is to describe the temperature $T(t)$ as a function of the voltage $u(t)$ under the assumption that the inlet flow as well as the inlet water temperature are kept constant. The behavior of the system is nonlinear due to saturation characteristics of the thyristor (Lindskog, 1996).

The dataset consists of 3000 samples, recorded every 3rd second, and originates from a real time identification run (performed by Koivisto (1995)), where the system was driven by an input signal of pseudo-random type. The data was divided into an estimation set of $N = 2000$ samples, see Figure 5.9 (b), and a validation set of 1000 samples. The time delay from input to output is between 12 to 15 seconds, (Koivisto, 1995). This yields that useful regressors stemming from the input are $u(t-4)$, $u(t-5)$ and so on.

[1]The figure has been kindly provided by Peter Lindskog.

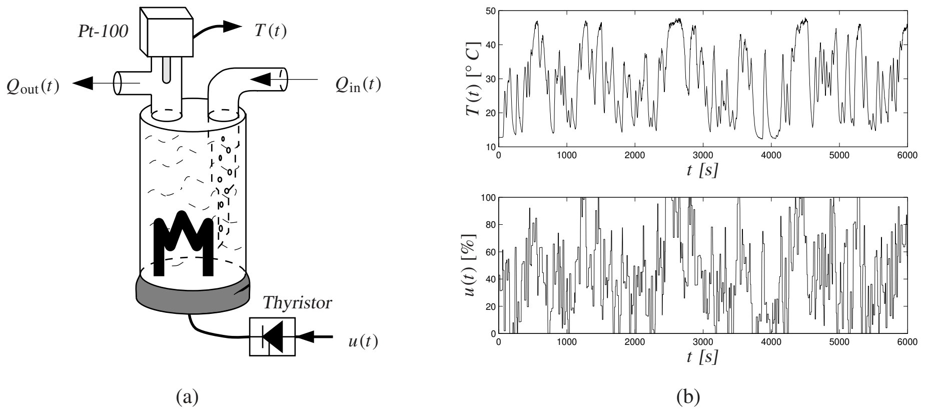


Figure 5.9 *(a) The water heating process. (b) Estimation dataset consisting of 2000 samples.*

We apply Algorithm 4.2 to the heater data using a NARX 224 model, i.e., we form the predictor according to

$$\hat{T}(t) = m(T(t-1),\ T(t-2),\ u(t-4),\ u(t-5))\,. \tag{5.24}$$

We then let the estimation dataset define our observation database, and use the voltage signal $u(t)$ in the validation dataset to simulate the corresponding temperature $T(t)$. The result from this procedure is shown in Figure 5.10 (a).

Method	RMSE
ARX 224	2.082
NARX 224 (MOD)	0.802
NARX 224 (fuzzy)	1.020

Table 5.1 *Root mean square errors for the heater simulations.*

For comparison we also tried a linear ARX model with the same regressor configuration as in (5.24). The resulting simulated output is shown in Figure 5.10 (b) and is clearly worse than the one in Figure 5.10 (a). Root mean square errors (RMSE) for the two experiments are summarized in Table 5.1. Here we have also included the result achieved by Lindskog (1996) while using a *fuzzy modeling* approach. Koivisto (1995) has reported similar results using neural network modeling.

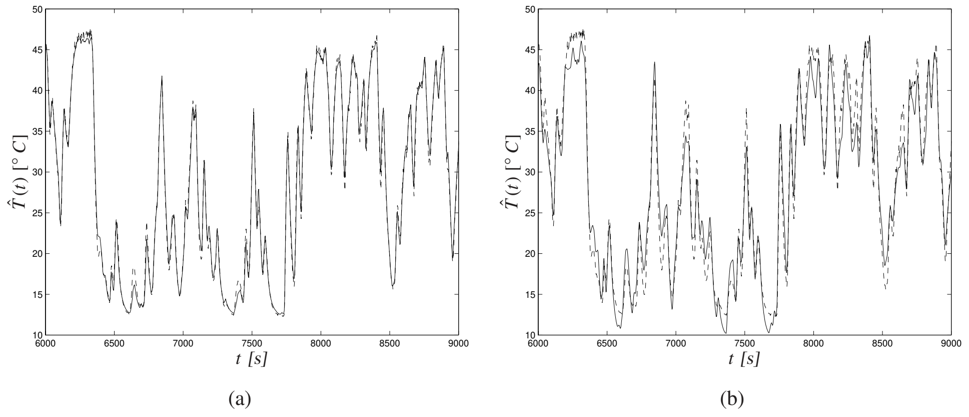


Figure 5.10 *(a) Result of a* MOD *simulation of the water heating system using an estimation database constructed from the dataset in Figure 5.9. Solid: Simulated output. Dashed: True system output. (b) Result of a simulation using a linear ARX model. Solid: Simulated output. Dashed: True system output.*

5.8.2 Simulation of a Hydraulic Robot Actuator

In this example we shall perform simulation of a hydraulic robot actuator. The application has previously been investigated by Sjöberg *et al.* (1995) and Gunnarsson and Krus (1990) and has to some extent become a "benchmark" example for nonlinear black-box modeling.

The system in question is a hydraulic actuator that controls the position of a robot arm. The arm position is directly coupled to the oil pressure in the actuator and is determined by a valve that regulates the oil flow into the actuator. The modeling aim is to model the oil pressure as a function of the valve opening.

The dataset consists of 1024 samples in total and has been split into two equally sized parts – one for estimation and one for validation. Figure 5.11 shows the estimation part. The valve opening is here denoted $u(t)$ and the oil pressure $p(t)$. As observed from the figure, the oil pressure signal has very oscillative settling periods after step changes in the valve size. This is primarily due to mechanical resonances in the robot arm.

It seems reasonable to start by first considering simple linear black-box models. Experiments performed by Sjöberg (1995) show that a model which predicts the pressure $p(t)$ using three old values of the pressure and two old values of the valve size signal (i.e., an ARX 321 model) is appropriate for this particular application. The result of a simulation on validation data using this configuration is shown in Figure 5.12 (a). The achieved root mean squared error, RMSE, for the simulated output is 0.925, but it is quite obvious from visual inspection that it would be possible to enhance this rather poor result by considering more complex nonlinear models.

A severe problem with this application from a local modeling point-of-view is of course

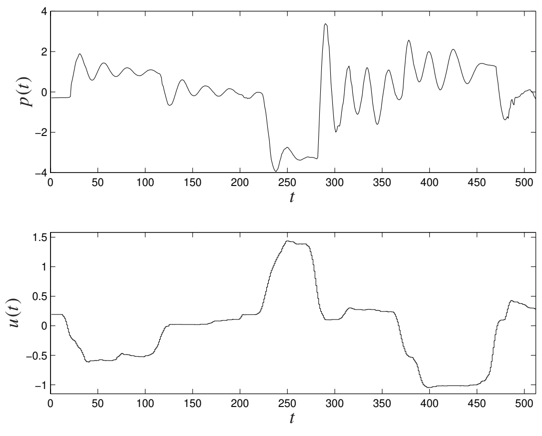


Figure 5.11 *Measured values of oil pressure $p(t)$ and valve position $u(t)$.*

the sparseness of data (512 data points in a five-dimensional regressor space). However, it might anyway be interesting to see how far we can get with the model-on-demand approach. Similar to the simulated nonlinear system in Subsection 5.7.2 we find that in order to obtain reasonable results we have to increase the model order by one. That is, we use a NARX 431 predictor,

$$\hat{p}(t) = m(p(t-1), p(t-2), p(t-3), p(t-4), u(t-1), u(t-2), u(t-3)).$$

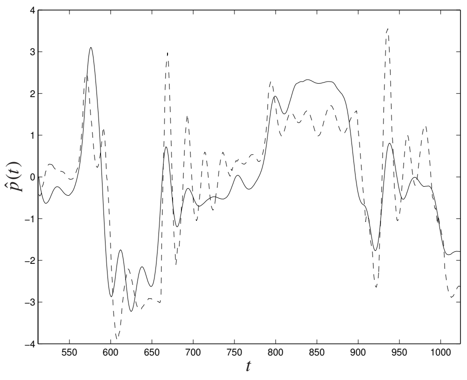
Figure 5.12 (b) shows a simulation on validation data using MOD-algorithm 4.1 and a local linear NARX$_1$ model based on the above regressor configuration. The achieved RMSE is 0.499 and represents a significant enhancement compared to the linear ARX model. It turns out though that the result can be even more improved by considering a conditionally global NARX$_2$ model. This can be achieved by altering the scaling matrix according to

$$\mathbf{M} = \operatorname{diag}(0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1).$$

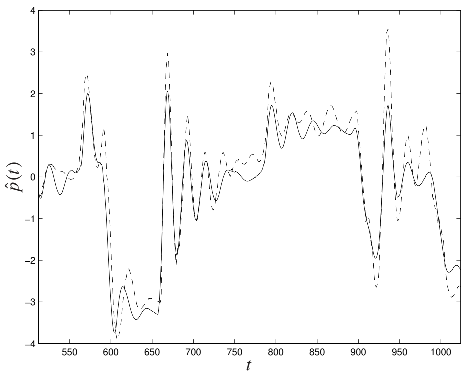
A simulation with this setup in the model-on-demand framework is shown in Figure 5.12 (c). The RMSE error is now 0.471. Trying the NARX$_3$ model and other choices of model parameters did not result in any further improvements. However, Sjöberg (1995) obtained a best value of 0.328 using a complex neural network structure. The conclusion is therefore that this dataset is too small to produce results directly comparable to the parametric case. Nevertheless, we have demonstrated how local modeling techniques even for rather small datasets quite drastically can improve the quality of the simulations compared to traditional linear black-box modeling.

5.8.3 Simulation of Buffer Vessel Dynamics

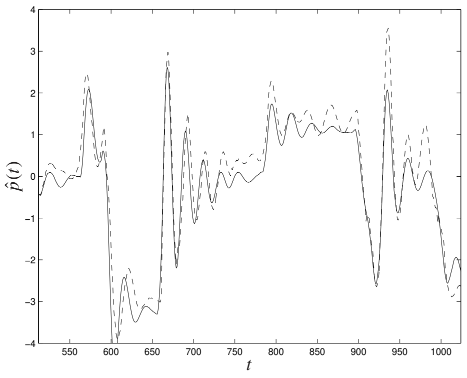
The dataset in this example has its origins in a typical process industry problem. It is taken from STORA's pulp factory in Skutskär, Sweden, and has previously been investigated by



(a)



(b)



(c)

Figure 5.12 *Simulation of the crane behavior using validation data: (a) ARX model. (b) NARX*$_1$ *model. (c) NARX*$_2$ *model. The solid lines are the simulated output and the dashed lines are the true system output.*

Method	RMSE
ARX 321	0.926
NARX 431 (Mod)	0.471
NARX 431 (NN)	0.328

Table 5.2 *The best achieved root mean square errors for the hydraulic robot simulations using different modeling philosophies.*

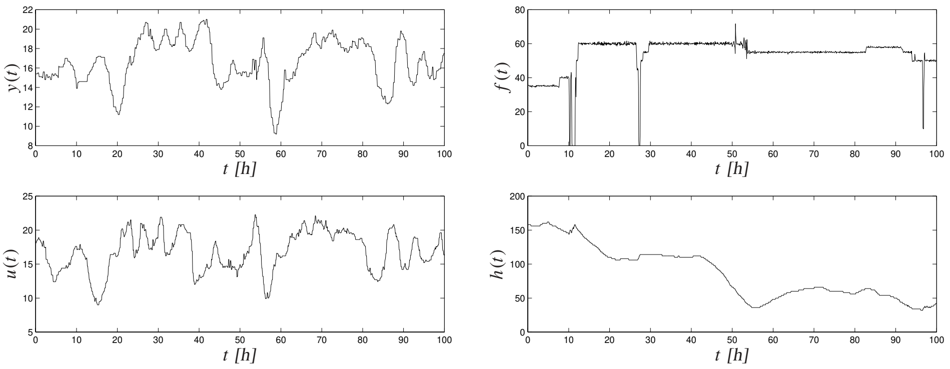


Figure 5.13 *Vessel data from the pulp factory in Skutskär, Sweden. The dataset consists of 1500 samples recorded every 4th minute.*

Andersson and Pucar (1994), and Ljung (1999) (from which this description is taken). The data describe the dynamics of one particular stage in the manufacturing process where the pulp is traveling trough a buffer vessel and is the subject for certain manipulations and operations (such as washing and bleaching).

The process is a multivariable system (MISO) with four channels. Figure 5.13 shows the recorded time history for each of them. The total number of samples is 1500 and the sampling interval T_s is 4 minutes. We denote the measurements as follows:

$$\begin{aligned} u(t) &: \text{The } \kappa\text{-number of the pulp flowing in,} \\ y(t) &: \text{The } \kappa\text{-number of the pulp flowing out,} \\ f(t) &: \text{The output flow,} \\ h(t) &: \text{The level of the vessel.} \end{aligned}$$

Here the κ-number denotes a quality marker which exact meaning is not that important for the modeling problem.

The original modeling problem was to estimate the residence time in the vessel. It turned out that a good way of doing so was to estimate the dynamics from $u(t)$ to $y(t)$, since that would give an indication of how long time it takes for a change in the input to have an effect on the output.

The sampling rate used is too fast, so the data is decimated a factor of 3, giving a sampling interval of 12 minutes. The resulting dataset is divided into an estimation part of $N = 300$ samples and a validation part of 200 samples. Figure 5.14 (a) shows the "best" ARX model according to Ljung (1999), which is formed using four lagged values of $y(t)$ and each of input signals. The time delay from $u(t)$ to $y(t)$ is twelve samples and the time delay from $f(t)$ and $h(t)$ is one sample. This clearly does not look good.

Ljung (1999) resolves the problem with a resampling procedure, by realizing that the approximate time delay in the vessel is proportional to flow divided by level. A simulation based on resampled y's and u's and an ARX 419 model is shown in Figure 5.14 (b). Here, though, we decide to use the original but decimated data for brute-force model-on-demand simulation using Algorithm 4.1 with local linear models.

We found that the important variables to consider in this case is the κ-numbers and the level measurements. The best result was obtained by a NARX model built up by four lagged values of $y(t)$, three lagged values of $u(t)$ and one lagged value of $h(t)$. The time-delay was eight samples from $u(t)$ and one sample from $h(t)$. The result of a simulation is displayed in Figure 5.14 (c). Here the relative importance of the level measurement in

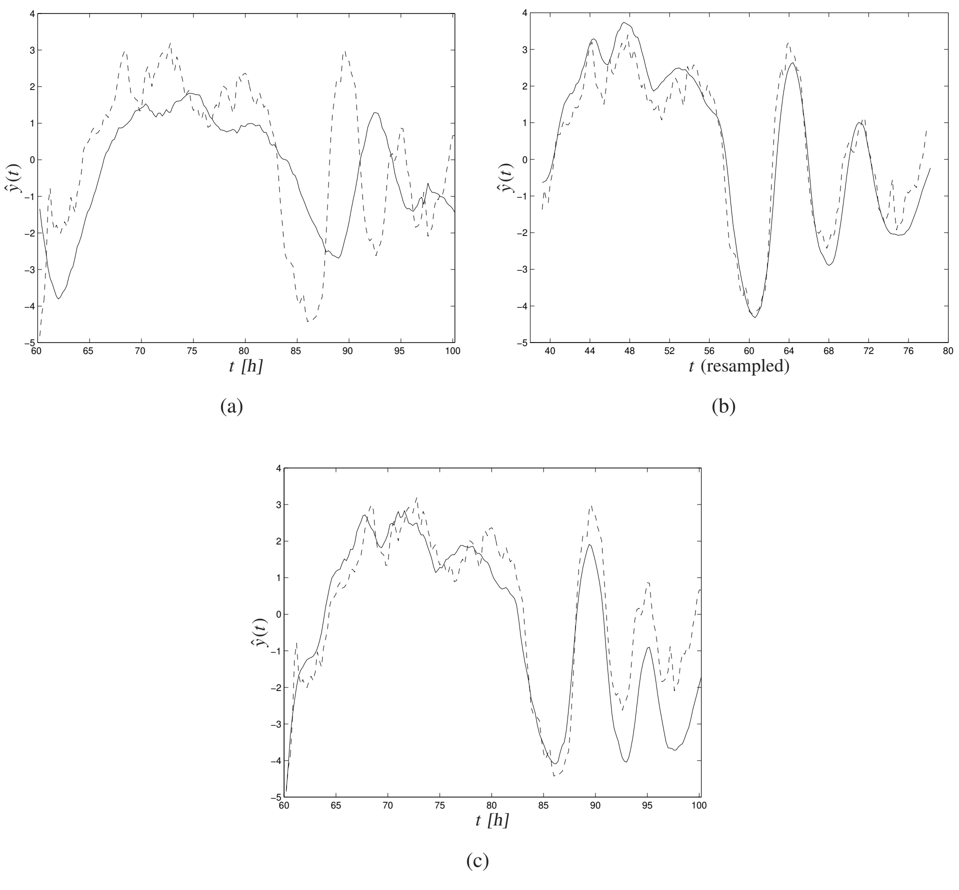


Figure 5.14 *Simulation results for the vessel data: (a) ARX model. (b) Resampled ARX model. (c) NARX model based on original data. The solid lines represent the simulated output, and the dashed lines the true system output.*

generating the distance measure (i.e., the corresponding element of the scaling matrix $\mathbf{M}$) has been increased a factor of two. The simulated output follows the true one quite well, at least up to $t = 90$, where it starts to deteriorate a bit. This shows that the model-on-demand approach provides good support for a straightforward nonlinear modeling where the need for prior physical knowledge is reduced.

Similar to the hydraulic crane in Subsection 5.8.2, the MOD simulation might have performed better if more measurements were available. Especially if there had been more measurements located at low levels in the vessel, since the validation data, as is clear from Figure 5.13, represent the dynamics around the level $h = 50$.

5.9 Conclusions

We have in this chapter demonstrated that the model-on-demand approach is a useful method for predicting and simulating nonlinear systems when sufficiently large datasets are available. A number of nonlinear model structures and associated design issues have been evaluated from a MOD perspective and we have concluded that the different NARX structures should be considered as the default choice in this context. Furthermore, it has been demonstrated that the prediction/simulation result in some situations can be enhanced by tuning certain parameters like the the scaling matrix.

The usefulness of the MOD algorithms has been illustrated in several numerical simulations. For all presented applications we have shown that the prediction errors are in order of magnitude directly comparable to more established modeling tools such as artificial neural nets and fuzzy identification. However, we have at the same time shown that the demand for large datasets is more obvious in the MOD case than for traditional parametric modeling approaches.

6

Applications to Frequency-domain Identification

In Chapter 5 we applied nonparametric methods to the time-domain prediction/identification problem. In this chapter we will instead focus our attention on the field of frequency-domain identification, which traditionally has been a natural area for nonparametric smoothing methods.

6.1 Introduction

Estimating the *spectrum* of a measured signal and the *frequency function* of a linear system are classical problems, see, e.g., Brillinger (1981), Jenkins and Watts (1968), Bendat and Piersol (1980), Ljung (1999) and Stoica and Moses (1997) . The methods can be divided into *parametric* ones, which estimate the parameters of an imposed parametric model (see Chapter 2), and then compute the corresponding frequency-domain counterparts, and *nonparametric* ones, that essentially smooth initial estimates of the frequency-domain quantities – in one way or another.

The procedures all involve some parameters that govern the resolution of the estimate. For parametric methods, the chosen model orders serve as such parameters. For nonparametric methods, the width of some kind of smoothing window (i.e., kernel) has to be chosen (see Chapters 3 and 4). The selection of such parameters of course reflects a bias-variance trade-off. With better resolution, (i.e., smaller bias), fewer data points can be involved in the estimate which lead to higher variance. There are many ways to strike this balance. Traditional methods have typically relied on subjective methods, like visual inspection of the estimate, for the final choice.

We shall here apply and extend the adaptive nonparametric methods described in the earlier chapters to the frequency-domain smoothing problems. This provides two potential advantages in the choice of resolution parameters:

- We can use bandwidth selection procedures that give automatic choices of the optimal value.
- We can consider *local* resolution parameters, so that the resolution is allowed to be frequency-dependent.

The chapter is organized as follows: Section 6.2 reviews the spectral estimation problem which is a classical application for nonparametric methods. The overview covers both the traditional setup as well as more recent treatments of the problem. The contributions to this part are minor, and it is mainly included here for reference and because of its close relationship to the subsequent material. The main contribution is instead Section 6.3, which deals with frequency response estimation. The traditional approach is briefly reviewed, and we show that several enhancements can be made by incorporating ideas and methods from earlier chapters. Section 6.4 discusses several computational issues associated with the estimation methods and shows how the special properties of the frequency data can be used to speedup the estimation. Section 6.5, finally, gives some concluding remarks. Parts of the material have earlier been published in Stenman *et al.* (1999)

6.2 Spectral Estimation

The *spectrum* $\Phi_u(\omega)$ of a stationary signal $u(t)$ is defined as the Fourier transform of its autocovariance function, i.e.,

$$\Phi_u(\omega) \triangleq \sum_{\tau=-\infty}^{\infty} R_u(\tau)e^{-i\omega\tau}, \tag{6.1}$$

where

$$R_u(\tau) \triangleq \mathrm{E}(u(t) - \mathrm{E}\,u(t))(u(t-\tau) - \mathrm{E}\,u(t-\tau)). \tag{6.2}$$

The spectrum describes the frequency content in the signal and is useful for detecting periodicities and cyclic patterns. The concept is closely connected to Fourier techniques, as seen by the following relationship,

$$\Phi_u(\omega) = \lim_{N\to\infty} \mathrm{E}\left\{ \frac{1}{N} \left| \sum_{t=1}^{N} u(t)e^{-i\omega t} \right|^2 \right\}, \tag{6.3}$$

which holds provided $R_u(\tau)$ decays sufficiently rapidly to zero.

In this section we will consider different frequency-domain-based approaches for estimating the spectrum from a *finite* data sequence,

$$u(t), \qquad t = 1, \ldots, N.$$

The raw material for doing so is either the *periodogram*, which is defined as the normed absolute square of the discrete Fourier transform of $u(t)$,

$$\hat{\hat{\Phi}}_{u,p}(\omega) \triangleq \frac{1}{N}\left|\sum_{t=1}^{N} u(t)e^{-i\omega t}\right|^2 = \frac{1}{N}\left|U_N(\omega)\right|^2, \tag{6.4}$$

or the *correlogram*

$$\hat{\hat{\Phi}}_{u,c}(\omega) \triangleq \sum_{\tau=-N+1}^{N-1} \hat{R}_u(\tau)e^{-i\omega\tau}, \tag{6.5}$$

where $\hat{R}_u(\tau)$ denotes an estimate of the autocovariance function (6.2). However, it can be shown (see Stoica and Moses (1997)) that (6.4) and (6.5) will coincide if $\hat{R}_u(\tau)$ is computed using the so-called *standard biased autocovariance function estimate*,

$$\hat{R}_u(\tau) = \frac{1}{N}\sum_{t=\tau+1}^{N} u(t)u(t-\tau), \qquad 0 \le \tau \le N-1, \tag{6.6}$$

with

$$\hat{R}_u(-\tau) = \hat{R}_u(\tau), \qquad 0 \le \tau \le N-1. \tag{6.7}$$

Hence we will in the following denote both the periodogram and the correlogram with the unified notation

$$\hat{\hat{\Phi}}_u(\omega) = \hat{\hat{\Phi}}_{u,p}(\omega) = \hat{\hat{\Phi}}_{u,c}(\omega),$$

and their properties can be investigated simultaneously. Let us first, however, illustrate the previously introduced concepts with an example, taken from Stoica and Moses (1997).

Example 6.1 Periodogram for a wide-band ARMA process

Consider the ARMA process

$$u(t) = \frac{B(q^{-1})}{A(q^{-1})}\epsilon(t) = H(q^{-1})\epsilon(t),$$

where

$$A(q^{-1}) = 1 - 1.3817q^{-1} + 1.5632q^{-2} - 0.8843q^{-3} + 0.4096q^{-4},$$
$$B(q^{-1}) = 1 + 0.3544q^{-1} + 0.3508q^{-2} + 0.1736q^{-3} + 0.2401q^{-4},$$

and where $\{\epsilon(t)\}$ is an independent and Gaussian distributed random sequence with unit variance. It is straightforward to show that its spectrum is given by

$$\Phi_u(\omega) = \left|H(e^{i\omega})\right|^2.$$

The spectrum (dashed curve) and the periodogram (solid curve) for a realization of length $N = 512$ are depicted in Figure 6.1. ❑

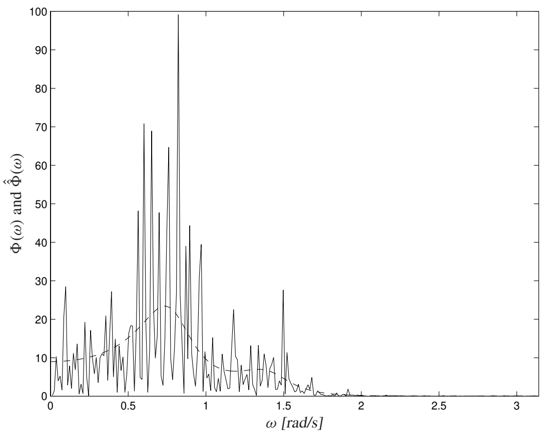


Figure 6.1 *Spectrum (dashed curve) for the ARMA process in Example 6.1 and the periodogram (solid curve) for a realization of length 512.*

6.2.1 Properties of the Periodogram

The periodogram and the correlogram are approximately asymptotically unbiased estimates of the spectrum. However, they are not *consistent* estimates, i.e., their variance do not tend to zero as N increases. This can for instance be seen from Figure 6.1, which shows that the periodogram is a very erratic function of frequency, but that its mean is approximately equal to the true spectrum. The reason for the bad properties of the periodogram could heuristically be explained as follows:

- The covariance estimate $\hat{R}_u(\tau)$ will be a poor estimate of $R_u(\tau)$ for large lags τ, since it then is a sum of only a few lag products divided by N. These "bad" estimates are incorporated in the Fourier transform through the sum in (6.5), thus creating an overall bad and noisy estimate.

- In addition, the bad covariance estimates $\hat{R}_u(\tau)$ will also cause that $\hat{\hat{\Phi}}_u(\omega)$ will be *biased* if $R_u(\tau)$ decays slowly to zero. This follows from the fact that $\hat{R}_u(\tau)$ will be much closer to zero than $R_u(\tau)$ is for large lags, which yields that the bias will be significant if $R_u(\tau)$ is not close to zero in this region (Stoica and Moses, 1997).

In particular, if

$$\omega_k \triangleq \frac{2\pi k}{N}, \qquad k = 0, \dots, N-1$$

denotes the grid of Fourier frequencies, it can be shown that the periodogram is asymptotically exponentially distributed (or, equivalently, $\chi^2(2)$ distributed) with mean $\Phi_u(\omega_k)$, and that the estimates at different frequencies are approximately independent (Brockwell and Davis, 1987). That is, the periodogram can be modeled as

$$\hat{\hat{\Phi}}_u(\omega_k) = \Phi_u(\omega_k) \cdot V_k + r_k, \tag{6.8}$$

where $r_k \to 0$ as $N \to \infty$ and the V_k's are independent random variables having the standard exponential distribution for $k = 1, \ldots, \lfloor (N-1)/2 \rfloor$. Here $\lfloor \cdot \rfloor$ denotes the integer part of the expression. Moreover, V_0 and $V_{\lfloor N/2 \rfloor}$ (if N is even) have χ_1^2-distribution. Thus, and because of the symmetry of the spectrum, it is natural to only consider the periodogram at the frequencies

$$\omega_k, \qquad k = 1, \ldots, n,$$

where $n = \lfloor (N-1)/2 \rfloor$.

6.2.2 Smoothing the Periodogram

Consistent estimates of the spectrum (6.1) can be obtained by smoothing the periodogram. There are roughly three main approaches for doing so (Fan and Gijbels, 1996). The first approach, upon which also most of the traditional methods are based, is to smooth the periodogram *directly*. The classical solution here is the *Blackman-Tukey* procedure (Blackman and Tukey, 1958), which reduces the variability in the *time domain* by introducing a weighting when (6.5) is formed, so that covariance estimates for large lag values τ receive smaller weights;

$$\hat{\Phi}_u(\omega) = \sum_{\tau=-\gamma}^{\gamma} w_\gamma(\tau) \hat{R}_u(\tau) e^{-i\omega\tau}. \tag{6.9}$$

The weighting $w_\gamma(k)$ is usually referred to as the *lag window* and it typically decays to zero with increasing $|\tau|$. The lag window controls the trade-off between frequency resolution and noise suppression. One of the most commonly used windows in spectral analysis is the *Hamming* window,

$$w_\gamma(\tau) = \begin{cases} \frac{1}{2}\left(1 + \cos\frac{\pi\tau}{\gamma}\right), & |\tau| < \gamma, \\ 0, & |\tau| \geq \gamma, \end{cases} \tag{6.10}$$

which is depicted in Figure 6.2 (a). Other popular lag windows include the *Bartlett* window,

$$w_\gamma(\tau) = \begin{cases} 1 - \frac{|\tau|}{\gamma}, & |\tau| < \gamma, \\ 0, & |\tau| \geq \gamma, \end{cases} \tag{6.11}$$

and the *Parzen* window,

$$w_\gamma(\tau) = \begin{cases} 1 - \frac{6\tau^2}{\gamma^2}\left(1 - \frac{|\tau|}{\gamma}\right), & |\tau| < \frac{\gamma}{2}, \\ 2\left(1 - \frac{|\tau|}{\gamma}\right)^3, & \frac{\gamma}{2} \leq |\tau| < \gamma, \\ 0, & |\tau| \geq \gamma. \end{cases} \tag{6.12}$$

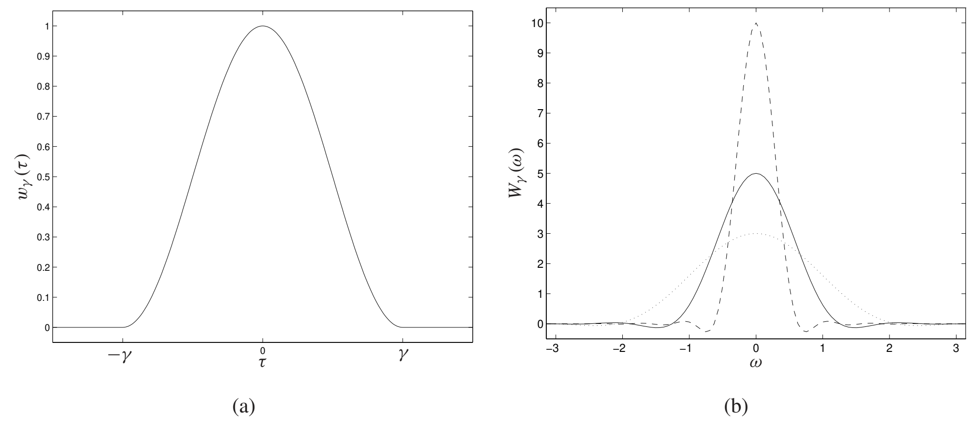


Figure 6.2 *(a) The Hamming lag window. (b) Hamming frequency windows of different widths γ. Solid: $\gamma = 5$, Dashed: $\gamma = 10$, Dotted: $\gamma = 3$.*

In the frequency domain, the Blackman-Tukey procedure (6.9) is equivalent to convolving the periodogram with the frequency-domain counterpart $W_\gamma(\omega)$ (the so-called *frequency window*) of the lag window $w_\gamma(\tau)$, that is,

$$\hat{\Phi}_u(\omega) = W_\gamma * \hat{\hat{\Phi}}_u(\omega) = \int_{-\pi}^{\pi} W_\gamma(\omega - \xi)\hat{\hat{\Phi}}_u(\xi)\, d\xi. \tag{6.13}$$

The frequency window is typically designed as a function centered about $\xi = 0$ which is characterized by the properties (Ljung, 1999)

$$\int_{-\pi}^{\pi} W_\gamma(\xi)\, d\xi = 1, \quad \int_{-\pi}^{\pi} \xi W_\gamma(\xi)\, d\xi = 0, \quad \int_{-\pi}^{\pi} \xi^2 W_\gamma(\xi)\, d\xi = M(\gamma), \tag{6.14a}$$

$$\int_{-\pi}^{\pi} W_\gamma^2(\xi)\, d\xi = \frac{1}{2\pi}\bar{W}(\gamma), \quad \int_{-\pi}^{\pi} |\xi|^3 W_\gamma(\xi)\, d\xi = C_3(\gamma). \tag{6.14b}$$

See Section 3.2.1 for a comparison with the corresponding properties for kernel functions. The Hamming frequency window is given by

$$2\pi \cdot W_\gamma(\omega) = \frac{1}{2}D_\gamma(\omega) + \frac{1}{4}D_\gamma\left(\omega - \frac{\pi}{\gamma}\right) + \frac{1}{4}D_\gamma\left(\omega + \frac{\pi}{\gamma}\right), \tag{6.15}$$

where

$$D_\gamma(\omega) \triangleq \frac{\sin\left(\gamma + \frac{1}{2}\right)\omega}{\sin \omega/2}, \tag{6.16}$$

and is depicted in Figure 6.2 (b) for different values of γ.

As seen from (6.13), the Blackman-Tukey method can be interpreted in the frequency domain as a nonparametric smoother of the same fashion as the methods described in Chapters 3 and 4. (The integral is in practice implemented as a sum). However, the smoothing (or resolution) parameter γ has traditionally been tuned by *ad hoc* methods, usually by visual inspection of the results.

By adapting the nonparametric methods described in earlier chapters to this framework, we will instead be able to obtain adaptive methods that tune such parameters by automatic procedures. In addition, if the local adaptive methods are used, a further benefit is that the tuning can be done *locally*, i.e., different resolutions can be used in different frequency bands.

The direct approach of smoothing the periodogram leads to a regression model of the standard form (1.4), i.e., $Y_k = m(X_k) + \epsilon_k$, with

$$Y_k = \hat{\hat{\Phi}}_u(\omega_k), \quad X_k = \omega_k, \qquad k = 1, \dots, n.$$

This provides the spectral estimate

$$\hat{\Phi}_u(\omega) = \hat{m}(\omega), \tag{6.17}$$

where $\hat{m}(\omega)$ is computed using some suitable local smoothing approach such as local polynomials or the model-on-demand methods.

One difficulty, though, which can be seen from (6.8), is that the periodogram is highly *heteroscedastic*, i.e., that the variance varies with the amplitude of the spectrum. This sometimes makes it hard to select the smoothing parameter γ appropriately. The second and third approaches for smoothing the periodogram is therefore to smooth the *logarithm* of the periodogram, which is considered to have a more homogeneous variance. This can be done either by direct smoothing of the log-periodogram (Wahba, 1980) or by using a localized maximum-likelihood smoother (Kooperberg *et al.*, 1995; Fan and Kreutzberger, 1998).

6.2.3 Smoothing the Log-periodogram

By taking the logarithm of the periodogram model (6.8), one obtains

$$\log \hat{\hat{\Phi}}_u(\omega_k) = \log \Phi_u(\omega_k) + \epsilon_k + \tilde{r}_k \tag{6.18}$$

where $\epsilon_k = \log V_k$, and

$$\tilde{r}_k = \log\left(1 + \frac{r_k}{\Phi_u(\omega_k)V_k}\right)$$

is an asymptotically negligible term. Since the V_k's asymptotically have the standard exponential distribution, it follows that ϵ_k is distributed according to

$$f_\epsilon(x) = e^{-e^x + x}. \tag{6.19}$$

Hence it is rather straightforward to show that

$$\mathrm{E}\,\epsilon_k = -C_0, \quad \text{and} \quad \mathrm{Var}\,\epsilon_k = \frac{\pi^2}{6},$$

where $C_0 \approx 0.57722$ denotes an Euler constant. See, for instance, Davis and Jones (1968). A straightforward approach is thus to ignore the asymptotically negligible term $\tilde{r}_k$ in (6.18). This again leads to a regression model of standard type but with

$$Y_k = \log \hat{\hat{\Phi}}_u(\omega_k) + C_0, \quad X_k = \omega_k, \qquad k = 1, \dots, n,$$

and with a zero mean noise term ϵ_k having variance $\sigma_k^2 \approx 1.6449$. Thus the local polynomial smoothing methods described in earlier chapters can be employed. Since the variance is known in this case, it might be preferable to use the C_p criterion. However, it should work equally well with other goodness-of-fit criteria such as AIC.

When the log-spectrum has been estimated at a specific frequency ω, the corresponding spectral estimate is obtained by the inverse transformation of (6.18), that is,

$$\hat{\Phi}_u(\omega) = \exp\{\hat{m}(\omega)\}. \tag{6.20}$$

6.2.4 Maximum-likelihood-smoothed Log-periodogram

The approach of local polynomial smoothing of the log-periodogram described in the previous subsection is not efficient, since the distribution of ϵ_k is non-Gaussian. It has been shown that the accuracy can be improved by considering a local *maximum-likelihood* estimator for the log-periodogram (Fan and Gijbels, 1996; Fan and Kreutzberger, 1998). Applying the localized maximum-likelihood ideas from Section 3.5 to the log-periodogram model (6.18), gives

$$l(y, m) = -e^{y-m} + y - m, \tag{6.21}$$

i.e., we get the localized log-likelihood function

$$\begin{aligned}\mathcal{L}_\omega(\beta) = \sum_{k=1}^{n} \Bigg[-\exp\Bigg(Y_k - \sum_{j=0}^{p} \beta_j (X_k - \omega)^j \Bigg) \\ + Y_k - \sum_{j=0}^{p} \beta_j (X_k - \omega)^j \Bigg] K_h(X_k - \omega). \end{aligned} \tag{6.22}$$

where

$$Y_k = \log \hat{\hat{\Phi}}_u(\omega_k), \quad X_k = \omega_k, \qquad k = 1, \dots, n.$$

Maximizing over β then provides the log-spectrum estimate

$$\hat{m}(\omega) = \hat{\beta}_0, \tag{6.23}$$

whereby the maximum-likelihood estimate of the spectrum follows from (6.20). It has been shown that the criterion (6.22) is similar to the so-called Whittle likelihood based on the model (6.8), with the extension that weights are introduced in (6.22) to localize the approximation (Fan and Kreutzberger, 1998). The likelihood function $\mathcal{L}_\omega(\beta)$ is a strictly concave function, so a unique maximizer exists.

Bandwidth Selectors for the Likelihood Model

Fan and Gijbels (1996) used asymptotics to derive a plug-in bandwidth selector for the likelihood model, which is implemented as a simple adjustment of a plug-in bandwidth obtained from ordinary least-squares smoothing of the log-periodogram. However, it is also possible to use the classical methods from Section 3.5, despite their rather high computational cost. For this particular application we have that

$$\begin{aligned}\dot{l}(y,m) &= e^{y-m} - 1,\\ \ddot{l}(y,m) &= -e^{y-m}.\end{aligned}$$

It is thus straightforward to implement likelihood cross-validation in accordance with (3.59). For the AIC case we have that

$$\mathrm{E}\left[\dot{l}(Y_k,\hat{m}(X_k))\right]^2 = \int_{-\infty}^{\infty}(e^x-1)^2 e^{-e^x+x}\,dx = 1. \tag{6.24}$$

Hence it follows that

$$\mathrm{AIC}(h) = -\frac{2}{n}\sum_{k=1}^{n} l(Y_k,\hat{m}(X_k)) + \frac{2}{n}\sum_{k=1}^{n}\mathrm{infl}(X_k). \tag{6.25}$$

where the likelihood influence function follows from equation (3.60).

It is of course also possible to derive local variants of the above bandwidth selection schemes, that similar to the methods in Section 4.4.5 select separate bandwidths for each fitting point. This is outside the scope of this overview, though, and we will not explore this topic any further here.

6.2.5 An Example

Let us illustrate the properties of the described nonparametric spectral estimation methods in numerical simulations.

Example 6.2 Wide-band ARMA model, cont'd

Consider again the wide-band ARMA model introduced in Example 6.1. Spectrum, periodogram, log-spectrum and log-periodogram for this model are shown in Figure 6.3. Note, as pointed out earlier, that the periodogram is highly heteroscedastic, while the variance of log-periodogram is more homogeneous.

We will start by considering constant/global bandwidth selection methods from Chapter 3. In Figure 6.4 the periodogram and log-periodogram have been smoothed using local quadratic models and tricube kernels. The bandwidths have been tuned automatically using the corrected AIC for the direct case, AIC for the likelihood case and C_p for the direct log-periodogram smoothing. This has resulted in the global bandwidths 0.57 for the direct smoother, 0.68 for the direct log-periodogram smoother and 0.64 for the maximum-likelihood smoother. The obtained mean absolute deviations (ADE) for this particular realization are 0.90, 0.72 and 0.55, respectively. The corresponding maximum deviations are 4.77, 3.75 and 2.84.

It may be of interest to compare the above result with the performance of the adaptive MOD algorithms from Chapter 4. However, the adaptive algorithms will in most situations produce unsatisfactory results as a consequence of the large variability of the periodogram (and log-periodogram). A possible remedy is to estimate the variance from data using a fixed bandwidth smoother and variance expression (4.42), and utilize this information as an additional weighting in a final smoothing stage.

Figures 6.5 and 6.6 show adaptively smoothed periodograms and log-periodograms$+C_0$ obtained from Algorithm 4.1 using local quadratic models and tricube kernels. The variance functions of the periodogram and log-periodogram have been estimated from data using constant bandwidth smoothers with small bandwidths ($h = 0.05$), and have been plugged in as extra weightings in the algorithm. The bandwidths have been selected locally using the localized C_p criterion. It is clear though that we do not gain that much in this case by allowing the bandwidth to be selected locally. ❑

	Mean ADE	Max ADE
direct smoother	0.90	4.77
log-smoother	0.72	3.75
log-likelihood	0.55	2.84

Table 6.1 *Mean deviations and maximum deviations for Example 6.2.*

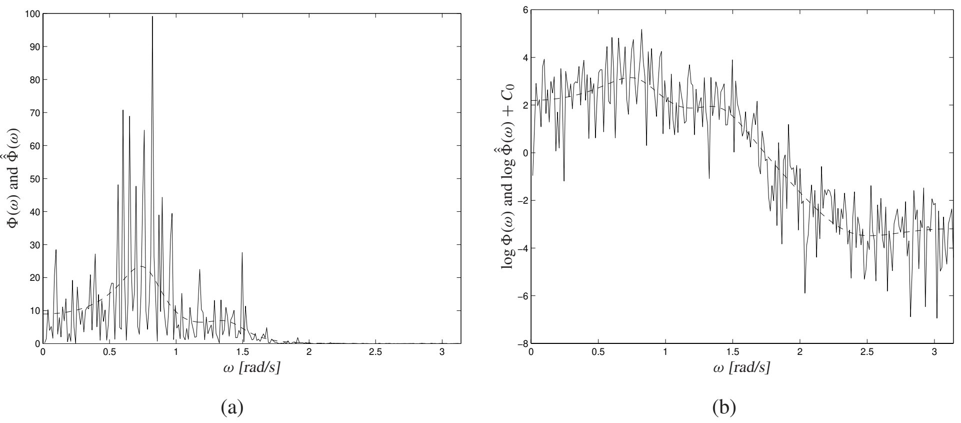


Figure 6.3 *(a) Spectrum and periodogram for the wide-band ARMA model in Example 6.2. (b) Log-spectrum and log-periodogram$+C_0$ for the same model.*

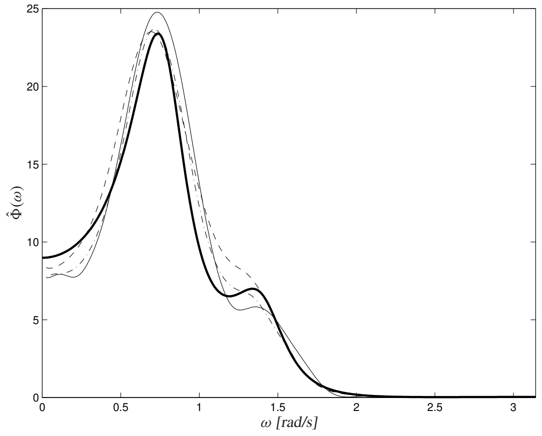


Figure 6.4 *Spectral estimates for the ARMA process in Example 6.2 based on constant bandwidths. Solid thick curve: true spectrum. Solid thin curve: Direct smoothed periodogram. Dashed curve: Smoothed log-periodogram. Dash-dotted curve: Maximum-likelihood-smoothed log-periodogram.*

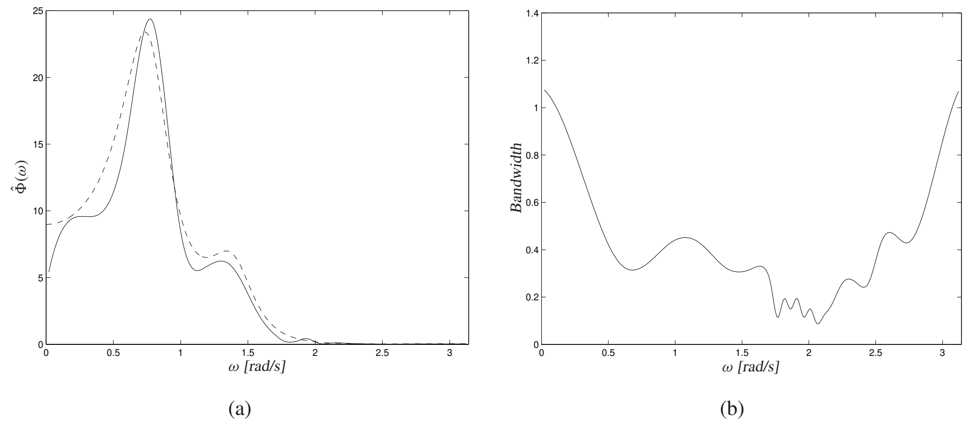


Figure 6.5 *(a) Adaptively smoothed periodogram. Solid line: Estimated spectrum. Dashed line: True spectrum. (b) Selected bandwidths.*

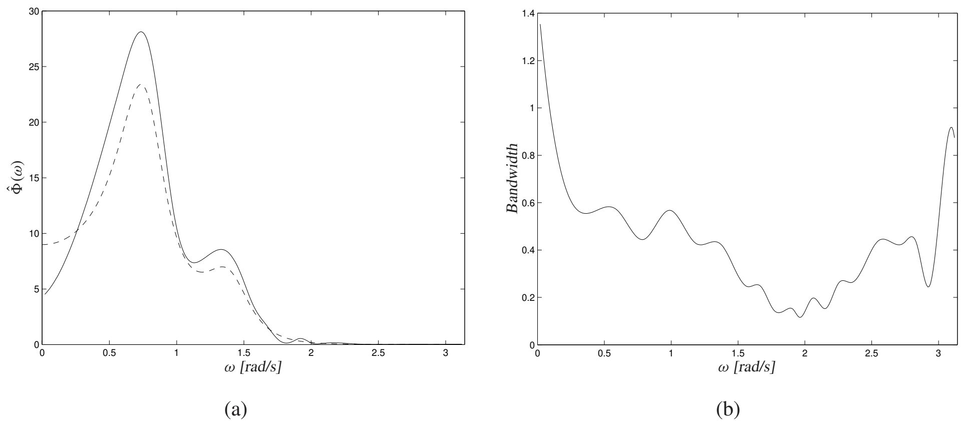


Figure 6.6 *(a) Adaptively smoothed log-periodogram. Solid line: Estimated spectrum. Dashed line: True spectrum. (b) Selected bandwidths.*

6.2.6 Cross-spectrum and Cross-periodogram

Similarly to the definition of spectrum (6.1), the *cross-spectrum* between two signals $u(t)$ and $y(t)$ is defined as the Fourier transform of their cross-covariance function, i.e.,

$$\Phi_{yu}(\omega) \triangleq \sum_{\tau=-\infty}^{\infty} R_{yu}(\tau)e^{-i\omega\tau} \tag{6.26}$$

where

$$R_{yu}(\tau) \triangleq \mathrm{E}(y(t) - \mathrm{E}\, y(t))(u(t-\tau) - \mathrm{E}\, u(t-\tau)). \tag{6.27}$$

Note though that since the cross-covariance in general is not symmetric, the cross-spectrum will normally be complex-valued. In the same way as (6.4), it is possible to form the *cross-periodogram* according to

$$\hat{\hat{\Phi}}_{yu}(\omega) \triangleq \frac{1}{N} Y_N(\omega) U_N^*(\omega). \tag{6.28}$$

The cross-spectrum can be estimated using the Blackman-Tukey procedure as follows;

$$\hat{\Phi}_{yu}(\omega) = \int_{-\pi}^{\pi} W_\gamma(\omega - \xi)\hat{\hat{\Phi}}_{yu}(\xi)\, d\xi. \tag{6.29}$$

An alternative is to use a local polynomial smoother and the norm $\ell(\varepsilon) = |\varepsilon|^2$. Since the cross-periodogram is complex-valued it might not be that meaningful to use the log-periodogram approach in this case.

6.2.7 Summary

We have in this section presented three nonparametric methods found in the literature for estimating signal spectra from periodograms. Which method is the best, still seems to be an open question, however. Fan and Gijbels (1996) claim that the likelihood smoother both has smaller bias at peak regions and smaller asymptotic variance than the other methods, and therefore recommend using it as the default spectral estimator. Simulation results that support this conclusion have been presented in Fan and Kreutzberger (1998).

The large variability of the periodogram makes it hard to apply adaptive methods in a direct manner. However, we have shown that reasonable results can be obtained if the variances are estimated in a pre-smoothing stage.

6.3 Frequency Response Estimation

Another traditional and closely related application of nonparametric estimation methods in the frequency domain, and the main motivation for the work in this chapter, occurs when estimating *frequency functions* of linear and time-invariant systems without imposing parametric models. The methods emanate from spectral estimation techniques as was described in the previous section, and have been thoroughly used and analyzed within signal processing and mathematical statistics. Good overviews of the topic are, for instance, given in Chapter 10 in Jenkins and Watts (1968), in Brillinger (1981), in Chapter 6 in Ljung (1999), and in Gardner (1988).

The basic setup can be described as follows: Consider a stable system described by the input-output relation

$$y(t) = \sum_{k=1}^{\infty} g_k u(t-k) + v(t) = G(q)u(t) + v(t), \tag{6.30}$$

where $\{g_k\}$ denotes the impulse response and $\{v(t)\}$ is a disturbance being a stationary stochastic process with spectrum $\Phi_v(\omega)$. The *frequency function*, $G(e^{i\omega})$, is then defined as

$$G(e^{i\omega}) = \sum_{k=1}^{\infty} g_k e^{-i\omega k}. \tag{6.31}$$

Given measurements from the process,

$$(y(t),\ u(t)), \qquad t = 1, \dots, N,$$

the goal is now to get an estimate of $G(e^{i\omega})$ without imposing any parametric model structure.

Since the input-output relation (6.30) is linear, a natural estimate of the true frequency function is the ratio

$$\boxed{\hat{\hat{G}}_N(e^{i\omega}) \triangleq \frac{Y_N(\omega)}{U_N(\omega)}} \tag{6.32}$$

where, as before,

$$Y_N(\omega) = \sum_{t=1}^{N} y(t)e^{-i\omega t} \quad \text{and} \quad U_N(\omega) = \sum_{t=1}^{N} u(t)e^{-i\omega t}$$

denote the discrete Fourier transform of $y(t)$ and $u(t)$, respectively. The estimate (6.32) is often called the *empirical transfer function estimate*, ETFE, since it is formed directly from the observations without any other assumptions than linearity of the system (Ljung, 1999).

Example 6.3 Empirical transfer function estimate

Consider the linear system

$$G(q) = C\frac{(q^2 - 2r\cos(\phi + \Delta\phi)q + r^2)(q^2 - 2r\cos(\phi - \Delta\phi)q + r^2)}{(q-k)(q^2 - 2r\cos(\phi)q + r^2)^2}, \tag{6.33}$$

where

$$r = 0.95, \qquad \phi = \frac{1.3\pi}{4}, \qquad \Delta\phi = \frac{0.03\pi}{4},$$
$$k = 0.5, \qquad \text{and} \qquad C = 0.5.$$

It is taken from Bodin (1995) and was also treated in Stenman (1997).

A data set was generated according to

$$y(t) = G(q)u(t) + \epsilon(t), \qquad t = 1, \dots, N, \tag{6.34}$$

where the input $\{u(t)\}$ was chosen as a unit PRBS (pseudo-random binary) sequence of length $N = 4096$, and $\{\epsilon(t)\}$ was an independent and normally distributed random sequence with zero mean and standard deviation $\sigma_\epsilon = 0.05$.

The amplitude and phase curves of the true frequency function are shown in Figure 6.7 (a). The amplitude curve has a damped peak around the frequency 1 rad/s. The corresponding ETFE plots are shown in Figure 6.7 (b). ❑

As seen from Example 6.3, the ETFE is a very crude and noisy estimate of the true frequency function. This has been known for a long time and is due to the construction of the estimate using the Fourier transform: We determine as many independent estimates as we have data points. Or equivalently, we have no compression of data. As a consequence of this, we have not imposed any relation between the system's properties at different frequencies.

6.3.1 Properties of the ETFE

The crudeness of the ETFE is caused by the fact that the observations are corrupted by measurement noise which propagates to the estimate through the Fourier transform, and

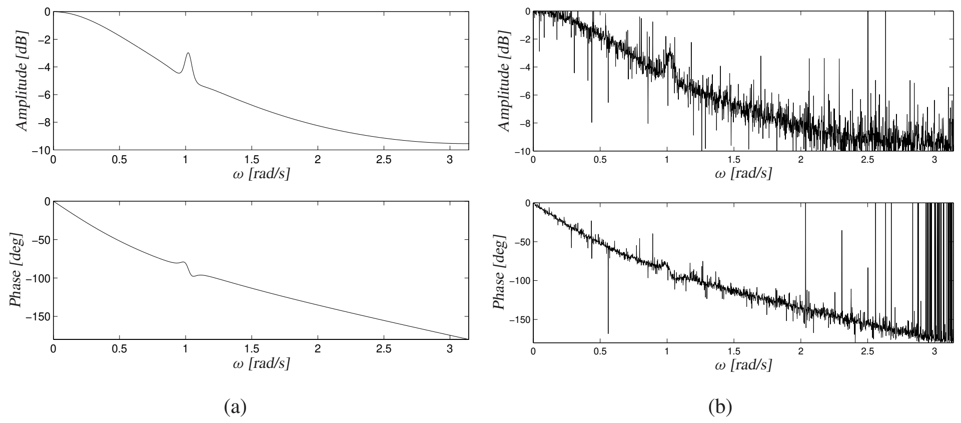


Figure 6.7 *Frequency functions of the system in Example 6.3. (a) The true frequency response. (b) The ETFE.*

leakage[1] effects. Hence the ETFE can be considered as a random variable with certain statistical properties. For future use it is useful to specify these properties more in detail. From Ljung (1999, Lemma 6.1), we have that

$$\mathrm{E}\,\hat{\hat{G}}_N(e^{i\omega}) = G(e^{i\omega}) + R_N^{(1)}, \tag{6.35}$$

$$\begin{aligned}
&\mathrm{E}[\hat{\hat{G}}_N(e^{i\omega}) - G(e^{i\omega})][\hat{\hat{G}}_N(e^{-i\xi}) - G(e^{-i\xi})] \\
&= \begin{cases} \dfrac{1}{|U_N(\omega)|^2}[\Phi_v(\omega) + R_N^{(2)}], & \text{if } \xi = \omega \\[2ex] \dfrac{R_N^{(2)}}{U_N(\omega)U_N(-\xi)}, & \text{if } |\xi - \omega| = \frac{2\pi k}{N}, \quad k = 1, \dots, N-1, \end{cases}
\end{aligned} \tag{6.36}$$

where

$$R_N^{(i)} \to 0 \text{ as } N \to \infty, \quad i = 1, 2. \tag{6.37}$$

From this result we see that the ETFE is an asymptotically unbiased estimate of the true transfer function, but that the variance does not decay to zero when the number of observations, N, increases. Instead it approaches the noise-to-signal ratio, $\Phi_v(\omega)/|U_N(\omega)|^2$, at the frequency in question.

[1]Leakage is an effect of finite data length and means that a single frequency component is "leaking" its energy to nearby frequencies.

6.3.2 Smoothing the ETFE

One way to improve the poor properties of the ETFE is to assume that the values of the true transfer function $G(e^{i\omega})$ at neighboring frequencies are related. The transfer function estimate at a certain frequency ω can then be obtained similarly to (6.13) as a weighted average of the neighboring ETFE values,

$$\hat{G}_N(e^{i\omega}) = \int_{-\pi}^{\pi} W_\gamma(\omega - \xi)\hat{\hat{G}}_N(e^{i\xi})\, d\xi, \tag{6.38}$$

where $W_\gamma(\xi)$ is a frequency window as introduced in Section 6.2. As before γ is a parameter that controls the width of the corresponding lag window, i.e., the *inverse* width of the frequency window. If the frequency window is wide, that is, γ is small, then many neighboring frequencies will be weighted together in (6.38). This will reduce the variance of $\hat{G}_N(e^{i\omega})$. However, a wide window also implies that frequency estimates located far away from ω, whose expected values may differ significantly from $G(e^{i\omega})$, will have a great deal of influence on the estimate. This will cause large bias. Thus, the window width controls the trade-off between the bias and the variance errors.

As pointed out in Chapter 4, when the variances of the observations are known, it is natural to include the inverse variances as an extra weighting in the smoother (recall equation (4.22)). According to (6.36), an improved version of the estimator (6.38) is then given by

$$\hat{G}_N(e^{i\omega}) = \frac{\displaystyle\int_{-\pi}^{\pi} W_\gamma(\omega - \xi)|U_N(\xi)|^2 \hat{\hat{G}}_N(e^{i\xi})\, d\xi}{\displaystyle\int_{-\pi}^{\pi} W_\gamma(\omega - \xi)|U_N(\xi)|^2\, d\xi}, \tag{6.39}$$

provided that $\Phi_v(\omega)$ is approximately constant within the support of the frequency window.

6.3.3 Asymptotic Properties of the Estimate

The frequency function estimate (6.39) has been thoroughly investigated in several books on spectral estimation. See, for example, Chapter 10 of Jenkins and Watts (1968), Chapter 6 of Brillinger (1981) or Chapter 6 of Ljung (1999). From Ljung (1999) we have the following asymptotic formulas (asymptotic both in N and γ):

Bias:

$$\begin{aligned} \mathrm{E}\,\hat{G}_N(e^{i\omega}) - G(e^{i\omega}) = M(\gamma)\left[\frac{1}{2}G''(e^{i\omega}) + G'(e^{i\omega})\frac{\Phi_u'(\omega)}{\Phi_u(\omega)}\right] \\ + O(C_3(\gamma)) + O(1/\sqrt{N}), \end{aligned} \tag{6.40}$$

Variance:

$$\operatorname{Var}\hat{G}_N(e^{i\omega}) = \mathrm{E}\left|\hat{G}_N(e^{i\omega}) - \mathrm{E}\,\hat{G}_N(e^{i\omega})\right|^2 = \frac{1}{N}\bar{W}(\gamma)\frac{\Phi_v(\omega)}{\Phi_u(\omega)} + o(\bar{W}(\gamma)/N). \tag{6.41}$$

Here the differentiations are with respect to ω. Recall also equation (6.14). Now, by combining these expressions, the AMSE of the estimate can be written

$$\begin{aligned} &\mathrm{E}\,|\hat{G}_N(e^{i\omega}) - G(e^{i\omega})|^2 \\ &\qquad \simeq M^2(\gamma)\left|\frac{1}{2}G''(e^{i\omega}) + G'(e^{i\omega})\frac{\Phi_u'(\omega)}{\Phi_u(\omega)}\right|^2 + \frac{1}{N}\bar{W}(\gamma)\frac{\Phi_v(\omega)}{\Phi_u(\omega)} \end{aligned} \tag{6.42}$$

For the windows introduced in Section 6.2.2, $M(\gamma)$ and $\bar{W}(\gamma)$ can for large values of γ be approximated as in Table 6.2 below. Suppose now that it holds that $M(\gamma) = M/\gamma^2$ and

	$M(\gamma)$	$\bar{W}(\gamma)$
Hamming	$\frac{\pi^2}{2\gamma^2}$	0.75γ
Bartlett	$\frac{2.78}{\gamma^2}$	0.67γ
Parzen	$\frac{12}{\gamma^2}$	0.54γ

Table 6.2 *Asymptotic values of $M(\gamma)$ and $\bar{W}(\gamma)$ for different frequency windows. The values are good approximations for $\gamma \geq 5$.*

$\bar{W}(\gamma) = \gamma \cdot \bar{W}$ as in Table 6.2. Then the value of γ that minimizes (6.42) (and thus gives a good trade-off between the bias and variance error) is

$$\gamma_{\text{opt}}(\omega) = \left(\frac{4M^2\left|\frac{1}{2}G''(e^{i\omega}) + G'(e^{i\omega})\frac{\Phi_u'(\omega)}{\Phi_u(\omega)}\right|^2 \Phi_u(\omega)}{\bar{W}\Phi_v(\omega)}\right)^{1/5} \cdot N^{1/5}. \tag{6.43}$$

Using this value of γ, the mean square error decays like

$$\mathrm{MSE}\Big(\hat{G}_N(e^{i\omega})\Big) \sim C \cdot N^{-4/5}. \tag{6.44}$$

This is an expression of a fashion we have already seen earlier in the thesis. However, this is not surprising, since the frequency response estimator (6.39) is formed in the same fashion as the kernel estimators reviewed in Chapter 3.

Now, if all quantities in equation (6.43) were known by the user, they could be plugged into this expression (compare with the plug-in bandwidth selectors in Chapter 3), and the window width γ could be allowed to be frequency dependent. In practice though, these are unknown and the user has to choose a suitable value of γ manually.

Example 6.4

Consider again the system introduced in Example 6.3. Since we in this case know the true frequency function, we can compute the optimal γ according to (6.43). Figure 6.8 shows the optimal and frequency-dependent width, γ_{opt}, for the Hamming lag window, assuming that $\Phi_u(\omega) = 1$ and $\Phi_v(\omega) = \Phi_\epsilon(\omega) = \sigma_\epsilon^2$.

In normal situations, though, the true frequency function is not known (since it is the quantity we want to determine), and the user has to decide upon the right amount of smoothing manually. In Figure 6.9 (a)–(c), the ETFE has been windowed with a Hamming window of different widths. In Figure 6.9 (a) a wide frequency window with $\gamma = 32$ is used. The curves are smooth, but the resolution at the peak is quite poor. In Figure 6.9 (b) a narrower window with $\gamma = 256$ is used. The resolution at the peak is now better, but at other frequencies the curves are noisier. Figure 6.9 (c) shows a compromise with window width $\gamma = 128$. ❑

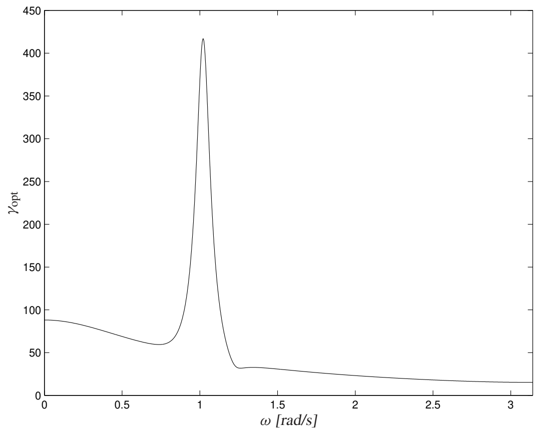


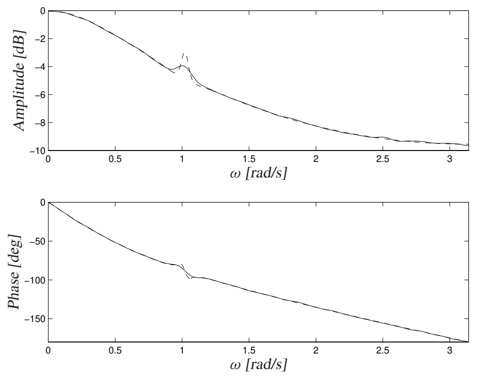
Figure 6.8 *Asymptotic optimal window width for the system in Example 6.3 when using the Hamming lag window.*

Example 6.4 clearly illustrates the trade-off between resolution (narrow window) and variance reduction (wide window), which has to be taken into account when using frequency windows of fixed widths. A typical procedure is to start by taking $\gamma = N/20$ and increase γ until the desired level of details is achieved (Ljung, 1999).

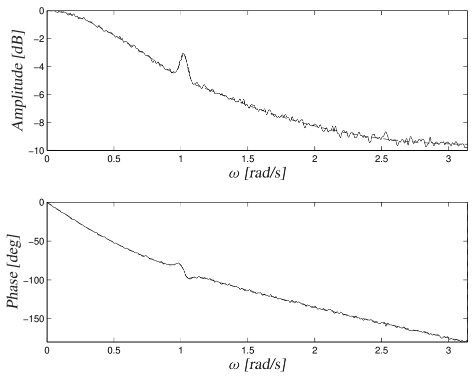
6.3.4 Estimating Frequency Functions by Spectral Analysis

Frequency function estimation is, as mentioned before, closely related to the spectral estimation problem described in Section 6.2. Consider again the system (6.30). It is straightforward to show that the cross-spectrum between $y(t)$ and $u(t)$ is given by

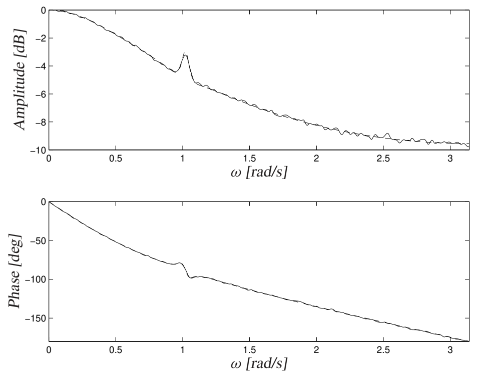
$$\Phi_{yu}(\omega) = G(e^{i\omega})\Phi_u(\omega) \tag{6.45}$$



(a) $\gamma = 32$



(b) $\gamma = 256$



(c) $\gamma = 128$

Figure 6.9 *Windowed frequency responses using Hamming windows of different widths γ. Solid: Windowed estimates. Dashed: True frequency response.*

provided $u(t)$ and $v(t)$ are uncorrelated. An alternate way of estimating $G(e^{i\omega})$ is thus the ratio

$$\hat{G}_N(e^{i\omega}) = \frac{\hat{\Phi}_{yu}(\omega)}{\hat{\Phi}_u(\omega)} \tag{6.46}$$

where $\hat{\Phi}_{yu}(\omega)$ and $\hat{\Phi}_u(\omega)$ are spectral estimates computed using some methods described in Section 6.2. In fact, this way of computing the frequency function estimate using the Blackman-Tukey procedure is one the most commonly used approaches, and is, because it is implemented in the time domain, the main reason for fixed resolution parameters in the traditional setting.

6.3.5 Automatic Smoothing of the ETFE

We saw earlier in this section that a major problem with the classical, Blackman-Tukey-based approaches for estimating the frequency function is that it is hard to find a global choice of resolution parameter that gives a satisfactory smoothing result over the entire span of the frequency axis. In addition, the parameter has to be chosen manually, typically by visual inspection of the resulting estimate.

A more appealing solution would be to have available procedures that provide *automatic* choices of the optimal value, and that enable *local* parameters so the resolution can be frequency dependent. Fortunately, though, this was exactly what we were dealing with in Chapters 3 and 4, and it turns out that the methods described there quite easily can be extended to the ETFE smoothing case.

A Local Polynomial Approach

From Section 6.3.1, it follows that the ETFE, at least asymptotically, can be modeled as the standard regression model, $Y_k = m(X_k) + \epsilon_k$, with

$$Y_k = \hat{\hat{G}}(e^{i\omega_k}) \in \mathbb{C}, \quad X_k = \omega_k \in \mathbb{R}, \qquad k = 1, \dots, n,$$

where, as before, $n = \lfloor (N-1)/2 \rfloor$ (since the ETFE is hermitian), and where ϵ_k is a complex-valued and zero mean disturbance with variance

$$\sigma_k^2 \simeq \frac{\Phi_v(\omega_k)}{|U_N(\omega_k)|^2}. \tag{6.47}$$

It is thus straightforward to apply the local smoothing methods from earlier chapters. The fact, though, that the regression function now is complex-valued requires some attention during implementation and computation, but causes in general no additional conceptual complications. For simplicity we assume a quadratic norm, $\ell(\varepsilon) = |\varepsilon|^2$, which yields that the local regression problems can be solved explicitly using ordinary least squares.

Variance Estimation

In order to use a localized goodness-of-fit criterion, such as C_p, and to enhance the quality of the estimate, it is important to have a good estimate of the variance expression (6.47), which depends upon the noise spectrum $\Phi_v(\omega_k)$ and the Fourier transform of the input $U_N(\omega_k)$.

Now the noise term $v(t)$ in (6.30) is in general not measurable so we cannot estimate $\Phi_v(\omega_k)$ directly. However, if the input-output data are available in addition to the complex ETFE numbers, $v(t)$ can be approximated by means of the residuals,

$$\hat{v}(t) = y(t) - \hat{G}_0(q)u(t), \tag{6.48}$$

where $\hat{G}_0(q)$ denotes a suitable initial estimate of the transfer function. In the frequency domain, this is equivalent to forming an estimate of $\Phi_v(\omega_k)$ by smoothing the “residual periodogram",

$$\hat{\hat{\Phi}}_v(\omega_k) = \frac{1}{N}\left|Y_N(\omega_k) - \hat{G}_0(e^{i\omega_k})U_N(\omega_k)\right|^2, \qquad k = 1, \ldots, n, \tag{6.49}$$

where $\hat{G}_0(e^{i\omega_k})$ denotes an initial frequency response estimate obtained from a presmoothing operation on the ETFE using, for instance, a constant bandwidth. The residual spectrum can then be estimated from (6.49) using any of the smoothing methods described in Section 6.2. Plugging in this quantity into (6.47) results in the variance estimate

$$\hat{\sigma}_k^2 = \frac{\hat{\Phi}_v(\omega_k)}{|U_N(\omega_k)|^2}. \tag{6.50}$$

It is of course also possible to obtain local variance estimates directly from the raw ETFE data, using a non-adaptive smooth with a small bandwidth. Consult Section 4.4.8 for details around this.

Example 6.5

Consider again the ETFE from the system introduced in Example 6.3. Figure 6.10 shows the obtained result after applying Algorithm 4.1 to this dataset using a local quadratic model and the tricube kernel. The noise variance has been estimated from data using the procedure above, with an initial estimate $\hat{G}_0$ obtained from the Blackman-Tukey procedure with $\gamma = 128$. The localized C_p was used for bandwidth selection.

The estimated frequency response is shown in Figure 6.10 (a) along with the true curves. The selected bandwidths are shown in Figure 6.10 (b). As expected, the adaptive approach results in smaller bandwidths near the peak. The larger bandwidths at the boundaries are mainly due to neighborhood truncations and the fact that we are using a nearest-neighbor-type bandwidth.

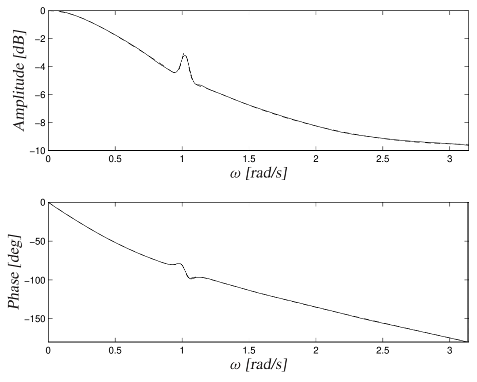
The corresponding asymptotically optimal width of the Hamming frequency window is shown in Figure 6.10 (c). Although this quantity is not directly comparable with the bandwidths in Figure 6.10 (b), we see that the two curves have roughly the same shape. ❑

The above example shows that the adaptive bandwidth selection procedure works and that it gives reasonable choices of bandwidths that very well matches the asymptotically optimal formula (6.43).

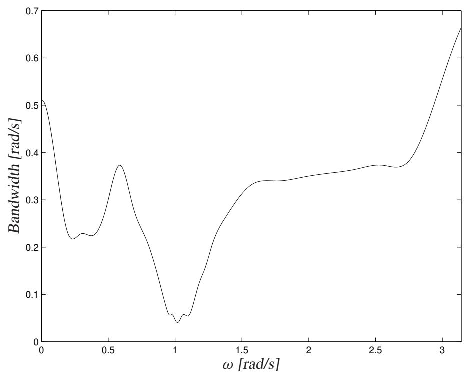
6.3.6 Confidence Bands for the Estimate

The smoothing method described in Subsection 6.3.5 provides point estimates of the frequency function for each frequency. In some situations, though, there might also be need for quantifying the errors associated with each such estimate. A useful diagnostic tool for such purposes is the concept of *confidence intervals*.

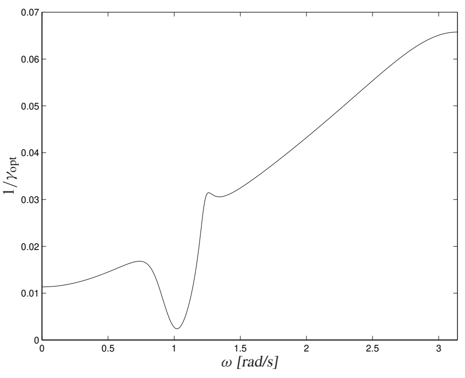
How to construct approximate pointwise confidence bands for the general smoothing case was demonstrated in Section 4.7. However, it turns out that this rather easily also can



(a)



(b)



(c)

Figure 6.10 *Result obtained using an adaptive local polynomial smoother. (a) Adaptively smoothed ETFE. Solid line: Estimate. Dashed line: True frequency response. (b) Selected bandwidths. (c) Asymptotic optimal width of the frequency window according to the expression (6.43).*

be carried over to the frequency domain. Recall that the frequency response effectively is estimated by means of a linear smoothing operation,

$$\hat{G}(e^{i\omega}) = \sum_k W_k(\omega)\hat{\hat{G}}_N(e^{i\omega_k}), \tag{6.51}$$

where the W_k's are determined either explicitly by the optimization approach, or implicitly by a kernel and a polynomial fitting procedure. Hence it follows that the variance of (6.51) is approximately

$$\operatorname{Var} \hat{G}(e^{i\omega}) = \hat{\sigma}^2(\omega) \cdot \|\mathbf{w}(\omega)\|^2, \tag{6.52}$$

where, as before, $\hat{\sigma}^2(\omega)$ denotes a suitable estimate of the complex noise variance at the frequency ω, and $\mathbf{w}(\omega)$ is the vector of smoothing weights. It is thus straightforward to construct approximate confidence bands for $G(e^{i\omega})$ (or more precisely, $\mathrm{E}\,\hat{G}(e^{i\omega})$) similar to the derivation in Section 4.7.

It can be shown that the real and imaginary parts of $\hat{G}(e^{i\omega})$ are asymptotically uncorrelated and jointly normally distributed with variances equal to half the value of (6.52). See, e.g., Ljung (1999). Since a sum of two squared such variables is exponentially distributed, or equivalently, χ^2-distributed with two degrees of freedom, an approximate $(1-\alpha)100\%$ confidence region for the squared magnitude can be obtained as

$$\left|\hat{G}(e^{i\omega}) - G(e^{i\omega})\right|^2 \leq \chi^2_\alpha(2) \cdot \frac{\hat{\sigma}^2(\omega)}{2} \cdot \|\mathbf{w}(\omega)\|^2 = r^2(\omega). \tag{6.53}$$

where $\chi^2_\alpha(2)$ denotes the α quantile of the $\chi^2(2)$ distribution. For the special case of 95% coverage it holds that $\chi^2_{0.05}(2) = 5.99$, so we get the particular value

$$r(\omega) = \sqrt{\frac{5.99}{2}} \cdot \hat{\sigma}(\omega) \cdot \|\mathbf{w}(\omega)\| = 1.73 \cdot \hat{\sigma}(\omega) \cdot \|\mathbf{w}(\omega)\|. \tag{6.54}$$

Note that the confidence region (6.53) here represents a circle with radius $r(\omega)$ around $\hat{G}(e^{i\omega})$ in the space of complex numbers. In order to make use of this information for Bode plotting, translation to corresponding amplitude and phase bands is required as indicated in Figure 6.11. However, from this figure it is easy to realize that

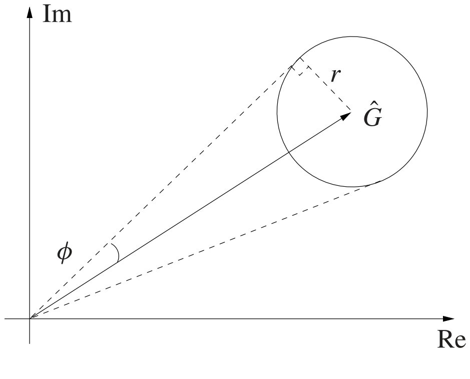


Figure 6.11 Confidence region for the frequency response estimate.

$$\phi(\omega) = \arcsin \frac{r(\omega)}{|\hat{G}(e^{i\omega})|}, \tag{6.55}$$

so we end up with the approximate confidence bands

$$I_{|G|}(\omega) = \Big(|\hat{G}(e^{i\omega})| - r(\omega),\ |\hat{G}(e^{i\omega})| + r(\omega)\Big), \tag{6.56a}$$

$$I_{\arg G}(\omega) = \Big(\arg \hat{G}(e^{i\omega}) - \phi(\omega),\ \arg \hat{G}(e^{i\omega}) + \phi(\omega)\Big), \tag{6.56b}$$

for the amplitude and phase respectively. Note that these expressions do not provide exact information of the uncertainty. They merely represent a way of transferring the information obtained in (6.54) from the complex plane to the Bode plot. To obtain more accurate confidence bands (especially for the phase), a more detailed analysis of the distributions of the amplitude and phase must be performed. Different possibilities exist depending on which quantities we want to display (the magnitude, the logarithm of the magnitude etc). It might also be possible to utilize numerical methods such as point-mass approaches or bootstrapping.

6.3.7 Applications

To further establish the usefulness of the proposed smoothing methods, we will here investigate some additional examples.

Example 6.6 The Åström system revisited

Recall the so-called Åström system, which earlier in Section 5.7.1 was studied in the time-domain prediction framework. This system has also been considered in a nonparametric setting by Ljung (1999) using fixed resolution parameters (i.e., using the Blackman-Tukey procedure). Here we shall instead allow the degree of smoothing to be frequency-dependent.

The system was simulated using a PRBS input of length $N = 1024$ and a Gaussian noise sequence $\{\epsilon(t)\}$ with variance $\sigma^2 = 1$. Figure 6.12 shows the amplitude plot of the ETFE for this dataset. Figures 6.13 (a) and (b) show the smoothed result and the corresponding bandwidths after applying an adaptive smoother with the same configuration as in Example 6.5. As expected, the adaptive scheme selects smaller bandwidths near the peak. The corresponding asymptotic optimal width for the Hamming window is depicted in Figure 6.13 (c) for reference. ❏

Example 6.7 Aircraft Flight Flutter Data

In this section we consider ETFE smoothing of a data set that origins from a real industrial application. The data set has earlier been investigated by Schoukens and Pintelon (1991) and Lindskog (1996).

When new aircrafts are developed they are normally evaluated through quite rigorous test flight programs. Among many other thing it is interesting to examine the mechanical limitations of the different parts of the aircraft. A commonly used measure of this is the so-called flight flutter condition in which an aircraft component at a specific airspeed starts to oscillate.

The experiments are usually performed by attaching special transducers to various points on the airframe, in this particular case the wings, thereby introducing mechanical vibrations artificially. Flying at a predetermined and constant speed, data is collected and analyzed off-line, giving information about whether to allow the the aircraft to fly faster or not. See Schoukens and Pintelon (1991) for further experimental details.

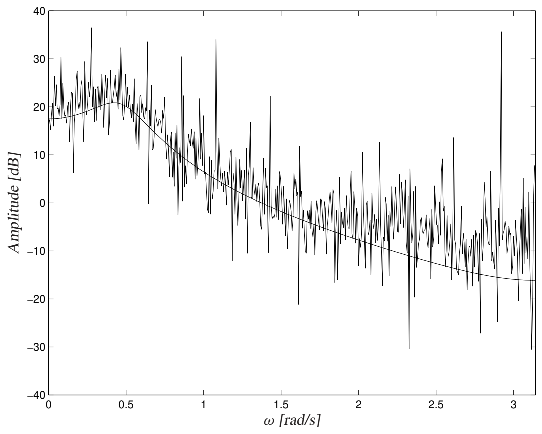


Figure 6.12 *The amplitude plot of the ETFE for the Åström system.*

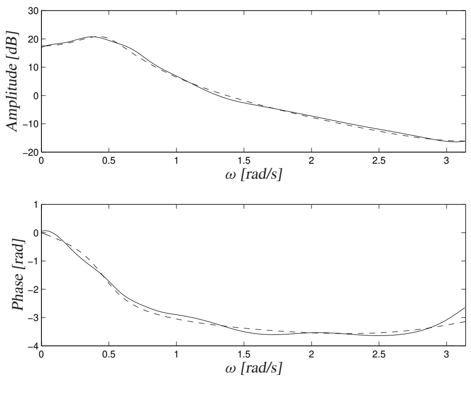
Data recorded during flight flutter test are normally noisy with a quite low signal to noise ratio, and for economical reasons only short data sequences are permitted. The data to be investigated origins from LMS International and was first used by Schoukens and Pintelon (1991).

Flutter data were obtained using burst swept sine (4–40 Hz) excitations that generated a force input $u(t)$ leading to an acceleration response which was taken as the measured output $y(t)$. The data was sampled at $f_s = 100$ Hz and consists of $N = 2048$ samples. The resulting excitation signal $u(t)$ and response signal $y(t)$ are shown in Figure 6.14.

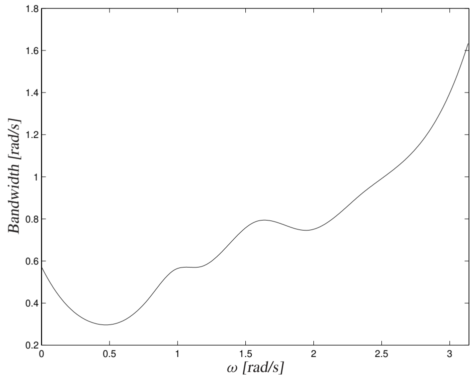
The goal was to model the frequencies in the frequency band of 4 to 11 Hz. The ETFE of the dataset was formed according to (6.32) and is shown as circles in Figure 6.15. The frequency response was estimated using Algorithm 4.1 in the same way as in the previous examples. The result is represented by the solid lines in the figure. The dotted lines represent 95% confidence intervals. A problem with this example similar to other real data applications, however, is that we do not know the true answer, so it is hard to evaluate the quality of the result. ❏

6.4 Computational Aspects

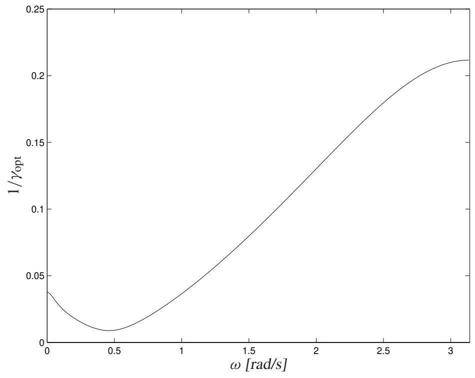
The automatic and adaptive smoothing procedures derived in Chapters 3 and 4 are, as we have seen, very useful in the spectral estimation framework when computing pointwise estimates. However, estimating the spectrum or the frequency function on a large grid of frequencies may be a quite time-consuming task, since we at each estimation point have to solve a number of regression problems for different bandwidths in order to minimize the chosen bandwidth selection criteria. For the local likelihood estimation approach this



(a)



(b)



(c)

Figure 6.13 *Result after smoothing the ETFE of the Åström system. (a) Solid line: estimate, Dashed line: true frequency response. (b) Selected bandwidths. (c) Asymptotic optimal width for a Hamming window.*

becomes even more expensive for large sample sizes, since every local optimization has to be performed using numerical search procedures.

However, the fact that we now are dealing with *scalar* functions and relatively modest dataset sizes, enables us to perform certain simplifications. Considerable speedups can for instance be obtained by considering recursive splitting ideas similar to those of LOCFIT (Loader, 1997):

- Start initially by computing estimates at the boundary points ω_i and ω_j of the grid, which yields the associated bandwidths h_i and h_j.

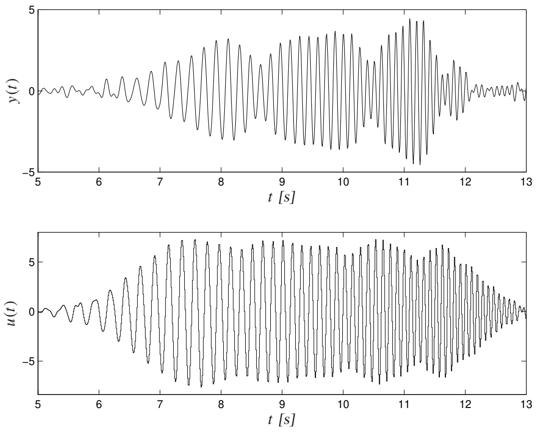


Figure 6.14 *Flight flutter data.*

- If

$$|\omega_i - \omega_j| \geq \zeta \cdot \min(h_i, h_j),$$

  split the interval into two equally sized pieces, and recursively apply the same procedure on the two halves.

This will significantly reduce the number of computations. Estimates at intermediate frequency points can be obtained using cubic spline interpolation between the fitted points. Empirical studies have shown that $\zeta = 0.7$ seems to provide a reasonable trade-off between accuracy and computational complexity.

Example 6.8

Consider again Example 6.5. In fact the frequency function for this system was estimated using the recursive scheme outlined above, but both the estimate and the bandwidth curves were interpolated when illustrated in Figure 6.10. The actual fitting points and selected bandwidths are shown in Figure 6.16 below.

As shown the recursive splitting approach results in more fitting points where smaller bandwidths are needed, that is, near the peak. The actual fits are computed at only 32 frequencies (compared to the original 2048 frequency points), which drastically decreases the computation time. ❏

6.5 Conclusions

In this chapter we have extended the nonparametric smoothing methods from earlier chapters to identification methods, that estimate spectra and frequency functions using automatic, adaptive, and possibly also frequency-dependent choice of frequency resolution.

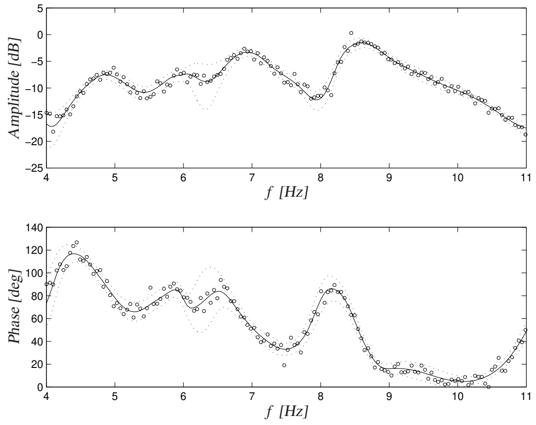


Figure 6.15 *Result after smoothing the flight flutter data. The ETFE values are represented by the circles, the solid lines are the smoothed frequency response, and the dotted lines represent 95% confidence intervals.*

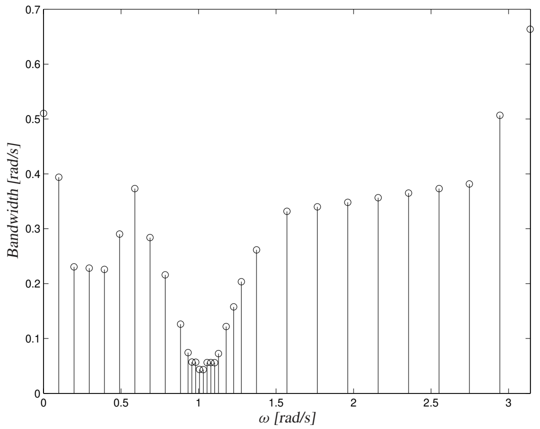


Figure 6.16 *Fitting locations and selected bandwidths for the automatic smoothing in Example 6.5.*

This gives several advantages over traditional spectral analysis techniques. Many frequency functions exhibit fine details to different degrees in different frequency bands. The local polynomial approach thus gives a useful alternative to multi-resolution techniques, based, e.g., on wavelets. We have also demonstrated how an automated procedure, based on a local goodness-of-fit criterion, leads to good choice of bandwidths that very well matches the asymptotically optimal choice.

7

Applications to Control

7.1 Introduction

A natural application for the local modeling procedures treated in the previous chapters is *control* of nonlinear processes. Control theory is a quite general discipline that deals with the problem of forcing the behavior of dynamical systems to achieve certain goals. The objective could for instance be to maintain the system output $y(t)$ at a desired setpoint regardless of the disturbances that affect the system (*regulation*), or to follow a given reference signal $r(t)$ (*tracking*). The most established and well developed part of control theory deals with linear, time-invariant systems, and time- and frequency-domain methods for synthesis and analysis are well established. When turning to the nonlinear domain, however, few constructive methods exist for a systematic controller design.

In this chapter we shall utilize both linear and nonlinear methods for nonlinear control design and we will in particular study how the time-domain methods of Chapter 5 can be used as modeling tools in this context. The intention of the presentation, though, is not to specify complete and ready-to-use solutions to the nonlinear control problem. Instead we will just briefly discuss how it can be solved by pointing out different possibilities and directions for future research.

For simplicity reasons and ease of notation, we will here restrict ourselves to single-input, single-output (SISO) systems of NARX type as described in Section 5.3.2. That is, we assume system descriptions of the form

$$y(t) = m(\varphi(t)) + \epsilon(t), \qquad t = 1, \dots, N \tag{7.1}$$

where as before $m(\cdot)$ is an unknown nonlinear mapping, $\varphi(t)$ is a regression vector consisting of lagged input-output data, and $\epsilon(t)$ is a noise term which typically is modeled as a sequence of i.i.d. random variables with zero means and variances σ_t^2. As a default model for (7.1) we will consider the local linear structure;

$$\boxed{m(\varphi(k)) = \beta_0 + \beta_1^T(\varphi(k) - \varphi(t)),} \tag{7.2}$$

which for each operating point $\varphi(t)$ will be fitted to the data $\{(y(k), \varphi(k))\}$ belonging to a neighborhood of $\varphi(t)$. In earlier chapters, when dealing with predictors, we have almost exclusively utilized the constant term β_0 in this polynomial expansion as an estimate. Here we will also make use of the input-output linearization around $\varphi(t)$ that (7.2) provides.

The organization of the chapter is as follows: Section 7.2 considers traditional adaptive control methods and discusses how local polynomial models can be adopted to this framework. Section 7.3 investigates several prediction-based methods. Section 7.4 generalizes these ideas to the *model predictive control* framework, and Section 7.5, finally, provides conclusions and discusses possible extensions of the presented methods.

7.2 Adaptive Control

The problem of controlling dynamical systems where the plant parameters are unknown or time-varying, has attracted considerable attention over the past 20–30 years, and is usually referred to as *adaptive control* (Narendra and Annaswamy, 1989; Åström and Wittenmark, 1995). Adaptive control algorithms can roughly be divided into two major classes – *direct* and *indirect* algorithms. In an *indirect* algorithm, it is supposed that the process model can be thought of as being parameterized by a finite-dimensional parameter vector θ;

$$m(\varphi(t)) = m(\varphi(t), \theta),$$

and that there is an estimator available that generates a sequence of estimates, $\{\hat{\theta}(t)\}$, of the plant parameters. These are fed to a design procedure

$$\varrho(t) = \chi\left(\hat{\theta}(t)\right), \tag{7.3}$$

that given a certain controller structure maps the plant parameters to corresponding controller parameters $\varrho(t)$. The actual controller is then implemented as

$$u(t) = \kappa(\eta(t), \varrho(t)), \tag{7.4}$$

where $\eta(t)$ is a vector of measurements constructed from past reference signals and input-output measurements of the process, and $u(t)$ is the resulting controller output, see Figure 7.1.

When using *direct* methods, it is instead assumed that it is possible to reparameterize the plant in terms of the controller parameters. The controller can then be estimated directly from data, i.e., the design mapping $\chi(\cdot)$ will be the identity. In the following, though, we will only consider the indirect case above.

In traditional treatments of adaptive control, it has been assumed that the process is *linear* and *time-invariant* with unknown parameters, or *linear* and *time-variant* with slowly

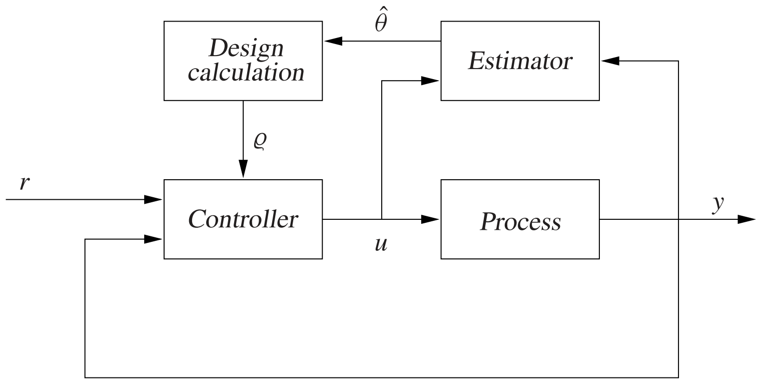


Figure 7.1 *An indirect adaptive controller.*

varying parameters. The standard solution has thus been to estimate the plant parameters θ using some recursive scheme such as recursive least squares (RLS) with forgetting factor, see Section 5.4.1.

However, assuming that the process is mildly nonlinear, an useful alternative could be to replace the recursive estimator with a local polynomial estimator similar to the MOD algorithms derived in Chapter 4. If then also the time stamps are included in the model as extra regressor components and the database is updated on-line during operation, this enables the possibility to use a parameter estimator that is local *both* in time and the regressor space.

The controller design can of course be done in a quite arbitrary way and a wide range of possibilities exist. In this section, though, we will only consider a simple linear approach where the design mapping $\chi(\cdot)$ is implemented using traditional pole placement techniques (Åström and Wittenmark, 1990). The idea is to determine a linear controller that gives desired locations of the closed-loop poles.

7.2.1 Pole Placement

A general linear controller with two degrees of freedom can be written

$$u(k) = \frac{T(q)}{R(q)} r(k) - \frac{S(q)}{R(q)} y(k) \tag{7.5}$$

where $R(q)$, $S(q)$ and $T(q)$ are polynomials in the forward time-shift operator q (Åström and Wittenmark, 1990). The controller consists of two parts; a feedback filter with transfer function $-S/R$ and a feed-forward filter with transfer function T/R, see Figure 7.2. If the process can be modeled by the linear relation

$$A(q)y(k) = B(q)u(k), \tag{7.6}$$

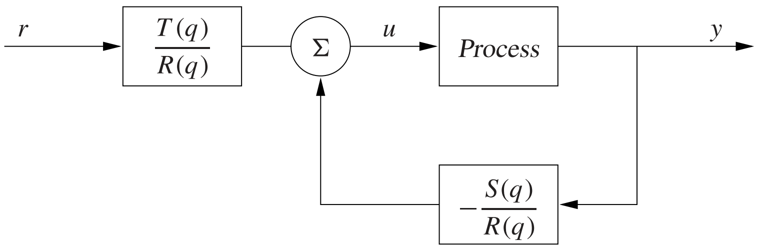


Figure 7.2 *A linear controller with two degrees of freedom.*

the closed-loop transfer function from reference to output is given by

$$y(k) = \frac{B(q)T(q)}{A(q)R(q) + B(q)S(q)} r(k) = \frac{B_m(q)}{A_m(q)} r(k) \tag{7.7}$$

The key idea behind the pole placement design method is now to choose $R(q)$ and $S(q)$ so that the closed-loop system gets desired poles $A_m(q)$. This can be done by solving the so-called *Diophantine* equation (or *Aryabhatta's* identity)

$$A(q)R(q) + B(q)S(q) = A_m(q)A_o(q). \tag{7.8}$$

where $A_o(q)$ is a so-called *observer polynomial* that takes care of possible cancelations between the numerator and denominator polynomials when the closed-loop transfer function is formed. The polynomial $T(q)$ then follows from the relation

$$B(q)T(q) = B_m(q)A_o(q). \tag{7.9}$$

The remaining problem is of course how to model the system dynamics. The basic idea in the context of traditional indirect adaptive control has been to estimate the plant parameters $A(q)$ and $B(q)$ on-line using a recursive estimator, and at each time instant update the controller parameters so that (7.8) and (7.9) are fulfilled.

It is obvious that this estimation stage could be replaced by a local polynomial estimator. Adopting the model-on-demand philosophy to this context, however, requires a minor modification. Since we earlier assumed a local linear model of the type (7.2), the model-on-demand estimator will at each time instant provide an input-output linearization of the form

$$\boxed{A(q)y(t) = B(q)u(t) + \alpha} \tag{7.10}$$

where $A(q)$ and $B(q)$ denote polynomials in the forward time-shift operator q extracted from $\hat{\beta}_1$, and

$$\alpha = \hat{\beta}_0 - \hat{\beta}_1^T \varphi(t)$$

represents an offset term. At a first sight, this seems to require a modification of the pole placement formulas above. However, it is easily seen from (7.10) that by adding an additional term γ to the controller in (7.5), i.e.,

$$u(t) = \frac{T(q)}{R(q)} r(t) - \frac{S(q)}{R(q)} y(t) + \gamma, \tag{7.11}$$

the same closed-loop properties as in (7.7) can be achieved if

$$\gamma = -\frac{1}{B(1)} \alpha. \tag{7.12}$$

Hence it is possible to use pole placement design in the same fashion as in the traditional, recursive estimation case, and we can outline the following algorithm:

ALGORITHM 7.1 MOD-based Pole Placement Control

Given MOD-estimation parameters, database of output/regressor pairs, and controller specifications in terms of desired closed-loop poles A_m, closed-loop zeros B_m and observer polynomial A_o:

Step 1: At the current operating point $\varphi(t)$, estimate a local linear model of the form (7.2). This can be performed using Algorithm 4.1 or Algorithm 4.2 as outlined in Chapter 4.

Step 2: Extract the $A(q)$ and $B(q)$ polynomials and the offset term α from the local model.

Step 3: Solve the Diophantine equation (7.8) in order to get $R(q)$ and $S(q)$, and use (7.9) to determine $T(q)$.

Step 4: Calculate γ from (7.12).

Step 5: Compute the control signal according to (7.11) and apply it to the plant.

The above steps are repeated at each sampling period. ❑

A problem with the pole placement approach is that it sometimes is hard to specify a closed-loop characteristic polynomial that gives a reasonable trade-off between tracking and control effort. A remedy is to consider a linear quadratic approach where the locations of the poles are given indirectly.

7.2.2 LQ Control

In most situations it feels more natural to express the specifications on the controller in terms of the variability of the tracking error and the control signal, rather than in terms of locations of the closed-loop poles and zeros. A natural way of doing so is to introduce the steady-state objective function

$$J = \mathrm{E}\left\{ (y(k) - r(k))^2 + \rho \cdot u^2(k) \right\} \tag{7.13}$$

where E as in earlier chapters denotes mathematical expectation. The control law minimizing (7.13) is usually referred to as the *linear quadratic* controller. A special case is the *minimum-variance* controller which corresponds to $\rho = 0$ (Åström and Wittenmark, 1990).

The linear quadratic control problem is usually solved in state-space formulation. However, it is also possible to solve it using input-output models as follows (provided $A(q)$ and $B(q)$ do not have common factors): The optimal feedback controller that minimizes (7.13) has the form of (7.5), where $R(q)$ and $S(q)$ satisfy the Diophantine equation

$$A(q)R(q) + B(q)S(q) = P(q)q^{n_a}, \tag{7.14}$$

where $P(q)$ is the solution to the spectral factorization problem

$$rP(q)P(q^{-1}) = \rho A(q)A(q^{-1}) + B(q)B(q^{-1}). \tag{7.15}$$

The corresponding $T(q)$ is given by

$$T(q) = \frac{P(1)}{B(1)}q^{n_a}. \tag{7.16}$$

See Åström and Wittenmark (1990) for a proof. An adaptive LQ controller based on local models can therefore be outlined according to the following algorithm:

ALGORITHM 7.2 MOD-based LQ Control

Given MOD-estimation parameters, database of output/regressor pairs, and controller specification in term of a control penalty ρ:

Step 1: At the current operating point $\varphi(t)$, estimate a local linear model. This can be performed using either Algorithm 4.1 or Algorithm 4.2.

Step 2: Extract the $A(q)$ and $B(q)$ polynomials and the offset term α from the local model.

Step 3: Solve the spectral factorization problem (7.15).

Step 4: Solve the Diophantine equation (7.14) in order to get $R(q)$ and $S(q)$, and use (7.16) to compute $T(q)$.

Step 5: Calculate γ from (7.12).

Step 6: Compute the control signal according to (7.11) and apply it to the plant.

The above steps are repeated at each sampling period. ❑

If $A(q)$ and $B(q)$ indeed do have common factors (especially unstable modes), some modifications of the above scheme are required. We will not do it here though. Consult Åström and Wittenmark (1990) for details.

Let us illustrate Algorithm 7.2 with a simple example that will be used throughout the chapter:

Example 7.1 System with state-dependent gain

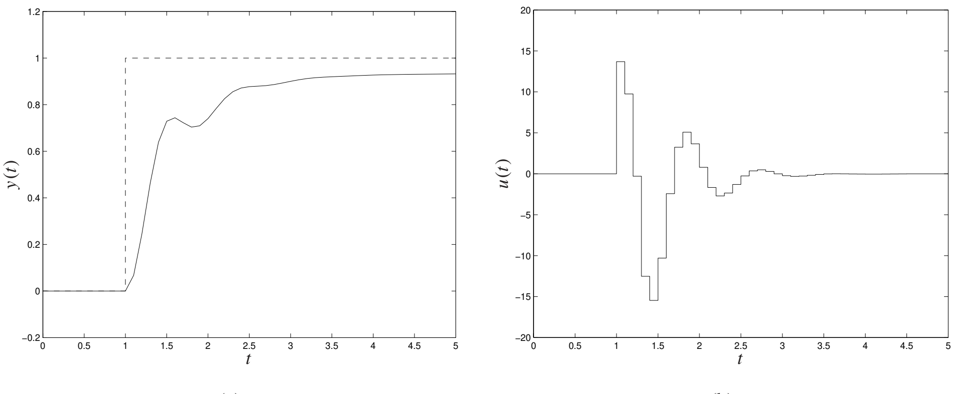
Consider the nonlinear system

$$\ddot{y}(t)\,(1+|y(t)|) = u(t). \tag{7.17}$$

It has the property that the gain of system decreases as the magnitude of the output signal increases. The open-loop system is unstable, so a NARX 221 database consisting of 3000 output and regressor pairs was generated using data collected from a closed-loop experiment using a proportional controller, $u(t) = K(r(t) - y(t))$, with $K = 1$ and a band-limited white noise reference signal $r(t)$ with power 2. The sampling interval was chosen as $T_s = 0.1$ seconds.

As a first attempt, we tried a fixed gain LQ controller with $\rho = 0.01$, based on a global linear ARX model estimated from the dataset. A simulation of the resulting closed loop system is shown in Figure 7.3. This does not look good. The simulated output is unable to reach the setpoint.

In Figure 7.4, the "adaptive" LQ controller of Algorithm 7.2 has instead been used. The system dynamics was estimated locally for each operating point $\varphi(t)$ using MOD-algorithm 4.1 with default values, a local linear model structure and the database above. The control penalty was chosen as in the fixed gain case, i.e., $\rho = 0.01$. In this case the output is able to reach the setpoint. However, the system response is quite slow, and there are small ripples in the control. Later in the chapter we will see how the performance can be improved by considering a model predictive controller. ❑



(a) (b)

Figure 7.3 Step response experiment using a fixed gain LQ controller based on a global estimated model of the system (7.17). (a) Reference signal (dashed) and system output (solid). (b) Control signal.

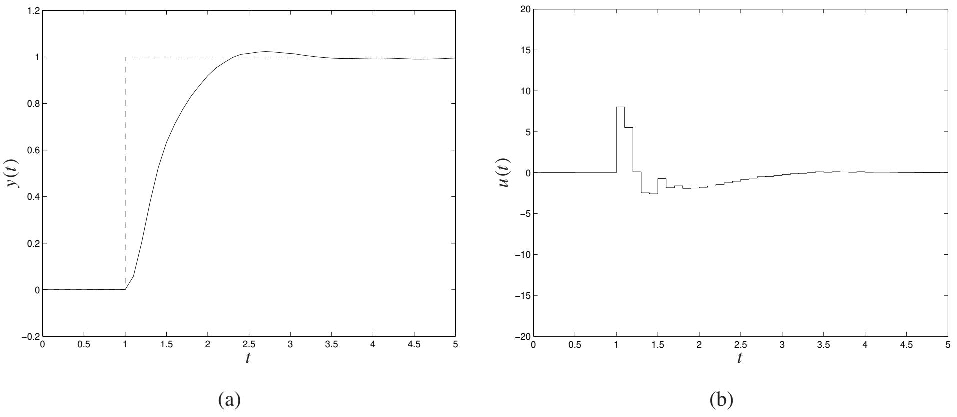


Figure 7.4 *Step response experiment using a local linear estimator and an adaptive LQ controller. (a) Reference signal (dashed) and system output (solid). (b) Control signal.*

7.2.3 Summary

Combining local modeling with traditional adaptive control techniques is, as has been shown, straightforward, and might be a useful approach for controlling mildly nonlinear processes. A classical drawback with recursive estimators in the traditional setup has been the problem of parameter drift, caused by the lack of excitation in the signals that occurs when the system output reaches the setpoint. Adaptive controllers that instead use local estimators like the MOD-algorithms will most likely not have that problem, provided, of course, that the observations stored in the database are of good quality and sufficiently informative.

It is worth pointing out that the combination of local polynomial models and traditional adaptive control techniques presented here is not an entirely new concept. A pole placement algorithm similar to the one in Algorithm 7.1 has earlier been discussed by Bontempi *et al.* (1998). Combinations of LQ control and local models similar to Algorithm 7.2 have been studied by Tanaka (1995), Atkeson *et al.* (1997*b*) and Passino and Yurkovich (1996).

7.3 Prediction-Based Approaches

A very important aspect of a model is its predictive power, i.e., its ability to forecast the future. Controllers that utilizes this feature of a model for on-line selection of control actions are usually referred to as *predictive controllers* and have become more and more popular in recent years.

How the control action is selected by the controller is of course a matter of the criterion used. A typical design goal for process control applications, though, is to require that a good control is one that drives the predicted output k steps ahead close to a given setpoint r, i.e.,

$$y(t+k) \cong r(t+k). \tag{7.18}$$

Several ways of constructing predictive controllers that achieve (7.18) have been suggested in the literature. See for instance Åström and Wittenmark (1995) for a survey from an adaptive control viewpoint. In this contribution we will mainly study how local models can be used in this context. It turns out that there are mainly three different approaches for incorporating the MOD approach into the predictive control framework:

- Predictive control based on input-output linearization around the current operating point.
- Predictive control based on linearization along a possible future trajectory.
- Predictive control based on numerical optimization.

They will all be studied in more detail later on in this section. Nevertheless, essential for each of them is that we have available an explicit expression for the output at future time instants. The derivation of such expressions will thus be the subject for the next subsection.

7.3.1 Expressions for the Output Prediction

The key idea behind prediction-based control methods is to rewrite the process model to obtain an explicit expression for the output at future time instants. Fortunately though, it turns out that this is rather straightforward to do for linear systems on input-output form.

Linear Time-Invariant Systems

We will start initially by making the assumption that the model description (7.10) obtained at time t will be valid over the next k samples, i.e., that the linearized model is time-invariant. This enables the use of established techniques, as found in standard textbooks on stochastic control. See, for instance, Åström and Wittenmark (1990) or Åström and Wittenmark (1995).

It turns out that it becomes easier to do the derivation in the backward time-shift operator q^{-1}. Equation (7.10) can be rewritten as

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \alpha, \tag{7.19}$$

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a},$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b}.$$

The standard trick is now to introduce the identity

$$1 = A(q^{-1})F_k(q^{-1}) + q^{-k}G_k(q^{-1}), \tag{7.20}$$

where $F_k(q^{-1})$ and $G_k(q^{-1})$ are polynomials of degrees $k-1$ and n_a-1 respectively. Substituting (7.20) into the plant model (7.19) yields

$$y(t) = B(q^{-1})F_k(q^{-1})u(t-1) + q^{-k}G_k(q^{-1})y(t) + F_k(1)\alpha,$$

i.e., the k-step-ahead predictor is given by

$$\hat{y}(t+k) = B(q^{-1})F_k(q^{-1})u(t+k-1) + G_k(q^{-1})y(t) + F_k(1)\alpha. \tag{7.21}$$

By partitioning $B(q^{-1})F_k(q^{-1})$ as

$$B(q^{-1})F_k(q^{-1}) = S_k(q^{-1}) + q^{-k}\tilde{S}_k(q^{-1}),$$

where $\deg S_k(q^{-1}) = k-1$ and $\deg \tilde{S}_k(q^{-1}) = n_b - 2$, equation (7.21) can be rewritten as

$$\begin{aligned}\hat{y}(t+k) = {}& S_k(q^{-1})u(t+k-1) \\ &+ \tilde{S}_k(q^{-1})u(t-1) + G_k(q^{-1})y(t) + F_k(1)\alpha.\end{aligned} \tag{7.22}$$

Here the first term depends on future control actions $u(t), \dots, u(t+k-1)$ whereas the remaining terms depend on known, measured quantities only.

Linear Time-Variant Systems

Let us now consider the linear but time-varying process model

$$A_\tau(q^{-1})y(\tau) = B_\tau(q^{-1})u(\tau-1) + \alpha_\tau, \qquad \tau = t+1, \dots, t+k, \tag{7.23}$$

where $A_\tau(q^{-1})$, $B_\tau(q^{-1})$ and α_τ are changing with time. Predictors for such models can probably be found in the vast literature on model predictive control, but we have here made our own derivation.

It turns out that the output predictor associated with the model (7.23) can be found as follows: Introduce the identity

$$\begin{aligned}1 = A_{t+k}(q^{-1}) + f_1 q^{-1}A_{t+k-1}(q^{-1}) + \ldots + f_{k-1}q^{-k+1}A_{t+1}(q^{-1}) \\ + q^{-k}G_{t,k}(q^{-1})\end{aligned} \tag{7.24}$$

which can be interpreted as the time-variant counterpart of (7.20). The coefficients of $F_{t,k}$ and $G_{t,k}$ can now be determined using repeated polynomial division. For instance, for f_1 we have

$$\begin{aligned}&\frac{q\left(1 - A_{t+k}(q^{-1})\right)}{A_{t+k-1}(q^{-1})} \\ &\quad = f_1 + \frac{f_2 q^{-1}A_{t+k-2}(q^{-1}) + \ldots + f_{k-1}q^{-k+2}A_{t+1}(q^{-1}) + q^{-k+1}G_{t,k}(q^{-1})}{A_{t+k-1}(q^{-1})} \\ &\quad = f_1 + \frac{R_1(q^{-1})}{A_{t+k-1}(q^{-1})}.\end{aligned} \tag{7.25}$$

By applying the same procedure on the remainder terms $R_i(q^{-1})$, the rest of the coefficients of $F_{t,k}(q^{-1})$ and $G_{t,k}(q^{-1})$ can be determined. The corresponding output predictor is given by

$$\begin{aligned}\hat{y}(t+k) =& \\ &\Big(B_{t+k}(q^{-1}) + f_1 q^{-1} B_{t+k-1}(q^{-1}) + \ldots + f_{k-1} q^{-k+1} B_{t+1}(q^{-1})\Big)\, u(t+k-1) \\ &+ G_{t,k}(q^{-1}) y(t) + F_{t,k}(q^{-1}) \alpha_{t+k}. \qquad (7.26)\end{aligned}$$

As in (7.22) this expression can be partitioned in one part that depends on future control moves and one part that depends on past measured data only.

$$\begin{aligned}\hat{y}(t+k) = S_{t,k}(q^{-1}) u(t+k-1) + \tilde{S}_{t,k}(q^{-1}) u(t-1)& \\ + G_{t,k}(q^{-1}) y(t) + F_{t,k}(q^{-1}) \alpha_{t+k}.& \qquad (7.27)\end{aligned}$$

7.3.2 Constant Future Control

Let us now try to utilize the just derived expressions in the predictive control framework. However, it is easy to realize from (7.22) and (7.27) that there in general exist infinitely many ways of achieving (7.18) if no additional constraints are put on the future control actions. The simplest choice, though, is to assume that all future control signals within the prediction horizon will remain constant and equal to $u(t)$, i.e.,

$$u(t) = u(t+1) = \ldots = u(t+k-1) = \bar{u}. \qquad (7.28)$$

This approach is usually referred to as *constant future control*. We will in the following discuss how it can be implemented given the three methods listed in the beginning of the section.

Constant Future Control Based on Local Linearizations

The simplest implementation of constant future control follows from the assumption of a time-invariant model description. By assuming that (7.18) holds, we have from (7.22) and (7.28) that

$$r(t+k) = S_k(1)\bar{u} + \tilde{S}_k(q^{-1}) u(t-1) + G_k(q^{-1}) y(t) + F_k(1)\alpha.$$

This gives the control signal

$$\bar{u} = \frac{r(t+k) - \tilde{S}_k(q^{-1}) u(t-1) - G_k(q^{-1}) y(t) - F_k(1)\alpha}{S_k(1)} \qquad (7.29)$$

which is applied to the process. Here we have used the *certainty equivalence* principle, according to which the control $u(t) = \bar{u}$ is determined under the assumption that the model exactly describes the true system. However, at the next sampling instant, a new local model and a new predictor are obtained, which together with (7.29) provides a new value of the control signal. This is an example of the so-called *receding horizon* concept.

Constant Future Control Based on Linearizations along a Trajectory

A natural extension of the above method is to apply the control signal obtained from (7.29) to the process, and make use of the local estimator to obtain a linear but time-varying model over the prediction horizon. It is then possible to use (7.27) in order to determine the constant future control sequence that satisfies (7.18). This gives the control law

$$\bar{u} = \frac{r(t+k) - \tilde{S}_{t,k}(q^{-1})u(t-1) - G_{t,k}(q^{-1})y(t) - F_{t,k}(q^{-1})\alpha_{t+k}}{S_{t,k}(1)} \tag{7.30}$$

which similarly to (7.29) is updated according to the receding horizon principle.

Constant Future Control Based on Optimization

A third and conceptually different approach of solving the constant future control problem follows from the use of a general and nonlinear k-step-ahead predictor. However, since we in the local polynomial modeling framework are unable to obtain a global description of the plant dynamics, it is not possible to obtain an explicit expression for the prediction as a function of future controls. Instead we have to resort to numerical optimization approaches.

In order to achieve the control goal (7.18), it is natural to introduce a loss function that measures the deviation from the desired setpoint. A natural such choice (with desirable analytical properties) is the quadratic loss

$$J = \big(r(t+k) - \hat{y}(t+k)\big)^2. \tag{7.31}$$

The task is now to choose the constant future control sequence (7.28) so that (7.31) is minimized. This can be done using a numerical optimization routine that simulates the system for different choices of $\bar{u}$, and chooses the one that gives the smallest loss value. It is well known, though, that more accurate results and faster convergence could be obtained if the optimization routine is provided with derivative information. Differentiating $J = J(\bar{u})$ with respect to $\bar{u}$ gives

$$\frac{\partial J(\bar{u})}{\partial \bar{u}} = 2\big(\hat{y}(t+k) - r(t+k))\big) \cdot \frac{\partial \hat{y}(t+k)}{\partial \bar{u}}. \tag{7.32}$$

The derivative of the predictor with respect to $\bar{u}$ can be determined using the expression for the time-varying predictor. It follows from (7.27) that

$$\frac{\partial \hat{y}(t+k)}{\partial \bar{u}} = S_{t,k}(1).$$

Hence the derivative of the loss function can be rewritten

$$\frac{\partial J(\bar{u})}{\partial \bar{u}} = 2(\hat{y}(t+k) - r(t+k))S_{t,k}(1). \tag{7.33}$$

The minimization of J can thus be carried out using either a gradient-based scheme,

$$\bar{u}^{(i)} = \bar{u}^{(i-1)} - \mu \frac{\partial J\big(\bar{u}^{(i-1)}\big)}{\partial \bar{u}}, \tag{7.34}$$

where the derivative (7.33) is estimated from the database by treating the sequence of linearizations obtained from the k-step-ahead predictor as a linear and time-varying system, or (if the second order derivative also is computed) a Newton-type scheme

$$\bar{u}^{(i)} = \bar{u}^{(i-1)} - \mu \left[\frac{\partial^2 J\left(\bar{u}^{(i-1)}\right)}{\partial \bar{u}^2} \right]^{-1} \frac{\partial J\left(\bar{u}^{(i-1)}\right)}{\partial \bar{u}}. \tag{7.35}$$

Note though that since the loss function (7.31) in general will have several local minima, it is important to select the initial value $\bar{u}^{(0)}$ with care. One possibility here is to use the value obtained from either (7.29) or (7.30).

Combining local modeling with optimization are not new ideas. Similar optimization approaches for local polynomial models have earlier been investigated by among others Atkeson *et al.* (1997*b*) and Bontempi *et al.* (1998) using names as *deadbeat* and *gradient based* control. However, in their treatments, they have only considered the case $k = 1$ which in general results in too aggressive controllers.

7.3.3 Minimum Effort Control

An alternative to the constant future control approach discussed in the previous subsection, is to determine the control law that fulfills (7.18) while minimizing the control effort,

$$J_{\mathbf{u}} = \sum_{l=0}^{k-1} u^2(t+l) \tag{7.36}$$

over the prediction horizon. This can be solved by regarding (7.18) as a constraint while minimizing (7.36). This topic will not be explored any further in this text however. The interested reader is referred to Åström and Wittenmark (1995) for further details in the linear and time-invariant case.

7.3.4 Summary

Although the predictive-based control approaches are very simple and straightforward they have some shortcomings. Controllers with short horizons k will most likely produce aggressive control actions, and will certainly not work for non-minimum phase systems, whose responses initially will be in the wrong direction. Furthermore, for large horizons, the closed-loop response will typically be slow. A solution is to form a criterion that takes into account the output over a sequence of future time instants, and possibly also includes penalties on the changes in the control signal. This is usually referred to as *model predictive control* and will described more thoroughly in the next section.

7.4 Model Predictive Control

Model predictive control, MPC, is a family of optimal-control related methods that selects control actions by on-line minimization of objective functions. It can be viewed as a generalization of the prediction-based methods described in Section 7.3. As the name indicates, the methods assume a model of the controlled process in order to compute predictions of the future output trajectory. The control action is then optimized so as to force this trajectory to meet certain control goals. There are many variants of model predictive control, for instance, *dynamic matrix control* and *extended horizon control*. See García *et al.* (1989) for a survey. If the process model is estimated on-line and updated in an adaptive control manner, one usually talks about *generalized predictive control*, GPC, (Clarke *et al.*, 1987*a*; Clarke *et al.*, 1987*b*).

Early formulations of MPC used linear models such as step response, impulse response or state-space models, and have turned out to be successful in controlling linear and mildly nonlinear processes. The performance degradation and instability noticed in the presence of strong nonlinearities, though, soon motivated the use of nonlinear models in the on-line optimization. However, both the difficulties in obtaining a good model of the nonlinear process and the excessive computational burden associated with the control optimization have been serious obstacles to widespread industrial implementations. Various simplifications and approximations based on linearization of the nonlinear process have therefore been proposed, see, e.g., Gattu and Zafiriou (1992). In this section we instead present the concept of *model-free predictive control* (or, alternatively, *model-on-demand predictive control*, MoDPC), which extends the MPC/GPC ideas to the model-on-demand framework.

7.4.1 The Basic MPC Formulation

The general predictive control problem can be stated as follows (Meadows and Rawlings, 1997): Given knowledge of the current system state, seek a control that minimizes the objective function

$$J = \phi(\hat{y}(t+N)) + \sum_{k=0}^{N-1} L(\hat{y}(t+k+1), u(t+k), \Delta u(t+k)). \tag{7.37}$$

Of the N future control actions that minimize J, only the first one is applied to the controlled process. When new measurements become available, a new optimization problem is formulated whose solution provides the next control action. This is thus again an instance of the receding horizon principle. A special feature of the formulation (7.37) is the presence of the control increment,

$$\Delta u(t+k) = u(t+k) - u(t+k-1), \tag{7.38}$$

in the objective. In some examples, for instance in process control applications, the change rate of the control action may be restricted. Rather than including the actuator dynamics in the model, it is instead common practice to include penalties on the control increment. Another advantage is that it is straightforward to include hard constraints on the control signal magnitude and the control increment, i.e.,

$$u_{\min} \le u(t+k) \le u_{\max}, \tag{7.39a}$$

$$\Delta u_{\min} \le \Delta u(t+k) \le \Delta u_{\max}. \tag{7.39b}$$

The criterion (7.37) is general and may be chosen to meet a wide range of process objectives such as maximization of profits, minimization of operational costs etc. However, in the sequel of this section we will focus on the specialized case of regulation to a given setpoint $r(t)$ (i.e., *tracking*) using a quadratic criterion,

$$J = \sum_{k=0}^{N-1} Q_e(k)(r(t+k+1) - \hat{y}(t+k+1))^2 \\ + Q_u(k)u^2(t+k) + Q_{\Delta u}(k)\Delta u^2(t+k) \quad (7.40)$$

where $Q_e(k)$, $Q_u(k)$ and $Q_{\Delta u}(k)$ represent weightings on the control error, control signal and control increment magnitudes respectively. If the process model is linear and $Q_{\Delta u}(k) \equiv 0$, it is clear that this criterion is closely related to the familiar linear-quadratic control problem (Bitmead *et al.*, 1990). Note also the striking resemblance with equation (7.13).

Optimization of the objective (7.40) can be quite demanding for large prediction horizons. To decrease the computational complexity it is thus very common to introduce constraints on the future control signals. A commonly used approach is to assume that the control increments are zero after $N_u \leq N$ steps;

$$\Delta u(t+k-1) = 0, \qquad k > N_u. \quad (7.41)$$

It is well known that this also has the effect of producing less aggressive controllers (Meadows and Rawlings, 1997). The quantity N_u is usually referred to as the *control horizon* in the MPC literature.

7.4.2 Optimization Based on Local Linearizations

The most obvious and straightforward way of incorporating local models into the MPC formulation, is to optimize the objective based on the current input-output linearization similar to the approach introduced in Section 7.3. A similar idea was adopted by Gattu and Zafiriou (1992) using state-space models, but we choose here to remain in the input-output framework, both since our obtained models are of this form and since we already have derived predictors for this class of plant models. The derivation will thus follow the standard approach for input-output models, as found in the standard literature on generalized predictive control, GPC, see, e.g., Clarke *et al.* (1987*a*).

Recall again the expression (7.22) for the output prediction. It showed that the predicted output's dependency of future controls can be expressed as

$$\begin{aligned}\hat{y}(t+k) &= S_k(q^{-1})u(t+k-1) + \tilde{S}_k(q^{-1})u(t-1) + G_k(q^{-1})y(t) + F_k(1)\alpha \\ &= S_k(q^{-1})u(t+k-1) + \bar{y}(t+k),\end{aligned}$$

where $\bar{y}(t+k)$ is introduced as a term that sums up all known and measured quantities. By also introducing the notations

$$\hat{\mathbf{y}} \triangleq \begin{pmatrix} \hat{y}(t+1) & \dots & \hat{y}(t+N) \end{pmatrix}^T, \quad (7.42a)$$

$$\tilde{\mathbf{u}} \triangleq \begin{pmatrix} u(t) & \dots & u(t+N-1) \end{pmatrix}^T, \tag{7.42b}$$

$$\bar{\mathbf{y}} \triangleq \begin{pmatrix} \bar{y}(t+1) & \dots & \bar{y}(t+N) \end{pmatrix}^T, \tag{7.42c}$$

$$\tilde{\mathbf{S}} \triangleq \begin{pmatrix} s_0 & 0 & \cdots & 0 \\ s_1 & s_0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ s_{N-1} & s_{N-2} & \cdots & s_0 \end{pmatrix}, \tag{7.42d}$$

where s_i denote the coefficients of $S_k(q^{-1})$, we have that all the N future output predictions can be collected in vector form according to

$$\hat{\mathbf{y}} = \bar{\mathbf{y}} + \tilde{\mathbf{S}}\tilde{\mathbf{u}}. \tag{7.43}$$

However, taking into account that the control horizon N_u typically is less than the prediction horizon N and that (7.41) holds, this can be rewritten as

$$\hat{\mathbf{y}} = \bar{\mathbf{y}} + \mathbf{S}\mathbf{u}, \tag{7.44}$$

where

$$\mathbf{u} \triangleq \begin{pmatrix} u(t) & \dots & u(t+N_u-1) \end{pmatrix}^T, \tag{7.45}$$

and

$$\mathbf{S} \triangleq \tilde{\mathbf{S}}\mathbf{\Lambda},$$

with

$$\mathbf{\Lambda} \triangleq \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix}^T.$$

The control increments (7.38) can also be expressed in vector form

$$\mathbf{\Delta u} = \mathbf{D}\mathbf{u} - \bar{\mathbf{u}} \tag{7.46}$$

by introducing the auxiliary quantities

$$\mathbf{D} \triangleq \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & -1 & 1 \end{pmatrix} \quad \text{and} \quad \bar{\mathbf{u}} \triangleq \begin{pmatrix} u(t-1) \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The objective (7.40) can thus be re-expressed in compact vector form according to

$$J = J(\mathbf{u}) = \left\|\mathbf{r} - \hat{\mathbf{y}}\right\|^2_{\mathbf{Q_e}} + \|\mathbf{u}\|^2_{\mathbf{Q_u}} + \|\mathbf{\Delta u}\|^2_{\mathbf{Q_{\Delta u}}}$$

$$= \|\mathbf{r} - \bar{\mathbf{y}} - \mathbf{S}\mathbf{u}\|^2_{\mathbf{Q_e}} + \|\mathbf{u}\|^2_{\mathbf{Q_u}} + \|\mathbf{D}\mathbf{u} - \bar{\mathbf{u}}\|^2_{\mathbf{Q_{\Delta u}}}, \tag{7.47}$$

where

$$\mathbf{r} \triangleq \begin{pmatrix} r(t+1) & \dots & r(t+N) \end{pmatrix}^T$$

denotes the desired (and possibly smoothed) reference trajectory, and $\mathbf{Q_e}$, $\mathbf{Q_u}$ and $\mathbf{Q_{\Delta u}}$ are diagonal matrices with entries $Q_e(k)$, $Q_u(k)$ and $Q_{\Delta u}(k)$ respectively.

Now, for the *unconstrained* case, i.e., when no hard bounds on the control and increment are present, the minimizing control sequence $\mathbf{u}$ is obtained explicitly as the ordinary least squares solution;

$$\boxed{\mathbf{u} = \left(\mathbf{S}^T\mathbf{Q_e}\mathbf{S} + \mathbf{Q_u} + \mathbf{D}^T\mathbf{Q_{\Delta u}}\mathbf{D}\right)^{-1}\left(\mathbf{S}^T\mathbf{Q_e}(\mathbf{r} - \bar{\mathbf{y}}) + \mathbf{D}^T\mathbf{Q_{\Delta u}}\bar{\mathbf{u}}\right)} \tag{7.48}$$

For the *constrained* case, the constraints (7.39) can be collected into the vector inequality

$$\mathbf{C}\mathbf{u} \le \mathbf{c}, \tag{7.49}$$

where

$$\mathbf{C} \triangleq \begin{pmatrix} I \\ -I \\ \mathbf{D} \\ -\mathbf{D} \end{pmatrix} \qquad \text{and} \qquad \mathbf{c} \triangleq \begin{pmatrix} u_{\max} \cdot \mathbf{1} \\ -u_{\min} \cdot \mathbf{1} \\ \Delta u_{\max} \cdot \mathbf{1} + \bar{\mathbf{u}} \\ -\Delta u_{\min} \cdot \mathbf{1} - \bar{\mathbf{u}} \end{pmatrix}.$$

We thus have the quadratic programming (QP) problem;

$$\boxed{\begin{array}{l} \min\limits_{\mathbf{u}} \ \mathbf{u}^T\left(\mathbf{S}^T\mathbf{Q_e}\mathbf{S} + \mathbf{Q_u} + \mathbf{D}^T\mathbf{Q_{\Delta u}}\mathbf{D}\right)\mathbf{u} - 2\left((\mathbf{r} - \bar{\mathbf{y}})^T\mathbf{Q_e}\mathbf{S} + \bar{\mathbf{u}}^T\mathbf{Q_{\Delta u}}\mathbf{D}\right)\mathbf{u} \\ \text{subject to} \quad \mathbf{C}\mathbf{u} \le \mathbf{c} \end{array}} \tag{7.50}$$

which can be efficiently solved using standard numerical optimization software.

Example 7.2

Consider again the nonlinear system introduced in Example 7.1, but assume that the magnitude of the control signal now is limited according to

$$|u(t+k)| \le 15, \qquad k = 0, \dots, N_u - 1.$$

We decided to try a receding horizon MPC controller of the form (7.50) for control, where the system dynamics at each sampling instant was estimated using MOD-algorithm 4.1 with default values.

The simulation result obtained after using the same estimation database as in Example 7.1 (i.e., 3000 samples, NARX 221 structure), and controller parameter $N = 10$, $N_u = 7$, $Q_e = 1$, $Q_u = 0$ and $Q_{\Delta u} = 0.001$ is shown in Figure 7.5. We see that the system output follows the reference reasonable well apart from a small overshoot. ❏

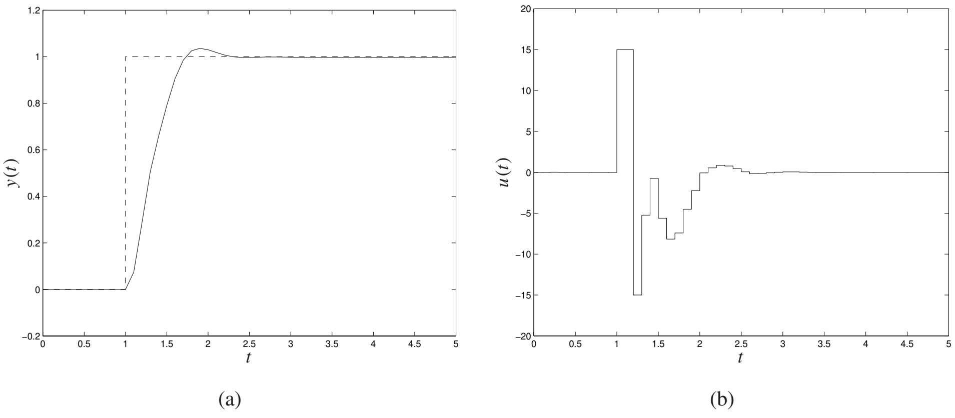


Figure 7.5 *Step response experiment using a constrained predictive controller based on fixed linearizations. (a) Reference signal (dashed) and system output (solid). (b) Control signal.*

7.4.3 Optimization Based on Linearization along a Trajectory

A drawback with the method described in the preceeding subsection, is that the predicted behavior of the process is determined on basis of the input-output linearization delivered by the estimator at time t. In this way we have implicitly assumed that the linearization is a valid model along the trajectory we predict.

This assumption will of course not hold in general, since when the currently computed control action is applied, the process state will most likely move to another point in the regressor space which might represent completely different dynamical properties.

A straightforward solution to this problem is to make use of an N-step-ahead predictor of model-on-demand type, and provide it with an input sequence $\tilde{\mathbf{u}}$ in order to obtain an approximate time-varying local linear model over the future N samples. Then the expressions derived in the end of Section 7.3.1 can be utilized when optimizing the objective function.

An obvious question is of course which input sequence that should be used when determining the time-varying model. A straightforward approach here, though, is to take advantage of the control sequence obtained from the optimization performed at the previous sampling instant. Another alternative is to use the result from Section 7.4.2 as an initial value.

The derivation of the optimal control sequence for the time-varying model is completely analogous to the time-invariant case in Section 7.4.2 and will therefore be omitted here. Note though, that the matrix $\tilde{\mathbf{S}}$ in this case will not have as simple (and recursively defined) structure as before.

From (7.27) it instead follows that

$$\tilde{\mathbf{S}} = \begin{pmatrix} s_{0,t,1} & 0 & \cdots & 0 \\ s_{1,t,2} & s_{0,t,2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ s_{N-1,t,N} & s_{N-2,t,N} & \cdots & s_{0,t,N} \end{pmatrix},$$

where $s_{i,t,k}$ denote the coefficients of $S_{t,k}(q^{-1})$. The vector $\bar{\mathbf{y}}$ is changed accordingly. The control sequence can then be computed using expressions similar to (7.48) or (7.50) depending on whether or not hard constraints are present in the criterion.

Example 7.3

Consider again the system introduced in Example 7.1 which was controlled using a constrained MPC controller based on local linearizations in Example 7.2. The simulation result for an MPC controller based on linearizations along a future trajectory is shown in Figure 7.6. The controller parameters was selected as in the previous example.

We see that this controller performs much better than the previous one. Since the latter controller uses linear models of the plant along the predicted future trajectory, it is aware of that the gain of the system will decrease as the output magnitude increases. Therefore it is more restrictive in its use of control energy as the output approaches the setpoint. ❏

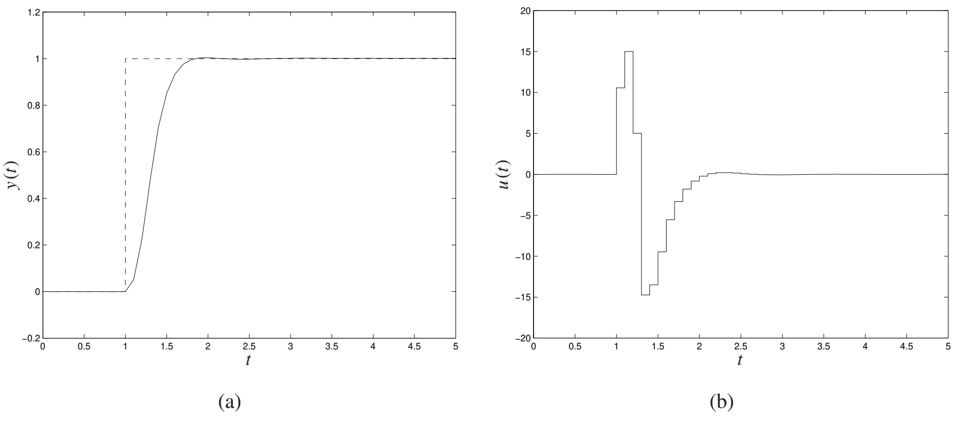


Figure 7.6 *Step response experiment using a constrained predictive controller based on linearizations along a trajectory. (a) Reference signal (dashed) and system output (solid). (b) Control signal.*

7.4.4 General Nonlinear Optimization

The most general solution to the nonlinear MPC problem is perhaps *brute force* optimization of the criterion (7.40) using nonlinear programming methods. The optimal control sequence $\mathbf{u}$ can thus be determined using a numerical optimization routine, that in each iteration simulates the system given its current value, and then updates it in a direction such that the value of the objective function J decreases.

As in the constant future control case, it is known that more accurate results normally will be obtained if the optimization routine is provided with gradient information. But fortunately, here the derivations in earlier sections come to our aid. From (7.47) it follows that

$$\nabla J = 2\mathbf{S}^T \mathbf{Q_e}(\hat{\mathbf{y}} - \mathbf{r}) + 2\mathbf{Q_u}\mathbf{u} + 2\mathbf{D}^T \mathbf{Q}_{\Delta\mathbf{u}}\Delta\mathbf{u}. \tag{7.51}$$

The gradient can thus be estimated from data using the sequence of local models obtained along the simulated trajectory.

A severe problem with the general optimization formulation, though, is that the optimization typically will be very complex and time-consuming for large control horizons N_u. We will most likely also get problems with local minima.

Example 7.4

Consider again the nonlinear system considered in the previous examples. A simulation using an optimization-based MPC controller with the same parameters as in the previous examples is shown in Figure 7.7. We see that this controller, contrary to what is expected, performs worse than the previous ones. This is probably due to the problem that the optimization routine gets stuck in local minima. ❏

7.4.5 Choice of Horizons

Since the controller part of the algorithms presented in this section essentially coincides with the traditional MPC/GPC approaches, the same tuning guidelines apply. The fact that we now are dealing with nonlinear systems may in some situations require special attention since model errors in the linearizations may cause instability. However, we shall below summarize the standard choices for linear systems found by Clarke *et al.* (1987*a*) and Meadows and Rawlings (1997) after excessive simulation studies.

The output horizon N

If the system is of non-minimum-phase type, the output horizon should be selected so that the later positive-going outputs are included in the objective. For linear discrete-time systems this typically means that N should exceed the degree of the $B(q^{-1})$ polynomial. In normal cases, though, it can be taken significantly larger, corresponding to the rise-time of the system. This is a quite natural assumption.

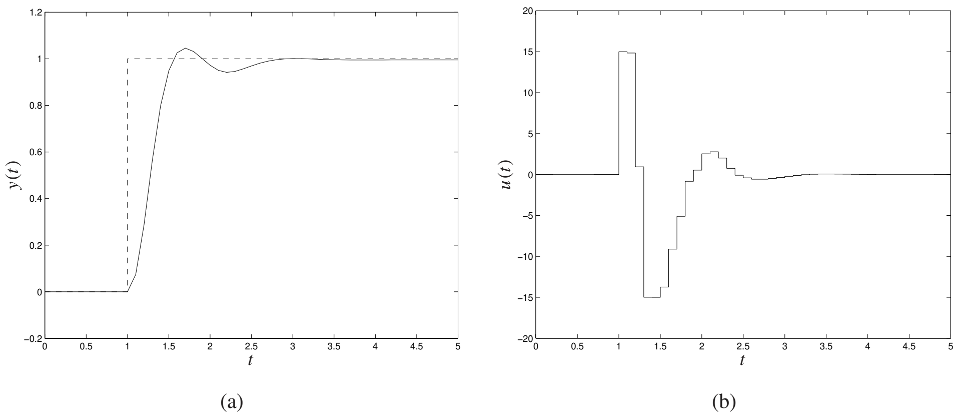


Figure 7.7 *Step response experiment using a constrained predictive controller based on nonlinear optimization. (a) Reference signal (dashed) and system output (solid). (b) Control signal.*

The control horizon N_u

The control horizon has found to be a very important design parameter, and is typically selected much shorter than the output horizon. For simple systems a value of $N_u = 1$ typically gives quite acceptable result. Increasing it normally results in a faster system response, but at the same time also larger and more aggressive controls.

One additional reason for choosing a short control horizon is that it is this parameter that essentially determines the size of the optimization problem. In the model-on-demand context, though, this argument might be of minor relevance since the algorithms anyway are quite demanding in terms of computational resources.

7.4.6 Some Additional Examples

To establish the usefulness of the proposed concept of model-free predictive control we will now apply it to some additional and more complex applications.

Example 7.5 **Regulation of the Narendra-Li system**

Recall the nonlinear system (5.23) which was used for simulation purposes in Chapter 5. We will now try to regulate this system using the model-free predictive control approach.

As in Narendra and Li (1996), we use the reference signal

$$r(t) = y_{\text{ref}}(t+1) = 0.75 \sin\left(\frac{2\pi(t+1)}{50}\right) + 0.75 \sin\left(\frac{2\pi(t+1)}{25}\right)$$

to check the tracking capabilities of the controller. We choose to use the same estimation database and the same setup as for the model-on-demand simulation in Section 5.7.2, i.e., 50 000 samples and a NARX 331 structure. Since the system is distinctly nonlinear, it is crucial to select the controller parameters with care in order to obtain a stable closed-loop system. We found that it was almost impossible to achieve an acceptable result with the linearizing controller of Section 7.4.2. Therefore the more complex and resource-demanding controller of Section 7.4.3 was used instead.

It turned out that a good choice of controller parameters was the values $N = 3$, $N_u = 2$, $Q_e = 1$, $Q_u = 0.1$, and $Q_{\Delta u} = 0.05$. The relative large penalty on the increment was needed to avoid excessive jumps in the control. Due to the way the estimation database was generated, the control magnitude was limited according to

$$|u(t+k)| \leq 2.5, \qquad k = 0, \dots, N_u - 1.$$

A simulation using the specified reference signal and the controller parameters above results in the behavior displayed in Figure 7.8. Note that the constraints on the the control magnitude never are active in this case. The performance is equivalent to the result obtained by Narendra and Li (1996) after using a complex neural-based controller.

A simulation using a square wave reference is shown in Figure 7.9. In this case it was possible to shorten the control horizon to $N_u = 1$. This resulted in a less oscillative control signal. Notice, though, the steady-state control error that occurs in this case. This is probably a consequence of the relatively short prediction horizon. Increasing this quantity, however, resulted in an unstable closed-loop system. ❏

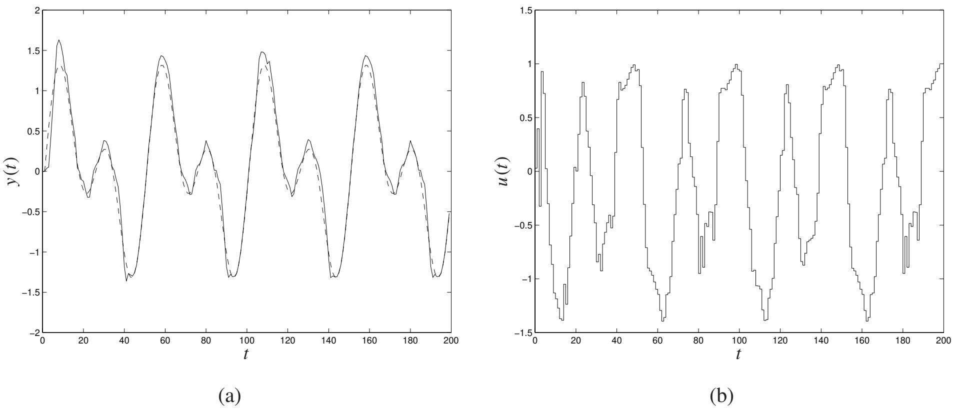


Figure 7.8 Model-free predictive control of the Narendra-Li system. (a) Reference (dashed) and system output (solid). (b) Control signal.

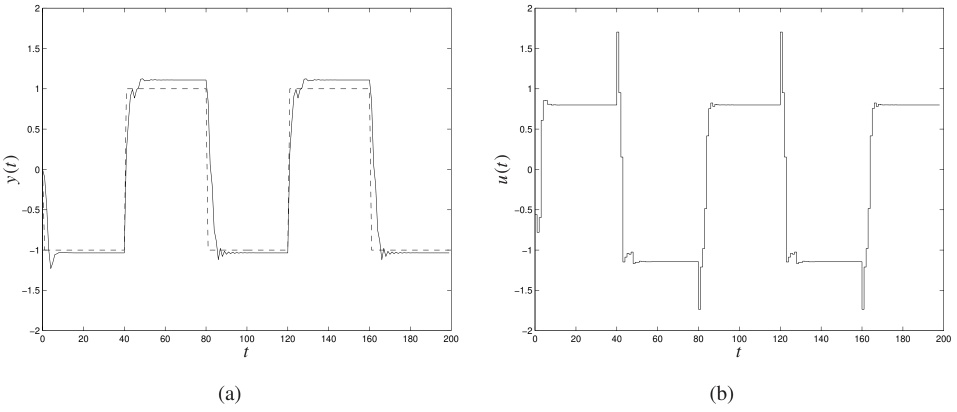


Figure 7.9 *Model-free predictive control of the Narendra-Li system. (a) Reference (dashed) and system output (solid). (b) Control signal.*

Example 7.6 Continuous stirred-tank reactor

As a last example we will consider control of a continuous stirred-tank reactor. It has earlier been studied by (among others) Meadows and Rawlings (1997). The reactor model is given by

$$\dot{x}_1(t) = -k_1 x_1(t) - k_3 x_1^2(t) + (x_F - x_1(t))u(t) \tag{7.52a}$$

$$\dot{x}_2(t) = k_1 x_1(t) - k_2 x_2(t) - x_2(t)u(t) \tag{7.52b}$$

and models certain chemical reactions in the reactor. The parameters have nominal values $k_1 = 50$, $k_2 = 100$, $k_3 = 10$ and $x_F = 10$. The control objective is to maintain the variable $x_2(t) = y(t)$ at the setpoint 1.0

The two steady-state solutions $(\bar{x}_1,\ \bar{x}_2,\ \bar{u})$ that corresponds to the desired output are

$$(2.5,\ 1.0,\ 25) \qquad \text{and} \qquad (6.6667,\ 1.0,\ 233.33).$$

Due to some special performance properties of the process, the solution with lower values is to prefer.

We assume that the states are not measurable, so we have to rely entirely on an input-output structure for modeling and control. In order to generate data, the system (7.52) was simulated using a Gaussian-type input with mean 25 and standard deviation $\sigma = 50$. A dataset of 5000 samples was collected from this simulation by sampling with $T_s = 0.002$ h. The resulting data were used to construct a MOD database of NARX 221 type.

As noted by Meadows and Rawlings (1997), an MPC controller without penalties on the control drives the system to the higher (undesirable) steady-state solution. The same is observed for controllers with short horizons and is probably a consequence of the system's non-minimum-phase properties. This behavior is even more critical in the MOD case, since the system state then moves out of the support of the stored data. We found that a good choice of parameters for the controller that brings the system to the desired state was $N = N_u = 7$, $Q_e = 1$, $Q_u = 0$, $Q_{\Delta u} = 2 \cdot 10^{-4}$, $u_{\min} = 0$ and $u_{\max} = 50$. A simulation based these values, MOD-algorithm 4.1 and initial conditions $x_1(0) = 2.9997$ and $x_2(0) = 1.11697$ is shown in Figure 7.10. The result is equivalent to that reported by Meadows and Rawlings (1997). ❑

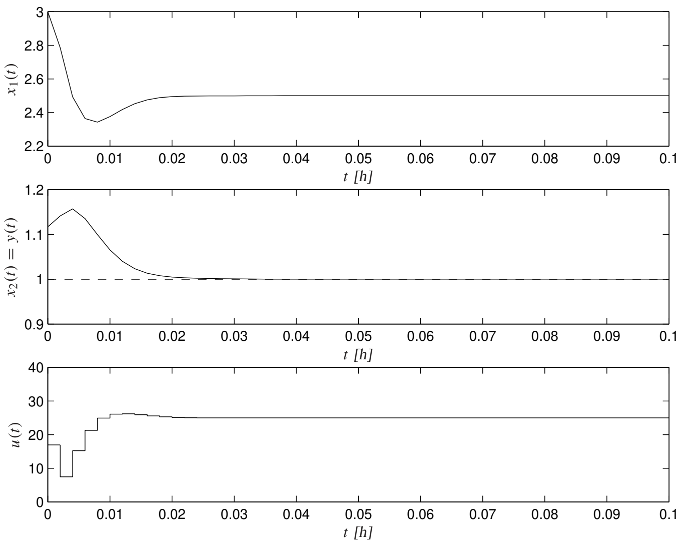


Figure 7.10 *Simulation result for the stirred-tank reactor.*

7.4.7 Summary

We have in this section presented the promising concept of *model-free predictive control*, that combines the idea of model-on-demand with established and well-known MPC/GPC techniques. The method is model-free in the sense that no global model of the process dynamics is required. Instead it relies on an on-line estimation scheme that depends upon process data stored in a database. The only model-related quantities that have to be decided upon by the user, are the parameters that control the configuration of the regression vector, i.e., the orders and delays. The remaining part of the modeling is performed automatically and on-line during operation.

By assuming that MOD estimator delivers local linear models, an advantage with the approach is that it is possible to use standard MPC techniques for the controller part. This also implies that standard tuning guidelines apply. Moreover, since the estimator at each sampling instant returns the best available linearization given data, the method has the the ability of controlling mildly nonlinear and time-varying systems. This has been demonstrated in several numerical simulations.

A drawback with the approach is that the performance of the controller is critically depending on the quality of the database. Furthermore, the concept will probably not work with strongly nonlinear systems where the local model is valid only in a very narrow neighborhood of the operating point. The controller also requires large computational resources due to the nature of the underlying estimation procedure.

7.5 Conclusions and Extensions

This chapter has discussed several possibilities of using the model-on-demand philosophy in a control framework, ranging from simple adaptive control ideas to more sophisticated predictive approaches. The most promising approach here seems to be the model-free predictive control (or MoDPC) approach of Section 7.4, although it is clear that its properties need to be further studied and analyzed.

All the described methods have relied upon the certainty equivalence principle, i.e., that the obtained local model exactly describes the true system dynamics. This is of course never true in reality. However, as demonstrated in Chapter 4, it is possible to derive statistical descriptions of the model uncertainty such as the variance of the estimate or confidence bands. To utilize such uncertainty descriptions from the local models in the control design is an obvious and natural extension of the concept.

8

Summary & Conclusions

The problem of modeling dynamical systems when large data volumes are available has been studied. The proposed solution, the model-on-demand estimator, stores all observations in a database, and computes estimates “on demand” as the need arises. When a model is really needed at (or around) a certain operating point, relevant data is retrieved from the database, and a modeling operation is performed on that subset.

It has been recommended that the model-on-demand estimator is formed as a weighted average of the observations belonging to small neighborhood around the operating point. Two conceptually different approaches for weight selection have been studied, where the first one is based on traditional kernel functions and the other one relies on an explicit optimization stage. Furthermore, two algorithms corresponding to these approaches have been presented and their asymptotic properties have been studied. It has been demonstrated through simulations that the optimization approach for some applications may produce more accurate predictions. However, it is at the same time more demanding in terms of computational resources.

Identification Applications

Chapters 5 and 6 have shown that the model-on-demand concept with success can be applied to system identification problems, both in the time and frequency domains.

For time-domain prediction/simulation, it has been demonstrated that the derived methods are very flexible, and that they for some applications produce smaller prediction errors than other proposed methods like neural nets. However, this performance improvement comes to the price of an increased computational complexity. Other potential drawbacks are that the model-on-demand method is critically depending on the number of data, and that it might be sensitive to the data distribution and effects at the boundary of the regressor space.

For the frequency domain problems, it has been shown that the nonparametric modeling concept provides methods that estimate spectra and frequency functions from data using automatic, adaptive and frequency-dependent frequency resolution. This gives several advantages over traditional spectral analysis techniques. Many frequency functions exhibit fine details to different degrees in different frequency bands. The local smoothing approach thus gives a useful alternative to multi-resolution techniques like wavelets. We have also demonstrated how an automated procedure, based on a local goodness-of-fit criterion, leads to good choice of bandwidths that very well matches the asymptotically optimal choice of frequency resolution.

Control Applications

In Chapter 7 we have outlined several possibilities of using the model-on-demand philosophy in a control framework, ranging from simple adaptive control ideas to more sophisticated predictive approaches. In particular we have proposed the concept of *model-free predictive control*, which combines the idea of model-on-demand with established and well-known MPC/GPC techniques. This approach holds the promise of high-performance nonlinear control without the need for global models.

A potential drawback with the method is that the performance of the controller is critically depending on the quality of the database. Furthermore, the concept will probably not work with strongly nonlinear systems where the local model is valid only in a very narrow neighborhood of the operating point. The controller also requires large computational resources due to the nature of the underlying estimation procedure. This can of course be a problem for high-performance applications with timing constraints.

Extensions & Future Work

There is a number of open questions that need to be further investigated. Some possible topics for future research are listed below.

- For the estimation part, a natural extension is to consider more complex statistical models and noise distributions. This will lead to research in the direction of local likelihood estimation.
- Another obvious extension in the general setting is the dataset searching problem. This will concern investigations regarding suitable data structures that will enable efficient searches for neighborhoods. Tree structures like *k-d* trees were discussed in Chapter 4 and could be a useful alternative here. When working with huge datasets, though, it would be more desirable to utilize a database management system (DBMS). How such a system could be integrated with the estimation procedure clearly needs to be further studied.
- A related question occurs when dealing with the time-domain prediction problem: How should the collected input-output data be organized in memory to enable largest possible flexibility? We have until now assumed that the data are stored as output-regressor pairs. However, this restricts the possibility of changing model structure

after the database is constructed. Applications involving time-varying systems where the database has to be updated on-line also raise the demand for forgetting mechanisms or data compression methods.

- For the frequency-domain applications, we have shown that it is important to estimate the variance from data in order to obtain good estimates. How the final result is affected by this initial smoothing stage needs to be investigated further. The same holds for the recursive procedure that was used to speed up the estimation process. Its sensitivity against certain classes of frequency functions needs to be further analyzed through simulations.

- The control applications of course also need to be more thoroughly studied and analyzed. All described control methods have relied upon the certainty equivalence principle, i.e., that the obtained models exactly reflect the true system dynamics. This is of course rarely true in reality. However, we have shown that it is possible to derive statistical descriptions of the model uncertainty such as variance of the estimate or confidence bands. To utilize such uncertainty descriptions from the local models in the control design is an obvious and natural extension.

Bibliography

Aha, D.W. (1989). Incremental, instance-based learning of independent and graded concept descriptions. In: *Proceedings of the Sixth International Machine Learning Workshop*. Morgan Kaufmann Publishers. pp. 387–391.

Aho, A.V., J.E. Hopcroft and J.D. Ullman (1983). *Data Structures and Algorithms*. Addison-Wesley.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In: *Proceedings of the 2nd International Symposium on Information Theory* (B.N. Petrov and F. Csàki, Eds.). pp. 267–281.

Andersson, T. and P. Pucar (1994). Estimation of residence time in continuous flow systems with dynamics. Technical Report LITH-ISY-R-1605. Dept of EE. Linköping University, S-581 83, Linköping, Sweden.

Åström, K.J. and B. Wittenmark (1990). *Computer Controlled Systems – Theory and Design*. second ed.. Prentice Hall.

Åström, K.J. and B. Wittenmark (1995). *Adaptive Control*. second ed.. Addison-Wesley.

Atkeson, C.G. and D.J. Reinkensmeyer (1988). Using associative content-addressable memories to control robots. In: *Proceedings of the 27th IEEE Conference on Decision and Control, Austin, Texas*. Vol. 1. pp. 792–797.

Atkeson, C.G., A.W. Moore and S. Schaal (1997*a*). Locally weighted learning. *Artificial Intelligence Review* **11**(1-5), 11–73.

Atkeson, C.G., A.W. Moore and S. Schaal (1997*b*). Locally weighted learning for control. *Artificial Intelligence Review* **11**(1-5), 75–113.

Bendat, J.S. and A.G. Piersol (1980). *Engineering Applications of Correlation and Spectral Analysis*. Wiley, New York.

Bentley, J.L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM* **18**(9), 509–517.

Bitmead, R.R., M. Gevers and V. Wertz (1990). *Adaptive Optimal Control: The Thinking Man's GPC*. Prentice Hall.

Björck, Å. (1996). *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia.

Blackman, R.B. and J.W. Tukey (1958). *The Measurement of Power Spectra*. Dover, New York.

Bodin, P. (1995). On wavelets and orthonormal bases in system identification. Licentiate thesis TRITA-REG-9502. Automatic Control, Dept. of Signals, Sensors and Systems, Royal Institute of Technology, Sweden.

Bontempi, G., M. Birattari and H. Bersini (1998). Lazy learning for local modeling and control. To Appear.

Bottou, L. and V.N. Vapnik (1992). Local learning algorithms. *Neural Computation* **8**(3), 437–441.

Braun, M.W., D.E. Rivera, A. Stenman, W. Foslien and C. Hrenya (1999). Multi-level pseudo-random signal design and just-in-time estimation applied to nonlinear identification of a RTP wafer reactor. To be presented at the 1999 American Control Conference, San Diego, California.

Brillinger, D.R. (1981). *Time Series: Data Analysis and Theory*. Holden-Day, San Fransisco.

Brockwell, P.J. and R.A. Davis (1987). *Time Series: Theory and Methods*. Springer.

Casdagli, M.C. (1991). Chaos and deterministic versus stochastic nonlinear modeling. *Journal of the Royal Stat. Soc. B* **54**, 301–328.

Casdagli, M.C. and A.S. Weigend (1994). Exploring the continuum between deterministic and stochastic modeling. In: *Time series prediction: Forecasting the future and understanding the past* (A.S. Weigend and N.A. Gershenfeld, Eds.). pp. 347–366. Addison-Wesley.

Char, B.W., K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan and S.M. Watt (1991). *Maple V – Language Reference Manual*. Springer-Verlag.

Chen, S., S.A. Billings, C.F.N. Cowan and P.M. Grant (1990). Non-linear system identification using radial basis functions. *International Journal of Systems Science* **21**(12), 2513–2539.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987*a*). Generalized predictive control – I. The basic algorithm. *Automatica* **23**, 137–148.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987*b*). Generalized predictive control – II. Extensions and interpretations. *Automatica* **23**, 149–160.

Cleveland, W.S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* **74**, 829–836.

Cleveland, W.S. and C. Loader (1994). Smoothing by local regression: Principles and methods. Technical report. AT&T Bell Laboratories. 600 Mountain Avenue, Murray Hill, NJ 07974, USA.

Cleveland, W.S. and E. Grosse (1991). Computational methods for local regression. *Statistics and Computing* **1**, 47–62.

Cleveland, W.S. and S.J. Devlin (1988). Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association* **83**, 596–610.

Cleveland, W.S., E. Grosse and M.-J. Shyu (1992). *A Package of C and Fortran Routines for Fitting Local Regression Models*. AT&T Bell Laboratories.

Cleveland, W.S., T. Hastie and C. Loader (1995). Adaptive local regression by automatic selection of the polynomial mixing degree. In preparation.

Cover, T.M. and P.E. Hart (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **13**, 21–27.

Craven, P. and G. Wahba (1979). Smoothing noisy data with spline functions. *Numerische Mathematik* **31**, 377–403.

Cybenko, G. (1996). Just-in-time learning and estimation. In: *Identification, Adaption, Learning* (S. Bittani and G. Picci, Eds.). NATO ASI series. Springer. pp. 423–434.

Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory* **36**(4), 961–1005.

Davenport, Jr., W.B. (1975). *Probability and Random Processes*. McGraw-Hill.

Davis, H.T. and R.H. Jones (1968). Estimation of the innovation variance of a stationary time series. *Journal of the American Statistical Association* **63**, 141–149.

Dennis, Jr., J.E. and R.B. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey.

Donoho, D.L. and I.M. Johnstone (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika* **81**, 425–455.

Draper, N.R. and H. Smith (1981). *Applied Regression Analysis*. 2nd ed.. John Wiley, New York.

Draper, N.R. and R.V. Nostrand (1979). Ridge regression and James-Stein estimation: Review and comments. *Technometrics* **21**(4), 251–466.

Efron, B. and R.J. Tibshirani (1993). *An Introduction to the Bootstrap*. Chapman & Hall.

Elmasri, R. and S.B. Navathe (1994). *Fundamentals of Database Systems.*. Addison-Wesley World Student Series. 2nd ed.. Benjamin/Cummings Publishing Company, Inc.

Epanechnikov, V.A. (1969). Non-parametric estimation of a multivariate probability density. *Theory of Probability and Its Applications* **14**, 153–158.

Fan, J. (1992). Design-adaptive nonparametric regression. *Journal of the American Statistical Association* **87**, 998–1004.

Fan, J. and E. Kreutzberger (1998). Automatic local smoothing for spectral density estimation. *Scandinavian Journal of Statistics* **25**, 359–369.

Fan, J. and I. Gijbels (1996). *Local Polynomial Modelling and Its Applications*. Chapman & Hall.

Fan, J., P. Hall, M.A. Martin and P. Patil (1996). On local smoothing of nonparametric curve estimators. *Journal of the American Statistical Association* **91**(433), 258–266.

Farmer, J.D. and J.J. Sidorowich (1987). Predicting chaotic time series. *Physical Review Letters* **58**(8), 845–848.

Farmer, J.D. and J.J. Sidorowich (1988). Exploiting chaos to predict the future and reduce noise. In: *Evolution, learning, and cognition* (Y.C. Lee, Ed.). World Scientific, Singapore.

Friedman, J.H. and W. Stuetzle (1981). Projection pursuit regression. *Journal of the American Statistical Association* **76**, 817–823.

Friedman, J.H., J.L. Bentley and R.A. Finkel (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software* **3**, 209–226.

García, C.G., D.M. Prett and M. Morari (1989). Model predictive control: Theory and practice – a survey. *Automatica* **25**(3), 335–348.

Garcia-Molina, H. and K. Salem (1992). Main memory database systems: An overview. *IEEE Transactions on Knowledge and Data Engineering* **4**(6), 509–516.

Gardner, W.A. (1988). *Statistical Spectral Analysis – A Nonprobabilistic Theory*. Prentice Hall, Englewood Cliffs, New Jersey.

Gasser, T. and H.-G. Müller (1979). Kernel estimation of regression functions. In: *Smoothing Techniques for Curve Estimation* (T. Gasser and M. Rosenblatt, Eds.). pp. 23–68. Springer-Verlag, Heidelberg.

Gattu, G. and E. Zafiriou (1992). Nonlinear quadratic dynamic matrix control with state estimation. *Ind. Eng. Chem. Res.* **31**(4), 1096–1104.

Gunnarsson, S. and P. Krus (1990). Modeling of a flexible mechanical system containing hydralic actuators. Technical report. Dept of EE. Linköping University, S-581 83, Linköping, Sweden.

Gustafsson, F. (1992). Estimation of Discrete Parameters in Linear Systems. PhD thesis. Dept of EE, Linköping University. S-581 83 Linköping, Sweden.

Härdle, W. (1990). *Applied Nonparametric Regression*. Cambridge University Press.

Hastie, T. and R. Tibshirani (1990). *Generalized Additive Models*. Chapman & Hall.

Hastie, T.J. and C. Loader (1993). Local regression: Automatic kernel carpentry (with discussion). *Statist. Sci.* **8**, 120–143.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan.

Henderson, H.V. and S.R. Searle (1979). Vec and vech operators for matrices, with some uses in Jacobians and multivariate statistics. *Canadian Journal of Statistics* **7**, 65–81.

Hurwich, C.M., J.S. Simonoff and C.L. Tsai (1998). Smoothing parameter selection in nonparametric regression using an improved akaike information criterion. *Journal of the Royal Statistical Society* **60**, 271–293.

Jenkins, G.M. and D.G. Watts (1968). *Spectral Analysis and its Applications*. Holden-Day, San Fransisco, California.

Johansen, T.A. and B.A. Foss (1995). Identification of non-linear system structure and parameters using regime decomposition. *Automatica* **31**(2), 321–326.

Kailath, T. (1980). *Linear Systems*. Prentice-Hall, Englewood Cliffs, New Jersey.

Katkovnik, V.Y. (1979). Linear and nonlinear methods of nonparametric regression analysis. *Soviet Automatic Control* **5**, 25–34.

Koivisto, H. (1995). A Practical Approach to Model Based Neural Network Control. PhD thesis. Tampere University of Technology, Tampere, Finland.

Kooperberg, C., C.J. Stone and Y.K. Truong (1995). Logspline estimation of a possible mixed spectral distribution. *J. Time Ser. Anal.* **16**, 359–389.

Kullback, S. and R.A. Leibler (1951). On information and sufficiency. *Ann. Math. Statis.* **22**, 79–86.

Lindskog, P. (1996). Methods, Algorithms and Tools for System Identification Based on Prior Knowledge. PhD thesis. Linköping University, Linköping, Sweden.

Ljung, L. (1999). *System Identification – Theory for the user*. Information and System Sciences Series. 2nd ed.. Prentice-Hall, New Jersey.

Ljung, L. and B. Wahlberg (1992). Asymptotic properties of the least-squares method for estimating transfer functions and disturbance spectra. *Advances in Applied Probability* **24**, 412–440.

Loader, C.R. (1995). Old faithful erupts: Bandwidth selection reviewed. Technical report. AT&T Bell Laboratories.

Loader, C.R. (1997). *Locfit: An Introduction*. AT&T Bell Laboratories.

Loftsgaarden, D.O. and G.P. Quesenberry (1965). A nonparametric estimate of a multivariate density function. *Annals of Mathematical Statistics* **36**, 1049–1051.

Mallows, C.L. (1973). Some comments on C_p. *Technometrics* **15**, 661–675.

MathWorks (1996). *MATLAB – Language Reference Manual, Version 5*. The MathWorks Inc.

Meadows, E.S. and J.B. Rawlings (1997). Model predictive control. In: *Nonlinear Process Control* (M.A. Henson and D.E. Seborg, Eds.). Chap. 5. Prentice Hall.

Moore, A.W., J. Schneider and K. Deng (1997). Efficient locally weighted polynomial regression predictions. In: *Proceedings of the 1997 International Machine Learning Conference* (D. Fisher, Ed.). Morgan Kaufmann Publishers.

Müller, H.-G. (1987). Weighted local regression and kernel methods for nonparametric curve fitting. *Journal of the American Statistical Association* **82**, 231–238.

Nadaraya, E. (1964). On estimating regression. *Theory of Probability and Its Applications* **10**, 186–190.

Narendra, K.S. and A.M. Annaswamy (1989). *Stable Adaptive Systems*. Prentice Hall.

Narendra, K.S. and S.M. Li (1996). Neural networks in control systems. In: *Mathematical Perspectives on Neural Networks* (P. Smolensky, M.C. Mozer and D.E. Rumelhart, Eds.). Chap. 11, pp. 347–394. Lawrence Erlbaum Associates.

Passino, K.M. and S. Yurkovich (1996). Fuzzy control. In: *The Control Handbook* (W.S. Levine, Ed.). IEEE Press.

Priestly, M.B. and M.T. Chao (1972). Non-parametric function fitting. *Journal of the Royal Statistical Society* **B 34**, 385–392.

Rivera, D. E., S. Adusumilli, A. Stenman, F. Gustafsson, T. McKelvey and L. Ljung. (1997). Just-in-time models for improved identification and control. In: *Proc. American Inst. of Chemical Engineers (AIChE) Annual Meeting*.

Ruppert, D. and M.P. Wand (1994). Multivariate locally weighted least squares regression. *Annals of Statistics*.

Sadjadi, F.A. (1993). An approximate k-nearest neighbor method. In: *Adaptive and Learning Systems II*. Proc. SPIE 1962.

Samet, H. (1990). *The Design and Analysis of Spatial Data Structures*. Computer Science Series. Addison-Wesley.

Schoukens, J. and R. Pintelon (1991). *Identification of Linear Systems – A Practical Guideline to Accurate Modeling*. Pergamon Press.

Sjöberg, J. (1995). Non-Linear System Identification with Neural Networks. PhD thesis. Dept. of EE, Linköping University. S-581 83 Linköping, Sweden.

Sjöberg, J., Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P.-Y. Glorennec, H. Hjalmarsson and A. Juditsky (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica* **31**, 1691–1724.

Skeppstedt, A., L. Ljung and M. Millnert (1992). Construction of composite models from observed data. *International Journal of Control* **55**(1), 141–152.

Söderström, T. and P. Stoica (1989). *System Identification*. Prentice-Hall International, Hemel Hempstead, Hertfordshire.

Stanfill, C. and D. Waltz (1986). Toward memory-based reasoning. *Communications of the ACM* **29**(12), 1213–1228.

Stenman, A. (1997). *Just-in-Time Models with Applications to Dynamical Systems*. Licentiate thesis LIU-TEK-LIC-1997:02. Department of Electrical Engineering, Linköping University. S-581 83 Linköping, Sweden.

Stenman, A. (1999). Model-free predictive control. Submitted to CDC '99.

Stenman, A. and F. Gustafsson (1999). Adaptive smoothing methods for frequency-domain identification. Submitted to Automatica.

Stenman, A., A.V. Nazin and F. Gustafsson (1997). Asymptotic properties of Just-in-Time models. In: *Preprints of the 11th IFAC Symposium on System Identification, Kitakyushu, Japan* (Y. Sawaragi and S. Sagara, Eds.). pp. 1249–1254.

Stenman, A., F. Gustafsson and L. Ljung (1996). Just in time models for dynamical systems. In: *Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan.* pp. 1115–1120.

Stenman, A., F. Gustafsson, D.E. Rivera, L. Ljung and T. McKelvey (1999). On adaptive smoothing of empirical transfer function estimates. To be presented at the IFAC World Congress, July 1999, in Beijing, China.

Stoica, P. and R. Moses (1997). *Introduction to Spectral Analysis*. Prentice-Hall, New Jersey.

Stone, C.J. (1977). Consistent nonparametric regression. *The Annals of Statistics* **5**, 595–620.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society* **B**(36), 111–147.

Sun, J. and C.R. Loader (1994). Simultaneous confidence bands in linear regression and smoothing. *The Annals of Statistics* **22**, 1328–1345.

Tanaka, K. (1995). Stability and stabilizability of fuzzy-neural-linear control systems. *IEEE Transactions on Fuzzy Systems* **3**(4), 438–447.

Tibshirani, J.R. and T.J. Hastie (1987). Local likelihood estimation. *Journal of the American Statistical Association* **82**, 559–567.

Tukey, J.W. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

Wahba, G. (1980). Automatic smoothing of the log periodogram. *Journal of the American Statistical Association* **75**, 122–132.

Wand, M.P. and M.C. Jones (1995). *Kernel Smoothing*. Chapman & Hall.

Watson, G. (1964). Smooth regression analysis. *Sankhyā* **A**(26), 359–372.

Weigend, A.S. (1996). Time series analysis and prediction. In: *Mathematical Perspectives on Neural Networks* (P. Smolensky, M.C. Mozer and D.E. Rumelhart, Eds.). Chap. 12, pp. 395–449. Lawrence Erlbaum Associates.

Notation

Acronyms

AIC	Akaike's Information Theoretic Criterion
AMISE	Asymptotic Mean Integrated Squared Error
AMSE	Asymptotic Mean Squared Error
ARMA	Auto-Regressive Moving Average
ARMAX	Auto-Regressive Moving Average with eXogenous input
ARX	Auto-Regressive with eXogenous input
C_p	Mallow's C_p estimate of risk
CV	Cross-validation
ETFE	Empirical Transfer Function Estimate
FIR	Finite Impulse Response
FPE	Akaike's Final Prediction Error
GCV	Generalized Cross-validation
GPC	Generalized Predictive Control
LQ	Linear Quadratic
MPC	Model Predictive Control
MISE	Mean Integrated Squared Error
MISO	Multiple Input Single Output
MOD	Model-on-demand
MSE	Mean Squared Error
NARX	Nonlinear ARX
NFIR	Nonlinear FIR
OE	Output Error

PRBS	Pseudo-Random Binary Signal
RMSE	Root Mean Squared Error
SISO	Single Input Single Output

Symbols

$\mathbb{C}$	The set of complex numbers
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^d$	Euclidean d-dimensional space
I	The identity matrix
$\simeq, \sim$	$a_N \simeq b_N$, if and only if $\lim_{N\to\infty}(a_N/b_N) = 1$
o	$a_N = o(b_N)$, if and only if $\lim\ \sup_{N\to\infty} \lvert a_N/b_N \rvert = 0$
O	$a_N = O(b_N)$, if and only if $\lim\ \sup_{N\to\infty} \lvert a_N/b_N \rvert < \infty$
Ω_k	A neighborhood around the current operating point that contains k data
σ^2	Noise variance
$\mathbf{X}$	The design matrix
$\mathbf{H}$	The hat matrix
e_k	The kth column of the identity matrix
$\mathbf{1}$	The vector whose elements are equal to 1
$R_u(\tau)$	Autocovariance function
$R_{yu}(\tau)$	Cross-covariance function
$\Phi_u(\omega)$	Spectrum of the signal u
$\hat{\hat{\Phi}}_u(\omega)$	Periodogram of the signal u
$\hat{\hat{G}}(e^{i\omega})$	Empirical transfer function estimate
$\mathcal{B}(X)$	The vector of basis functions associated with a polynomial model
$\mathcal{D}_m(x)$	The $d \times 1$ derivative vector which ith entry is equal to $(\partial/\partial x_i)m(x)$
$\mathcal{H}_m(x)$	The $d \times d$ Hessian matrix which (i, j)th entry is equal to $\partial^2/(\partial x_i \partial x_j)m(x)$

Operators and Functions

$(\cdot)_+$	The positive part of the expression
$\lfloor \cdot \rfloor$	The integer part of the expression
$\lVert \cdot \rVert$	Vector norm, $\lVert x \rVert^2 = x^T x$
$\lVert \cdot \rVert_M$	Scaled vector norm, $\lVert x \rVert_M^2 = x^T M x$
x^*	Complex conjugate
$\arg\min\limits_x f(x)$	The minimizing argument of the function $f(\cdot)$ w.r.t. x
$\mathrm{E}\, X$	Mathematical expectation of the random variable X
$\mathrm{Var}\, X$	Variance of the random vector X

$\mathrm{infl}(X_i, x)$	The influence function
$\mu_k(m)$	$\int x^k m(x)\, dx$
$R(m)$	$\int m^2(x)\, dx$
A^T	The transpose of the matrix A
$\mathrm{tr}\, A$	The trace of the matrix A
$\|A\|,\ \det A$	The determinant of the matrix A
$\mathrm{vec}\, A$	The *vector* of the matrix A, obtained by stacking the columns of A underneath each other in order from left to right
$\mathrm{vech}\, A$	The *vector-half* of the matrix A, obtained from $\mathrm{vec}\, A$ by eliminating the above-diagonal entries of A

Index