Oliver Nelles

Nonlinear System Identification

From Classical Approaches to Neural Networks and Fuzzy Models

With 422 Figures

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milano
Paris
Singapore
Tokyo

Engineering ONLINE LIBRARY
http://www.springer.de/engine/

Springer

16. Linear Dynamic System Identification

The term *linear system identification* often refers exclusively to the identification of linear *dynamic* systems. In this chapter's title the term "dynamic" is explicitly mentioned to emphasize the clear distinction from static systems. An understanding of the basic concepts and the terminology of linear dynamic system identification is required in order to study the identification of *nonlinear dynamic* systems, which is the subject of all subsequent chapters. The purpose of this chapter is to introduce the terminology, concepts, and algorithms for linear system identification. Since this book deals extensively with local linear models as a very promising approach to nonlinear system identification, most of the methods discussed in this chapter can be transferred to this particular class of nonlinear models. It is one of the main motivations for the use of local linear model approaches that many existing and well-understood linear techniques can be successfully extended for nonlinear processes. A more detailed treatment of linear system identification can be found in [40, 81, 171, 172, 193, 233, 360]. Practical experience can be easily gathered by playing around with the MATLAB system identification toolbox [234].

This chapter is organized as follows. First, a brief overview of linear system identification is given to characterize the models and methods discussed here. Section 16.3 introduces the terminology used for naming the different linear model structures and explains the basic concept of the optimal predictor and prediction error methods for estimating linear models from data. After a brief discussion of time series models in Sect. 16.4, the linear models are classified into two categories: models with output feedback (Sect. 16.5) and models without output feedback (Sect. 16.6). Section 16.7 analyzes some advanced aspects that have been omitted in the preceding sections for the sake of an easier understanding. Recursive algorithms are summarized in Sect. 16.8. The extension to models with multiple inputs and outputs is presented in Sect. 16.10. Some specific aspects for identification with data measured in closed loop are introduced in Sect. 16.11. Finally, a summary gives some guidelines for the user.

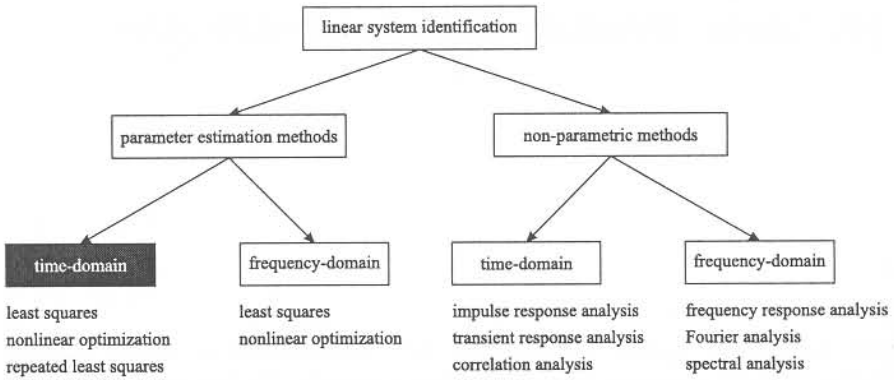

Fig. 16.1. Overview of linear system identification methods. Only the methods within the dark shaded box are discussed in this chapter. Note that the *methods* not the models are classified into *parametric* and *non-parametric* ones. Non-parametric models, such as a finite impulse response model, may indeed be estimated with a parametric method if the infinite series is approximated by a finite number of parameters

16.1 Overview of Linear System Identification

Figure 16.1 gives an overview of linear system identification methods. They can be distinguished into parametric and non-parametric approaches. It is helpful to distinguish clearly the *model* and the type of *method* applied to determine the degrees of freedom of the model. The model can be parametric or non-parametric:

- *Parametric models* can (or are assumed to be able to) describe the true process behavior exactly with a *finite* number of parameters. A typical example is a differential or difference equation model. Often the parameters have a direct relationship to physical quantities of the process, e.g., mass, volume, length, stiffness, viscosity.
- *Non-parametric models* generally require an *infinite* number of parameters to describe the process exactly. A typical example is an impulse response model.

Furthermore, parametric and non-parametric methods can be distinguished:

- *Parametric methods* determine a relatively small number of parameters. Usually these parameters are optimized according to some objective. A typical example is parameter estimation by linear regression. Parametric methods can also be used for determination of approximate non-parametric models whose number of parameters have been reduced to a finite number. A typical example is a finite impulse response (FIR) model that approximates the infinite impulse response of a process.
- *Non-parametric methods* are more flexible than parametric methods. They are used if less structure is imposed on the model. A typical example is Fourier analysis, which yields functions of frequency and thus is not describable by a finite number of parameters. Although eventually, in their actual implementation, non-parametric methods exhibit a certain (finite) number of "parameters" (e.g., for a discrete time Fourier analysis, the complex amplitudes for all discretized frequency intervals), this number is huge and independent of any model structure. Rather the number of "parameters" depends on factors such as the number of data samples or the quantization.

This chapter focuses on parametric models and methods for linear system identification. For a detailed discussion of non-parametric approaches refer to [81, 171]. Furthermore, this chapter considers only time-domain approaches.

16.2 Excitation Signals

The input signal $u(k)$ of the process under consideration plays an important role in system identification. Clearly, the input signal is the only possibility to influence the process in order to gather information about its behavior. Thus, the question arises: How should the input signal be chosen?

In most real-world applications there exist a large number of constraints and restrictions on the choice of the input signal. Certainly for any real process the input signal must be bounded, i.e., between a minimum $u_{\min}$ and maximum value $u_{\max}$. Furthermore, the measurement time is always limited. Besides these basic restrictions in the ideal case the user is free to design the input signal. This situation may arise for pilot plants or industrial processes that are not in regular operation. However, most often the situation is far from ideal. Typically safety restrictions must be obeyed, and one is not allowed to push the plant to its limits. If the plant is in normal operation, usually no or only slight changes to the standard input signal are allowed in order to meet the process goals, e.g., the specifications of the produced product. In the following, some guidelines for input signal design are given, which should be heeded whenever possible.

Figure 16.2 shows a process in which all disturbances are transferred to the output in the noise $n(k)$. Disturbances that in reality affect the input or some internal process states can be transformed to the process output by a proper frequency shaping by means of a filter. Because the noise $n(k)$ cannot be influenced, the input signal is the user's only degree of freedom to determine the signal-to-noise ratio. Thus, the input *amplitudes* should exploit the full range from $u_{\min}$ to $u_{\max}$ in order to maximize the power of the input signal and consequently the signal-to-noise ratio. Therefore, it is reasonable to switch between $u_{\min}$ and $u_{\max}$.

The spectrum of the input signal determines the *frequencies* where the power is put in. Obviously, the identified model will be of higher quality for the frequencies that are strongly excited by the input signal than for those

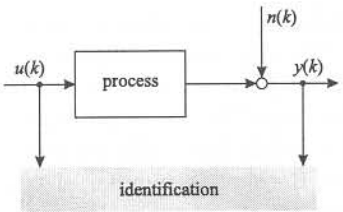

Fig. 16.2. Input and disturbed output of a process are measured and used for identification

that are not. If the input signal is a sine wave, only information about one single frequency is gathered, and the model quality at this frequency will be excellent at the cost of other frequency ranges. So, for input signal design the purpose of the model is of fundamental importance. If the emphasis is on the static behavior the input signal should mainly excite low frequencies. If the model is required to operate only at some specific frequencies an additive mixture of sine waves with exactly these frequencies is the best choice for the input signal. If the model is utilized for controller design a good match of the process around the Nyquist frequency ($-180°$ phase shift) is of particular importance. An excitation signal for model-based controller design is best generated in closed loop [116]. If very little is known about the intended use of the model and the characteristics of the process, a white input signal is the best choice since it excites all frequencies equally well. Note, however, that often very high frequencies do not play an important role, especially if the sampling time T_0 is chosen very small. Although in practice it is quite common to choose the sampling frequency as high as possible with the equipment used, it is advisable to choose the sampling time at about one twentieth to one tenth of the settling time of the process [170]. If sampling is performed much faster the damping of the process typically is so large at high frequencies that it makes no sense to put too much energy in these high frequency ranges. Furthermore, most model structures will be a simplified version of reality and thus independent of the excitation signal; structural errors will inevitably be large in the high frequency range.

Example 16.2.1. Input Signals for Excitation
The following figures illustrate some typical input signals and the corresponding output of a first order system with gain $K = 1$ and time constant $T = 8s$ sampled with $T_0 = 1$s that follows the difference equation

$$y(k) = 0.1175u(k-1) + 0.8825y(k-1)\,. \tag{16.1}$$

This process is excited with each of the input signals shown in Figs. 16.3–16.7, and 100 measurements are taken. These samples are used for identification of a first order ARX model; see Sect. 16.3.1. The process is disturbed with filtered white noise of variance 0.01. Note that the noise filter is chosen equal to the denominator dynamics of the process $1/A$ in order to meet the

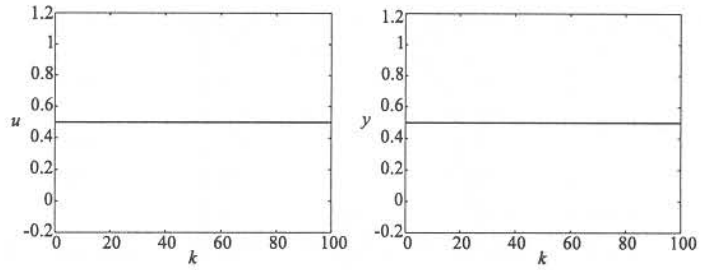

Fig. 16.3. Excitation with a constant signal (left) and the undisturbed process output (right)

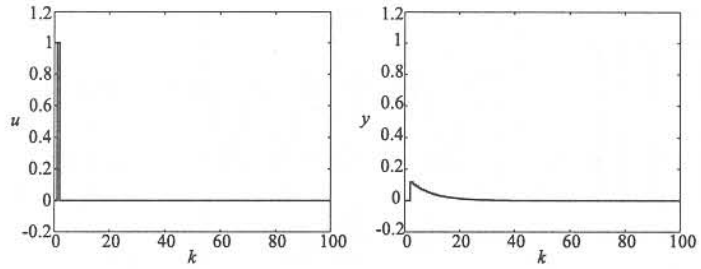

Fig. 16.4. Excitation with an impulse signal (left) and the undisturbed process output (right)

assumption of the ARX model. Because the model structure and the process structure are identical the bias error (Sect. 7.2) of the model is equal to zero, and all errors are solely due to the noise.

The results of the identification are given in each figure and summarized in Table 16.1. The comparison of the input signals demonstrates the following:

- *Constant:* Only suitable for identification of one parameter, here the static gain K, which is given by $b_1/(1-a_1)$. Not suitable for identification because no dynamics are excited. The parameters b_1 and a_1 cannot be estimated independently; only the ratio $b_1/(1-a_1)$ is correctly identified.
- *Impulse:* Not well suited for identification. In particular, the gain is estimated very inaccurately.
- *Step:* Well suited for identification. Low frequencies are emphasized. The static gain is estimated very accurately.
- *Rectangular:* Well suited for identification. Depending on the frequency of the rectangular signal a desired frequency range can be emphasized. For the signal in Fig. 16.6 the time constant is estimated very accurately.
- *PRBS (pseudo random binary signal):* Well suited for identification. Imitates white noise in discrete time with a deterministic signal and thus excites all frequencies equally well.

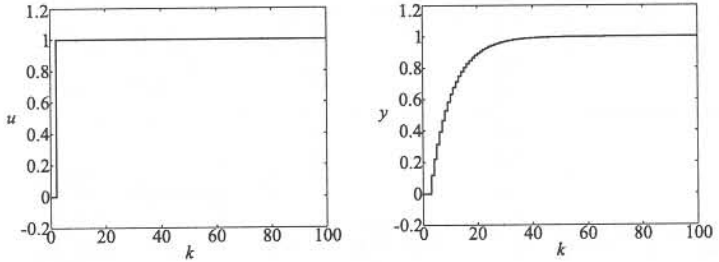

Fig. 16.5. Excitation with a step signal (left) and the undisturbed process output (right)

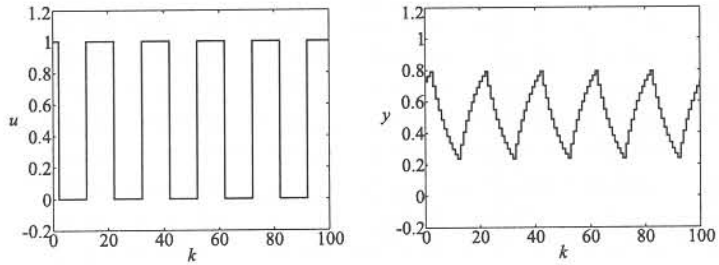

Fig. 16.6. Excitation with a rectangular signal (left) and the undisturbed process output (right)

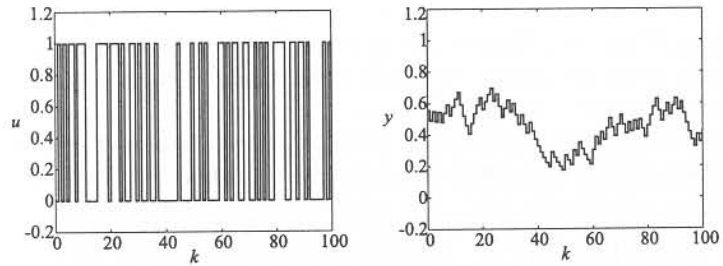

Fig. 16.7. Excitation with a PRBS (pseudo random binary signal) (left). The PRBS is an deterministic approximation of white noise in discrete time [171]. The undisturbed process output is shown at the right

16.3 General Model Structure

In this section a general linear model structure is introduced from which all linear models can be derived by simplifications. This general model is not normally used in practice; it just serves as a unified framework. The output $y(k)$ of a deterministic linear system at time k can be computed by filtering the input $u(k)$ through a linear filter $G(q)$ (q denotes the forward shift operator, i.e., $q^{-1}x(k) = x(k-1)$, and thus it is the time domain counterpart of the $z = e^{j\omega}$-operator in the frequency domain):

Table 16.1. Identification results for different excitation signals

Input signal	b_1	a_1	K	T [s]
Constant	0.2620	-0.7392	1.0048	3.3098
Impulse	0.0976	-0.8570	0.6826	6.4800
Step	0.1220	-0.8780	0.9998	7.6879
Rectangular	0.1170	-0.8834	1.0033	8.0671
PRBS	0.1201	-0.8796	0.9980	7.7964
True process	0.1175	-0.8825	1	8

$$y(k) = G(q)u(k) = \frac{\widetilde{B}(q)}{\widetilde{A}(q)}u(k)\,. \tag{16.2}$$

In general, the linear transfer function $G(q)$ may possess a numerator $\widetilde{B}(q)$ and a denominator $\widetilde{A}(q)$. In addition to the deterministic part, a stochastic part can be modeled. By filtering white noise $v(k)$ through a linear filter $H(q)$ any noise frequency characteristic can be modeled. Thus, an arbitrary noise signal $n(k)$ can be generated by

$$n(k) = H(q)v(k) = \frac{\widetilde{C}(q)}{\widetilde{D}(q)}v(k)\,. \tag{16.3}$$

A general linear model describing deterministic and stochastic influences is obtained by combining both parts (see Fig. 16.8a)

$$y(k) = G(q)u(k) + H(q)v(k)\,. \tag{16.4}$$

The filter $G(q)$ is called the *input transfer function*, since it relates the input $u(k)$ to the output $y(k)$, and the filter $H(q)$ is called the *noise transfer function*, since it relates the noise $v(k)$ to the output $y(k)$. These transfer functions $G(q)$ and $H(q)$ can be split into their numerator and denominator polynomials; see Fig. 16.8b. For future analysis it is helpful to separate a possibly existent common denominator dynamics $A(q)$ from $G(q)$ and $H(q)$; see Figs. 16.8c and 16.8d. Thus, $F(q)A(q) = \widetilde{A}$ and $D(q)A(q) = \widetilde{D}$. If $\widetilde{A}(q)$ and $\widetilde{D}(q)$ do not share a common factor then simply $A(q) = 1$. These notations of the transfer functions in Fig. 16.8a and the polynomials in Fig. 16.8d have been accepted standards since the publication of Ljung's book [233]. So the general linear model can be written as

$$y(k) = \frac{B(q)}{F(q)A(q)}u(k) + \frac{C(q)}{D(q)A(q)}v(k) \tag{16.5}$$

or equivalently as

$$A(q)y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}v(k)\,. \tag{16.6}$$

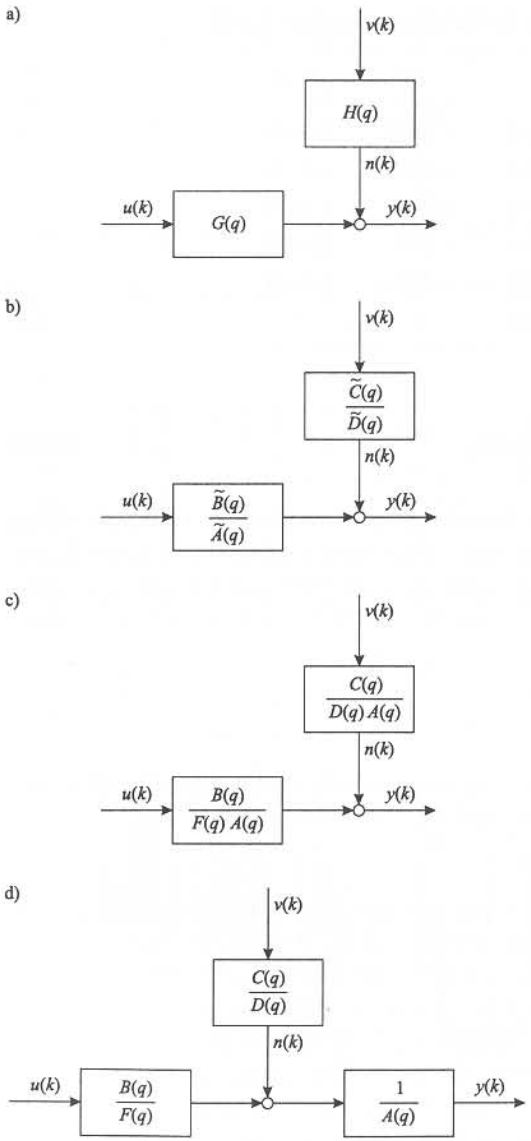

Fig. 16.8. A general linear model

By making special assumptions on the polynomials A, B, C, D, and F the widely applied linear models are obtained from this general form. Before these simpler linear models are introduced it is helpful to make a few general remarks on the terminology and to discuss some general aspects that are valid for all types of linear models.

16.3.1 Terminology and Classification

Unfortunately the standard terminology of linear dynamic models is quite confusing. The reason for this is the historic development of these models within different disciplines. Thus, some expressions stem from time series modeling in economics. Economists typically analyze and try to predict time series such as stock prices, currency exchange rates, and unemployment rates. A common characteristic of all these applications is that the relevant input variables are hardly known and the number of possibly relevant inputs is huge. Therefore, economists started by analyzing the time series on its own without taking any input variables into account. Such models result from the general model in Fig. 16.8 and (16.6) by discarding the input, that is, $u(k) = 0$. Then the model becomes fully stochastic. Such a time series model is depicted in Fig. 16.9, opposed to the purely deterministic model shown in Fig. 16.10. From this time series point of view the terminology used in the following is logical and straightforward. Ljung's book [233] established this as the now widely accepted standard in system identification.

A time series model with just a denominator polynomial (Fig. 16.11)

$$y(k) = \frac{1}{D(q)} v(k) \tag{16.7}$$

is called an *autoregressive* (AR) model.

A time series model with just a numerator polynomial (Fig. 16.11)

$$y(k) = C(q)v(k) \tag{16.8}$$

is called a *moving average* (MA) model.

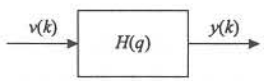

Fig. 16.9. A general linear time series model. The model input $v(k)$ is a white noise signal. There is no deterministic input $u(k)$

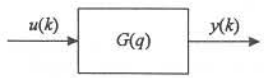

Fig. 16.10. A general linear deterministic model. The model input $u(k)$ is a deterministic signal. There is no stochastic influence such as a white noise $v(k)$

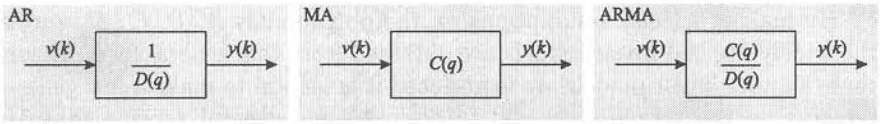

Fig. 16.11. An overview of time series models: autoregressive (AR), moving average (MA), and autoregressive moving average (ARMA) models

A time series model with a numerator and denominator polynomial (Fig. 16.11)

$$y(k) = \frac{C(q)}{D(q)} v(k) \tag{16.9}$$

is called an *autoregressive moving average* (ARMA) model.

It is obvious that a model based on the time series only, without taking any relevant input variable into account, cannot be very accurate. Therefore, more accurate models are constructed by incorporating one (or more) input variable(s) into the model. This input $u(k)$ is called an *exogenous* input. With these considerations, the time series models in Fig. 16.11 can be extended by adding an "X" for exogenous input. To extend a moving average time series model with an exogenous input is highly uncommon. Thus, something like "MAX" is rarely used.

Figure 16.12 gives an overview of the most important linear input/output models, which are briefly discussed in the following. All models on the left hand side of Fig. 16.12 are denoted by AR... and belong to the class of *equation error* models. Their characteristic is that the filter $1/A(q)$ is common to both the deterministic process model and the stochastic noise model. All models on the right hand side of Fig. 16.12 belong to the class of *output error* models, which is characterized by a noise model that is independent of the deterministic process model.

The *autoregressive with exogenous input* (ARX)[1] model (Fig. 16.12) is an extended AR model:

$$y(k) = \frac{B(q)}{A(q)} u(k) + \frac{1}{A(q)} v(k) . \tag{16.10}$$

Here the term "autoregressive" is related to the transfer function from the input $u(k)$ to the output $y(k)$ as well as to the noise transfer function from $v(k)$ to $y(k)$. Thus, the deterministic and the stochastic part of the ARX model possess an identical denominator dynamics. For a more detailed discussion refer to Sect. 16.5.1.

[1] In a considerable part of the literature and in older publications in particular, the ARX model is called an "ARMA" model to express the fact that both a numerator and a denominator polynomial exist. However, as discussed above, this book follows the current standard terminology established in [233], where ARMA stands for the time series model in (16.9).

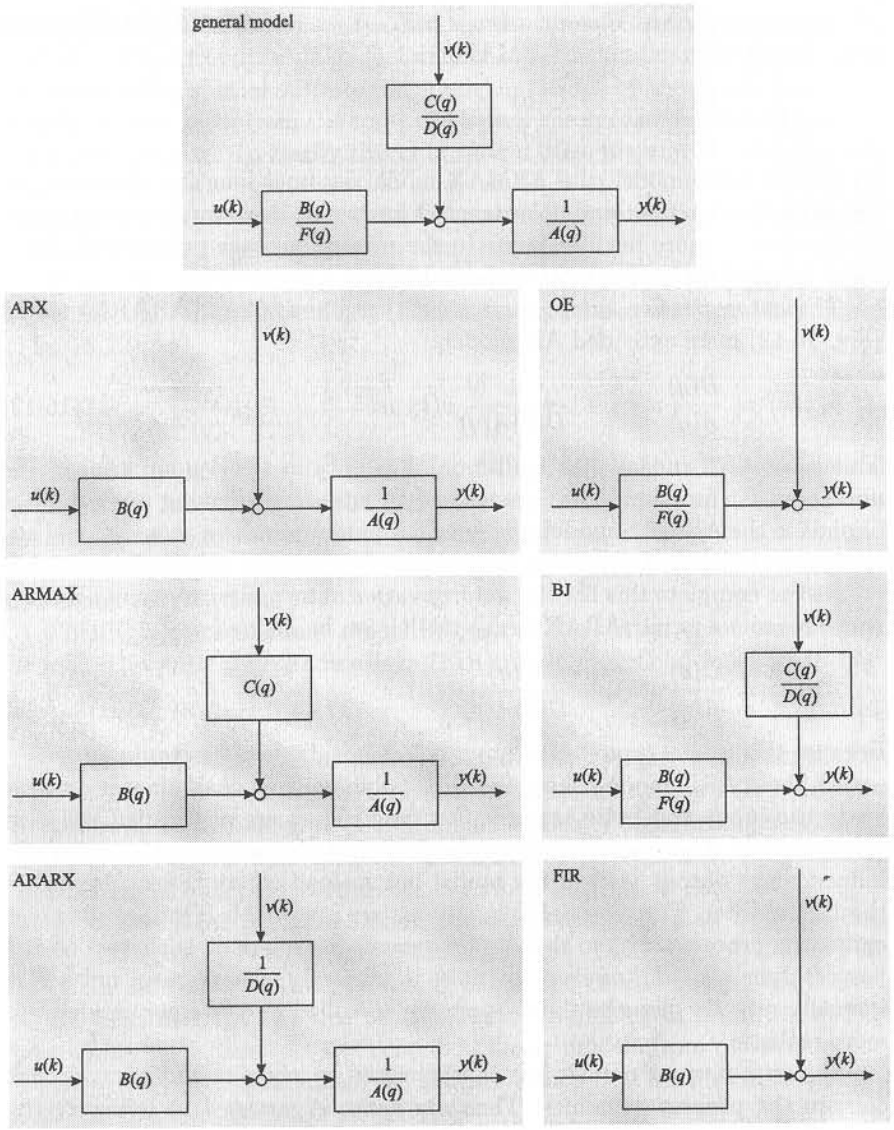

Fig. 16.12. An overview of common linear dynamic models

The *autoregressive moving average with exogenous input* (ARMAX) model (Fig. 16.12) is an extended ARMA model:

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{C(q)}{A(q)}v(k)\,. \tag{16.11}$$

As for the ARX model, the ARMAX model assumes identical denominator dynamics for the input and noise transfer functions. However, the noise transfer function is more flexible owing to the moving average polynomial. For a more detailed discussion refer to Sect. 16.5.2.

The *autoregressive autoregressive with exogenous input* (ARARX) model (Fig. 16.12) is an extended AR model:

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{D(q)A(q)}v(k)\,. \tag{16.12}$$

This is an ARX model with additional flexibility in the denominator of the noise transfer function. Thus, instead of an additional moving average filter $C(q)$ as in the ARMAX model, the ARARX model possesses an additional autoregressive filter $1/D(q)$. For a more detailed discussion refer to Sect. 16.5.3.

Just to complete this list the *autoregressive autoregressive moving average with exogenous input* (ARARMAX) model can be defined as

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{C(q)}{D(q)A(q)}v(k)\,. \tag{16.13}$$

Because this model type is hardly used it is not discussed in more detail.

All these AR... models share the $A(q)$ polynomial as denominator dynamics in the input and noise transfer functions. They are also called *equation error* models. This corresponds to the fact that the noise does not directly influence the output $y(k)$ of the model but instead enters the model before the $1/A(q)$ filter. These model assumptions are reasonable if indeed the noise enters the process early, so that its frequency characteristic is shaped by the process dynamics. If, however, the noise is primarily measurement noise that typically directly disturbs the output, the so-called *output error* models are more realistic.

The output error models are characterized by noise models that do not contain the process dynamics. Thus, the noise is assumed to influence the process output directly. The terminology of these models does not follow the rules given above for an extension of the time series models. Rather, the point of view changes from time series models (where the noise model is in the focus) to input/output models (where the attention turns to the deterministic input).

The most straightforward input/output model is the *output error* (OE) model (Fig. 16.12):

$$y(k) = \frac{B(q)}{F(q)}u(k) + v(k)\,. \tag{16.14}$$

This OE model is one special model in the class of output error models. Unfortunately it is difficult to distinguish between the class of output error models and this special output error model in (16.14) by the name. Therefore, it must become clear from the context whether the special model or the model class is referred to. To clarify this confusion a little bit, the abbreviation OE always refers to the special output error model in (16.14). In contrast to the ARX model, white noise enters the OE model directly without any filter. For a more detailed discussion refer to Sect. 16.5.4.

This OE model can be enhanced in flexibility by filtering the white noise through an ARMA filter. This defines the *Box-Jenkins* (BJ) model (Fig. 16.12):

$$y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}v(k)\,. \tag{16.15}$$

The BJ model relates to the ARARMAX model as the OE model relates to the ARX model. The input and noise transfer functions are separately parameterized and therefore independent. The special cases of a BJ model $C(q) = 1$ or $D(q) = 1$ do not have special names. For a more detailed discussion refer to Sect. 16.5.5.

Finally, a quite different model belongs to the output error model class, as well. The *finite impulse response* (FIR) model is defined by (Fig. 16.12)

$$y(k) = B(q)u(k) + v(k)\,. \tag{16.16}$$

The FIR model is an OE or an ARX model without any feedback, that is, $F(q) = 1$ or $A(q) = 1$, respectively. As an extension of the FIR model the *orthonormal basis functions* (OBF) model is also of significant practical interest. However, the OBF model does not fit well in the framework presented here. The FIR and OBF models are described in detail in Sect. 16.6.

At a first sight all these different model structures may be quite confusing. However, it is sufficient to remember the ARX, ARMAX, OE, FIR, and OBF models for an understanding of the rest of this book. Nevertheless, all concepts discussed in this chapter are of fundamental importance. Figures 16.13–16.16 illustrate the described linear models from different points of view. Table 16.2 summarizes the simplifications that lead from the general model to the specific model structures.

Note that for the sake of simplicity the processes and models are assumed to possess no dead time. However, in any equation a dead time dT_0 can easily be introduced by replacing the input $u(k)$ with the d steps delayed input $u(k-d)$. Furthermore, it is assumed that the processes and models have no direct path from the input to the output (i.e., they are strictly proper), so that $u(k)$ does not immediately influence $y(k)$. Thus, terms like $b_0u(k)$ do not appear in the difference equations. This assumption is fulfilled for almost any real-world process.

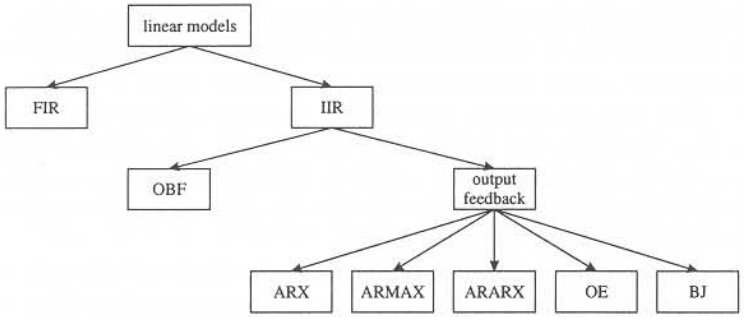

Fig. 16.13. Classification of linear models according to finite impulse response (FIR) and infinite impulse response (IIR) filters

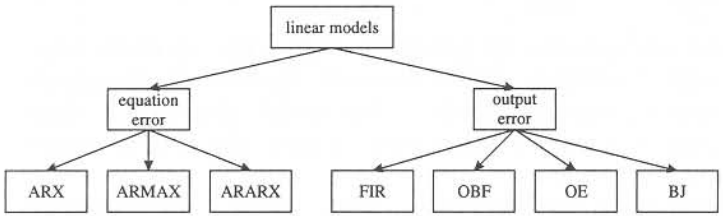

Fig. 16.14. Classification of linear models according to equation error and output error models

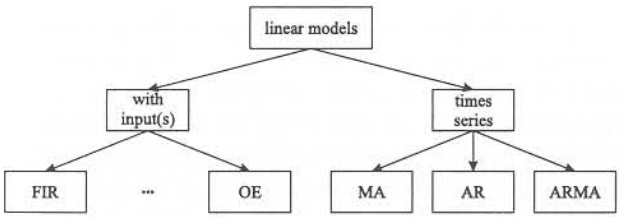

Fig. 16.15. Classification of linear models according to input/output and time series models

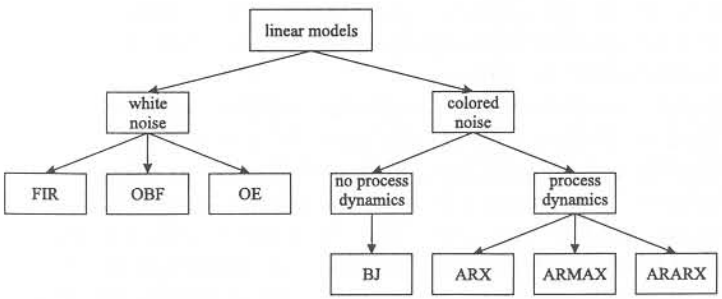

Fig. 16.16. Classification of linear models according to noise properties. The box entitled "process dynamics" refers to noise filter, which include the process denominator dynamics $1/A(q)$

Table 16.2. Common linear models

Model structures	Model equations
MA	$y(k) = C(q)\, v(k)$
AR	$y(k) = 1/D(q)\, v(k)$
ARMA	$y(k) = C(q)/D(q)\, v(k)$
ARX	$y(k) = B(q)/A(q)\, u(k) + 1/A(q)\, v(k)$
ARMAX	$y(k) = B(q)/A(q)\, u(k) + C(q)/A(q)\, v(k)$
ARARX	$y(k) = B(q)/A(q)\, u(k) + 1/D(q)A(q)\, v(k)$
ARARMAX	$y(k) = B(q)/A(q)\, u(k) + C(q)/D(q)A(q)\, v(k)$
OE	$y(k) = B(q)/F(q)\, u(k) + v(k)$
BJ	$y(k) = B(q)/F(q)\, u(k) + C(q)/D(q)\, v(k)$
FIR	$y(k) = B(q)\, u(k) + v(k)$

16.3.2 Optimal Predictor

Probably the most common application of a model is forecasting the future behavior of a process. Two cases have to be distinguished: *simulation* and *prediction*. If the response of the model to an input sequence has to be calculated while the process outputs are unknown, this is called *simulation*. If, however, the process outputs are known up to some time instant, say $k-1$, and it is asked for the model output l steps in the future, this is called *prediction*. Very often one is interested in the one-step prediction, i.e., $l = 1$, and if nothing else is explicitly stated in the following prediction will mean one-step prediction. Figures 16.17 and 16.18 illustrate the difference between simulation and prediction; see also Sects. 1.1.2 and 1.1.3.

Simulation. It is obvious from Fig. 16.17 that simulation is fully deterministic:

$$\hat{y}(k) = G(q)u(k)\,. \tag{16.17}$$

Thus, the noise model $H(q)$ seems irrelevant for simulation. Note, however, that the noise model $H(q)$ influences the estimation of the parameters in $G(q)$ and therefore it affects the simulation quality although $H(q)$ does not explicitly appear in (16.17).

Because the process output is unknown, no information about the disturbances is available. In order to get some "feeling" how the disturbed process output qualitatively may look, it is possible to generate a white noise signal $w(k)$ with proper variance by a computer [233], to filter this signal through the noise filter $H(q)$, and to add this filtered noise to the deterministic model output

$$\hat{y}(k) = G(q)u(k) + H(q)w(k)\,. \tag{16.18}$$

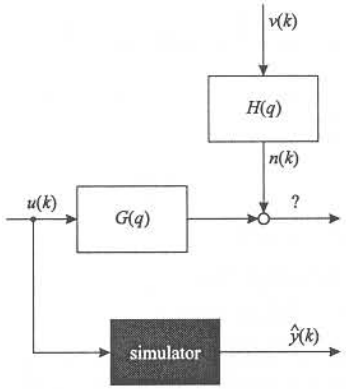

Fig. 16.17. For simulation, only the inputs are known. No information about the real process output is available

Note, however, that (16.18) is just a better qualitative output than (16.17). The smaller simulation error can be expected from (16.17) since $w(k)$ is a different white noise signal than the original but not measurable $v(k)$.

Prediction. In contrast to simulation, for prediction the information about the previous process output can be utilized. Thus, the *optimal predictor* should combine the inputs and previous process outputs in some way. So the optimal *linear* predictor can be defined as the linear combination of the filtered inputs and the filtered outputs

$$\begin{aligned}\hat{y}(k|k-1) \;=\; & s_0 u(k) + s_1 u(k-1) + \ldots + s_{ns} u(k-ns) \\ & + t_1 y(k-1) + \ldots + t_{nt} y(k-nt)\end{aligned} \tag{16.19}$$

or

$$\hat{y}(k|k-1) \;=\; S(q)u(k) + T(q)y(k)\,. \tag{16.20}$$

Note that the filter $T(q)$ does not contain the term t_0 since of course the value $y(k)$ is not available when predicting $\hat{y}(k|k-1)$. For most real-world processes $s_0 = 0$ as well, because the input does not instantaneously influence the output, i.e., the model is strictly proper.

The term $S(q)u(k)$ contains information about the deterministic part of the predictor while the term $T(q)y(k)$ introduces a stochastic component into the predictor since only $y(k)$ is disturbed by noise.

The following question arises: What are the best filters $S(q)$ and $T(q)$? More exactly speaking, which filters result in the smallest squared prediction error (prediction error variance)? It can be shown that the optimal predictor is [233]

$$\hat{y}(k|k-1) \;=\; \frac{G(q)}{H(q)}u(k) + \left(1 - \frac{1}{H(q)}\right) y(k) \tag{16.21}$$

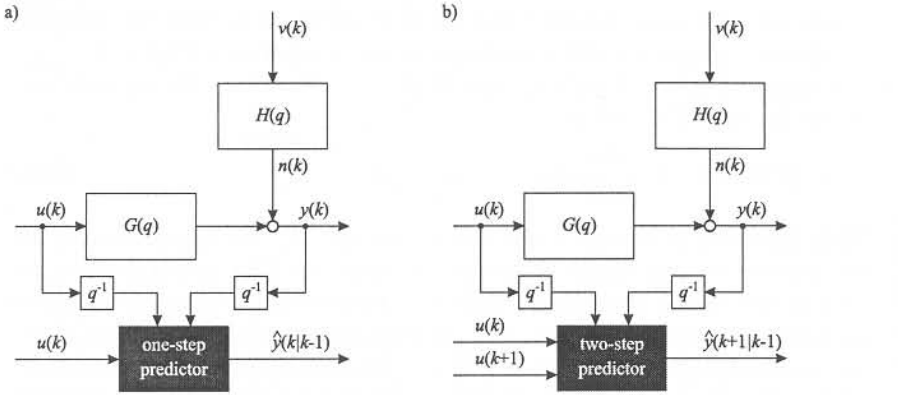

Fig. 16.18. a) One-step prediction and b) two-step prediction. The expression "$|k-1$" means "on the information available at time instant $k-1$". For prediction, besides the inputs the previous process outputs are known. Note that if the prediction horizon l becomes very large the importance of the information about the previous process outputs decreases. Thus, as $l \to \infty$ prediction approaches simulation; see Fig. 16.17

or

$$H(q)\hat{y}(k|k-1) \;=\; G(q)u(k) + (H(q)-1)\,y(k)\,. \tag{16.22}$$

Thus, $S(q) = G(q)/H(q)$ and $T(q) = 1 - 1/H(q)$.

It is very helpful to discuss some special cases in order to illustrate this optimal predictor equation.

- *ARX model:* $G(q) = B(q)/A(q)$ and $H(q) = 1/A(q)$. Therefore, the optimal predictor for an ARX model is

$$\hat{y}(k|k-1) \;=\; B(q)u(k) + (1 - A(q))\,y(k)\,. \tag{16.23}$$

  Thus, the inputs are filtered through the $B(q)$ polynomial and the process outputs are filtered through the $1 - A(q)$ polynomial. Consequently, the predicted model output $\hat{y}(k|k-1)$ can be generated by applying simple moving average filtering. Because an ARX model implies correlated disturbances, namely white noise filtered through $1/A(q)$, the process output contains valuable information about the disturbances. This information allows one to make a prediction on the actual disturbance at time instant k, which is implicitly performed by the term $((1 - A(q))y(k)$.
- *ARMAX model:* $G(q) = B(q)/A(q)$ and $H(q) = C(q)/A(q)$. Therefore, the optimal predictor for an ARMAX model is

$$\hat{y}(k|k-1) \;=\; \frac{B(q)}{C(q)}u(k) + \left(\frac{C(q) - A(q)}{C(q)}\right) y(k)\,. \tag{16.24}$$

  This equation is much more difficult than the ARX predictor. A characteristic is that both the input and the process output are filtered through

filters with the same denominator dynamics $C(q)$. Note that the ARMAX predictor contains the ARX predictor as the special case $C(q) = 1$.

- *OE model:* $G(q) = B(q)/A(q)$ and $H(q) = 1$. Therefore, the optimal predictor for an OE model is

$$\hat{y}(k|k-1) = \frac{B(q)}{A(q)}u(k)\,. \tag{16.25}$$

This, however, is exactly a *simulation*; see (16.17)! No information about the process output enters the predictor equation. The reason for this obviously lies in the noise filter $H(q) = 1$. Intuitively this can be explained as follows. The term $T(q)y(k)$ in (16.20) contains the information about the stochastic part of the model. If the process is disturbed by unfiltered white noise, as is assumed in the OE model, there is no correlation between disturbances $n(k)$ at different times k. Thus, knowledge about previous disturbances that is contained in $y(k)$ does not help to predict into the future. Thus, the simulation of the model is the optimal prediction in the case of an OE model. At first sight, it seems strange to totally ignore the knowledge of $y(k)$. However, an incorporation of the white noise corrupted $y(k)$ into the predictor would deteriorate the performance.

16.3.3 Some Remarks on the Optimal Predictor

It is important to make some remarks on the optimal predictor which have been omitted above for easier understanding.

- Equation (16.21) for the optimal predictor can be derived as follows. The starting point is the model equation

$$y(k) = G(q)u(k) + H(q)v(k)\,. \tag{16.26}$$

The optimal predictor should be capable of extracting all information out of the signals. Thus, the prediction error, i.e., the difference between the process output $y(k)$ and the predicted output $\hat{y}(k|k-1)$, should be equal to the white noise $v(k)$, since this is the only unpredictable part in the system:

$$v(k) = y(k) - \hat{y}(k|k-1)\,. \tag{16.27}$$

This equation can be used to eliminate $v(k)$ in (16.26). Then the following relationship results:

$$y(k) = G(q)u(k) + H(q)\left(y(k) - \hat{y}(k|k-1)\right)\,. \tag{16.28}$$

If in this equation $\hat{y}(k|k-1)$ is isolated the optimal predictor in (16.21) results. The optimal predictor thus leads to white residuals. Therefore, an analysis of the spectrum of the real residuals can be used to test whether the model structure is appropriate.

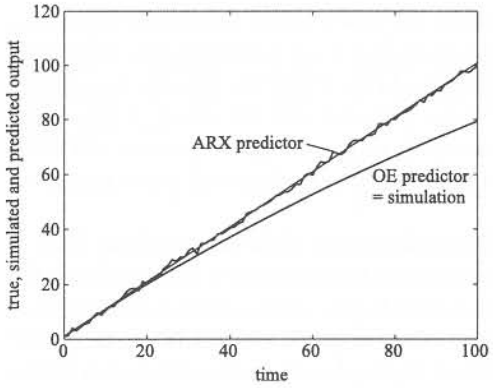

Fig. 16.19. OE and ARX predictor for a process with integral behavior. An OE model of the process $q/(q-1)$ disturbed by white output noise is assumed to be identified to $q/(q-0.995)$. The optimal OE predictor in (16.25) simulates the process

- It has been demonstrated above that the optimal predictor for ARX models includes previous inputs and process outputs while the optimal predictor for OE models includes only previous inputs. This statement is correct if the transfer functions $G(q)$ and $H(q)$ are identical with the model. However, if the model only approximates the process, as is the case in all real-world applications, this statement is not necessarily valid any more. Consider, for example, a process with integral behaviour $G(q) = Kq/(q-1)$ and additive white measurement noise at the output. Assume that an OE model is applied, which indeed is the correct structure for these noise properties, and the parameter K is not estimated exactly. Then, for this OE model, the ARX predictor may yield better results than the OE predictor. The explanation for this fact is that the error of the simulated output obtained by the OE predictor increases linearly with time proportional to the estimation error in K, while the ARX predictor is based on process outputs and thus cannot "run away" from the process. Figure 16.19 compares the behavior of an OE and an ARX predictor for this process.
  Because the model parameters have not been identified exactly the model implements no integrator but a first order time lag behavior (with a large time constant). Thus, the OE predictor quality becomes worse as time proceeds. In contrast, if the ARX predictor in (16.49) is utilized for the OE model the prediction always remains close to the true process because it is also based on process outputs. The price to be paid is the introduction of the disturbances into the prediction since the process output is corrupted by noise.
  Note that this example illustrates an extreme case since the investigated process is not stable. Nevertheless, in practice even for stable processes it can be advantageous to use the ARX predictor for models belonging to the output error class. This is because non-modeled nonlinear effects can

lead to significant deviations between the process and the simulated model output, while a one-step prediction with the ARX predictor will follow the operating point better. Generally, the ARX predictor can be reasonably utilized for output error models if the disturbances are small. Thus, there exists some kind of tradeoff between the wrongly assumed noise model when using the ARX predictor and the model/process mismatch when using the OE predictor.

- The optimal predictor in (16.21) is only stable if the noise filter $H(q)$ is minimum phase. Stability of the predictor is a necessary condition for the application of the prediction error methods (see next section). If $H(q)$ were non-minimum phase, $1/H(q)$ and thus the predictor would be unstable. However, then the noise filter could be replaced by its minimum phase counterpart, i.e., the conjugate complex $H^*(q) = H(q^{-1})$, because the purpose of the filter $H(q)$ is merely to shape the frequency spectrum of the disturbance $n(k)$ by filtering $v(k)$. But the spectrum of the disturbance $n(k)$ is determined only by $|H(q)|^2$, which is equal to $|H|^2 = H(q)H^*(q)$ (*spectral factorization*). Thus, both filters $H(q)$ and $H^*(q)$ result in the same frequency shaping, and therefore the filter that is stable invertible can be selected for the optimal predictor.
- Another assumption not yet mentioned is made in the optimal predictor equation (16.21). The influence of the initial conditions is neglected. Consider, for example, an OE model

$$\begin{aligned} y(k) &= b_1 u(k-1) + \ldots + b_m u(k-m) \\ &\quad - a_1 y(k-1) - \ldots - a_m y(k-m) \,. \end{aligned} \tag{16.29}$$

For this model m initial conditions have to be assumed; at time $k = 0$ these are the values of $y(-1), \ldots, y(-m)$. Typically, these initial conditions are set to zero. This assumption is reasonable since for stable systems the initial conditions decay exponentially with time. Strictly speaking, the optimal predictor in (16.21) is only the *stationary* optimal predictor. If the initial conditions were to be taken into account the optimal predictor would become *time-variant* and would asymptotically approach (16.21). This relationship is well known between the Kalman filter, which represents the optimal time-variant predictor considering the initial conditions and its stationary solution, the Luenberger observer, which is valid only as time $\to \infty$. Nevertheless, since the initial conditions decay rapidly, in practice the stationary counterpart, i.e., the optimal predictor in (16.21), is sufficiently accurate and much simpler to deal with.

16.3.4 Prediction Error Methods

Usually the optimal predictor is used for measuring the performance of the corresponding model. The prediction error is the difference between the desired model output (= process output) and the one-step prediction performed by the model

$$e(k) = y(k) - \hat{y}(k|k-1) \,. \tag{16.30}$$

In the following, the term *prediction error* is used as a synonym for *one-step* prediction error. With the optimal predictor in (16.21) the prediction error becomes

$$e(k) = \frac{1}{H(q)} y(k) - \frac{G(q)}{H(q)} u(k) \,. \tag{16.31}$$

For example, an OE model has the following prediction error: $e(k) = y(k) - G(q)u(k)$. Most identification algorithms are based on the minimization of a loss function that depends on this one-step prediction error. Although this is the most common choice it can be reasonable to minimize another error measure. For example, *predictive control* algorithms utilize a model to predict a number of steps, say l, into the future. In this case, the performance can be improved by minimizing the error of an l-step prediction $e(k) = y(k) - \hat{y}(k|k-l)$ where l is the prediction horizon [230, 340, 341, 411]. Because the computation of such a multi-step predictor becomes more and more involved for larger prediction horizons l, even for model-based predictive control typically the one-step prediction error in (16.30) is used for identification.

For reasons discussed in Sect. 2.3, the sum of squared prediction errors is usually used as the loss function, i.e., with N data samples

$$J = \sum_{i=1}^{N} e^2(i) \,. \tag{16.32}$$

It is discussed in Sect. 2.3.1 that this choice is optimal (in the maximum likelihood sense) if the noise is Gaussian distributed. Another property of the sum of squared errors loss function is that the parameters of the ARX model structure can be estimated by linear optimization techniques; see Sect. 16.5.1.

Note that a sensible minimization of the loss function in (16.32) requires the predictor to be *stable*, which in turn requires that $G(q)$ is stable and $H(q)$ is minimum phase. Otherwise, the mismatch between process and model due to different initial values would not decay exponentially (as in the stable case); rather they would influence the minimization procedure decisively.

The loss function in (16.32) can be extended by filtering the prediction errors through a linear filter $L(q)$. Since the unfiltered prediction error is, (see (16.31))

$$e(k) = \frac{1}{H(q)} \left(y(k) - G(q)u(k) \right) \,, \tag{16.33}$$

the filtered prediction error e_F can be written as

$$e_F(k) = \frac{L(q)}{H(q)} \left(y(k) - G(q)u(k) \right) \,. \tag{16.34}$$

Obviously, the filter $L(q)$ has the same effect as the inverse noise model $1/H(q)$. Thus, the filter $L(q)$ can be fully incorporated into the noise model

$H(q)$ or vice versa. The understanding of this relationship is important for some of the identification algorithms discussed in the following sections. For more details about this relationship refer to Sect. 16.7.4.

16.4 Time Series Models

A time series is a signal that evolves over time[2], such as the Dollar/Euro exchange rate, the Dow Jones index, the unemployment rate in a country, the world's population, the amount of rain fall in a particular area, or the sound of a machine received with a microphone. A characteristic of all time series is that the current value is usually dependent on previous values. Thus, a dynamic model is required for a proper description of a time series. Furthermore, typically the driving inputs, i.e., the variables that influence the time series, are not known, are not measurable, or are so huge in number that it is not feasible to include them in the model. It is no coincidence that the typical examples for time series listed above are mostly non-technical. Often in engineering applications the relationships between different quantities are quite well understood, and at least some knowledge about the basic laws is available. Then it is more reasonable to build a model with deterministic inputs and possibly additional stochastic component. In economy and social sciences the dependencies between different variables are typically much more complex, and thus time series modeling plays a greater role in these disciplines.

Because time series models (as defined here) do not take any deterministic input into account, the task is simply to build a model for the time series with the information about the past realization of this time series only. Because no external inputs $u(k)$ are considered, it is clear that such a model will be of relatively low quality. Nevertheless it may be possible to identify a model that allows short term predictions (typically one-step predictions) with sufficient accuracy or which allows to gain insights about the underlying process.

Since no inputs are available, time series models are based on the following idea. The time series is thought to be generated by a (hypothetical) white noise signal, which drives a dynamic system. This dynamic system is then identified with the time series data. The main difficulty is that the input of this system, that is, the white noise signal, is unknown. Since this chapter deals with linear system identification the dynamic system is assumed to be linear, and the following model of the time series results (see also Fig. 16.9):

$$y(k) = H(q)v(k) = \frac{C(q)}{D(q)}v(k)\,, \tag{16.35}$$

where $y(k)$ is the time series and $v(k)$ is the artificial white noise signal. In the following three sections two special cases of (16.35) and finally the general

[2] In some cases the signal may not depend on time but rather on space (e.g. in geology) or some other quantity.

time series model in (16.35) are briefly discussed. The model (16.35) can be further extended by an integrator to deal with non-stationary processes. For more details refer to [47].

16.4.1 Autoregressive (AR)

The autoregressive time series model shown in Fig. 16.20 is very common since it allows one to shape the frequency characteristics of the model with a few *linear* parameters. In many technical applications of time series modeling one is interested in resonances, i.e., weakly damped oscillations at certain frequencies which may be hidden under a high noise level. Then an AR (or ARMA) model of the time series is a powerful tool for analysis. An oscillation is represented by a weakly damped conjugate complex pole pair in $1/D(q)$. Compared with other tools for frequency analysis such as a Fourier transform, an AR (or ARMA) model does not suffer from leakage effects due to a discretization of the frequency range. Rather, AR (or ARMA) models, in principle, allow one to determine frequencies and amplitudes with arbitrary accuracy. In practice, AR (or ARMA) models are usually preferred if the number of considered resonances is small or a smoothed version of the spectrum is desired because then the order of the models can be chosen reasonably low and the parameters can be estimated accurately.

The time series is thought to be constructed by filtering white noise $v(k)$:

$$y(k) = \frac{1}{D(q)}v(k)\,. \tag{16.36}$$

The difference equation makes the linear parameterization of the AR model obvious

$$y(k) = -d_1 y(k-1) - \ldots - d_m y(k-m) + v(k)\,. \tag{16.37}$$

The prediction error simply becomes

$$e(k) = D(q)y(k)\,. \tag{16.38}$$

By taking the prediction error approach, the parameter estimation is a least squares problem, which can be easily solved. It corresponds to the estimation of an ARX model without numerator parameters, i.e., $B(q) = 1$ (see Sect. 16.5.1 for details). Another common way to estimate an AR model, is to correlate (16.37) with $y(k-\kappa)$. For $\kappa > 0$ this results to

$$\mathrm{corr}_{yy}(\kappa) = -d_1 \mathrm{corr}_{yy}(\kappa - 1) - \ldots - d_m \mathrm{corr}_{yy}(\kappa - m) \tag{16.39}$$

because the previous outputs $y(k-\kappa)$ do not depend on the current noise $v(k)$, i.e., $\mathrm{E}\{y(k-\kappa)v(k)\} = 0$. For $\kappa = 0, -1, \ldots$ additional, increasingly

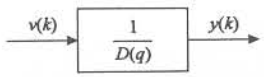

Fig. 16.20. AR model

complex terms like σ^2, $d_1\sigma^2$, etc. appear in (16.39) where σ^2 is the noise variance. Usually only the equations for $\kappa \geq 0$ are used to estimate the noise variance and the parameters d_i. For these different values of κ, (16.39) are called the *Yule-Walker equations*. In a second step, these Yule-Walker equations are solved by least squares; compare this with the COR-LS approach in Sect. 16.5.1.

The Yule-Walker equations are the most widely applied method for AR model estimation. Generally, most time series modeling is based on the estimation of the correlation function. This is a way to eliminate the fictitious unknown white noise signal $v(k)$ from the equations. The correlation function represents the useful information contained in the data in a compressed form.

16.4.2 Moving Average (MA)

For the sake of completeness the moving average time series model (Fig. 16.21) will be mentioned here, too. It has less practical significance in engineering applications because a moving average filter does not allow one to model oscillations with a few parameters like an autoregressive filter does. Furthermore, in contrast to the corresponding deterministic input/output model (the FIR model), the MA model is *nonlinear* in its parameters if the prediction error approach is taken.

The MA model is given by

$$y(k) = C(q)v(k)\,. \tag{16.40}$$

The difference equation makes the nonlinear parameterization of MA model more obvious:

$$y(k) = v(k) + c_1 v(k-1) - \ldots + c_m v(k-m)\,. \tag{16.41}$$

Since $v(k-i)$ are unknown, in order to estimate the parameters c_i, the $v(k-i)$ have to be approximated by a previously built model. Thus, the approximated $\hat{v}(k-i)$, which replace the true but unknown $v(k-i)$ in (16.37), themselves depend on the parameters of a model estimated a priori. This relationship can also be understood by considering the prediction error

$$e(k) = \frac{1}{C(q)}y(k) \tag{16.42}$$

or

$$e(k) = -c_1 e(k-1) - \ldots - c_m e(k-m) + y(k)\,. \tag{16.43}$$

More clever algorithms exist to estimate MA models more efficiently than via a direct minimization of the prediction errors; see [47].

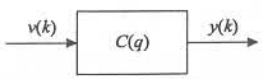

Fig. 16.21. MA model

16.4.3 Autoregressive Moving Average (ARMA)

The combination of an autoregressive part and a moving average part enhances the flexibility of the AR model. The resulting ARMA model shown in Fig. 16.22 is given by

$$y(k) = \frac{C(q)}{D(q)}v(k)\,. \tag{16.44}$$

The difference equation is

$$\begin{aligned} y(k) = &- d_1 y(k-1) - \ldots - d_m y(k-m) \\ &+ v(k) + c_1 v(k-1) - \ldots + c_m v(k-m)\,. \end{aligned} \tag{16.45}$$

The prediction error becomes

$$e(k) = \frac{D(q)}{C(q)}y(k) \tag{16.46}$$

or

$$\begin{aligned} e(k) = &- c_1 e(k-1) - \ldots - c_m e(k-m) \\ &+ y(k) + d_1 y(k-1) + \ldots + d_m y(k-m). \end{aligned} \tag{16.47}$$

For estimation of the nonlinear parameters in the ARMA model, the following approach can be taken; see Sect. 16.5.2. In the first step, a high order AR model is estimated. Then the residuals $e(k)$ in (16.38) are used as an approximation of the white noise $v(k)$. With this approximation the parameters c_i and d_i of an ARMA model are estimated by least squares. Then iteratively the following two steps are repeated: (i) approximation of $v(k)$ by (16.47) with the ARMA model obtained in the previous iteration, (ii) estimation of new a new ARMA model utilizing the approximation for $v(k)$ from step (i). This two-step procedure avoids the direct nonlinear optimization of the parameters and it is sometimes called the Hannan-Rissanen algorithm [47]. Other advanced methods are again based on the correlation idea introduced in Sect. 16.4.1, leading to the innovations algorithm or the Yule-Walker equations for ARMA models. The best (asymptotically efficient, compare Sect. B.7) estimators of AR, MA, and ARMA models are based on the maximum likelihood method [47]. However, this requires the application of a nonlinear optimization technique such as the Levenberg-Marquardt algorithm; see Chap. 4. Good initial parameter values for a local search can be obtained by any of the above strategies.

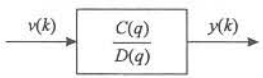

Fig. 16.22. ARMA model

16.5 Models with Output Feedback

This section discusses linear models with output feedback. The models in this class by far are the most widely known and applied. Alternative linear models are described in the subsequent section and either do not employ any feedback or the feedback path is independent of the estimated parameter. In the following subsections on model with output feedback, the model structures are introduced together with appropriate algorithms for parameter estimation. To fully understand these algorithms, knowledge of the linear and nonlinear local optimization techniques discussed in Part I is required.

16.5.1 Autoregressive with Exogenous Input (ARX)

The ARX model is by far the most widely applied linear dynamic model. Usually an ARX model is tried first and only if it does not perform satisfactory are more complex model structures examined. This is not the case because the ARX model would be especially realistic and would match the structure of many real-world processes. Rather, the popularity of the ARX model comes from its easy-to-compute parameters. The parameters can be estimated by a linear least squares technique since the prediction error is linear in the parameters. Consequently, a reliable recursive algorithm for online use, the RLS, exists as well; see Sect. 16.8.1.

The ARX model is depicted in Fig. 16.23, and is described by

$$A(q)y(k) = B(q)u(k) + v(k)\,. \tag{16.48}$$

The optimal ARX predictor is

$$\hat{y}(k|k-1) = B(q)u(k) + (1 - A(q))y(k)\,, \tag{16.49}$$

which can be written as

$$\begin{aligned}\hat{y}(k|k-1) = {} & b_1u(k-1) + \ldots + b_mu(k-m) \\ & - a_1y(k-1) - \ldots - a_my(k-m)\,.\end{aligned} \tag{16.50}$$

assuming $\deg(A) = \deg(B) = m$. Note that contrary to the continuous time process description, in discrete time the numerator and denominator polynomials usually have the same order.

The ARX predictor is stable (it possesses no feedback!) even if the $A(q)$ polynomial and therefore the ARX model is unstable. This fact allows one to model unstable processes with an ARX model. However, the plant has to be stabilized in order to gather data. It is a feature of all equation error models

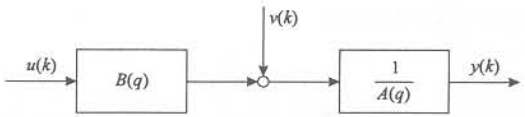

Fig. 16.23. ARX model

that the $A(q)$ polynomials only appear in the numerator of their predictors, and thus the predictors are stable even if $A(q)$ is unstable.

With (16.49) the prediction error of an ARX model is

$$e(k) = A(q)y(k) - B(q)u(k)\,. \tag{16.51}$$

The term $A(q)y(k)$ acts as a whitening filter on the correlated disturbances. The measured output $y(k)$ can be split into two parts: the undisturbed process output $y_u(k)$ and the disturbance $n(k)$, where $y(k) = y_u(k) + n(k)$. Since $n(k) = 1/A(q)v(k)$ with $v(k)$ being white noise $A(q)y(k) = A(q)y_u(k) + v(k)$. Thus, the filter $A(q)$ in (16.51) makes the disturbances and consequently $e(k)$ white.

As can be seen from Fig. 16.23, one characteristic of the ARX model is that the disturbance, i.e., the white noise $v(k)$, is assumed to enter the process before the denominator dynamics $A(q)$. This fact can be expressed in another way by saying that the ARX model has a noise model of $1/A(q)$. So the noise is assumed to have denominator dynamics identical to those of the process. This assumption may be justified if the disturbance enters the process early, although even in this case the disturbance would certainly pass through some part of the numerator dynamics $B(q)$ as well. However, most often this assumption will be violated in practice. Disturbances at the process output, as assumed in an OE model in Sect. 16.5.4, are much more common.

Figure 16.24 shows three different configurations of the ARX model. Note that all three configurations represent the same ARX model, but they suggest a different interpretation. The true process polynomials are denoted as $B(q)$ and $A(q)$, while the model polynomials are denoted as $\hat{B}(q)$ and $\hat{A}(q)$.

Figure 16.24a represents the most common configuration. The prediction error $e(k)$ for an ARX model is called *equation error* because it is the difference in the equation $e(k) = \hat{A}(q)y(k) - \hat{B}(q)u(k)$; see (16.31). The term "equation error" stresses the fact that it is *not* the difference between the process output $y(k)$ and $\hat{B}(q)/\hat{A}(q)u(k)$, which is called the *output error*; see also Sect. 16.5.4. Considering Fig. 16.24a, it is obvious that if the model equals the true process, i.e., $\hat{B}(q) = B(q)$ and $\hat{A}(q) = A(q)$, the equation error $e(k) = \hat{A}(q)n(k) = A(q)n(k)$. Thus, if the assumption made by the ARX model, namely that the disturbance is white noise filtered through $1/A(q)$, is true, then the equation error $e(k)$ is white noise since $n(k) = 1/A(q)v(k)$. For each model structure the prediction errors have to be white if all assumptions made are valid because then all information is exploited by the model.

Figure 16.24b depicts another configuration of the ARX model based on the predictor equation. With the ARX predictor in (16.49) the same equation error as in Fig. 16.24a results. Figure 16.24b can schematically represent any linear model by implementing the corresponding optimal predictor.

Figure 16.24c relates the ARX model to the OE model; see Sect. 16.5.4. This representation makes clear that the equation error $e(k)$ is a filtered version of the output error $e_{\mathrm{OE}}(k)$. Note that $e_{\mathrm{OE}}(k)$ and $\hat{y}_{\mathrm{OE}}(k)$, respectively,

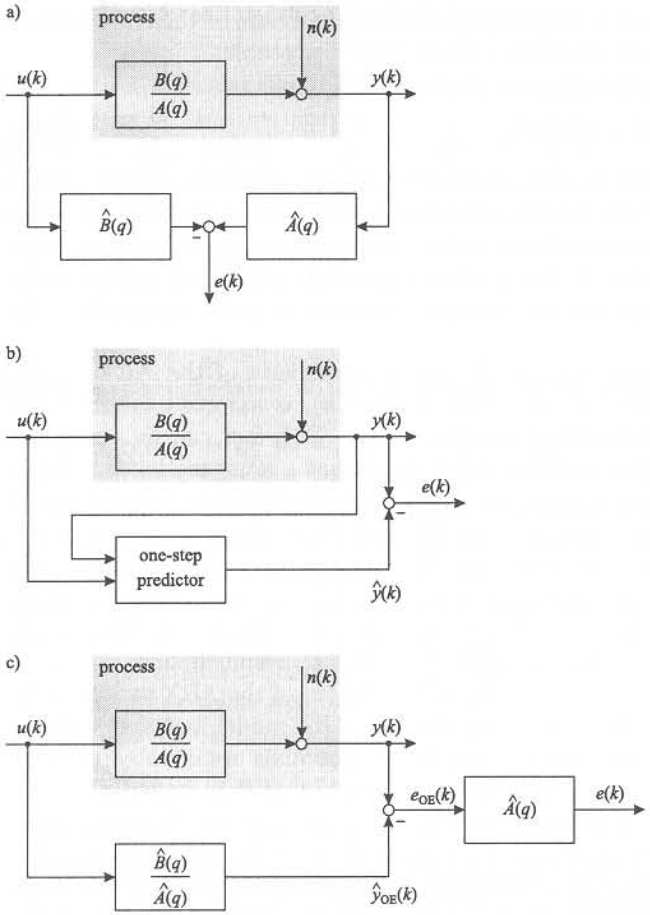

Fig. 16.24. Different representation schemes for the ARX model: a) equation error configuration, b) predictor configuration, c) pseudo-parallel configuration with filtering of the error signal [81]. All configurations realize the same ARX model

denote the output error and the output of an OE model; thus they are different from the prediction error $e(k)$ and the predicted output $\hat{y}(k)$ of an ARX model. Figure 16.24a, b and c represent just different perspectives of the same ARX model, and shall help us to better understand the relationships between the different model structures.

Least Squares (LS). The reason for the popularity of the ARX model is that its parameters can be estimated by a linear least squares (LS) technique. For N available data samples the ARX model can be written in the following matrix/vector form with $N-m$ equations for $k = m+1, \ldots, N$ where $\underline{\hat{y}}$ is the vector of model outputs while $\underline{y}$ is the vector of process outputs that are the desired model outputs:

$$\underline{\hat{y}} = \underline{X}\,\underline{\theta} \tag{16.52}$$

with

$$\underline{\hat{y}} = \begin{bmatrix} \hat{y}(m+1) \\ \hat{y}(m+2) \\ \vdots \\ \hat{y}(N) \end{bmatrix}, \quad \underline{y} = \begin{bmatrix} y(m+1) \\ y(m+1) \\ \vdots \\ y(N) \end{bmatrix}, \quad \underline{\theta} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \\ a_1 \\ \vdots \\ a_m \end{bmatrix}, \tag{16.53}$$

$$\underline{X} = \begin{bmatrix} -y(m) & \cdots & -y(1) & u(m) & \cdots & u(1) \\ -y(m+1) & \cdots & -y(2) & u(m+1) & \cdots & u(2) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y(N-1) & \cdots & -y(N-m) & u(N-1) & \cdots & u(N-m) \end{bmatrix}. \tag{16.54}$$

If the quadratic loss function in (16.32) is minimized, the optimal parameters of the ARX model can be computed by LS as (see Chap. 3)

$$\underline{\hat{\theta}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}. \tag{16.55}$$

For an computation of the estimate in (16.55) the matrix $\underline{X}^{\mathrm{T}}\underline{X}$ has to be non-singular. This is the case if the input $u(k)$ is persistently exciting. The big advantage of the ARX model is its linear-in-the-parameters structure. All features of linear optimization techniques apply, such as a fast one-shot solution that yields the global minimum of the loss function. The main drawback of the ARX model is that its noise model $1/A(q)$ is unrealistic. Additive output noise is much more common. The difficulties arising from this fact are discussed next. For more details concerning the least squares solution refer to Chap. 3.

Consistency Problem. The ARX model and a more realistic process description are compared in Fig. 16.25. Because often the real process is not disturbed, as assumed by the ARX model, some difficulties can be expected. Indeed it can be shown that if the process does not meet the noise assumption made by the ARX model, the parameters are estimated *biased* and *non-consistent*. A bias means that the parameters systematically deviate from their optimal values, i.e., the parameters are systematically over- or underestimated. Non-consistency means that this bias does not even approach zero as the number of data samples N goes to infinity; see Sect. B.7 for more details on the bias and consistency definitions.

Even worse, the errorbars calculated from the estimate of the covariance matrix of the parameter estimate (see Chap. 3) may indicate that the estimate is quite accurate even if the bias is very large [81]. The reason for this undesirable behavior is that the derivations of many theorems about the LS in Chap. 3 assume a deterministic regression matrix $\underline{X}$. However, as can be seen in (16.54), the regression matrix $\underline{X}$ contains measured process outputs

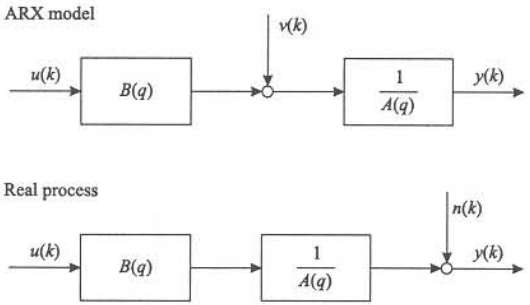

Fig. 16.25. An ARX model assumes a noise model $1/A(q)$, while more realistically a process is disturbed at the output by a noise $n(k)$, which can be white noise $v(k)$ or colored noise, e.g., $n(k) = C(q)/D(q)v(k)$

$y(k)$ that are non-deterministic owing to the disturbances. Thus, the covariance matrix cannot be calculated by (3.34) and consequently the errorbar cannot be derived as shown in Sect. 3.1.2.

Because consistency is probably the most important property of any estimator, several strategies have been developed to avoid the non-consistent estimation for an ARX model. The idea of most of these approaches is to retain the linear-in-the-parameters property of the ARX model since this is its greatest advantage over other model structures.

Next, two such strategies are presented. The first strategy offers an alternative to the prediction error method, and the parameters are estimated with the help of instrumental variables. The idea of the second method is to work with correlation functions of the measured signals instead of the signals themselves.

Another alternative is to choose more general model structures such as ARMAX or OE that are nonlinear in their parameters and to develop algorithms that allow one to estimate the nonlinear parameters by the repeated application of a linear least squares technique. These approaches are discussed in the sections on the corresponding model structures.

Instrumental Variables (IV) Method. A very popular and simple remedy against the consistency problem of the conventional ARX model estimation is the *instrumental variables (IV)* method. It is an alternative to the prediction error methods. The starting point is the difference $\underline{e}$ of the process output $\underline{y}$ and the ARX model output $\underline{\hat{y}}$ for all data samples in matrix/vector form (see (16.52))

$$\underline{e} = \underline{y} - \underline{\hat{y}} = \underline{y} - \underline{X}\,\underline{\theta}\,. \tag{16.56}$$

The least squares estimate that results from a minimization of the sum of squared prediction errors ($\underline{e}^{\mathrm{T}}\underline{e} \longrightarrow \min$) is

$$\underline{\hat{\theta}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{16.57}$$

The idea of the IV method is to multiply (16.56) with a matrix $\underline{Z}$ that has the same dimension as the regression matrix $\underline{X}$. The columns of $\underline{Z}$ are called *instrumental variables* and are chosen by the user to be uncorrelated with the noise, and therefore if all information is exploited they are also uncorrelated with $\underline{e}$. This means that $\underline{Z}^{\mathrm{T}}\underline{e} = \underline{0}$ because each row in $\underline{Z}^{\mathrm{T}}$ is orthogonal to $\underline{e}$ (since they are uncorrelated and $\underline{e}$ has zero mean). Multiplying (16.56) with $\underline{Z}^{\mathrm{T}}$ from the left yields

$$\underline{0} = \underline{Z}^{\mathrm{T}}\underline{y} - \underline{Z}^{\mathrm{T}}\underline{X}\,\underline{\theta} \tag{16.58}$$

and consequently

$$\underline{Z}^{\mathrm{T}}\underline{y} = \underline{Z}^{\mathrm{T}}\underline{X}\,\underline{\theta}\,. \tag{16.59}$$

If $\underline{Z}^{\mathrm{T}}\underline{X}$ is non-singular, which is the case for persistent excitation and a proper choice of $\underline{Z}$, the IV estimate becomes

$$\underline{\hat{\theta}} = \left(\underline{Z}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{Z}^{\mathrm{T}}\underline{y}\,. \tag{16.60}$$

Obviously, the IV estimate is equivalent to the LS estimate if $\underline{Z}^{\mathrm{T}} = \underline{X}^{\mathrm{T}}$. Note, however, that the columns in $\underline{X}$ cannot be used as instrumental variables since the columns containing $y(k-i)$ regressors are disturbed by noise. Thus, $\underline{X}$ is correlated with $\underline{e}$, i.e., $\underline{X}^{\mathrm{T}}\underline{e} \neq \underline{0}$.

If the instrumental variables in $\underline{Z}$ are uncorrelated with the noise the IV estimate is *consistent*. Although all choices of $\underline{Z}$ that fulfill this requirement lead to a consistent estimate, the variance of the estimate depends strongly on $\underline{Z}$. Recall that the parameter variance is proportional to $(\underline{Z}^{\mathrm{T}}\underline{X})^{-1}$; see Sect. 3.1.1. Thus, the variance error is the smaller the higher is the correlation between the instrumental variables in $\underline{Z}$ and the regressors in $\underline{X}$.

Now, the question arises, how to choose $\underline{Z}$? The answer is that the instrumental variables should be highly correlated with the regressors (columns in $\underline{X}$) in order to make the variance error small. For an easier understanding of a suitable choice of $\underline{Z}$ it is convenient to reconsider the ARX regression matrix in (16.54):

$$\underline{X} = \begin{bmatrix} -y(m) & \cdots & -y(1) & u(m) & \cdots & u(1) \\ -y(m+1) & \cdots & -y(2) & u(m+1) & \cdots & u(2) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y(N-1) & \cdots & -y(N-m) & u(N-1) & \cdots & u(N-m) \end{bmatrix}. \tag{16.61}$$

The second half of $\underline{X}$ consists of delayed input signals, which are undisturbed. Consequently, the best instrumental variables for these regressors are the regressors themselves. However, for the first half of $\underline{X}$ the $y(k-i)$ regressors cannot be used in $\underline{Z}$ because the uncorrelation conditions have to be met. Good instrumental variables for the $y(k-i)$ would be an undisturbed version of these regressors. They can be approximated by filtering $u(k)$ through a process model. Thus, the following four-step algorithm can be proposed:

1. Estimate an ARX model from the data $\{\underline{u}(k), \underline{y}(k)\}$ by

$$\underline{\hat{\theta}}_{\mathrm{ARX}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{16.62}$$

2. Simulate this model:

$$y_u(k) = \frac{\hat{B}(q)}{\hat{A}(q)}u(k) \tag{16.63}$$

where $\hat{B}(q)$ and $\hat{A}(q)$ are determined by $\underline{\hat{\theta}}_{\mathrm{ARX}}$.

3. Construct the following instrumental variables:

$$\underline{Z} = \begin{bmatrix} -y_u(m) & \cdots & -y_u(1) & u(m) & \cdots & u(1) \\ -y_u(m+1) & \cdots & -y_u(2) & u(m+1) & \cdots & u(2) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_u(N-1) & \cdots & -y_u(N-m) & u(N-1) & \cdots & u(N-m) \end{bmatrix}.$$

4. Estimate the parameters with the IV method by

$$\underline{\hat{\theta}}_{\mathrm{IV}} = \left(\underline{Z}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{Z}^{\mathrm{T}}\underline{y}\,. \tag{16.64}$$

Because the ARX model parameters estimated in the first step are biased the ARX model may not be a good model of the process. Nevertheless, the simulated process output $y_u(k)$ can be expected to be reasonably close to the measured process output $y(k)$ so that the correlation between $\underline{Z}$ and $\underline{X}$ is high. The IV method can be further improved by repeating Steps 2–4. Then in each iteration for the simulation in Step 2 the IV estimated model from the previous Step 4 can be applied. This procedure converges very fast and experience teaches that more than two or three iterations are not worth any effort.

Ljung proposes performing the following additional five steps after going through Steps 1–4 [233].

5. Compute the residuals:

$$e_{\mathrm{IV}}(k) = \hat{A}(q)y(k) - \hat{B}(q)u(k) \tag{16.65}$$

where $\hat{B}(q)$ and $\hat{A}(q)$ are determined by $\underline{\hat{\theta}}_{\mathrm{IV}}$.

6. Estimate an AR time series model for the residuals to extract the remaining information from $e_{\mathrm{IV}}(k)$. The AR filter acts as a whitening filter, i.e., it is supposed to decorrelate the residuals. Remember that the residuals should be as close to white noise as possible since then the process output is fully explained by the model besides the unpredictable part of the noise. The dynamic order of the AR time series model is chosen as $2m$ (or $n_a + n_b$ if $n_a = \deg(A)$ and $n_b = \deg(B)$ are not identical). Thus, the following relationship is postulated:

$$e_{\mathrm{IV}}(k) = \frac{1}{L(q)}v(k) \qquad \text{or} \quad L(q)e_{\mathrm{IV}}(k) = v(k) \tag{16.66}$$

with the white noise $v(k)$. Refer to Sect. 16.4.1 for a more detailed description of the estimation of AR time series models.

7. Filter the instruments calculated in Step 3 with the filter $\hat{L}(q)$ estimated in Step 6:

$$y_M^L(k) = L(q)y_u(k) \qquad \text{and} \quad u^L(k) = L(q)u(k)\,. \tag{16.67}$$

Filter the process output $y^L(k) = L(q)y(k)$ and the regressors (columns in $\underline{X}$) denoted as $\underline{X}^L$.

8. Construct the following instrumental variables: $\underline{Z}^L =$

$$\begin{bmatrix} -y_M^L(m) & \cdots & -y_M^L(1) & u^L(m) & \cdots & u^L(1) \\ -y_M^L(m+1) & \cdots & -y_M^L(2) & u^L(m+1) & \cdots & u^L(2) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_M^L(N-1) & \cdots & -y_M^L(N-m) & u^L(N-1) & \cdots & u^L(N-m) \end{bmatrix}. \tag{16.68}$$

9. Estimate the parameters with the IV method by

$$\underline{\hat{\theta}}_{\mathrm{IV}}^L = \left(\left(\underline{Z}^L\right)^{\mathrm{T}}\underline{X}^L\right)^{-1}\left(\underline{Z}^L\right)^{\mathrm{T}}\underline{y}^L\,. \tag{16.69}$$

Note that the instrumental variables introduced above are *model dependent*, i.e., they are calculated on the basis of the actual model; see (16.63). A simpler (but less effective) approach is to use *model independent* instruments. This avoids the first LS estimation step, which computes a first model to generate the instruments. A typical choice for model independent instrumental variables is

$$\underline{z} = [u(k-1) \;\cdots\; u(k-2m)]^{\mathrm{T}}\,. \tag{16.70}$$

For more details about the IV method, the optimal choice of the instrumental variables, and its mathematical relationship to the prediction error method, refer to [233, 360].

Correlation Functions Least Squares (COR-LS). The correlation function least squares (COR-LS) method proposed in [172] avoids the consistency problem by the following idea. Instead of computing the LS estimate directly from the signals $u(k)$ and $y(k)$ as is done in (16.52), the COR-LS method calculates correlation functions first. The starting point is the linear difference equation

$$y(k) = b_1u(k-1)+\ldots+b_mu(k-m)-a_1y(k-1)-\ldots-a_my(k-m)\,. \tag{16.71}$$

This equation is multiplied by the term $u(k-\kappa)$:

$$\begin{aligned} u(k-\kappa)y(k) = {} & b_1u(k-\kappa)u(k-1) + \ldots + b_mu(k-\kappa)u(k-m) \\ & -a_1u(k-\kappa)y(k-1) - \ldots - a_mu(k-\kappa)y(k-m). \end{aligned} \tag{16.72}$$

Now the sum over $N-\kappa$ data samples, e.g., $k = \kappa+1, \ldots, N$, can be calculated in order to generate estimates of correlation functions (see Sect. B.6)

$$\sum_{k=\kappa+1}^{N} u(k-\kappa)y(k) = \tag{16.73}$$

$$b_1 \sum_{k=\kappa+1}^{N} u(k-\kappa)u(k-1) + \ldots + b_m \sum_{k=\kappa+1}^{N} u(k-\kappa)u(k-m)$$

$$-a_1 \sum_{k=\kappa+1}^{N} u(k-\kappa)y(k-1) - \ldots - a_m \sum_{k=\kappa+1}^{N} u(k-\kappa)y(k-m).$$

Thus, this equation can be written as

$$\begin{aligned}\mathrm{corr}_{uy}(\kappa) = {} & b_1\ \mathrm{corr}_{uu}(\kappa-1) + \ldots + b_m\,\mathrm{corr}_{uu}(\kappa-m) \\ & - a_1\ \mathrm{corr}_{uy}(\kappa-1) - \ldots - a_m\,\mathrm{corr}_{uy}(\kappa-m).\end{aligned} \tag{16.74}$$

Obviously, (16.74) possesses the same form as (16.71), only the signals $u(k)$ and $y(k)$ are replaced by the auto-correlation functions $\mathrm{corr}_{uu}(\kappa)$ and the cross-correlation functions $\mathrm{corr}_{uy}(\kappa)$. Thus, the least squares estimation in (16.55) can be applied on the level of correlation functions as well by changing the the regression matrix and the output vector to $\underline{X}_{\mathrm{corr}} =$

$$\begin{bmatrix} \mathrm{corr}_{uu}(0) & \cdots & \mathrm{corr}_{uu}(1-m) & -\mathrm{corr}_{uy}(0) & \cdots & -\mathrm{corr}_{uy}(1-m) \\ \mathrm{corr}_{uu}(1) & \cdots & \mathrm{corr}_{uu}(2-m) & -\mathrm{corr}_{uy}(1) & \cdots & -\mathrm{corr}_{uy}(2-m) \\ \vdots & & \vdots & \vdots & & \vdots \\ \mathrm{corr}_{uu}(l-1) & \cdots & \mathrm{corr}_{uu}(l-m) & -\mathrm{corr}_{uy}(l-1) & \cdots & -\mathrm{corr}_{uy}(l-m) \end{bmatrix} \tag{16.75}$$

$$\underline{y}_{\mathrm{corr}} = \begin{bmatrix} \mathrm{corr}_{uy}(1) \\ \mathrm{corr}_{uy}(2) \\ \vdots \\ \mathrm{corr}_{uy}(l) \end{bmatrix}, \tag{16.76}$$

where it is assumed that the correlation functions are used from $\kappa = 1-m$ to $\kappa = l$. Note that the number of terms in the sum that approximates the correlation functions decreases as the time shift $|\kappa|$ increases. Therefore, as l in (16.75) increases, the effect of the correlation decreases; in the extreme case the sum contains only one term. Nevertheless, the full range of possible correlation functions can be utilized, and then the number of rows in $\underline{X}_{\mathrm{corr}}$ becomes $N-1$.

Figures 16.26 and 16.27 show examples for the auto- and cross-correlation functions. The simulated process follows the first order difference equation $y(k) = 0.1u(k-1) + 0.9y(k-1)$. In Fig. 16.26 the input signal is white. Therefore the auto-correlation function $\mathrm{corr}_{uu}(\kappa)$ is a Dirac impulse. In Fig. 16.27 the input signal is low-pass filtered and therefore the auto-correlation function is wider. The cross-correlation functions of both figures look similar; the one in Fig. 16.27 is smoother owing to the lower frequency input signal. For non-positive time shifts the cross-correlations are about zero since $y(k)$ (for a causal process) does not depend on the future inputs

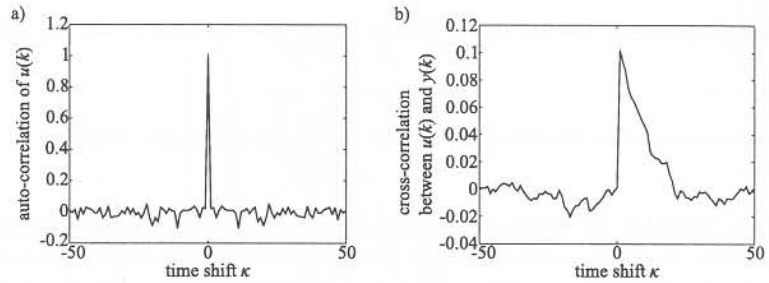

Fig. 16.26. a) Auto-correlation and b) cross-correlation functions for a white input sequence $\{u(k)\}$ of 1000 data samples and time shifts κ between -50 and 50. The process used is $y(k) = 0.1u(k-1) + 0.9y(k-1)$

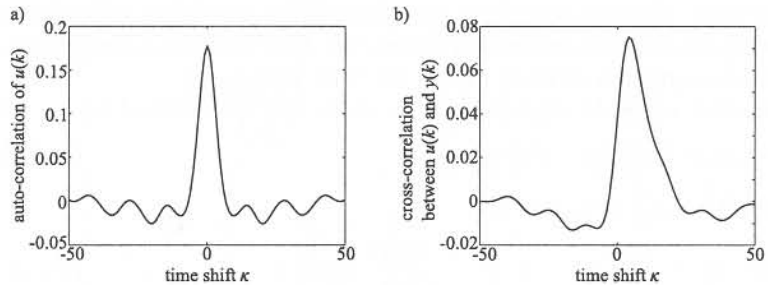

Fig. 16.27. a) Auto-correlation and b) cross-correlation functions for a low frequency input sequence $\{u(k)\}$ of 1000 data samples and time shifts κ between -50 and 50. The process used is $y(k) = 0.1u(k-1) + 0.9y(k-1)$. Compare with Fig. 16.26

$u(k-\kappa)$, $\kappa \leq 0$. Thus, the cross-correlation function in Fig. 16.26 jumps at $\kappa = 1$ on its maximum value and decays as the correlation between $y(k)$ and inputs κ time steps in the past decreases. As the number of samples increases, the random fluctuations in the correlation functions decrease. For an infinite number of data samples the cross-correlation function in Fig. 16.26 would be identical to the impulse response of the process. This makes the relationship between the signals and the correlation functions obvious.

The drawback of the COR-LS method is the higher computational effort. However, the correlation functions can possibly be exploited for estimation of the dynamic process order as well; see Sects. 16.9 and B.6. So the additional effort may be justified. The advantage of this COR-LS compared with the conventional ARX method is that the regression matrix X consists of virtually deterministic values since the correlation with $u(k-\kappa)$ eliminates the noise in $y(k)$ because $u(k)$ is uncorrelated with $n(k)$. Consequently, the COR-LS method yields *consistent* estimates. Experience shows that the COR-LS method is well capable of attenuating noise, and it is especially powerful if the noise spectrum lies in the same frequency range as the process dynamics

and thus filtering cannot be applied to separate the disturbance for the signal [172].

16.5.2 Autoregressive Moving Average with Exogenous Input (ARMAX)

The ARMAX model is probably the second most popular linear model after the ARX model. Some controller designs such as minimum variance control are based on an ARMAX model and exploit the information in the noise model [176]. Compared with the ARX, the ARMAX model is more flexible because it possesses an extended noise model. Although with this extension the ARMAX model becomes nonlinear in its parameters, quite efficient multistage linear least squares algorithms are available for parameter estimation, circumventing nonlinear optimization techniques. Furthermore, a straightforward recursive algorithm (RELS) exists; see Sect. 16.8.1.

The ARMAX model is depicted in Fig. 16.28, and is described by

$$A(q)y(k) = B(q)u(k) + C(q)v(k)\,. \tag{16.77}$$

The optimal ARMAX predictor is

$$\hat{y}(k|k-1) = \frac{B(q)}{C(q)}u(k) + \left(1 - \frac{A(q)}{C(q)}\right)y(k)\,. \tag{16.78}$$

The ARMAX predictor is stable even if the $A(q)$ polynomial and therefore the ARMAX model is unstable. However, the polynomial $C(q)$ is required to be stable.

With (16.78) the prediction error of an ARMAX model is

$$e(k) = \frac{A(q)}{C(q)}y(k) - \frac{B(q)}{C(q)}u(k)\,. \tag{16.79}$$

Studying the above equations reveals that the ARMAX model is an extended ARX model owing to the introduction of the filter $C(q)$. If $C(q) = 1$ the ARMAX simplifies to the ARX model. Owing to the additional filter $C(q)$ the ARMAX model is very flexible. For example, with $C(q) = A(q)$ the ARMAX model can imitate an OE model; see Sect. 16.5.4.

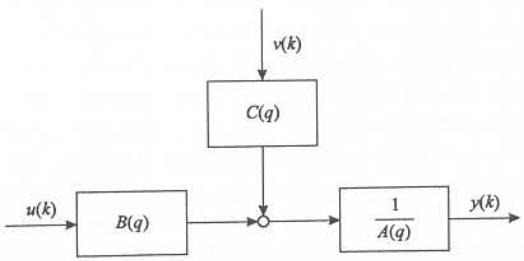

Fig. 16.28. ARMAX model

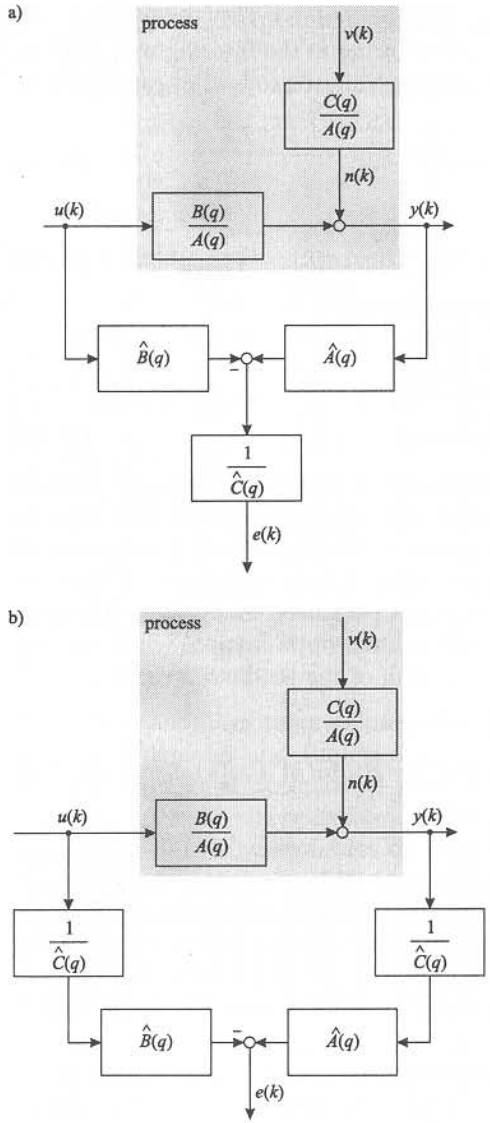

Fig. 16.29. ARMAX model in equation error configuration

Because the noise filter $C(q)/A(q)$ contains the model denominator dynamics, the ARMAX model belongs to the class of equation error models. This is also obvious from the ARMAX configuration depicted in Fig. 16.29; see Fig. 16.24. If $\hat{A}(q) = A(q)$, $\hat{B}(q) = B(q)$, and $\hat{C}(q) = C(q)$ the residuals $e(k)$ are white. Thus $\hat{A}(q)/\hat{C}(q)$ acts as a whitening filter.

Estimation of ARMAX Models. The prediction error (16.79) of an ARMAX model is nonlinear in its parameters owing to the filtering with $1/C(q)$. However, the prediction error can be expressed in the following *pseudo-linear* form:

$$C(q)e(k) = A(q)y(k) - B(q)u(k)\,, \tag{16.80}$$

which can be written as

$$e(k) = A(q)y(k) - B(q)u(k) + (1 - C(q))\,e(k)\,. \tag{16.81}$$

This results in the following difference equation:

$$\begin{aligned} e(k) = {} & a_1 y(k-1) + \ldots + a_m y(k-m) \\ & - b_1 u(k-1) - \ldots - b_m u(k-m) \\ & - c_1 e(k-1) - \ldots - c_m e(k-m)\,. \end{aligned} \tag{16.82}$$

The above equation formally represents a linear regression. However, because the $e(k-i)$ that estimate the unknown $v(k-i)$ (compare the first point in Sect. 16.3.3) are not measured but have to be calculated from previous residuals, the corresponding parameters are called to be pseudo-linear. Therefore, (16.81) and (16.82) allow two approaches for parameter estimation. The most straightforward approach is based on nonlinear optimization, while the second strategy exploits the pseudo-linear form of the prediction error.

Nonlinear optimization of the ARMAX model parameters.

1. Estimate an ARX model $A(q)y(k) = B(q)u(k) + v(k)$ from the data $\{\underline{u}(k), \underline{y}(k)\}$ by

$$\hat{\underline{\theta}}_{\mathrm{ARX}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{16.83}$$

2. Optimize the ARMAX model parameters with a nonlinear optimization technique, e.g., with a nonlinear least squares method such as the Levenberg-Marquardt algorithm; see Chap. 3. The ARX model parameters obtained in Step 1 can be used as initial values for the a_i and b_i parameters.
   An efficient nonlinear optimization requires the computation of the gradients. The gradient of the squared prediction error $e^2(k) = (y(k) - \hat{y}(k))^2$ is $-2\,e(k)\,\partial\hat{y}(k)/\partial\theta$. Thus, the gradients of the predicted model output have to be calculated. It is convenient to multiply (16.78) by $C(q)$ in order to get rid of the denominators:

$$C(q)\hat{y}(k|k-1) = B(q)u(k) + (C(q) - A(q))y(k)\,. \tag{16.84}$$

   Differentiation of (16.84) with respect to a_i yields [233]

$$C(q)\frac{\partial\hat{y}(k|k-1)}{\partial a_i} = -y(k-i)\,, \tag{16.85}$$

   which leads to

$$\frac{\partial\hat{y}(k|k-1)}{\partial a_i} = -\frac{1}{C(q)}y(k-i)\,. \tag{16.86}$$

   Differentiation of (16.84) with respect to b_i yields [233]

$$C(q)\frac{\partial\hat{y}(k|k-1)}{\partial b_i} = u(k-i)\,, \tag{16.87}$$

   which leads to

$$\frac{\partial\hat{y}(k|k-1)}{\partial b_i} = \frac{1}{C(q)}u(k-i)\,. \tag{16.88}$$

   Differentiation of (16.84) with respect to c_i yields [233]

$$\hat{y}(k-i|k-i-1) + C(q)\frac{\partial\hat{y}(k|k-1)}{\partial c_i} = y(k-i)\,, \tag{16.89}$$

   which leads to

$$\frac{\partial\hat{y}(k|k-1)}{\partial c_i} = \frac{1}{C(q)}\left(y(k-i) - \hat{y}(k-i|k-i-1)\right)\,. \tag{16.90}$$

   Thus, the gradient can be easily computed by filtering the regressors $-y(k-i)$, $u(k-i)$, and $e(k-i) = y(k-i) - \hat{y}(k-i|k-i-1)$ through the filter $1/C(q)$. The residuals $e(k)$ approach the white noise $v(k)$ as the algorithm converges.

The drawbacks of the nonlinear optimization approach are the high computational demand and the existence of local optima. The danger of convergence to a local optimum is reduced, however, if the initial parameter values are close to the optimal ones. In [233] experiences are reported that the globally optimal parameters of ARMAX models are "usually found without too much problem", while for OE and BJ models "convergence to false local minima is not uncommon".

Multistage least squares for ARMAX model estimation. This algorithm is sometimes called extended least squares (ELS)[3].

1. Estimate an ARX model $A(q)y(k) = B(q)u(k) + v(k)$ from the data $\{\underline{u}(k), \underline{y}(k)\}$ by

$$\hat{\underline{\theta}}_{\mathrm{ARX}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{16.91}$$

2. Calculate the prediction errors of this ARX model:

$$e_{\mathrm{ARX}}(k) = \hat{A}(q)y(k) - \hat{B}(q)u(k)\,, \tag{16.92}$$

   where $\hat{B}(q)$ and $\hat{A}(q)$ are determined by $\hat{\underline{\theta}}_{\mathrm{ARX}}$.
3. Estimate the ARMAX model parameters a_i, b_i, and c_i from (16.82) with LS by approximating the ARMAX residuals as $e(k-i) \approx e_{\mathrm{ARX}}(k-i)$.

[3] Often ELS denotes the *recursive* version of this algorithm. Here, for the sake of clarity the recursive algorithm is named RELS; see Sect. 16.8.3.

Steps 2–3 of the ELS algorithm can be iterated until convergence is reached. Then, of course, in Step 2 the residuals from the previously (in Step 3) estimated ARMAX model are used and in Step 3 the ARMAX residuals are approximated by the residuals of the ARMAX model from Step 2. The ARMAX prediction error should approach white noise as all information is going to be exploited by the model and then $e(k)$ approaches the white noise $v(k)$. Note that the prediction error of the ARMAX model can be obtained by filtering either the ARX model error or $u(k)$ and $y(k)$ in (16.92) with $1/C(q)$ as shown in Fig. 16.29. The speed of convergence with the ELS algorithm may be somewhat faster than with nonlinear optimization. However, the (mild) local optima problem can, of course, not be solved.

In [233] an ARX model of higher order than m is proposed for Step 1 to obtain a better approximation of the white noise $v(k)$. Ideally, $e(k)$ converges to $v(k)$.

The ARMAX model can be extended to the ARIMAX model, where "I" stands for integration. The noise model is extended by an integrator to $C(q)/(1-q^{-1})A(q)$. This allows for drifts in the output signal. Alternatively, the data can be filtered with the inverse integrator $1-q^{-1}$ (see Sects. 16.3.4 and 16.7.5), or the noise model can be made flexible enough that the integrator is found automatically [233].

16.5.3 Autoregressive Autoregressive with Exogenous Input (ARARX)

The ARARX model can be seen as the counterpart of the ARMAX model. While the disturbance is filtered through an MA filter $C(q)v(k)$ for the ARMAX model, it goes through an AR filter $1/D(q)v(k)$ for the ARARX model. The ARARX model is not as common as the ARX or ARMAX model since the additional model complexity often does not pay off.

The ARARX model is depicted in Fig. 16.30 and is described by

$$A(q)y(k) = B(q)u(k) + \frac{1}{D(q)}v(k)\,. \tag{16.93}$$

The optimal ARARX predictor is

$$\hat{y}(k|k-1) = D(q)B(q)u(k) + (1-D(q)A(q))\,y(k)\,. \tag{16.94}$$

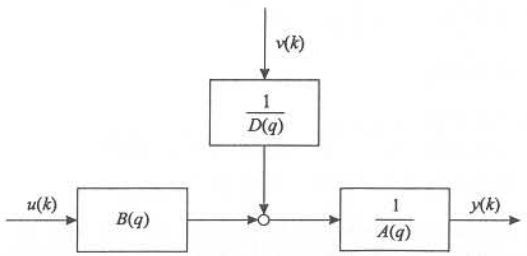

Fig. 16.30. ARARX model

The ARARX predictor is stable even if the $A(q)$ or $D(q)$ polynomials and therefore the ARARX model itself are unstable.

With (16.94) the prediction error of an ARARX model is

$$e(k) = D(q)A(q)y(k) - D(q)B(q)u(k)\,. \tag{16.95}$$

Studying the above equations reveals that the ARMAX model, like the ARMAX model, is an extended ARX model owing to the introduction of the filter $D(q)$. If $D(q)=1$ the ARARX simplifies to the ARX model. Owing to the additional filter $D(q)$ the ARARX model is more flexible than the ARX model. However, because $D(q)$ extends the denominator dynamics compared with the extension of numerator dynamics in the ARMAX model, the denominator dynamics $A(q)$ cannot be (partly) canceled in the noise model.

Because the noise filter $1/D(q)A(q)$ contains the model denominator dynamics, the ARARX model belongs to the class of equation error models. This is also obvious from the ARARX configuration depicted in Fig. 16.31; see Fig. 16.24. If $\hat{A}(q)=A(q)$, $\hat{B}(q)=B(q)$, and $\hat{D}(q)=D(q)$ the residuals $e(k)$ are white. Thus $\hat{D}(q)\hat{A}(q)$ acts as a whitening filter.

The parameters of the ARARX model can be estimated either by a nonlinear optimization technique or by a repeated least squares and filtering approach [172].

Nonlinear optimization of the ARARX model parameters.

1. Estimate an ARX model $A(q)y(k) = B(q)u(k) + v(k)$ from the data $\{\underline{u}(k), \underline{y}(k)\}$ by

$$\hat{\underline{\theta}}_{\mathrm{ARX}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{16.96}$$

2. Optimize the ARARX model parameters with a nonlinear optimization technique. The ARX model parameters obtained in Step 1 can be used as initial values for the a_i and b_i parameters. The gradients of the model's prediction (16.94) can be computed as follows.
Differentiation of (16.94) with respect to a_i yields [233]

$$\frac{\partial\hat{y}(k|k-1)}{\partial a_i} = -D(q)y(k-i)\,. \tag{16.97}$$

Differentiation of (16.94) with respect to b_i yields [233]

$$\frac{\partial\hat{y}(k|k-1)}{\partial b_i} = D(q)u(k-i)\,. \tag{16.98}$$

Differentiation of (16.94) with respect to d_i yields [233]

$$\frac{\partial\hat{y}(k|k-1)}{\partial d_i} = B(q)u(k-i) - A(q)y(k-i) = -e_{\mathrm{ARX}}(k-i). \tag{16.99}$$

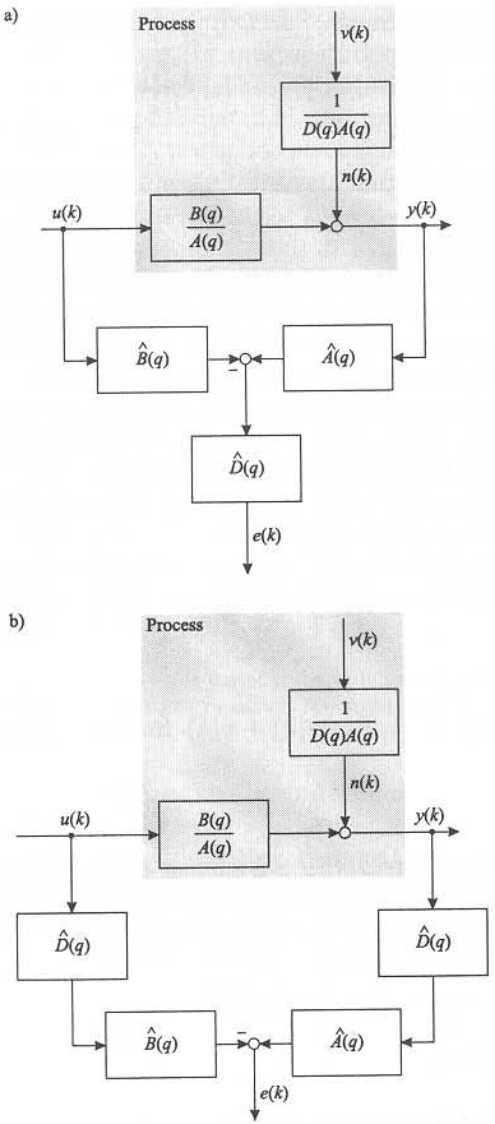

Fig. 16.31. ARARX model in equation error configuration

Repeated least squares and filtering for ARARX model estimation (generalized least squares (GLS)).

1. Estimate an ARX model $A(q)y(k) = B(q)u(k) + v(k)$ from the data $\{\underline{u}(k), \underline{y}(k)\}$ by

$$\hat{\underline{\theta}}_{\mathrm{ARX}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{16.100}$$

2. Calculate the prediction errors of this ARX model:

$$e_{\mathrm{ARX}}(k) = \hat{A}(q)y(k) - \hat{B}(q)u(k)\,, \tag{16.101}$$

   where $\hat{B}(q)$ and $\hat{A}(q)$ are determined by $\hat{\underline{\theta}}_{\mathrm{ARX}}$.
3. Estimate the d_i parameters of the following AR model by least squares (see Sect. 16.4.1)

$$e_{\mathrm{ARX}}(k) = \frac{1}{D(q)}v(k)\,. \tag{16.102}$$

   Compare (16.93) and Fig. 16.31a for a motivation of this AR model. The prediction error $e(k)$ in Fig. 16.31a becomes white, i.e., equal to $v(k)$, if $e_{\mathrm{ARX}}(k)$ in (16.101) is filtered through $D(q)$.
4. Filter the input $u(k)$ and process output $y(k)$ through the estimated filter: $\hat{D}(q)$

$$u^D(k) = \hat{D}(q)u(k) \qquad \text{and} \quad y^D(k) = \hat{D}(q)y(k)\,. \tag{16.103}$$

5. Estimate the ARARX model parameters a_i and b_i by an ARX model estimation with the filtered input $u^D(k)$ and output $y^D(k)$; see Fig. 16.31b.

Steps 3–5 of the GLS algorithm can be iterated until convergence is reached.

16.5.4 Output Error (OE)

Together with the ARX and ARMAX model the OE model is the most widely used structure. It is the simplest representative of the output error model class. The noise is assumed to disturb the process additively at the output, not somewhere inside the process as is assumed for the equation error models. Output error models are often more realistic models of reality, and thus they often perform better than equation error models. However, because the noise models do not include the process denominator dynamics $1/A(q)$, all output error models are nonlinear in their parameters and consequently they are harder to estimate.

The OE model is depicted in Fig. 16.32, and is described by

$$y(k) = \frac{B(q)}{F(q)}u(k) + v(k)\,. \tag{16.104}$$

It is standard in linear system identification literature to denote the denominator of process models belonging to the output error class as $F(q)$,

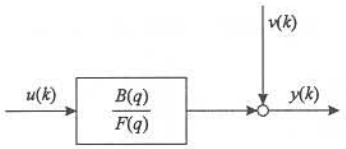

Fig. 16.32. OE model

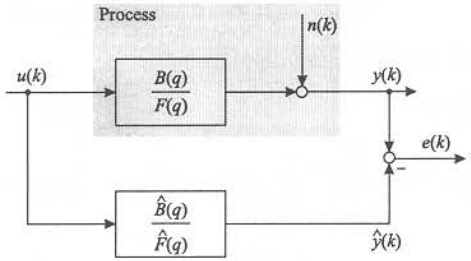

Fig. 16.33. OE model in parallel to the process

while the denominators of equation error models such as ARX, ARMAX, and ARARX are denoted as $A(q)$ [233]. Of course, these are just notational conventions to emphasize the different noise assumptions; a model denoted as $B(q)/A(q)$ can be exactly identical to a model denoted as $B(q)/F(q)$.

The optimal OE predictor is in fact a simulator because it does not make any use of the measurable process output $y(k)$:

$$\hat{y}(k|k-1) = \hat{y}(k) = \frac{B(q)}{F(q)}u(k)\,. \tag{16.105}$$

Note that the notation "$|k-1$" can be discarded for the OE model because the optimal prediction is not based on previous process outputs.

Furthermore, note that the OE predictor is unstable if the $F(q)$ polynomial is unstable. Therefore the OE model cannot be used for modeling unstable processes. The same holds for all other models belonging to the class of output error models.

With (16.105) the prediction error of an OE model is

$$e(k) = y(k) - \frac{B(q)}{F(q)}u(k)\,. \tag{16.106}$$

Figure 16.33 depicts the OE model in parallel to the process. The prediction error of the OE model is the difference between the process output and the simulated model output. The disturbance $n(k)$ is assumed to be white.

Figure 16.34 relates the residuals of an OE model to the residuals of an ARX model. Owing to the equation error configuration of the ARX model (see Fig. 16.24c) the ARX model residuals can be interpreted as filtered OE residuals:

$$e_{\mathrm{ARX}}(k) = F(q)\,e_{\mathrm{OE}}(k)\,. \tag{16.107}$$

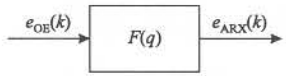

Fig. 16.34. Relationship between ARX model residuals and OE model residuals. The ARX model residuals can be obtained by filtering the OE model residuals through $F(q)$

Assume that $\hat{F}(q) = F(q)$ and $\hat{B}(q) = B(q)$. If the process noise is white ($n(k) = v(k)$) then $e_{\mathrm{OE}}(k) = v(k)$ is white as well, while $e_{\mathrm{ARX}}(k) = F(q)v(k)$ is correlated. If, however, the process noise is correlated such that $n(k) = 1/F(q)v(k)$ then $e_{\mathrm{OE}}(k) = 1/F(q)v(k)$ is correlated, while $e_{\mathrm{ARX}}(k) = v(k)$ is white. This relationship allows an output error parameter estimation based on repeated linear least squares and filtering, although the parameters are nonlinear. In the above discussion $F(q)$ and $\hat{F}(q)$ can be replaced by $A(q)$ and $\hat{A}(q)$ if the argumentation is starting from the ARX model point of view.

It is helpful to illuminate why the predicted output of an OE model is nonlinear in its parameters (see (16.105))

$$\begin{aligned}\hat{y}(k) = {} & b_1u(k-1) + \ldots + b_mu(k-m) \\ & - f_1\hat{y}(k-1) - \ldots - f_m\hat{y}(k-m)\,.\end{aligned} \tag{16.108}$$

Compared with the ARX model, the measured output in (16.50) is replaced with the predicted (or the simulated, which is the same for OE) output in (16.108). Here lies the reason for the nonlinearity of the parameters in (16.108). The predicted model outputs $\hat{y}(k-i)$ depend themselves on the model parameters. So in the terms $f_i\,\hat{y}(k-i)$ both factors depend on model parameters, which results in a nonlinear dependency. To overcome these difficulties one may be tempted to approximate in (16.108) the model outputs $\hat{y}(k-i)$ by the measured process outputs $y(k-i)$. Then the OE model simplifies to the ARX model, which is indeed linear in its parameters.

The parameters of the OE model can be estimated either by a nonlinear optimization technique or by a repeated least squares and filtering approach exploiting the relationship to the ARX model [193].

Nonlinear optimization of the OE model parameters.

1. Estimate an ARX model $F(q)y(k) = B(q)u(k) + v(k)$ from the data $\{\underline{u}(k), \underline{y}(k)\}$ by

$$\underline{\hat{\theta}}_{\mathrm{ARX}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,, \tag{16.109}$$

   where the parameters in $\underline{\hat{\theta}}$ are now denoted as f_i and b_i instead of a_i and b_i.
2. Optimize the ARARX model parameters with a nonlinear optimization technique. The ARX model parameters obtained in Step 1 can be used as initial values for the f_i and b_i parameters. The gradients of the model's

prediction (16.105) can be computed as follows. First, (16.105) is written in the following form:

$$F(q)\hat{y}(k) = B(q)u(k)\,. \tag{16.110}$$

Differentiation of (16.110) with respect to b_i yields

$$F(q)\frac{\partial\hat{y}(k)}{\partial b_i} = u(k-i)\,, \tag{16.111}$$

which leads to

$$\frac{\partial\hat{y}(k)}{\partial b_i} = \frac{1}{F(q)}u(k-i)\,. \tag{16.112}$$

Differentiation of (16.110) with respect to f_i yields

$$\hat{y}(k-i) + F(q)\frac{\partial\hat{y}(k)}{\partial f_i} = 0\,, \tag{16.113}$$

which leads to

$$\frac{\partial\hat{y}(k)}{\partial f_i} = -\frac{1}{F(q)}\hat{y}(k-i)\,. \tag{16.114}$$

Repeated least squares and filtering for OE model estimation.

1. Estimate an ARX model $F(q)y(k) = B(q)u(k) + v(k)$ from the data $\{\underline{u}(k), \underline{y}(k)\}$ by

$$\underline{\hat{\theta}}_{\mathrm{ARX}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,, \tag{16.115}$$

   where the parameters in $\underline{\hat{\theta}}$ are now denoted as f_i and b_i instead of a_i and b_i.
2. Filter the input $u(k)$ and process output $y(k)$ through the estimated filter $\hat{F}(q)$:

$$u^F(k) = \frac{1}{\hat{F}(q)}u(k) \quad \text{and} \quad y^F(k) = \frac{1}{\hat{F}(q)}y(k)\,. \tag{16.116}$$

3. Estimate the OE model parameters f_i and b_i by an ARX model estimation with the filtered input $u^F(k)$ and output $y^F(k)$; see Fig. 16.34.

Steps 2–3 of this algorithm can be iterated until convergence is reached. This algorithm exploits the relationship of the ARX and OE model prediction errors. It becomes intuitively clear from another point of view as well. In Sects. 16.3.4 and 16.7.4 it is shown that a noise model and filtering with the inverse noise model are equivalent. Thus, the ARX noise model $1/A(q)$ has the same effect as filtering of the data through $A(q)$. The filtering with $1/F(q)$ in (16.116) tries to compensate this effect, leading to the noise model 1, which corresponds to an OE model.

16.5.5 Box-Jenkins (BJ)

The Box-Jenkins (BJ) model belongs to the class of output error models. It is an OE model with additional degrees of freedom for the noise model. While the OE model assumes an additive white disturbance at the process output, the BJ allows any colored disturbance. It may be generated by filtering white noise through a linear filter with arbitrary numerator and denominator.

The BJ model is depicted in Fig. 16.35, and is described by

$$y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}v(k)\,. \tag{16.117}$$

Thus, the BJ model can be seen as the output error class counterpart of the ARARMAX model, which belongs to the equation error class. For the equation error models the special case $D(q) = 1$ corresponds to the ARMAX model and the special case $C(q) = 1$ corresponds to the ARARX model. These special cases for the BJ model do not have specific names. For $C(q) = D(q)$ the BJ simplifies to the OE model. Note that the BJ model can imitate all equation error models if the order of the noise model is high enough. Then the denominator of the noise model $D(q)$ may (but of course does not have to) include the process denominator dynamics $F(q)$.

Of all linear models discussed so far the BJ model is the most general and flexible. It allows one to estimate separate transfer functions with arbitrary numerators and denominators from the input to the output and from the disturbance to the output. However, on the other hand the flexibility of the BJ model requires one to estimate a large number of parameters. For most applications this is either not worth the price or not possible owing to data set that are too small and noisy. Consequently, the BJ model is seldom applied in practice.

The optimal BJ predictor is

$$\hat{y}(k|k-1) = \frac{B(q)D(q)}{F(q)C(q)}u(k) + \frac{C(q)-D(q)}{C(q)}y(k)\,. \tag{16.118}$$

Note that the notation "$|k-1$" cannot be discarded as for the OE model because the optimal prediction of a BJ model utilizes previous process outputs in order to extract the information contained in the correlated disturbances $n(k) = C(q)/D(q)v(k)$.

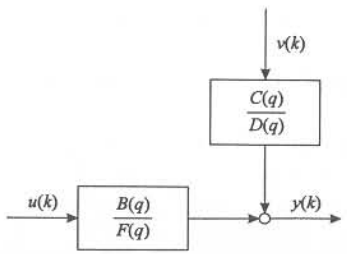

Fig. 16.35. Box-Jenkins model

With (16.118) the prediction error of a BJ model is

$$e(k) = \frac{D(q)}{C(q)}y(k) - \frac{B(q)D(q)}{F(q)C(q)}u(k)\,. \tag{16.119}$$

Typically, a BJ model is estimated by nonlinear optimization, where first an ARX model is estimated in order to determine the initial parameter values for b_i and f_i. The gradients of the model's prediction (16.118) can be computed as follows. First, (16.118) is written in the following form:

$$F(q)C(q)\hat{y}(k|k-1) = B(q)D(q)u(k) + F(q)(C(q) - D(q))y(k). \tag{16.120}$$

Differentiation of (16.120) with respect to b_i yields

$$F(q)C(q)\frac{\partial \hat{y}(k|k-1)}{\partial b_i} = D(q)u(k-i)\,, \tag{16.121}$$

which leads to

$$\frac{\partial \hat{y}(k|k-1)}{\partial b_i} = \frac{D(q)}{F(q)C(q)}u(k-i)\,. \tag{16.122}$$

Differentiation of (16.120) with respect to c_i yields

$$F(q)\left(\hat{y}(k-i|k-i-1) + C(q)\frac{\partial \hat{y}(k|k-1)}{\partial c_i}\right) = F(q)y(k-i), \tag{16.123}$$

which leads to

$$\frac{\partial \hat{y}(k|k-1)}{\partial c_i} = \frac{1}{C(q)}\left(y(k-i) - \hat{y}(k-i|k-i-1)\right)\,. \tag{16.124}$$

Differentiation of (16.120) with respect to d_i yields

$$F(q)C(q)\frac{\partial \hat{y}(k|k-1)}{\partial d_i} = B(q)u(k-i) - F(q)y(k-i)\,, \tag{16.125}$$

which leads to

$$\frac{\partial \hat{y}(k|k-1)}{\partial d_i} = \frac{B(q)}{F(q)C(q)}u(k-i) - \frac{1}{C(q)}y(k-i)\,. \tag{16.126}$$

Differentiation of (16.120) with respect to f_i yields

$$C(q)\left(\hat{y}(k-i|k-i-1) + F(q)\frac{\partial \hat{y}(k|k-1)}{\partial f_i}\right) = -D(q)y(k-i), \tag{16.127}$$

which leads to

$$\frac{\partial \hat{y}(k|k-1)}{\partial f_i} = -\frac{1}{F(q)}\left(\frac{D(q)}{C(q)}y(k-i) + \hat{y}(k-i|k-i-1)\right)\,. \tag{16.128}$$

16.5.6 State Space Models

Instead of input/output state space models can also be considered. A state space OE model takes the following form:

$$\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{b}\,u(k) \tag{16.129a}$$

$$y(k) = \underline{c}^{\mathrm{T}}\underline{x}(k) + v(k)\,. \tag{16.129b}$$

The easiest and most straightforward way to obtain a state space model from data is to estimate an input/output model, e.g., an OE model (see Sect. 16.5.4),

$$y(k) = \frac{B(q)}{F(q)}u(k) + v(k) \tag{16.130}$$

and use these parameters in a canonical state space form, e.g.,

$$\underline{x}(k+1) = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \\ -f_m & -f_{m-1} & \cdots & -f_1 \end{bmatrix}\underline{x}(k) + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}u(k) \tag{16.131a}$$

$$y(k) = \begin{bmatrix} b_m & b_{m-1} & \cdots & b_1 \end{bmatrix}\underline{x}(k) + v(k)\,. \tag{16.131b}$$

The major advantage of a state space representation is that prior knowledge from first principles can be incorporated in the form equations and can be utilized to pre-structure the model [186]. Furthermore, the number of regressors is usually smaller in state space models than in input/output models. For a system of mth order a state space model possesses $m+1$ regressors ($x_1(k), \ldots, x_m(k)$ and $u(k)$) while an input/output model requires $2m$ regressors ($u(k-1), \ldots, u(k-m)$ and $y(k-1), \ldots, y(k-m)$). The smaller number of regressors is not very important for linear systems. However, for nonlinear models this is a significant advantage since the number of regressors corresponds to the input dimensionality; see Sect. 17.1. Finally, for processes with multiple inputs and outputs the state space representation is well suited. For a direct identification of state space models the following cases can be distinguished:

- If all states are measurable the parameters in $\underline{A}$, $\underline{b}$, and $\underline{c}$ can be estimated by a linear optimization technique. Unfortunately, the true states of the process will seldom lead to a canonical state space realization as in (16.131a) and (16.131b). Thus, without any incorporation of prior knowledge all entries of the system matrix and vectors must be assumed to be non-zero. For an mth order model with such a full parameterization $m^2 + 2m$ parameters have to be estimated. Usually this can only be done if a regularization technique is applied to reduce the variance error of the model; see Sects. 7.5 and 3.1.4.