Chapter 3

Parametric Identification of Dynamical Systems

This chapter deals with the identification of dynamical systems described by mathematical models. In particular, it is focused on the system identification of *parametric* models for linear time-invariant (LTI) systems. For a thorough treatment, see the book [1].

3.1 Introduction

The aim of *system identification* is to find mathematical models of dynamical systems on the basis of experimental data.

The main elements of system identification techniques are:

- a *data set*, usually a sequence of input/output data of length N. Denoting by $u(t)$ and $y(t)$ the input and the output of a system, one has

$$\mathcal{Z}^N = \{u(1), y(1), \ldots, u(N), y(N)\}; \tag{3.1}$$

- a class of *candidate models* with a given structure, containing the model that must be identified (e.g., transfer functions of fixed order);
- a *rule* to choose the "best" model inside the previously chosen family;
- a set of *model validation* techniques.

Such elements are briefly described below.

[1] Ljung L., *System Identification: Theory for the User (2nd ed.)*, Prentice Hall: Upper Saddle River, NJ, 1999.

Data set

Output data can be obtained by measuring system signals, while inputs are usually chosen by the user. In this course, only input/output data in the time domain will be considered, like $\mathcal{Z}^N$ in (3.1). In several applications, the input of the system can be freely chosen. Notice that, choosing different input signals may reflect in different quality of the identified model. Other aspects regard the length of the experiment (i.e. the data set length N), the sampling time, etc. All these aspects consist in the so-called *experiment design*, which has the purpose of obtaining a data set as informative as possible. This constitutes the first step of the identification process. Unfortunately, it is not always possible to freely choose the input, for instance when the operating conditions of the system are fixed, or when it is inserted in a control loop. Since the data are measured at a given sampling time, the identification procedure will be performed in *discrete time*.

Model class of a given structure

The choice of the model class represents the most critical step of the identification procedure, since it heavily affects the final result. A model can be viewed as a map from input to output, i.e.

$$\hat{y}(t;\theta) = g\left(\theta, \mathcal{Z}^{t-1}\right)$$

where $\hat{y}(t;\theta)$ represents the predicted output at time t, $\mathcal{Z}^{t-1}$ contains the data gathered till time $t-1$, and θ is a set of parameters which defines a specific model inside the class.

So, the model class is defined by:

$$\mathcal{M} = \{g\left(\theta, \cdot\right) : \theta \in \Theta \subseteq \mathcal{S}\}$$

where Θ is a set representing the a-priori knowledge on the model, and $\mathcal{S}$ is the space of elements θ.

In general, models can be divided in two types:

- *Parametric* models, where θ is a parameter vector of finite dimension (e.g., the coefficient of the polynomials of a transfer function $G(z)$);
- *Non-parametric* models, where θ describes some basic characteristics of the model, like the impulse or the step response (in time domain), or the frequency response (in the frequency domain) of an LTI model.

Regarding the parametric identification, models whose parameters derive from physical laws (or other kind of information) are called *gray-box* models, while models

whose parameters just represent the input/output behavior without any other physical meaning are called *black-box* models. Other model classifications depend on their complexity (linear/nonlinear, time-invariant/time-varying, lumped/distributed parameters, etc.).

In general, a fundamental requirement for a model class is its ability to represent the features of interest for the considered application.
In this course, we consider linear time-invariant (LTI) models, described by rational transfer functions.

Choice of the "best" model

Once gathered data and chosen the model class $\mathcal{M}$, the identification problem consists in finding the element $\hat{\theta}^*$ which minimizes a generic cost function $J(\theta)$ inside the given model class, i.e.

$$\hat{\theta}^* = \arg\min_{\theta \in \Theta} J(\theta)$$

So, an *optimization problem* must be solved, whose complexity depends on the complexity of the model class $\mathcal{M}$ and of the cost function $J(\theta)$.

Model validation

Once the model $g(\hat{\theta}^*, \cdot)$ has been identified, it must be *validated* to assess its quality and its capacity to satisfy the requirements. Several validation procedures can be used. The weight given to each procedure depends on the use one has to do with the identified model. For instance, a model used to design a control system will have different requirements with respect to one used to predict the output of a system.

A model may fail the validation step for many reasons:

- the optimization algorithm may not converge (e.g., due to numerical problems);
- wrong choice of the cost function $J(\theta)$;
- model class not appropriate;
- data set not enough informative.

As previously stated, often the critical step relies on the choice of the model class (e.g., for LTI models, the choice of the model order). Practically, the identification procedure is usually iterated until the validation criteria are satisfied, by changing at any iteration one or more elements of the problem, see Fig. 3.1.

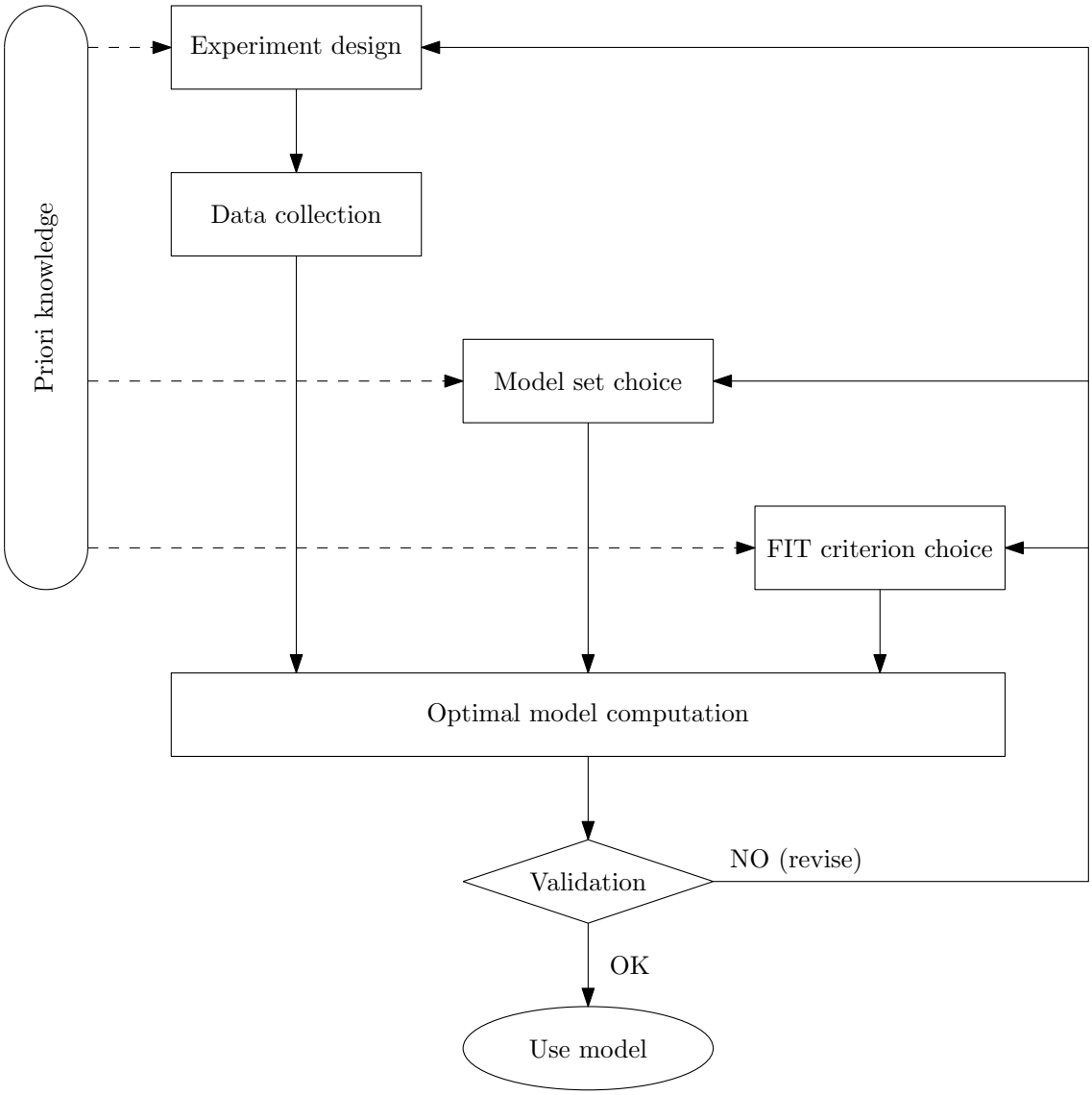

Figure 3.1: Sketch of the identification procedure.

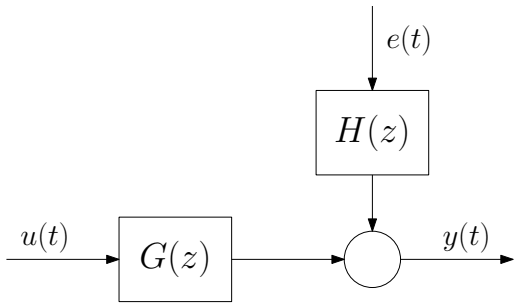

Figure 3.2: General class for identifying LTI models.

3.2 Linear time-invariant models

We refer to *parametric identification* when the model class is represented by a structure which describes the system input/output behavior and the identification error with a finite number of numerical values (parameters). A widely spread model class is that related to LTI models of the form

$$y(t) = G(z)u(t) + H(z)e(t) \tag{3.2}$$

where $u(t)$ is the deterministic input, $e(t)$ is a white stochastic process, $G(z)$ and $H(z)$ are suitable transfer functions[2].

The term $G(z)u(t)$ represents the deterministic component of the model and describes the relation between the input $u(t)$ and the output $y(t)$. The term $H(z)e(t)$ models the stochastic component which affects the output $y(t)$; it is usually due to measurement noise or unmodeled dynamics of the deterministic component.
The model (3.2) is fully defined if the following terms are known:

- the impulse response sequences $\{g(k)\}_{k=0}^{\infty}$, $\{h(k)\}_{k=0}^{\infty}$ of stable transfer functions $G(z)$ and $H(z)$, respectively:

$$G(z) = \sum_{k=0}^{\infty} g(k)z^{-k}, \qquad H(z) = \sum_{k=0}^{\infty} h(k)z^{-k};$$

- the probability density function $f_{\boldsymbol{e}}(e)$ of the stochastic process $e(t)$.

A possible alternative consists in finding a finite number of parameters θ to describe the model. In this way, the LTI model class includes all the models of the form

$$y(t) = G(z,\theta)u(t) + H(z,\theta)e(t) \tag{3.3}$$

[2]With a slight abuse of notation, we denote by z both the complex variable of transfer functions and the time shift operator such that, for a generic discrete-time sequence $u(k)$, $z\,u(k) = u(k+1)$, and $z^{-1}\,u(k) = u(k-1)$.

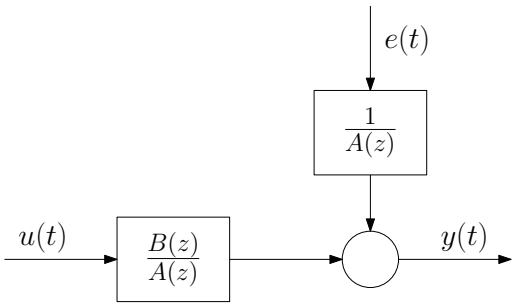

Figure 3.3: Block scheme of an ARX model.

where $e(t)$ is a white stochastic process with probability density $f_{\boldsymbol{e}}(e,\theta)$, where $\theta \in \Theta \subseteq \mathbb{R}^d$ represents the parameter vector. Generally, the stochastic process $e(t)$ is represented by only two parameters: the mean (usually zero) and the variance $\sigma^2_{\boldsymbol{e}}$. If the noise is Gaussian, it corresponds to the knowledge of the density $f_{\boldsymbol{e}}(e)$.

The easiest way to parameterize a model is to choose θ as the vector containing the coefficients of the numerator and denominator polynomials of the rational transfer functions $G(z)$ and $H(z)$. In the following, the widely used LTI model structures are reported.

3.2.1 ARX Models

Suppose that the stochastic process $y(t)$ is generated by the following linear regression:

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t) \quad (3.4)$$

Let us define the polynomials

$$A(z) = 1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}, \qquad B(z) = b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b} \quad (3.5)$$

So, one may write

$$A(z)y(t) = B(z)u(t) + e(t).$$

Then, the transfer functions of the LTI model (3.3) are defined as

$$G(z,\theta) = \frac{B(z)}{A(z)}, \qquad H(z,\theta) = \frac{1}{A(z)}.$$

A natural choice of the parameter vector is

$$\theta = [a_1 \ \ldots \ a_{n_a} \ b_1 \ \ldots \ b_{n_b}]^T \quad (3.6)$$

where the dimension of the parameter vector is $d = n_a + n_b$.

A model with these characteristics is named ARX (Auto Regressive with eXogenous input). Notice that, from a physical viewpoint, a strong assumption is enforced:

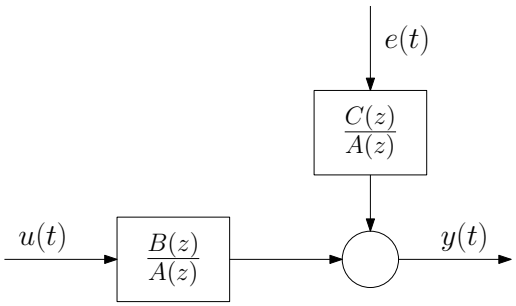

Figure 3.4: Block scheme of an ARMAX model.

the noise $e(t)$ is filtered by a transfer function having the same denominator (i.e. the same poles) of the deterministic transfer function $G(z)$.

In the special case when $n_a = 0$, such model is called FIR (Finite Impulse Response), since the transfer function $G(z)$ becomes

$$G(z) = b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}$$

and then $g(k) = 0$, for all $k > n_b$.

3.2.2 ARMAX Models

A richer model structure than ARX is the so-called ARMAX (Auto Regressive Moving Average with eXogenous input). In this case, $y(t)$ is generated through the following regression:

$$\begin{aligned} y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) \\ + e(t) + c_1 e(t-1) + \cdots + c_{n_c} e(t-n_c) \end{aligned}$$

Let us define the polynomial

$$C(z) = 1 + c_1 z^{-1} + \cdots + c_{n_c} z^{-n_c}$$

and assume that $A(z)$ and $B(z)$ are the same polynomials defined in (3.5). One has,

$$A(z)y(t) = B(z)u(t) + C(z)e(t).$$

So, the transfer functions in (3.3) are defined as

$$G(z,\theta) = \frac{B(z)}{A(z)}, \qquad H(z,\theta) = \frac{C(z)}{A(z)},$$

The parameter vector of an ARMAX model is defined as

$$\theta = [a_1 \ \ldots \ a_{n_a} \ b_1 \ \ldots \ b_{n_b} \ c_1 \ \ldots \ c_{n_c}]^T$$

where $d = n_a + n_b + n_c$. Notice that polynomials $A(z)$ and $C(z)$ are supposed to be monic[3]. Moreover, like for ARX models, the denominator of $G(z)$ and $H(z)$ is the same.

[3]A polynomial is called *monic* if its leading coefficient (i.e., the nonzero coefficient of highest degree) is 1.

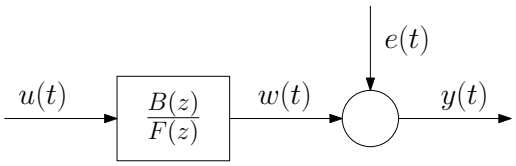

Figure 3.5: Block scheme of an OE model.

3.2.3 OE Models

If the assumption that $G(z)$ and $H(z)$ have the same denominator (like in ARX and ARMAX models) is not realistic, we must use different model classes. A possible way is to assume that the output $y(t)$ is generated by:

$$y(t) = w(t) + e(t)$$

where

$$w(t) + f_1 w(t-1) + \cdots + f_{n_f} w(t-n_a) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b).$$

By defining the polynomial $F(z)$ as

$$F(z) = 1 + f_1 z^{-1} + \cdots + f_{n_f} z^{-n_f}$$

one has

$$F(z)w(t) = B(z)u(t)$$

and the LTI transfer functions become

$$G(z,\theta) = \frac{B(z)}{F(z)}, \qquad H(z,\theta) = 1.$$

This model is called OE (Output Error), since it assumes that the additive noise $e(t)$ directly affects the output. The parameter vector turns out to be:

$$\theta = [b_1 \ \ldots \ b_{n_b} \ f_1 \ \ldots \ f_{n_f}]^T$$

where $d = n_b + n_f$.

3.2.4 BJ Models

A further degree of freedom can be added assuming that the two transfer functions $G(z)$ and $H(z)$ are completely independent, that is

$$G(z,\theta) = \frac{B(z)}{F(z)}, \qquad H(z,\theta) = \frac{C(z)}{D(z)},$$

where

$$D(z) = 1 + d_1 z^{-1} + \cdots + d_{n_d} z^{-n_d}$$

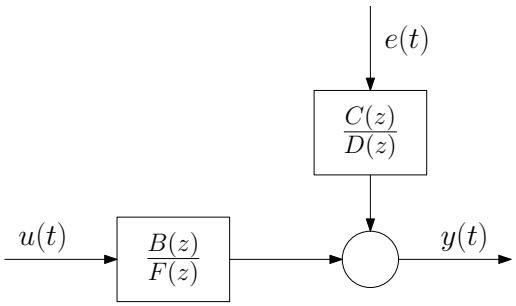

Figure 3.6: Block scheme of a Box-Jenkins model.

and $B(z)$, $C(z)$, $F(z)$ are the previously defined polynomials. Such models are named BJ (Box-Jenkins). In this case, the parameter vector is

$$\theta = [b_1 \ \dots \ b_{n_b} \ c_1 \ \dots \ c_{n_c} \ d_1 \ \dots \ d_{n_d} \ f_1 \ \dots \ f_{n_f}]^T$$

and its dimension is $d = n_b + n_c + n_d + n_f$.

All the model structures previously described can be viewed as special cases of a general family of models of the form

$$A(z)y(t) = \frac{B(z)}{F(z)}u(t) + \frac{C(z)}{D(z)}e(t).$$

A further parameter to fix is the delay r between the input $u(t)$ and the output $y(t)$. If this information is not a-priori known, it is essential to embed it in the model structure, which becomes

$$A(z)y(t) = \frac{B(z)}{F(z)}z^{-r}u(t) + \frac{C(z)}{D(z)}e(t).$$

From a practical viewpoint, the delay can be embedded in polynomial $B(z)$, setting to zero its first r coefficients. In fact,

$$B(z)z^{-r} = b_1 z^{-r-1} + \cdots + b_{n_b} z^{-r-n_b}.$$

3.3 Criteria to choose the model

Once the model class has been fixed, one has to choose a criteria to select the best model inside it on the basis of the input/output data $\mathcal{Z}^N$. The basic idea is to compare the output of the unknown system $\{y(t)\}$ with that obtained by the model $\{y_m(t)\}$, for the same input sequence $\{u(t)\}$. Now, let us analyze the output of the model $\{y_m(t)\}$. It is composed of a deterministic part y_d and a stochastic one y_s:

$$y_m(t) = \underbrace{G(z,\theta)u(t)}_{y_d(t)} + \underbrace{H(z,\theta)e(t)}_{y_s(t)}.$$

Since $e(t)$ is unknown, it is required to do a *forecast* of the value of $y(t)$ (and in particular of $y_s(t)$) based on the chosen model and on the available data. A natural choice is to set $y_m(t)$ as the *prediction* of the output at time t on the basis of the data $\mathcal{Z}^{t-1}$ gathered till the previous time step, that is

$$y_m(t) = \hat{y}(t|t-1).$$

For instance, if we like to choose $y_m(t)$ on the basis of the minimum mean square error (in the Bayesian setting), one has

$$y_m(t) = \mathbf{E}\left[\boldsymbol{y}(t)|\mathcal{Z}^{t-1}\right] = \int_{-\infty}^{\infty} y f_{\boldsymbol{y}|\mathcal{Z}}(y|\mathcal{Z}^{t-1})dy,$$

whose computation requires the knowledge of the probability density of the process $y(t)$.
In general, given a model $\mathcal{M}(\theta)$, the *prediction error* at time t is defined as:

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t|t-1).$$

The parametric identification methods based on the prediction error usually aim at minimizing a cost function

$$J(\varepsilon(t,\theta)) = V_N(\theta, \mathcal{Z}^N)$$

where the expression on the right emphasizes the dependence of the cost function on both the unknown parameters θ and the data set $\mathcal{Z}^N$. Thus, the optimal value of the parameters is given by:

$$\hat{\theta}_N^* = \arg\min_{\theta\in\Theta} V_N\left(\theta, \mathcal{Z}^N\right). \tag{3.7}$$

In the following, it will be shown how to compute the linear prediction at minimum mean square error (LMMSE) $\hat{y}(t|t-1)$, for the LTI model classes described in Section 3.2:

$$y(t) = G(z)u(t) + H(z)e(t) = G(z)u(t) + v(t) \tag{3.8}$$

where $v(t) = H(z)e(t)$. Since, by construction, $G(z)$ has a delay at least 1, one has:

$$\hat{y}(t|t-1) = G(z)u(t) + \hat{v}(t|t-1). \tag{3.9}$$

Let us suppose that $H(z)$ is an inversely stable filter, i.e. $H^{-1}(z)$ is still stable. This means that both zeros and poles of $H(z)$ strictly lie inside the unit circle.

By (3.8), one has

$$v(t) = y(t) - G(z)u(t). \tag{3.10}$$

Moreover, since $v(t) = H(z)e(t)$, one may write

$$v(t) = \sum_{k=0}^{\infty} h_k e(t-k) = e(t) + \sum_{k=1}^{\infty} h_k e(t-k) \triangleq e(t) + m(t-1)$$

where $m(t-1)$ depends only on data till time $t-1$. Since $e(t)$ is not predictable (and zero mean), the best prediction of v is $\hat{v}(t|t-1) = m(t-1)$. So,

$$\hat{v}(t|t-1) = H(z)e(t) - e(t) = (H(z) - 1)\, e(t).$$

Since $H(z)$ is invertible, $e(t) = H^{-1}(z)v(t)$ and then

$$\hat{v}(t|t-1) = \left(1 - H^{-1}(z)\right) v(t).$$

By substituting in (3.9), and by exploiting (3.10), one has

$$\begin{aligned} \boxed{\hat{y}(t|t-1)} &= G(z)u(t) + \left(1 - H^{-1}(z)\right) v(t) \\ &= G(z)u(t) + \left(1 - H^{-1}(z)\right) (y(t) - G(z)u(t)) \\ &= \boxed{\left(1 - H^{-1}(z)\right) y(t) + H^{-1}(z)G(z)u(t)}. \end{aligned} \tag{3.11}$$

So, the prediction error becomes:

$$\begin{aligned} \boxed{\varepsilon(t,\theta)} &= y(t) - \hat{y}(t|t-1) = H^{-1}(z)y(t) - H^{-1}(z)G(z)u(t) \\ &= \boxed{H^{-1}(z)\left(y(t) - G(z)u(t)\right)}. \end{aligned} \tag{3.12}$$

In the following, the prediction error will be analyzed for the previously introduced model classes.

ARX

Let us consider an ARX model. Its transfer functions are:

$$G(z,\theta) = \frac{B(z)}{A(z)}, \qquad H(z,\theta) = \frac{1}{A(z)},$$

and then,

$$H^{-1}(z) = A(z), \qquad H^{-1}(z)G(z) = B(z).$$

So, the prediction error has the following form:

$$\varepsilon(t,\theta) = A(z)y(t) - B(z)u(t).$$

An important property of ARX models is that the prediction error is a linear function of the parameters. In fact, it is possible to define the *regressor* as

$$\varphi(t) = [-y(t-1) \ \ldots \ -y(t-n_a) \ u(t-1) \ \ldots \ u(t-n_b)]^T.$$

By considering the parameter vector θ defined in (3.6), equation (3.4) can be written as

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

Therefore, the prediction $\hat{y}(t|t-1)$ is

$$\hat{y}(t|t-1) = \varphi^T(t)\theta$$

and the prediction error becomes:

$$\varepsilon(t,\theta) = y(t) - \varphi^T(t)\theta.$$

ARMAX

The transfer functions of an ARMAX model are:

$$G(z,\theta) = \frac{B(z)}{A(z)}, \qquad H(z,\theta) = \frac{C(z)}{A(z)},$$

and then,

$$H^{-1}(z) = \frac{A(z)}{C(z)}, \qquad H^{-1}(z)G(z) = \frac{B(z)}{C(z)}.$$

According to (3.11), the output prediction becomes:

$$\hat{y}(t|t-1) = \left(1 - \frac{A(z)}{C(z)}\right) y(t) + \frac{B(z)}{C(z)} u(t)$$

that is,

$$C(z)\hat{y}(t|t-1) = B(z)u(t) + (C(z) - A(z))y(t). \tag{3.13}$$

Let us sum the term $(1-C(z))\hat{y}(t|t-1)$ two both members of (3.13). One obtains:

$$\begin{aligned}\hat{y}(t|t-1) &= B(z)u(t) + (1-A(z))y(t) + (1-C(z))\underbrace{(\hat{y}(t|t-1) - y(t))}_{-\varepsilon(t,\theta)} \\ &= B(z)u(t) + (1-A(z))y(t) + (C(z)-1)\varepsilon(t,\theta). \end{aligned} \tag{3.14}$$

By the definition of polynomials A, B and C, (3.14) corresponds to the recursion:

$$\begin{aligned}\hat{y}(t|t-1) &= -a_1 y(t-1) - \cdots - a_{n_a} y(t-n_a) + b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) \\ &\quad + c_1 \varepsilon(t-1,\theta) + \cdots + c_{n_c}\varepsilon(t-n_c,\theta) \\ &= \varphi^T(t,\theta)\theta \end{aligned} \tag{3.15}$$

where $\varphi(t,\theta)$ is the *pseudo-regressor* defined as:

$$\varphi(t,\theta) = [-y(t-1)\cdots - y(t-n_a)\ u(t-1)\ldots u(t-n_b)\ \varepsilon(t-1,\theta)\ldots\varepsilon(t-n_c,\theta)]^T.$$

So, the prediction error becomes:

$$\varepsilon(t,\theta) = y(t) - \varphi^T(t,\theta)\theta.$$

One may notice that, the pseudo-regressor depends on the past prediction errors $\varepsilon(t-1,\theta),\ldots,\varepsilon(t-n_c,\theta)$, and then on the unknown parameter vector θ that must be estimated. For this reason, the procedure to compute the prediction error is performed by a *pseudo-linear* regression. In other word, $\varepsilon(t,\theta)$ is a *nonlinear* function of θ.

OE

OE models have the following transfer functions:

$$G(z,\theta) = \frac{B(z)}{F(z)}, \qquad H(z,\theta) = 1,$$

and then,

$$H^{-1}(z) = 1, \qquad H^{-1}(z)G(z) = \frac{B(z)}{F(z)}.$$

So, the output prediction is:

$$\hat{y}(t|t-1) = (1-1)y(t) + \frac{B(z)}{F(z)}u(t) = \frac{B(z)}{F(z)}u(t),$$

Let us define

$$w(t,\theta) \triangleq \frac{B(z)}{F(z)}u(t),$$

one has

$$w(t,\theta) = -f_1 w(t-1,\theta) - \cdots - f_{n_f} w(t-n_f,\theta) + b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b)$$

and then the prediction satisfies the pseudo-linear regression:

$$\begin{aligned} \hat{y}(t|t-1) &= -f_1 w(t-1,\theta) - \cdots - f_{n_f} w(t-n_f,\theta) \\ &\quad + b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) \\ &= \varphi^T(t,\theta)\theta. \end{aligned} \tag{3.16}$$

In this case, the *pseudo-regressor* is:

$$\varphi(t,\theta) = [u(t-1)\ldots u(t-n_b) \ -w(t-1,\theta) - \cdots - w(t-n_f,\theta)]^T.$$

Therefore, the prediction error turns out to be:

$$\varepsilon(t,\theta) = y(t) - \varphi^T(t,\theta)\theta.$$

which is nonlinear in θ.
Notice that $w(t-k,\theta) = \hat{y}(t-k|t-k-1)$ and hence the pseudo-regressor can be computed by the previous predictions of the output.

Remark 3.1. Notice that, in general, the prediction $\hat{y}(t|t-1)$ and the prediction error $\varepsilon(t,\theta)$ which form the pseudo-regressors of ARMAX and OE models, are exact only if the available data at time $t-1$ are infinite, i.e. $\{u(k), y(k)\}_{k=-\infty}^{t-1}$. Of course, one may assume that such data is zero before a given time (e.g., for $t < 0$), but in this case the prediction is only *suboptimal*. An exception regards the ARX model, where prediction depends only on n_a past samples of the output and on n_b past samples of the input.

3.3.1 Choice of the cost function

As previously stated, the parameter vector is obtained by minimizing a cost function $V_N\left(\theta, \mathcal{Z}^N\right)$. A common choice is:

$$V_N\left(\theta, \mathcal{Z}^N\right) = \frac{1}{N}\sum_{t=1}^{N} \ell(L(z)\varepsilon(t,\theta))$$

where

- $\ell(\cdot)$ is a scalar function, usually positive. A standard choice is:

  $$\ell(\varepsilon) = \frac{1}{2}\varepsilon^2$$

  which corresponds to the least-square computation of θ.

- $L(z)$ is a filter on the prediction error, which allows to weight some frequencies, to reduce the high-frequency noise, etc. For LTI models, the introduction of $L(z)$ is equivalent to filter the input/output data:

  $$\begin{aligned} L(z)\varepsilon(t,\theta) &= L(z)H^{-1}(z)(y(t) - G(z)u(t)) \\ &= H^{-1}(z)(\underbrace{L(z)y(t)}_{y_f(t)} - G(z)\underbrace{L(z)u(t)}_{u_f(t)}) \\ &= H^{-1}(z)L(z)(y(t) - G(z)u(t)) \end{aligned}$$

  So, the introduction of $L(z)$ is equivalent to change the transfer function of the stochastic channel from $H(z)$ to $\dfrac{H(z)}{L(z)}$.

3.4 Selection of the optimal model

In this section, the problem of computing the "optimal" parameter vector $\hat{\theta}_N^*$ is addressed. So, once the model class has been chosen and a cost function has been defined, the optimization problem (3.7) must be solved.

Let us start with the easier case which allows a closed-form solution.

3.4.1 Linear regression and least square

As previously reported, the prediction error of an ARX model is

$$\varepsilon(t,\theta) = y(t) - \varphi^T(t)\theta.$$

By choosing the cost function as $\ell(\varepsilon) = \frac{1}{2}\varepsilon^2$ and $L(z) = 1$,

$$V_N\left(\theta, \mathcal{Z}^N\right) = \frac{1}{2N}\sum_{t=1}^{N}(y(t) - \varphi^T(t)\theta)^2$$

one has the least-square criterion

$$\hat{\theta}_N^* = \arg\min_{\theta} V_N\left(\theta, \mathcal{Z}^N\right). \tag{3.17}$$

Notice that the cost function is:

$$V_N\left(\theta, \mathcal{Z}^N\right) = \frac{1}{2N}\left(\sum_{t=1}^{N} y(t)^2 - 2\sum_{t=1}^{N}\theta^T\varphi(t)y(t) + \sum_{t=1}^{N}\theta^T\varphi(t)\varphi^T(t)\theta\right). \tag{3.18}$$

To compute the optimal parameter vector, we enforce the gradient of $V_N\left(\theta, \mathcal{Z}^N\right)$ to be null:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}V_N\left(\theta, \mathcal{Z}^N\right) = \frac{1}{2N}\sum_{t=1}^{N}\left(-2\varphi(t)y(t) + 2\varphi(t)\varphi^T(t)\theta\right)\Big|_{\theta=\hat{\theta}_N^*} = 0,$$

from which it is possible to obtain the so-called *normal equations*:

$$\left(\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right)\theta = \frac{1}{N}\sum_{t=1}^{N}\varphi(t)y(t). \tag{3.19}$$

If the matrix $R(N) \in \mathbb{R}^{d\times d}$ defined as

$$R(N) \triangleq \frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)$$

is invertible, the normal equations (3.19) has a unique solution, known as *least-square parametric estimation*:

$$\hat{\theta}_N^{LS} = \left(\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right)^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)y(t). \tag{3.20}$$

The matrix $R(N) \in \mathbb{R}^{d\times d}$ plays a fundamental role in the parameter estimation of an ARX model. In the following, some properties are reported and discussed:

- $R(N)$ is the Hessian of $V_N\left(\theta, \mathcal{Z}^N\right)$, that is $\frac{\mathrm{d}^2}{\mathrm{d}\theta^2}V_N\left(\theta, \mathcal{Z}^N\right)$. Since the cost function is the sum of squares, then surely $R(N) \geq 0$. In particular, if $R(N) > 0$ then $R(N)$ is invertible, and the solution of (3.17) is unique. Otherwise, the optimization problem has infinite solutions (*non-identifiability* condition).

- $R(N)$ is the sum of N matrices $\varphi(t)\varphi^T(t)$ of rank 1. Therefore, it holds $\operatorname{rank} R(N) \leq N$. If $d = n_a + n_b$ is the number of parameters to be estimated, in order that $R(N)$ be invertible, it needs $N \geq d$, i.e. the number of input/output data must be greater or equal to the number of parameters. Notice that this condition is only necessary, but not sufficient.

- Let $P(N) = NR(N)$. Matrix $P(N)$ can be computed by the following recursion:

$$P(t) = P(t-1) + \varphi(t)\varphi^T(t).$$

Of course, increasing the number of terms $\varphi(t)\varphi^T(t)$, it is more likely that $R(N)$ becomes invertible. Of course, this heavily depends on the choice of the input (e.g., try to compute $P(t)$ for a FIR of order 2 with a constant input...what happens?).

- Elements of $R(N)$ are of the form $\frac{1}{N}\sum_{t=1}^{N} y(t-i)y(t-j)$, $\frac{1}{N}\sum_{t=1}^{N} y(t-i)u(t-j)$ and $\frac{1}{N}\sum_{t=1}^{N} u(t-i)u(t-j)$, i.e. they are the sample estimates of the covariance functions $R_y(\tau)$, $R_{yu}(\tau)$ and $R_u(\tau)$, respectively.

Let us now analyze the consistency of the least-square estimation. Assume that data are generated by a linear regression with parameter vector θ_0:

$$y(t) = \varphi^T(t)\theta_0 + e_0(t).$$

For the moment, no assumption on the signal e_0 is enforced. By (3.20),

$$\begin{aligned}\hat{\theta}_N^{LS} &= \left(\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right)^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)(\varphi^T(t)\theta_0 + e_0(t)) \\ &= R(N)^{-1}\left(R(N)\theta_0 + \frac{1}{N}\sum_{t=1}^{N}\varphi(t)e_0(t)\right) \\ &= \theta_0 + R(N)^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)e_0(t)\end{aligned}$$

What is the behavior of the estimation error when the number of data N goes to infinity? Assume that $\varphi(t)$ and $e_0(t)$ are stationary and the ergodicity property holds. Then,

$$\lim_{N\to\infty}(\hat{\theta}_N^{LS} - \theta_0) = \lim_{N\to\infty} R(N)^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)e_0(t) \triangleq (R^*)^{-1} f^*$$

where

$$\begin{aligned}R^* &\triangleq \lim_{N\to\infty} R(N) = \mathbf{E}\left[\varphi(t)\varphi^T(t)\right], \\ f^* &\triangleq \lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)e_0(t) = \mathbf{E}\left[\varphi(t)e_0(t)\right]\end{aligned}$$

and the equalities on the right follow from the ergodicity of $\varphi(t)$ and $e_0(t)$. So, if $N \to \infty$ one has $\hat{\theta}_N^{LS} \to \theta_0$ (i.e., the least-square estimate is consistent) if:

i) R^* is non singular (it depends on the input);

ii) $f^* = 0$ (it depends on the characteristics of e_0). This holds, e.g., if the input u is a deterministic process, or anyway independent from e_0, and in addition, one of the following conditions hold:

   - $e_0(t)$ is a zero-mean white process;

– $n_a = 0$ (FIR model).

If $n_a > 0$ and e_0 is not white, then, in general, $f^* \neq 0$, and the LS estimate is not consistent.

3.4.2 Numerical solutions to the optimization problem

In the general case, the minimum of the cost function

$$V_N\left(\theta, \mathcal{Z}^N\right) = \frac{1}{N}\sum_{t=1}^{N} \ell(\varepsilon(t,\theta))$$

cannot be computed analytically. Moreover, in general, such function is non-convex in parameters, leading to local minima. In this case, the solution of the optimization problem can be computed through numerical methods which make use of *iterative* procedures like the following:

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} + \mu^{(i)} f^{(i)}.$$

The term $f^{(i)}$ denotes the *search direction* and it is chosen on the basis of the information on function $V_N\left(\theta, \mathcal{Z}^N\right)$ available at the i-th iteration step. The term $\mu^{(i)}$ is computed to provide a decrease of the function, i.e. such that $V_N(\hat{\theta}^{(i+1)}, \mathcal{Z}^N) < V_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$.

It is possible to define three classes of iterative methods, depending on the choice of the direction $f^{(i)}$:

1. using only values of $V_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$;
2. using values of $V_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$ and of the gradient $V'_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$;
3. using values of $V_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$, of $V'_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$ and of the Hessian $V''_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$.

A typical example of the third kind is the *Newton algorithm*, where

$$f^{(i)} = -\left[V''_N\left(\hat{\theta}^{(i)}, \mathcal{Z}^N\right)\right]^{-1} V'_N\left(\hat{\theta}^{(i)}, \mathcal{Z}^N\right).$$

This algorithm guarantees convergence in one step (with $\mu = 1$) if the cost function is quadratic and convex in θ.

If the weighting function $\ell(\cdot)$ is chosen such that

$$\ell(\varepsilon(t,\theta)) = \frac{1}{2}\varepsilon^2(t,\theta)$$

we obtain the so-called *nonlinear least-squares* problem:

$$\min_{\theta} \frac{1}{2N}\sum_{t=1}^{N} \varepsilon^2(t,\theta).$$

In this case, the gradient of $V_N\left(\theta, \mathcal{Z}^N\right)$ is

$$V'_N\left(\theta, \mathcal{Z}^N\right) = -\frac{1}{N}\sum_{t=1}^{N}\psi(t,\theta)\varepsilon(t,\theta) \tag{3.21}$$

where $\psi(t,\theta) = -\frac{\mathrm{d}}{\mathrm{d}\theta}\varepsilon(t,\theta)$.

A common family of iterative algorithms to compute the minimum of a function has this structure:

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} - \mu_N^{(i)}\left[R_N^{(i)}\right]^{-1} V'_N\left(\hat{\theta}^{(i)}, \mathcal{Z}^N\right),$$

where

- $R_N^{(i)}$ is a matrix which changes the search direction;
- $\mu_N^{(i)}$ is the "size" of the step, such that $V_N(\hat{\theta}^{(i+1)}, \mathcal{Z}^N) < V_N(\hat{\theta}^{(i)}, \mathcal{Z}^N)$.

If $R_N^{(i)}$ is the identity matrix, the classical *gradient method* is obtained, which turns out to be quite inefficient in proximity of the point of minimum. Better performance are obtained by the Newton method, which corresponds to the following choice:

$$R_N^{(i)} = V''_N\left(\hat{\theta}^{(i)}, \mathcal{Z}^N\right).$$

Notice that, from (3.21) one has:

$$V''_N\left(\theta, \mathcal{Z}^N\right) = \frac{1}{N}\sum_{t=1}^{N}\psi(t,\theta)\psi^T(t,\theta) - \frac{1}{N}\sum_{t=1}^{N}\left(\frac{\mathrm{d}}{\mathrm{d}\theta}\psi(t,\theta)\right)\varepsilon(t,\theta),$$

where $\frac{\mathrm{d}}{\mathrm{d}\theta}\psi(t,\theta)$ is the Hessian of $\varepsilon(t,\theta)$ (matrix $d \times d$).

Notice that, the computation of $\frac{\mathrm{d}}{\mathrm{d}\theta}\psi(t,\theta)$ can be onerous. In many cases, it can be avoided since it is sufficient to have a good approximation of $V''_N(\theta, \mathcal{Z}^N)$ in a neighborhood of the minimum (far from the minimum, the cost function can be rarely approximated by a quadratic function, and then the gradient and the Newton methods have similar performance). But if the minimum is in correspondence of a θ_0 for which the prediction error $\varepsilon(t,\theta_0)$ is a white noise, one has

$$\mathbf{E}\left[\left(\frac{\mathrm{d}}{\mathrm{d}\theta}\psi(t,\theta_0)\right)\varepsilon(t,\theta_0)\right] = 0$$

and then,

$$V''_N\left(\theta, \mathcal{Z}^N\right) \simeq \frac{1}{N}\sum_{t=1}^{N}\psi(t,\theta)\psi^T(t,\theta).$$

This approximation of $V''_N(\theta, \mathcal{Z}^N)$ leads to the so-called Gauss-Newton method, which corresponds to:

$$R_N^{(i)} = \frac{1}{N}\sum_{t=1}^{N}\psi(t,\hat{\theta}^{(i)})\psi^T(t,\hat{\theta}^{(i)})$$

If the matrix $R_N^{(i)}$ is close to singularity, regularization techniques can be adopted, like e.g., that of Levenberg-Marquardt, where a damping coefficient $\lambda > 0$ is introduced:

$$R_N^{(i)} = \frac{1}{N}\sum_{t=1}^{N}\psi(t,\hat{\theta}^{(i)})\psi^T(t,\hat{\theta}^{(i)}) + \lambda I$$

where λ is a scalar used to control the convergence (in this case $\mu_N^{(i)} = 1$). Notice that, when $\lambda \to 0$ the Gauss-Newton method is obtained, while if $\lambda \to \infty$ one has the gradient method.

Let us now focus on how to compute the gradients for some of the model classes previously introduced. Since $\varepsilon(t,\theta) = y(t) - \hat{y}(t|t-1,\theta)$, the above mentioned methods require the computation of

$$\psi(t,\theta) = -\frac{\mathrm{d}}{\mathrm{d}\theta}\varepsilon(t,\theta) = \frac{\mathrm{d}}{\mathrm{d}\theta}\hat{y}(t|t-1,\theta).$$

For ARMAX models, recall that by (3.13)

$$C(z)\hat{y}(t|t-1) = B(z)u(t) + (C(z) - A(z))y(t)$$

and hence,

$$\begin{aligned} C(z)\frac{\partial}{\partial a_k}\hat{y}(t|t-1) &= -y(t-k), && k = 1,\ldots,n_a \\ C(z)\frac{\partial}{\partial b_k}\hat{y}(t|t-1) &= u(t-k), && k = 1,\ldots,n_b \\ C(z)\frac{\partial}{\partial c_k}\hat{y}(t|t-1) + z^{-k}\hat{y}(t|t-1) &= y(t-k), && k = 1,\ldots,n_c \end{aligned}$$

The previously three equation families can be compactly represented as

$$C(z)\frac{\mathrm{d}}{\mathrm{d}\theta}\hat{y}(t|t-1) = \varphi(t,\theta),$$

that is

$$C(z)\psi(t,\theta) = \varphi(t,\theta).$$

So, $\psi(t,\theta)$ is obtained by filtering the pseudo-regressor $\varphi(t,\theta)$ by the filter $\frac{1}{C(z)}$. In practice, since $C(z)$ is unknown, at each instant the current estimate of this polynomial will be used (i.e., the coefficients of $C(z)$ will be those contained in the corresponding part of the estimated vector $\hat{\theta}^{(i)}$).

In a similar way, in case of OE models, the predictor will satisfy:

$$F(z)\hat{y}(t|t-1) = B(z)u(t),$$

and then,

$$\begin{aligned} F(z)\frac{\partial}{\partial b_k}\hat{y}(t|t-1) &= u(t-k), && k = 1,\ldots,n_b \\ F(z)\frac{\partial}{\partial f_k}\hat{y}(t|t-1) + z^{-k}\hat{y}(t|t-1) &= 0, && k = 1,\ldots,n_f \end{aligned}$$

Remembering that for an OE model $\hat{y}(t-k|t-k-1) = w(t-k|t-k-1)$, and the pseudo-regressor is

$$\varphi(t,\theta) = [u(t-1)\dots u(t-n_b) \ -w(t-1,\theta) - \cdots - w(t-n_f,\theta)]^T,$$

one has

$$F(z)\psi(t,\theta) = \varphi(t,\theta).$$

Of course, the filtering procedure of the pseudo-regressor needs an initial condition. In general, the choice $\varphi(0,\theta) = 0$ is not the best one. A possible alternative is to set

$$\varphi(0,\theta) = \eta$$

and to include η in the vector parameter θ to be estimated.

3.5 Recursive system identification

In Section 3.4.1, the least square estimate has been introduced to solve the identification problem for linear regression models(such as ARX, FIR, etc.). In fact, the unique solution of problem

$$\min_{\theta} \sum_{t=1}^{N} (y(t) - \varphi^T(t)\theta)^2$$

is given by the least squares estimate

$$\hat{\theta}_N^{LS} = \left(\sum_{t=1}^{N} \varphi(t)\varphi^T(t) \right)^{-1} \sum_{t=1}^{N} \varphi(t)y(t) \tag{3.22}$$

provided that the matrix $\sum_{t=1}^{N} \varphi(t)\varphi^T(t) \in \mathbb{R}^{d\times d}$ is nonsingular. Being the sum of N matrices of rank 1, a necessary condition is that $N \geq d$.

The computation of (3.22) is straightforward once we have collected the entire input-output dataset $\{u(1), y(1), \cdots u(N), y(N)\}$. However, in many practical applications one would like to estimate the parameter vector θ *online*, i.e., while the input-output data are being collected. A typical example is that of *fault detection*, in which the estimate of the parameter θ is continuously updated and compared to an ideal reference θ_0, in order to check if the system is significantly deviating from its nominal behavior. Another application is *adaptive control*, in which the identified model is used to update in real time the parameters of a feedback controller, which is designed according to the estimated parameter vector $\hat{\theta}$. This means that at each time t we aim at computing

$$\hat{\theta}_t = \arg\min_{\theta} \sum_{k=1}^{t} (y(k) - \varphi^T(k)\theta)^2.$$

Obviously, the solution is given by the least squares estimate

$$\hat{\theta}_t^{LS} = \left(\sum_{k=1}^{t} \varphi(k)\varphi^T(k) \right)^{-1} \sum_{k=1}^{t} \varphi(k)y(k) \tag{3.23}$$

but it would be clearly impractical to use equation (3.23) to compute the parameter estimates at each time t, as one needs to process a steadily increasing number of data as t grows.

As an alternative, we look for a *recursive solution* that updates the estimate at time t by using only the previous estimate at time $t-1$ and the new data pair $y(t), \varphi(t)$. Let

$$R(t) = \sum_{k=1}^{t} \varphi(k)\varphi^T(k) \qquad f(t) = \sum_{k=1}^{t} \varphi(k)y(k)$$

And rewrite the equation of the least squares estimate (3.23) as

$$\hat{\theta}_t = R(t)^{-1} f(t).$$

Now, it is easy to see that $R(t)$ and $f(t)$ can be updated in a recursive manner, by using just the current data pair $y(t), \varphi(t)$ at time t, according to

$$\begin{aligned} R(t) &= R(t-1) + \varphi(t)\varphi^T(t) \ , \\ f(t) &= f(t-1) + \varphi(t)y(t) \ . \end{aligned}$$

By exploiting the fact that $\hat{\theta}_{t-1} = R(t-1)^{-1} f(t-1)$ implies $f(t-1) = R(t-1)\hat{\theta}_{t-1}$, one has

$$\begin{aligned} \hat{\theta}_t &= R(t)^{-1} f(t) = R(t)^{-1}(f(t-1) + \varphi(t)y(t)) \\ &= R(t)^{-1} \left(R(t-1)\hat{\theta}_{t-1} + \varphi(t)y(t) \right) = \\ &= R(t)^{-1} \left[\left(R(t) - \varphi(t)\varphi^T(t) \right) \hat{\theta}_{t-1} + \varphi(t)y(t) \right] = \\ &= R(t)^{-1} R(t)\hat{\theta}_{t-1} + R(t)^{-1}\varphi(t)y(t) - R(t)^{-1}\varphi(t)\varphi^T(t)\hat{\theta}_{t-1} = \\ &= \hat{\theta}_{t-1} + R(t)^{-1}\varphi(t) \left[y(t) - \varphi^T(t)\hat{\theta}_{t-1} \right] . \end{aligned}$$

Hence, one can update the estimate $\hat{\theta}_t$ at each time t by using the following recursive algorithm

$$\begin{aligned} \hat{\theta}_t &= \hat{\theta}_{t-1} + R(t)^{-1}\varphi(t) \left[y(t) - \varphi^T(t)\hat{\theta}_{t-1} \right] \\ R(t+1) &= R(t) + \varphi(t+1)\varphi^T(t+1) \end{aligned} \tag{3.24}$$

Inverting the matrix $R(t) \in \mathbb{R}^{d \times d}$ requires $O(d^3)$ operations at each time step. In order to derive a more efficient algorithm, it is better to propagate the inverse

$P(t) = R(t)^{-1}$. In fact, by exploiting the Matrix Inversion Lemma[4], one gets

$$\begin{aligned} P(t) &= \left[R(t-1) + \varphi(t)\varphi^T(t)\right]^{-1} = \\ &= P(t-1) - P(t-1)\varphi(t)\left[1 + \varphi^T(t)P(t-1)\varphi(t)\right]^{-1}\varphi^T(t)P(t-1) = \\ &= P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1 + \varphi^T(t)P(t-1)\varphi(t)} \end{aligned}$$

Moreover,

$$\begin{aligned} R(t)^{-1}\varphi(t) &= P(t)\varphi(t) \\ &= \left[P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1 + \varphi^T(t)P(t-1)\varphi(t)}\right]\varphi(t) \\ &= \frac{P(t-1)\varphi(t)}{1 + \varphi^T(t)P(t-1)\varphi(t)}. \end{aligned}$$

By substituting the above relationships into (3.24) one obtains the *Recursive Least Squares (RLS)* algorithm

$$\begin{aligned} \hat{\theta}_t &= \hat{\theta}_{t-1} + L(t)[y(t) - \varphi^T(t)\hat{\theta}_{t-1}] \\ L(t) &= \frac{P(t-1)\varphi(t)}{1 + \varphi^T(t)P(t-1)\varphi(t)} \\ P(t) &= P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1 + \varphi^T(t)P(t-1)\varphi(t)} \end{aligned} \tag{3.25}$$

The RLS algorithm has computational complexity $O(d^2)$. It should be remarked that the first equation in (3.25) uses the output prediction error $y(t) - \hat{y}(t|t-1) = y(t) - \varphi^T(t)\hat{\theta}_{t-1}$ in order to update the parameter estimate from time $t-1$ to time t, by means of the RLS gain $L(t)$, which in turn depends on the matrix $P(t)$ and the regressor vector $\varphi(t)$.

3.5.1 Initialization of the RLS algorithm

There are several possible ways to initialize the RLS recursion. If one uses the form (3.24), the matrices $R(t)$ must be full rank (i.e., $\text{rank}(R(t)) = d$), in order to be invertible. Hence, one can use an initial batch of data for $t = 1, 2, \ldots, t_0$ such that $\text{rank}(R(t_0)) = d$ (notice that it must be $t_0 \geq d$). Then, set

$$\hat{\theta}_{t_0} = R(t_0)^{-1} f(t_0)$$

and iterate the RLS algorithm from t_0 onwards (i.e., for $t = t_0 + 1, t_0 + 2, \cdots$). Notice that this ensures that the resulting estimates $\hat{\theta}_t$ are exactly equal to the batch solution (3.23), for all $t > t_0$.

On the other hand, if one uses the form (3.25), it is possible to start the recursion straight away at time $t = 1$ (i.e., when the first data pair $y(1), \varphi(1)$ is available).

[4]The Matrix Inversion Lemma states: $(\bar{A} - \bar{B}\bar{C}\bar{D})^{-1} = \bar{A}^{-1} + \bar{A}^{-1}\bar{B}(\bar{C}^{-1} - \bar{D}\bar{A}^{-1}\bar{B})^{-1}\bar{D}\bar{A}^{-1}$.

This can be done by choosing any $\hat{\theta}_0 \in \mathbb{R}^d$ and any matrix $P(0) = P_0 \in \mathbb{R}^{d\times d}$ such that $P_0 = P_0^T > 0$. In this case, the RLS algorithm will return at every time t the solution of the following optimization problem

$$\hat{\theta}_t = \arg\min_{\theta} \sum_{k=1}^{t} \left(y(k) - \varphi^T(k)\theta\right)^2 + (\theta - \hat{\theta}_0)^T P_0^{-1}(\theta - \hat{\theta}_0) \tag{3.26}$$

where it is apparent the influence of the initialization. Clearly, the larger is P_0, the smaller is such an influence. In any case, the effect of the initialization will decay asymptotically and the estimate resulting from (3.26) will converge to the least squares solution (3.23), as t grows.

3.5.2 Exponential data weighting

A possible problem of the RLS algorithm is that it may not be sufficiently reactive to *abrupt changes* (i.e., rapid variations in θ), once a large amount of data has been processed. A key idea to overcome this problem is to weight more the last data with respect to the old ones. For example, this can be done by introducing an *exponential data weighting*, which corresponds to solving the optimization problem

$$\min_{\theta} \sum_{k=1}^{t} \lambda^{t-k} \left(y(k) - \varphi^T(k)\theta\right)^2 + \lambda^t(\theta - \hat{\theta}_0)^T P_0^{-1}(\theta - \hat{\theta}_0)$$

where $\lambda \in (0, 1)$ is the *forgetting factor*, enforcing higher weights for more recent data. Typical values for λ are chosen in the range $[0.95, 0.999]$, with smaller values corresponding to more reactive algorithms. The resulting Exponentially Weighted RLS algorithm (EW-RLS) turns out to be

$$\begin{aligned}
\hat{\theta}_t &= \hat{\theta}_{t-1} + L(t)\left[y(t) - \varphi^T(t)\hat{\theta}_{t-1}\right] \\
L(t) &= \frac{P(t-1)\varphi(t)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)} \\
P(t) &= \frac{1}{\lambda}\left[P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1 + \varphi^T(t)P(t-1)\varphi(t)}\right]
\end{aligned}$$

The presence of λ in the second and third equation prevents $P(t)$ and $L(t)$ from vanishing to zero. On the other hand, a too small λ will lead to large values of the covariance $P(t)$ of the estimation error, which clearly affects the quality of the estimate (once again, a reduction in the bias error may result in an increase of the error variance).

3.6 Model quality assessment

As reported in the identification procedure of Section 3.1, once a model has been identified, it is necessary to *validate* it, that is to evaluate if it is coherent with the

available information on the system (a-priori knowledge, data, etc.), and if it can be effectively used for the purpose it has been built.

Quality of a mathematical model will also depend on the specific application at hand and how such model will be used. For instance, a model used to design a feedback control system needs different requirements with respect to one used to predict the output in given working conditions. Nevertheless, there exist some techniques to evaluate the model quality which can be applied in general. In the following, some of these techniques will be described.

In the general scheme reported in Fig. 3.1, it can be noticed that one of the elements to be reconsidered during the validation phase is the choice of the model class. Of course, if some a-priori information are available about the system to be identified, such information must be used to choose the proper model class. In particular, it can be chosen:

- the type of model (linear or nonlinear, input/output or state-space, gray-box or black-box, etc.);
- the structure of the model (e.g., for LTI systems: ARX, ARMAX, OE, BJ, etc.);
- the number of parameters to be estimated.

There are several aspects to consider when the model class is chosen. In general, the model quality can be quantitatively measured through a suitable index J. Usually, such index is composed of two parts:

$$J = J_B + J_V$$

where J_B represents a *bias* term (depending on the chosen model class, but not on the data), and J_V is a *variance* term (it depends on the data and the dimension of the parameter vector). In other words, J_B takes into account the "distance" between the chosen model class and the system which has generated the data, while J_V is related to the variance of the estimation error $\theta - \hat{\theta}_N$, which usually grows with the number of parameters d to be estimated (for the same length N of the available data). This means that a model rich and flexible (i.e., with a complex structure and many parameters) will lead to a low bias error but to a large variance error. On the contrary, a simple model favors J_V with respect to J_B.

One more interesting aspect regards the *computational burden* to compute a model. A complex model or a complex cost function $V_N(\theta, \mathcal{Z}^N)$ might need high computational resources to estimate the parameter vector. As previously stated, a fundamental aspect regards the use of the estimated model: complex models could be difficult to simulate, the control system design might become too complex, etc.

So, the final choice of the model class will be a trade-off between the previously described aspects. The evaluation of such aspects may take into account several features, including:

- a-priori information on the system;
- preliminary analysis of data (before computing the estimate);
- comparisons of various model structures;
- a-posteriori model validation.

3.6.1 Prior knowledge

Prior knowledge on the system is strictly related to the specific problem. One may choose to favor physical aspects (less parameters, models adapted to the system at hand) or a black-box model (simpler from an algorithmic viewpoint). Nonlinearities (usually a-priori known at least approximately) can be considered or not, and data can be transformed (e.g., filtered) before proceeding to the estimation phase. In any case, a golden rule is to start considering simple structures (*principle of parsimony*).

3.6.2 Data analysis

There are several methods to estimate the order of a linear system (and then of the chosen model class):

- using a non-parametric frequency estimation, for instance through spectral analysis, to find resonance peaks, bandwidth, etc;
- adding to the model a new variable $y(t-n-1)$ or a measured noise $w(t)$, and analyze the correlation with $y(t)$ (if the correlation is negligible, it is not convenient to increase the order of the model);
- studying the condition number of the matrix $\frac{1}{N}\sum_{t=1}^{N}\psi(t,\theta)\psi^T(t,\theta)$ when computing the estimate with the Gauss-Newton methods (see Section 3.4.2): if such matrix is close to be singular, this means that there are some directions in the regressor space for which $\psi(t,\theta)$ is almost zero, and then there is a redundancy in the parameter vector. In this case, it is convenient to reduce the model order.

3.6.3 Comparison between different model structure

In general, the quality of an identified model is evaluated on the basis of its capacity to "reproduce the data" generated by the unknown system. But what does "repro-

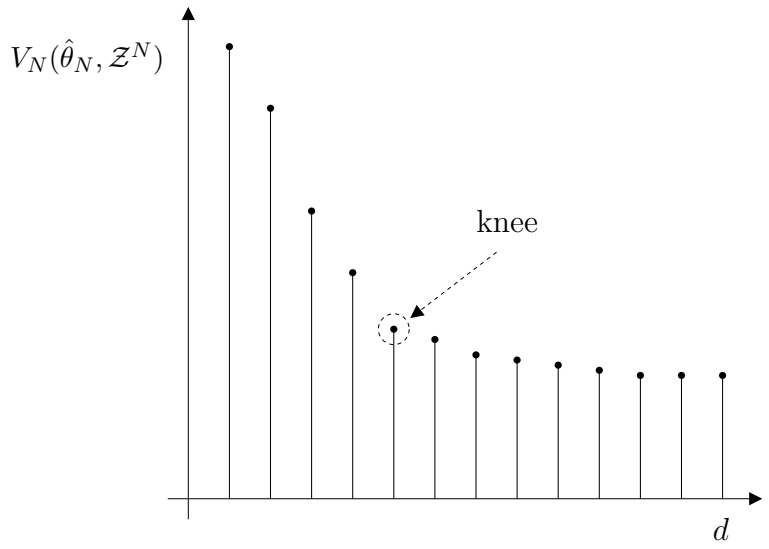

Figure 3.7: Minimal value of the cost function for different number of parameters.

duce the data" mean? A possible answer can be given by analyzing its k-step ahead *predictive capacity*. Two main cases are:

$$k = 1 \quad \Longrightarrow \quad \hat{y}(t|t-1) = (1 - H^{-1}(z, \hat{\theta}_N))y(t) + H^{-1}(z, \hat{\theta}_N)G(z, \hat{\theta}_N)u(t)$$
$$k = \infty \quad \Longrightarrow \quad \hat{y}_{sim}(t) = G(z, \hat{\theta}_N)u(t)$$

When $k = \infty$ we talk about *simulation*, because the system output $\hat{y}_{sim}(t)$ is computed by simulating the identified model $G(z, \hat{\theta}_N)$ with the input $u(t)$. Clearly, the predictive capacity in simulation is a stronger requirement than prediction, since it does not use the knowledge of the past system output.

A first measure of the quality of a model is given by the value of the cost function for the optimal parameter vector:

$$V_N\left(\hat{\theta}_N, \mathcal{Z}^N\right) = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t|t-1))^2$$

Of course, the less this value the more the model can reproduce the data. However, by increasing the complexity of the model (and hence the number of parameters $d = \dim(\theta)$) the cost always decreases. Let us consider two model classes of increasing complexity

$$\mathcal{M}_1 \subset \mathcal{M}_2,$$

that is, the case for which $\Theta_1 \subseteq \mathbb{R}^{n_1}$, $\Theta_2 \subseteq \mathbb{R}^{n_2}$, with $n_2 > n_1$, and Θ_1 can be obtained by a reduction of Θ_2 to $\mathbb{R}^{n_1}$ (i.e., by setting to zero the exceeding parameters). One has,

$$V_N(\hat{\theta}_N^{(1)}, \mathcal{Z}^N) \geq V_N(\hat{\theta}_N^{(2)}, \mathcal{Z}^N)$$

where $\hat{\theta}_N^{(1)}$ and $\hat{\theta}_N^{(2)}$ are the optimal parameters for $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively.

In Fig. 3.7, a qualitative plot of the cost function for different number of parameters d is reported. Until a given value of d, the cost function decreases significantly,

thanks to a better ability of the model to reproduce data. On the contrary, for greater values of d, the cost function decreases slowly, since the new parameters allow a better reproduction of the noise affecting data (*overfit*). Of course, such contribution is useless, since the noise for different realizations is unpredictable. The knee of the curve can be used as a heuristic method to find the best order for the chosen model class.

A quantitative approach consists in choosing the optimal number of parameters on the basis of suitable numerical criteria. The basic idea is to associate a penalty to each parameter, such that the final cost increases for increasing model order, in the presence of overfit. In particular, inside a given model class $\mathcal{M}$, the optimal order d^* will be that minimizing the following function

$$\mathcal{I}(\mathcal{M}) = V_N(\hat{\theta}_N, \mathcal{Z}^N)(1 + U(\theta)).$$

Depending on the function $U(\theta)$, different criteria can be defined. The most used are:

- **AIC**, *Akaike Information Criterion*, where

$$U(\theta) = \frac{2d}{N};$$

- **FPE**, *Final Prediction Error* criterion, where

$$U(\theta) = \frac{2d}{N-d};$$

- **MDL**, *Minimum Description Length* criterion, where

$$U(\theta) = \frac{\log N}{N} d.$$

For "low" values of d, the cost $\mathcal{I}$ is decreasing (since $V_N(\hat{\theta}_N, \mathcal{Z}^N)$ decreases for increasing d). On the contrary, by increasing the parameter number d, the cost grows because $V_N(\hat{\theta}_N, \mathcal{Z}^N)$ is almost constant and the term $U(\theta)$ dominates, see Fig. 3.8.

Till now, all the reasonings were related to a comparison between different structures of the same cost function $V_N(\hat{\theta}_N, \mathcal{Z}^N)$ used to obtain the optimal model inside a given class. Of course, it is possible to use different cost functions for model estimation and comparison. A classical example is to use the simulation error to compare different models,

$$J_{sim} = \frac{1}{N} \sum_{t=1}^{N} \left(y(t) - \hat{y}_{sim}(t)\right)^2 = \frac{1}{N} \sum_{t=1}^{N} \left(y(t) - G(z, \hat{\theta}_N) u(t)\right)^2. \tag{3.27}$$

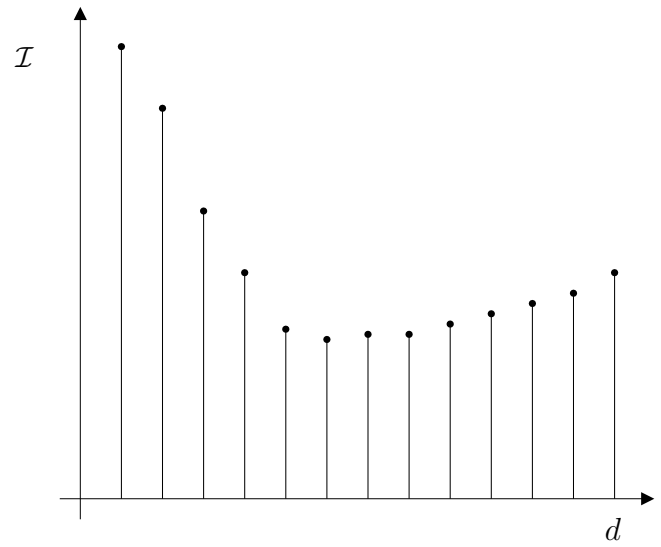

Figure 3.8: Plot of the cost function $\mathcal{I}$.

Often, when comparing different model classes, it is convenient to use the percentage FIT index, defined as

$$FIT = \left\{ 1 - \frac{J_{sim}}{\frac{1}{N}\sum_{t=1}^{N} \left(y(t) - \bar{y}\right)^2} \right\} \times 100$$

where $\bar{y} = \frac{1}{N}\sum_{t=1}^{N} y(t)$ is the sample mean of the output. 100% FIT denotes that the estimated model can perfectly reproduce data in simulation. This corresponds to the ideal case where data are generated exactly by the identified transfer function $G(z, \hat{\theta}_N)$ in absence of noise.

Notice that, a high FIT in simulation (i.e. a small error J_{sim}) is a stronger requirement with respect to the minimization of $V_N(\hat{\theta}_N, \mathcal{Z}^N)$, and so it is a harder test to assess the quality of the identified model. An exception is related to the OE model class, where $\hat{y}_{sim}(t) = \hat{y}(t|t-1)$.

Comparison with different data

A common measure of the quality of the identified model is its ability to reproduce different data sets with respect to that used to estimate the parameters. In fact, a model is not identified to reproduce the data set $\mathcal{Z}^N$ used in the estimation procedure, but to represent the system behavior in different working conditions (e.g., for input signals different to that used in $\mathcal{Z}^N$).

To compare models with different structures, a good practice is to use a different data set with respect to that used for estimation. So, the available data set must be divided in two subsets:

- $\mathcal{Z}_1^N = \{u_1(1), y_1(1), \ldots, u_1(N), y_1(N)\}$: *estimation data*;
- $\mathcal{Z}_2^M = \{u_2(1), y_2(1), \ldots, u_2(M), y_2(M)\}$: *validation data*.

Estimation data are used to estimate the parameter θ, by using the preferred identification technique (e.g., one of those described in this chapter). To highlight the dependence on the estimation data, let us denote the computed parameter vector as $\hat{\theta}_N(\mathcal{Z}_1^N)$. Finally, one can assess the quality of the identified model through a quantitative criterion based on the validation data $\mathcal{Z}_2^N$. The straightforward method is to use the 1-step ahead prediction error

$$V_N(\hat{\theta}_N(\mathcal{Z}_1^N), \mathcal{Z}_2^M) = \frac{1}{M}\sum_{t=1}^{M}(y_2(t) - \hat{y}_2(t|t-1))^2$$

where the subscript 2 in the right expression means that the data used to compute the cost function are referred to the validation set $\mathcal{Z}_2^M$. In particular, notice that the prediction $\hat{y}_2(t|t-1)$ is computed employing the second data set, by the model $\hat{\theta}_N(\mathcal{Z}_1^N)$, that is

$$\hat{y}_2(t|t-1) = \left[1 - H^{-1}(z, \hat{\theta}_N(\mathcal{Z}_1^N))\right] y_2(t) + H^{-1}(z, \hat{\theta}_N(\mathcal{Z}_1^N))G(z, \hat{\theta}_N(\mathcal{Z}_1^N))u_2(t).$$

Clearly, different cost functions can be chosen, like J_{sim} defined in (3.27). If only one data set is available, it is a good practice to divide it in two subsets to be used for estimation and validation, respectively. In case of few available data, it is preferred to use the larger subset for identifying the model, in order to obtain a sufficiently accurate model estimation.

3.6.4 Model validation: residual analysis

An important role in model validation is given by the prediction errors or *residuals*:

$$\varepsilon(t, \hat{\theta}_N) = y(t) - \hat{y}(t|t-1).$$

Residuals represents the part of data that a model is not able to reproduce. For this reason, analyzing their statistical properties may give some information about model quality. In particular, let us consider:

$$\hat{R}_{\varepsilon u}^N(\tau) = \frac{1}{N-\tau}\sum_{t=1}^{N-\tau}\varepsilon(t+\tau)u(t), \qquad -M \leq \tau \leq M \tag{3.28}$$

$$\hat{R}_{\varepsilon}^N(\tau) = \frac{1}{N-\tau}\sum_{t=1}^{N-\tau}\varepsilon(t+\tau)\varepsilon(t), \qquad 1 \leq \tau \leq M \tag{3.29}$$

where M is an integer greater than 1 and typically $M \ll N$. Of course, $\hat{R}_{\varepsilon u}^N(\tau)$ and $\hat{R}_{\varepsilon}^N(\tau)$ are estimates of the cross covariance function between ε and u, and of the covariance function of residuals ε, respectively. Evidently, both functions must be "small" for the following reasons.

- Residuals must not depend on the peculiar data set employed. In particular, they must not be correlated with the input, otherwise the model quality may change for different input. For this reason, it is required that $R^N_{\varepsilon u}(\tau)$ be small. One more reasoning is the following: if there is a dependence between ε and u, then there is a contribute on the output y which derives from the input u and which is not been taken by the identified model. So, the identified model can be improved.

- If the residuals are correlated, then a part of $\varepsilon(t)$ can be foreseen on the basis of $\mathcal{Z}^{t-1}$, i.e. on the data available till the previous time step. Also in this case, the model can be improved.

From a theoretical viewpoint, if we assume that data are generated by a model belonging to the chosen LTI model class, then the following equation really holds

$$y(t) = G_o(z)u(t) + H_o(z)e(t).$$

for some $G_o(z)$, $H_o(z)$. If the identification algorithm is able to estimate the exact parameters of the model (in principle, this is possible under some hypothesis if the number of data N tends to infinity), by using (3.12) the prediction error becomes

$$\begin{aligned} \varepsilon(t) &= H_o^{-1}(z)\left(y(t) - G_o(z)u(t)\right) \\ & \quad H_o^{-1}(z)\left(G_o(z)u(t) + H_o(z)e(t) - G_o(z)u(t)\right) = e(t). \end{aligned}$$

In other words, in case of perfect identification, the prediction error is a white stochastic process, and then $R_{\boldsymbol{\varepsilon}}(\tau) = 0$ for $\tau \neq 0$. For this reason, the value of the cost function $V_N\left(\hat{\theta}_N, \mathcal{Z}^N\right)$ (also knows as *loss function*) is an estimate of the variance σ_e^2 of the process $e(t)$.

Similarly, it can be stated that in the ideal case, $\varepsilon(t)$ must be not correlated with input $u(t)$ (if $u(t)$ is a stochastic process, it is sufficient that it is not correlated with $e(t)$), and hence $R_{\boldsymbol{\varepsilon u}}(\tau) = 0, \forall\tau$.

Since (3.28)-(3.29) are only estimates of the real covariance functions, it is not reasonable to require they are exactly zero. For this reason, it is sufficient they are small. But how can we quantify these reasonings? There exist some statistical tests able to assess (with a given confidence level) if a sequence can be interpreted as a realization of a white noise. Let us consider the case $\hat{R}^N_{\varepsilon}$ (a similar reasoning can be repeated for $\hat{R}^N_{\varepsilon u}$). Let us suppose that the residuals $\{\varepsilon(t)\}_{t=1}^N$ are really white, with variance $\sigma^2_{\boldsymbol{e}}$. Then, it can be proved that asymptotically (i.e., for $N \to \infty$) the vector

$$E_M = \sqrt{N}\begin{bmatrix} \hat{R}^N_{\varepsilon}(1) \\ \hat{R}^N_{\varepsilon}(2) \\ \vdots \\ \hat{R}^N_{\varepsilon}(M) \end{bmatrix}$$

has a Gaussian distribution, with zero mean and covariance matrix $\sigma_e^4 I$. So, a possible whiteness test on $\{\varepsilon(t)\}_{t=1}^N$ can be performed, by checking that for any $\tau \neq 0$, the estimated covariance $\hat{R}_\varepsilon^N(\tau)$ belongs to a suitable confidence interval $[-\gamma_\varepsilon,\ \gamma_\varepsilon]$. The size of this interval can be computed from the desired confidence level. For instance, for 99%-test (i.e., accepting a probability to fail equal to 1%) the threshold is

$$\gamma_\varepsilon = \frac{\alpha}{\sqrt{N}} R_\varepsilon^N(0)$$

where α is such that $\mathrm{P}\left(|\boldsymbol{x}| < \alpha\right) = 0.99$, with $\boldsymbol{x}$ normal random variable. To compute α in Matlab, one can use the function `erfinv`:

$$\alpha = \sqrt{2}\,\texttt{erfinv(0.99)} = 2.58.$$

3.7 Input selection

In some cases, the user is free to choose the input to apply to the system to gather data for identification. Such a choice may strongly affect the quality of the identification procedure. For this reason, it is convenient to consider the following aspects.

- The asymptotic characteristics of the estimated parameter vector do not depend on the waveform of the input, but only on its frequency content (spectrum). In particular, the more the input power (i.e. the signal-to-noise ratio), the less the estimate variance.
- Due to physical constraints (e.g., saturations), the input must be bounded. It is important to take this aspect into account, and to employ the saturated input for model estimation.
- Input signal must have a considerable frequency content in the band of interest. The idea is to choose an input with a flat spectrum in such a band.

Clearly, these are some contrasting requirements. The aim is to obtain the greatest input power (in the frequency band of interest) respecting the enforced bounds. In the following, some typical input signals used for system identification are reported.

White noise. It has a flat spectrum and, in principle, if suitably filtered it allows to obtain any other spectrum. The constraint on its maximum amplitude $\bar{u}$ can be respected by discarding the samples with an amplitude greater than that limit. For Gaussian white noise, if its standard deviation σ is chosen such that $3\sigma = \bar{u}$, such a procedure will lead to small spectra distortions.

Random Binary Sequence (RBS). It is a random signals which may assume only two values. For instance, it can be generated by taking the (scaled) sign of a Gaussian white noise.

Pseudo-Random Binary Sequence (PRBS). It is a periodic signal, with similar characteristics of a white noise. Actually, it is generated through a deterministic algorithm, but it is difficult to predict. If M is the period, the spectrum is composed of M impulse of the same amplitude. With respect to the white noise, the properties of non correlation hold not only asymptotically, but for any set of a length which is an integer multiple of the period.

Sum of sinusoids. This signal has the form:

$$u(t) = \sum_{k=1}^{d} A_k \sin(\omega_k t + \phi_k).$$

Also in this case, the spectrum is composed of a set of impulse at frequency ω_k. By changing the parameters $d,\ \omega_k,\ \phi_k$, it is possible to concentrate the signal power at fixed frequencies.

Chirp. A chirp signal is a sinusoid whose frequency changes with continuity inside an interval $[\omega_1,\ \omega_2]$, in a given time period $0 \leq t \leq M$:

$$u(t) = A \cos\left(\omega_1 t + (w_2 - \omega_1)\frac{t^2}{2M}\right).$$

Notice that, some of the previous inputs are periodic, other not. A periodic signal has the drawback to excite only M distinct frequencies. On the other side, at least two benefits can be found by using from the use of periodic inputs. First, it is possible to averaging the output among periods, and working with the mean data. In this case, the signal-to-noise ratio is increased by a factor equal to the number of periods used. Moreover, it is possible to estimate the noise affecting the output, since the difference between the output at various time steps is due uniquely to noise (once the transient is over).

If input/output data are not zero mean, a good practice is to remove mean from data before estimation. In general, this operation helps improving the performance of the estimated model.