INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. 2013; **27**:667–687
Published online 13 September 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/acs.2335

A new paradigm for parameter estimation in system modeling

Simone Garatti*,† and Sergio Bittanti

Dipartimento di Elettronica ed Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133, Milan, Italy

SUMMARY

In this paper, we consider a basic problem in system identification, that of estimating the unknown parameters of a given model by using input/output data. Available methods (extended Kalman filtering, unscented Kalman filtering, particle filtering, maximum likelihood, prediction error method, etc.) have been extensively studied in the literature, especially in relation to consistency analysis. Yet, other important aspects, such as computational complexity, have been somewhat overlooked so that, when such methods are used in practical problems, remarkable drawbacks may arise. This is why parameter estimation is often performed using empirical procedures. This paper aims to revisit the issue of setting up an *estimator* that is able to provide reliable estimates at low computational cost. In contrast to other paradigms, the main idea in the new introduced two-stage estimation method is to retrieve the estimator through simulation experiments in a training phase. Once training is terminated, the user is provided with an explicitly given estimator that can be used over and over basically with no computational effort. The advantages and drawbacks of the two-stage approach as well as other traditional paradigms are identified with an illustrative example. A more concrete example of tire parameter estimation is also provided. Copyright © 2012 John Wiley & Sons, Ltd.

Received 6 July 2011; Revised 8 August 2012; Accepted 19 August 2012

KEY WORDS: parameter estimation; gray-box modeling; system identification

1. INTRODUCTION

This paper focuses on the very basic problem of estimating unknown parameters in a given plant by using observed data [1–8]. To be precise, suppose that data are generated by a dynamical system (continuous time or discrete time, linear or nonlinear, finite or infinite dimensional, noise free or subject to disturbances) depending on a certain parameter vector $\theta \in \mathbb{R}^q$. The system is denoted by $P(\theta)$ as in Figure 1, where $u(t)$ and $y(t)$ are the input and output measurable signals, whereas $e(t)$ is a nonmeasurable exogenous input, possibly equal to 0 if the system is noise free. t denotes time and can be either continuous or discrete. For simplicity, we will assume that $u(t)$ and $y(t)$ take value in $\mathbb{R}$, that is, $P(\theta)$ is SISO. Extensions to the MIMO case are however possible.

In the considered problem, we assume that, whereas an exact mathematical model (and a corresponding simulator) for $P(\theta)$ is available, the current value of parameter θ is unknown and it therefore has to be retrieved on the basis of an experiment on the plant (white-box identification [8]). The system behavior is, thus, observed for a certain time interval over which a number N of input and output observations $\bar{D}^N = \{\bar{y}(1), \bar{u}(1), \bar{y}(2), \bar{u}(2), \ldots, \bar{y}(N), \bar{u}(N)\}$ are collected.‡ The issue is then how to exploit the information contained in the data to obtain a fair estimate of the uncertain parameter θ.

*Correspondence to: Simone Garatti, Dipartimento di Elettronica ed Informazione, Politecnico di Milano, Piazza L. da Vinci 32, 20133, Milan, Italy.

†E-mail: sgaratti@elet.polimi.it

‡In the case of continuous time systems, observations have to be intended as sampled data points, that is, $\bar{y}(i)$, $\bar{u}(i) = \bar{y}(t_0 + iT), \bar{u}(t_0 + iT)$, where t_0 is the initial time and T is the sampling period.

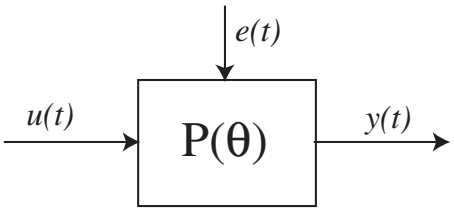

Figure 1. The data generating system.

Conceptually, the estimation problem is solved by introducing a suitable estimator (also called estimation algorithm) that is merely a function $\widehat{f} : \mathbb{R}^{2N} \to \mathbb{R}^q$, which maps the measured observations $\bar{D}^N = \{\bar{y}(1), \bar{u}(1), \bar{y}(2), \bar{u}(2), \ldots, \bar{y}(N), \bar{u}(N)\}$ to an estimate, say $\widehat{\theta} := \widehat{f}(\bar{y}(1), \bar{u}(1), \ldots, \bar{y}(N), \bar{u}(N))$, for the true parameter θ.

To be reliably used in practice, the estimator $\widehat{f}$ must satisfy some requirements. Among others, the following ones are relevant in a number of real applications:

(1) An accuracy property must hold, that is, it must be guaranteed that the estimation error $\|\widehat{\theta}-\theta\|$ is below a given threshold, at least for θ taking values in a range of interest.[§] Clearly, the type of guarantees may change depending on the specific framework at hand. In, for example, a stochastic framework, the mean square error $\mathbb{E}\|\widehat{\theta} - \theta\|^2$ is commonly considered, with the aim of achieving both asymptotic and nonasymptotic characterizations.
(2) The estimator must be able to return the estimate $\widehat{\theta}$ with low computational effort. In other words, the estimator function $\widehat{f}$ must be simple enough to be implemented as a computer algorithm with acceptable computational complexity according to the available resources.
(3) The estimator must be fully automatic, that is, no human supervision is allowed. In this regard, it is worth recalling the difficulties sometimes encountered when resorting to extended Kalman filtering (EKF). A common experience is the required efforts for the trial-and-error-based tuning of the initialization of the estimation error covariance matrix to achieve a good estimate. This tuning must be repeated each time the parameter θ to be estimated takes a new value, thus leading to a procedure requiring human supervision. Such human interaction, however, is not realistic in a number of problems; therefore, a fully automatic procedure must be adopted.

Clearly, the importance of each of the aforementioned requirements depends on the application at hand. In existing literature, however, estimators have seldom been evaluated according to criteria other than that of accuracy. Yet, in the estimation practice, other requirements can be more pressing than is commonly believed. The following example illustrates this point.

Example 1 (Pacejka's model parameters estimation)
One of the most critical aspects in modeling the dynamics of a vehicle is the determination of the lateral force generated by the interaction between tires and soil. The underlying physical phenomenon is rather complex, and for its description, one often resorts to empirical models, the most renowned of which is undoubtedly the *Pacejka's magic formula* [9]. This is a nonlinear function supplying lateral force as a function of steering angle. The formula contains a number of parameters, the values of which have to be tuned to distinguish between different kinds of tires, with their own characteristics in terms of size, constitutive material, inflation pressure, deterioration, and so forth. To complete the model of the vehicle behavior, therefore, a further step is required, that of assigning a sensible value to the Pacejka tire parameters. In Pacejka's magic formula, parameters have no clear physical meaning; usually, they are estimated from experimental data.

During the car's life, however, the tire condition may change because of aging, the variability of the inflation pressure, or many other reasons (including the fact that the car owner may decide to

[§] $\|\cdot\|$ denotes some selected norm, possibly, but not necessarily, the Euclidean one.

DOI: 10.1002/acs

substitute the tires!). Therefore, to guarantee the accuracy of the model, the tire parameter value must be reestimated periodically with new fresh data. For instance, one may want to estimate the Pacejka parameters from measurements of the car's lateral acceleration as a response to the steering angle. Rather than solving a single estimation problem, the issue here is to set up an *estimator* that supplies reliable estimates of the Pacejka parameters for any type of tire and any type of operating condition (requirement no. 1). If such an estimator has to be embedded in an electronic device installed in the car, for example, as a part of the control unit, then the computational effort must be low because of the onboard limited computational resources (requirement no. 2) and the estimation algorithm must be fully automatic as no human supervision is allowed during car operation (requirement no. 3). □

In the literature, the problem of designing an estimator for θ based on the available mathematical model of $P(\theta)$ has been addressed many times according to a variety of paradigms. Among others, the most popular techniques are based on Kalman filtering (KF), [10–13], prediction error (PE), [4, 6, 8], or maximum likelihood (ML) [6, 14–16] methods. All these techniques are well understood by theorists and practitioners and have been proven to be effective in a huge number of applications. However, there are cases, such as the estimation of the Pacejka parameters, where the provided estimator may be not satisfactory with reference to the requirements given earlier. Therefore, further approaches are needed.

More specifically, one critical point of available approaches is that the estimator $\widehat{f}$ turns out to be *implicitly* defined by means of an estimation criterion, and hence, the generation of an estimate may be computationally demanding, unless a severe deterioration of accuracy is accepted. As a matter of fact, one of the remarkable contributions of the Swedish school of system identification was to recognize that existing paradigms could be cast in a unifying framework because they were based on the same basic ingredients: (i) data; (ii) model structure (here fixed being in a white-box setting); (iii) estimation criterion (implicitly defining the estimator); and (iv) estimation algorithm, where the latter is the numerical procedure implementing the chosen criterion, [6, 17, 18]. For example, in the PE approach, the estimation criterion is that of minimizing the empirical prediction error variance, whereas, consequently, the algorithm is the numerical optimization method. The same split between estimation criterion and algorithm applies for other paradigms too, and in all cases, generating estimates can be computationally costly (see the discussion in Section 2).

At a higher level of abstraction, one can also argue that available techniques, being so rigidly stuck to a given criterion, do not offer enough modularity to address a failure: if, for some reason, available paradigms provide estimators that are not satisfactory, then the designer has not enough degrees of freedom to modify the paradigms to cope with the problem at hand.

In this paper, we propose a new estimator design paradigm, the *two-stage* (TS) method. It allows all the requirements given earlier to be taken into account and naturally offers modularity similar to that offered to the designer in black-box identification. By means of these properties, we believe that the TS approach may be a valid alternative when other existing paradigms fail.

The main conceptual difference with existing paradigms is that, in the TS approach, there is no proper estimation criterion through which the estimator $\widehat{f}$ is defined. Rather, $\widehat{f}$ is empirically tuned through extensive simulation runs of the model of $P(\theta)$. In other words, the model of $P(\theta)$ is used to generate a number of simulated input/output sequences corresponding to different values of θ, and then, an estimator is empirically trained so as to perform well for these simulated data.[¶] In this way, the entire computational burden is relegated to the offline training phase, and once the estimator has been obtained, it can be used basically with no computational cost.

Clearly, the training of the estimator is the most critical part of the approach, consisting of a high-dimensional, therefore ill-conditioned, curve fitting problem. The final goal can be effectively

[¶]It is perhaps worth noticing that, in standard approaches, the knowledge of the mathematical model of $P(\theta)$ is analytically exploited to, for example, compute the prediction error gradient, or the likelihood function, or a linear approximation of the model. In the two-stage (TS) paradigm, the model of $P(\theta)$ is exploited to simulate possible behaviors of the true system corresponding to different values of the unknown parameters. In this respect, the TS paradigm fully belongs to our *simulation-based* age, initiated by the advent of Monte Carlo methods.

DOI: 10.1002/acs

achieved by means of an intermediary step that aims to generate a set of *artificial data* from the input/ouput sequences. The final algorithm thus develops in two stages: the first one transfers the information contained in the input/ouput sequences into the artificial data, whereas the second one establishes the link between these artificial data and parameter θ.

1.1. Structure of the paper

The paper is organized as follows. First, traditional approaches to parameter estimation are briefly summarized in Section 2, and their advantages and drawbacks are identified. The new TS approach is then discussed in Sections 3 and 4. Section 5 presents a benchmark example allowing the comparison between different techniques, whereas Section 6 is devoted to the application of the TS approach to the concrete problem of parameter estimation in Pacejka's magic formula.

2. TRADITIONAL PARADIGMS FOR ESTIMATION

2.1. Kalman filter-based approaches

In Kalman filter-based methods [10–13], parameter θ is seen as a state variable by introducing an additional state equation of the type $\theta(k+1) = \theta(k)$ or $\dot{\theta}(t) = 0$, depending on whether time is discrete or continuous.[‖] Then, the estimation problem is reformulated as a state prediction problem, so that the function $\widehat{f}$ mapping the data into the estimate is *implicitly* defined by the Kalman filter equations.

As is well known, even if $P(\theta)$ were a linear system, the resulting prediction problem would be nonlinear because of the introduction of the additional state equation. Thus, one resorts to nonlinear KF.

The most common approaches are the extended Kalman filter (EKF) [3, 10, 12, 13] and the unscented Kalman filter (UKF) [13, 19–21]. The major issue of EKF and UKF is that an initial guess for the initial estimation error mean and covariance matrix must be supplied. The convergence of the parameter estimate is very sensitive to the tuning of such initialization, and there are celebrated (yet simple) examples showing the possible divergence/nonconvergence (see, e.g., [22] and the example in Section 5). In general, local convergence is achievable only, [22–26], and human intervention is typically required.

Another approach one can resort to is the so-called particle filter (PF). PF basically reconstructs the *a posteriori* probability distribution of θ by letting a cloud of possible parameter values evolve through the system equations, [13]. It has been proved in [27, 28] that, under suitable assumptions, the PF estimate converges towards the true parameter, and this is one reason for its increasing popularity. On the other hand, this estimation algorithm requires an intensive simulation of the model's evolution each time an estimate for the unknown parameter vector has to be generated; therefore, it is computationally demanding.

2.2. The prediction error paradigm

In the PE approach, [6], the loss function

$$V(\theta) = \sum_{i=1}^{N} \left(\bar{y}(i) - \widehat{y}(i,\theta)\right)^2$$

is considered, where $\widehat{y}(i,\theta)$ is a predictor of the system output derived through the model equation for $P(\theta)$ and the available input/output data up to time $i-1$. The estimate of θ is obtained by minimizing $V(\theta)$, viz.

$$\widehat{\theta} = \arg\min V(\theta),$$

[‖]Often the additional equation takes the form $\theta(k+1) = \theta(k) + w(k)$ or $\dot{\theta}(t) = w(t)$, where w is white noise with suitable variance, so as to improve the reactivity of the algorithm.

DOI: 10.1002/acs

so that the estimator $\widehat{f}$ mapping observations into estimates is *implicitly* defined by the optimization problem itself. This latter is typically tackled by resorting to gradient-like methods. Although PE has become the mainstream in black-box identification problems, it applies to white-box identification as well, with no conceptual twisting.

The PE paradigm has been around for decades and has been analyzed in great detail. Its main advantages are its solid theoretical background for accuracy analysis and its general applicability. With regards to the latter, it should be observed that the gradient of the prediction error can be computed with generality once a model of the plant is available, possibly by numerical approximations.

The main issue of the PE paradigm, instead, is that $V(\theta)$ is typically a nonlinear nonconvex function with many local minima that may trap the numerical solution far away from the global minimizer, [29, 30]. Ignoring this problem would lead to biased (inconsistent) estimates. Hence, minimization is typically carried out by means of multiple attempts, that is, by running the gradient-like method many times with different initializations chosen from a grid in the parameter space and then by choosing the estimate that gives the smallest value for the loss function. As is clear, the finer the grid, the better the chance to converge towards the global minimizer; however in this case, the procedure may be computationally demanding, see also [8].

2.3. Indirect inference

Indirect inference (II) is an approach to parameter estimation initially developed in econometrics, [31–33]. The idea is to introduce an intermediate class of simple black-box models $Q(\beta)$, characterized by a vector β of parameters, along with an identification algorithm that permits the model $Q(\beta)$ to be fitted to the u, y data. In particular, $\beta(\bar{D}^N)$ is the parameter obtained from the experimental data $\bar{D}^N = \{\bar{y}(1), \bar{u}(1), \bar{y}(2), \bar{u}(2), \ldots, \bar{y}(N), \bar{u}(N)\}$. The II estimate, then, is defined as the value $\widehat{\theta}$ such that the parameter vector β identified from data obtained by simulating $P(\widehat{\theta})$ is as close as possible to $\beta(\bar{D}^N)$.

More precisely, let $\bar{\beta}(\theta) = E[\beta(D^N(\theta))]$ be the mean of the intermediate model parameter vector identified when data are generated according to the mathematical model of $P(\theta)$ with a generic θ. Then, the following loss function is introduced:

$$V(\theta) = (\beta(\bar{D}^N) - \bar{\beta}(\theta))^T W(\beta(\bar{D}^N) - \bar{\beta}(\theta)),$$

where W is a weighting matrix. The II estimate of the true plant parameter is obtained as the value of θ minimizing the loss, viz.

$$\widehat{\theta} = \arg\min V(\theta).$$

In practice, minimization is performed by resorting, for example, to gradient-based techniques, where the evaluation of $V(\theta)$, as well as of its gradient, for a given value of θ is performed via numerical simulations (that is, by generating a bunch of data sequence from the model of $P(\theta)$ and approximating $\bar{\beta}(\theta)$ with the empirical mean).

As it can be seen, II presents some similarities with the PE paradigm and, indeed, shares the same advantages and drawbacks with the PE approach.

2.4. Maximum likelihood

The ML approach is another well-known estimation method taken from statistics, [14, 15, 34]. ML amounts to computing the likelihood of possible values of θ given the observed data and then finding the maximum of such likelihood function. Again, the estimator $\widehat{f}$ turns out to be *implicitly* defined.

Maximum likelihood is a cornerstone achievement in the theory of estimation. However, it suffers in practice from the drawback that the reconstruction of the probability density of observed data as a function of the parameter θ is difficult unless few exceptional cases. Moreover, the maximization of the likelihood function presents the same criticality in terms of local optimal points as the PE paradigm, see [29, 30].

DOI: 10.1002/acs

Recently, an interesting ML approach based on particle filtering and the so-called expectation–maximization (EM) algorithm has been proposed in [16, 35]. This approach gets rid of some of the drawbacks of ML. However, its computational complexity may be critical.

3. THE TWO-STAGE PARADIGM

Motivated by the requirements identified in the introduction, we propose in this section a new estimator construction paradigm. Preliminary studies on this approach were presented in the conference papers [36–38].

3.1. Main idea

The idea behind the TS paradigm is that of resorting to offline intensive simulation runs to explicitly reconstruct the function $\widehat{f} : \mathbb{R}^{2N} \to \mathbb{R}^q$ mapping measured input/output data into an estimate for the parameter θ. To be precise, we use the *simulator* of the model of $P(\theta)$ to generate input/output data for a number of different values of the unknown parameter θ, chosen so as to densely cover a certain range of interest. That is, we collect N measurements

$$D_1^N = \{y^1(1), u^1(1), \ldots, y^1(N), u^1(N)\}$$

for $\theta = \theta_1$, N measurements

$$D_2^N = \{y^2(1), u^2(1), \ldots, y^2(N), u^2(N)\}$$

for $\theta = \theta_2$, and so forth, to work out a set of, say m, pairs $\{\theta_i, D_i^N\}$ as summarized in Table I. This set of data is referred to as the *simulated data chart*. See Section 4.1 for more details on the choice of the samples θ_i.

From the simulated data chart, the function $\widehat{f}$ is reconstructed as the map minimizing the estimation error over simulated data, that is,

$$\widehat{f} \leftarrow \min_f \frac{1}{m} \sum_{i=1}^{m} \left\| \theta_i - f(y^i(1), u^i(1), \ldots, y^i(N), u^i(N)) \right\|^2. \tag{1}$$

In other words, in the TS paradigm, there is no proper estimation criterion by means of which $\widehat{f}$ is defined. Simply, $\widehat{f}$ is constructed so as to work well on simulated experiments, relying then on generalization properties of data-based approaches such as (1) for achieving a good performance over all situations, including unforeseen ones (see Sections 4.1 and 4.3). Once $\widehat{f}$ is found, then the true θ corresponding to a given experimental data sequence $\bar{D}^N = \{\bar{y}(1), \bar{u}(1), \ldots, \bar{y}(N), \bar{u}(N)\}$ is estimated as

$$\widehat{\theta} = \widehat{f}(\bar{y}(1), \bar{u}(1), \ldots, \bar{y}(N), \bar{u}(N)).$$

Clearly, the minimization in (1) is a formidable task, requiring suitable algorithmic tricks to be accomplished. A two-stage algorithm (from which the name of the approach derives) is discussed next.

Table I. The simulated data chart as the starting point of the two-stage method.

θ_1	$D_1^N = \{y^1(1), u^1(1), \ldots, y^1(N), u^1(N)\}$
θ_2	$D_2^N = \{y^2(1), u^2(1), \ldots, y^2(N), u^2(N)\}$
$\vdots$	$\vdots$
θ_m	$D_m^N = \{y^m(1), u^m(1), \ldots, y^m(N), u^m(N)\}$

DOI: 10.1002/acs

3.2. The TS algorithm

As is clear, solving (1) requires the preliminary choice of a class $\mathcal{F}$ of functions among which optimization is performed. This is indeed a critical issue because of the high dimensionality of the problem (f depends upon $2N$ variables, normally a very large number if compared with the number m of experiments), and the notorious bias versus variance dilemma [6] arises: if $\mathcal{F}$ is a class of low-complexity functions, then it is difficult to replicate the relationship linking D^N to θ for all values of θ (bias error); conversely, if $\mathcal{F}$ is a class of high-complexity functions, then overfitting adversely affects the solution (variance error).

To achieve a sensible compromise between bias and variance, the TS approach is proposed. In this method, the selection of the family of functions $\mathcal{F}$ is split in two steps. This splitting is the key to select a proper family and, in turn, to obtain a good estimator $\widehat{f}$.

To be more precise, the objective of the first step is to reduce the dimensionality of the problem, by generating a new data chart composed of *m short* sequences, each with $n \ll N$ points. We will call such sequences *compressed artificial data sequences* and the corresponding chart the *compressed artificial data chart*. In the second step, the map between these artificial observations and the parameter θ is identified. By combining the results of the two steps, the estimator $\widehat{f}$ is finally retrieved.

In what follows are more details on each of the two stages.

First stage. The first step consists of compressing of the information conveyed by input/output sequences D_i^N to obtain new data sequences $\widetilde{D}_i^n$ of reduced dimensionality. Whereas in the data D_i^N the information on the unknown parameter θ_i is scattered in a long sequence of N samples, in the new compressed artificial data $\widetilde{D}_i^n$ such information is contained in a short sequence of n samples ($n \ll N$). This leads to a new compressed artificial data chart constituted by the pairs $\{\theta_i, \widetilde{D}_i^n\}$, $i = 1, \ldots, m$, see Table II.

Each compressed artificial data sequence $\widetilde{D}_i^n$ can be derived from D_i^N by resorting to standard identification procedures. That is, one fits a simple model to each sequence $D_i^N = \{y^i(1), u^i(1), \ldots, y^i(N), u^i(N)\}$ and then takes the parameters of this model, say $\alpha_1^i, \alpha_2^i, \ldots, \alpha_n^i$, as compressed artificial data, that is, $\widetilde{D}_i^n = \{\alpha_1^i, \ldots, \alpha_n^i\}$.

To fix ideas, we suggest the following as a typical method. For each $i = 1, 2, \ldots, m$, the data sequence

$$D_i^N = \{y^i(1), u^i(1), \ldots, y^i(N), u^i(N)\}$$

is concisely described by an ARX model:

$$\begin{aligned} y^i(t) = {} & \alpha_1^i y^i(t-1) + \cdots + \alpha_{n_y}^i y^i(t-n_y) \\ & + \alpha_{n_y+1}^i u^i(t-1) + \cdots + \alpha_{n_y+n_u}^i u^i(t-n_u), \end{aligned}$$

with a total number of parameters $n = n_y + n_u$. The parameters $\alpha_1^i, \alpha_2^i, \ldots, \alpha_n^i$ of this model can be worked out by means of the least squares algorithm ([4, 6]),

Table II. The compressed artificial data chart.

θ_1	$\widetilde{D}_1^n = \{\alpha_1^1, \ldots, \alpha_n^1\}$
θ_2	$\widetilde{D}_2^n = \{\alpha_1^2, \ldots, \alpha_n^2\}$
$\vdots$	$\vdots$
θ_m	$\widetilde{D}_m^n = \{\alpha_1^m, \ldots, \alpha_n^m\}$

DOI: 10.1002/acs

$$\begin{bmatrix} \alpha_1^i \\ \vdots \\ \alpha_n^i \end{bmatrix} = \left[\sum_{t=\bar{n}+1}^{N} \varphi^i(t)\varphi^i(t)^T \right]^{-1} \cdot \sum_{t=\bar{n}+1}^{N} \varphi^i(t) y^i(t), \tag{2}$$

$$\varphi^i(t) = [y^i(t-1) \dots y^i(t-n_y)\, u^i(t-1) \dots u^i(t-n_u)]^T, \quad \bar{n} = \max(n_u, n_y),$$

and are used as compressed artificial data. It is worth noticing that the simple model class selected to produce the compressed artificial data does not have any physical meaning and it must not be thought of as a mathematical model for the plant $P(\theta)$; this class plays a purely instrumental and intermediary role in the process of exposing the hidden relationship between the unknown parameter and the original collected data. Hence, it does not matter if the ARX models do not tightly fit data sequences D_i^N; what really matters is that $\alpha_1^i, \alpha_2^i, \dots, \alpha_n^i$ capture the variability of the θ_i. In this connection, it should be observed that the choice of the ARX model order (which must be the same for all the data sequences in the simulated data chart) is not very critical and it can be performed by successive trials (see also next Section 4.3 for further details).

In conclusion, the first stage of the method aims to find a function $\widehat{g} : \mathbb{R}^{2N} \to \mathbb{R}^n$ transforming each simulated data sequence D_i^N into a new sequence of compressed artificial data $\widetilde{D}_i^n$ conveying the information on θ_i. As compressed artificial data, we take the parameters of a simple model, identified from D_i^N. In this way, function $\widehat{g}$ is defined by the chosen class of simple models together with the corresponding identification algorithm.

Second stage. Once the compressed artificial data chart in Table II has been worked out, the problem becomes that of finding a map $\widehat{h} : \mathbb{R}^n \to \mathbb{R}^q$ that fits the m compressed artificial observations into the corresponding parameter vectors, that is,

$$\widehat{h} \leftarrow \min_h \frac{1}{m} \sum_{i=1}^{m} \left\| \theta_i - h(\alpha_1^i, \dots, \alpha_n^i) \right\|^2. \tag{3}$$

Function minimization in (3) is reminiscent of the original minimization problem in (1). However, being n small, the bias versus variance error trade-off is no longer an issue, and it is possible to resort to one of the many methods of function fitting available in the literature.

As for the choice of h, it is possible to, for example, select a linear function: $h(\alpha_1^i, \dots, \alpha_n^i) = c_1\alpha_1^i + c_2\alpha_2^i + \dots + c_n\alpha_n^i$, $c_i \in \mathbb{R}^q$, that is, each component of h is just a linear combination of the compressed artificial data $\alpha_1^i, \alpha_2^i, \dots, \alpha_n^i$. The parameters c_i appearing here can then be easily computed via least squares at a low computational cost. Of course, such a way of parameterizing h is computationally cheap but potentially loose. Better results are expected by choosing a class of nonlinear functions, such as neural networks or nonlinear autoregressive exogenous (NARX) models. The minimization in (3) can be performed by resorting to standard algorithms developed for these classes of nonlinear functions.

3.3. Use of the TS method

The TS method is based on two functions: $\widehat{g}$ and $\widehat{h}$. The former is the *compression function*, transforming simulated data into compressed artificial data. The latter is the *fitting function*, providing the map from the compressed artificial data to the unknown parameter. Whereas $\widehat{g}$ is chosen by the designer by selecting the intermediary identification algorithm in the first stage, in the second stage, instead, the designer chooses a suitable class of functions, and $\widehat{h}$ is identified by fitting the extracted parameter values to the corresponding compressed artificial data.

Once $\widehat{g}$ and $\widehat{h}$ are available, the estimator $\widehat{f}$ mapping input/output data into the estimate for θ is given by $\widehat{h} \circ \widehat{g} = \widehat{h}(\widehat{g}(\cdot))$, that is, by the composition of $\widehat{g}$ and $\widehat{h}$, as shown in Figure 2. In this way, $\widehat{f}$ is explicitly given. When a new input/ouput sequence, say $\bar{D}^N = \{\bar{y}(1), \bar{u}(1), \dots, \bar{y}(N), \bar{u}(N)\}$, is observed from the real plant, the corresponding unknown parameter θ is simply estimated as $\widehat{\theta} = \widehat{h}(\widehat{g}(\bar{D}^N))$.

DOI: 10.1002/acs

Figure 2. The estimator function composition.

4. COMPLEMENTARY DISCUSSIONS ON THE TS APPROACH

In this section, we provide some discussions about implementation aspects as well as properties and advantages of the TS approach.

4.1. Sampling issues

The first implementation choices the user is required to perform is the sampling of the parameter vector θ. The idea is to randomly extract values for the unknown parameter vector θ over a range of interest. Although, in general, uniform distribution can be considered, other probability distributions can be used if some *a priori* information is available (see, e.g., [39] for algorithms to perform random extractions from various probability distribution).

The issue, then, is how many samples need to be randomly extracted to ensure a given accuracy, that is, to ensure that the empirical cost

$$\frac{1}{m}\sum_{i=1}^{m}\|\theta_i - h(\alpha_1^i,\ldots,\alpha_n^i)\|^2$$

is close enough to its probabilistic counterpart

$$E\|\theta - h(\alpha_1,\ldots,\alpha_n)\|^2.$$

If this was the case, indeed, the map reconstructed by optimizing the empirical cost would be satisfactory not only for the extracted θ_i but also for other, unseen, instances of θ. In the context of the TS paradigm, the easiest way is to resort to cross-validation to assess the accuracy of the obtained estimator *a posteriori* (see Section 4.3 for details).

It is perhaps worth remarking, however, that the convergence of the empirical cost to the probabilistic one does not depend on the dimensionality of θ, that is, convergence does not suffer from the curse of dimensionality. The reason is the same why Monte Carlo methods are successful in computing multiple integrals: the overall effort for computing an integral in high dimensional spaces is the same as for a one-dimensional integral. This is the magic of randomization, [39, 40].

Another issue in the first step of the TS approach regards the input sequences to be chosen. It is clear that the estimator must be trained over input sequences that should resemble those arising in real experiments. In this respect, the following should be noted:

- In some cases, one knows that all the real experiments will be carried out with the same input or with input belonging to a given class (e.g., a white noise) from which it is easy to extract some representative realizations. In these cases, the input selection is dictated by the problem itself.
- The method presents a sort of robustness to variation in the input signal thanks to the first step. Indeed, in such a step, an intermediate model is built aiming at reproducing part of the system dynamics. Such dynamics is input independent in a number of cases.

4.2. Computational complexity in the TS paradigm

As is clear, the training of the TS estimator relies on an intensive simulation of the plant and on the processing of the simulated data chart. This indeed may result in a high computational burden, which however has to be tackled offline, before that any real data sequence is seen (see also Remark 1). In other words, the computational burden is relegated to the estimator training phase only, where one tries to adapt the behavior of $\widehat{f}$ to as many situations as possible. Once training is terminated,

DOI: 10.1002/acs

the final product of the TS paradigm is an estimator $\widehat{f}$ that is explicitly given. Hence, when real data sequences enter the stage, the estimates are generated at extremely low computational cost, by evaluating $\widehat{f}$ in correspondence with the measured data sequences, and $\widehat{f}$ can be used over and over without human supervision.

This is in contrast to other estimation paradigms, where, $\widehat{f}$ being implicitly defined, each generation of an estimate requires a computationally demanding data processing, which sometimes need to be performed under the designer's supervision.

Remark 1 (Loss function minimization in the TS and other approaches)
The minimization problem in (3) may present multiple minima as in the minimization of the loss function in the PE/ML/II approaches. Hence, at a first glance, it may seem that the TS approach is as critical as PE/ML/II. Evidently, this is not so, and indeed, there are two substantial differences with other approaches that it is worth clarifying.

(1) In the PE/ML/II, minimization has to be performed each time an estimate has to be generated from available data. This leads to high computational effort. In the TS approach, instead, minimization is performed once and for all during the estimator training and before the estimator is used with actual data. Estimates are therefore generated at very low computational cost.
(2) A common aspect of PE, ML, and II is that optimization is performed over the space of the original parameter θ. Therefore, to retrieve the unknown plant parameter, the *global* minimum point must be found. Indeed, a local minimum point can be very far away from the true parameter, even if the corresponding cost is close to that associated to the global minimum. The risk of obtaining a mistaken estimate cannot be neglected.

    In the TS approach, instead, local minima are not an issue. The reason is that the optimization is not performed over the original parameter space but over the parameter space of the class of functions (e.g., the neural network class) used in the second stage. The location of minima in the neural network parameter space has nothing to do with the location of minima in the original parameter space. What really matters is the value taken by the performance index in (3) in correspondence with the chosen neural network. In this regard, a low value of the cost index can be achieved even in local minima. On top of that, such a value is known to the user, whereas in the PE/ML/II approaches, once one reaches a local minimum point, there is no way to assess its distance from the global minimum point. The only remedy in PE/ML/II frameworks is to resort to intensive computational trials in the hope of spotting the global minimum.

4.3. Cross-validation and modularity of the TS paradigm

In the construction of the TS estimator, a number of choices are left to the user. Among others, these include the choice of the number N of simulations for training, the choice of the model class (including its order) to be used in the first step, and the choice of the class of nonlinear maps to be used in the second step. This degree of freedom is at the same time the bane and delight of the TS approach. Although, on the one hand, there are no simple recipes to follow, on the other hand, it supplies a great flexibility that permits one to cope with a large variety of situations. In some sense, it is the same degree of freedom we have in black-box identification [6]: the choice of the model class as well as other choices is definitively left to the user, depending on the problem at hand.

In this respect, it is perhaps worth noticing that, because the training is performed via simulation, before any real data sequence is seen, the user can intensively rely on cross-validation as a means to test his or her own specific choices. This is very different from the standard identification setting where cross-validation requires collecting new real data points and hence can be very costly. In the TS approach, cross-validation merely requires the extraction of some extra values θ_1^{CV}, $\theta_2^{CV}, \ldots, \theta_r^{CV}$ for the unknown parameter vector and the generation *via simulation* of the corresponding input/output data sequences $D_i^{CV} = y^{i\,CV}(1), u^{i\,CV}(1), y^{i\,CV}(2), u^{i\,CV}(2), \ldots, y^{i\,CV}(N), u^{i\,CV}(N)$, $i = 1, 2, \ldots, r$. This cross-validation data record has to be used once the training has been completed and an estimator $\widehat{f}$ has been obtained from the simulated data chart.

DOI: 10.1002/acs

One can compute $\widehat{f}(D_i^{CV})$, $i = 1, 2, \ldots, r$ and compare it with θ_i^{CV} by, for example, inspecting the empirical mean square error $\frac{1}{r}\sum_{i=1}^{r}\|\theta_i^{CV} - \widehat{f}(D_i^{CV})\|^2$. Interestingly enough, the empirical mean square error thus computed can be used to assess the actual mean square error $E\|\theta - \widehat{f}(D^N)\|^2$ provided that r is suitably chosen as specified by the following theorem (see, for example, [39]).

Theorem 1 (Hoeffding)
Assume that $\|\theta - \widehat{f}(D^N)\|^2 \leqslant \sigma$. Fix two real numbers $\gamma > 0$ and $\delta > 0$. If

$$r > \frac{\sigma^2}{2\gamma^2}\ln(1/\delta), \tag{4}$$

then we have that

$$E\|\theta - \widehat{f}(D^N)\|^2 \leqslant \frac{1}{r}\sum_{i=1}^{r}\|\theta_i^{CV} - \widehat{f}(D_i^{CV})\|^2 + \gamma,$$

with probability greater than or equal to $1 - \delta$.

Theorem 1 says that $E\|\theta - \widehat{f}(D^N)\|^2$ can be approximated by $\frac{1}{r}\sum_{i=1}^{r}\|\theta_i^{CV} - \widehat{f}(D_i^{CV})\|^2$ with arbitrary precision as long as r is sufficiently high. Note that the theorem statement holds true with a certain probability $1 - \delta$ only. This is a consequence of the fact that $\frac{1}{r}\sum_{i=1}^{r}\|\theta_i^{CV} - \widehat{f}(D_i^{CV})\|^2$ is a random element depending on the extracted θ_i^{CV}. Hence, the mismatch between the empirical and the actual mean square error can be smaller than γ for some extractions and not for others, and δ refers to the probability of extracting *bad* θ_i^{CV}. It is, however, important to note that r depends on δ logarithmically so that a very small value of δ can be forced in without significantly affecting r. As regards σ, it should be noted that it can be easily computed whenever θ takes value in a compact domain.

Finally, it should be noted that, although cross-validation can be used to assess the performance of other estimation approaches, in the TS paradigm if cross-validation reveals that the obtained $\widehat{f}$ is not satisfactory, the designer may go over the training phase to improve the performance of $\widehat{f}$ by suitably modifying the choices previously made, for example, by increasing N, changing the order n of ARX models in the first stage, or modifying the complexity of the neural network in the second stage by adding or removing neuron layers. One can even realize that ARX models in the first stage are not enough to distinguish between the dynamic properties associated with distinct parameters (this may be the case for highly nonlinear systems) and can switch to NARX models, for example, as well as to other models. In this situation, the high modularity of the TS paradigm is appreciated in that the same high-level general idea can be applied irrespective of the low-level implementation details. The designer can try his or her preferred methods for the first stage and the second one over and over until a satisfactory $\widehat{f}$ is obtained.

4.4. Comparison with indirect inference

Although both the TS and II approaches rely on an intermediate identification step, they are *completely different* from a conceptual point of view. Indeed, in the II approach, there is no attempt to directly reconstruct (via *samples*) the map linking data sequences to parameters. The direct reconstruction of such a map is the core of the TS approach, and it permits the generation of estimates at low computational cost. The intermediate step in the TS approach is merely a (necessary) technicality to ease the map reconstruction problem that otherwise would be intractable. In this respect, the identification of the intermediate model is performed with the aim of compressing the information only.

As already discussed in Section 2.3, the role of the intermediate identification in the II approach is quite different. In particular, in this approach, the estimator of the unknown parameter θ turns out to be implicitly defined by the minimization of a loss function, with potentially many local minima. This makes the computation of the estimate extremely costly.

DOI: 10.1002/acs

5. A BENCHMARK EXAMPLE

In this section, a simple example is presented to allow for a sharper comparison between different approaches. The following data generation mechanism is considered:

$$x_1(k+1) = \frac{1}{2}x_1(k) + u(k) + v_{11}(k), \tag{5a}$$

$$x_2(k+1) = (1-\theta^2)\sin(50\,\theta^2)\cdot x_1(k) - \theta\cdot x_2(k) + \frac{\theta}{1+\theta^2}\cdot u(k) + v_{12}(k), \tag{5b}$$

$$y(k) = x_2(k) + v_2(k), \tag{5c}$$

where θ is an unknown real parameter in the range $[-0.9, 0.9]$ and $v_{11} \sim \mathrm{WGN}(0,1)$, $v_{12} \sim \mathrm{WGN}(0,1)$, and $v_2 \sim \mathrm{WGN}(0, 0.1)$ (WGN = white Gaussian noise) are mutually uncorrelated noise signals. In all our experiments, system (5) was initialized with $x_1(0) = 0 = x_2(0)$.

To test the behavior of various estimation approaches, we extracted 200 values for the parameter θ uniformly in the interval $[-0.9, 0.9]$, and for each extracted value of θ, we generated $N = 1000$ observations of the output variable y associated with an input u generated as $\mathrm{WGN}(0,1)$ uncorrelated with the disturbances. Each time, the $N = 1000$ pairs of input/output observations were made available to the estimation algorithms, each of which returned an estimate of the corresponding θ. Thus, for each estimation algorithm, we obtained 200 estimates $\widehat{\theta}$, which then were compared with the corresponding 200 true values of θ.

We will give a graphical visualization of such a comparison by plotting the obtained estimates against the true parameter values. In other words, for each point in the figures later, the x-axis is the extracted value for θ, whereas the y-axis is the corresponding estimate $\widehat{\theta}$ supplied by the implemented estimation method. Of course, a good estimator must return points displaced nearby the bisector of the first and third quadrants.

All simulations were performed in the `Matlab` environment by means of a standard 2.40 GHz dual-processor computer.

5.1. *Kalman filters*

To apply both EKF and UKF, system (5) was rewritten as

$$\begin{aligned}
x_1(k+1) &= \frac{1}{2}x_1(k) + u(k) + v_{11}(k),\\
x_2(k+1) &= (1-x_3(k)^2)\sin(50\,x_3(k)^2)\cdot x_1(k) - x_3(k)\cdot x_2(k) + \frac{x_3(k)}{1+x_3(k)^2}\cdot u(k) + v_{12}(k),\\
x_3(k+1) &= x_3(k) + w(k),\\
y(k) &= x_2(k) + v_2(k),
\end{aligned}$$

where x_3 is an additional state variable representing parameter θ. Herein, we will report the simulation results obtained by taking as $w(k)$ a $\mathrm{WGN}(0, 10^{-6})$.

For each extracted value of θ in the range $[-0.9, 0.9]$, the estimate was obtained as the one-step ahead prediction $\widehat{\theta} = \widehat{x}_3(1001|1000)$. Such a computation was carried over with the EKF, UKF, and PF algorithms.

Figures 3 and 4 display the result of EKF and UKF in different operating conditions. Specifically, Figure 3 depicts the results obtained with the following initialization: $\widehat{x}_1(0) = \widehat{x}_2(0) = 0$, $\widehat{x}_3(0) = 0$, and

$$P(0) = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{6}$$

DOI: 10.1002/acs

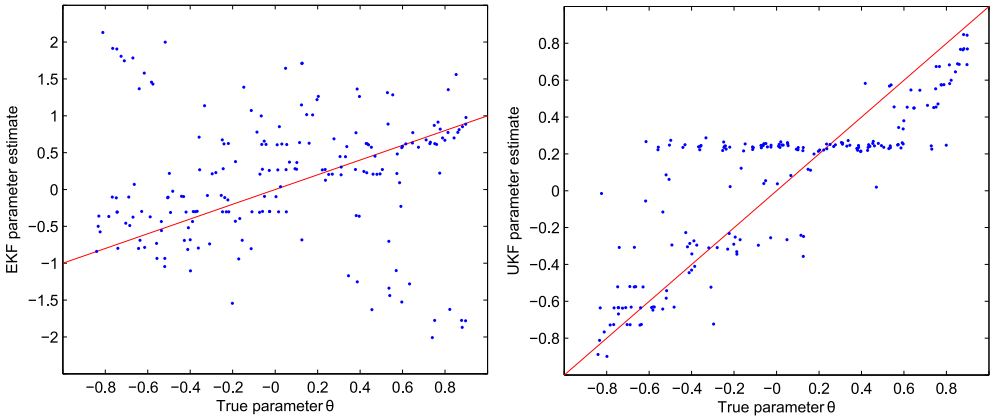

Figure 3. Estimates of θ versus true parameter values (large initial variance)—EKF on the left, UKF on the right.

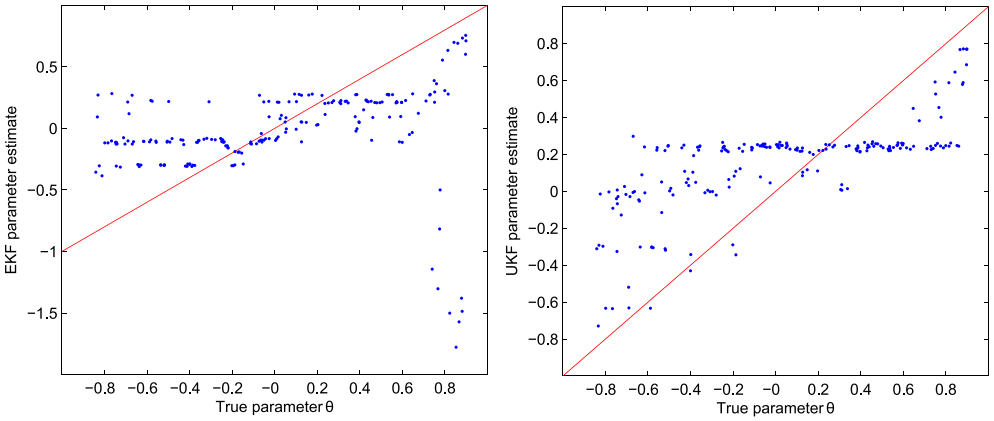

Figure 4. Estimates of θ versus true parameter values (small initial variance)—EKF on the left, UKF on the right.

($P(0)$ is the initial covariance of the estimation error). Figure 4, instead, displays the results obtained when

$$P(0) = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 10^{-2} \end{bmatrix}. \tag{7}$$

With regards to the computational complexity, EKF took about 11 s to return the whole 200 estimates (with an average time of 0.055 s per estimate), whereas UKF required a total of about 200 s (with an average time of 1 s per estimate).

As it appears, the EKF and UKF behaviors are quite different from the optimal expected behavior. In many instances, the estimate does not converge to the true value of θ. Furthermore, the estimator behavior strongly depends on the choice of $P(0)$, and anyhow, local convergence can be achieved at most.**

As regards the PF, the obtained results with 1000 particles are shown in Figure 5.

As can be seen, PF provides more satisfactory estimates. Although the performance can be further improved by increasing the number of particles, the actual bottleneck of the PF estimation method

**Perhaps it is worth noticing that further simulations were performed by changing the initialization of $\widehat{x}_3(0)$ (precisely, to -0.8, -0.3, 0.3, and 0.8), but such simulations are not reported because of space limitations. The results, however, were similar to those presented in this paper, and the drawn conclusions remain valid.

DOI: 10.1002/acs

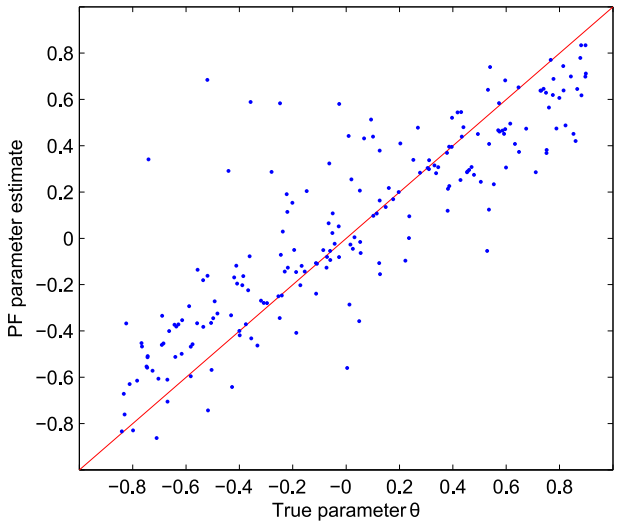

Figure 5. Estimates of θ vs. true parameter values for particle filter estimator.

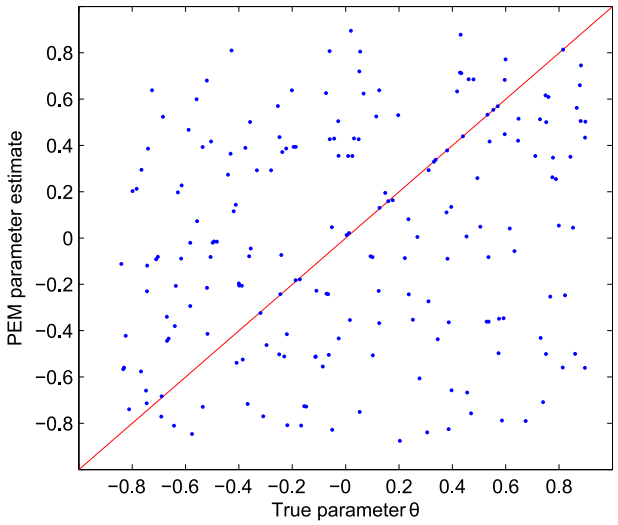

Figure 6. Estimates of θ versus true parameter values for PE estimator (one initialization only).

remains its computational complexity. Indeed, PF took 4675.89 s with an average time of 23.38 s per estimate.

5.2. *Prediction error approaches*

The PE estimation method was implemented by resorting to the `idgrey` models of the `Matlab System Identification Toolbox`, see [41].

Figure 6 depicts the results obtained with a single initialization of the PE algorithm obtained by choosing a value at random from the interval $[-0.9, 0.9]$ (note that initial states were known, $x_1(0) = 0 = x_2(0)$, and they needed not to be estimated). Overall, calculations took 18.23 s with an average of 0.09 s per estimate. The returned estimates are rather spread out, revealing the presence of many local minima trapping the PE solution far away from the true parameter value.

We, therefore, ran the PE algorithm with multiple initializations and chose the estimate returning the lowest loss. The results with five initializations and with ten initializations are shown in Figure 7.

DOI: 10.1002/acs

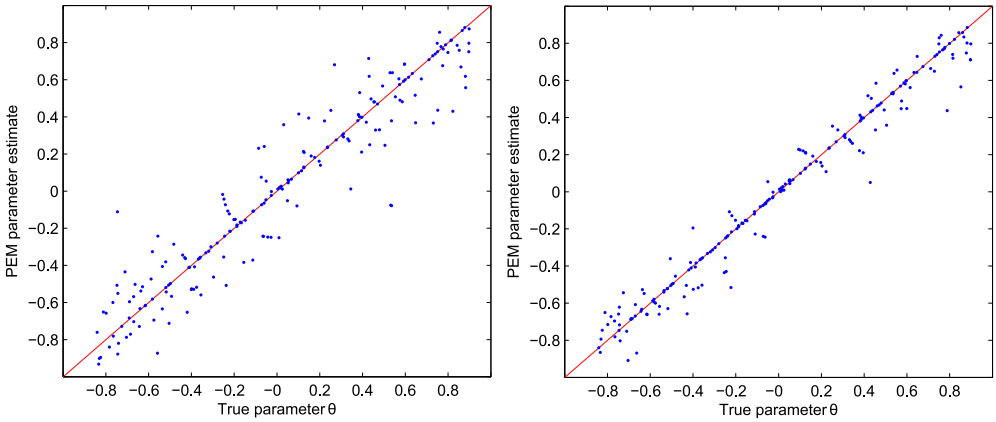

Figure 7. Estimates of θ versus true parameter values for PE estimator—five initializations on the left, 10 initializations on the right.

As it appears, the performance becomes better and better as the number of initializations is allowed to increase. However, the computational complexity increases too. The algorithm with five initializations required 94.28 s to generate the 200 estimates (with an average of 0.47 s per estimate), whereas the algorithm with 10 initializations took 179, 14 s (with an average of 0.9 s per estimate).

5.3. The two-stage approach

According to the discussion in Section 3, the TS estimator was obtained by means of the training over a set of simulation data. Once the training was terminated, the performance was tested against the same 200 experiments previously used for the other methods.

As regards the training, $m = 1500$ new values of θ were uniformly extracted from the interval $[-0.9, 0.9]$, and correspondingly, 1500 sequences of 1000 pairs of input/output values were simulated to construct the *simulated data chart*.

For each sequence $y^i(1), u^i(1) \ldots, y^i(1000), u^i(1000)$, $i = 1, \ldots, 1500$, the compressed artificial data sequence was obtained by identifying, using the least squares algorithm, the coefficients $\alpha_1^i, \ldots, \alpha_{10}^i$ of an ARX(5,5) model $(y^i(t) = \alpha_1^i y(t-1) + \cdots + \alpha_5^i y(t-5) + \alpha_6^i u(t-1) + \cdots + \alpha_{10}^i u(t-5))$. The final estimator $\widehat{h}(\alpha_1^i, \ldots, \alpha_{10}^i)$, instead, was computed by resorting to a neural network with 10 inputs $(\alpha_1^i, \ldots, \alpha_{10}^i)$ and one output $(\widehat{\theta})$. The network was a standard feed-forward neural network with two layers (ten neurons in the first layer and one neuron in the second one, which was also the output layer). The network was trained with the 1500 artificial observations by the usual back-propagation algorithm. Overall, the training took 31.45 s.

The obtained estimator was then applied to the 200 previously used data sequences.[††] Again, the returned 200 estimates $\widehat{\theta}$ were compared with the corresponding 200 values of θ. The TS estimators required 1.95 s to generate the whole 200 estimates (i.e., an average time of 0.01 s per estimate), while its performance can be seen in Figure 8.

6. THE PACEJKA'S PARAMETERS ESTIMATION PROBLEM

In this section, the TS approach is applied to a concrete problem, that of estimating the Pacejka's tire parameters from measurements of the car's lateral acceleration and steering angle. First, a model for the vehicle dynamics including interactions with tires is presented in Section 6.1. This will permit us to precisely formulate the estimation problem. The experimental results obtained by applying the TS method to this problem are reported then in Section 6.2. A comparison with the estimation results provided by the PF is also given.

[††]Perhaps it is worth stressing that this **200** values of θ and the corresponding data sequences were not used in the training phase of the TS approach.

DOI: 10.1002/acs

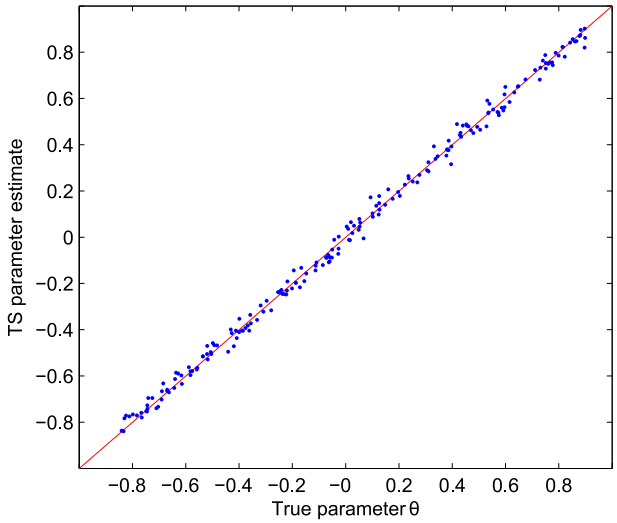

Figure 8. Estimates of θ versus true parameter values for the TS estimator.

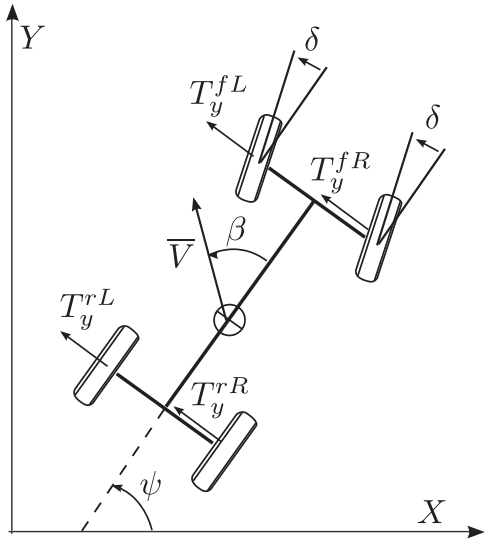

Figure 9. The car model.

6.1. *Car model description and problem formulation*

For simplicity, we will consider a car moving at constant speed V, and we will model the car's lateral dynamics only. In the sequel, x and y will denote the car's longitudinal and lateral axes, respectively, whereas X and Y form an inertial coordinate frame (Figure 9). Our model input is the steering angle δ, that is, the angle between the front wheels' longitudinal axis and the car's longitudinal axis (we suppose the steering angle is the same for both front wheels), whereas the output is the car body lateral acceleration denoted by a_y. $\overline{V}$ is the velocity vector in the inertial coordinate frame, whereas β is the car's sideslip angle (i.e., the angle between the car's longitudinal axis and $\overline{V}$) and ψ is the car's yaw angle (i.e., the angle between the car's longitudinal axis and the X axis).

Assuming that all angles are small enough to approximate function sin with its argument and cos with the constant 1, the model of the car's lateral dynamics is obtained as a force and moment equilibrium ([9]):

DOI: 10.1002/acs

$$\begin{cases} -mV(\dot{\beta}+\dot{\psi}) &= T_y^{fL}+T_y^{fR}+T_y^{rL}+T_y^{rR}, \\ J\ddot{\psi} &= (T_y^{fL}+T_y^{fR})\cdot l_f-(T_y^{rL}+T_y^{rR})l_r, \\ a_y &= -V(\dot{\beta}+\dot{\psi}), \end{cases} \tag{8}$$

where T_y denotes the lateral forces generated by one tire, whereas superscripts f, r, L, and R distinguish between *front*, *rear*, *Left*, and *Right* wheels. l_f and l_r indicate the distance between front and rear wheels and the car's center of mass. Finally, m and J denote the car mass and moment of inertia, respectively.

As for the lateral forces generated by tires, we resort to the Pacejka's magic formula [9]:

$$T_y = D\sin\{C\arctan[B\bar{\alpha}-E(B\bar{\alpha}-\arctan(B\bar{\alpha}))]\}+S_V,$$

where

$$\begin{aligned} S_H &= p_{H1}+p_{H2}f_z+p_{H3}\gamma, \\ S_V &= Q[(p_{V1}+p_{V2}f_z)+(p_{V3}+p_{V4}f_z)\gamma], \\ \bar{\alpha} &= \alpha+S_H, \\ C &= p_{C1}, \\ \mu &= (p_{D1}+p_{D2}f_z)(1-p_{D3}\gamma^2), \\ D &= \mu Q, \\ E &= (p_{E1}+p_{E2}f_z)[1-(p_{E3}+p_{E4}\gamma)\mathrm{sign}(\bar{\alpha})], \\ K &= p_{K1}F_z\sin[2\arctan(Q/(p_{K2}\cdot F_z))](1-p_{K3}|\gamma|)], \\ B &= K/(C\cdot D). \end{aligned}$$

In this formula, α is the wheel sideslip angle (i.e., the angle between the wheel longitudinal axis and the wheel velocity), which can be obtained as follows:

$$\begin{aligned} \alpha^f &= \beta+l_f\cdot\dot{\psi}/V-\delta, \\ \alpha^r &= \beta-l_r\cdot\dot{\psi}/V. \end{aligned}$$

Q is the vertical load acting on the wheel that can be computed by modeling the car roll dynamics to take into account the vertical load shift:

$$J_r\ddot{\vartheta}+C_r\dot{\vartheta}+K_r\vartheta = h\cdot m\cdot g\sin(\vartheta)-a_y\cdot h\cdot m\cdot\cos(\vartheta),$$

$$Q^R = N\frac{l/2-h\cos(\vartheta)}{l} \qquad Q^L = N\frac{l/2+h\cos(\vartheta)}{l},$$

where ϑ is the roll angle, g the gravitational acceleration, h the altitude of the center of mass, l the car semiaxis, $N = m\cdot g$ the car vertical load, and K_r, C_r and J_r are suitable constants.

Eventually, going back to the Pacejka's formula, γ is the so-called camber angle, here supposed to be constant, F_z the wheel nominal load (which is constant too), and $f_z = (Q-F_z)/F_z$ is the relative load. All other parameters are the so-called Pacejka tire parameters, and their value determines the tire response.

Summarizing, Pacejka's magic formula is merely a nonlinear function of variables α and Q supplying the generated lateral force T_y. Such function is parameterized by the vector of Pacejka parameter, say θ, so that $T_y = F_\theta(\alpha, Q)$. Altogether, the car's lateral dynamics can be modeled through a continuous time nonlinear system $P(\theta)$, depending on an uncertain parameter vector $\theta\in\mathbb{R}^q$. The system input u is the steering angle δ, and output y is the car's lateral acceleration a_y.

To retrieve the unknown value of the parameter vector θ, the system's input and output are observed through an angular position sensor and an accelerometer. The observations cover a certain time interval leading to a number N of input and output observations $\bar{D}^N = \{\bar{y}(1),\bar{u}(1),\ldots,\bar{y}(N),\bar{u}(N)\}$. The issue then is how to build a suitable parameter estimator, that is, a map $\widehat{f}: \mathbb{R}^{2N}\to\mathbb{R}^q$, which exploits the information conveyed by data $\bar{D}^N = \{\bar{y}(1),\bar{u}(1),\ldots,\bar{y}(N),\bar{u}(N)\}$ so as to produce fair estimates of the values taken by the uncertain parameter vector θ.

DOI: 10.1002/acs

6.2. *Experimental results*

The car model described in Section 6.1 has been implemented in `Dymola/Modelica` and then simulated through `Matlab`. We then applied the TS approach to construct an estimator able to retrieve the Pacejka tire parameters from the measurements of the car's lateral acceleration while the car, moving at constat speed $V = 10$ m/s, is steered along a chicane (left plot of Figure 10). This manoeuver lasts about 10 s. The corresponding lateral acceleration has been computed through the simulator. To take into account possible measurement errors, we added a zero mean white noise with suitable variance (0.01 m/s^2). A typical response of the lateral acceleration (for a particular type of tires) is shown in the right plot of Figure 10. For simplicity, we supposed that the Pacejka tire parameters were the same for the four tires of the car (i.e., tires are all of the same type with same aging and pressure condition) and that, among all Pacejka parameters, only p_{K1} and p_{K2} were subject to changeability (while all others were fixed). In other words, the vector θ to be estimated is constituted by $\theta_1 = p_{K1}$ and $\theta_2 = p_{K2}$. This choice was dictated by the fact that the calculation of lateral force by means of the Pacejka's magic formula is much more sensitive to parameters p_{K1} and p_{K2} than all other parameters. Typically, p_{K1} ranges between -42.83 and -27.32, whereas p_{K2} between -1.22 and -0.98.

To apply the TS method, $m = 2000$ values for θ were uniformly extracted from the rectangle $[-42.83, -27.32] \times [-1.22, -0.98]$, and correspondingly, we ran 2000 simulations of the car model, each time adopting the steering angle signal in Figure 10 as input. By sampling the input and output signals with a period of 0.02 s, we then obtained 2000 input/output sequences each 500 samples long ($N = 500$): $u^i(1), y^i(1), u^i(2), y^i(2), \ldots, u^i(500), y^i(500)$, $i = 1, 2, \ldots, 2000$. These sequences, together with the 2000 extracted values for θ, formed the simulated data chart.

For the generation of the compressed artificial data chart, an ARX(4,4) model was considered: $y^i(t) = \alpha_1^i y^i(t-1) + \cdots + \alpha_4^i y^i(t-4) + \alpha_5^i u^i(t-1) + \cdots + \alpha_8^i u^i(t-4)$. By performing the identification over the sequence $u^i(1), y^i(1), u^i(2), y^i(2), \ldots, u^i(500), y^i(500)$ for $i = 1, 2, \ldots, 2000$, the returned parameters $\alpha_1^i, \alpha_2^i, \ldots, \alpha_8^i$, $i = 1, 2, \ldots, 2000$, constituted the compressed artificial data chart. The specific choice of the order of ARX models was eventually obtained after few trial-and-error attempts.

The final estimator $\widehat{h}(\alpha_1^i, \alpha_2^i, \ldots, \alpha_8^i)$ was instead derived by resorting to a feed-forward two-layer neural network, with twenty neurons in the hidden layer and two linear neurons in the output layer. The network weights were trained by the usual back-propagation algorithm. Again, the order and the structure of the neural network were chosen by means of cross-validation.

The entire offline process for the training of the TS estimator took about 20 min on a standard 2.40 GHz dual-processor computer, and it produced an explicit estimator $\widehat{f} = \widehat{h} \circ \widehat{g}$ defined as the composition of the least squares algorithm and the trained neural network.

To test the performance of the obtained TS estimator, we applied it to 800 new (validation) data sequences, generated from 800 fresh values for parameter θ. Then, 800 estimates $\widehat{\theta}$ were computed

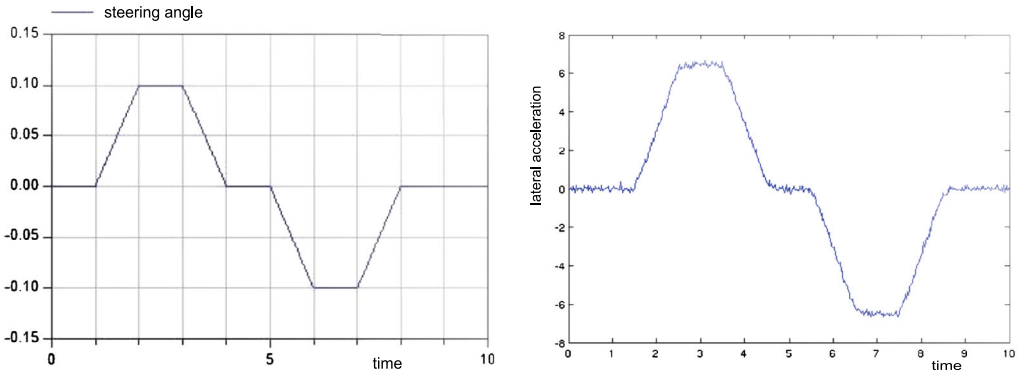

Figure 10. Standard steering angle trend (left) and corresponding lateral acceleration (right).

DOI: 10.1002/acs

by simply evaluating $\widehat{f}$ for each data sequence. We remark that the computation of the whole set of 800 estimates took 8.16 s only, that is, about 0.01 s per single estimate.

The performance of the TS estimator can be appreciated in Figure 11, where the estimates $\widehat{\theta}_1$ and $\widehat{\theta}_2$ of the first and second parameters are compared with the true values of θ_1 and θ_2. As in previous sections, the x-axis depicts the value of the first (second) parameter, whereas the y-axis provides the returned estimate. To simplify the visualization of the results, we have drawn the bisector (dashed line) and a continuous line representing an optimal linear fit of the points in the graph.

As it appears, the TS estimator supplies fairly accurate estimates, especially for the first parameter. In this respect, we note that the first parameter is one order of magnitude more important than the second one in determining the lateral force supplied by the Pacejka's magic formula, and this reflects into the better identifiability of θ_1.

For the sake of comparison, we also applied the PF estimation algorithm to the first 50 data sequences out of the 800 we used in the TS approach. The estimates were computed by allowing a cloud of 500 particles to evolve through the filter equations.

Figure 12 plots the estimates $\widehat{\theta}_1$ and $\widehat{\theta}_2$ returned by PF versus the true values of the parameters. The PF estimates are much more scattered around the bisector. In addition, the PF algorithm required 5 h to compute a *single estimate* (the computation of the whole 50 estimates took about 250 h). This is in striking contrast with the few seconds required by the TS estimator for computing the entire 800 estimates.

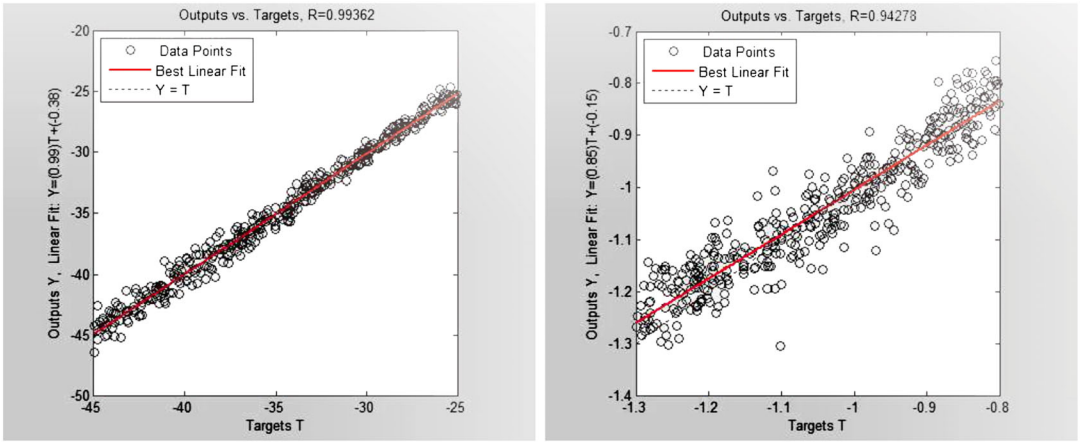

Figure 11. TS estimator: parameter θ_1 estimates versus true values (left) and parameter θ_2 estimates versus true values (right).

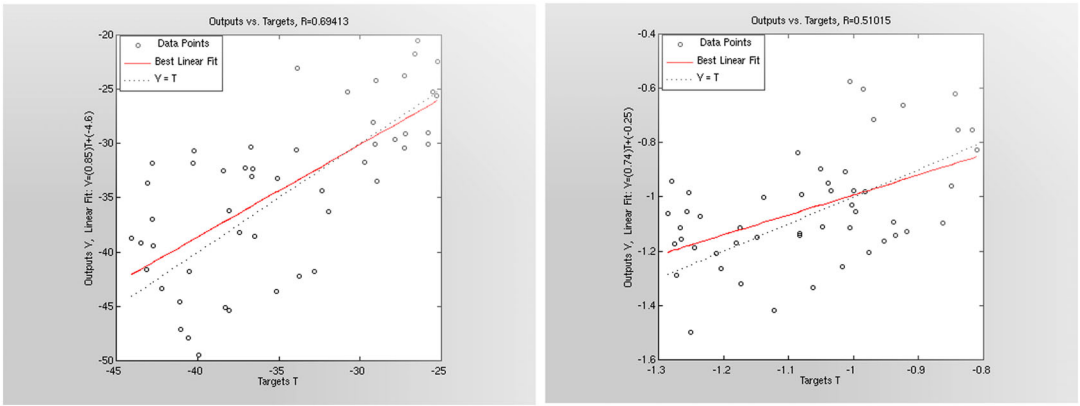

Figure 12. Particle filter estimator: parameter θ_1 estimates versus true values (left) and parameter θ_2 estimates versus true values (right).

DOI: 10.1002/acs

7. CONCLUSIONS

Parameter estimation in a given model is one of the most common problems to be tackled in the practice of system identification. Available methods (EKF, UKF, PF, ML, PEM, etc.) have been extensively studied in the literature, especially in relation to accuracy analysis; however, other aspects, such as computational complexity, have been somewhat overlooked. When such methods are used in practice, remarkable drawbacks may be encountered, and parameter estimation is often performed *de facto* using empirical procedures.

The TS approach provided in this paper tries to bridge this gap between the theory and the practice of estimation, by providing a method that is based on a general, technically sound, principle, which is also flexible enough to account for various requirements (accuracy, time complexity, etc.). This paper sheds light on the general idea behind the method. Admittedly, we do not provide here a formal theoretical investigation that could hopefully be the subject of future research.

ACKNOWLEDGEMENTS

This paper is supported by the MIUR national project *Identification and adaptive control of industrial systems* and by CNR-IEEIT. The authors are grateful to Dynasim for the support and to Carlo Sandroni for performing the simulations in Section 6. We are also grateful to anonymous reviewers for interesting comments and suggestions.

REFERENCES

1. Westcott JH. The parameter estimation problem. In *Proceedings of the 1st IFAC World Congress*, Moscow, Russia, 1960.
2. Mayne DQ. Parameter estimation. *Automatica* 1966; **3**:245–255.
3. Gelb A, Kasper Jr JF, Nash Jr RA, Price CF, Sutherland Jr AA. *Applied Optimal Estimation*. MIT press: Cambridge, MA, 1974.
4. Söderström T, Stoica P. *System Identification*. Prentice-Hall: Englewood Cliffs, NJ, 1989.
5. Bittanti S, Picci G (eds). *Identification, Adaptation, Learning – The Science of Learning Models from Data*. Springer-Verlag: Berlin, Gemrany, 1996.
6. Ljung L. *System Identification: Theory for the User*. Prentice-Hall: Upper Saddle River, NJ, 1999.
7. Kailath T, Sayed AH, Hassabi B. *Linear Estimation*. Prentice-Hall: Upper Saddle River, NJ, 2000.
8. Bohlin T. *Practical Grey-box Identification: Theory and Applications*. Springer-Verlag: London, UK, 2006.
9. Pacejka HB. *Tire and Vehicle Dynamics*. Elsevier: Oxford, UK, 2005.
10. Anderson BDO, Moore JB. *Optimal Filtering*. Prentice Hall: Englewood Cliffs, NJ, 1979.
11. Su JK, Kamen EW. *Introduction to Optimal Estimation*. Springer: Englewood Cliffs, NJ, 1999.
12. Grewal MS, Andrews AP. *Kalman Filtering – Theory and Practice using Matlab*. John Wiley & Sons: New York, NY, 2001.
13. Simon D. *Optimal State Estimation*. John Wiley & Sons: Hoboken, NJ, 2006.
14. Åström KJ. Maximum likelihood and prediction error methods. *Automatica* 1980; **16**:551–574.
15. Hannan EJ, Deistler M. *The Statistical Theory of Linear Systems*. John Wiley and sons: New York, NY, 1988.
16. Schon T, Wills A, Ninness B. System identification of nonlinear state-space models. *Automatica* 2011; **37**(1):39–49.
17. Ljung L. Perspective on system identification. *Annual Reviews in Control* 2010; **34**(1):1–12.
18. Ljung L, Hjalmarsson H, Ohlsson H. Four encounters with system identification. *European Journal of Control* 2011; **17**(5-6):449–471.
19. Julier SJ, Uhlmann JK, Durrant-Whyte HF. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transaction on Automatic Control* 2000; **45**(3):477–482.
20. Wan EA, van der Merwe R. The unscented Kalman filter. In *Kalman Filtering and Neural Networks*, Haykin S (ed.). John Wiley & Sons: New York, NY, USA, 2001.
21. Julier SJ, Uhlmann JK. Unscented filtering and nonlinear estimation. *Proceedingds of the IEEE* 2004; **92**(3):401–402.
22. Ljung L. Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems. *IEEE Transaction on Automatic Control* 1979; **24**(1):36–50.
23. Morall PE, Grizzle JW. Observer design for nonlinear systems with discrete-time measurements. *IEEE Transaction on Automatic Control* 1995; **40**(3):395–404.
24. Song Y, Grizzle JW. The extended Kalman filter as a local asymptotic observer for nonlinear discrete-time systems. *Journal of Mathematical Systems, Estimation, and Control* 1995; **5**(1):59–78.
25. Boutayeb M, Rafaralay H, Darouch M. Convergence analysis of the extended Kalman filter used as an observer for nonlinear deterministic discrete-time systems. *IEEE Transaction on Automatic Control* 1997; **42**(4):581–586.
26. Reif K, Unbehauen R. The extended Kalman filter as an exponential observer for nonlinear systems. *IEEE Transaction on Signal Processing* 1999; **47**(8):2324–2328.

DOI: 10.1002/acs

27. Crisan D, Doucet A. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on signal processing* 2002; **50**(3):736–746.
28. Hu XL, Schön TB, Ljung L. A basic convergence result for particle filtering. *IEEE Transaction on Signal Processing* 2008; **56**(4):1337–1348.
29. Bohlin T. On the problem of ambiguities in maximum likelihood identification. *Automatica* 1971; **7**:199–210.
30. Söderström T. On the uniqueness of maximum likelihood identification. *Automatica* 1975; **11**:193–197.
31. Gourieroux C, Monfort A, Renault E. Indirect inference. *Journal of Applied Econometrics* 1993; **8**:85–118.
32. Gourieroux C, Monfort A. *Simulation-based Econometric Methods*. Oxford University Press: New York, NY, 1996.
33. De Luna X, Genton MG. Robust simulation-based estimation of ARMA models. *Journal of Computational and Graphical Statistics* 2001; **10**:370–387.
34. Åström KJ, Bohlin T. Numerical identification of linear dynamic systems from normal operating records. In *IFAC Symposium on Self-Adaptive Systems*, Teddington, UK, 1965.
35. Ninness B, Wills A, Henriksen SJ, Schon T. Computational system identification. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, 2011.
36. Garatti S, Bittanti S. Estimation of white-box model parameters via artificial data generation: a two stage approach. In *Proceedings of the 17th IFAC world congress*, Seoul, Korea, 2008.
37. Bittanti S, Garatti S. Revisiting the basic issue of parameter estimation in system identification - a new approach for multi-value estimation. In *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 2008.
38. Garatti S, Bittanti S. Parameter estimation in the Pacejka's tyre model through the TS method. In *Proceedings of the 15th IFAC Symposium on System Identification*, Saint-Malo, France, 2009.
39. Tempo R, Calafiore G, Dabbene F. *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer-Verlag: London, UK, 2005.
40. Campi MC. Why is resorting to fate wise? A critical look at randomized algorithms in systems and control. *European Journal of Control* 2010; **5**:419–430.
41. Ljung L. *System Identification Toolbox™ 7 – User Guide*. The Mathworks Inc.: Natick, MA, 2009.

DOI: 10.1002/acs