On the smoothness of nonlinear system identification☆

Antônio H. Ribeiro [a,c,*], Koen Tiels [c,d], Jack Umenberger [c], Thomas B. Schön [c], Luis A. Aguirre [b]

[a] *Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais, Brazil*
[b] *Department of Electronic Engineering, Universidade Federal de Minas Gerais, Brazil*
[c] *Department of Information Technology, Uppsala University, Sweden*
[d] *Department of Mechanical Engineering, Eindhoven University of Technology, The Netherlands*

ARTICLE INFO

ABSTRACT

We shed new light on the *smoothness* of optimization problems arising in prediction error parameter estimation of linear and nonlinear systems. We show that for regions of the parameter space where the model is not contractive, the Lipschitz constant and β-smoothness of the objective function might blow up exponentially with the simulation length, making it hard to numerically find minima within those regions or, even, to escape from them. In addition to providing theoretical understanding of this problem, this paper also proposes the use of multiple shooting as a viable solution. The proposed method minimizes the error between a prediction model and the observed values. Rather than running the prediction model over the entire dataset, multiple shooting splits the data into smaller subsets and runs the prediction model over each subset, making the simulation length a design parameter and making it possible to solve problems that would be infeasible using a standard approach. The equivalence to the original problem is obtained by including constraints in the optimization. The new method is illustrated by estimating the parameters of nonlinear systems with chaotic or unstable behavior, as well as neural networks. We also present a comparative analysis of the proposed method with multi-step-ahead prediction error minimization.

1. Introduction

Prediction error methods (Ljung, 1999) are a widespread class of methods for parameter estimation of dynamic models, which estimate the parameters by minimizing the error between predicted and measured trajectories. Many well-known estimation methods fit into this framework, such as minimizing the one-step-ahead prediction error or the free-run-simulation error. While the classical literature focuses primarily on the estimation of linear systems (Ljung, 1999), the framework is general and enjoys appealing asymptotic properties for the general nonlinear setup (Ljung, 1978).

Minimizing the one-step-ahead prediction usually yields an easier optimization problem, the minimization of the free-run simulation error or other recurrent structures, however, may produce more accurate models. These recurrent models often have a smaller generalization error (Aguirre, Barbosa, & Braga, 2010; Ribeiro & Aguirre, 2018) and better capability when it comes to long-term prediction (Su, McAvoy, & Werbos, 1992). Minimizing recurrent structures is used, for instance, to improve the model structure selection of polynomial models (Piroddi, 2008), fine-tune parameters of nonlinear state–space (Paduart et al., 2010) and block-oriented models (Schoukens & Tiels, 2017).

It is common knowledge among practitioners that the optimization problem resulting from a recurrent model structure is harder to solve (Piroddi & Spinelli, 2003). For linearly parametrized models and convex loss functions, minimization of one-step-ahead prediction error leads to a convex optimization problem; for recurrent model structures, the ensuing optimization is, in general, non-convex, complicating the search for global optima. Even during local optimization, recurrent model structures can lead to cost functions with poor smoothness properties, including many 'jagged' local minima, cf. Fig. 2(a) for an illustration. The current understanding of the relationship between model internal dynamics and the smoothness properties of the cost function is, however, imprecise and provides little insight into ways of circumventing the problem. Furthermore, the few

studies that do investigate the objective function properties in this context are focused on linear systems, see e.g. Eckhard, Bazanella, Rojas, and Hjalmarsson (2017).

The purpose of this paper is twofold. First, we aim to provide insight into the properties of the objective function arising in prediction error estimation problems in a general nonlinear setup. Specifically, we show how the smoothness of the objective function depends on two factors: the simulation length and the decay rate of the recurrent part of the prediction model. Second, we illustrate how this theoretical insight might be leveraged for the design and analysis of practical system identification methods.

The use of *multiple shooting* is analyzed in the context of prediction error minimization. This technique reformulates the optimization problem that arises from minimizing the difference between the output of a prediction model and the observed values. Rather than running the prediction model over the entire dataset, the multiple shooting formulation splits the dataset into smaller subsets and runs the prediction model over each subset. The equivalence with the original problem is obtained by including equality constraints in the optimization problem. This method results in a smoother objective function since it works with shorter simulations and prevents trajectories from diverging too much.

The multiple shooting formulation has reportedly provided improvements in the parameter estimation of ordinary differential equations (Baake, Baake, Bock, & Briggs, 1992; Bock, 1983; Sarode, Kumar, & Kulkarni, 2015) and in the solution of optimal control (Bock & Plitt, 1984; Carraro, Geiger, & Rannacher, 2014; Geisert & Mansard, 2016). In the context of system identification, multiple shooting has been used for estimating polynomial nonlinear space–state models (Van Mulders, Schoukens, Volckaert, & Diehl, 2010) and output error models (Ribeiro & Aguirre, 2017) in settings where conventional methods fail to provide good solutions. Here, we extend this method to the entire class of prediction error methods. In addition, theoretical arguments are put forward to help understand why and when the proposed method is useful. We also present a comparative analysis with multi-step-ahead prediction error minimization (Farina & Piroddi, 2011, 2012; Terzi, Fagiano, Farina, & Scattolini, 2018), showing strengths and weaknesses of each method at a conceptual level and, also, with numerical examples.

2. Prediction error methods

While prediction error methods are widely known, they are often introduced from a linear perspective (Ljung, 1999). In the present section, this is accomplished in a nonlinear setting.

Consider the dataset $\mathcal{Z}^N = \{(\mathbf{u}[k], \mathbf{y}[k]), k = 1, 2, \ldots, N\}$ containing N measured inputs and outputs of a dynamical system. Prediction error methods assume an internal model that delivers output predictions $\hat{\mathbf{y}}[k], k = 1, 2, \ldots, N$. A cost function is defined as the distance between predictions and measured values:

$$V = \frac{1}{N} \sum_{k=1}^{N} \|\mathbf{y}[k] - \hat{\mathbf{y}}[k]\|^2. \tag{1}$$

The prediction model is, usually, parametrized by a parameter vector θ and, although such a dependence is not made explicit in the notation, $\hat{\mathbf{y}}[k]$ depends upon θ. An estimate $\hat{\theta}$ of the parameter vector may be obtained by minimizing V.

Next we assume a dynamical, stochastic, discrete-time system as the data generating process. Let:

$$\underline{\mathbf{u}}[k] = (\mathbf{u}[k], \ldots, \mathbf{u}[k - n_u]),$$
$$\underline{\mathbf{y}}[k-1] = (\mathbf{y}[k-1], \ldots, \mathbf{y}[k - n_y]),$$

where n_y, n_u are the maximum input and output lags. The examples we give next will differ in how they account for the noise

in the model. But, in a noiseless situation, the data generating process for all of them corresponds to a difference equation $\mathbf{y}[k] = \mathbf{f}^*(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k])$. The parametrized function $\mathbf{f}_\theta$ is defined and, if the noise assumption and model structure are correct, the minimization of the cost function V would yield the estimate $\hat{\theta}$ such that, as $N \to \infty$, then $\mathbf{f}_{\hat{\theta}} \to \mathbf{f}^*$ (or to a function with different structure but the same performance in predicting the training dataset). See Appendix D for a more precise description of the asymptotic properties of nonlinear prediction error methods.

Choosing the "true" model structure (one for which there exists a θ^* such that $\mathbf{f}_{\theta^*} = \mathbf{f}^*$) might be impossible in a practical application. Nevertheless the assumption is not so restrictive as it might appear, since there exist families of universal approximator functions (e.g. neural networks and polynomials) for which the distance $\|\mathbf{f}_\theta - \mathbf{f}^*\|$ might be made arbitrarily small within a compact set.

2.1. Nonlinear ARX models

The nonlinear ARX (*autoregressive with exogenous input*) model encodes an output that is corrupted by white *process* noise. That is, it assumes the data were generated by the stochastic discrete-time system:

$$\mathbf{y}[k] = \mathbf{f}^*(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k]) + \mathbf{v}[k], \tag{2}$$

where $\mathbf{v}[k]$ is a zero-mean white noise. This assumption yields (cf. Appendix D) the prediction model:

$$\hat{\mathbf{y}}[k] = \mathbf{f}_\theta(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k]). \tag{3}$$

The minimization of the cost function in Eq. (1) for this prediction model yields an estimator with the desired asymptotic properties.

2.2. Nonlinear OE models

The OE (*output error*) model encodes an output that is corrupted by white *measurement* noise. That is, it is assumed that the data were generated by:

$$\bar{\mathbf{y}}[k] = \mathbf{f}^*(\bar{\mathbf{y}}[k-1], \ldots, \bar{\mathbf{y}}[k - n_y], \underline{\mathbf{u}}[k]),$$
$$\mathbf{y}[k] = \bar{\mathbf{y}}[k] + \mathbf{v}[k],$$

where, again, $\mathbf{v}[k]$ is zero-mean white noise and $\bar{\mathbf{y}}[k]$ represents the noiseless output. This assumption yields the following prediction model:

$$\tilde{\mathbf{y}}[k] = \mathbf{f}_\theta(\tilde{\mathbf{y}}[k-1], \ldots, \tilde{\mathbf{y}}[k - n_y], \underline{\mathbf{u}}[k]),$$
$$\hat{\mathbf{y}}[k] = \tilde{\mathbf{y}}[k]. \tag{4}$$

Here $\tilde{\mathbf{y}}[k]$ represents an estimate of the noiseless output $\bar{\mathbf{y}}[k]$, which should approach the true value as $\theta \to \theta^*$.

2.3. Nonlinear ARMAX models

The nonlinear ARMAX (*autoregressive moving average with exogenous input*) model encodes an output that is corrupted by additive zero-mean *process* noise. In this case, the propagation equation allows the noise term $\mathbf{v}[k]$ to be propagated by the dynamics:

$$\mathbf{y}[k] = \mathbf{f}^*(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k], \mathbf{v}[k-1], \ldots, \mathbf{v}[k - n_v]) + \mathbf{v}[k].$$

This assumption allows the model to account for some forms of colored process noise and results in the following prediction model:

$$\tilde{\mathbf{v}}[k] = \mathbf{y}[k] - \mathbf{f}_\theta(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k], \tilde{\mathbf{v}}[k-1], \ldots, \tilde{\mathbf{v}}[k - n_v])$$
$$\hat{\mathbf{y}}[k] = \mathbf{f}_\theta(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k], \tilde{\mathbf{v}}[k-1], \ldots, \tilde{\mathbf{v}}[k - n_v]). \tag{5}$$

Here $\tilde{\mathbf{v}}[k]$ represents an estimate of the noise corrupting the system and will approach the true noise if the estimated parameter vector approaches the true parameter value θ^*.

2.4. General nonlinear state–space framework

From now on, we will focus on a state–space representation that is general enough to encompass the prediction model from the above examples (for an appropriate choice of the functions $\mathbf{h}$ and $\mathbf{g}$). For this representation, the predicted output is given by:

$$\mathbf{x}[k] = \mathbf{h}(\mathbf{x}[k-1], \underline{\mathbf{z}}[k]; \boldsymbol{\theta}), \tag{6a}$$

$$\hat{\mathbf{y}}[k] = \mathbf{g}(\mathbf{x}[k], \underline{\mathbf{z}}[k]; \boldsymbol{\theta}), \tag{6b}$$

where $\mathbf{x}[k]$ denotes the internal state vector at instant k. For the ARX model the transition state would be empty $\mathbf{x} = \varnothing$; for the OE model $\mathbf{x}[k] = (\bar{\mathbf{y}}[k-1], \ldots, \bar{\mathbf{y}}[k-n_v])$; and, for the ARMAX model $\mathbf{x}[k] = (\tilde{\mathbf{v}}[k-1], \ldots, \tilde{\mathbf{v}}[k-n_y])$. Here $\underline{\mathbf{z}}[k] = (\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k])$. Using both inputs $\underline{\mathbf{u}}[k]$ *and* autoregressive terms $\mathbf{y}[k-1]$ is what allows this state–space representation to encompass ARX, ARMAX and OE models, and also other representations such as the polynomial greybox models proposed in Noël and Schoukens (2018).

2.5. Initial conditions

In order to guarantee the desirable asymptotic properties, the prediction model needs to be simulated starting with the appropriate initial conditions $\mathbf{x}_0$. Since the true initial condition $\mathbf{x}_0^*$ is unknown, there are two possible approaches when estimating the parameters.

The first approach is to fix $\mathbf{x}_0$, for some $\mathbf{x}_0 \approx \mathbf{x}_0^*$, and minimize the cost function (1). This approach is based on the idea that, for an asymptotically stable system, the influence of the initial conditions on the output will decrease over time for many cases of interest (Boyd & Chua, 1985) and, hence, even if $\mathbf{x}_0 \neq \mathbf{x}_0^*$ we can still obtain a good estimate of the parameters. In this case the first samples may be discarded, to make sure the transient errors are not too large. For the ARMAX model, an appropriate choice of initial values would be $\tilde{\mathbf{v}}[k] = \mathbf{0}$, $k = 1, \ldots, n_v$ and, for the OE model, $\hat{\mathbf{y}}[k] = \mathbf{y}[k]$, $k = 1, \ldots, n_y$.

The second approach consists in including $\mathbf{x}_0$ in the optimization problem, so it converges to $\mathbf{x}_0^*$ and improves the quality of the parameter estimates. The optimization problem to be solved in this case is to minimize V with both $\boldsymbol{\theta}$ and $\mathbf{x}_0$ as optimization variables:

$$\min_{\boldsymbol{\theta}, \mathbf{x}_0} V. \tag{7}$$

3. Smoothness of prediction error methods

The theorem below relates the Lipschitz constant of V, and its gradient (i.e. β-smoothness), to the simulation length N. The Lipschitz constant of the cost function and the β-smoothness both play a crucial role in optimization (Nesterov, 1998). Lower values imply that local (Taylor) expansions of the cost function are more predictive of the cost function, and that the optimization algorithm can still converge while taking larger steps. It also gives an upper bound on how distinct in performance two close local minima may be.

The first part of the theorem below can be seen as a formalization of the exploding gradient problem, often studied in the context of neural networks (Pascanu, Mikolov, & Bengio, 2013). The second part provides information about the explosion of second-order derivatives and curvature and it is, to the best of our knowledge, novel. As a result of the analysis, it is found that not only *walls* (resulting from large first-order derivatives) might be formed in non-contractive regions of the parameter space, but also regions with exploding curvature with multiple close local minima (cf. Fig. 2(a)). A recurrent neural network-oriented

perspective of these results is discussed in a concurrent work from our group (Ribeiro, Tiels, Aguirre, & Schön, 2020).

Theorem 1. *Let $\mathbf{h}(\mathbf{x}, \underline{\mathbf{z}}; \boldsymbol{\theta})$ and $\mathbf{g}(\mathbf{x}, \underline{\mathbf{z}}; \boldsymbol{\theta})$ in (6) be Lipschitz in $(\mathbf{x}, \boldsymbol{\theta})$ with constants L_h and L_g on a compact and convex set $\Omega = (\Omega_{\mathbf{x}}, \Omega_{\underline{\mathbf{z}}}, \Omega_{\boldsymbol{\theta}})$. With $\{\underline{\mathbf{z}}[k]\}_{k=1}^N \subseteq \Omega_{\underline{\mathbf{z}}}$ and $(\Omega_{\mathbf{x}}, \Omega_{\boldsymbol{\theta}}) \subseteq \mathbb{R}^{N_x} \times \mathbb{R}^{N_\theta}$. If there exists at least one choice of $(\mathbf{x}_0, \boldsymbol{\theta})$ for which there is an invariant set contained in Ω, then, for trajectories and parameters confined within Ω:*

(1) The cost function V defined in (1) is Lipschitz with constant[1]:

$$L_V = \begin{cases} \mathcal{O}(L_h^{2N}) & \text{if } L_h > 1, \\ \mathcal{O}(N) & \text{if } L_h = 1, \\ \mathcal{O}(1) & \text{if } L_h < 1. \end{cases} \tag{8}$$

(2) If the Jacobian matrices of $\mathbf{h}$ and $\mathbf{g}$ are also Lipschitz with respect to $(\mathbf{x}, \boldsymbol{\theta})$ on Ω, then the gradient of the cost function ∇V is also Lipschitz with constant:

$$L_V' = \begin{cases} \mathcal{O}(L_h^{3N}) & \text{if } L_h > 1, \\ \mathcal{O}\left(N^3\right) & \text{if } L_h = 1, \\ \mathcal{O}\left(1\right) & \text{if } L_h < 1. \end{cases} \tag{9}$$

Proof. See Appendix B.

For contractive models,[2] under certain regularity conditions, we have $L_h < 1$ and, according to the above theorem, both the Lipschitz constant and the β-smoothness of the cost function can be bounded by a constant that, asymptotically, does not depend on the simulation length. All contractive systems have a unique fixed point inside the contractive region, and all trajectories converge to such a fixed point (Rudin, 1964, Theorem 9.23). Systems with richer nonlinear dynamic behaviors, such as limit cycles and chaotic attractors, and also unstable systems, are *non-contractive* and will always have $L_h \geq 1$. The Lipschitz constants and β-smoothness for these systems may, according to Theorem 1, blow up exponentially (or polynomially for some limit cases) with the maximum simulation length.

Less formally, for models that have infinitely long dependencies (i.e. are non-contractive) the distance between trajectories of models that are close in the parameter space might become progressively larger along the simulation length because errors will accumulate. This might yield very intricate objective functions in some parts of the parameter space and make the optimization problem very dependent on the initial estimate.

4. Multiple shooting

The theoretical results from the previous section suggest that long simulation lengths might yield regions of the parameter space where the cost function is intricate, hence hard for the optimization algorithm to navigate.

In this section, we propose the application, in the context of prediction error methods, of a technique called *multiple shooting* for which the maximum simulation length is a design parameter. This enables solving problems that would be impossible or very hard to solve in the setting of Section 2, which will be named *single shooting* from now on.

---

[1] Where $\mathcal{O}$ denotes the big O notation. It should be read as: $L(N) = \mathcal{O}(g(N))$ if and only if there exist positive integers M and N_0 such that $|L(N)| \leq Mg(N)$ for all $N \geq N_0$.

[2] We say that a dynamical system $\mathbf{x}[k+1] = \mathbf{h}(\mathbf{x}[k])$ is *contractive* if, for all $\mathbf{x}$ and $\mathbf{w}$, it satisfies $\|\mathbf{h}(\mathbf{x}) - \mathbf{h}(\mathbf{w})\| < L\|\mathbf{x} - \mathbf{w}\|$, for $L < 1$.
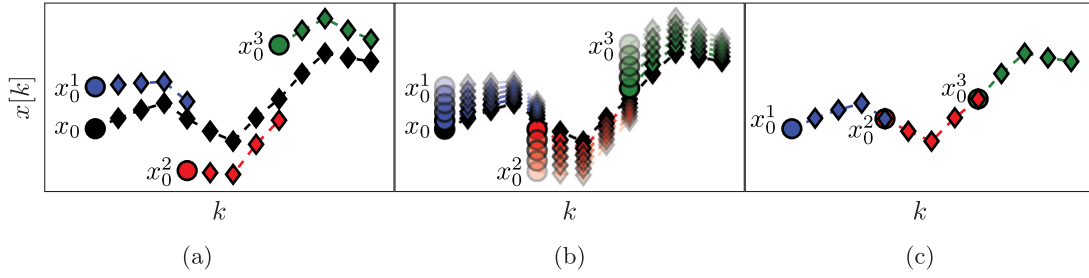
Fig. 1. Multiple shooting illustration. In **black**, we present the simulation of the dynamic system through the entire window length using the single initial condition x_0 (represented by ●). The simulated values are represented by ◆. Dividing the window length into three sub-intervals and simulating the system in each of these, for initial conditions x_0^1, ●, x_0^2, ●, and x_0^3, ●, results in the three different simulations represented by ◆, ◆ and ◆, respectively. In (a), the end of one simulation does not coincide with the beginning of the next one ($x^{i-1}[m_i] \neq x_0^i$). In (b), we show what happens as $\|x^{i-1}[m_i] - x_0^i\| \to 0$. And, in (c), we show that, when $x^{i-1}[m_i] = x_0^i$, the concatenation of these short simulations is equivalent to a single one carried out over the entire window length.

4.1. Method formulation

For the *multiple shooting* formulation, rather than simulating the prediction model (6) through the entire dataset from a single initial condition vector $\mathbf{x}_0$, the data is split into M intervals $\{[m_i + 1, m_{i+1}] \mid i = 1, \ldots, M\}$, $0 = m_1 < m_2 < \cdots < m_M < m_{M+1} = N$, each one with its own set of initial conditions $\mathbf{x}_0^i \in \mathbb{R}^{N_x}$. The ith vector of initial conditions $\mathbf{x}_0^i$ is used to compute $\hat{\mathbf{y}}^i[k]$ over $m_i + 1 \leq k \leq m_{i+1}$:

$$\mathbf{x}^i[k] = \mathbf{h}(\mathbf{x}^i[k-1], \underline{\mathbf{z}}[k]; \boldsymbol{\theta}), \text{ for } \mathbf{x}^i[m_i] = \mathbf{x}_0^i, \tag{10a}$$

$$\hat{\mathbf{y}}^i[k] = \mathbf{g}(\mathbf{x}^i[k], \underline{\mathbf{z}}[k]; \boldsymbol{\theta}). \tag{10b}$$

Since the length of the simulation is limited to the shorter interval $[m_i + 1, m_{i+1}]$, the trajectory is less likely to strongly diverge and this typically helps the optimization procedure by making the objective function *smoother*.

Let $\Delta m_i = m_{i+1} - m_i$, we define:

$$V_i = \frac{1}{\Delta m_i} \sum_{k=m_i+1}^{m_{i+1}} \|\mathbf{y}[k] - \hat{\mathbf{y}}^i[k]\|^2, \tag{11}$$

to be the cost function associated with the ith interval, where the prediction $\hat{\mathbf{y}}^i[k]$ depends upon $\boldsymbol{\theta}$ and $\mathbf{x}_0^i$, according to (10). The *multiple shooting* formulation makes use of the following objective function:

$$V^M = \sum_{i=1}^{M} \frac{\Delta m_i}{N} V_i. \tag{12}$$

This objective function includes states $\mathbf{x}_0^1, \ldots, \mathbf{x}_0^M$ as free variables in the optimization. Hence, rather than reinforcing the cohesion of the states $\mathbf{x}[k]$ by defining them through a recurrence relation that casts a dependency of $\mathbf{x}[k]$ all the way back to the initial condition $\mathbf{x}_0$, as in the *single shooting* formulation, the cohesion between subsequent states is achieved through optimization constraints, resulting in the following problem:

$$\min_{\boldsymbol{\theta}, \mathbf{x}_0^1, \ldots, \mathbf{x}_0^M} V^M, \tag{13}$$

subject to: $\mathbf{x}^{i-1}[m_i] = \mathbf{x}_0^i$,

for $i = 2, 3, \ldots, M$.

The next theorem provides the equivalence between (7) and (12). The theorem and its corollary are a formalization of the intuition provided in Fig. 1 and they give further insight into how the constraints in the *multiple shooting* formulation are used to imitate a single simulation throughout the entire dataset.

Theorem 2. *If* $\mathbf{x}^{i-1}[m_i] = \mathbf{x}_0^i$, *for* $i = 2, 3, \ldots, M$ *and* $\mathbf{x}_0^1 = \mathbf{x}_0$, *then* $V = V^M$.

Proof. Let us call $\mathbf{x}[k]$, $\hat{\mathbf{y}}[k]$ and $\mathbf{x}^i[k]$, $\hat{\mathbf{y}}^i[k]$ the states and predictions in, respectively, the single shooting simulation and in the ith multiple shooting interval. For a fixed i, if $\mathbf{x}[m_i] = \mathbf{x}_0^i$ then $\mathbf{x}[k] = \mathbf{x}^i[k]$ for all $k \in [m_i + 1, m_{i+1}]$. Hence, inside this same interval, $\hat{\mathbf{y}}[k] = \hat{\mathbf{y}}^i[k]$. Applying this for every i it follows from the respective definitions that: $V = \sum_{i=1}^{M} \frac{\Delta m_i}{N} V_i = V^M$.

Corollary 3. *The pair* $(\boldsymbol{\theta}^*, \mathbf{x}_0^*)$ *is a global solution of* (7) *if and only if there exist* $(\mathbf{x}_0^2, \ldots, \mathbf{x}_0^M)$ *such that* $(\boldsymbol{\theta}^*, \mathbf{x}_0^*, \mathbf{x}_0^2, \ldots, \mathbf{x}_0^M)$ *is a global solution of the optimization problem* (13).

Multiple shooting can be understood as a generalization of the single shooting case. That is because, if $M = 1$ and $\Delta m_1 = N$, both methods result in the same optimization problem. Multiple shooting, however, might provide some advantages. The method is more amenable to parallelization, since each cost function V_i and its associated derivatives can be computed independently and, possibly, in parallel. Also, long simulations usually yield larger numerical errors due to finite precision errors that accumulate along the simulation, hence multiple shooting shorter simulation intervals also attenuate this problem. Finally, multiple shooting cost function has better smoothness properties that will be investigated next.

4.2. Properties of the cost function

For the multiple shooting method, the Lipschitz constant of V^M and of its gradient, L_{V^M} and L'_{V^M} depend asymptotically on $\Delta m_{\max} = \max_{1 \leq i \leq M} \Delta m_i$ rather than on N (See Appendix C). For instance, if $L_h > 1$:

$$L_{V^M} = \mathcal{O}(L_h^{2\Delta m_{\max}}); \quad L'_{V^M} = \mathcal{O}(L_h^{3\Delta m_{\max}}). \tag{14}$$

Since $\Delta m_{\max}$ is a design parameter, it is possible to have some control over the Lipschitzness and β-smoothness of the objective function in the non-contractive region of the parameter space ($L_h \geq 1$).

5. Implementation and numerical examples

In this section, numerical examples are presented. The cost function smoothness is investigated through the lens of the theoretical results in Section 3 and it is shown how *multiple-shooting* might help mitigate some problems.

The equality-constrained problem (12), which arises from the multiple shooting formulation, is solved using an implementation of the sequential quadratic programming solver originally described in Lalee, Nocedal, and Plantenga (1998) available in the SciPy library (Virtanen et al., 2020).[3] The procedure used for computing the derivatives is explained in Appendix A. Additional numerical examples are provided in Appendix F.

---

[3] `scipy.optimize.minimize(method='trust-constr')`.

5.1. OE model for a chaotic system

This example illustrates how multiple shooting makes prediction error methods more robust w.r.t. the choice of initial conditions for the optimization. A dataset with $N = 200$ samples is generated using the logistic map (May, 1976):

$$y[k] = \theta y[k-1](1 - y[k-1]), \tag{15}$$

with $\theta = 3.78$. From the generated dataset we try to estimate an *output error* model with the same structure.

Fig. 2(a) illustrates the objective function for the *single shooting* case. The data generating system, Eq. (15), presents chaotic behavior for $\theta \in [3.57, 4]$, which explains the very intricate objective function in this region. For chaotic systems, small variations in the parameters may cause large variations in the system trajectory and, consequently, abrupt changes in the *free-run simulation error*. This explains why the cost function used in estimating an OE model has many local minima for this problem.

The solutions found by the solver, for different initial guesses, are also displayed in Fig. 2(a). Notice that the solver fails to find the true solution because it always gets trapped in local stationary points near the initial guess. Even in this noiseless scenario, the estimation problem is very challenging due to the chaotic nature of the system. Hence, for a simulation that is sufficiently long, the trajectories will differ significantly even for small parameter variations.

Multiple shooting makes the problem easier by limiting the maximum simulation length. Fig. 2(b), (c) and (d) display the objective function and the solutions found by the solver starting from different initial guesses. The estimation procedure becomes easier as Δm_{max} is made smaller. For Fig. 2(c) the solver already converges to the true parameter for some initial guesses, but not for all of them. For Fig. 2(d), the solver converges to the true solution for all initial guesses that have been tested.

For the multiple shooting case, besides θ, the initial conditions are also optimization parameters. To help with the visualization of this multidimensional problem, Fig. 2(b), (c) and (d) display the main curve corresponding to the objective function for the true initial conditions and faded lines corresponding to the objective function for perturbed initial conditions. Another consequence of the problem having more parameters than displayed in the figure is that the cost function found by the solver does not need to lie on any of the objective function curves displayed in the figure, since it may have a different set of initial conditions $\mathbf{x}_0^i$.

It is important to highlight that the mechanism used here is not to take the system outside of the chaotic regime, but rather avoid simulating the system for too long. By doing that, we avoid the major problem that arises in the identification of chaotic systems, i.e. the high sensitivity to parameters and initial conditions. This results in a better behaved objective function (cf. Fig. 2). The constraints allow the equivalence with the original prediction error problem (according to Theorem 2 and Corollary 3).

Table 1 gives the number of function evaluations and the running time for the four situations displayed in Fig. 2. The convergence happens within just a few iterations for $\Delta m_{max} = N$ (single shooting) because any initial point is probably very close to some optimal *local* solution. As we reduce Δm_{max} the objective function becomes less intricate and this is reflected in the convergence of the solver. For $\Delta m_{max} = 10$ the solver takes much longer to converge. We believe this happens because the local solution is not so close in the parameter space to the initial guess anymore. As Δm_{max} is further decreased, however, the convergence becomes faster because it is dealing with, what is believed to be, a smoother problem that can be handled more accurately by low order approximations.

Table 1
Computational cost for the logistic map estimation. The number of function evaluations and total running time until convergence. Minimum, maximum and median are given over 15 runs for the situations presented in Fig. 2.

Δm_{max}	Function evaluations			Run time (s)		
	min	median	max	min	median	max
N	1	15	23	0.01	0.2	1.1
10	43	1000[a]	1000[a]	0.7	24.7	27.3
5	29	115	645	1.0	2.9	26.9
2	21	50	65	1.7	2.9	3.8

[a] The number of iterations is limited to 1000 and the solver is interrupted when this number is reached.

5.2. Pendulum and inverted pendulum

Consider the following discrete-time nonlinear system:

$$\begin{cases} x_1[k+1] = x_1[k] + \delta x_2[k] \\ x_2[k+1] = -\delta \frac{g}{l} \sin x_1[k] + (1 - \delta \frac{k_a}{m})x_2[k] + \delta \frac{1}{m}u[k] \end{cases}$$
$$y[k] = x_1[k] \tag{16}$$

which corresponds to a pendulum model, discretized using the Euler approximation $\dot{x}(t) \approx \frac{x((k+1)\delta) - x(k\delta)}{\delta}$, where g is the gravity acceleration, m is the mass connected to the extremity of the pendulum, l is the length of the (massless) rod connecting the mass to the pivot point, and k_a is the linear friction constant. It has two states: the angle of the mass (x_1) and the angular velocity (x_2). The input $u[k]$ is the force applied to the mass.

This system has multiple equilibrium points, namely, $(x_1, x_2) = (\pm\pi i, 0)$ for $i = 0, 1, 2, 3, \ldots$. The equilibrium points at $(x_1, x_2) = (\pm 2\pi i, 0)$ are stable and the equilibrium points at $(x_1, x_2) = (\pi \pm 2\pi i, 0)$ are unstable. For this system, with $g = 9.8$, $l = 0.3$, $m = 3$, $k_a = 2$ and $\delta = 0.01$, we define three different datasets: (a) A dataset for which small inputs are applied to the system, that stays under the influence of the stable point $(x_1, x_2) = (0, 0)$ and $y[k]$ stays, approximately, inside the range $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$; (b) A dataset for which the system is maintained close to the unstable point $(x_1, x_2) = (\pi, 0)$ by a linear controller; and, (c) A dataset for which the input is large enough to drive the pendulum to full rotations around its center. The output corresponding to those three situations are displayed in Fig. 3.

Fixing $m = 3$ and $\delta = 0.01$ parameters $\frac{g}{l}$ and k_a of an output error model with the structure presented in (16) were estimated from the data. A visualization of the cost function is presented in Fig. 4 together with numerical solutions found by means of the single shooting and multiple shooting formulation starting from different initial conditions.

For dataset (a), the single shooting formulation is able to recover the true parameters from data for most of the initial conditions. Some exceptions occur when initialized far away from the correct initial conditions. For datasets (b) and (c), for which the system needs, respectively, to operate close to the unstable dynamics or to account for the existence of multiple fixed points, the cost function is highly intricate and full of local minima. In this case, the optimization algorithm, even when initialized close to the local solution, fails to converge to reasonable solutions. This result is consistent with Theorem 1 and how the smoothness of the objective function degenerates (exponentially) on sets of the parameter space for which the prediction model is non-contractive, such as the trajectories close to the unstable fixed point of the system (16). The use of multiple shooting yields an objective function that looks similar to a paraboloid in the region of interest for the three cases and the optimization procedure converges to the true parameter regardless of the initialization.

For nonlinear ARX, ARMAX or OE models the states are directly measured (possibly with some noise contamination) and the initialization of the optimization parameters $\mathbf{x}_0^i$ follows naturally
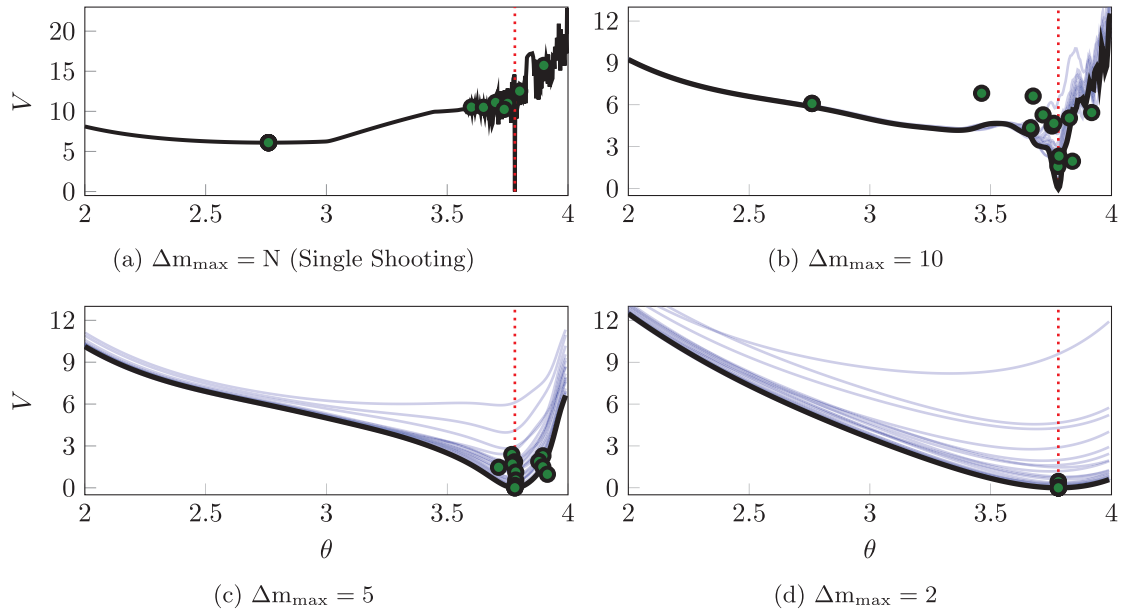
Fig. 2. Logistic map parameter estimation. *Cost function* of the optimization problem for $\mathbf{x}_0^i$ fixed in its true values (in black) and for disturbed versions of the these initial conditions (in blue). We present the result for four values of $\Delta m_{\max}$, and omit disturbed initial condition objective functions for the single shooting case ($\Delta m_{\max} = N$) to make it easier to visualize. The green circles, ●, indicate the pair (θ, V) corresponding to a solution found by the solver. There are 15 circles in each figure (some of them overlapping), the circles correspond to solutions for different initial guesses. As initial guesses we picked values of θ uniformly spaced between 3.2 and 3.9, with $\mathbf{x}_0^i$ picked from randomly disturbed versions of the true initial conditions (which are known because we generated the data ourselves). The true value $\theta = 3.78$ is indicated by the dotted red vertical line.
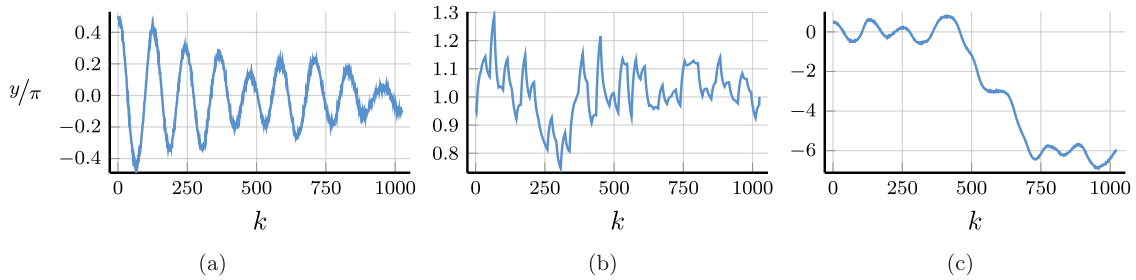

Fig. 3. Angle over time for pendulum parameter estimation. Display the *output signal* $y[k]$ for the three different datasets used for estimating the parameters. The dataset size is $N = 1024$ samples. (a) The input applied in this case is a zero-mean Gaussian random input with standard deviation $\sigma_u = 10$, each random value being held for 20 samples. The input in this case is unable to drive the system away from the influence of the stable fixed point $(0, 0)$. (b) The input $u[k]$ in this case is obtained by the control law: $u[k] = 40\delta e[k-1] - 78.8\delta e[k-2] + 38.808\delta e[k-3] + 1.02u[k-1] - 0.02u[k-2]$, where the error is the difference between the reference and the output: $e[k] = r[k] - y[k]$. The reference is $r[k] = \pi + \Delta r[k]$ where $\Delta r[k]$ is a zero-mean Gaussian random input with standard deviation $\sigma_r = 0.2$, each random value being held for 20 samples; and, (c) same as (a) but with the larger standard deviation $\sigma_u = 50$, which is able to drive the pendulum to complete full rotations. For (a) and (c) zero-mean Gaussian white noise with standard deviation $\sigma_r = 0.03$ was added to the output.

from the considerations in Section 2.5. Here, however, the state variable x_2 is not measured. This variable can be interpreted as the derivative of x_1, so a finite difference approximation was used in the initialization of the intermediary initial conditions $\mathbf{x}_0^i$. Although the high-pass behavior of the derivative amplifies the noise, this choice was still better than a completely arbitrary one.

6. Comparison with multi-step-ahead prediction error minimization

6.1. Multi-step-ahead prediction error minimization

Multiple-shooting is presented here as a possible way of limiting the simulation interval $\Delta m_{\max}$. A method that appears in the system identification literature that also has a similar effect is the multi-step-ahead prediction error minimization (MSA-PEM) (Farina & Piroddi, 2011, 2012; Terzi et al., 2018). The approach fits well into the moving horizon framework (Terzi et al., 2018) and is popular for system identification in model predictive control

application. The method has been studied primarily in a linear model setting, nevertheless it can be extended to a nonlinear setting (Farina & Piroddi, 2012).

In the MSA-PEM estimation, the simulation is truncated to a fixed number K of steps backwards. That is, for each $k = 1, \ldots, N$, we define an auxiliary variable $\tilde{\mathbf{x}}_k[i]$ and an initial condition $\tilde{\mathbf{x}}_{0,k}$. Starting from $\tilde{\mathbf{x}}_k[k-K] = \tilde{\mathbf{x}}_{0,k}$ the system is propagated using the state equation: $\tilde{\mathbf{x}}_k[i] = \mathbf{h}(\tilde{\mathbf{x}}_k[i-1], \underline{\mathbf{z}}[i]; \boldsymbol{\theta})$ to simulate the evolution of this auxiliary state variable for $i = k - K, \ldots, k$. The prediction is then computed using $\hat{\mathbf{y}}[k] = \mathbf{g}(\tilde{\mathbf{x}}_k[k], \underline{\mathbf{z}}[k]; \boldsymbol{\theta})$. The parameters are obtained by minimizing a cost function similar to (1).

Multiple shooting is equivalent, in the sense of Theorem 2 and Corollary 3, to solving the original (single shooting) problem regardless of the choice of simulation interval $\Delta m_{\max}$. MSA-PEM, on the other hand, is equivalent to the original formulation only if $K = N$. The next example illustrates how the statistical properties of the method change as K varies from 1 to N.
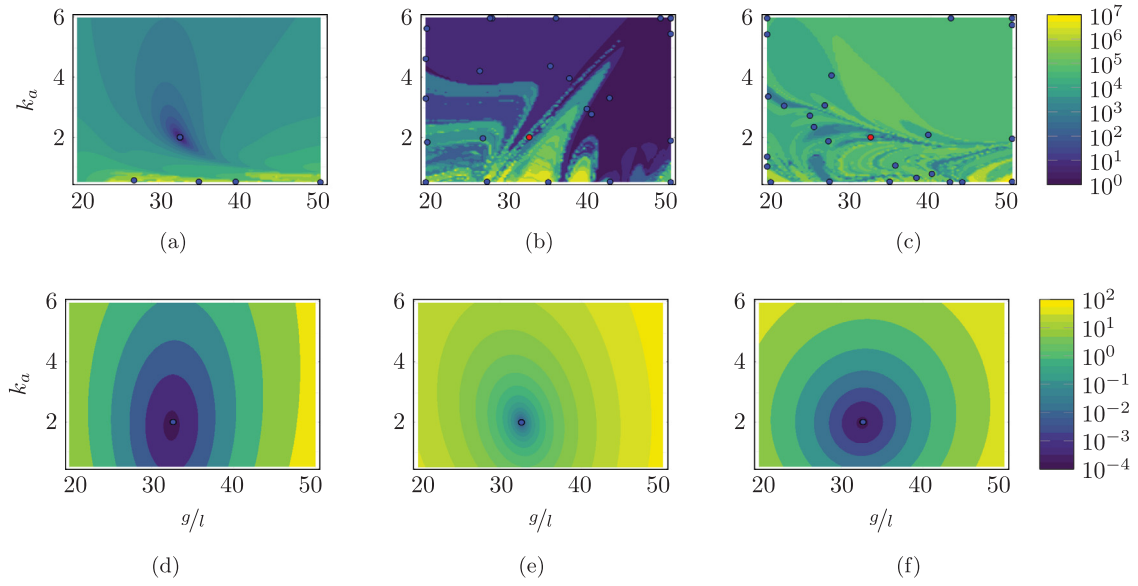
Fig. 4. Contour plot of the cost function. Figures (a), (b) and (c) correspond to the cost function V from single shooting simulation for the datasets (a), (b) and (c) generated as described in Fig. 3 caption; and, in (d), (e) and (f) the cost function for the same problems is displayed for the multiple shooting formulation with $\Delta m_{max} = 16$. The true parameter is indicated by a red circle, ●, and the solutions found by the solver are indicated by blue circles, ●. There are 25 blue circles in each figure (some of them overlapping with each other), each circle corresponds to the solution for a different initial guess. The red circles might not be visible in some cases because they are hidden under the blue circles. Some solutions are outside of the displayed region and the corresponding blue dots are displayed at the edge of the plot. Initial optimization guesses for θ were picked from a grid of points uniformly spaced in the rectangle $[20, 50] \times [0.5, 6]$. It is important to highlight that the plots show a two-dimensional projection of a cost function that is defined on an extended parameter space that includes the initial conditions $\mathbf{x}_0^i$, $i = 1, \dots, M$ as parameters, which were fixed to the true values when generating the contour plots. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.2. Example: estimating the output error model for second-order under-damped system

A dataset with $N = 300$ samples is generated using the equation:

$$\bar{y}[k] = \theta_1 \bar{y}[k] + \theta_2 \bar{y}[k-2] + \theta_3 u[k-1] \tag{17a}$$

$$y[k] = \bar{y}[k] + v[k]. \tag{17b}$$

Here $\bar{y}$ represents the noiseless output, and v represents the white output noise that is introduced during the data generation. At each k, $v[k]$ is a Gaussian random variable with standard deviation $\sigma_v = 0.05$.

Let $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ denote the parameter vector used for the data generation. Three different settings are considered: (a) $\boldsymbol{\theta} = (0.5, -0.2, 2)$; (b) $\boldsymbol{\theta} = (1.5, -0.7, 0.5)$; and, (c) $\boldsymbol{\theta} = (1.8, -0.95, 0.1)$. The three settings correspond to underdamped linear systems with different response-times: in (a) the system responds *fast* to input changes and has a time constant $\tau = 0.25$; in (c), it responds *slowly* and $\tau = 0.9$; and, (b) is an *intermediate* setting with $\tau = 0.75$.

From the synthetically generated data, the parameters of a model with the same structure are estimated. The experiment is repeated 100 times, each time corresponding to a different realization (i.e., different random seed) of the data generation. The result of the Monte Carlo procedure is displayed in Fig. 5 for the parameter estimation using ARX, standard (single shooting) output error, multiple shooting, and MSA-PEM. We show the difference between estimated and true values for the first parameter, $\hat{\theta}_1 - \theta_1$. Similar results have been obtained for $\hat{\theta}_2$ and $\hat{\theta}_3$.

Due to the presence of white output error, using *output error estimation* yields consistent and well-behaved results. *ARX estimation*, on the other hand, is biased, and its histogram is not centered around zero. This is more evident either in 5(b) or in 5(c). In Fig. 5(a), which correspond to the fast response system,

there is not much difference between the two estimation procedures, mostly because any disturbance fade away very quickly, yielding similar statistical properties for the two estimators.

Multiple shooting estimation results are similar to the (single shooting) output error estimation. For some choices of Δm_{max}, the optimization problem might actually be made harder due to the increase in the problem dimension. In this case, the algorithm seems to fail a few times, which can be observed in the histogram as a few outliers in the estimated parameter distribution. Except for these outliers, all the different choices of Δm_{max} yield very similar distributions for the estimated parameters.

For *MSA-PEM*, the estimation results vary qualitatively among the three settings. For the intermediate setting, midway qualitative behavior between ARX and output error estimation is obtained, approaching that of output error estimation as K increases. For the fast-response system, a similar interpretation is possible, however, there is not much difference between using ARX and output error estimation to begin with (cf. Fig. 5(a)), hence all K-step-ahead choices yield similar results. Finally, for the slow-response setting, minimizing the multi-step-ahead prediction error does not offer a direct compromise between ARX and output error estimation and, for $K = 10$ or $K = 20$, it is possible to observe a bimodal distribution with one peak distant from zero. And $K = 7$ yields the best results.

In Fig. 6, the computational cost of multiple shooting and MSA-PEM are compared. The average *running time divided by the number of function evaluations* is displayed in Fig. 6(a) and (b). This accounts for the computational cost of computing the cost function, derivatives, and performing the matrix factorizations needed by the optimization algorithm at each iteration. Fig. 6(c) and (d) display the number of cost function evaluations needed for the optimization algorithm to converge. The next section studies time and memory complexity of the methods and explains the obtained results in a more general context.
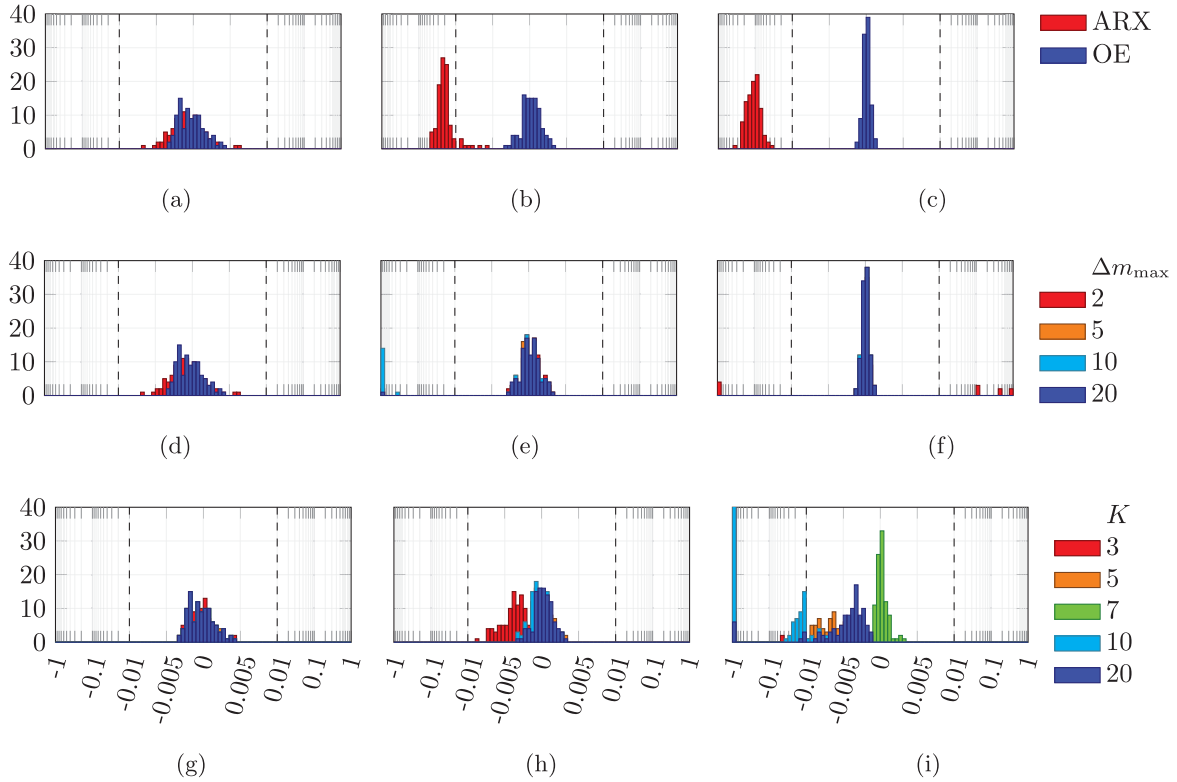
Fig. 5. Monte Carlo simulation of the estimates. *Histogram* showing $(\hat{\theta}_1 - \theta_1)$, for 100 different realizations of the estimation experiment. In (a) to (c), results for ARX and the standard (single shooting) output error estimation. The results are displayed: in (a) for the fast-response system with $\tau = 0.25$; in (b) for the intermediate setting $\tau = 0.75$; and, in (c) the slow-response system with $\tau = 0.9$. In (d) to (f), results for multiple-shooting estimation in the same three settings. The results are displayed for $\Delta m_{max} = \{2, 5, 10, 20\}$, which often overlap. In (g) to (i), results for the multi-step ahead prediction in the three settings. We show the results for $K = \{3, 5, 7, 10, 20\}$. In order to facilitate the visualization two different scales were used for the *x*-axis: between -0.01 and 0.01 the scale is linear; and logarithmic scale is used for values below -0.01 or above 0.01.

6.3. Computational cost

As observed in Fig. 6(b), MSA-PEM has an average running time per iteration that grows linearly with the number of stages. Since the problem dimension and the factorization cost per iteration remain constant, it is the time complexity of computing the cost function V (and its derivatives) that accounts for the linear growth. In Appendix E, we discuss in detail the computation of V and arrive at the time complexity of $\mathcal{O}(NK(n_y + n_u))$.

Multiple shooting estimation, on the other hand, has an average running time per iteration that decreases with the maximum simulation length Δm_{max}. This is displayed in Fig. 6(a). In this case, the cost of computing V is $\mathcal{O}(N(n_y + n_u))$ and does not depend on the simulation length. The increase in the problem dimension, however, yields more costly factorizations (cf. Appendix A), which results in the increased computational cost per iteration.

For multiple shooting, shorter simulation lengths Δm_{max} allow the solver to converge with fewer function evaluations. This is in agreement with the theory presented in this paper, which suggests longer simulation lengths might result in poor smoothness properties and make the optimization problem harder to solve. Applying the same reasoning to MSA-PEM, it is natural to expect that smaller values of K would yield convergence with fewer iterations. This, however, does not seem to be the case for the slow time–response setting, maybe because MSA-PEM often converges to the wrong parameters in this setting (as shown in Fig. 5(i)).

In general, the number of function evaluations for MSA-PEM is lower than for multiple shooting because the resulting problem is unconstrained, rather than constrained, for which more efficient procedures are available.

Table 2
Tradeoffs: multiple shooting *vs.* MSA-PEM. N_θ is the number of parameters. N_x is the number of states that need to be propagated through the simulation ($N_x = n_y = 2$). Δm_{max} is the maximum simulation length used for multiple shooting and K is the number of steps ahead predicted in the MSA-PEM.

	Multiple shoot.	MSA-PEM
Problem dimension	$N_\theta + \frac{N}{\Delta m_{max}} \cdot N_x$	N_θ
# of constraints	$\frac{N}{\Delta m_{max}} \cdot N_x - 1$	$\varnothing$
Time-complexity (computing V)	$\mathcal{O}(N(n_y + n_u))$	$\mathcal{O}(NK(n_y + n_u))$
Asymp. statistical properties	Independent of Δm_{max}	Approach expected properties as $K \to N$
# of independent subproblems	$\frac{N}{\Delta m_{max}}$	N

Finally, MSA-PEM and multiple shooting are both amenable to parallelization, even though this is not explored in the examples. During MSA-PEM implementation, it is possible to instantiate N different processes (or threads) for computing each prediction $\hat{y}[k]$ (and the corresponding derivatives) in parallel. Multiple shooting, on the other hand, subdivides the problem into $(N/\Delta m_{max})$ independent subproblems.

6.4. Cost function smoothness and estimation tradeoffs

Both MSA-PEM and multiple shooting allow the user to control the maximum simulation length, hence the smoothness of the cost function (cf. Section 3), by setting K for the MSA-PEM and Δm_{max} for the multiple shooting. These parameters, however, affect diverse aspects of the estimation problem, which are summarized in Table 2. Multiple shooting has an exact equivalence
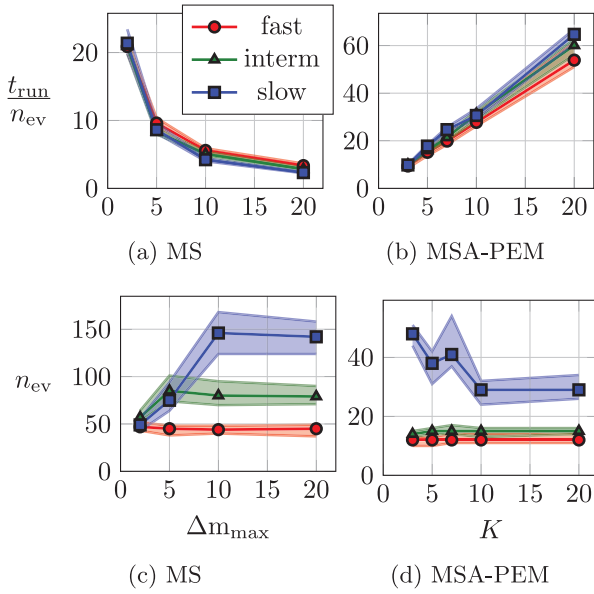
Fig. 6. Execution time comparison. Two components of the time for the optimization algorithm to converge are displayed: plots (a) and (b), display the *total running time* t_{run} divided by the number of function evaluations n_{ev}; and, plots (c) and (d), display the *number of function evaluations* n_{ev}. In (a) and (c), the results are for multiple shooting estimation and the *x*-axis is the maximum simulation length Δm_{max}. In (b) and (d), the results are for MSA-PEM and the *x*-axis is the number of steps ahead K. The results are displayed for: the fast-response system with $\tau = 0.25$ in red; the intermediate setting $\tau = 0.75$ in green; and, the slow-response system with $\tau = 0.9$ in blue. The results are for the 100 different realizations of experiment: the solid line is the median and the shaded region is the interquartile range. That is, the shaded region illustrate the statistical dispersion of the observed values and show the region between the first and third quartiles (i.e. 25th and 75th percentiles). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

with the original single shooting problem regardless of the simulation length and the parameter Δm_{max} offers a tradeoff between the problem dimension and smoothness properties of the cost function. MSA-PEM keeps the problem dimension fixed, and, as K increases, it is possible to approach the behavior of methods with distinct statistical properties (not always in a smooth way, as shown in Fig. 5(i)), at the expense of an increased computational cost and worse smoothness properties.

Some refinements are proposed in the literature. For instance, Farina and Piroddi (2011) propose to average over different values of K, to obtain a slower but smoother convergence to output error estimation as $K \rightarrow N$. In Farina and Piroddi (2012), on the other hand, the method starts from the one-step-ahead prediction solution and increases K by one in each iteration until convergence. We give an numerical example using such approach in Appendix F.

7. Conclusion

The relevance of this paper lies in the rather general setting for which the proposed methods and results hold. The main technical contribution is to show that for dynamic prediction models that are non-contractive (i.e. do not converge asymptotically to a single stable point) in the region of interest, the upper bound for the Lipschitz constant and the β-smoothness blows up exponentially with the simulation length, and this can make the optimization problem very hard to solve. This was illustrated using numerical examples with systems that are non-contractive due to the presence of chaotic regions and unstable equilibrium

points. Because of these regimes, the objective function becomes very intricate in some regions of the parameter space and the optimization algorithm fails to find a good solution. Even for problems that are contractive in the region of interest, multiple shooting might help in preventing the solver from getting stuck in undesirable regions of the parameter space, hence facilitating the convergence to a good solution (cf. Section 5.2).

Multiple shooting makes the simulation length a design parameter and hence allows solving optimization problems that would be infeasible in a single shooting setting. The price paid compared to single shooting methods is that a nonlinear constrained optimization problem must be solved instead of an unconstrained one. It also makes it harder to generalize to situations other than batch training, such as online training. MSA-PEM is another approach that allows some control over the smoothness of the cost function, but with different tradeoffs (see Table 2) between smoothness, asymptotical properties and computational cost to be taken into consideration.

Code avaibility

The code for reproducing the examples is available in: github.com/antonior92/MultipleShootingPEM.jl.

Appendix A. Computing the derivatives

A.1. Sensitivity equations

Let the Jacobian matrices of $\mathbf{h}(\mathbf{x}, \underline{\mathbf{z}}; \boldsymbol{\theta})$ with respect to $\mathbf{x}$ and to $\boldsymbol{\theta}$ evaluated at the point $(\mathbf{x}[k], \underline{\mathbf{z}}[k]; \boldsymbol{\theta})$ be denoted, respectively, as A_k and B_k. Similarly, the Jacobian matrices of $\mathbf{g}(\mathbf{x}, \underline{\mathbf{z}}; \boldsymbol{\theta})$ are denoted as C_k and F_k. Also, we denote the Jacobian matrices of $\hat{\mathbf{y}}[k]$ with respect to $\boldsymbol{\theta}$ and to $\mathbf{x}_0$ as $J_{\boldsymbol{\theta}}[k]$ and $J_{\mathbf{x}_0}[k]$. And the Jacobian matrices of $\mathbf{x}[k]$ are denoted as $D_{\boldsymbol{\theta}}[k]$ and $D_{\mathbf{x}_0}[k]$.

A direct application of the chain rule to (6) gives a recursive formula for computing the derivatives of the predicted output in relation to the parameters in the interval $1 \leq k \leq N$:

$$D_{\boldsymbol{\theta}}[k] = A_k D_{\boldsymbol{\theta}}[k-1] + B_k \text{ for } D_{\boldsymbol{\theta}}[0] = \mathbf{0}$$
$$J_{\boldsymbol{\theta}}[k] = C_k D_{\boldsymbol{\theta}}[k] + F_k. \tag{18}$$

A similar recursive formula may be used for computing the derivatives of the predicted output in relation to the initial conditions:

$$D_{\mathbf{x}_0}[k] = A_k D_{\mathbf{x}_0}[k-1] \text{ for } D_{\mathbf{x}_0}[0] = \mathbf{I} \tag{19}$$
$$J_{\mathbf{x}_0}[k] = C_k D_{\mathbf{x}_0}[k].$$

Finally, we define $D[k] = [D_{\boldsymbol{\theta}}[k], D_{\mathbf{x}_0}[k]]$ and $J[k] = [J_{\boldsymbol{\theta}}[k], J_{\mathbf{x}_0}[k]]$.

A.2. Single shooting

For the cost function V defined as in (1), its gradient ∇V is given by:

$$\nabla V = \frac{2}{N} \sum_{k=1}^{N} J[k]^T (\hat{\mathbf{y}}[k] - \mathbf{y}[k]), \qquad (20)$$

Its Hessian $\nabla^2 V$ is given by:

$$\nabla^2 V = \frac{2}{N} \sum_{k=1}^{N} \left(J[k]^T J[k] + \mathbf{S}[k] \right). \qquad (21)$$

where $\mathbf{S}[k] = \sum_{j=1}^{N_y} \hat{y}_j[k] \nabla^2 \hat{y}_j[k]$. Ignoring $\mathbf{S}[k]$ is a common approximation used in least-squares algorithms, that will also be used here when computing derivatives numerically.

A.3. Multiple shooting

In order to solve the problem using the sequential quadratic programming solver (Lalee et al., 1998) we must be able to compute: (i) The cost function V^M; (ii) its gradient ∇V^M; (iii) the constraints; (iv) the Jacobian matrix of the constraints (which can be represented using a sparse representation); and, (v) for any given vector $\mathbf{p}$, the product of the Lagrangian[4] Hessian and the vector $\mathbf{p}$ (the full Lagrangian Hessian matrix does not need to be computed). The following sequence provides a way of computing all derivatives required by the optimizer.

Algorithm 1 (*Derivatives*)**.** For a given parameter θ and set of initial conditions:

(1) For $i = 1, \dots, M$, do:

    (a) For $k = \mathrm{m}_i + 1, \dots, \mathrm{m}_{i+1}$:

        (i) Compute $\mathbf{x}^i[k]$ and $\hat{\mathbf{y}}^i[k]$ with (6).
        (ii) Compute A_k, B_k, C_k and F_k.
        (iii) Compute $D^i[k]$ and $J^i[k]$ with the sensitivity equations.

    (b) Compute V_i using Eq. (11).
    (c) Compute ∇V_i using a formula equivalent to (20).
    (d) Approximate the product of the Hessian with a given vector, $\nabla^2 V_i \mathbf{p}$, using the first terms from a expression equivalent to (21)

(2) Compute V^M with (12);
(3) Compute $\nabla V^M = \sum_{i=1}^{M} \frac{\Delta \mathrm{m}_i}{N} \nabla V_i$;
(4) Compute the value of the constraint from the values of $\mathbf{x}^i[\mathrm{m}_{i+1}]$, $i = 1, \dots, M$;
(5) Compute the Jacobian matrix of the constraints from $J^i[\mathrm{m}_{i+1}]$, $i = 1, \dots, M$;
(6) Compute $\nabla^2 V^M \mathbf{p} = \sum_{i=1}^{M} \frac{\Delta \mathrm{m}_i}{N} \nabla^2 V_i \mathbf{p}$;
(7) Compute the product of the Hessian $\boldsymbol{\lambda}^T \mathbf{c}(\boldsymbol{\phi})$ with a vector $\mathbf{p}$ using 2-point finite differences;
(8) Compute the product of the Lagrangian Hessian and a vector $\nabla^2 \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\lambda}) \mathbf{p}$, summing the Hessians computed in steps 6 and 7.

Some approximations were used for computing the second derivatives: (1) for computing the Hessian of the objective function, the standard least-squares approximation for the Hessian is used; and, (2) for computing the Hessian of the constraint we use finite-difference approximation. The use of finite differences

---

[4] The Lagrangian is given by: $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\lambda}) = V(\boldsymbol{\phi}) + \boldsymbol{\lambda}^T \mathbf{c}(\boldsymbol{\phi})$.

here comes inexpensively because we only need to evaluate the Hessian times a vector, and not the full matrix. Hence, it can be done at the cost of an extra Jacobian matrix evaluation.

Notice that step (1) from the above algorithm can be parallelized, with different processes (or threads) performing the computation for different values of i.

Appendix B. Proofs

B.1. Preliminary results

Lemma 4. *Let $\mathbf{f}$ and $\mathbf{g}$ be two Lipschitz functions on Ω with constants L_f and L_g. Then,*

    (a) $\mathbf{f} + \mathbf{g}$ is also a Lipschitz function on Ω with the (best) Lipschitz constant upper bounded by $(L_f + L_g)$;
    (b) if, additionally, $\mathbf{f}$ and $\mathbf{g}$ are bounded by M_f and M_g on Ω, then $\mathbf{fg}$ is also a Lipschitz function on Ω with the (best) Lipschitz constant upper bounded by $(L_f M_g + L_g M_f)$.

B.2. Proof of Theorem 1 (a)

Assume two different trajectories resulting from simulating the system (6) with parameters and initial conditions $(\mathbf{x}_0, \boldsymbol{\theta})$ and $(\mathbf{w}_0, \boldsymbol{\phi})$, respectively. We denote the corresponding trajectories by $\mathbf{x}$ and $\mathbf{w}$. Let us call:

$$\|\Delta \hat{\mathbf{y}}[k]\| = \|\mathbf{g}(\mathbf{x}[k], \underline{\mathbf{z}}[k]; \boldsymbol{\theta}) - \mathbf{g}(\mathbf{w}[k], \underline{\mathbf{z}}[k]; \boldsymbol{\phi})\|. \qquad (22)$$

Because $\mathbf{h}$ and $\mathbf{g}$ are Lipschitz in $(\mathbf{x}, \boldsymbol{\theta})$ we have:

$$\|\mathbf{h}(\mathbf{x}, \underline{\mathbf{z}}, \boldsymbol{\theta}) - \mathbf{h}(\mathbf{w}, \underline{\mathbf{z}}, \boldsymbol{\phi})\|^2 \leq L_h^2 \left(\|\mathbf{x} - \mathbf{w}\|^2 + \|\boldsymbol{\theta} - \boldsymbol{\phi}\|^2 \right),$$

$$\|\mathbf{g}(\mathbf{x}, \underline{\mathbf{z}}, \boldsymbol{\theta}) - \mathbf{g}(\mathbf{w}, \underline{\mathbf{z}}, \boldsymbol{\phi})\|^2 \leq L_g^2 \left(\|\mathbf{x} - \mathbf{w}\|^2 + \|\boldsymbol{\theta} - \boldsymbol{\phi}\|^2 \right),$$

for all $(\mathbf{x}, \underline{\mathbf{z}}, \boldsymbol{\theta})$ and $(\mathbf{w}, \underline{\mathbf{z}}, \boldsymbol{\phi})$ in $(\Omega_{\mathbf{x}}, \Omega_{\underline{\mathbf{z}}}, \Omega_{\boldsymbol{\theta}})$. Applying these relations recursively we get that:

$$\|\Delta \hat{\mathbf{y}}[k]\|^2 \leq L_g^2 L_h^{2k} \|\mathbf{x}_0 - \mathbf{w}_0\|^2 + L_g^2 \left(\sum_{\ell=0}^{k} L_h^{2\ell} \right) \|\boldsymbol{\theta} - \boldsymbol{\phi}\|^2.$$

Since L_h is positive, the constant multiplying the second term in the above equation is always larger than the constant multiplying the first one. Hence, taking the square root on both sides of the above inequality and after simple manipulations, we get:

$$\|\Delta \hat{\mathbf{y}}[k]\| \leq L_g S(k) \|[\boldsymbol{\theta}, \mathbf{x}_0]^T - [\boldsymbol{\phi}, \mathbf{w}_0]^T\|. \qquad (23)$$

where:

$$S(k) = \sqrt{\sum_{\ell=0}^{k} L_h^{2\ell}} = \begin{cases} \sqrt{k+1} & \text{if } L_h = 1 \\ \sqrt{\dfrac{L_h^{2k+2} - 1}{L_h^2 - 1}} & \text{if } L_h \neq 1. \end{cases} \qquad (24)$$

Since Ω is compact and $\hat{\mathbf{y}}[k]$ is a (Lipschitz) continuous function of the parameters and initial conditions, then $\hat{\mathbf{y}}[k]$ is bounded in Ω, i.e. $\|\hat{\mathbf{y}}[k]\| \leq M(k)$. And, it follows from (23) and from the existence of an invariant set[5] in Ω that $M(k) = \mathcal{O}(S(k))$.

The following inequality follows from (1):

$$|V(\boldsymbol{\theta}, \mathbf{x}_0) - V(\boldsymbol{\phi}, \mathbf{w}_0)| \leq \frac{2}{N} \sum_{k=1}^{N} (L_y + M(k)) \|\Delta \hat{\mathbf{y}}[k]\|, \qquad (25)$$

---

[5] There are multiple ways to guarantee the invariant set premise will hold, but a very simple way is to just choose $\mathbf{h}$ such that $\mathbf{h}(\mathbf{0}, \underline{\mathbf{z}}; \mathbf{0}) = \mathbf{0}$. In this case, $\{\mathbf{0}\}$ is an invariant set and if $\Omega_{\boldsymbol{\theta}}$ contains this point the premise is satisfied. For this specific case, one can just choose $[\boldsymbol{\phi}, \mathbf{w}_0] = \mathbf{0}$ and it follows from (23) that $\|\hat{\mathbf{y}}[k]\| \leq L_g S(k) \|[\boldsymbol{\theta}, \mathbf{x}_0]\| = \mathcal{O}(S(k))$. The more general case, for any invariant set, follows from a similar deduction.

where $L_y = \max_{1 \leq k \leq N} \|\mathbf{y}[k]\|$. And, by putting together (25) and (23):

$$|V(\boldsymbol{\theta}, \boldsymbol{x}_0) - V(\boldsymbol{\phi}, \boldsymbol{w}_0)| \leq L_{V_1} \left\| [\boldsymbol{x}_0, \boldsymbol{\theta}]^T - [\boldsymbol{w}_0, \boldsymbol{\phi}]^T \right\|,$$

for $L_V = \left(\frac{2L_g}{N} \sum_{k=1}^N (L_y + M(k)) S(k) \right)$. The asymptotic analysis of this expression with regard to N yields (8).

B.3. Proof of Theorem 1 (b)

It follows from (20) that:

$$\|\nabla V(\boldsymbol{\theta}, \boldsymbol{x}_0) - \nabla V(\boldsymbol{\phi}, \boldsymbol{w}_0)\| \leq \frac{2}{N} \sum_{k=1}^N L_y \|\Delta J[k]\| + \|\Delta(J[k]\hat{\mathbf{y}}[k])\|, \quad (26)$$

where we have used the notation $\Delta J[k]$ to denote the difference between $J[k]$ evaluated at $(\boldsymbol{\theta}, \boldsymbol{x}_0)$ and $(\boldsymbol{\phi}, \boldsymbol{w}_0)$. Analogously, $\Delta(J[k]\hat{\mathbf{y}}[k])$ denote the difference between $J[k]\hat{\mathbf{y}}[k]$ evaluated at the two distinct points.

From Eqs. (18) and (19) it follows that:

$$J_{\boldsymbol{\theta}}[k] = C_k \sum_{\ell=1}^k \left(\prod_{j=1}^{k-\ell} A_{k-j+1} \right) B_\ell + F_k; \quad J_{\mathbf{x}_0}[k] = C_k \prod_{\ell=1}^k A_{k-\ell+1}. \quad (27)$$

Since, the Jacobian of $\mathbf{h}$ is Lipschitz with Lipschitz constant L_h', it follows that:

$$\|\Delta A_j\|^2 \leq (L_h')^2 \left(\|\mathbf{x}[j] - \mathbf{w}[j]\|^2 + \|\boldsymbol{\theta} - \boldsymbol{\phi}\|^2 \right). \quad (28)$$

Using a procedure analogous to the one used to get Eq. (23), it follows that:

$$\|\Delta A_j\| \leq L_h' S(j) \ \|[\boldsymbol{\theta}, \boldsymbol{x}_0]^T - [\boldsymbol{\phi}, \boldsymbol{w}_0]^T\|, \quad (29)$$

where $S(j)$ is defined as in (24). An identical formula holds for B_j and a similar formula, replacing L_h' with L_g', holds for C_j and F_j.

Since $\mathbf{h}$ and $\mathbf{g}$ are Lipschitz with Lipschitz constants L_h and L_g it follows that $\|A_j\| \leq L_h$, $\|B_j\| \leq L_h$, $\|C_j\| \leq L_g$ and $\|F_j\| \leq L_g$. Hence, it follows from (23), (27), (29) and the repetitive application of Lemma 4 that $\|\Delta J_{\boldsymbol{\theta}}[k]\|$, $\|\Delta J_{\mathbf{x}_0}[k]\|$, $\|\Delta(J_{\boldsymbol{\theta}}[k]\hat{\mathbf{y}}[k])\|$ and $\|\Delta(J_{\mathbf{x}_0}[k]\hat{\mathbf{y}}[k])\|$ are upper bounded by $\|[\boldsymbol{\theta}, \boldsymbol{x}_0]^T - [\boldsymbol{\phi}, \boldsymbol{w}_0]^T\|$ multiplied by the following constants:

$$L_{J_{\boldsymbol{\theta}}}(k) = \sum_{\ell=1}^k P(k, \ell) + L_g' S(k) \ ; \quad L_{J_{\mathbf{x}_0}}(k) = P(k, 1)$$

$$L_{J_{\boldsymbol{\theta}}\hat{\mathbf{y}}}(k) = \sum_{\ell=1}^k Q(k, \ell) + T(k)S(k) \ ; \quad L_{J_{\mathbf{x}_0}\hat{\mathbf{y}}}(k) = Q(k, 1),$$

where $T(k) = (L_g' M(k) + L_g^2)$ and:

$$P(k, \ell) = L_h^{k-\ell} \left(L_g L_h' \sum_{j=\ell}^k S(j) + L_h L_g' S(k) \right)$$

$$Q(k, \ell) = L_h^{k-\ell} \left(M(k) L_g L_h' \sum_{j=\ell}^k S(j) + L_h T(k) S(k) \right).$$

Hence,

$$\|\nabla V(\boldsymbol{\theta}, \boldsymbol{x}_0) - \nabla V(\boldsymbol{\phi}, \boldsymbol{w}_0)\| \leq L_V' \|[\boldsymbol{\theta}, \boldsymbol{x}_0]^T - [\boldsymbol{\phi}, \boldsymbol{w}_0]^T\|,$$

where

$$L_V' = \frac{2}{N} \sum_{k=1}^N \left(L_y (L_{J_{\boldsymbol{\theta}}}(k) + L_{J_{\mathbf{x}_0}}(k)) + L_{J_{\boldsymbol{\theta}}\hat{\mathbf{y}}}(k) + L_{J_{\boldsymbol{\theta}}\hat{\mathbf{y}}}(k) \right).$$

Putting everything together the asymptotic analysis of L_V' results in (9).

Appendix C. Lipschitz analysis for the multiple shooting

The next theorem follows from basic inequality manipulation and relates the Lipschitzness and β-smoothness of the cost function V^M with that of its components V_i.

Theorem 5. Defining V^M as in (12), if each component V_i is Lipschitz continuous with constant L_{V_i} then V^M is also Lipschitz with constant equal to or smaller than $L_{V^M} = \max(L_{V_1}, \ldots, L_{V_M})$. Additionally, if the gradient of each component ∇V_i is Lipschitz continuous with constant L_{V_i}' then ∇V^M is also Lipschitz with constant equal to or smaller than $L_{V^M}' = \max(L_{V_1}', \ldots, L_{V_M}')$.

Proof. For $\boldsymbol{\theta}_{\text{ext}} = (\boldsymbol{\theta}, \mathbf{x}_0^1, \ldots, \mathbf{x}_0^M)$ and $\boldsymbol{\phi}_{\text{ext}} = (\boldsymbol{\phi}, \mathbf{w}_0^1, \ldots, \mathbf{w}_0^M)$ we have that:

$$|V^M(\boldsymbol{\theta}_{\text{ext}}) - V^M(\boldsymbol{\phi}_{\text{ext}})| \leq$$

$$\sum_{i=1}^M \frac{\Delta m_i}{N} |V_i(\boldsymbol{\theta}, \mathbf{x}_0^i) - V_i(\boldsymbol{\phi}, \mathbf{w}_0^i)| \leq$$

$$\sum_{i=1}^M \frac{\Delta m_i}{N} L_{V_i} \|[\boldsymbol{\theta}, \mathbf{x}_0^i]^T - [\boldsymbol{\phi}, \mathbf{w}_0^i]^T\| \leq$$

$$L_{V^M} \|\boldsymbol{\theta}_{\text{ext}} - \boldsymbol{\phi}_{\text{ext}}\|,$$

where $L_{V^M} = \max(L_{V_1}, \ldots, L_{V_M})$. And similarly, $L_{V^M}' = \max(L_{V_1}', \ldots, L_{V_M}')$, which yields the second result.

Appendix D. Asymptotic properties of prediction error methods

D.1. Notation and data generation process

Consider $\mathbf{y}[k]$ and $\mathbf{u}[k]$ to be one realization of the random variables $\mathbf{Y}[k]$ and $\mathbf{U}[k]$. And denote:

$$\underline{\mathbf{U}}[k] = [\mathbf{U}[k], \ldots, \mathbf{U}[k - n_u]],$$
$$\underline{\mathbf{Y}}[k - 1] = [\mathbf{Y}[k - 1], \ldots, \mathbf{Y}[k - n_y]].$$

Hence, rewriting the data generation difference equations for nonlinear ARX, output error and ARMAX (see Sections 2.1–2.3) with the random variables yield, respectively:

- $\mathbf{Y}[k] = \mathbf{f}^*(\underline{\mathbf{Y}}[k - 1], \underline{\mathbf{U}}[k]) + \mathbf{V}[k]$
- $\begin{cases} \bar{\mathbf{Y}}[k] = \mathbf{f}^*(\bar{\mathbf{Y}}[k - 1], \ldots, \bar{\mathbf{Y}}[k - n_y], \underline{\mathbf{U}}[k]) \\ \mathbf{Y}[k] = \bar{\mathbf{Y}}[k] + \mathbf{V}[k], \end{cases}$
- $\mathbf{Y}[k] = \mathbf{f}^*(\underline{\mathbf{Y}}[k - 1], \underline{\mathbf{U}}[k], \mathbf{V}[k - 1], \ldots, \mathbf{V}[k - n_v]) + \mathbf{V}[k].$

Here $\mathbf{V}[k]$ is a random variable representing the noise that is injected in the system. Notice that there is a deterministic additive relation between $\mathbf{Y}[k]$ and $\mathbf{V}[k]$. Hence, if $\mathbf{Y}[k]$ and $\mathbf{U}[k]$ are determined, so is $\mathbf{V}[k]$, or, conversely, if $\mathbf{V}[k]$ and $\mathbf{U}[k]$ are determined, so is $\mathbf{Y}[k]$.

D.2. Optimal output prediction

Let us define the *optimal output prediction* at time k as the following conditional expectation:

$$\hat{\mathbf{y}}_*[k] = E \left\{ \mathbf{Y}[k] \ \Big| \ \underline{\mathbf{U}}[k] = \underline{\mathbf{u}}[k], \ \underline{\mathbf{Y}}[k - 1] = \underline{\mathbf{y}}[k - 1] \right\}, \quad (30)$$

which is, in the least square sense, the best prediction for the output given its previous values.[6]

---

[6] This prediction provides the smallest squared conditional expected error between the predicted and observed values:
$$\hat{\mathbf{y}}_*[k] = \arg_{\hat{\mathbf{Y}}} \min E \left\{ \|\mathbf{Y}[k] - \hat{\mathbf{Y}}\|^2 \ \Big| \ \underline{\mathbf{u}}[k], \underline{\mathbf{y}}[k - 1] \right\}.$$

For the nonlinear ARX, output error and ARMAX models the *optimal output predictions* are given, respectively, by:

- $\hat{\mathbf{y}}_*[k] = \mathbf{f}^*(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k])$.

- $\begin{cases} \bar{\mathbf{y}}[k] = \mathbf{f}^*(\bar{\mathbf{y}}[k-1], \ldots, \bar{\mathbf{y}}[k-n_y], \underline{\mathbf{u}}[k]) \\ \hat{\mathbf{y}}_*[k] = \bar{\mathbf{y}}[k], \end{cases}$

- $\begin{cases} \tilde{\mathbf{v}}[k] = \mathbf{y}[k] - \mathbf{f}^*(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k], \tilde{\mathbf{v}}[k-1], \ldots, \tilde{\mathbf{v}}[k-n_v]) \\ \hat{\mathbf{y}}_*[k] = \mathbf{f}^*(\underline{\mathbf{y}}[k-1], \underline{\mathbf{u}}[k], \tilde{\mathbf{v}}[k-1], \ldots, \tilde{\mathbf{v}}[k-n_v]) \end{cases}$

which follows from a direct application of the definition (30) to the stochastic difference equations that are assumed for the data generation in each case.

D.3. Ideal cost function

Ideally, the model predicted output $\hat{\mathbf{y}}[k]$ should be as close as possible to the optimal one $\hat{\mathbf{y}}_*[k]$. The distance between the two series is quantified by the cost function:

$$V_* = \frac{1}{N} \sum_{k=1}^{N} \|\hat{\mathbf{y}}_*[k] - \hat{\mathbf{y}}[k]\|^2. \tag{31}$$

Now, back to the examples, given a parametrized function $\mathbf{f}_\theta$ with the correct model structure (that is: there exists θ^* such that $\mathbf{f}_{\theta^*} = \mathbf{f}^*$), it follows from Eqs. (3), (4) and (5) that θ^* yields $V_* = 0$, and hence θ^* is a minimizer of V_*.

D.4. Uniform convergence of V

The optimal cost function V_* is not available for optimization. Nevertheless, under some mild regularity conditions, it has been proved that $V \to V_*$ in probability as $N \to \infty$ and that *this convergence is uniform* (Ljung, 1978).

If V_* has a single minimum θ^* and V is convex in a convex set containing θ^* the minimizer of V converges to the minimizer of V_* (Newey & McFadden, 1994, Theorem 2.1). Alternative conditions for this to hold are given in Ljung (1978). For our three examples, this would imply $\mathbf{f}_\theta \to \mathbf{f}^*$ (convergence in probability for $N \to \infty$). If the conditions are not satisfied, the uniform convergence, at least, guarantees that the minimizer has an equivalent performance (as $N \to \infty$). Additionally, in Ljung and Caines (1980) it is shown that if the solution is unique the estimator has an asymptotic normal distribution.

Appendix E. Computational cost of MSA-PEM

As mentioned in Section 6.3, the computation of the cost function V (and its derivatives) for the MSA-PEM method has linear time complexity with K. The rate at which the time complexity grows, however, depends on implementation choices. We use the model estimation described in the experiment in Section 6.2 to explain the tradeoffs of those choices. Propagating $K = 3$ steps-ahead the linear system $y[k] = \theta_1 y[k] + \theta_2 y[k-2] + \theta_3 u[k-1]$ can be performed in two ways. The first is to introduce intermediate variables:

$$\tilde{y}_k[k-2] \leftarrow \theta_1 y[k-3] + \theta_2 y[k-4] + \theta_3 u[k-3],$$
$$\tilde{y}_k[k-1] \leftarrow \theta_1 \tilde{y}_k[k-2] + \theta_2 y[k-3] + \theta_3 u[k-2]$$
$$\hat{y}[k] \leftarrow \theta_1 \tilde{y}_k[k-1] + \theta_2 \tilde{y}_k[k-2] + \theta_3 u[k-1].$$

The other is to simplify the computation in advance, resulting in a system of higher order:

$$\hat{y}[k] \leftarrow \theta_1' y[k-3] + \theta_2' y[k-4] +$$
$$+\theta_3' u[k-1] + \theta_4' u[k-2] + \theta_5' u[k-3],$$

where the θ' coefficients depend on θ (for instance $\theta_1' = \theta_1^3 - 2\theta_1\theta_2$) and are computed in advance. For the first approach, computing the cost function has time-complexity $\mathcal{O}(NK(n_y + n_u))$; for the second, the computational cost is slightly smaller: $\mathcal{O}(N(K + n_u + n_y))$. The reduced computational cost is obtained because some computations are performed in advance and stored. Hence, the additional time-efficiency comes at the cost of storing the computation in advance, and the memory of the first method is $\mathcal{O}(n_y + n_u)$, while in the second approach the memory complexity is $\mathcal{O}(K + n_y + n_u)$.

For linear systems, it is easy to find the optimal simplified computation in advance by using Diophantine equations (Farina & Piroddi, 2011). In the experiment in Section 6.2, however, we use the less time-efficient approach. The general tone of our discussion justifies this choice since this implementation can easily be applied to generic systems. And, while heuristic solutions might be employed to simplify the computations in advance, there is no general approach for all types of nonlinear systems. Regardless of this design choice, however, the observed linear growth of the computational cost with K is expected.

Appendix F. Additional experiments

F.1. Neural network for modeling pilot plant

This example makes use of data from the level process station described in Example 1 from Ribeiro and Aguirre (2018). As in the original paper, we use a neural network to model the water column height as a function of the voltage applied to a control valve that modulates the water flow. We compare three different training methods: (i) minimizing the one-step-ahead prediction error (NN ARX) (ii) minimizing the free-run simulation error using single shooting method (NN OE – SS); and, (iii) minimizing the free-run simulation error using multiple shooting method (NN OE – MS).

The neural network (NN) training depends on the weight initialization, hence the performance of the neural network can be regarded as a random variable and is displayed in Fig. 7, which compares the empirical cumulative distribution of the *mean square error* (MSE) over the validation dataset for the three methods. A linear ARX model ($n_y = 1$ and $n_u = 1$) was trained and tested under the same conditions to serve as the baseline. Methods (i) and (ii) and the linear ARX baseline were described in Ribeiro and Aguirre (2018). Method (iii) is introduced in this paper.

The cumulative distribution function gives, for each x-axis value, the probability of the method to yield a validation MSE smaller than or equal to this value. It was estimated from 100 realizations of the neural network training procedure. Fig. 7 shows that for more than 90% of the realizations, estimating the parameters by minimizing the free-run simulation (NN OE) offers significant advantages over the minimization of the one-step-ahead error (NN ARX). When using a standard single shooting formulation, however, it also makes the parameter estimation procedure more sensitive to the initial conditions, with the algorithm yielding some really bad results for some initial choices (Ribeiro & Aguirre, 2018). This results in a long-tailed distribution for the MSE (Fig. 7). More precisely, in 10 out of 100 realizations the NN OE – SS model yields a performance that is inferior to the linear ARX baseline, some of the realizations worse than the linear baseline by a factor of 100. The performance of the NN OE – SS and NN OE – MS is very similar for 90% of the realizations, the tail of the distribution, however, is very different, with the multiple shooting procedure rarely producing very bad results. In order to highlight the differences, results where NN OE – SS and NN OE
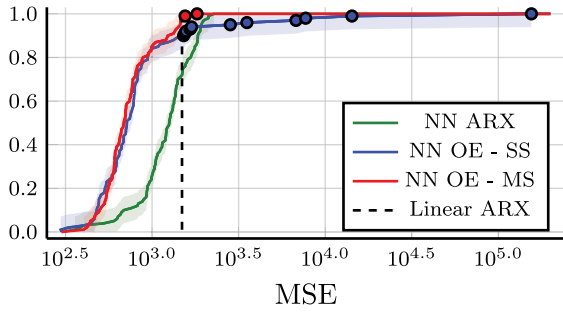
Fig. 7. Neural network performance. Displays the empirical cumulative distribution of the free-run simulation MSE over the validation dataset. The results obtained in Ribeiro and Aguirre (2018) for an ARX neural network (NN ARX) and an (single shooting) output error neural network (NN OE - SS) are displayed together with the result obtained estimating the parameters using multiple shooting (NN OE - MS). A Linear ARX model is considered as a baseline and is displayed by the dashed line. The multiple shooting estimation uses $\Delta m_{max} = 3$ and the training is restricted to 2000 iterations of the optimization algorithm or until either the gradient or the step size drops below 10^{-12}. The other models were estimated exactly as in Ribeiro and Aguirre (2018). All the neural network models have 10 nodes in the hidden layer, $n_y = n_u = 1$ and were trained with the same training dataset. Each curve is the result of 100 realizations and, for each realization, the neural network initial weights $w_{i,j}^{(n)}$ are drawn from a normal distribution with zero-mean and standard deviation $\sigma = (N_{s(n-1)})^{-0.5}$ and the bias terms $\gamma_i^{(n)}$ are initialized with zeros (LeCun, Bottou, Orr, & Müller, 1998). Realizations of NN OE - SS and NN OE - MS that perform worse than the baseline are indicated respectively as blue, ●, and red circles, ●. Confidence intervals (95%) are displayed as shaded regions around the estimated cumulative distribution, these have been computed using the Dvoretzky–Kiefer–Wolfowitz inequality (Dvoretzky, Kiefer, & Wolfowitz, 1956).

– MS are worse than the baseline are presented, respectively, as blue and red circles in Fig. 7.

This example illustrates how the use of multiple shooting alleviates the problem of high sensitivity to initial conditions, making it possible to estimate output error models with extra robustness against variations of the initial conditions and lower probability of getting trapped at local minima with very bad performance.

This example also shows the limitations of the multiple shooting formulation. The training time for NN OE – MS model is 282 s, for NN OE – SS model is 3.9 s, and for NN ARX model is 3.3 s. This means that the single shooting parameter estimation could be repeated, roughly, 70 times for each multiple shooting run. Hence solving the single shooting problem several times and choosing the best result would also avoid very bad solutions and could, still, be computationally less expensive than solving the multiple shooting problem. The longer training time is due to two factors: (i) per iteration the multiple shooting approach takes, roughly, 3.5 times more than the single shooting approach; and, (ii) it takes, approximately, 20 times more iterations to converge. Both are consequences of the fact that a higher dimensional *constrained* optimization problem is being solved.

F.2. Incremental initialization approach for the MSA-PEM

Here, we revisit the setup of the first numerical example presented in Farina and Piroddi (2012). A dataset with $N = 500$ samples is generated using the equation:

$$\bar{y}[k] = \theta_1 u[k-1]u[k-2] + \theta_2 u[k-1]\bar{y}[k-1] \tag{32a}$$

$$y[k] = \bar{y}[k] + v[k]. \tag{32b}$$

The data generator parameters are $\theta_1 = 0.6$ and $\theta_2 = -0.5$, $\bar{y}$ represents the noiseless output and v the white output noise. The input u is generated by the following AR process:

$$u[k] = 0.99u[k-1] + 0.1\eta[k], \tag{33}$$

where η is white Gaussian noise with variance one. The output noise v has variance $0.25\lambda_y^2$, where $\lambda_y^2 = 0.18^2$ is the output variance.

A model with the same structure as the data generator is estimated using MSA-PEM. Fig. 8 displays the cost function and the result of parameter estimation for several initial optimization conditions. Besides the results of the vanilla implementation, in blue, we also show, in black, the refinement proposed in Farina and Piroddi (2012). For this refinement, the prediction starts from the one-step-ahead solution, and in each iteration increases K by one. That is, the K MSA-PEM problem is initialized from the (partial) solution of the $K-1$ MSA-PEM estimation. The intermediary results for the refined implementation are displayed in the plot. With the algorithm being iterated up to the Kth iteration.

The effect of increasing the simulation length can be observed in this example, with a larger K yielding less smooth cost functions. The refinement proposed in Farina and Piroddi (2012) allows the estimation to approach the desired asymptotic statistical properties while optimizing more complicated cost functions progressively. The results in Fig. 8, show this refinement yield improvements over the vanilla implementation, maybe because it avoids optimizing the harder optimization problems from a distant initial estimate.

Appendix G. Equality-constrained optimization solver

This appendix describes the implementation of the solver described in Lalee et al. (1998), which is used in the numerical examples. The method is able to solve large-scale equality-constrained optimization problems:

$$\min_{\theta} V(\theta), \tag{34}$$

subject to $\mathbf{c}(\theta) = 0$

where $V: \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{c}: \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable functions.

The algorithm solves a sequence of Taylor approximations in order to gradually converge towards a *local* solution of (34). At the kth iteration, the algorithm builds a local model around the current iterate θ_k, computes the update $\mathbf{p}_k$ by solving a *quadratic programming* problem and then updates the solution:

$$\theta_{k+1} = \theta_k + \mathbf{p}_k. \tag{35}$$

The algorithm is a *trust-region method* (Conn, Gould, & Toint, 2000), in the sense that the update must respect $\|\mathbf{p}_k\| \le \Delta_k$, for which Δ_k is known as *trust radius* and should reflect the trust the algorithm has on the local approximation of the cost function. If the current local approximation is not a good one, the method does not allow a very large step to be taken.

G.1. Quadratic programming subproblem

At each iteration, in order to compute the step update $\mathbf{p}_k$, the method solves the trust-region quadratic programming (QP) sub-problem:

$$\min_{\mathbf{p}} \nabla V(\theta_k)^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \nabla_{\theta\theta}^2 \mathcal{L}(\theta_k, \lambda_k)^T \mathbf{p}, \tag{36}$$

subject to: $J(\theta_k)\mathbf{p} + \mathbf{c}(\theta_k) = \mathbf{r}_k$;

$$\|\mathbf{p}\| \le \Delta_k,$$

where θ_k is the current iterate, λ_k is the current approximation of Lagrange multipliers, $\nabla V(\theta_k)^T$ is the gradient of V and $J(\theta_k)$ is the Jacobian matrix of $\mathbf{c}$. As in Appendix A, $\mathcal{L}(\theta_k, \lambda_k)^T$ denotes the Lagrangian:
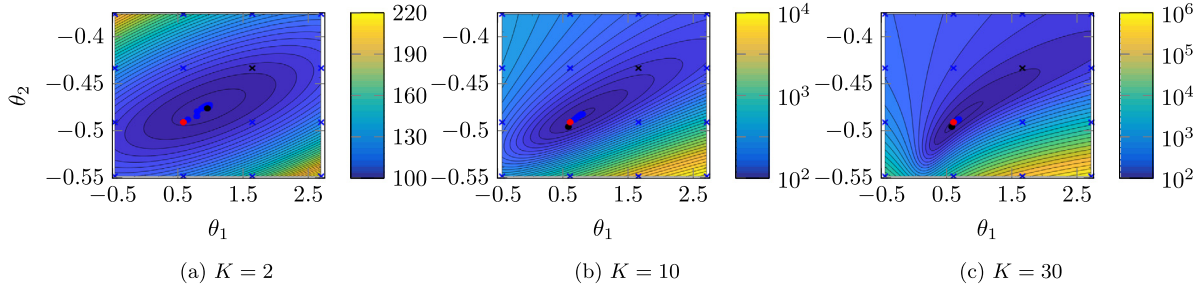
Fig. 8. Contour plot of the MSA-PEM cost function. The figures correspond to the K-step-ahead cost function for increasing values of K. The true parameters are indicated with a red circle ●. Initial conditions are indicated with blue crosses ✕ and the resulting estimated parameters with blue circles ●. The default method in Farina and Piroddi (2012) is to start from the one-step-ahead solution (✕) and increase K by one in each iteration until convergence. The intermediary results (up to the $K = 2, 10, 30$ iteration) for the refined implementation are displayed in the plot and are indicated with a black circle ●. In (a), the color scale is linear, and in (b) and (c), the color scale is logarithmic. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k) = V(\boldsymbol{\theta}_k) + \boldsymbol{\lambda}_k^T \mathbf{c}(\boldsymbol{\theta}_k), \tag{37}$$

and $\nabla^2_{\theta\theta} \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k)^T$ is the Hessian in relation to the variable $\boldsymbol{\theta}$.

The current approximation of the Lagrange multipliers $\boldsymbol{\lambda}_k$ is obtained by solving the least-squares problem:

$$\min_{\boldsymbol{\lambda}} \|\nabla_x \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k)\|^2 \Rightarrow \min_{\boldsymbol{\lambda}} \|\nabla V(\boldsymbol{\theta}_k) + J(\boldsymbol{\theta}_k)\boldsymbol{\lambda}\|^2. \tag{38}$$

The vector $\mathbf{r}_k$ is included in the problem in order to guarantee linear and trust-region constraints are always compatible. $\mathbf{r}_k$ is defined as $J(\boldsymbol{\theta}_k)\mathbf{v}_k + \mathbf{c}(\boldsymbol{\theta}_k)$ for a value of $\mathbf{v}_k$ satisfying:

$$\min_{\mathbf{v}} \|J(\boldsymbol{\theta}_k)\mathbf{v} + \mathbf{c}(\boldsymbol{\theta}_k)\|^2, \tag{39}$$

subject to $\|\mathbf{v}\| \leq \eta \Delta_k$,

for $0 < \eta < 1$ (in our implementation $\eta = 0.8$). For this choice of $\mathbf{r}_k$ the linear constraints are always compatible with trust-region constraints.

The sub-steps are solved in rather economical fashion using efficient methods to get an *inexact* solution to each of the sub-problems. The QP problem (36) is solved using a variation of the *dogleg* procedure (described in Byrd, Hribar, and Nocedal (1999), p.886) and (39) is solved using the projected conjugate gradient (CG) algorithm (Gould, Hribar, & Nocedal, 2001).

G.2. Implementation details

Here we define the *merit function*, ϕ, which combines the constraints and the objective function into a single number that can be used to compare two points and to reject or accept a given step. It is given by:

$$\phi(\boldsymbol{\theta}; \mu) = V(\boldsymbol{\theta}) + \mu \|\mathbf{c}(\boldsymbol{\theta})\|, \tag{40}$$

where the *penalty parameter* μ is updated through the iterations. This parameter must increase monotonically for the algorithm to converge. Some additional guidelines are provided in Byrd et al. (1999), p.891.

The selection of the trust radius Δ_k and the step rejection mechanism are based on the ratio:

$$\rho_k = \frac{\phi(\boldsymbol{\theta}_k; \mu) - \phi(\boldsymbol{\theta}_k + \mathbf{p}_k, \mu)}{q_\mu(0) - q_\mu(p_k)}, \tag{41}$$

between the reduction of the merit function and the reduction predicted by the local (Taylor approximation) model:

$$q_\mu(p) = \nabla V(\boldsymbol{\theta}_k)^T p + \frac{1}{2} p^T \nabla^2_{xx} \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k)p$$
$$+ \mu \|c(\boldsymbol{\theta}_k) + J(\boldsymbol{\theta}_k)p\|.$$

The ratio measures the agreement between the observed and expected reduction and is used as decision variable when accepting or rejecting the update and enlarging or reducing the trust radius Δ_k.

G.3. Algorithm overview

The full algorithm is summarized next.

Algorithm 2 (*Trust-region Sequential QP Solver*). At each iteration, until some stop criterion is met (e.g. $\|\nabla_x \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k)\|_\infty < 10^{-8}$), repeat:

(1) Compute $V(\boldsymbol{\theta}_k)$, $\nabla V(\boldsymbol{\theta}_k)$, $c(\boldsymbol{\theta}_k)$ and $J(\boldsymbol{\theta}_k)$;
(2) Compute least squares Lagrange multipliers $\boldsymbol{\lambda}_k$;
(3) Compute $\nabla^2_{xx} \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k)$;
(4) Apply dogleg method in order to compute v_k and r_k (such that the resulting problem is feasible);
(5) Compute p_k using the projected CG method;
(6) Choose penalty parameter μ_k;
(7) Compute reduction ratio ρ_k;
(8) Accept or reject step p_k using ρ_k as decision variable;
(9) Enlarge or reduce trust-radius using ρ_k as decision variable.

References

Aguirre, L. A., Barbosa, B. H., & Braga, A. P. (2010). Prediction and simulation errors in parameter estimation for nonlinear systems. *Mechanical Systems and Signal Processing*, 24(8), 2855–2867.

Baake, E., Baake, M., Bock, H., & Briggs, K. (1992). Fitting ordinary differential equations to chaotic data. *Physical Review A*, 45(8), 5524.

Bock, H. (1983). Recent advances, parameter identification problems for ODE. In *Numerical treatment of inverse problems in differential and integral equations* (pp. 95–121).

Bock, H. G., & Plitt, K.-J. (1984). A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2), 1603–1608.

Boyd, S., & Chua, L. (1985). Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems*, 32, 1150–1161.

Byrd, R. H., Hribar, M. E., & Nocedal, J. (1999). An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4), 877–900.

Carraro, T., Geiger, M., & Rannacher, R. (2014). Indirect multiple shooting for nonlinear parabolic optimal control problems with control constraints. *SIAM Journal on Scientific Computing*, 36(2), A452–A481.

Conn, A. R., Gould, N. I. M., & Toint, P. L. (2000). *MPS-SIAM series on optimization, Trust-region methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Dvoretzky, A., Kiefer, J., & Wolfowitz, J. (1956). Asymptotic Minimax Character of the Sample Distribution function and of the Classical Multinomial Estimator. *The Annals of Mathematical Statistics*, 27, 642–669.

Eckhard, D., Bazanella, A. S., Rojas, C. R., & Hjalmarsson, H. (2017). Cost function shaping of the output error criterion. *Automatica*, 76, 53–60.

Farina, M., & Piroddi, L. (2011). Simulation error minimization identification based on multi-stage prediction. *International Journal of Adaptive Control and Signal Processing*, 25(5), 389–406.

Farina, M., & Piroddi, L. (2012). Identification of polynomial input/output recursive models with simulation error minimisation methods. *International Journal of Systems Science*, 43(2), 319–333.

Geisert, M., & Mansard, N. (2016). Trajectory Generation for Quadrotor based systems using Numerical Optimal Control. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 2958–2964). IEEE.

Gould, N. I., Hribar, M. E., & Nocedal, J. (2001). On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4), 1376–1395.

Lalee, M., Nocedal, J., & Plantenga, T. (1998). On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, 8(3), 682–706.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K.-R. (1998). Efficient BackProp. In *Lecture notes in computer science, Neural networks: Tricks of the trade* (pp. 9–50). Berlin, Heidelberg: Springer.

Ljung, L. (1978). Convergence analysis of parametric identification methods. *IEEE Transactions on Automatic Control*, 23, 770–783.

Ljung, L. (1999). *System identification: Theory for the user* (2nd ed.). Prentice Hall.

Ljung, L., & Caines, P. E. (1980). Asymptotic normality of prediction error estimators for approximate system models. *Stochastics*, 3, 29–46.

May, R. M. (1976). Simple mathematical models with very complicated dynamics. *Nature*, 261(5560), 459–467.

Nesterov, Y. (1998). *Introductory lectures on convex programming*. Springer Science & Business Media.

Newey, W. K., & McFadden, D. (1994). Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4, 2111–2245.

Noël, J. P., & Schoukens, J. (2018). Grey-box state-space identification of nonlinear mechanical vibrations. *International Journal of Control*, 91, 1118–1139.

Paduart, J., Lauwers, L., Swevers, J., Smolders, K., Schoukens, J., & Pintelon, R. (2010). Identification of nonlinear systems using Polynomial Nonlinear State Space models. *Automatica*, 46, 647–656.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning, Vol. 28* pp. 1310–1318.

Piroddi, L. (2008). Simulation Error Minimisation Methods for NARX Model identification. *International Journal of Modelling, Identification and Control*, 3(4), 392–403.

Piroddi, L., & Spinelli, W. (2003). An identification algorithm for polynomial NARX models based on simulation error minimization. *International Journal of Control*, 76, 1767–1781.

Ribeiro, A. H., & Aguirre, L. A. (2017). Shooting methods for parameter estimation of Output Error Models. *IFAC-PapersOnLine*, 50, 13998–14003.

Ribeiro, A. H., & Aguirre, L. A. (2018). Parallel Training Considered Harmful?: Comparing series-parallel and parallel feedforward network training. *Neurocomputing, 316*, 222–231.

Ribeiro, A. H., Tiels, K., Aguirre, L. A., & Schön, T. B. (2020). Beyond exploding and vanishing gradients: Attractors and smoothness in the analysis of recurrent neural network training. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS), Vol. 108* pp. 2370–2380.

Rudin, W. (1964). *International series in pure and applied mathematics, Principles of mathematical analysis*. McGraw-Hill.

Sarode, K. D., Kumar, V. R., & Kulkarni, B. (2015). Embedded multiple shooting methodology in a genetic algorithm framework for parameter estimation and state identification of complex systems. *Chemical Engineering Science*, 134, 605–618.

Schoukens, M., & Tiels, K. (2017). Identification of block-oriented nonlinear systems starting from linear approximations: A survey. *Automatica, 85*, 272–292.

Su, H. T., McAvoy, T. J., & Werbos, P. (1992). Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial and Engineering Chemistry Research*, 31(5), 1338–1352.

Terzi, E., Fagiano, L., Farina, M., & Scattolini, R. (2018). Learning multi-step prediction models for receding horizon control. In *2018 European control conference (ECC)* (pp. 1335–1340).

Van Mulders, A., Schoukens, J., Volckaert, M., & Diehl, M. (2010). Two nonlinear optimization methods for black box identification compared. *Automatica, 46*, 1675–1681.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al., and S. . Contributors (2020). SciPy 1.0–Fundamental Algorithms for scientific computing in Python. *Nature Methods*.

Antônio H. Ribeiro was born in Brazil in 1992. He received his B.Sc, M.Sc. and Ph.D. degrees in Electrical Engineering from the Federal University of Minas Gerais (Brazil) in 2016, 2017 and 2020, respectively. He is currently a postdoctoral associate in the Computer Science department of the same university. He has held a visiting position in the Department of Information Technology at Uppsala University (Sweden) during one year of his Ph.D. His research interest include system identification, signal processing, machine learning and its applications to health care and medicine.

Koen Tiels received the degree of master in Electromechanical Engineering in July 2010 and the degree of Doctor in Engineering (Ph.D.) in March 2015, both from the Vrije Universiteit Brussel (VUB), Brussels, Belgium. He was a post-doctoral researcher in the period 2015–2018 at the same university. From February 2018 until January 2020, he joined the Department of Information Technology, Division of Systems and Control at Uppsala University, as a postdoctoral researcher. Since February 2020, Koen is an Assistant Professor within the Control Systems Technology (CST) Group, Department of Mechanical Engineering at the Eindhoven University of Technology. His main interests are in the field of nonlinear system identification and machine learning.

Jack Umenberger received the B.E. (Hons 1) and Ph.D. degrees in Mechatronics Engineering from The University of Sydney, Australia, in 2012 and 2017, respectively. He is currently a Postdoctoral Associate at the Massachusetts Institute of Technology, and was previously a postdoctoral researcher at Uppsala University, Sweden. His research interests include system identification, robust adaptive control, and reinforcement learning

Thomas B. Schön is the Beijer Professor of Artificial Intelligence in the Department of Information Technology at Uppsala University (Sweden). He received the Ph.D. degree in Automatic Control in Feb. 2006, the M.Sc. degree in Applied Physics and Electrical Engineering in Sep. 2001, the B.Sc. degree in Business Administration and Economics in Jan. 2001, all from Linköping University. He has held visiting positions with the University of Cambridge (UK), the University of Newcastle (Australia) and Universidad Técnica Federico Santa María (Valparaíso, Chile). In 2018, he was elected to The Royal Swedish Academy of Engineering Sciences (IVA) and The Royal Society of Sciences at Uppsala. He received the Tage Erlander prize for natural sciences and technology in 2017 and the Arnberg prize in 2016, both awarded by the Royal Swedish Academy of Sciences (KVA). He was awarded the Automatica Best Paper Prize in 2014, and in 2013 he received the best Ph.D. thesis award by The European Association for Signal Processing (EURASIP). He received the best teacher award at the Institute of Technology, Linköping University in 2009. Schön has a broad interest in developing new algorithms and mathematical models capable of learning from data. He is a Senior member of the IEEE and a fellow of the ELLIS society.

Luis A. Aguirre received the Ph.D. degree from The University of Sheffield, U.K., in 1994. He joined the Department of Electronics Engineering, Federal University of Minas Gerais (UFMG), in 1995, where he currently serves as a Full Professor. He has authored two books and was the Editor-inChief of Enciclopédia de Automática (3 volume set), sponsored by the Brazilian Society of Automation (SBA) and published by Editora Blücher. His research includes the identification of nonlinear system, gray-box identification, nonlinear dynamics, and the analysis of dynamical networks. From 2009 to 2012, he served as the Editor-in-Chief of Controle & Automação: Revista da Sociedade Brasileira de Automática, (Springer Verlag) under the name Journal of Control, Automation and Electrical Systems.