POLITECNICO DI TORINO
Repository ISTITUZIONALE

Control of MIMO nonlinear systems: A data-driven model inversion approach

Original
Control of MIMO nonlinear systems: A data-driven model inversion approach / Novara, C., Milanese, M.. - In: AUTOMATICA. - ISSN 0005-1098. - 101:(2019), pp. 417-430. [10.1016/j.automatica.2018.12.026]

Availability:
This version is available at: 11583/2800692 since: 2020-03-13T16:50:57Z

Publisher:
Elsevier Ltd

Published
DOI:10.1016/j.automatica.2018.12.026

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Elsevier postprint/Author's Accepted Manuscript

© 2019. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/.The final authenticated version is available online at:
http://dx.doi.org/10.1016/j.automatica.2018.12.026

(Article begins on next page)

17 November 2026

Control of MIMO nonlinear systems: a data-driven model inversion approach

Carlo Novara[a] and Mario Milanese[b]

[a] *Dept. of Electronics and Telecommunications, Politecnico di Torino, Italy. Email:* carlo.novara@polito.it

[b] *Modelway srl, Torino, Italy. Email:* mario.milanese@modelway.it

Abstract

A data-driven control design approach for Multiple Input Multiple Output nonlinear systems is presented in this paper. The approach, called Nonlinear Inversion Control (NIC), is based on the identification of a polynomial prediction model of the system to control and the on-line inversion of this model. The main features of the NIC approach can be summarized as follows: it does not require a physical model of the plant to control which, in many real-world situations, may be difficult to derive; it can guarantee a priori properties such as closed-loop stability and tracking error accuracy; it is general, systematic, numerically efficient and relatively simple. Extensive simulations are carried out to test the numerical efficiency of the NIC approach. A simulated example of industrial interest is also presented, concerned with control of a robotic manipulator.

1 Introduction

1.1 Overview

Consider a MIMO (Multiple Input Multiple Output) nonlinear discrete-time system of the form

$$y_t = h\left(u_t^-, y_t^-, \xi_t^-\right) \tag{1}$$

$$u_t^- \doteq (u_{t-1}, \ldots, u_{t-n})$$
$$y_t^- \doteq (y_{t-1}, \ldots, y_{t-n})$$
$$\xi_t^- \doteq (\xi_{t-1}, \ldots, \xi_{t-n})$$

where $u_t \in U \subset \mathbb{R}^{n_u}$ is the input, $y_t \in \mathbb{R}^{n_y}$ is the output, $\xi_t \in \Xi \subset \mathbb{R}^{n_\xi}$ is an unmeasured noise, n is the system order and $t \in \mathbb{Z}$ is the time. U is a compact set with non-empty interior, accounting for possible input constraints/saturations. The noise set Ξ is defined as $\Xi \doteq \left\{\xi \in \mathbb{R}^{n_\xi} : \|\xi\| \leq \bar{\xi}\right\}$ (in other words, the noise ξ_t is assumed unknown but bounded).

Suppose the system (1) is unknown but a set of input-output measurements, collected from (1), is available:

$$\mathcal{D} \doteq \{\tilde{y}_t, \tilde{u}_t\}_{t=1-L}^{0} \tag{2}$$

where $\tilde{u}_t \in U$, $\tilde{y}_t \in Y$, the set $Y \subset \mathbb{R}^{n_y}$ is compact with non-empty interior, and the tilde is used to indicate the collected data.

Let $\mathcal{Y}^0 \subseteq R^n$ be a set of initial conditions of interest, $R \subset \mathbb{R}^{n_y}$ a compact set with non-empty interior, $\mathcal{R} \doteq \{\boldsymbol{r} = (r_1, r_2, \ldots) : r_t \in R, \forall t\}$ a set of reference signals of interest and $\Xi \doteq \{\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots) : \xi_t \in \Xi, \forall t\}$ the set of all possible noise sequences.

The problem is to control the system (1) such that, for any $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots) \in \Xi$, and for any initial condition $y_0^- \in \mathcal{Y}^0$, the system output sequence $\boldsymbol{y} = (y_1, y_2, \ldots)$ tracks any reference sequence $\boldsymbol{r} = (r_1, r_2, \ldots) \in \mathcal{R}$.

The standard approaches to this problem (or similar control problems) are based on first-principle models: they assume that an accurate physical model describing the system dynamics is available. Typical examples are feedback linearization [30,22], Lyapunov function based control [5,44] and NMPC (Nonlinear Model Predictive Control) [17,36,23]. However, in many real-world applications, deriving an accurate model is a hard task, since the system dynamics is not well known and/or too complex. Another relevant problem is that, even when a reliable model is available, designing an effective (nonlinear) controller may be difficult. Robust methods might be used to deal with the problem of model uncertainty, [21,59,65]. However, deriving the necessary uncertainty models is not an easy task even for LTI (Linear Time Invariant) systems (see e.g. [12] and the references therein), and it is an open problem in the case of nonlinear systems.

Data-driven methods have been introduced to cope with these problems. Such methods fall into three main categories: indirect techniques, which employ the data to identify a black-box model to use for control design, [43,11,3,34,58,8]; direct techniques, which obtain the controller (often in the form of an inverse model) directly from the available input-output data, [66,1,10,2,50,18,16]; internal model control techniques, where a model is identified from data and the controller (obtained by some inversion approach) is either identified from data or designed from the model, [29,7,49]. There is a fourth category, represented by reinforcement learning/adaptive techniques, where the controller (or the model or both of them) is tuned using the data measured during online operations, [58,26,25,35,67].

Since adaptive techniques can be employed within any more general approach, this latter category can be seen as a sub-category of the previous three, depending on which part of the control system is updated online (the model, the controller or both of them). Several data-driven approaches are described in [45], with useful discussions on their practical advantages and drawbacks. Relevant overviews can be found in [27,24]. However, especially when the system is nonlinear and affected by external disturbances, guaranteeing general a-priori stability and accuracy properties is not easy for many data-driven methods. Typically, the controller is first designed using some heuristic or more systematic technique, then it is tested/tuned in simulation, and finally it is implemented (and possibly tuned) on the real plant to control.

In this paper, we propose a novel data-driven control method for MIMO nonlinear systems, called Nonlinear Inversion Control (NIC), allowing us to overcome several problems of the existing methodologies. The main features of the NIC method are the following: it is suitable for general MIMO nonlinear systems affected by external disturbances; it does not require a physical model of the plant to control; it can guarantee a priori properties such as closed-loop stability and tracking error accuracy; it is relatively simple; it is systematic, in the sense that the design parameters can be chosen by means of a clear procedure; it is numerically efficient, allowing a "fast" implementation on real-time processors. The NIC method is based on the identification from the data (2) of a polynomial prediction model for the system (1), and the online inversion of this model through an efficient optimization. A key feature is that closed-loop stability is directly enforced by the identification algorithm used to derive the model, thus avoiding the need of additional on-line operations finalized at stabilization.

1.2 Paper technical contributions and organization

The main contribution of the paper consists in proposing a novel control approach for MIMO nonlinear systems, showing interesting advantages with respect to (wrt) the state of the art methods. While these general advantages have already been discussed in Section 1.1, here we focus on the technical contributions of the paper, which are the following: The first one is an algorithm for the identification of polynomial multi-step prediction models for nonlinear MIMO systems; an important aspect of this algorithm is that the identified models can guarantee closed-loop stability when used to compute the NIC control law. The second contribution is represented by efficient optimization algorithms allowing, on the basis of the identified model, the computation of the NIC control law for MIMO systems. The third contribution consists in a theoretical closed-loop stability analysis, providing sufficient conditions for a NIC controller to stabilize a MIMO nonlinear plant (this condition is used by the identification algorithm). The fourth contribution is given by extensive simulations, finalized at evaluating the numerical efficiency of the NIC approach, in view of its implementation on real-time processors. The fifth contribution is a numerical example of industrial interest, concerned with control of a robot manipulator.

The paper is organized as follows. Section 2 introduces the notation used in the paper. Section 3 proposes the prediction model identification algorithm. Section 4, after giving a general overview of the NIC approach, presents the optimization algorithms for the NIC control law computation. In Section 5, the closed-loop stability analysis is developed. In Section 6, the overall NIC design procedure is described. In Section 7, the numerical simulations for testing the efficiency of NIC control are carried out. The numerical example regarding control of a robot manipulator is presented in Section 8. Section 9 is dedicated to a comparison with other control approaches related to NIC. Section 10 concludes the main body of the paper. The proof of the main theoretical result of the paper is reported in the Appendix.

2 Notation

A column vector $x \in \mathbb{R}^{n_x \times 1}$ is denoted by $x = (x_1, \ldots, x_{n_x})$. A row vector $x \in \mathbb{R}^{1 \times n_x}$ is denoted by $x = [x_1, \ldots, x_{n_x}] = (x_1, \ldots, x_{n_x})^\top$, where $\top$ indicates the transpose.

A discrete-time signal (i.e., a sequence of vectors) is denoted with the bold style: $\boldsymbol{x} = (x_1, x_2, \ldots)$, where $x_t \in \mathbb{R}^{n_x \times 1}$ and $t = 1, 2, \ldots$ indicates the discrete time; $x_{i,t}$ is the ith component of the signal $\boldsymbol{x}$ at time t.

The ℓ_p norm of a vector $x = (x_1, \ldots, x_{n_x})$ is defined as

$$\|x\|_p \doteq \begin{cases} \left(\sum_{i=1}^{n_x} |x_i|^p\right)^{\frac{1}{p}}, & p < \infty, \\ \max_i |x_i|, & p = \infty. \end{cases}$$

The ℓ_p norm of a signal $\boldsymbol{x} = (x_1, x_2, \ldots)$ is defined as

$$\|\boldsymbol{x}\|_p \doteq \begin{cases} \left(\sum_{t=1}^{\infty} \sum_{i=1}^{n_x} |x_{i,t}|^p\right)^{\frac{1}{p}}, & p < \infty, \\ \max_{i,t} |x_{i,t}|, & p = \infty, \end{cases}$$

where $x_{i,t}$ is the ith component of the signal $\boldsymbol{x}$ at time t.

The L_p norm of a function with domain $X \subseteq \mathbb{R}^{n_x}$ and codomain in $\mathbb{R}$, is defined as

$$\|f\|_p \doteq \begin{cases} \left[\int_X \|f(x)\|_p^p \, dx\right]^{\frac{1}{p}}, & p \in (1, \infty), \\ \operatorname{ess\,sup}_{x \in X} \|f(x)\|_\infty, & p = \infty. \end{cases}$$

These norms give rise to the well-known ℓ_p and L_p Banach spaces. For simplicity, the norms without the subscript are defined as $\|\cdot\| \doteq \|\cdot\|_\infty$ for both the ℓ_∞ and L_∞ cases.

3 Polynomial prediction model

In this section, an identification algorithm is presented, allowing us to derive a prediction model for the system (1). The online inversion of this model will be the key operation of the control approach proposed in Section 4.

Consider that the system (1) can be represented in the

τ-step ahead prediction form

$$
\begin{aligned}
y_{t+\tau} &= g\left(u_t^+, q_t^-, \xi_t^v\right) \qquad (3)\\
u_t^+ &\doteq (u_{t+\tau-1}, \ldots, u_t)\\
q_t^- &\doteq \left(u_t^-, y_t^-\right)\\
\xi_t^v &\doteq (\xi_{t+\tau-1}, \ldots, \xi_{t-n})\\
g(\cdot) &\doteq h^{\tau+1}(\cdot)
\end{aligned}
$$

where h^i is obtained iterating i times equation (1) and τ is the prediction horizon.

The prediction model that we introduce is an approximation of the system (3), of the form

$$\hat{y}_{t+\tau} = f\left(u_t, q_t^-\right). \qquad (4)$$

For simplicity, this model is supposed of the same order as the system (3) but all the results presented in the paper hold also when the order is different. A systematic procedure for the choice of the order n and the prediction horizon τ will be presented in Section 6.

A parametric structure is taken for the vector-valued function f. In particular, each component $f_j \in \mathbb{R}$ of $f \in \mathbb{R}^{n_y}$ is parametrized as

$$f_j(\cdot) = \sum_{i=1}^{N} \alpha_{ij}\phi_i(\cdot) \qquad (5)$$

where $\phi_i \in \mathbb{R}$ are polynomial basis functions, $\alpha_{ij} \in \mathbb{R}$ are parameters to be identified and $j = 1, \ldots, n_y$. The parameters α_{ij} are collected in the matrix $\alpha = [\alpha_1, \ldots, \alpha_{n_y}] \in \mathbb{R}^{N\times n_y}$, with $\alpha_j \in \mathbb{R}^{N\times 1}$.

In general, the basis function choice is a crucial step, [60,28,55]. Here, the main motivations for choosing polynomial functions are two: 1) polynomials have proven to be effective approximators in a huge number of problems; 2) as we will see later, they allow a very efficient controller evaluation. Systematic indications for the choice of the polynomial functions will be given in Section 6.

The model parameters α_{ij} are identified from the data (2) by means of convex optimization: We first define

$$z_j \doteq \begin{bmatrix} \left(\tilde{y}_{j,n-L+\tau}\right)^\top \\ \vdots \\ \left(\tilde{y}_{j,0}\right)^\top \end{bmatrix}$$

$$\Phi \doteq \begin{bmatrix} \phi_1\left(\tilde{u}_{n-L}, \tilde{q}_{n-L}^-\right) & \cdots & \phi_N\left(\tilde{u}_{n-L}, \tilde{q}_{n-L}^-\right) \\ \vdots & \ddots & \vdots \\ \phi_1\left(\tilde{u}_{-\tau}, \tilde{q}_{-\tau}^-\right) & \cdots & \phi_N\left(\tilde{u}_{-\tau}, \tilde{q}_{-\tau}^-\right) \end{bmatrix}$$

where $\tilde{y}_{j,i} \in \mathbb{R}$ is the jth component of $\tilde{y}_i \in \mathbb{R}^{n_y}$ and the tilde denotes the samples obtained from the data set (2). We then introduce the set SC (stability constraints), defined as

$$
\begin{aligned}
&SC(l,\gamma,\sigma) \doteq \{\beta \in \mathbb{R}^N :\\
&\|\tilde{y}_{l,i+\tau} - \tilde{y}_{l,j+\tau} + (\mathbf{\Phi}_j - \mathbf{\Phi}_i)\beta\|\\
&\le \gamma\left\|\tilde{y}_i^- - \tilde{y}_j^-\right\| + 2\sigma,\ j \in \mathcal{T}, i \in \Upsilon_j\}
\end{aligned}
$$

where $\mathbf{\Phi}_k$ is the kth row of Φ, $\mathcal{T} \doteq \{n-L, \ldots, -\tau\}$, Υ_k is the index set

$$\Upsilon_k \doteq \left\{i : \left\|\left(\tilde{u}_k, \tilde{u}_k^-\right) - \left(\tilde{u}_i, \tilde{u}_i^-\right)\right\| \le \zeta\right\},$$

ζ is the minimum value for which every set Υ_k contains at least two elements, and l, γ and σ are the (generic) arguments of the set function $SC(\cdot)$. Note that the set SC is defined by a set of linear inequalities in β and σ, and is thus convex in β and σ. The set SC is suitably constructed to reduce below γ the Lipschitz constant of the function $\Delta \doteq g - f$, when N is sufficiently large [50]. As we will see in in Section 5, this is a fundamental requirement to enforce closed-loop stability when the identified model is used in the NIC control algorithm.

The matrix $\alpha \in \mathbb{R}^{N\times n_y}$, whose entries α_{ij} are the parameters of the model (4)-(5), is identified by means of the following convex algorithm. Note that the algorithm is "self-tuning", in the sense that most of the required parameters are automatically chosen, without involving extensive heuristic procedures (see also Section (6)). In the whole paper, any algorithm will be denoted by a recapitulatory formula such as out = algorithm_name(par), where "out" is the output of the algorithm (e.g., an estimate or an optimal solution of a problem) and "par" indicates the set of parameters/quantities needed to run the algorithm.

Identification algorithm 1 $\hat{\alpha} = \mathrm{id_poly_1}(\mathcal{D}, \hat{\gamma}_\Delta)$.
For $j = 1, \ldots, n_y$, compute the jth column α_j of α as follows:

(1) Solve the following

(a) preliminary optimization problem 1:

$$\sigma_0 = \min_{\alpha_0 \in \mathbb{R}^N} \|z_j - \mathbf{\Phi}\alpha_0\|$$

(b) preliminary optimization problem 2:

$$
\begin{aligned}
\hat{\alpha}_0 &= \arg\min_{\alpha_0 \in \mathbb{R}^N} \|\alpha_0\|_1\\
\text{s.t.} \quad &\|z_j - \mathbf{\Phi}\alpha_0\| \le \sigma_0 + \rho\|z_j\|
\end{aligned}
$$

where $\rho \cong 0$, $\rho > 0$.

(2) Solve the optimization problem

$$
\begin{aligned}
(\hat{\alpha}_j, \hat{\sigma}_\Delta) &= \arg\min_{(\alpha_j,\sigma)} \sigma\\
\text{s.t.} \quad &\text{(i)} \quad \alpha_j \in SC(j, \hat{\gamma}_\Delta, \sigma)\\
&\text{(ii)} \quad \|z_j - \Phi\alpha_j\|_p \le \sigma\Lambda\\
&\text{(iii)} \quad \|\alpha_j\|_1 \le \eta_0
\end{aligned}
$$

where $\eta_0 \doteq \|\hat{\alpha}_0\|_1$ and $\Lambda \doteq \frac{\|z_j - \mathbf{\Phi}\hat{\alpha}_0\|_p}{\|z_j - \mathbf{\Phi}\hat{\alpha}_0\|}$.

The identified model is defined by (4)-(5), with $\alpha = \hat{\alpha} = [\hat{\alpha}_1, \ldots, \hat{\alpha}_{n_y}]$.

After the preliminary operations carried out in step 1, an optimization problem is solved in step 2, providing the model parameters. This optimization problem, representing the core of the algorithm, can be explained as follows:

The constraint (i) forces the function $\Delta \doteq g - f$ to have a Lipschitz constant non larger than $\hat{\gamma}_\Delta$: a result in [50] shows that this condition can be theoretically guaranteed for a sufficiently large number of data L. On the other hand, it will be shown in Section 5 that choosing this constant smaller than 1 guarantees closed-loop stability.

In Section 5 it will be also shown that reducing the prediction error $\|z_j - \Phi\alpha_j\|_p$ allows reducing the tracking error. Clearly, there is a trade-off between stability and tracking performance: to satisfy the constraint (i) with $\hat{\gamma}_\Delta < 1$, a sufficiently large value of $\hat{\sigma}$ is required, and this may decrease the tracking error accuracy. Note that any vector norm p in (ii) can be used to reduce the prediction error. Typical choices are $p = 2$ or $p = \infty$.

Bounding the ℓ_1 norm leads to sparse vectors α_j, i.e. vectors with a certain number of zero components, [63,62,47]. Sparsity of the model coefficient vectors is important to ensure a low complexity of the model, limiting a well known issue such as the curse of dimensionality and allowing us to avoid over-fitting problems. Sparsity leads also to an efficient implementation of the model/controller on real-time processors, which may have limited memory and computation capacities. The parameter ρ in step 1 determines the trade-off between model accuracy and sparsity: larger values lead to sparser (but possibly less accurate) models.

Remark 1 *If the plant to control is characterized by unbounded trajectories, a preliminary stabilizing controller may be required to collect the data needed for model identification. Clearly, this holds for any design method, when an accurate model is not available and has to be identified from data or just validated. The preliminary controller can also be a human operator, who is able to drive the system within a bounded domain, see [16]. In any case, the preliminary controller should be tuned to obtain an "erratic" motion of the plant to control, in order to guarantee a certain level of richness of the collected data. Note that several nonlinear systems are characterized by trajectories that are unstable (in the sense of Lyapunov) but bounded (see, e.g., the Duffing oscillator in [48] and the robot manipulator in Section 8). For these kinds of system, no preliminary controllers are required.* □

4 NIC control

In this section, a novel control approach for MIMO nonlinear systems is proposed, called Nonlinear Inversion Control (NIC), relying on the efficient online inversion of the model (4).

The basic idea of this approach is the following: at each time $t > 0$, given a reference $r_{t+\tau}$ and the current regressor q_t^-, a command u_t^* is looked for, such that the model output $\hat{y}_{t+\tau}$ is close to $r_{t+\tau}$:

$$\hat{y}_{t+\tau} = f\left(u_t^*, q_t^-\right) \cong r_{t+\tau}. \tag{6}$$

Note that the latter equality may be not exact for two reasons: 1) $r_{t+\tau}$ may not be reachable; that is, no $u_t^* \in U$ may exist for which $\hat{y}_{t+\tau}$ is exactly equal to $r_{t+\tau}$; 2) values of u_t^* with a limited ℓ_2 norm may be of interest in order to have a not too high command activity. This kind of inversion is an approximate right-inversion and can be operated also when f is not injective wrt u_t^* (e.g., for some $r_{t+\tau}$ and q_t^-, more than one u_t^* may exist such that (6) holds).

The command input yielding (6) is found solving the optimization problem

$$u_t^* = \arg\min_{\mathfrak{u}\in U} J\left(\mathfrak{u}, r_{t+\tau}, q_t^-\right) \tag{7}$$

$$J\left(\mathfrak{u}, r_{t+\tau}, q_t^-\right) \doteq \left\|r_{t+\tau} - f\left(\mathfrak{u}, q_t^-\right)\right\|_2^2 + \mu\left\|\mathfrak{u}\right\|_2^2 \tag{8}$$

where $\mu \geq 0$ is a design parameter, determining the trade-off between tracking precision and command activity (see Section 6 for more indications on the choice of μ). The NIC control law is fully defined by (7).

Note that the objective function (8) is in general non-convex. Moreover, the optimization problem (7) has to be solved on-line, and this may take a long time compared to the sampling time used in the application of interest. To overcome these problems, three algorithms are proposed in the next subsections, allowing an efficient computation of the optimal command input u_t^* for the following cases: 1) SIMO (Single Input Multiple Output) system; 2) MIMO system with prediction model affine in u_t; 3) general MIMO system.

Remark 2 *If the components of* y_t *and/or* u_t *have range of variations with different scales, weighted norms can be used in (8). The weights on the outputs can be chosen as* $\|(\tilde{y}_{j,1-L}, \ldots, \tilde{y}_{j,0})\|_2^{-1}$, $j = 1, \ldots, n_y$. *The weights on the inputs can be chosen as* $\|(\tilde{u}_{j,1-L}, \ldots, \tilde{u}_{j,0})\|_2^{-1}$, $j = 1, \ldots, n_u$. □

Remark 3 *The NIC approach can be easily applied also in the case where the plant to control is affected by a measured disturbance* $v_t \in \mathbb{R}^{n_v}$, *i.e. when the plant is described by an equation of the form* $y_t = h\left(u_t^-, y_t^-, v_t^-, \xi_t^-\right)$, *where* $v_t^- \doteq (v_{t-1}, \ldots, v_{t-n})$. *The only modification required in this case is to include* v_t^- *in the vector* q_t^-. *All the algorithms, discussions and results presented in the paper will hold without significant changes.* □

Remark 4 *Any of the NIC algorithms presented in the next subsections can be implemented in combination with a linear controller, using a parallel architecture. The NIC algorithm is used to stabilize the system around the trajectories of interest, reducing at the same time the tracking error. The linear controller may allow a further reduction of the tracking error in steady-state conditions (e.g., using an integral action), [19,52].* □

4.1 SIMO system

The system to control is here supposed to have a single input: $u_t \in U \subset \mathbb{R}$**.** In this situation, for given $r_{t+\tau}$

and q_t^-, the objective function (8) is a polynomial in the scalar variable $\mathfrak{u}$. Its minima can thus be found computing the roots of its derivative, as done in the following algorithm.

Optimization algorithm 1 $u^* = K^{(1)}\left(J, r_{t+\tau}, q_t^-\right)$. *Compute the optimal input as*

$$u^* = \arg\min_{\mathfrak{u}\in U^s} J\left(\mathfrak{u}, r_{t+\tau}, q_t^-\right)$$
$$U^s \doteq \left(\text{Rroots}\left(J'\left(\mathfrak{u}, r_{t+\tau}, q_t^-\right)\right) \cap U\right) \cup \{\underline{u}, \overline{u}\}$$

where J' is the derivative of J wrt $\mathfrak{u}$, Rroots (J') *denotes the set of all real roots of J', and $\underline{u}$ and $\overline{u}$ are the boundaries of U.*

Remark 5 *The derivative J' can be computed analytically. Moreover, U^s is composed by a finite number of elements, of the order of the polynomial degree of J:*

$$\text{card}\left(U^s\right) < \deg\left(J\left(\mathfrak{u}, r_{t+\tau}, q_t^-\right)\right) + 2$$

where card *is the set cardinality and* deg *indicates the polynomial degree. The evaluation of u^* through Algorithm 1 is thus extremely efficient, since it just requires to find the real roots of a univariate polynomial whose analytical expression is known and to compute the objective function for a small number of values. See Section 7. These nice features hold thanks to the fact that polynomial functions are used. For other types of basis functions (sigmoids, Gaussians, harmonic functions, etc.), the optimization problem (7), even in the univariate case, may be non-trivial to solve.* □

4.2 MIMO system with prediction model affine in u_t

Suppose that an accurate prediction model (4) affine in u_t is found (this model can be of any degree in the other variables). Then, the objective function (8) is convex in $\mathfrak{u}$, being the sum of two terms given by the square norms of functions affine in $\mathfrak{u}$. The problem of minimizing such a convex function is standard. However, minimization has to be performed on-line and widely used algorithms such as those based on interior-point techniques may not be fast enough, [9].

Here, a novel algorithm is proposed, based on a coordinate minimization scheme where, at each iteration, the cost function (8) is minimized wrt a single component of the command input.

Optimization algorithm 2 $u^* = K^{(2)}\left(J, u0, r_{t+\tau}, q_t^-\right)$.

(1) Set $j = 1$, $\mathfrak{u}^{(0)} = u0 \in U$ (a simple choice is taking $u0$ as the center of U).

(2) For $i = 1, \cdots, n_u$, compute

$$\mathfrak{u}_i^{(j)} = K^{(1)}\left(J_i^{(j)}, r_{t+\tau}, q_t^-\right)$$

where $K^{(1)}\left(J_i^{(j)}, r_{t+\tau}, q_t^-\right)$ is Algorithm 1, minimizing wrt $\mathfrak{u}_i$ the cost function

$$J_i^{(j)} \doteq J(\mathfrak{u}_1^{(j)}, \ldots, \mathfrak{u}_{i-1}^{(j)}, \mathfrak{u}_i, \mathfrak{u}_{i+1}^{(j-1)}, \ldots, \mathfrak{u}_{n_u}^{(j-1)}, r_{t+\tau}, q_t^-).$$

(3) If $\left|J_{n_u}^{(j)} - J_{n_u}^{(j-1)}\right| < \varepsilon_J$, where ε_J is a user-defined precision parameter, stop and return the optimal input

$$u^* = \mathfrak{u}^{(j)} = \left(\mathfrak{u}_1^{(j)}, \ldots, \mathfrak{u}_{n_u}^{(j)}\right);$$

else set $j := j + 1$ and goto 2.

Remark 6 *Convergence of the coordinate minimization scheme in Algorithm 2 to a global minimum is guaranteed, [64].* □

Remark 7 *From Remark 5 and the convergence properties of the coordinate minimization scheme, [64], it follows that the evaluation of u^* through Algorithm 2 is extremely fast. See Section 7.* □

4.3 General MIMO system

In several real-world applications, the command input acts on the plant to control in a nonlinear way, resulting in a function g highly nonlinear in the input. In these situations, controlling the system can be significantly more difficult, since this function may be non-invertible (more precisely, non-injective) wrt u_t. The present approach allows us to efficiently deal with these cases. The following algorithm is proposed, based on the application of Optimization algorithm 2 for several different starting points.

Optimization algorithm 3 $u^* = K^{(3)}\left(J, r_{t+\tau}, q_t^-\right)$.

(1) Set $j = 1$, $u0 = u_{t-1}$.

(2) Compute

$$\mathfrak{u}^{(j)} = K^{(2)}\left(J, u0, r_{t+\tau}, q_t^-\right)$$

where $K^{(2)}\left(J, u0, r_{t+\tau}, q_t^-\right)$ is Algorithm 2.

(3) If $j = N_{iter}$, where N_{iter} is the user-defined maximum number of iterations, stop and return the locally optimal input

$$u^* = \mathfrak{u}^{(j)};$$

else set $j := j + 1$ and goto 2, changing the starting point $u0$.

The choice of $u0$ for $j > 1$ can be made using a space-filling curve, such as the (multi-dimensional) Peano Curve, Hilbert curve, H-tree fractal, or a suitably randomized curve/set of points, [4]. These kinds of curves/sets of points allow for a systematic exploration of a given domain of interest. Of course, they cannot represent a solution to the fundamental problem of the curse of dimensionality, [6], but they may lead to better solutions wrt less systematic methods.

The maximum number of iterations N_{iter} in step 3 can be chosen on the basis of the sampling time used in the application of interest: N_{iter} should be chosen not

larger (with a certain margin) than the sampling time divided by the time taken to compute $\mathfrak{u}^{(j)}$. This time can be estimated through simulations and/or hardware-in-the-loop experiments (these latter typically give quite precise indications).

To reduce the computation time, a stopping criterion that can be used in addition to the previous one (based on N_{iter}) is the following: stop when

$$\begin{aligned} &J\left(\mathfrak{u}^{(j)}, r_{t+\tau}, q_t^-\right) < \max\left(\varepsilon_1, \varepsilon_2 R_t^c\right) \\ &R_t^c \doteq \tfrac{1}{\tau+1}\left\|r_{t+\tau} - y_{t-1}\right\|_2^2 \end{aligned} \tag{9}$$

where ε_1 and ε_2 are user-defined precision parameters (absolute the former, relative the latter). The quantity R_t^c, called the *reachability indicator*, measures how far is the reference from the current output. The stopping criterion thus works as follows: If the reference is reachable, then the algorithm stops when a value close to zero of the objective function is found (i.e., a value smaller than ε_1 or $\varepsilon_2 R_t^c$). If the reference is not reachable, then the algorithm is allowed to stop also when the objective function is not close to zero.

Remark 8 *As already observed, the objective function (8) is in general non-convex. While in the scalar and affine cases we are able to find a global solution, here we find a solution that is not guaranteed to be global. However, we will show in Section 7 by means of extensive simulations that Algorithm 3 is able to find solutions close to a global one, in quite short times.* □

Remark 9 *To solve polynomial optimization problems, eigenvalue techniques, [15,14], sum of squares, [57,13,33,20,37], or Gröbner bases, [56], could be used. However, these approaches appear at present not suitable for fast on-line applications, since they may require large computation times and/or large memory occupations.* □

Remark 10 *In Algorithms 2 and 3, all inputs are used together in an optimal way to control all outputs. This is analogous to what happens in standard MIMO NMPC.* □

Remark 11 *Several control methods (either first-principle-based or data-driven) require assumptions like invertibility wrt the input, input decoupling or weak input decoupling. Feedback linearization is a typical example, since it assumes that the system to control is affine in the input and, in the MIMO case, that the function multiplying the input is invertible (allowing full input decoupling). Another example is given by the method proposed in [25], where a weak decoupling condition is required. However, this is not the case of NMPC-like methods (including NIC), where the idea of optimizing on-line a suitable objective function allows one to avoid assumptions such as invertibility, decoupling or weak decoupling. The on-line optimization is able to find, at each time instant, an optimal combination of all the command inputs, in order to attain the desired control goal.* □

5 Closed-loop stability analysis

In this section, we study the behavior of the closed-loop system formed by the feedback connection of the plant (1) and the controller (7). This system is defined by:

$$\begin{aligned} y_t &= h\left(u_t^-, y_t^-, \xi_t^-\right) \\ u_t^- &= \left(u_{t-1}^*, \ldots, u_{t-n}^*\right) \end{aligned} \tag{10}$$

where $t = 1-\tau, 2-\tau, \ldots$, and, for $t \le 0$, $r_t = \tilde{y}_t$ and $u_t^* = \tilde{u}_t$. The plant initial condition $y_0^- \in \mathcal{Y}^0 \subseteq R^n$ is assumed, where $R \subset \mathbb{R}^{n_y}$ is a compact set with non-empty interior. In our formulation, the set R is the "set of interest", that is, the set where the trajectories of the closed-loop system (10) are required to occur.

The block diagram of the closed-loop system is shown in Figure 1, where "plant" is the system (1) (or the system defined by the first equation in (10)) and K^* represents the controller (7), implemented through any of the optimization algorithms of Section 4, providing the command input u_t^*.

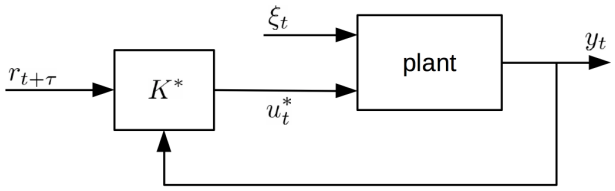


Figure 1. Closed-loop system.

A basic assumption for our closed-loop analysis is that the function h is Lipschitz continuous on the compact set $U^n \times Y^n \times \Xi^n$. This assumption is reasonable: a large number of real-world systems are described by functions that are Lipschitz continuous on a compact set. Without loss of generality, we also assume that $U^n \times Y^n \times \Xi^n$ contains the origin.

To study the behavior of this closed-loop system, the following stability notion is introduced.

Definition 1 *A nonlinear system (possibly time-varying) with input r_t, output y_t, and noise ξ_t is finite-gain ℓ_∞ stable on $\left(\mathcal{Y}^0, \mathcal{R}, \Xi\right)$ if finite non-negative constants γ_{yr}, $\gamma_{y\xi}$ and λ_y exist such that*

$$\|\boldsymbol{y}\| \le \gamma_{yr}\|\boldsymbol{r}\| + \gamma_{y\xi}\|\boldsymbol{\xi}\| + \lambda_y$$

for any $(y_0^-, \boldsymbol{r}, \boldsymbol{\xi}) \in \mathcal{Y}^0 \times \mathcal{R} \times \Xi$, where $\boldsymbol{r} = (r_1, r_2, \ldots)$, $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots)$ and $\boldsymbol{y} = (y_1, y_2, \ldots)$. □

Note that, for $\mathcal{R} = \ell_\infty$ and $\Xi = \ell_\infty$, this stability definition coincides with the standard finite-gain stability definition given, e.g., in [31].

In the following, some key quantities for our study are introduced; a stability/accuracy result is then presented and discussed. Let us define the functions

$$\begin{aligned} g^o\left(u_t^v, y_t^-\right) &\doteq g\left(u_t^v, y_t^-, 0\right) \\ \Delta\left(u_t^v, y_t^-\right) &\doteq g^o\left(u_t^v, y_t^-\right) - f\left(u_t^p, y_t^-\right) \\ F\left(r_{t+\tau}, u_t^p, y_t^-\right) &\doteq r_{t+\tau} - f\left(u_t^p, y_t^-\right) \end{aligned} \tag{11}$$

where $u_t^v \doteq (u_t^+, u_t^-)$, $u_t^p \doteq (u_t, u_t^-)$, g is the function of the system in the prediction form (3) and f is the function of the prediction model (4).

From Lipschitz continuity of h (by assumption) and f (by definition), it follows that Δ and F are Lipschitz continuous on $\Omega_\Delta \doteq U^{n+\tau} \times Y^n$ and $\Omega_F \doteq U^{n+1} \times Y^{n+1}$, respectively. Thus, constants γ_Δ and γ_F exist such that

$$\begin{aligned}\gamma_\Delta < \infty : \ \|\Delta(u,y) - \Delta(u,y')\| \le \gamma_\Delta \|y - y'\| \\ \forall y, y' \in Y^n, \ \forall u \in U^{n+\tau}, \\ \gamma_F < \infty : \ \|F(r,u,y) - F(r,u,y')\| \le \gamma_F \|y - y'\| \\ \forall y, y' \in Y^n, \ \forall u \in U^{n+1}, \forall r \in Y.\end{aligned}$$

Lipschitz continuity of Δ and F in turn implies that constants σ_Δ and σ_F exist, such that

$$\begin{aligned}\left\|\Delta\left(u_t^v, r_t^-\right)\right\| \le \sigma_\Delta < \infty \\ \left\|F\left(r_{t+\tau}, u_t^p, r_t^-\right)\right\| \le \sigma_F < \infty\end{aligned} \tag{12}$$

for any $\boldsymbol{u} = (u_1, u_2, \ldots) \in U^\infty$ and any $\boldsymbol{r} = (r_1, r_2, \ldots) \in \mathcal{R}$.

The following result provides sufficient conditions for finite-gain stability of the closed-loop system (10) and a guaranteed bound on the tracking error.

Theorem 1 *Assume that*

$$\gamma_c \doteq \gamma_\Delta + \gamma_F < 1 \tag{13}$$

$$Y \supseteq R \oplus E \tag{14}$$

where $\oplus$ indicates the set sum,

$$\begin{aligned}E &\doteq \{\eta \in \mathbb{R}^{n_y} : \|\eta\| \le \bar{e}\} \\ \bar{e} &\doteq \tfrac{1}{1-\gamma_c}\left(\sigma_\Delta + \sigma_F + \gamma_\xi \bar{\xi}\right)\end{aligned}$$

and γ_ξ is the Lipschitz constant of g wrt ξ_t^v. Then:

(i) The closed-loop system (10) is finite-gain ℓ_∞ stable on $\left(\mathcal{Y}^0, \mathcal{R}, \Xi\right)$.

(ii) The tracking error signal $\boldsymbol{e} = (e_1, e_2, \ldots)$, with $e_t \doteq r_t - y_t$, is bounded as

$$\|\boldsymbol{e}\| \le \bar{e} \tag{15}$$

for any $\left(y_0^-, \boldsymbol{r}, \boldsymbol{\xi}\right) \in \mathcal{Y}^0 \times \mathcal{R} \times \Xi$.

Proof. See the appendix. □

This theorem shows that two key conditions are sufficient to guarantee closed-loop stability. The first one, i.e., inequality (13), is satisfied if:

(i) $\gamma_\Delta < 1$. This requires the model f to be an accurate approximation of the true function g^o. In particular, f must be accurate in describing the dependence of g^o wrt y_t^- (being γ_Δ the Lipschitz constants wrt y_t^- of Δ). An important point is that the proposed identification algorithm 1 provides models satisfying this condition when the number of data is sufficiently large, see the discussion below the algorithm.

(ii) $\gamma_F < 1 - \gamma_\Delta$. This is not particularly restrictive in general: It is certainly satisfied if $\mu = 0$ and $r_{t+\tau}$ is reachable (i.e., in the range of $f\left(\cdot, q_t^-\right)$) for all t. Indeed, in this case, solving problem (7) makes an exact model inversion. That is, $\hat{y}_{t+\tau} = r_{t+\tau}$ for all t, and thus $\gamma_F = 0$. If μ is chosen sufficiently small and $r_{t+\tau}$ is sufficiently close to a reachable value, supposing that γ_F satisfies the assumption is reasonable. On the contrary, if these requirements are not met, condition (13) may be not satisfied, leading to an unstable behavior and possibly to a diverging tracking error. Clearly, if non-reachable trajectories have to be tracked, the only solution is to physically change u_t and/or U, in order to increase the control authority. This means that the actuators of the plant to control have to be changed. The constant γ_F can be estimated from the available data by means of the validation procedure in [39], thus allowing the verification of (13).

The second stability condition, i.e., the set inclusion (14), essentially requires the set explored by the data (i.e., Y) to be sufficiently large to contain the set where the trajectories of interest are defined (i.e., R), plus a border region bounded by $\bar{e}$.

The theorem also provides a tight bound (in a worst-case sense) on the tracking error, and this bound is exactly $\bar{e}$. The control goal is clearly to reduce $\bar{e}$, and this happens if:

(i) The model f is an accurate approximation of g^o. Note that the identification algorithm 1, besides enforcing the inequality $\gamma_\Delta < 1$ to hold, aims also at minimizing σ_Δ ($\hat{\sigma}_\Delta$ is an estimate of σ_Δ), which is bound on the error $g_o - f$.

(ii) The reference signals are properly chosen, so that σ_F is sufficiently small. Obviously, the reference signals should be consistent with the physical properties of the plant. For instance, in a mechanical system, the states are typically the positions and the velocities. The position references can be generated as sequences of values ranging in the position physical domains with reasonable variations. The velocity references can be generated as the derivatives of the position references. More in general, under stability conditions, if the reference signals are not characterized by abrupt variations, then $r_{t+\tau}$ can be reached from y_t^-, yielding values of $F\left(r_{t+\tau}, u_t^p, r_t\right)$ and σ_F close to zero. A reference generator will be proposed below, allowing us to make σ_F as small as desired.

(iii) The bound $\gamma_\xi \bar{\xi}$, accounting for the effect of the noise on the system output, is sufficiently small. This term is a worst-case upper bound and typically is tight for high frequency noises (this happens in general, for both linear and nonlinear control systems). As a matter of fact, it has been observed in several numerical experiments that the proposed control scheme is able to significantly reduce the effect of low frequency noises (see, e.g., the example in Section 8).

The constants $\gamma_F, \sigma_F, \sigma_\Delta$ and $\gamma_\xi \bar{\xi}$ can be estimated from the data (using, e.g., the algorithms in [39] and [41]), allowing us to obtain an estimate of $\bar{e}$. It is then possible to (approximately) verify assumption (13) and, when Y and R are simple sets (e.g., rectangles or ellipsoids), also assumption (14). If the estimate of $\bar{e}$ indicates that

(14) is not satisfied, it is necessary to collect more data to enlarge the set Y. Clearly, we can only estimate these constants, and thus we cannot be totally sure that the assumptions are satisfied. Note however that this is a problem common to any approach (data-driven or first-principle-based): in most real applications, we have only an approximate knowledge of the plant to control and thus we can only be confident - but never sure - that a controller will work when applied to the real plant.

In summary, the model f should be accurate in describing the variability wrt y_t^- (to guarantee stability) and, at the same time, in predicting the system output (to reduce σ_Δ). The identification algorithm 1 has been developed in this view, allowing us to derive models fulfilling such requirements.

On the basis of these considerations, we can conclude that assumptions (13) and (14) are reasonable, since they require that a sufficiently accurate model has been identified and that the reference values are reachable (or approximately reachable). These kinds of requirements are common to most control approaches for nonlinear systems.

In ideal conditions (i.e., no disturbance, no modeling error and perfect inversion for all t) the constants $\gamma_\xi\bar{\xi}$, σ_Δ and σ_F are null, implying that the tracking error is null. If only the disturbance is null, the bound $\bar{e}$ is smaller wrt the general case, since the term $\gamma_\xi\bar{\xi}$ is null. If the inversion is not exact (this happens, e.g., when $r_{t+\tau}$ is not reachable) but assumptions (13) and (14) hold, our algorithms in any case guarantee a bounded tracking error. If the the plant to control has an unstable zero dynamics and U is a bounded set, the condition of perfect inversion in general cannot be met, resulting in a non-zero constant σ_F in Theorem 1. In this case, the tracking error cannot be made arbitrarily small for arbitrary reference signals. In general, in the presence of unstable zero dynamics, it is not possible to have at the same time a bounded input and an arbitrarily small tracking error for arbitrary reference inputs, whatever is the control method used, [8]. This is a structural limitation of the plant, independent of the control method (of course, our method is also affected by this limitation).

Note that we are not interested in an asymptotic convergence result. We derive a bound on the tracking error that holds at each time t. In ideal conditions (no disturbances, no model errors, perfect inversion), this bound is null, demonstrating that, in such ideal conditions, our approach is able to provide the best possible performance.

Another important aspect is reference generation: a simple and effective indication is to use reference signals with reasonable variations (e.g., using a pass-band filter). A more systematic reference generation algorithm is now proposed, allowing us to reduce the constant σ_F, making it possibly null. The reference generator is the following:

$$\begin{aligned} \breve{u}_t &= K^* \left(J, \breve{r}_{t+\tau}, (\breve{u}_t^-, \breve{r}_t^-)\right) \\ r_{t+\tau} &= f\left(\breve{u}_t^p, \breve{r}_t^-\right) \end{aligned} \tag{16}$$

where f is the model (4), K^* is one of the control algorithms presented in Section 4, $\breve{u}_t^p = (\breve{u}_t, \breve{u}_t^-)$, $\breve{r}_t$ is a "parent" reference and $\breve{r}_t^- \doteq (\breve{r}_{t-1}, \ldots, \breve{r}_{t-n})$. If $\mu = 0$ and $\breve{r}_{t+\tau}$ is reachable (that is, a $\breve{u}_t \in U$ exists such that $\breve{r}_{t+\tau} = f(\breve{u}_t^p, \breve{r}_t)$), then $r_{t+\tau} = \breve{r}_{t+\tau}$, see (7) and (8). If it is not reachable, $r_{t+\tau}$ is the best reachable approximation of $\breve{r}_{t+\tau}$. Generating the reference according to (16) with $\mu = 0$, we have $F(r_{t+\tau}, u_t^p, r_t) = 0$, implying that $\sigma_F = 0$. When $\mu > 0$, $F(r_{t+\tau}, u_t^p, r_t) \simeq 0$, implying that $\sigma_F \simeq 0$. In practice, the reference generator (16) allows us to verify when a reference value is reachable and, when it is not, to choose a suitable reachable approximation of it.

Remark 12 *The stability analysis of this section has been developed within a nonlinear Set Membership (SM) framework, [39,42,40]. In [42], a unified SM theory for identification, prediction and filtering of nonlinear systems is shown. The present work can be seen as a natural extension of this theory to control.* □

6 NIC design procedure

In this section, the overall NIC design procedure is summarized, consisting of two main steps: model identification and controller design.

6.1 Prediction model identification

Model identification can be performed as follows:

1. Choose a $\hat{\gamma}_\Delta < 1$. As discussed above, taking $\hat{\gamma}_\Delta < 1$ is a proxy for closed-loop stability. On the other hand, too small values of $\hat{\gamma}_\Delta$ may lead to large tracking errors. Note that the choice of the parameter ρ in the Identification Algorithm 1 is not critical and can be carried out by means of a simple trial and error procedure.

2. Choose a maximum order n_{max} and a maximum prediction horizon τ_{max}.

3. For $n = 1, \ldots, n_{max}$ and for $\tau = 1, \ldots, \tau_{max}$:

3a. Apply the identification algorithm 1 to the data set (2):

$$\hat{\alpha}(n, \tau) = \text{id_poly_1}\left(\mathcal{D}, \hat{\gamma}_\Delta\right).$$

3b. Compute the following normalized prediction error:

$$NPE(n, \tau) \doteq \frac{1}{\tau n_y} \sum_{j=1}^{n_y} \left\| z_j - \mathbf{\Phi}\hat{\alpha}_j(n, \tau) \right\|_2. \tag{17}$$

4. Choose the order and prediction horizon according to

$$(n^*, \tau^*) = \arg\min_{(n,\tau)} \left(NPE(n, \tau) + \varsigma n\right) \tag{18}$$

where ς is used to penalize models with high order. An effective choice can be $\varsigma = 0.01 \max_{(n,\tau)} NPE(n, \tau)$.

5. The selected model is the one defined by the parameter matrix $\hat{\alpha}(n^*, \tau^*)$. □

Remark 13 *If the columns of z have range of variations with different scales, a weighted mean can be used in (17). The weights can be chosen as $\|(\tilde{y}_{j,1-L}, \ldots, \tilde{y}_{j,0})\|_2^{-1}$, $j = 1, \ldots, n_y$.* □

In this procedure, the model order and prediction horizon are chosen to reduce the normalized prediction error $NPE(n,\tau)$. This index was conceived to obtain an optimal trade-off between two contrasting criteria: on one hand, reducing the prediction error allows reducing the tracking error (see Theorem 1); on the other hand, increasing the prediction horizons in general yields an increase of the robustness and stability properties of the closed-loop system (this typically happens also in NMPC, see, e.g., Chapter 1 in [17]). The term ςn was included to penalize high order models. An example of application of this procedure is shown in Figure 4.

The above model identification procedure can be repeated for different polynomial degrees, starting from low ones (e.g., 1) and moving to higher ones, until no significant reductions of $NPE(n,\tau)$ are observed.

An effective choice for the basis function set to use in the model parametrization (5) is the following:

$$\begin{aligned}\{\phi_i(x), i=1,\dots,N\} = \{&x_{i_1}^{j_1}x_{i_2}^{j_2};\\ &i_1=1,\dots,n_x, i_2=i_1,\dots,n_x;\\ &j_1,j_2=0,\dots,j_p\} \doteq BF\end{aligned} \tag{19}$$

where $x=(x_1,\dots,x_{n_x})\in\mathbb{R}^{n_x}$, $n_x=n_u+n(n_u+n_y)$, $2j_p=d_p$ is the polynomial degree and $N=(1+j_pn_x)(2+j_pn_x)/2$. To identify a model affine in u_t, the basis function set can be chosen as follows:

$$\begin{aligned}\{\phi_i(x), i=1,\dots,N\} = \{&x_{i_1}^{j_1}x_{i_2}^{j_2}\in BF:\\ &j_1\le 1 \text{ for } i_1\le n_u, j_2\le 1 \text{ for } i_2\le n_u\\ &j_1+j_2\le 1 \text{ for } i_1,i_2\le n_u\} \doteq BFA\end{aligned} \tag{20}$$

where $N=n_2(1+2n_1-n_2)/2$, $n_1=1+n_x+(j_p-1)(n_x-n_u)$, $n_2=1+j_p(n_x-n_u)$.
In both cases, the independent variables x_i can be scaled to range in the interval $[-1,1]$, in order to avoid bad numerical conditioning. This is a simple and effective basis function generation technique, not affected by a combinatorial explosion of the terms in the series (5), yielding at the same time a sufficiently large and flexible set of functions allowing an accurate approximation.

6.2 Controller design

Once the prediction model has been identified, control design just consists in choosing the parameter $\mu\ge 0$ in (8). This parameter determines the trade-off between tracking precision and command activity, and its choice is not critical: If u is not subject to relevant limitations, we can just set $\mu=0$. Otherwise, the value of μ should be progressively increased (starting from 0), until the desired trade-off between control performance and command activity is obtained. Given all the chosen parameters, the controller is fully defined by (7), and implemented through one of the optimization algorithms of Section 4.

7 Optimization performance evaluation

This section describes some extensive simulation studies that were carried out to test the numerical efficiency of the optimization algorithms proposed in Section 4.

The following optimization problem was considered:

$$\begin{aligned}&u^* = \arg\min_{\mathfrak{u}\in U} J(\mathfrak{u},r,q)\\ &J(\mathfrak{u},r,q) \doteq \|r-f(\mathfrak{u},q)\|_2^2+\mu\|\mathfrak{u}\|_2^2\end{aligned} \tag{21}$$

where $f(\cdot)$ is a polynomial function of degree d_p and $\mathfrak{u}\in U\subset\mathbb{R}^m$, $r,q\in\mathbb{R}^m$. This problem is analogous to (7) but the dependence on time is not relevant. The value $\mu=0$ was taken since, with this value, if r is in the range of $f(\cdot,q)$, we know the global minimum of $J(\mathfrak{u},r,q)$ to be 0.
Values of m in the set $\{1,2,4,6,8\}$ and values of d_p in the set $\{1,2,4,6\}$ were considered, corresponding to MIMO systems with up to 8 command inputs and models with polynomial degree up to 6. Note that in all the applications tackled up to now with the present approach, degrees from 2 to 4 led to a satisfactory prediction and control performance (see [19,52,38,48] and the numerical example presented in Section 8).

7.1 Monte Carlo simulation 1

For each combination of m and d_p in these sets, a Monte Carlo simulation was carried out, consisting of 50 main trials, each consisting of 100 subtrials (total number of trials: $5\times4\times50\times100=100\,000$).
In each main trial, $f(\cdot)$ was defined as a polynomial function of degree d_p of the form (4)-(5), with sparse vectors α_j. In particular, a number n_s of nonzero coefficients was assumed, with n_s ranging in the interval $[0,500]$ in function of m and d_p (the nonzero coefficients were chosen according to a Gaussian distribution with zero mean and unitary variance). The maximum number of non-zero coefficients (500) was chosen on the basis of our experience: in the numerous simulated and real applications that we dealt with in the past, the number of non-zero coefficients was always of the order of some hundreds.
In each subtrial, a sequence $r_i=f(u_i^{true},q_i)$ was generated, where q_i and u_i^{true} are vectors with random entries (chosen according to a uniform distribution with support $[-1,1]$), and $i=1,\dots,100$. In this way, r_i is guaranteed to be in the range of $f(\cdot)$. Then, for each i, the optimization problem (21) was solved. Note that the decision variable $\mathfrak{u}$ can be in general different from the "true" input u_i^{true}.
For $m=1$, the optimization algorithm 1 was used to solve the problem. For $m>1$ and $d_p=1$, the optimization algorithm (2) was used, with $\varepsilon_J=1\mathrm{e}-6$. Otherwise, the optimization algorithm (3) was used, with $u0$ chosen randomly, assuming $N_{iter}=100$ and adopting the stopping criterion (9), with $\varepsilon_1=0.05$ and $\varepsilon_2=0$. Since the values of $J(\mathfrak{u},r,q)$ are in average around 1, this choice of ε_1 corresponds to accepting a worst-case degradation of 5% wrt a global minimum.
For each combination of the dimension m and the polynomial degree d_p, the following indexes were used to evaluate the algorithm performance:

- $E_2 \doteq \frac{1}{5000}\sum_{i=1}^{5000}\left(J(u_i^*,r_i,q_i)-J(u_i^{true},r_i,q_i)\right)$, where u_i^* is the solution of the optimization problem (21), computed for each random sample, and 5000 is obtained as the number of main trials (50) times the

number of subtrials (100). Note that, in the present case, we know that $J\left(u_i^{true}, r_i, q_i\right) = 0$.

- $E_\infty \doteq \max_{i=1,\ldots,5000} \left(J\left(u_i^*, r_i, q_i\right) - J\left(u_i^{true}, r_i, q_i\right)\right)$.
- $T_{sc} \doteq$ average time taken by a Matlab .m function to solve a single optimization problem on a laptop with an i7 3Ghz processor and 16 MB RAM. The average was computed over the 5000 samples of the Monte Carlo simulation.
- $T_m \doteq$ average time taken by a compiled Simulink mex function to solve a single optimization problem on the same laptop. This function was generated in 10 of the 50 main trials, since this operation is relatively complex. The average was thus computed over $10 \times 100 = 1000$ samples of the Monte Carlo simulation.

The obtained results, shown in Table 1, can be commented as follows.

- All the three proposed optimization algorithms are able to find precise solutions (i.e., giving values of the objective function close to zero) in short times for all the considered input dimensions and polynomial degrees.
- The values of E_2 and E_∞ obtained by the optimization algorithms 1 and 2 confirm the above theoretical argumentations, by which these algorithms are guaranteed to find always a global minimum of the objective function. Note that algorithm 2 is suited for models that are affine in $\mathfrak{u}$ but can be of any degree in the other variables.
- Using compiled mex functions allows a significant reduction of the computation times only for problems involving polynomials with a not too high degree in $\mathfrak{u}$. A possible interpretation is that the Simulink automatic compiler looses efficiency for large degree polynomials.

For a fair comparison, we made a second evaluation of T_{sc}, considering only the 10 main trials used to evaluate T_m. The obtained values of T_{sc} (averaged over $10 \times 100 = 1000$ samples of the Monte Carlo simulation) are very similar to those reported in the 7^{th} column of Table 1 (averaged over all the 5000 samples). In particular, no differences wrt the values appearing in this column were observed, larger than 12%. For this reason, the values of T_{sc} averaged in the 10 main trials are not reported in the Table.

7.2 Monte Carlo simulation 2

The results obtained by the optimization algorithm 3 show that, accepting a small degradation of the minimization performance wrt the global minimum ($\leq 5\%$ in the above Monte Carlo simulations) the optimization problems are solved in quite short times.

In order to better investigate this aspect, other Monte Carlo simulations were carried out: The input dimension $m = 4$ and the polynomial degree $d_p = 4$ were assumed. The values $N_{iter} = 100$, $\varepsilon_1 = 0.01, 0.02, \ldots, 0.1$, and $\varepsilon_2 = 0$ were considered for the optimization algorithm 3. For each value of ε_1, a Monte Carlo simulation was performed, consisting of 50 main trials, each consisting of 100 subtrials (total number of trials: $10 \times 50 \times 100 = 50\,000$). The main and sub trials were analogous to the ones described above. The values of T_{sc}, E_2 and E_∞ obtained in these simulations are plotted in Figure 2 in function of ε_1. It can be observed that E_2 and E_∞ increase linearly with ε_1, while T_{sc} steeply decreases for small values of ε_1 and then remains almost constant. It can be concluded that the "optimal" value of ε_1 is 0.02, providing the best trade-off between speed and accuracy. Note that these considerations are not restricted to this numerical example but are general: ε_1 (together with ε_2, if necessary) is a parameter allowing us to efficiently determine the trade-off between speed and accuracy of our algorithms, whatever is the application of interest.

m	d_p	n_s	alg.	E_2	E_∞	T_{sc} [s]	T_m [s]
1	1	3	1	1.2e-14	3.2e-14	2.7e-4	<1.0e-4
	2	6	1	1.9e-13	1.9e-12	3.0e-4	<1.0e-4
	4	15	1	2.1e-13	1.4e-12	3.4e-4	<1.0e-4
	6	28	1	1.1e-13	6.5e-13	3.7e-4	<1.0e-4
2	1	5	2	4.3e-12	1.6e-11	8.7e-4	<1.0e-4
	2	15	3	5.0e-3	0.048	1.6e-3	1.4e-4
	4	45	3	4.1e-3	0.022	2.2e-3	5.6e-4
	6	81	3	5.2e-3	0.034	5.4e-3	$>T_{sc}$
4	1	9	2	7.5e-5	4.2e-4	7.8e-4	<1.0e-4
	2	45	3	8.2e-3	0.039	3.1e-3	4.5e-4
	4	116	3	0.013	0.047	0.038	$>T_{sc}$
	6	197	3	0.014	0.046	0.17	$>T_{sc}$
6	1	13	2	4.3e-4	9.7e-4	1.9e-3	<1.0e-4
	2	81	3	0.013	0.048	0.011	1.2e-3
	4	197	3	0.016	0.048	0.21	$>T_{sc}$
	6	339	3	0.021	0.049	0.76	$>T_{sc}$
8	1	17	2	5.0e-4	8.6e-4	2.4e-3	<1.0e-4
	2	116	3	0.019	0.048	0.10	3.1e-3
	4	289	3	0.027	0.049	1.6	$>T_{sc}$
	6	500	3	0.032	0.049	8.3	$>T_{sc}$

Table 1
Monte Carlo simulation 1 results.

We believe that the simulation study of this Section is important since it shows that, according to the results in Table 1, the NIC algorithms can actually be implemented on real-time processors in industrial applications.

8 Numerical example: Control of a 2-DOF robot manipulator

8.1 2-DOF robot manipulator

The 2-DOF (2-degrees of freedom) robot manipulator depicted in Figure 3 has been considered, where z_1 and z_2 are the angular positions of the two segments of the robot arm, u_1 and u_2 are the control torques acting on these segments, l_1 and l_2 are the segment lengths, and M_1 and M_2 are the segment masses. The parameter values $l_1 = 0.8$ m, $l_2 = 0.7$ m, $M_1 = 2.5$ Kg, $M_2 = 2$ Kg have been assumed.

This robot manipulator is a MIMO system (with 2 inputs and 2 outputs), described by the following

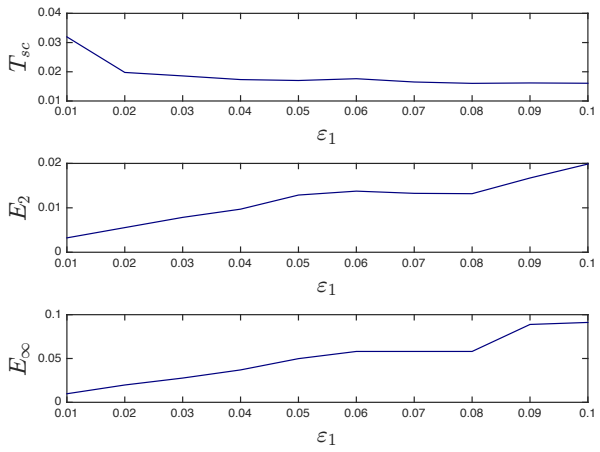


Figure 2. Trade-off between computation time and optimization accuracy.

continuous-time state-space nonlinear equations:

$$\begin{aligned} \dot{z}(\tau) &= A^c(z(\tau))z(\tau) + B^c(z(\tau))u(\tau) \\ y(\tau) &= \begin{bmatrix} z_1(\tau) \\ z_2(\tau) \end{bmatrix} \end{aligned} \tag{22}$$

where τ is the continuous time, $z(\tau) = [z_1(\tau)\ z_2(\tau)\ \dot{z}_1(\tau)\ \dot{z}_2]^\top$ is the state and $u(\tau) = [u_1(\tau)\ u_2(\tau)]^\top$ is the input. The expressions of $A^c(z(\tau)) \in \mathbb{R}^{4\times4}$ and $B^c(z(\tau)) \in \mathbb{R}^{4\times2}$ can be found in [32].

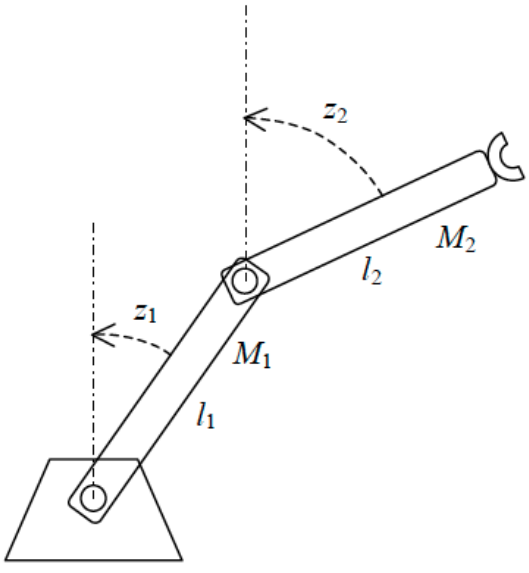


Figure 3. Robot Manipulator.

8.2 Data generation

A set of $L = 5000$ data was generated by simulation of (22):

$$\mathcal{D} \doteq \{\tilde{y}_t, \tilde{u}_t, \}_{k=-4999}^{0}.$$

The data were collected with a sampling time $T_s = 0.02$ s, using the following input signals:

$$u_j(\tau) = \begin{cases} -20z_j(\tau), \text{ if } |z_j(\tau)| \geq 1.75 \text{ rad} \\ 0, \quad \text{if } l < \tau \leq l + 500,\ l = 500, 1500, 2500, 3500, \\ \qquad \text{and } |z_j(\tau)| < 1.75 \\ A_u \sin(\omega_{j1}\tau) + A_u \sin(\omega_{j2}\tau), \quad \text{otherwise,} \end{cases} \tag{23}$$

where $j = 1, 2$, $A_u = \text{rand}[50, 150]$ Nm, $\omega_{11} = \text{rand}[0.05, 0.09]$ rad/s, $\omega_{12} = \text{rand}[0.5, 0.11]$ rad/s $\omega_{21} = \text{rand}[0.04, 0.1]$ rad/s $\omega_{11} = \text{rand}[0.7, 1.2]$ rad/s. The notation $A_u = \text{rand}[50, 150]$ means that A_u is a number, randomly chosen according to a uniform distribution in the interval $[50, 150]$. The feedback input on the first line of (23) was applied in order to limit the working range of z_1 and z_2 to the interval $[-\pi, \pi]$ rad (the gain -20 and the threshold 1.75 rad were chosen thorough preliminary simulations). Measurement noises were added to y_j, $j = 1, 2$, simulated as uniform noises with amplitude 0.02 rad, corresponding to a signal-to-noise amplitude ratio $SNR \cong 100$. Note that this noise level is reasonable for advanced industrial applications, where very accurate angular position sensors are available.

8.3 Controller design

From the data described above, two NIC controllers were designed following the procedure of Section 6: The first one, called NIC1, is based on a general nonlinear model. The second one, called NIC2, is based on a model affine in u (note however that the fact that the system to control is affine in u does not imply that the corresponding prediction form is affine; in this example the prediction form is not affine). These controllers are described in Table 2. For both controllers, the model basis functions were generated according to (19) (or (20) in the affine case), where the independent variables were scaled to range in the interval $[-1, 1]$. The data with index $t = 1 - L, \ldots, -20, -10, 0$ were used to form the constraints (i) in the identification algorithm 1 (we did not use all the data in order to reduce the memory occupation). The level curves of the function $NPE(n, \tau) + \varsigma n$ used in (18) to optimally select the model order and prediction horizon for the controller NIC1 are shown in Figure 4. For the NIC1 controller, N_{iter} was chosen equal to 50 since we observed that the algorithm converges after about 30, maximum 40, iterations. We did not use the stopping criterion (9) and thus $\varepsilon_1, \varepsilon_2$ were not required.

Looking at Table 2, the following observations can be made:

- The sparsification properties of the algorithm do not give significant reductions of the number of basis functions. The reason is that the total number of basis functions is already relatively low. Sparsification is more relevant when the regressor is of higher dimension and the polynomial degree is larger, leading to problems with many hundreds or thousands basis functions. In any case, bounding the ℓ_1 norm of the model coefficient vectors is useful for regularization reasons.
- Algorithm (1) forces the model to (approximately)

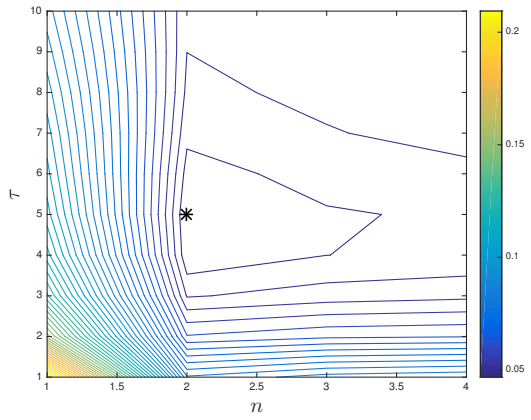


Figure 4. Level curves of the function $NPE(n,\tau)+\varsigma n$. The optimal point is denoted by *.

satisfy the condition $\hat{\gamma}_\Delta < 1$. Supposing that $\gamma_\Delta \cong \hat{\gamma}_\Delta$, the stability condition $\gamma_F + \gamma_\Delta < 1$ required by Theorem 1 is satisfied for both the NIC1 and NIC2 controllers.

- The computational times for the control design phase (referred to a laptop with 3 Ghz core i7 processor, 16 GB RAM and the CVX tool [50]) resulted quite low, considering that the set used for design consists of 5000 data. The control algorithm on-line evaluation times resulted also quite low, confirming the results of Section 7, and showing that the proposed algorithms can be effectively implemented on real-time processors.

For comparison, the controller in [46] has been considered, designed by means of a two-step method, consisting in LPV model identification and Gain Scheduling (GS) design.

8.4 Controller tests

A first simulation was performed to test all the controllers in the task of reference tracking. Zero initial conditions were assumed. A reference signal of length 5000 samples (corresponding to 100 s) was used, defined as a random sequence of step signals with amplitudes in the interval $[-\pi,\pi]$, filtered by a second-order filter with a cutoff frequency of 10 rad/s. This filter was inserted in order to ensure not too high variations. The outputs were corrupted by random uniform noises with amplitude 0.02 rad ($SNR \cong 100$). In Figure 5, the angular positions of the system controlled by the NIC1 algorithm are compared with the references for the first 20 s of this simulation. Note that the two position references were chosen quite similar to each other (but not equal) in order to allow the manipulator to reach in a simple way any position in its range.

A second simulation was performed to test the controllers in the task of disturbance attenuation. Null initial conditions and reference were assumed. An output disturbance signal of length 1000 samples (corresponding to 20 s) was considered, defined as a sequence of two steps (one for each output channel) of amplitude 1 rad, filtered by a second-order filter with a cutoff frequency of 10 rad/s. The outputs were also corrupted by random uniform noises with amplitude 0.02 rad ($SNR \cong 100$). In Figure 6, the angular positions of the system controlled by the NIC1 algorithm are shown, together with the disturbance signals.

	NIC1	NIC2
maximum polynomial degree d_{max}	8	8 (degree 1 in u_t)
selected polynomial degree	4	4 (degree 1 in u_t)
maximum model order n_{max}	4	4
selected model order	2	2
maximum prediction horizon τ_{max}	10	10
selected prediction horizon	5	5
total number of basis functions	231	187
number of selected basis functions for the two outputs	185 198	156 163
μ	0.01	0.01
$\hat{\gamma}_\Delta$	0.85	0.85
estimated γ_F	0.06	0.09
identification algorithm run time [s] (average)	185	167
control algorithm	$K^{(3)}$	$K^{(2)}$
control algorithm sampling time [s]	0.02	0.02
control algorithm on-line evaluation time [s] (average)	2.1e-3	8.1e-4

Table 2
Description of the NIC controllers.

8.5 Monte Carlo simulations

A Monte Carlo (MC) simulation was carried out, where the procedure data generation - control design - tracking test was repeated 200 times. For each trial, the tracking performance was evaluated by means of the Root Mean Square tracking errors, defined as

$$RMS_i \doteq \sqrt{\frac{1}{5000}\sum_{t=1}^{5000}(r_{i,t}-y_{i,t})^2},\ i=1,2$$

where $r_{i,t}$ is the ith component of the reference signal and $y_{i,t}$ is the ith component of the controlled system output. The average errors $\overline{RMS}_i$ resulting from the MC simulation are reported in Table 3.

A second MC simulation was then carried out, similar to the previous one, except that a smaller SNR level was assumed, in both the data generation and tracking test phases. In particular, the noises corrupting the outputs were characterized by a signal-to-noise ratio $SNR \cong 10$. This SNR level is too low for a modern robotic application but here it is useful to test the NIC approach in the

	NIC1	NIC2	GS
$\overline{RMS}_1$[rad]	0.159	0.160	0.167
$\overline{RMS}_2$[rad]	0.114	0.115	0.152

Table 3
Robot Manipulator. Average RMS tracking errors.

presence of significant noises. In this second MC study, only NIC controllers based on purely nonlinear prediction models were designed. The average errors $\overline{RMS}_i$ resulting from the MC simulation for these controllers are $\overline{RMS}_1 = 0.192$ rad and $\overline{RMS}_2 = 0.169$ rad.

8.6 Comments

From the obtained results, it can be concluded that the NIC control systems are quite effective, showing a precise tracking, a significant disturbance attenuation capability, and robustness versus measurement noises. In comparison with the GS method of [46], the NIC approach is simpler, since a polynomial model of the form (4) has in general a significantly simpler structure wrt an LPV model (and, in particular, wrt a state-space LPV model). Moreover, the tracking results obtained by the NIC controllers are similar to (or even slightly better than) those obtained by the GS controller, despite the fact that this latter uses a stronger information on the system (22) (i.e., the information that (22) is a quasi-LPV system). Note that we applied NIC control in several simulated examples where Gaussian noises were used, including braking control [19], control of the Duffing oscillator [52], control of air and charging systems of diesel engines [38]. No significant differences wrt the case of uniform noise considered here were noted, in terms of control design and performance.

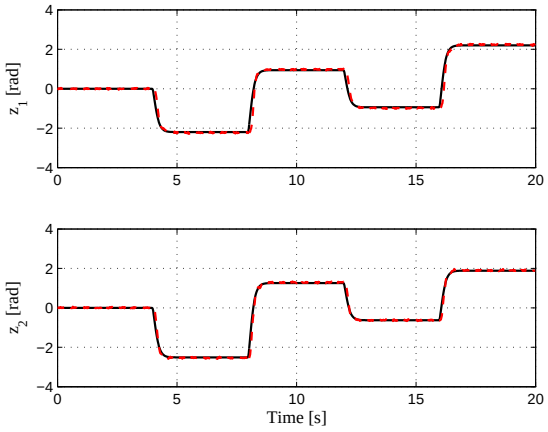


Figure 5. Robot Manipulator. Continuous (black) line: reference. Dashed (red) line: closed-loop system output.

9 Comparison with related methods

The NIC approach has a clear analogy with NMPC (see, e.g., [17,36,23]): in both the approaches, the command input is chosen on the basis of some prediction and on-line optimization. However, NIC has two advantages wrt standard NMPC: 1) NIC does not need a physical model; 2) NIC is in general numerically more efficient than standard NMPC. Non-standard NMPC

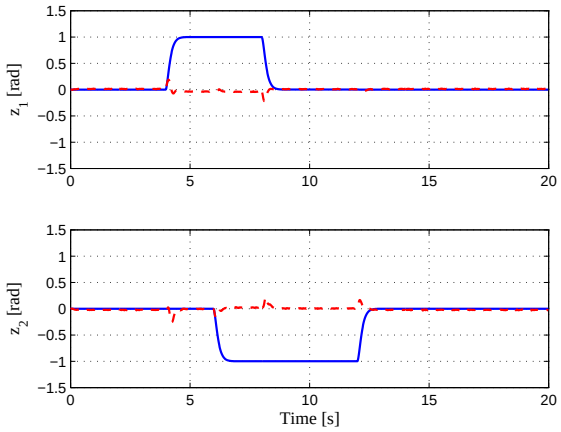


Figure 6. Robot Manipulator. Continuous (blue) line: disturbance. Dashed (red) line: closed-loop system output.

approaches that are more similar to NIC are those presented in [61,56]. They use polynomial models that can be identified from data, and thus can avoid the need of a physical model. However, these approaches do not give a priori stability/accuracy guarantees and also do not appear to be suitable for fast on-line applications in the MIMO case. In summary, NIC can be seen as a data-driven efficient NMPC.

The NIC approach has a similarity with the DFK (Direct FeedbacK design) method of [50]: both the approaches use an inverse model to control the system of interest. In DFK, the inverse model is directly identified from data. In NIC, inversion is performed using a direct model. The main advantage of NIC is that it allows the control of systems described by a function h non-invertible with respect to u_t. DFK is in general simpler and does not require any kind of model but may not be applicable when h is not invertible. Moreover, DFK is based on a full-state feedback scheme and its application to MIMO systems needs still to be developed.

A simplified version of the NIC approach is presented in [19,52,51]. However, these works only consider the case where the system to control is SISO (Single Input Single Output) and the model prediction horizon is 1. Besides these general aspects, the innovative contributions provided in the present paper with respect to [19,52,51] can be summarized as follows: The identification algorithm proposed here is novel, since it is able to deal with MIMO systems and prediction horizons larger than 1; it is also more systematic and, wrt the ones in [19,52,51], less time consuming. The optimization algorithms for MIMO system control presented in this paper are novel. No closed-loop stability analysis is made in [19,52]. The stability analysis is carried out in [51] only for SISO systems. The NIC design procedure developed in Section 6 is original. The numerical analysis about efficiency of the MIMO control algorithms is novel.

10 Conclusion

A novel data-driven control approach for MIMO nonlinear systems has been proposed in the paper. This approach, called NIC (Nonlinear Inversion Control) shows interesting advantages wrt the state of the art methods: On one hand, the approach is characterized by a theo-

retical foundation, possibly allowing us to guarantee a priori important properties such as closed-loop stability and tracking error accuracy. On the other hand, the approach presents important features by which it can be very effective in real-world applications: it is systematic and relatively simple; it can be applied to a wide class of nonlinear systems; it is numerically efficient.

The approach has been tested with success in several numerical examples, including control of braking systems [19], control of the Duffing oscillator [52], cancer immunotherapy control [53], control of a robot manipulator (this paper), and in two real-data applications, concerned with control of the air and charging systems for diesel engines [38] and glucose regulation in type 1 diabetic patients [54].

Future research activities will be devoted to the utilization of basis functions alternative to polynomials and to the systematic comparison of our minimization techniques with other techniques, based on sum of squares, eigenvalue methods or Gröbner bases.

11 Appendix

Proof of Theorem 1. From (3) and (11), it is immediate to verify that the closed-loop system (10) is described by the following equation:

$$\begin{aligned} &y_{t+\tau} = g(u_t^v, y_t^-, \xi_t^v) \\ &= r_{t+\tau} - F\left(r_{t+\tau}, u_t^p, y_t^-\right) + \Delta\left(u_t^v, y_t^-\right) + g_t^\xi \xi_t^v \end{aligned} \tag{24}$$

where $u_t^v \doteq \left(u_{t+\tau-1}^*, \ldots, u_{t-n}^*\right)$ and g_t^ξ is a row vector such that

$$\begin{aligned} &g(u_t^v, y_t^-, \xi_t^v) = g^o(u_t^v, y_t^-) + g_t^\xi \xi_t^c \\ &\left\| g_t^\xi \right\| \leq \gamma_\xi, \ \forall t. \end{aligned}$$

Existence of this vector is guaranteed as g is Lipschitz continuous on $U^{n+\tau} \times Y^n \times \Xi^{n+\tau}$. From (24), we obtain the following error equation:

$$\begin{aligned} &e_{t+\tau} = F\left(r_{t+\tau}, u_t^p, y_t^-\right) - \Delta\left(u_t^v, y_t^-\right) - g_t^\xi \xi_t^v \\ &= F\left(r_{t+\tau}, u_t^p, r_t^- - e_t^-\right) - \Delta\left(u_t^v, r_t^- - e_t^-\right) - g_t^\xi \xi_t^v \end{aligned}$$

where $e_t \doteq r_t - y_t$, $r_t^- \doteq (r_{t-1}, \ldots, r_{t-n})$ and $e_t^- \doteq (e_{t-1}, \ldots, e_{t-n})$. Since Δ and F are Lipschitz continuous on Ω_Δ and Ω_F, respectively, the following representations can be introduced:

$$\begin{aligned} F\left(r_{t+\tau}, u_t^p, r_t - e_t^-\right) &= a_t^F e_t^- + F\left(r_{t+\tau}, u_t^p, r_t^-\right) \\ \Delta\left(u_t^v, r_t^- - e_t^-\right) &= -a_t^\Delta e_t^- + \Delta\left(u_t^v, r_t^-\right) \end{aligned}$$

where a_t^F and a_t^Δ are bounded as

$$\left\| a_t^F \right\| \leq \gamma_F, \ \left\| a_t^\Delta \right\| \leq \gamma_\Delta \tag{25}$$

and $F\left(r_{t+\tau}, u_t^p, r_t^-\right)$ and $\Delta_{\tau-i}\left(u_t^v, r_t^-\right)$ are bounded according to (12). The closed-loop error dynamics is thus described by

$$\begin{aligned} &e_{t+\tau} = \left(a_t^F + a_t^\Delta\right) e_t^- \\ &+F\left(r_{t+\tau}, u_t^p, r_t^-\right) - \Delta\left(u_t^v, r_t^-\right) - g_t^\xi \xi_t^v. \end{aligned}$$

From (12), (13) and (25), it follows that

$$\|e_{t+\tau}\| \leq \gamma_c \left\|e_t^-\right\| + \sigma_F + \sigma_\Delta + \gamma_\xi \bar{\xi},$$

for any $y_t^- \in Y^n$, $r_{t+\tau} \in R$, $r_t^- \in R^n$, $u_t \in U$ and $\xi_t^v \in \Xi$. This inequality implies that

$$\begin{aligned} \|e_{t+\tau}\| &\leq \gamma_c \max\left(\|\boldsymbol{e}\|, \|e^-\|\right) + \sigma_F + \sigma_\Delta + \gamma_\xi \bar{\xi} \\ &= \gamma_c \|\boldsymbol{e}\| + \sigma_F + \sigma_\Delta + \gamma_\xi \bar{\xi} \end{aligned} \tag{26}$$

where $\boldsymbol{e} = (e_1, e_2, \ldots)$ and $e^- \doteq (e_0, \ldots, e_{1-\tau-n}) = 0$. Note that e^- is zero as the reference is chosen as $r_t = \tilde{y}_t$, for $t \leq 0$. Since (26) holds for $t = 1-\tau, 2-\tau, \ldots$, we obtain the following inequality:

$$\|\boldsymbol{e}\| \leq \gamma_c \|\boldsymbol{e}\| + \sigma_F + \sigma_\Delta + \gamma_\xi \bar{\xi}.$$

Being $\gamma_c < 1$ by assumption, we can write

$$\|\boldsymbol{e}\| \leq \frac{1}{1-\gamma_c}\left(\sigma_\Delta + \sigma_F + \gamma_\xi \bar{\xi}\right) \tag{27}$$

which holds if $\left(y_0^-, \boldsymbol{r}, \boldsymbol{\xi}\right) \in \mathcal{Y}^0 \times \mathcal{R} \times \Xi$ and $y_t^- \in Y^n$, $\forall t > 0$. The first condition, i.e., $\left(y_0^-, \boldsymbol{r}, \boldsymbol{\xi}\right) \in \mathcal{Y}^0 \times \mathcal{R} \times \Xi$ is true by construction. The second condition, i.e.

$$y_t^- \in Y^n, \forall t > 0 \tag{28}$$

can be proven considering that

$$\|\boldsymbol{y}\| = \|\boldsymbol{r} - \boldsymbol{e}\| \leq \|\boldsymbol{r}\| + \|\boldsymbol{e}\|.$$

Thus, from (27),

$$\|\boldsymbol{y}\| \leq \|\boldsymbol{r}\| + \tfrac{1}{1-\gamma_c}\left(\sigma_\Delta + \sigma_F + \gamma_\xi \bar{\xi}\right) = \|\boldsymbol{r}\| + \bar{e}$$

which can be written as

$$\|y_{t+1}\| \leq \|\boldsymbol{r}\| + \bar{e}, \ t = 1, 2, \ldots \tag{29}$$

This inequality holds if $y_k^- \in Y^n$, $\forall k \leq t$. Condition (28) is now proved by induction.

First, consider that $y_0^- \in Y^n$ by choice. From (29), if $y_0^- \in Y^n$, then $\|y_1\| \leq \|\boldsymbol{r}\| + \bar{e}$ and thus, from assumption (14), $y_1 \in Y$, implying that $y_1^- \in Y^n$. Repeating this reasoning for all $t \geq 0$ we conclude that $y_t^- \in Y^n$ for all $t \geq 0$. This fact and inequalities (27) and (29) imply that claims (i) and (ii) hold true. □

References

[1] G. C. M. De Abreu, R. L. Teixeira, and J. F. Ribeiro. A neural network-based direct inverse control for active control of vibrations of mechanical systems. In *Proceedings of the sixth Brazilian Symposium on Neural Networks*, pages 107–112, 2000.

[2] D. B. Anuradha, G. Prabhaker Reddy, and J. S .N. Murthy. Direct inverse neural network control of a continuous stirred tank reactor (cstr). In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2009.

[3] K. J. Astrom and B. Wittenmark. *Adaptive Control*. Addison-Wesley, Reading, MA, 1995.

[4] Michael Bader. *How to Construct Space-Filling Curves*, pages 15–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[5] S. Battilotti. Robust stabilization of nonlinear systems with pointwise norm-bounded uncertainties: a control Lyapunov function approach. *IEEE Transactions on Automatic Control*, 44(1):3 – 17, 1999.

[6] R. Bellman. *Dynamic Programming*. P (Rand Corporation). Princeton University Press, 1957.

[7] M.D. Brown, G. Lightbody, and G.W. Irwin. Nonlinear internal model control using local model networks. *IEE Proceedings - Control Theory and Applications*, 144(6):505–514, 1997.

[8] J.B.D. Cabrera and K. S. Narendra. Issues in the application of neural networks for tracking based on inverse control. *IEEE Transactions on Automatic Control*, 44(11):2007–2027, 1999.

[9] G.C. Calafiore and L. El Ghaoui. *Optimization Models*. Cambridge University Press, 2014.

[10] M.C. Campi and S.M. Savaresi. Direct nonlinear control design: The virtual reference feedback tuning (VRFT) approach. *IEEE Transactions on Automatic Control*, 51(1):14–27, 2006.

[11] Fu-Chuang Chen and H.K. Khalil. Adaptive control of a class of nonlinear discrete-time systems using neural networks. *IEEE Transactions on Automatic Control*, 40(5):791–801, 1995.

[12] J. Chen and G. Gu. *Control-Oriented System Identification: An H_∞ Approach*. John Wiley & Sons, New York, 2000.

[13] G. Chesi, A. Garulli, A. Tesi, and A. Vicino. *Homogeneous Polynomial Forms for Robustness Analysis of Uncertain Systems*. Springer, 2009.

[14] P. Dreesen, K. Batselier, and B. De Moor. Back to the roots: Polynomial system solving, linear algebra, systems theory. In *16th IFAC Symposium on System Identification*, Brussels, Belgium, 2012.

[15] P. Dreesen and B. De Moor. Polynomial optimization problems are eigenvalue problems. In *Model-Based Control*, pages 49–68. Springer, 2009.

[16] L. Fagiano and C. Novara. Learning a nonlinear controller from data: theory, computation and experimental results. *IEEE Transactions on Automatic Control*, 61(7):1854–1868, 2016.

[17] R. Findeisen, F. Allgower, and L.T. Biegel. Assessment and future directions of nonlinear model predictive control. In *Lecture Notes in Control and Information Sciences*. Springer, 2007.

[18] S. Formentin, P. De Filippi, M. Corno, M. Tanelli, and S.M. Savaresi. Data-driven design of braking control systems. *IEEE Transactions on Control Systems Technology*, 21(1):186–193, 2013.

[19] S. Formentin, C. Novara, S.M. Savaresi, and M. Milanese. Active braking control system design: the D2-IBC approach. *IEEE/ASME Transactions on Mechatronics*, 20(4):1573–1584, 2015.

[20] G. Franzè. A nonlinear sum-of-squares model predictive control approach. *IEEE Transactions on Automatic Control*, 55(6):1466–1471, 2010.

[21] A. Freeman and V. Kokotovic. *Robust Nonlinear Control Design*. Birkhï¿œuser, Boston, 1996.

[22] L.B. Freidovich and H.K. Khalil. Performance recovery of feedback-linearization-based designs. *IEEE Transactions on Automatic Control*, 53(10):2324 – 2334, 2008.

[23] L. Grune and J. Pannek. Nonlinear model predictive control - theory and algorithms. In *Communications and Control Engineering*. Springer, 2011.

[24] Z. Hou, H. Gao, and F. L. Lewis. Data-driven control and learning systems. *IEEE Transactions on Industrial Electronics*, 64(5):4070–4075, May 2017.

[25] Z. Hou and S. Jin. Data-driven model-free adaptive control for a class of mimo nonlinear discrete-time systems. *IEEE Transactions on Neural Networks*, 22(12):2173–2188, Dec 2011.

[26] Z. Hou and S. Jin. A novel data-driven control approach for a class of discrete-time nonlinear systems. *IEEE Transactions on Control Systems Technology*, 19(6):1549–1558, 2011.

[27] Z. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3 – 35, 2013. Data-based Control, Decision, Scheduling and Fault Diagnostics.

[28] K. Hsu, C. Novara, T. Vincent, M. Milanese, and K. Poolla. Parametric and nonparametric curve fitting. *Automatica*, 42/11:1869–1873, 2006.

[29] K. J. Hunt and D. Sbarbaro. Neural networks for nonlinear internal model control. *Control Theory and Applications, IEE Proceedings D*, 138(5):431–438, 1991.

[30] A. Isidori. *Nonlinear Control Systems*. Springer, 1995.

[31] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 1996.

[32] A. Kwiatkowski and H. Werner. LPV control of a 2-DOF robot using parameter reduction. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, 2005.

[33] J.B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Imperial College Press, 2009.

[34] A.U. Levin and K.S. Narendra. Control of nonlinear dynamical systems using neural networks. II. observability, identification, and control. *IEEE Transactions on Neural Networks*, 7(1):30–42, 1996.

[35] F.L. Lewis, D. Vrabie, and K.G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *Control Systems, IEEE*, 32(6):76–105, 2012.

[36] L. Magni, D.M. Raimondo, and F. Allgower. Nonlinear model predictive control - towards new challenging applications. In *Lecture Notes in Control and Information Sciences*. Springer, 2009.

[37] C. Maier, C. Bohm, F. Deroo, and F. Allgower. Predictive control for polynomial systems subject to constraints using sum of squares. In *49th IEEE Conference on Decision and Control*, Atlanta, GA, USA, 2010.

[38] M. Milanese, I. Gerlero, C. Novara, G. Conte, M. Cisternino, C. Pedicini, V. Alfieri, and S. Mosca. Nonlinear mimo data-driven control design for the air and charging systems of diesel engines. In *SAE - ICE2015 - 12th International Conference on Engines and Vehicles*, 2015.

[39] M. Milanese and C. Novara. Set membership identification of nonlinear systems. *Automatica*, 40/6:957–975, 2004.

[40] M. Milanese and C. Novara. Model quality in identification of nonlinear systems. *IEEE Transactions on Automatic Control*, 50(10):1606–1611, 2005.

[41] M. Milanese and C. Novara. Computation of local radius of information in SM-IBC identification of nonlinear systems. *Journal of Complexity*, 23:937–951, 2007.

[42] M. Milanese and C. Novara. Unified set membership theory for identification, prediction and filtering of nonlinear systems. *Automatica*, 47(10):2141–2151, 2011.

[43] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[44] S.G. Nersesov and M.M. Haddad. On the stability and control of nonlinear dynamical systems via vector Lyapunov functions. *IEEE Transactions on Automatic Control*, 51(2):203 – 215, 2006.

[45] M. Norgaard, O. Ravn, N. Poulsen, and L. K. Hansen. *Neural Networks for Modeling and Control of Dynamic Systems*. Springer, 2000.

[46] C. Novara. Set membership identification of state-space LPV systems. In P. Lopes dos Santos, T.P. Azevedo Perdicoúlis, C. Novara, J.A. Ramos, and D.E. Rivera, editors, *Linear Parameter-Varying System Identification – New Developments and Trends, Advanced Series in Electrical and Computer Engineering Vol. 14*, pages 65–93. World Scientific, 2011.

[47] C. Novara. Sparse identification of nonlinear functions and parametric set membership optimality analysis. *IEEE Transactions on Automatic Control*, 57(12):3236–3241, 2012.

[48] C. Novara. Polynomial model inversion control: numerical tests and applications. *arXiv*, (1509.01421), 2015.

[49] C. Novara, M. Canale, M. Milanese, and M.C. Signorile. Set Membership inversion and robust control from data of nonlinear systems. *International Journal of Robust and Nonlinear Control*, 24(18):3170–3195, 2014.

[50] C. Novara, L. Fagiano, and M. Milanese. Direct feedback control design for nonlinear systems. *Automatica*, 49(4):849–860, 2013.

[51] C. Novara and S. Formentin. Data-Driven Inversion-Based Control of Nonlinear Systems with Guaranteed Closed-Loop Stability. *IEEE Transactions on Automatic Control*, 63(4):1147–1154, 2018.

[52] C. Novara, S. Formentin, S.M. Savaresi, and M. Milanese. Data-driven design of two degree-of-freedom nonlinear controllers: the D2-IBC approach. *Automatica*, 72:19–27, 2016.

[53] C. Novara and M. Karimshoushtari. A data-driven model inversion approach to cancer immunotherapy control. In *55th IEEE Conference on Decision and Control*, Las Vegas (Nevada), USA, 2016.

[54] C. Novara, I. Rabbone, and D. Tinti. Data-driven polynomial mpc and application to blood glucose regulation in a diabetic patient. In *Proc. of the European Control Conference*, pages 1722–1727, Limassol, Cyprus, 2018.

[55] C. Novara, T. Vincent, K. Hsu, M. Milanese, and K. Poolla. Parametric identification of structured nonlinear systems. *Automatica*, 47(4):711 – 721, 2011.

[56] R.S. Parker. Nonlinear model predictive control of a continuous bioreactor using approximate data-driven models. In *American Control Conference*, Anchorage, AK, USA, 2002.

[57] P.A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2000.

[58] M.M. Polycarpou. Stable adaptive neural control scheme for nonlinear systems. *IEEE Transactions on Automatic Control*, 41(3):447–451, 1996.

[59] Z. Qu. *Robust Control of Nonlinear Uncertain Systems*. Wiley series in nonlinear science, 1998.

[60] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B.Delyon, P. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31:1691–1723, 1995.

[61] G. Ravi Sriniwas and Y. Arkun. A global solution to the nonlinear model predictive control algorithms using polynomial arx models. *Computers and Chemical Engineering*, 21(4):431–439, 1997.

[62] R Tibshirani. Regression shrinkage and selection via the Lasso. *Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[63] J.A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030 –1051, mar. 2006.

[64] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. of Optimization Theory and Applications*, 109(3):475–494, 2001.

[65] B. Yao and M. Tomizuka. Adaptive robust control of MIMO nonlinear systems in semi-strict feedback forms. *Automatica*, 37, 2001.

[66] A. Yesildirek and L. Lewis. Feedback linearization using neural networks. *Automatica*, 31(11):1659–1664, 1995.

[67] Shen Yin, Hao Luo, and S.X. Ding. Real-time implementation of fault-tolerant control systems with performance optimization. *IEEE Transactions on Industrial Electronics*, 61(5):2402–2411, 2014.