2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Direct data-driven design of neural reference governors

Daniele Masti†, Valentina Breschi⋆, Simone Formentin⋆, Alberto Bemporad†

Abstract— **In this paper, we present a direct data-driven approach to synthesize model reference controllers for constrained nonlinear dynamical systems. To this aim, we employ a hierarchical structure composed by a receding-horizon reference governor and a data-driven low-level controller. Unlike existing approaches, here we jointly design the two blocks by solving a single optimization task, exploiting the fact that the inner controller will never be used alone. The performance of the proposed method is assessed by means of two simulation examples, involving the control of two highly nonlinear benchmark systems.**

I. Introduction

Designing an effective control law without knowing the physics of the plant has always been a topic of interest in the control system community. The indirect approach usually employed in this case is to first identify an open-loop model of the underlying system via system identification procedures [1], validate the learned model, and then proceed with the design of a model-based controller. While this strategy is convenient, as it effectively introduces a decoupling layer between modeling and control design, it can also cause an undue burden on the system identification step, since many control techniques do not actually require a fully-featured model to synthesize an appropriate control law. Although exceptions exist (see, for instance, [2], [3], [4] in the context of predictive controllers), this means that there exists a large misalignment between the target of most identification techniques, which aim for the best open-loop simulation accuracy, and the actual requirements of control design.

This discrepancy has stimulated research into *direct data-driven control* techniques [5], [6], that rely on input-output data to directly synthesize a control law, without first identifying an open-loop model of the plant. Within this setting, approaches exist that rely either on the real-time acquisition of data (like in Reinforcement Learning approaches [7]), or exploiting in batch mode a previously collected dataset [8]. Among the latter class of approaches, we recall the *Virtual Reference Feedback Tuning* (VRFT) method, which has been applied successfully to control linear and nonlinear dynamical systems [9], [10], [11]. Within the VRFT framework, the design of the control law is recast into an identification problem, where the aim is to reproduce the collected input sequence via a *virtual tracking error* specially crafted so that the attained closed-loop behavior matches the response of a user-provided reference model. Nonetheless, off-the-shelf architectures of this kind do not allow taking signal constraints into account. For this reason, several extensions have thus been proposed to overcome this limitation. For instance, in [12] a combined control+antiwindup design strategy based on artificial neural network is presented; in [13] a Q-learning approach is used to enhance a baseline VRFT controller; in [14] a method to deal with constraints on the sensitivity function is proposed in the context of $\mathcal{H}_\infty$ control. Nonetheless, to obtain the desired results, all such methods are no longer one-shot learning procedures as the VRFT approach and they can deal only with specific kinds of constraints. A more general approach is presented in [15], where a reference governor [16] is used on top of a standard VRFT-based controller to ensure constraints satisfaction and to boost closed-loop performance. This approach is still a one-shot data-acquisition procedure, but, at the same time, it requires a 2 Degrees-Of-Freedom (2 DOF) architecture that is harder to tune. Furthermore, for the overall scheme to handle constraints, the VRFT-based controller will never be used alone, but such a characteristic is not accounted for in the design phase.

In this work, we overcome these limitations by envisioning a data-driven receding-horizon control solution that can guarantee constraints satisfaction. To do so, we resort to ideas taken from control-oriented system identification, by learning short-term fixed-horizon models based on Artificial Neural Networks to obtain a simplified linear time-varying representation of the implicit controller. Moreover, we exploit the fact that the inner controller will be always paired with a reference governor to avoid the (more complex) search for a controller that can work alone.

The paper is organized as follows. In Section II we formally state the problem of learning both the external receding-horizon reference governor and the inner controller. Section III is devoted to the presentation of the proposed controller design approach. Numerical results are reported in Section IV and some concluding remarks are drawn in Section V.

II. Setting and goals

Consider the following discrete-time dynamical system

$$\Sigma_P = \begin{cases} x_{k+1} = f_\Sigma(x_k, u_k) \\ y_k = h_\Sigma(x_k) \end{cases} \quad (1)$$

† Daniele Masti and Alberto Bemporad are with IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy {daniele.masti, alberto.bemporad}@imtlucca.it

⋆ Valentina Breschi and Simone Formentin are with Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy {valentina.breschi, simone.formentin}@polimi.it

This project was partially supported by the Italian Ministry of University and Research under the PRIN'17 project "Data-driven learning of constrained control systems" , contract no. 2017J89ARP

with $u_k \in \mathbb{R}^{N_U}$, $y_k \in \mathbb{R}^{N_Y}$, and f_Σ, h_Σ being two *unknown* nonlinear functions.

Suppose that a set $Z_\mathcal{N} = \{(u_1, y_1), \ldots, (u_\mathcal{N}, y_\mathcal{N})\}$ comprising $\mathcal{N}$ input/output pairs acquired from Σ_P is available.

Our first aim is to exploit the available dataset $Z_\mathcal{N}$ to learn a controller C_θ, a-priori parameterized by a vector θ of parameters, so that the reference-to-output behavior of the resulting closed-loop system corresponds to the input-output behavior of a given reference model $\mathcal{M}$. The latter is described by the possibly nonlinear mapping $f_\mathcal{M}$

$$y^\mathcal{M}_{k+1} = f_\mathcal{M}(y^\mathcal{M}_k, \ldots, y^\mathcal{M}_{k-M}, r_k, r_{k-M}) \tag{2}$$

with $y^\mathcal{M}_k \in \mathbb{R}^{N_Y}$ and $r_k \in \mathbb{R}^{N_Y}$. Our second and concurrent goal is to design an outer loop with a model predictive reference governor, so to boost performance and handle signal constraints [16].

To accomplish the first learning task, according to the VRFT framework let us define as *virtual reference* the sequence $\{r^\star_k\}$, $r^\star_k \in \mathbb{R}^{N_Y}$, that would reproduce $\{y_k\}$ if fed to $\mathcal{M}$, namely the minimizer of the optimization problem

$$\begin{aligned} \{r^\star_k\} = \arg\min_{\{r_k\}} \quad & \sum_{k=M}^{\mathcal{N}} \mathcal{L}_\mathrm{R}(y^\mathcal{M}_k, y_k) \\ \text{subject to} \quad & \text{Equation (2)} \end{aligned} \tag{3}$$

where $\mathcal{L}_\mathrm{R} : \mathbb{R}^{N_Y} \times \mathbb{R}^{N_Y} \to \mathbb{R}$ is an appropriate loss function. We stress that many approaches have been devised to solve problem (3), and that explicit exact recursive expressions for $\{r^\star_k\}$ have been found for various classes of reference models $\mathcal{M}$ for $\mathcal{L}_\mathrm{R}(\cdot,\cdot) = \|y^\mathcal{M}_k - y_k\|^2_2$, see *e.g.,* [10].

Once $\{r^\star_k\}$ has been computed, one way to synthesize the controller C_θ that achieves closed-loop performance closest to the one of $\mathcal{M}$ is to look for the controller that best fits $\{u_k\}$ when fed by the *virtual tracking error* $e^\star_k \triangleq r^\star_k - y_k$:

$$\begin{aligned} \min_{C_\theta \in \mathcal{H}} \quad & \sum_{k=M}^{\mathcal{N}} \mathcal{L}_\mathrm{C}(\hat{u}_k, u_k) \\ \text{subject to} \quad & \hat{u}_{k+1} = C_\theta(u_k, \ldots, u_{k-M}, e^\star_{k+1}, \ldots, e^\star_{k-M}) \end{aligned} \tag{4}$$

where $\mathcal{L}_\mathrm{C} : \mathbb{R}^{N_U} \times \mathbb{R}^{N_U} \to \mathbb{R}$ is again an appropriate loss function. The above *model-free* VRFT technique was first introduced in [9] in the linear time-invariant setting, and then successfully employed to design controller for various classes of nonlinear systems (see *e.g.,* [10]).

Albeit appealing due to its simple nature, solving a one-step-ahead prediction-error minimization problem like the one in (4) can be problematic. Indeed, we cannot guarantee that the resulting control law may be suited to be repeatedly iterated over time, although accurate over a short horizon. Various approaches have been devised to overcome this issue, but they usually result into a tangible increase of the computational complexity of the overall learning scheme [17]. While this is unavoidable if the aim is to synthesize a stand-alone controller C_θ, we will show here that *this is not the case if our goal is to use C_θ within a reference governor scheme*, like the one of [15].

Indeed, as described in [15], a reference governor for an already existing VRFT controller consists of solving at each sampling step k the following optimization problem:

$$\begin{aligned} \min_{R_k} \quad & \mathcal{L}_\mathrm{RG}(Y_k, U_k, R_k, \bar{Y}) \\ \text{subject to} \quad & Y_k = \mathbf{F}^\mathcal{M}(\bar{x}_k, R_k) \left\{ \begin{array}{l} \text{Fixed horizon} \\ \text{closed loop} \\ \text{reference dynamics} \end{array} \right. \\ & U_k = \mathbf{C}(\bar{x}_k, E_k) \left\{ \begin{array}{l} \text{Fixed horizon VRFT} \\ \text{controller dynamics} \end{array} \right. \\ & R_k = [r_k, \ldots, r_{k+T}]' \in \mathcal{R} \\ & U_k = [u_k, \ldots, u_{k+T}]' \in \mathcal{U} \\ & Y_k = [y_{k+1}, \ldots, y_{k+T}]' \in \mathcal{Y} \\ & E_k = [r_k - y_k, \ldots, r_{k+T} - y_{k+T}]' \end{aligned} \tag{5}$$

where $\bar{Y} \in \mathbb{R}^{N_Y T}$ is the reference to be tracked, T is the chosen prediction horizon, $\mathcal{L}_\mathrm{RG}$ is a suitable loss function, and $\bar{x}_k$ is a feedback information vector, which encompasses the internal states of both the controller and the underlying system. By looking at the optimization problem in (5), it is clear that the performance of the governor is linked to the prediction capabilities of $\mathbf{F}^\mathcal{M}$ and the capability of $\mathbf{C}$ to make the internal loop behave as closely as possible to $\mathbf{F}^\mathcal{M}$ itself *within the chosen prediction horizon*. This means that if we do not want to use the internal controller alone, we can directly learn the mapping $\mathbf{C}$ from $\bar{x}_k$ and R_k to U_k, rather than solving (4).

Remark 1: Without loss of generality, $\mathcal{L}_\mathrm{RG}$ may also depend on Y_k and U_k to further shape the overall characteristics of the closed loop.

III. Data-driven design of reference governors

Based on the above intuition, consider the problem of fitting a fixed horizon VRFT controller like $\mathbf{C}$ introduced in (5). At time k and over the horizon T, the value of the signals $\hat{u}_{k+n}$, $n = 0, \ldots, T$, are only a function of the virtual tracking error $e^\star_k, \ldots, e^\star_{k+n}$ and the initial state x_k, *i.e.,* there exists a set of maps f_n so that

$$\hat{u}_{k+n} = f_n(x_k, e^\star_k, \ldots, e^\star_{k+n}) \text{ for } n = 0, \ldots, T \tag{6}$$

for each $T \in \mathbb{N}$. Without loss of generality, we can set $x_k = [u_{k-1}, \ldots, u_{k-M}, e^\star_{k-1}, \ldots, e^\star_{k-M+1}]$ and define the map $H_T : \mathbb{R}^{N_x \times N_Y (T+1)} \to \mathbb{R}^{N_U (T+1)}$ compactly as

$$\hat{U}_T \triangleq \begin{bmatrix} \hat{u}_k \\ \vdots \\ \hat{u}_{k+T} \end{bmatrix} = H_T(x_k, E^T_k) \tag{7}$$

where $E^T_k \triangleq [e^\star_k, \ldots, e^\star_{k+T}]'$.

Compared to learning a recursive law C_θ as in (4), retrieving H_T in (7) from data is more convenient as (i) it completely circumvents the problem of using a fitting procedure able to minimize the *simulation error* of C_θ, and (ii) it allows us to resort to standard *prediction error* approaches. However, this kind of map is still quite problematic for the additional design of a reference governor, due to its nonlinear behavior with respect to the decision variable E_k.

A first solution to overcome this issue would be to resort to nonlinear optimal control schemes, but this would be computationally expensive as it would require the evaluation of the Jacobian of the involved map at least once for each step k.

Nonetheless, in a large number of applications the expressiveness of nonlinear schemes like the one in (7) may as well be not necessary, especially when one aims at obtaining accurate controllers for short-term predictions only. To exploit this fact, we fit a map in which the predicted control action is nonlinearly dependent *only* on the information vector x_k (that comes from feedback), and possibly on some reference trajectory $\bar{E}_k^T = [\bar{e}_k, \dots, \bar{e}_{k+T}]'$ (which will likewise not be a decision variable), whereas it is *linearly* dependent on the control variables $\Delta E_k^T \doteq E_k^T - \bar{E}_k^T$. This is equivalent to fit a partial first-order Taylor approximation of the map in (7).

Indeed, by assuming H_T to be at least C^1 in the neighborhood of a given trajectory, it holds that

$$\begin{aligned} H_T(x_k, e_k, \dots, e_{k+T}) \approx H_T(\bar{x}_k, \bar{e}_k, \dots, \bar{e}_{k+T}) + \\ \frac{\partial H_T}{\partial x_k}(\bar{x}_k, \bar{e}_k, \dots, \bar{e}_{k+T})\Delta x_k + \\ \sum_{j=0}^{T} \frac{\partial H_T}{\partial e_{k+j}}(\bar{x}_k, \bar{e}_k, \dots, \bar{e}_{k+T})\Delta e_{k+j}. \end{aligned} \tag{8}$$

Since identifying $\frac{\partial H_T}{\partial x_k}\Delta x_k$ is not necessary as x_k is not a decision variable for the governor, our goal becomes to design a controller by fitting a predictor of the form

$$\hat{U}_T = G_T(x_k, \bar{E}_k^T) + F_T(x_k, \bar{E}_k^T)\Delta E_k^T. \tag{9}$$

The above architecture mandates the introduction of $\bar{E}_k$. A possible approach to bypass this requirement is to exploit the fact that holding an explicit relationship to a time-dependent reference trajectory can still be more expressive then necessary in short-term predictions [18]. In this light, one can remove the dependence on $\bar{E}_k$ from (9) by assuming $\bar{E}_k = \mathbf{0}$, so as to obtain a controller parameterized as a predictor of the form:

$$\hat{U}_T = G_T(x_k) + F_T(x_k)E_k^T. \tag{10}$$

While the affine form in (10) is quite convenient also from a computational point of view, it severely limits the expressiveness of the predictor as a whole, which is particularly concerning as this architecture cannot be used to represent any nonlinear input characteristic that is dependent on the input itself.

To overcome this issue, we drop the dependency from $\bar{E}_T$ only in G_T, while maintaining it in F_T. This allows us to choose the real input sequence E_k^T as reference trajectory and to set $\bar{E}_k^T = E_k^T$, following the rationale of classical linearization-based control, according to which the most accurate linearization point is the one corresponding to the then-applied input trajectory. Once the resulting controller is deployed, the quantity $\bar{E}_k$ is instead chosen as the shifted optimal sequence computed at the previous time step. This finally results in trying to learn a controller of the form

$$\hat{U}_T = G_T(x_k) + F_T(x_k, E_k^T)E_k^T \tag{11}$$

which is still an approximation of a first-order Taylor expansion of H_T, but it does not require the design of $\bar{E}_k$. At the same time, this structure is sensibly more expressive than the one in (10) as it can better represents systems with non-affine input characteristics.

We point out that, compared to a standard sensitivity-based approach, evaluating $F_T(x_k, E_k^T)$ comes at no additional cost, making the computational requirements of the resulting controller comparable to a classical Linear Time Varying (LTV) solution.

Under the assumption that a data-driven controller designed to match the reference model $\mathcal{M}$ is feasible for Σ_P, then in closed loop it holds that $y_{k+1}^{\mathcal{M}} \approx y_{k+1}$. This also means that $e_{k+i} \approx e_{k+i}^{\mathcal{M}} = r_{k+i} - y_{k+i}^{\mathcal{M}}$. This approximation can be exploited to design the external reference governor.

Suppose that a fixed term predictor (11) has been trained to reproduce u_k and that the reference model $f_{\mathcal{M}}$ can be expanded into the same predictive form considered previously, namely

$$\hat{Y}^{\mathcal{M}} = \begin{bmatrix} y_{k+1}^{\mathcal{M}} \\ \vdots \\ y_{k+T+1}^{\mathcal{M}} \end{bmatrix} = G^{\mathcal{M}}(x_k^{\mathcal{M}}) + F_T^{\mathcal{M}}(x_k^{\mathcal{M}}, \bar{R}_k^T)R_k^T \tag{12}$$

with

$$\begin{aligned} R_k^T &= [r_k, \dots, r_{k+T}]', \\ x_k^{\mathcal{M}} &= [r_{k-1}, \dots, r_{k-M+1}, y_k^{\mathcal{M}}, \dots, y_{k-M+1}^{\mathcal{M}}] \end{aligned}$$

for some choice of $\bar{R}_k^T$. Then, the data-driven reference governor can be parameterized at each control step by solving

$$\begin{aligned} \min_{R_k^T} \quad & \mathcal{L}_{\mathrm{RG}}(\hat{Y}^{\mathcal{M}}, U_T, R_k^T, \bar{Y}) \\ \text{subject to} \quad & \hat{Y}^{\mathcal{M}} = G^{\mathcal{M}}(x_k^{\mathcal{M}}) + F_T^{\mathcal{M}}(x_k^{\mathcal{M}}, \bar{R}_k^T)R_k^T \\ & U_T = G_T(x_k) + F_T(x_k, \bar{E}_k^T)E_k^T \\ & E_k^T = [r_k - y_k, r_{k+1} - y_{k+1}^{\mathcal{M}}, \dots, \\ & \qquad r_{k+T} - y_{k+T}^{\mathcal{M}}]' \\ & \bar{E}_k^T = [\bar{r}_k - y_k, \bar{r}_{k+1} - \bar{y}_{k+1}^{\mathcal{M}}, \dots, \\ & \qquad \bar{r}_{k+T} - \bar{y}_{k+T}^{\mathcal{M}}]' \\ & U_T \in \mathcal{X}, \hat{Y}^{\mathcal{M}} \in \mathcal{Y}. \end{aligned} \tag{13}$$

where $\bar{Y}^{\mathcal{M}} = G^{\mathcal{M}}(x_k^{\mathcal{M}}) + F_T^{\mathcal{M}}(x_k^{\mathcal{M}}, \bar{R}_k^T)\bar{R}_k^T$ is the output of the reference model computed for $\bar{R}_k^T$.

If $\mathcal{X}, \mathcal{Y}$ are polytopes and $\mathcal{L}_{\mathrm{RG}}$ is either linear or quadratic, Problem (13) can be then reliably solved within real-time constraints [19], [20]. Note that, compared to the initial design problem in Equation (5), we have now separated $\bar{x}_k$ into two distinct quantities, namely the information coming from feedback x_k and $x_k^{\mathcal{M}}$, which depend on the reference model that approximately represents the inner closed-loop. We further highlight that while the fixed horizon predictor for $\hat{U}_T$ algebraically depends on r_k, the predictions $\hat{Y}^{\mathcal{M}}$ are obtained by using a strictly proper law. Indeed, we cannot expect the closed-loop system to be "delay-free" and, at the same time, we want to exploit the most recent feedback information to compute the current control action. This also

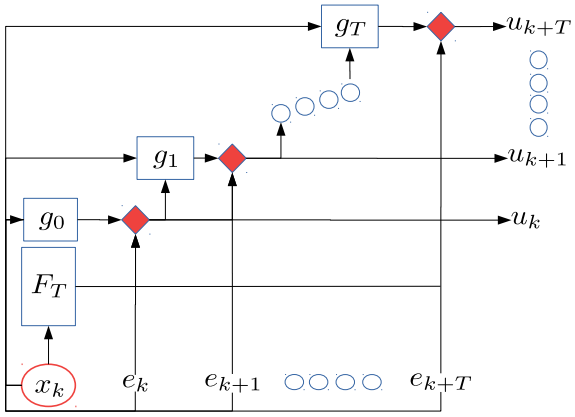

Fig. 1: Illustrative scheme of the empolyed ANN structure, adapted from [21].

ensures that the controller is not subject to mismatches, at least on the first step of the prediction horizon.

Remark 2: The presented architecture does not include an integral action and it is naturally prone to experience offset when tracking constant references. From a practical point of view, this offset can be seen as a constant disturbance $q \in \mathbb{R}^{N_y}$ that affects the output of the system. A way to overcome this issue would then be to recursively estimate the magnitude of such a disturbance and compensate the reference model accordingly. Indeed, for a classical single-output case, a recursive estimator for q can be designed as

$$\hat{q}_k = \hat{q}_{k-1} - \alpha(\hat{y}_k^{\mathcal{M}} - y_k), \tag{14}$$

where $\hat{y}_k^{\mathcal{M}}$ is the predicted evolution of the closed-loop system at time $k-1$, y_k is the feedback information at time k and $\alpha \in \mathbb{R}$. The estimate $\hat{q}_k$ can then be integrated in a reference governor scheme, which will now aim to solve

$$\begin{aligned} \min_{R_k^T} \quad & \mathcal{L}_{\text{RG}}(\hat{Y}^{\mathcal{M}}, U_T, R_k^T, \bar{Y}) \\ \text{subject to} \quad & \hat{Y}^{\mathcal{M}} = G^{\mathcal{M}}(x_k^{\mathcal{M}}) + F_T^{\mathcal{M}}(x_k^{\mathcal{M}}, \bar{R}_k^T) R_k^T + \hat{q}_k \mathbf{1} \\ & \text{as in Equation (13)} \end{aligned} \tag{15}$$

where $\mathbf{1} \in \mathbb{R}^T = [1, \dots, 1]'$. Similar correction will likewise be applied to $\bar{Y}^{\mathcal{M}}$. ■

A. ANNs for controller parameterization

As already noted in [21], directly fitting a set of discrete maps like the ones in Equations (6) can be quite an inefficient process, since no information from the predictions of the earlier terms along the horizon is necessarily re-used to predict the later ones. At best, this would make the training procedure more difficult to carry out. More likely, it will require the use of more powerful regression techniques to achieve the desired fitting performance, especially if we use approximators like Artificial Neural Networks (ANNs), that naturally benefit from the stacking of nonlinear layers/components.

A possible way to leverage the knowledge obtained from the approximators employed in shorter-term predictions is to use their outputs $\hat{u}_k$ as additional input for the regressor involved in the prediction of u_{k+n}, $n = 1, \dots, T$. In particular, when training the controller within our two-degree of freedom structure, we obtain

$$\begin{aligned} \hat{u}_{k+n} = \; & f_n(x_k) + g_n(x_k, \bar{E}_k^n) \begin{bmatrix} E_k^n \\ \hat{u}_{k+n-1} \end{bmatrix} \\ = \; & \tilde{f}_n(x_k) + \tilde{g}_n(x_k, \bar{E}_k^n) E_k^n \end{aligned} \tag{16}$$

for some choice of $\bar{E}_k^n$. This process can be repeated all over the horizon and might also include, when applicable, additional terms other than the one immediately preceding the n-th term. We stress that the number of past linear terms is a *hyper-parameter* of the approach, that can be tuned to trade-off between the accuracy of the predictor and the sparsity pattern of the resulting constraint structure.

The problem of learning *in parallel* a set of maps like (11) can be then recast into solving the following optimization problem:

$$\begin{aligned} \min_{\substack{f_0, \dots, f_T \\ g_0, \dots, g_T}} \quad & \sum_{k=M}^{\mathcal{N}-T} \mathcal{L}_{\text{train}}(\hat{O}_k, O_k) \\ & \text{subject to} \\ & \hat{O}_k = \begin{bmatrix} \hat{u}_k \\ \vdots \\ \hat{u}_{k+T} \end{bmatrix} O_k = \begin{bmatrix} u_k \\ \vdots \\ u_{k+T} \end{bmatrix} \\ & \hat{u}_k = f_0(x_k) + g_0(x_k, \bar{e}_k) e_k^\star \\ & \hat{u}_{k+i} = f_i(x_k) + \\ & \qquad g_i\left(x_k, \begin{bmatrix} \bar{e}_k \\ \vdots \\ \bar{e}_{k+i} \\ \hat{u}_{k+i-1} \end{bmatrix}\right) \begin{bmatrix} e_k^\star \\ \vdots \\ e_{k+i}^\star \\ \hat{u}_{k+i-1} \end{bmatrix} \\ & \text{for } i = 1, \dots, T \end{aligned} \tag{17}$$

where $\mathcal{L}_{\text{train}} : \mathbb{R}^{((T+1)\times N_U)\times((T+1)\times N_U)} \to \mathbb{R}$ is a suitable loss function. In this work we select $\mathcal{L}_{\text{train}}$ as the Mean Absolute Error (MAE) [22].

The peculiar structure of the considered training problem restricts the pool of the functional approximators that can be used to represent the maps f_i and g_i. Instead of developing application-specific solutions, here we exploit the nature of ANNs as directed acyclic computational graphs to build the structure of the constraints in (17) into the topology of the network itself. Choosing neural networks is also convenient due their flexibility to be trained with any differentiable loss function, their theoretical universal approximation power [23], and the availability of well maintained frameworks [24] that can aid the design. This approach also allows us to reduce the whole learning procedure to a standard regression problem, while retaining the capability to easily access the outputs of f_i, g_i as intermediate results of the single sub-components of the network, whose overall scheme is reported in Figure 1.

We stress that the topology of the involved ANNs closely follows the structure highlighted in (16). In particular, all the components f_i are grouped in a single discrete sub-network, while each g_i is a standalone object. In this work

we restrict our analysis to a densely connected feed-forward topology for each subnetwork, meaning that each sub-ANN is comprised by a stack of nonlinear hidden layers connected to a final linear output layer with appropriate output shape.

IV. Simulation Case Studies

In this section, we present the preliminary results obtained by employing the proposed technique on two nonlinear benchmarks, *i.e.,* a discrete-time approximation of the well-known nonlinear two-tanks system[1]

$$\Sigma_{\text{tank}} = \begin{cases} x_{1,k+1} = x_{1,k} - k_1\sqrt{x_{1,k}} + k_2 v_k \\ x_{2,k+1} = x_{2,k} + k_3\sqrt{x_{1,k}} - k_4\sqrt{x_{2,k}} \\ v_k = \begin{cases} 1.5 & \text{if } u_k \geq 1.0 \\ -0.5 & \text{if } u_k \leq -1.0 \\ u_k + 0.5 & \text{otherwise} \end{cases} \\ y_k = x_{2,k} \end{cases} \tag{18}$$

where $x_{i,t}$ denotes the i-th component of $x_t \in \mathbb{R}^2$ and $k_1 = 0.5$, $k_2 = 0.4$, $k_3 = 0.2$, $k_4 = 0.3$, and the following Hammerstein-Wiener model

$$\Sigma_{\text{HW}} = \begin{cases} x_{k+1} = \begin{bmatrix} 0.7555 & 0.25 \\ -0.1991 & 0 \end{bmatrix} x_k + \\ \qquad \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} v_k \\ v_k = \begin{cases} 1 & \text{if } u_k \geq 1.0 \\ -1 & \text{if } u_k \leq -1.0 \\ \text{sign}(u_k)\sqrt{|u_k|} & \text{otherwise} \end{cases} \\ w_k = [0.6993 \;\; -0.4427] x_k \\ y_k = w_k + 5\sin(w_k) \end{cases} \tag{19}$$

Note that both models exhibit a saturation-like nonlinearity on the input.

As reference model $\mathcal{M}$ for Σ_{tank} we considered the following Linear Time-Invariant (LTI) model

$$\mathcal{M}_{\text{tank}}(z) = \frac{0.4z^{-2}}{1 - 0.6z^{-1}}$$

where z^{-1} is the delay operator. For Σ_{HW} we instead use the reference model

$$\mathcal{M}_{\text{HW}}(z) = \frac{0.65z^{-1}}{1 - 0.35z^{-1}}.$$

We stress that both the reference models are linear so that in closing the loop we aim at compensating the plant nonlinearity, while fully exploiting its operating regime. We remark that the reference model is a hyper-parameter of the VRFT framework and its optimal choice is linked to the nature of the target process. We refer the interested reader to [2], [25] for further reading on the topic.

For both these benchmark systems, we collected a training dataset $Z_{\mathcal{N}}$ of 20000 samples, generated by exciting the system with a sequence of variable-amplitude step signals of period 7 steps, with amplitudes drawn from a Gaussian distribution with zero mean and standard deviation $\sigma = 1$ and then clipped to be within the interval $[-1, 1]$. After the experiment, the output signal was empirically normalized

[1]Possible tank overflows are here neglected.

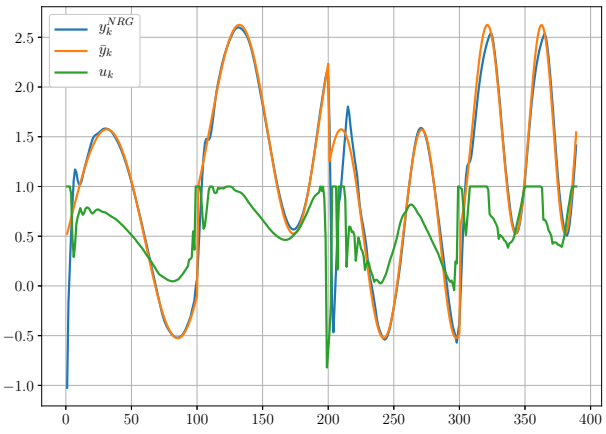

Fig. 2: Closed-loop performance (Σ_{tank}).

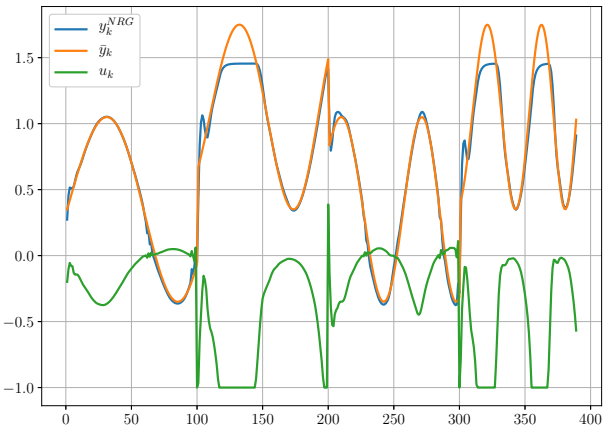

Fig. 3: Closed-loop performance (Σ_{HW}).

using the empirical mean and standard deviation computed from the dataset, before being used for training. After the normalization, all the signals were superimposed with a Gaussian white noise drawn from a distribution with zero mean and standard deviation $\sigma_w = 0.02$.

With these data, we trained a fixed-horizon predictor with $T = 6$, in which every sub-network is composed of 4 Rectified Linear Units (ReLU) [26] hidden layers with 20 neurons followed by a final linear output layer. The state size was set to $M = 7$. The implementation was carried out using Keras [27]. A Lasso [28] penalization was also added to all neurons to regularize the resulting models.

The reference governor was tuned by setting

$$\mathcal{L}_{\text{RG}} = \|\hat{Y}^{\mathcal{M}} - \bar{Y}\|_2^2$$

and by imposing a constraint on the input, namely that $|u_k| \leq 1$, $\forall k$.

The reference trajectory at time k is set by shifting the optimal sequence computed at the previous step $k-1$. We also assumed that a preview of the future references was available and further embedded in our structure, together with the recursive disturbance compensator in (14), with gain α set to 0.1.

Figure 2 shows the closed-loop performance obtained on Σ_{tank} by the proposed approach when tracking a square wave superimposed with a sine-sweep reference signal. We can see how the proposed approach, while not being that effective during the initial steps of the transient, is well equipped to properly track also high-frequency reference signals. Similar considerations can be drawn for the closed-loop performance associated to the system Σ_{HW} shown in Figure 3. We can see how the proposed approach achieves satisfactory performance, showing good capabilities to deal with the strong nonlinearity of the input characteristic of the system when u_k crosses 0. In both Figures 2 and 3 we also appreciate how the disturbance observer does not cause any windup issue, that would severely affect closed-loop performance.

Remark 3 (Computational insights): As a whole, the employed predictor has $\approx 10^4$ parameters. The training procedure was carried out in a few minutes using an Intel Core i5-6200U CPU laptop with 16GB of RAM. The controller, which relied on a general-purpose solver [29], required around 10^{-2} seconds to compute the control action at each time step. Considering the general-purpose nature of the employed libraries, computational requirements can probably be significantly reduced by using more specialized solvers, such as the one in [30], and performance-oriented learning frameworks. ■

V. Conclusions

We presented a novel approach to directly synthesize nonlinear constrained controllers within the model-reference framework. To this end, we exploited a hierarchical structure, with a model-reference inner controller paired with an external reference governor. In designing the neural reference governor, we exploit results from control-oriented system identification to directly learn short-term fixed horizon ANN-based predictors of the virtual internal control law from data. The resulting scheme has shown good performance on two nonlinear benchmarks, ensuring good closed-loop reference tracking performance, and suggesting that it can be competitive with more traditional control design scheme.

These results, albeit preliminary, jointly with the modest computational resources required by the approach, suggest its possible use in embedded fast-sampling applications, which will be the focus of future works together with an in detail comparison with control-aware system-identification and constrained data-driven control schemes.

References

[1] K. J. Keesman, *System identification: an introduction*. Springer-Verlag London, 2011.

[2] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

[3] F. Smarra, A. Jain, T. de Rubeis, D. Ambrosini, A. D'Innocenzo, and R. Mangharam, "Data-driven model predictive control using random forests for building energy optimization and climate control," *Applied Energy*, 2018.

[4] E. Terzi, L. Fagiano, M. Farina, and R. Scattolini, "Learning-based predictive control for linear systems: A unitary approach," *Automatica*, vol. 108, p. 108473, 2019.

[5] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[6] S. Formentin, K. Van Heusden, and A. Karimi, "A comparison of model-based and data-driven controller tuning," *International Journal of Adaptive Control and Signal Processing*, vol. 28, no. 10, pp. 882–897, 2014.

[7] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2019.

[8] S. Formentin, S. Savaresi, and L. Del Re, "Non-iterative direct data-driven controller tuning for multivariable systems: theory and application," *IET control theory & applications*, vol. 6, no. 9, pp. 1250–1257, 2012.

[9] M. C. Campi, A. Lecchini, and S. M. Savaresi, "Virtual reference feedback tuning: a direct method for the design of feedback controllers," *Automatica*, vol. 38, no. 8, pp. 1337–1346, 2002.

[10] S. Formentin, D. Piga, R. Tóth, and S. M. Savaresi, "Direct learning of LPV controllers from data," *Automatica*, vol. 65, pp. 98–110, 2016.

[11] V. Breschi and S. Formentin, "Direct data-driven design of switching controllers," *International Journal of Robust and Nonlinear Control*, 2019.

[12] V. Breschi, D. Masti, S. Formentin, and A. Bemporad, "NAW-NET: neural anti-windup control for saturated nonlinear systems," in *Proc. 59th IEEE Conf. on Decision and Control*, 2020. To appear.

[13] M.-B. Radac, R.-E. Precup, and R.-C. Roman, "Data-driven model reference control of MIMO vertical tank systems with model-free VRFT and Q-learning," *ISA transactions*, vol. 73, pp. 227–238, 2018.

[14] A. Sadeghzadeh and H. Momeni, "Virtual closed loop identification: A new method for low-order H_∞ controller design," *IFAC Proceedings Volumes*, vol. 42, no. 10, pp. 314–319, 2009.

[15] D. Piga, S. Formentin, and A. Bemporad, "Direct data-driven control of constrained systems," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1422–1429, 2017.

[16] A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 415–419, 1998.

[17] M. Farina and L. Piroddi, "Simulation error minimization identification based on multi-stage prediction," *International Journal of Adaptive Control and Signal Processing*, vol. 25, no. 5, pp. 389–406, 2011.

[18] G. Liu and V. Kadirkamanathan, "Predictive control for non-linear systems using neural networks," *International Journal of Control*, vol. 71, no. 6, pp. 1119–1132, 1998.

[19] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Transaction Automatic Control*, vol. 62, no. 12, pp. 6094–6109, 2017.

[20] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 18–33, 2013.

[21] D. Masti, F. Smarra, A. D'Innocenzo, and A. Bemporad, "Learning affine predictors for MPC of nonlinear systems via artificial neural networks," in *Proceedings of the 21st IFAC World Congress*, 2020.

[22] "Mean absolute error," in *Encyclopedia of Machine Learning* (C. Sammut and G. I. Webb, eds.), pp. 652–652, Springer US, 2010.

[23] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, pp. 930–945, May 1993.

[24] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, 2017.

[25] V. Breschi and S. Formentin, "Virtual reference feedback tuning with data-driven reference model selection," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, vol. 120, pp. 37–45, PMLR, 2020.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Int. Conf. on Machine Learning (ICML)*, pp. 807–814, Omnipress, 2010.

[27] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[28] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[29] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001.

[30] A. Bemporad, "A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 525–531, 2017.