Online design of experiments by active learning for system identification of autoregressive models

Kui Xie and Alberto Bemporad

***Abstract*—In this paper, we investigate the use of active-learning (AL) strategies to generate the input excitation signal at runtime for system identification of linear and nonlinear autoregressive models. We adapt various existing AL approaches for static model regression to the dynamic context, coupling them with a Kalman filter to update the model recursively, and also cope with the presence of input and output constraints. The increased efficiency in terms of sample usage of the proposed AL approaches with respect to random excitation is evaluated on a few examples.**

***Index Terms*—Design of experiments, active learning, system identification, extended Kalman filtering**

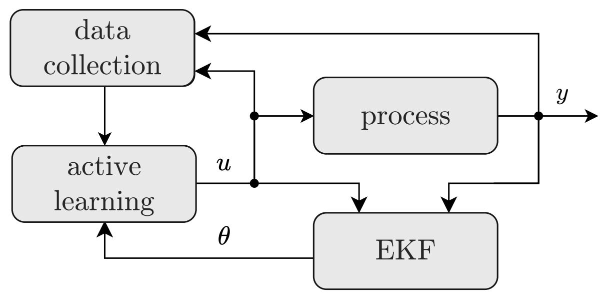

Fig. 1. Online active learning method for system identification.

I. INTRODUCTION

Many system identification approaches exist, both for linear [1] and nonlinear systems [2], [3]. Very often, these methods rely on an existing training dataset for estimating the model parameters that best approximate the system's behavior. No matter how good the chosen model class and advanced the method used to solve the training problem are, ultimately, the quality of the identified model depends on the richness of the information in the training dataset. Relying solely on collecting more data can be costly, may result in excessive redundancy without substantially increasing the information content, and make the optimization problem required to estimate the model parameters more complex, due to the larger number of loss terms in the objective function to minimize [4]–[6].

The problem of optimal design of experiments (DoEs) has been studied for decades, dating back to the 1930s [7]. In the machine learning literature, the related problem of selecting the most informative samples to query for the target value is referred to as *active learning* (AL) [8], [9]. AL strategies aim to reduce the number of required training samples by allowing the training algorithm to select the feature vectors to query. Several AL methods exist in the literature, mostly for classification problems [10], but also contributions exist for regression problems [11]–[17].

The existing AL methods mainly focus on learning *static* models to explain the relationship between feature vectors and targets. These samples can be arbitrarily selected from a dense set of admissible values, a pre-determined discrete pool, or a stream of feature-vector samples [8]. However, actively learning *dynamical* models is more challenging because not all the components of the feature vector can be changed instantaneously. Research on AL for system identification is therefore limited, and it is primarily restricted to specific classes of models such as Gaussian processes [18], [19], and neural-network state-space models [20]. Furthermore, these approaches assume that the state x_k is measurable, while the model is often identified from the input/output data.

In this paper, we extend the AL methods reported in [17] for the regression of static functions to the dynamic context, focusing on learning black-box parametric models in input/output form. Specifically, we consider the problem of identifying autoregressive models, either linear (ARX) or nonlinear (NARX). To update the model parameters as new samples are acquired, we rely on a linear or extended Kalman filter (EKF) [21], [22], depending on how the model is parameterized. A schematic diagram of the proposed strategy is shown in Fig. 1.

The proposed recursive approach employs online optimization, based on the data collected so far, to design the experiment at runtime. As for supervised AL of static models, the developed DoE strategies ensure that the collected data are informative and diverse [8], i.e., respectively, are acquired to minimize modeling errors and explore the state/action space, trying to avoid repetitions. Based on the AL method for regression proposed in [17], we use an acquisition method based on a non-probabilistic measure of the uncertainty associated with output predictions to sample the system where uncertainty is expected to be most significant, and employ inverse-distance weighting (IDW) functions to ensure the exploration of areas not visited before. Recently, a related online AL algorithm has been used for improving the sample efficiency of reinforcement learning (RL) [23] and model predictive coverage control [24].

In this paper, we consider both one-step-ahead AL formu-

The authors are with the IMT School for Advanced Studies Lucca, Piazza San Francesco 19, Lucca, Italy. Email: {kui.xie,alberto.bemporad}@imtlucca.it

lations, based on the uncertainty associated with the following predicted output, and a less myopic multi-step-ahead AL approach based on the uncertainty related to the predicted outputs over a finite horizon, also taking into account the presence of input and output constraints; the latter are treated as soft to avoid excessive conservativeness, especially at early stages when the model is very uncertain. Although the online computation burden of the AL algorithm is limited, especially when the input can be selected from a discrete set (e.g., as in the case of pseudo-random binary signal excitation, where the set has only two elements), we also consider the possibility of running the AL algorithm offline on a digital twin of the system, saving the generated input signal, and then going on the actual process.

The paper is structured as follows. In Section II, we will present the proposed algorithm for NARX models. Numerical experiments on linear and nonlinear autoregressive systems will be reported in Section III. Lastly, we will draw conclusions in Section IV.

II. ONLINE ACTIVE LEARNING OF NARX MODELS

Let us consider the problem of identifying a strictly-causal Nonlinear AutoRegressive eXogenous Model (NARX):

$$\hat{y}_k = f(x_{k-1}, \theta) \tag{1a}$$

$$x_{k-1} = [y'_{k-1} \ \ldots \ y'_{k-n_a} \ u'_{k-1} \ \ldots \ u'_{k-n_b}]' \tag{1b}$$

where $\hat{y}_k \in \mathbb{R}^{n_y}$, $u_k \in \mathbb{R}^{n_u}$, $n_a \geq 0$, $n_b \geq 0$, $\theta \in \mathbb{R}^{n_\theta}$ is the vector of parameters to learn, and $k = -\max\{n_a, n_b\}, -\max\{n_a, n_b\}+1, \ldots, 0, 1, \ldots$ is the sample index. For example, f could be a linear model, $f(x, \theta) = \theta' x$, or a small-scale neural network with weight/bias terms collected in the vector θ. Our goal is to *actively* generate control inputs u_k at runtime, $k = 0, 1, \ldots, N-1$, to efficiently learn the parameter vector θ, solving the posed system identification problem in a sample-efficient manner.

From now on, we assume that all the input and output signals have been properly scaled. For instance, if lower and upper bounds $u_{\min}$, $u_{\max}$, $y_{\min}$, $y_{\max}$ on the possible values of the signals are known, we can scale these signals to the interval $[-1, 1]$ using the scaling function $\sigma : \mathbb{R} \to \mathbb{R}$, $\sigma(\alpha) = \frac{2}{\alpha_{\max} - \alpha_{\min}} \left(\alpha - \frac{\alpha_{\max} + \alpha_{\min}}{2}\right)$, where $\alpha_{\max}$ and $\alpha_{\min}$ are the maximum and minimum values of the signal.

In the sequel, we will denote by θ_k the model parameter vector obtained by training the model with the outputs collected up to time k and inputs up to time $k-1$. We assume that, as in most practical applications, the input u_k is subject to constraints $u_k \in \mathcal{U}$, where $\mathcal{U}$ represents the set of valid inputs. For instance, $\mathcal{U} = \{u \in \mathbb{R}^{n_u} : u_{\min} \leq u \leq u_{\max}\}$ or, in alternative, a finite set $\mathcal{U} = \{u^1, \ldots, u^M\}$, such as $\mathcal{U} = \{-1, 1\}$ in the case of pseudorandom binary sequence excitation.

A. One-step-ahead active learning

Assume that at each time k we have an *acquisition function* $a : \mathbb{R}^{n_x} \to \mathbb{R}$ given to solve a problem of AL for regression [8] (as shown later, in general a changes with k). Ideally, given a new measurement y_k, we would like to choose $x_k = \arg\max_x a(x)$. However, at time k, the only component in x_k that can be freely chosen is the current input u_k, given that all the remaining components involve measured outputs and past inputs. Hence, we restrict the acquisition problem to

$$u_k = \arg\max_{u \in \mathcal{U}} a(x_k(u)) \tag{2a}$$

$$x_k(u) \triangleq [y'_k \ \ldots \ y'_{k-n_a+1} \ u' \ u'_{k-1} \ \ldots \ u'_{k-n_b+1}]' \tag{2b}$$

where $x_k : \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ defines the feature vector corresponding to a given input selection. Note that when the input u_k is chosen from a finite set $\mathcal{U} = \{u^1, \ldots, u^M\}$, problem (2a) can be easily solved by enumeration, analogously to *pool-based* active learning algorithms [8]. The new collected sample (u_k, y_{k+1}) can be immediately used to update the process model θ_{k+1}. In this paper, we use an EKF to update the model parameters or simply a linear Kalman filter in case $f(x_{k-1}, \theta) = \theta' \bar{f}(x_{k-1})$. Generally, the acquisition function $a(x)$ optimized in (2a) depends on θ_k and all past feature-vector/target samples (x_{k-1}, y_k) collected so far.

In [17], the active learning method for regression called ideal was proposed, utilizing inverse distance weighting (IDW) functions. The acquisition function consists of two nonnegative terms: $a(x) = s^2(x) + \delta z(x)$, where the *IDW variance* function, $s^2(x) = \sum_{j=0}^{k} v_j(x) \|y_j - f(x, \theta_k)\|_2^2$, serves as a proxy for the variance of the output y predicted by the model at x, the function $z : \mathbb{R}^{n_x} \to \mathbb{R}$ is an *IDW exploration* function, the function $v_j : \mathbb{R}^{n_x} \to \mathbb{R}$ is an *IDW weight* function, and $\delta \geq 0$ is a tradeoff coefficient between exploitation (of the model θ_k to estimate model uncertainty) and pure exploration (since $z(x) = 0$ at each x_j sampled so far, $\forall j = 0, \ldots, k$).

Besides ideal, we will consider also the alternative incremental AL methods reviewed in [17, Section 3.4]: the greedy method GS_x [25, Algorithm 1], the greedy method iGS [15, Algorithm 3], and the query-by-committee method QBC [13], [26].

B. Initialization

As typically done in most AL algorithms, we start by using *passive* learning to gather N_i initial pairs of input/output samples, $N_i \geq 0$. The simplest way is to use random sampling, i.e., generate $u_0, \ldots, u_{N_i-1}$ randomly, or use the K-means algorithm, cf. [17, Section 3.1].

C. Constraints

To attempt satisfying also *output* constraints, we add a penalty in (2a) on the expected violation of output constraints. For instance, the satisfaction of output constraints

$$y_{\min} \leq y \leq y_{\max} \tag{3}$$

can be encouraged by introducing the penalty term

$$\begin{aligned} p(x) = \rho \sum_{i=1}^{n_y} \{&\max\{\hat{y}_{k+1,i}(x, \theta_k) - y_{\max,i}, 0\}^2 \\ &+ \max\{y_{\min,i} - \hat{y}_{k+1,i}(x, \theta_k), 0\}^2\} \end{aligned} \tag{4}$$

where $\hat{y}_{k+1} = f(x, \theta_k)$ is the next output predicted by the current model with parameter vector θ_k, and ρ is a penalty parameter, $\rho \geq 0$. Then, we solve the following problem

$$u_k = \arg\max_{u \in \mathcal{U}} \; a(x_k(u)) - p(x_k(u)). \tag{5}$$

A drawback of the formulation (5) with the penalty term (4) is that it does not account for the model uncertainty, which might be quite large during the early phase of sampling. To address this issue, we consider the confidence interval proposed in [27] for IDW functions, which is defined as $\hat{y}_{k+1}(x, \theta_k) \pm \kappa_\alpha s(x)$, where $s(x)$ is the square root of the IDW variance function $s^2(x)$ and the scaling factor κ_α is set as the upper α sample quantile of $|CV_i|/s_{-(i-1)}(x_{i-1})$, $i = 0, \ldots, k$, where α is a constant, typically set to 90%, $CV_i = y_i - \hat{y}_i(x_{i-1}, \theta_k)$ is the cross-validation error at x_{i-1}, $s^2_{-i}(x_i) = \sum_{j=0,\, j \neq i}^{k} v_{j(i)}(x_i)(y_j - \hat{y}_{i+1}(x_i, \theta_k))$, $v_{j(i)}(x_i) = w_j(x_i)/\sum_{l=0,\, l\neq i}^{k} w_l(x_i)$, and $w_l(x)$ is the weighting function. To prevent over-shrinking the constraint set in (5), we impose a limit on the quantity $\kappa_\alpha s(x) \leq \beta(y_{\max} - y_{\min})$, where, in this case, we set $\beta = \frac{1}{3}$. Finally, in (5) we take into account uncertainty by replacing $p(x)$ with

$$\begin{aligned} p(x) \;\; = \;\; & \rho \sum_{i=1}^{n_y} \{\max\{\hat{y}_{k+1,i}(x, \theta_k) - y_{\max,i} + \kappa_\alpha s(x), 0\}^2 \\ & + \max\{y_{\min,i} - \hat{y}_{k+1,i}(x, \theta_k) + \kappa_\alpha s(x), 0\}^2\}. \end{aligned} \tag{6}$$

D. Alternative active-learning methods

We review three different AL methods for regression, alternative to ideal, slightly adapted here to generate input signals for system identification. We exclude the iRDM method [16] as it cannot be used for recursive model learning.

1) Greedy method GS_x*:* The non-model-based method GS_x [25, Algorithm 1] selects the next sample x_k by maximizing the minimum distance from existing samples. In analogy with (5), we extend GS_x based on the acquisition problem $u_k = \arg\max_{u\in\mathcal{U}} d_x(x_k(u)) - p(x_k(u))$, where $d_x(x) = \min_{i=0}^{k} \|x - x_i\|_2^2$ is the minimum distance from existing (scaled) samples.

2) Greedy method iGS: Given the predictor $f(x_k(u), \theta_k)$ trained on the available samples, the greedy sampling technique iGS [15, Algorithm 3] can be used to select the next input u_k by solving the acquisition problem $u_k = \arg\max_{u\in\mathcal{U}} \; d_x(x_k(u)) d_y(x_k(u)) - p(x_k(u))$, where $d_x(x)$ is the same as in the GS_x method and $d_y(x) = \min_{i=0}^{k} \|\hat{y}_{k+1}(x, \theta_k) - y_i\|_2^2$ is the predicted minimum distance in the output space from existing output samples.

3) Query-by-Committee method QBC: The Query-by-Committee (QBC) method for regression [13], [26] utilizes K_{QBC} different predictors θ_k^j, $j = 1, \ldots, K_{QBC}$. In AL of static models, predictors are typically obtained by bootstrapping the acquired dataset. In contrast, as we acquire the samples online, we create K_{QBC} different models by running K_{QBC} (extended) Kalman filters in parallel and, at each time step, only update $K_{QBC} - 1$ models after acquiring the new sample y_k. This adaptation of QBC aims to select the input u_k that maximizes the variance of the estimated output prediction $\hat{y}_{k+1}(x_k(u), \theta_k^j)$: $u_k = \arg\max_{u\in\mathcal{U}} \sum_{j=1}^{K_{QBC}} \left\| \hat{y}_{k+1}(x, \theta_k^j) - \frac{\sum_{h=1}^{K_{QBC}} \hat{y}_{k+1}(x, \theta_k^h)}{K_{QBC}} \right\|_2^2 - p(x)$, with $x = x_k(u)$.

Algorithm 1 Online design of experiments for system identification of autoregressive models using active learning and inverse-distance based exploration (ideal-sysid).

Input: Set $\mathcal{U}$ of admissible inputs, number N_i of passively-sampled inputs, length N of the experiment to design, exploration hyperparameter $\delta \geq 0$, number $L \geq 1$ of prediction steps, possible upper and lower bounds $y_{\max}$, $y_{\min}$, penalty parameter $\rho \geq 0$ on output constraint violations.

1. Generate N_i samples $u_0, \ldots, u_{N_i - 1}$ by passive learning (e.g., random sampling)
2. Excite the system and collect $y_0, \ldots, y_{N_i-1}$;
3. Estimate $\theta_{N_i - 1}$;
4. **For** $k = N_i, \ldots, N$ **do**:

4.1. measure y_k;

4.2. update θ_k by (extended) Kalman filtering;

4.3. **If** $k < N$, get u_k by solving problem (7), with penalty p as in (4) or (6) to handle possible output constraints;

5. **End**.

Output: Estimated parameter vector θ_N; input excitation $u_0, \ldots, u_{N-1}$.

E. Multi-step prediction

So far we have been concerned only with the one-step-ahead prediction $\hat{y}_{k+1}$ and, consequently, with the acquisition of the new control input u_k. To circumvent such a possible myopic view, we can take a *predictive* approach and extend the formulation to optimize a finite sequence of future inputs $U_k = [u'_k \; u'_{k+1} \; \ldots \; u'_{k+L-1}]'$, where $L \geq 1$ is the desired prediction horizon. In this case, the predicted regressor vector $\hat{x}_{k+j}$ entering the acquisition function a contains either outputs $\hat{y}_{k+h}$ predicted by model θ_k ($h \geq 1$) or measured outputs y_{k+h} ($h \leq 0$), and either current and future inputs $u_{k+h} = u^h$ ($h \geq 0$) or past inputs u_{k+h} (for $h < 0$). We will denote the predicted regressor as $\hat{x}_{k+j} \triangleq x_{k+j}(U)$ to recall that it depends on the first $j+1$ inputs in U. Moreover, since future measured outputs y_{k+j+1} are not available, replacing them by surrogates $\hat{y}_{k+j+1} = f(\hat{x}_{k+j}, \theta_k)$ cause the IDW variance $s^2(\hat{x}_{k+j}) = 0$, and thus $a(\hat{x}_{k+j}) = z(\hat{x}_{k+j})$, $\forall j > 0$. Therefore, the multi-step-ahead active learning problem can be formulated as follows:

$$U_k = \arg\max_{U\in\mathcal{U}^L} s^2(x_k(U)) + \delta \sum_{j=0}^{L-1} z(x_{k+j}(U)) - p(x_{k+j}(U)) \tag{7}$$

where $\mathcal{U}^L \triangleq \mathcal{U} \times \ldots \times \mathcal{U}$. Note that (7) coincides with (5) when $L = 1$, as $x_k(U) = x_k(u)$.

Based on the receding-horizon mechanism used in model predictive control, after solving (7), only the current input u_k is applied to excite the process, while the remaining moves u_{k+j} are discarded, for all $j = 1, \ldots, L-1$. Then, after acquiring the new measurement y_{k+1} and updating the model to get the new parameter vector θ_{k+1}, problem (7) is solved again to get U_{k+1}, and so on.

The overall algorithm for online input design for system identification based on the ideal active learning approach, denoted by ideal-sysid, is summarized in Algorithm 1.

F. Numerical complexity

In analyzing the complexity of Algorithm 1, we assume that the model parameter vector θ is estimated by EKF, whose complexity is $O(n_\theta^2)$ per step. Therefore, the recursive training has a complexity of $O(n_\theta^2 N)$, plus the cost of evaluating (1) and, if the model f in (1) is not linear with respect to θ, its Jacobian with respect to θ (N times) for the EKF updates. In the QBC case, $(K_{QBC} - 1)N$, more EKF-related computations are required, assuming that the different EKFs are run in parallel also during the random-sampling phase. In the case of pool-based sampling ($\mathcal{U} = \{u^1, \dots, u^M\}$), the complexity of solving problem (7) requires $(N - N_i + 1)M^L$ evaluations of the acquisition function $a(x)$, or $(N - N_i + 1)M$ evaluations for $L = 1$.

III. NUMERICAL EXPERIMENTS

We test Algorithm 1 (ideal-sysid) for the identification of linear and nonlinear autoregressive models, and compare it to passive learning and variants of Algorithm 1 obtained by replacing ideal with one of the alternative AL methods reviewed in Section II-D. We use random sampling to generate the initial N_i samples for all AL methods. The model parameter vector $\theta_k \in \mathbb{R}^{n_\theta}$ is recursively estimated by (extended) Kalman filtering with covariance matrices $P_k \in \mathbb{R}^{n_\theta \times n_\theta}$, $Q \in \mathbb{R}^{n_\theta \times n_\theta}$ and $R \in \mathbb{R}^{n_y \times n_y}$, as described in [22]. We set $P_0 = \frac{1}{10^{-3}N} I_{n_\theta}$, $Q = 0$, and $R = 10^{-2} I_{n_y}$. We only report results based on one-step prediction due to space limitations. To quantify the overall quality of prediction on the training and test datasets, we measure the root-mean-square error (RMSE), $RMSE = \sqrt{\frac{1}{N_{\max}} \sum_{k=1}^{N_{\max}} (y_k - \hat{y}_k(x_{k-1}, \theta_k))^2}$, where $N_{\max}$ is the total number of samples in the set ($N_{\max} = N$ for the training set and $N_{\max} = N_{\text{test}}$ for the test set). All computations were performed in MATLAB R2023b.

A. Linear ARX example

Firstly, we test the proposed AL approaches for learning a linear ARX model with $n_a = 3$, $n_b = 3$, $n_u = 1$, and $n_y = 1$. We generate noisy synthetic data by simulating the following system

$$\begin{aligned} y_k = {} & \theta_{a,1} y_{k-1} + \theta_{a,2} y_{k-2} + \theta_{a,3} y_{k-3} \\ & + \theta_{b,1} u_{k-1} + \theta_{b,2} u_{k-2} + \theta_{b,3} u_{k-3} + \alpha \eta_k \end{aligned} \tag{8}$$

from zero initial condition, where $\theta_a = [\frac{9}{10}\ -\frac{3}{10}\ \frac{1}{3}]'$, $\theta_b = [\frac{1}{3}\ -\frac{1}{5}\ \frac{1}{15}]'$, $\eta_k \sim \mathcal{N}(0, 1)$, and $\alpha = 0.005$. The overall vector of model parameters is therefore $\theta = [\theta_a'\ \theta_b']'$, that we wish to reconstruct. We use pool-based sampling by letting $\mathcal{U} = \{-1, -1 + \frac{1}{M}, \dots, -1 + \frac{2M-1}{M}, 1\}$ with $M = 20$.

The prediction model is the linear model $f(x_{k-1}, \theta_k) = \theta_k' x_{k-1}$, where the parameter vector θ_k is updated recursively by a linear Kalman filter after measuring each new sample y_k.

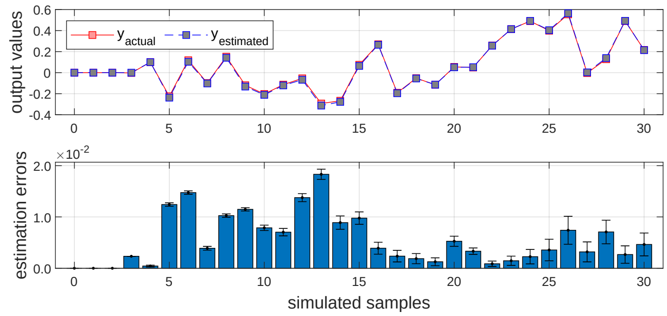

Fig. 2. ideal-sysid applied to system (8). Upper plot: measurements y_k (red squares) and predictions $\hat{y}_k = \theta_N' x_{k-1}$ (purple squares). Lower plot: average estimation errors and vertical lines denote mean absolute deviation ($L = 1$).

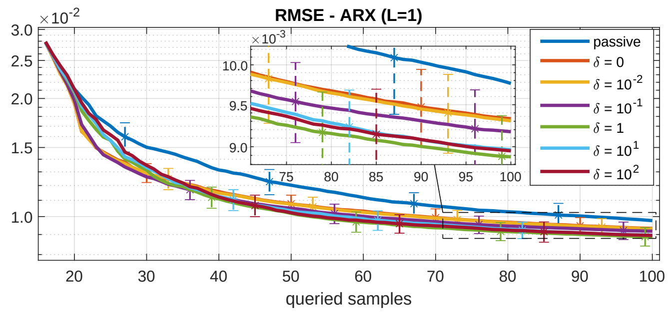

Fig. 3. Effects of the exploration parameter δ in ideal-sysid and comparison with passive learning. Vertical lines denote mean absolute deviation ($L = 1$).

We set the ideal-sysid parameters $\delta = 1$, $N_i = 16$, $N = 100$, $\rho = 0$ (no output constraints), $L = 1$, and apply Algorithm 1. We also generate a test dataset by running (8) from zero initial condition for $N_{\text{test}} = 30$ steps. Fig. 2 compares the one-step-ahead estimates $\hat{y}_k = \theta_N' x_{k-1}$ with respect to the measured outputs y_k on test data after running Algorithm 1 for N steps (upper plot) and the lower plot shows the average estimation errors over 200 runs with the same test dataset. To adequately evaluate results, we enlarge the test set size and set $N_{\text{test}} = 300$ in all remaining experiments.

Fig. 3 shows how the median RMSE over 200 runs decreases with the number of acquired samples N for different values of δ in the ideal-sysid method, with a limited sensitivity with respect to the hyper-parameter δ. With $\delta \ll 1$, pure exploitation performs better at the early stage, while only limited exploration occurs later. We will set $\delta = 1$ in the remaining tests. The figure clearly shows that ideal-sysid outperforms passive learning (random sampling of $u_k \in \mathcal{U}$, in this case): the RMSE decreases faster with the number of samples acquired.

Next, we compare the performance of ideal-sysid when the AL method ideal is replaced by GS$_x$, iGS, or QBC in Algorithm 1. The median RMSE and its mean absolute deviation over 200 runs are depicted in Fig. 4 (upper plot). ideal-sysid outperforms GS$_x$, iGS, and passive learning, and is comparable to QBC. For all the considered AL methods, the RMSE values are identical for $k \le N_i - 1$, indicating that they share the same initial randomly generated samples,

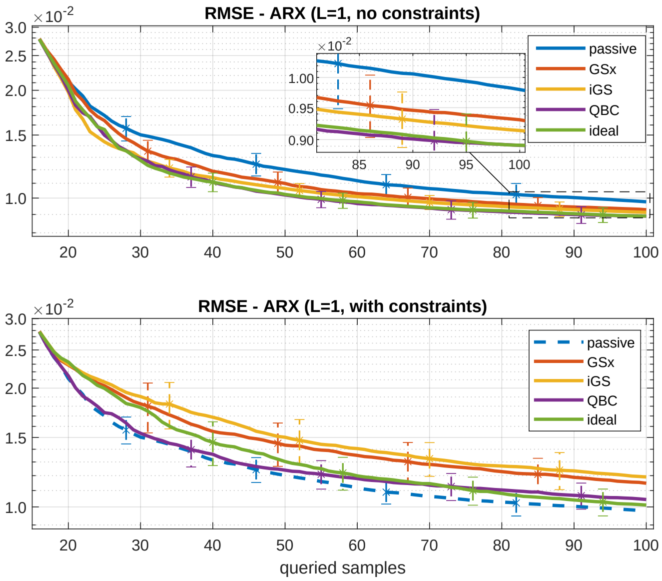

Fig. 4. AL problem (8) predicted with an ARX model, median RMSE without: no constraints (upper plot) and with constraints (9) (lower plot). Constraints are ignored during passive sampling. Vertical lines denote mean absolute deviation ($L = 1$, $\rho = 10^6$).

resulting in the same parameters learned by the Kalman filter. The initial samples are also used to initialize the Kalman filters employed by QBC to estimate different models $\theta_k^j \in \mathbb{R}^{n_\theta}$ with corresponding covariance matrices $P_k^j \in \mathbb{R}^{n_\theta \times n_\theta}$, $j = 1, \ldots, K_{QBC}$, as described in Section II-D3.

To excite the system to learn the parameters under the output constraints

$$-0.25 \leq y_k \leq 0.25 \tag{9}$$

we set $\rho = 10^6$ in Algorithm 1 and repeat the online input design procedure. The obtained RMSE results are presented in Fig. 4 (lower plot). It is evident that ideal-sysid consistently outperforms GS_x and iGS, and performs better than QBC after 72 samples are collected. The good performance of the passive method is because output constraints are ignored and violated. Instead, as illustrated by Fig. 5 (upper plot, purple dots), introducing the constraint-violation penalty p defined in (4) is quite effective in preventing the outputs from exceeding the bounds in (9).

Note that as model accuracy improves, the number of constraint violations (red dots) also decreases. The initial $N_i = 6$ samples (grey dots) are generated using passive learning. The figure also shows the output samples (yellow dots) obtained without imposing a constraint violation penalty ($\rho = 0$).

Fig. 5 (lower plot) illustrates the output samples (purple dots) generated by ideal-sysid under the scenario where the constraints are tightened, as defined in (6) along with the adjusted constraint values (orange dashed lines) and predictions $\hat{y}_k$ (green dots).

To evaluate the robustness of the AL methods against mismatches between the chosen model class and the system, we consider data generated by the NARX system

$$y_k = \theta' x_{k-1} + a y_{k-1}^3 + b y_{k-2}^2 + \alpha \eta_k \tag{10}$$

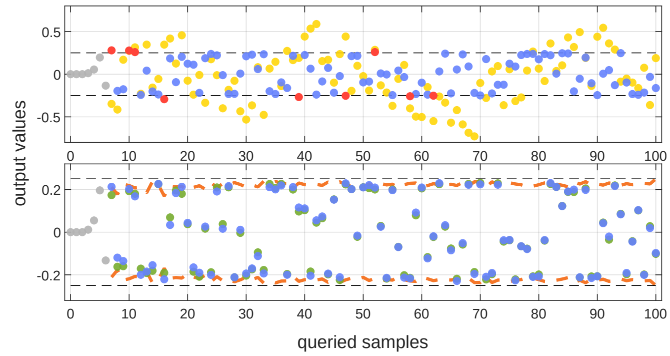

Fig. 5. Initial samples (grey dots). Upper plot: actual system outputs without penalties ($\rho = 0$, yellow dots) and with constraint violation penalties in (4) ($\rho = 10^6$, purple dots and red dots, violating the constraints). Lower plot: actual system outputs (purple dots) with shrunk constraints (orange dashed lines in (6), $\rho = 10^6$) and predictions (green dots). (ideal-sysid, $L = 1$).

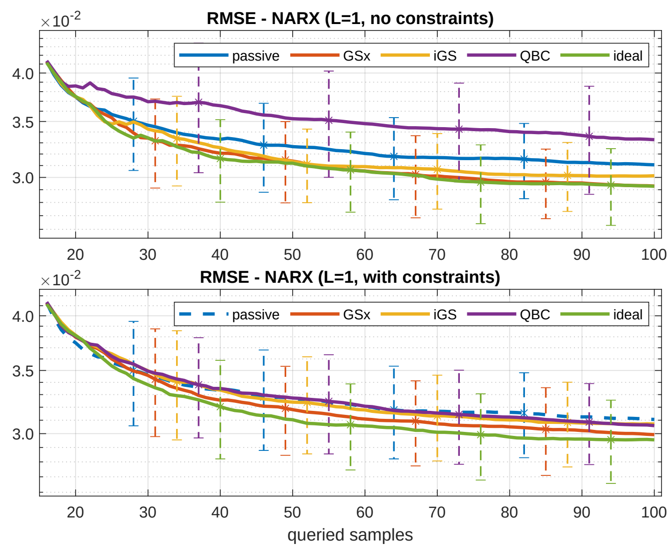

Fig. 6. AL problem (10) predicted with an ARX model, median RMSE without (upper plot) and with constraints (9) (lower plot). Vertical lines denote mean absolute deviation.

where θ, α and η_k are the same as in the ARX example (8), $a = -0.1$, and $b = 0.1$. We still apply the original ARX model, $\hat{y}_k = \theta_k' x_{k-1}$, to predict the output y_k of the nonlinear system (10) and use a linear Kalman filter to estimate θ_k. Fig. 6 (upper plot) shows that, without constraints, ideal-sysid is superior to passive, iGS, and QBC, and is comparable to GS_x; in the constrained case (lower plot), except for passive that can collect more informative samples by violating the constraints, ideal-sysid still provides the best performance. The suboptimal performance of QBC and iGS might be due to the persistence of the estimated model uncertainty due to the inherent model bias.

B. *NARX neural network example*

Finally, we demonstrate the effectiveness of the proposed AL approach for actively learning a NARX neural network model $y_k = W_{2,k}^T \left(\sigma \left(W_{1,k}^T x_{k-1} + b_{1,k} \right) \right) + b_{2,k}$, where

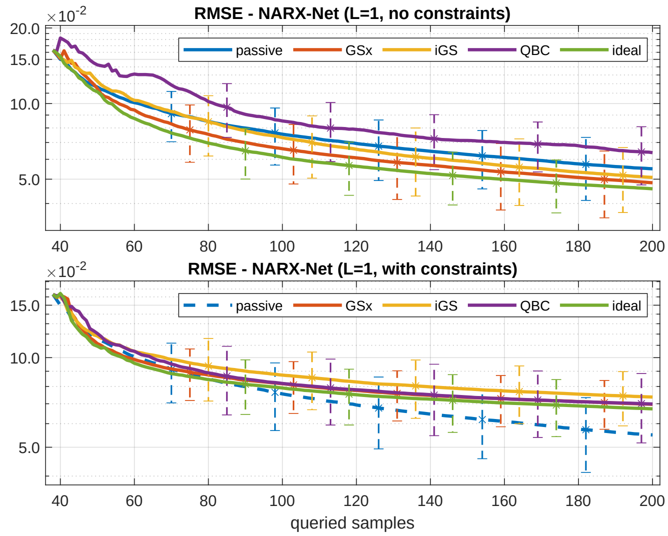

Fig. 7. AL problem (11) predicted with a NARX model, median RMSE without (upper plot) and with constraints (9) imposed in all methods but passive sampling (lower plot). Vertical lines denote mean absolute deviation.

$n_a = 2$, $n_b = 2$, $n_u = 1$, $n_y = 1$ and $\sigma(x) = \frac{1}{1+e^{-x}}$, from noisy synthetic data generated by a system

$$y_k = W_2^T \left(\sigma\left(W_1^T x_{k-1} + b_1\right)\right) + b_2 + \alpha\eta_k, \tag{11}$$

with $x_{k-1} = [y_{k-1},\, y_{k-2},\, u_{k-1},\, u_{k-2}]^T$, parameters

$$W_1 = \begin{bmatrix} -0.55 & 0.58 & -1.50 & 0.16 \\ 0.17 & -0.85 & 0.87 & -1.9 \\ -0.19 & 0.80 & -0.24 & -0.67 \end{bmatrix}^T \quad b_1 = \begin{bmatrix} -1.8 \\ -1.0 \\ -0.41 \end{bmatrix}$$
$$W_2 = \begin{bmatrix} 1.4 & -1.3 & -0.29 \end{bmatrix}^T \quad b_2 = 1.2,$$

η_k and α are the same in (8). The overall vector of model parameters is $\theta_k = \left[\mathrm{vec}(W_{1,k}^T)^T \; b_{1,k}^T \; \mathrm{vec}(W_{2,k}^T)^T \; b_{2,k}^T\right]^T$ to be estimated by EKF. Fig. 7 shows that ideal-sysid is superior to GS$_x$, iGS, and QBC in both cases. Note again in the lower plot that the RMSE of passive is the lowest in the constrained case, due to its ability to get more informative data by violating the output constraints

IV. CONCLUSIONS

This paper has introduced different active learning methods for online experiment design tailored to the identification of autoregressive models. The proposed technique ideal-sysid stands out as promising due to its consistent behavior across different scenarios, including both linear and nonlinear models, as well as those with or without soft output constraints. Future work will focus on extending the proposed approach to identifying linear and nonlinear *state-space* models from input/output data, which poses additional challenges due to the presence of hidden states.

REFERENCES

[1] L. Ljung, *System Identification : Theory for the User*. Prentice Hall, 2 ed., 1999.

[2] L. Ljung, C. Andersson, K. Tiels, and T. Schön, "Deep learning and system identification," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.

[3] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. Ribeiro, and T. Schön, "Deep networks for system identification: a survey," *arXiv preprint 2301.12832*, 2023.

[4] I. M. Y. Mareels and M. Gevers, "Persistency of excitation criteria for linear, multivariable, time-varying systems," *Mathematics of Control, Signals and Systems*, vol. 1, no. 3, pp. 203–226, 1988.

[5] T.-C. Lee, Y. Tan, and D. Nešić, "Stability and persistent excitation in signal sets," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1188–1203, 2015.

[6] C. R. Rojas, J. S. Welsh, G. C. Goodwin, and A. Feuer, "Robust optimal experiment design for system identification," *Automatica*, vol. 43, no. 6, pp. 993–1008, 2007.

[7] R. Fisher, *The Design of Experiments*. Edinburgh: Oliver & Boyd, 1935.

[8] B. Settles, *Active Learning*. Synthesis lectures on artificial intelligence and machine learning, Morgan & Claypool, 2012.

[9] P. Kumar and A. Gupta, "Active learning query strategies for classification, regression, and clustering: a survey," *Journal of Computer Science and Technology*, vol. 35, no. 4, pp. 913–945, 2020.

[10] C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. Yu, "Active learning: A survey," in *Data Classification: Algorithms and Applications* (C. Aggarwal, ed.), ch. 22, pp. 572–605, Chapman and Hall/CRC Press, 2014.

[11] D. MacKay, "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.

[12] D. Cohn, Z. Ghahramani, and M. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.

[13] R. Burbidge, J. Rowland, and R. King, "Active learning for regression based on query by committee," in *Int. Conf. on Intelligent Data Engineering and Automated Learning*, pp. 209–218, 2007.

[14] W. Cai, Y. Zhang, and J. Zhou, "Maximizing expected model change for active learning in regression," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 51–60, 2013.

[15] D. Wu, C.-T. Lin, and J. Huang, "Active learning for regression using greedy sampling," *Information Sciences*, vol. 474, pp. 90–105, 2019.

[16] Z. Liu, X. Jiang, H. Luo, W. Fang, J. Liu, and D. Wu, "Pool-based unsupervised active learning for regression using iterative representativeness-diversity maximization (iRDM)," *Pattern Recognition Letters*, vol. 142, pp. 11–19, 2021.

[17] A. Bemporad, "Active learning for regression by inverse distance weighting," *Information Sciences*, vol. 626, pp. 275–292, May 2023. Code availble at http://cse.lab.imtlucca.it/~bemporad/ideal.

[18] S. Tang, K. Fujimoto, and I. Maruta, "Actively learning Gaussian process dynamical systems through global and local explorations," *IEEE Access*, vol. 10, pp. 24215–24231, 2022.

[19] H. S. A. Yu, C. Zimmer, and D. Nguyen-Tuong, "Batch active learning in gaussian process regression using derivatives," *arXiv preprint arXiv:2408.01861*, 2024.

[20] E. Lundby, A. Rasheed, I. Halvorsen, D. Reinhardt, S. Gros, and J. Gravdahl, "Deep active learning for nonlinear system identification," *arXiv preprint arXiv:2302.12667*, 2023.

[21] L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, no. 1, pp. 36–50, 1979.

[22] A. Bemporad, "Recurrent neural network training with convex loss and regularization functions by extended Kalman filtering," *IEEE Transactions on Automatic Control*, vol. 68, no. 9, pp. 5661–5668, 2023.

[23] K. Seel, A. Bemporad, S. Gros, and J. Gravdahl, "Variance-based exploration for learning model predictive control," *IEEE Access*, vol. 11, pp. 60724–60736, 2023.

[24] R. Rickenbach, J. Köhler, A. Scampicchio, M. N. Zeilinger, and A. Carron, "Active learning-based model predictive coverage control," *IEEE Transactions on Automatic Control*, pp. 1–16, 2024.

[25] H. Yu and S. Kim, "Passive sampling for regression," in *IEEE Int. Conf. on Data Mining*, pp. 1151–1156, 2010.

[26] T. RayChaudhuri and L. Hamey, "Minimisation of data collection by active learning," in *Proc. Int. Conf. on Neural Networks*, vol. 3, pp. 1338–1341, 1995.

[27] V. R. Joseph and L. Kang, "Regression-based inverse distance weighting with applications to computer experiments," *Technometrics*, vol. 53, no. 3, pp. 254–265, 2011.