Neural Processing Letters (2023) 55:4257–4289
https://doi.org/10.1007/s11063-022-11040-z

An Outlier-Robust Growing Local Model Network for Recursive System Identification

Jéssyca A. Bessa[1] · **Guilherme A. Barreto**[1] · **Ajalmar R. Rocha-Neto**[1]

Accepted: 15 September 2022 / Published online: 24 September 2022

Abstract
In this paper, we develop a self-growing variant of the local model network (LMN) for recursive dynamical system identification. The proposed model has the following features: growing online structure, fast recursive updating rules, better memory use (no storage of covariance matrices is required), and outlier-robustness. In this regard, efficiency in performance and simplicity of implementation are the essential qualities of the proposed approach. The proposed growing version of the LMN model results from a synergistic amalgamation of two simple but powerful ideas. For this purpose, we adapt the neuron insertion strategy of the resource-allocating network to LMN model, and replaces the standard OLS rule for parameter estimation with outlier-robust recursive rules. A comprehensive evaluation involving three SISO and one MIMO benchmarking data sets corroborates the proposed approach's superior predictive performance in outlier-contaminated scenarios compared to fixed-size LMN-based models.

Keywords Local model network · Growing models · System identification · Least mean estimate

1 Introduction

A dynamical system is a natural or artificial process whose inputs and outputs present temporal (or causal) dependence. By using observed data in the form of input-output time series, instead of the equations of Physics, the goal of system identification is to learn that temporal dependence, or dynamics, between input and output variables [32]. The identified model can then be useful to describe, control, and simulate the systems of interest [50]. For this

✉ Jéssyca A. Bessa
bessa.jessyca@ifce.edu.br

Guilherme A. Barreto
gbarreto@ufc.br

Ajalmar R. Rocha-Neto
ajalmar@ifce.edu.br

[1] Graduate Program in Teleinformatics Engineering, Center of Technology, Federal University of Ceará, Campus of Pici, Fortaleza, Ceará, Brazil

reason, system identification is an essential component of modern data-driven control and automation engineering [9].

The field of computational intelligence has contributed with several methods to system identification, starting probably with the Takagi-Sugeno (TS) fuzzy model [54] introduced in the mid-1980s, and continuing with neural network-based approaches, such as the well-known multilayer perceptron (MLP) network [43], the radial basis functions (RBF) [11] network, and the self-organizing map (SOM) [2, 3, 41, 56, 61]. MLP-based models implement a global approach to system identification since the whole training set is used for building a single predictive model. Alternatively, the TS-, RBF-, and SOM-based models implement a local model approach, in the sense that multiple localized submodels are built, each submodel using a partition of the data [13, 15]. The outputs of the multiple submodels can be either aggregated or used individually to predict the model's output.

Of particular interest to this paper is the local model network (LMN) approach to system identification [4, 5, 14, 42, 47]. The LMN model was introduced by [25] as a generalization of the RBF network so that to each hidden unit is associated with a linear regression submodel (and, hence, to a vector of coefficients). The idea of using local regression models within an RBF network was suggested in several works around the same time, for example, in the related field of time series prediction [26, 52]. One positive consequence of using the LMN model is that it requires less hidden units than the RBF network due to a better interpolation property resulting from the linear regression submodels.

One drawback of the local approach is that the number of submodels is a hyperparameter, i.e., it has to be specified before training effectively takes place. This limitation is solved using growing models in which submodels are progressively inserted until a certain error-based performance requisite (e.g., minimum acceptable accuracy) is reached. This approach is also suitable to handle nonstationarity since the system dynamics may change over time, and the model may be required to adapt its structure to deal with the new demands of the task. The growing model approach to system identification and related tasks, such as time series prediction, anomaly detection, fault diagnosis, and control, has received contributions from different computational intelligence areas. For example, a TS-based growing model, in which new fuzzy rules are inserted, is proposed in [1]. SOM-based growing models are introduced in [31] and [27], while an RBF-based growing model is proposed in [20]. Finally, an LMN-based growing model is recently introduced in [8] and applied to time series prediction.

In addition to a growing structure, a desirable requisite for a predictive model is robustness to outliers, here understood in a broad sense as resilience to anomalous measurements. Such abnormal samples are common in real-world applications and can be removed to some extent before model building in offline settings. In online system identification scenarios, outliers should be suitably handled by robust parameter estimation methods [19]. In this sense, standard parameter estimation methods used in system identification, such as the ordinary least squares (OLS), the least mean squares (LMS), and the recursive least squares (RLS), are optimal under the assumption of Gaussianity of the errors [30, 63]. However, such an assumption is unrealistic in outlier-rich applications, and the blind application of non-robust parameter estimation methods often leads to biased models. Adequate handling of outliers by the previously mentioned works on system identification is addressed only in [5] and [8], who introduced a fixed-size and a growing robust LMN model, respectively.

In the current paper, we aim at extending the scope of application of the local model network (LMN) to recursive dynamical system identification by introducing an outlier-robust variant capable of inserting new local models when necessary. The proposed model has the following features: (*i*) growing online structure for handling process nonstationarity; (*ii*) fast recursive updating rule for rapid tracking of changes in system dynamics; (*iii*) better

memory use since no storage of covariance matrices is necessary, as required in RLS-like learning rules; and (iv) outlier-robustness for adequate treatment of abnormal input-output samples. To achieve these requisites, the proposed growing outlier-robust version of the LMN model is built upon the neuron insertion strategy of the resource-allocating network (RAN) [45] and the outlier-robustness concepts originating from the M-estimation framework [19]. The proposed approach is evaluated in three SISO and one MIMO benchmarking data sets for recursive system identification. The achieved results consistently reveal the superior performance of the proposed model over alternative models.

The remainder of the paper is organized as follows. The fixed-size LMN model and its application to system identification is described in Sect. 2. The proposed growing LMN-based approach is developed in Sect. 4, including all the necessary math for its correct understanding, while its outlier-robust variant is presented in Sect. 5. Extensions of the proposed growing models to the identification of MIMO systems are developed in Sect. 6. Experiments are described in Sect. 7 with discussion of the reported results. The paper is concluded in Sect. 8.

2 The Fixed-Size LMN-NARX Model

2.1 The NARX Model

Firstly, let us define the type of input-ouput dynamic model we are interested in. In this regard, the dynamics of SISO systems are assumed to be adequately described by a nonlinear autoregressive model with exogenous inputs (NARX) [32]

$$y(t) = G\left(y(t-1),\ \ldots\ , y(t-q); u(t-1),\ \ldots, u(t-p)\right), \tag{1}$$

where p and q denote the input and output memory orders, respectively. The target nonlinear function $G(\cdot) : \mathbb{R}^{p+q} \to \mathbb{R}$ is assumed to be unknown. Observed data in the form of input-output time series are used to build an approximating model $\hat{G}(\cdot)$ for the target function $G(\cdot)$ by means of any universal function approximation model, such as neural networks and fuzzy-based approaches. Whatever the choice, the n-th input regression vector $\mathbf{x}(t) \in \mathbb{R}^{p+q}$ is mounted by concatenating q past observed outputs $y \in \mathbb{R}$ and p past inputs $u \in \mathbb{R}$ into a regression vector

$$\mathbf{x}(t) = [y(t-1)\ \cdots\ y(t-q)\ |\ u(t-1)\ \cdots\ u(t-p)]^T. \tag{2}$$

When the LMN model is used to learn the dynamics of a given input-output system using the input vectors as defined in Eq. (2), one gets a LMN-based NARX model, or LMN-NARX, for short.

As already mentioned, we aim at developing an outlier-robust variant of the LMN-NARX model suitable for online system identification. The standard LMN-NARX model has been mostly used for offline system identification, a task in which the user can considerable time to test network configurations with different numbers of hidden units [47]. Furthermore, the model structure and parameters are kept fixed and unchangeable as time goes by. In online system identification, adaptability is a crucial aspect of the model. As such, one expects a predictive model that change its archictecture and adjust parameters accordingly as time goes by, being capable of dealing with changes in the system dynamics and outliers naturally. For this purpose, the number of hidden units of the LMN model should not be fixed in advance but instead allowed to grow so that new hidden units could be inserted continuously. Previous

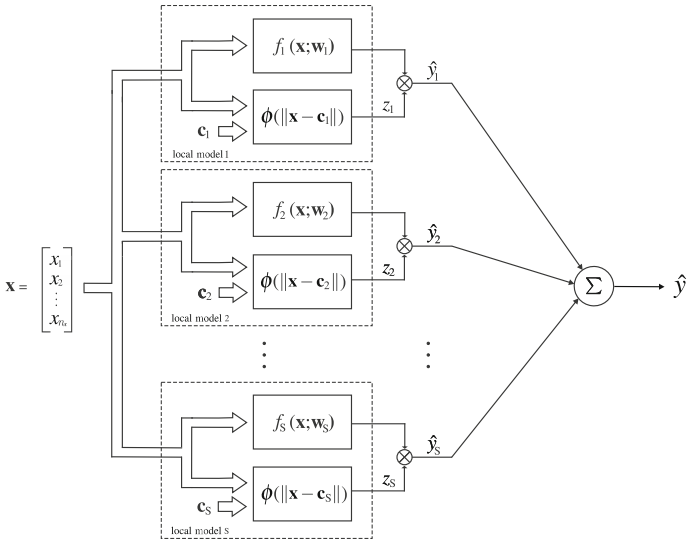

Fig. 1 The architecture of the LMN network

and new parameters (those belonging to inserted hidden units) are modified recursively to correctly track the system dynamics instead of using batch methods such as the OLS method.

2.2 The LMN Model

The LMN model (see Fig. 1) is better understood as an extension of the RBFN model, a classical feedforward neural network architecture with a single hidden layer of neurons [11, 40, 46, 60]. These hidden neurons, usually called radial basis function units, have nonlinear activation functions, while output neurons use linear ones. A general expression for the output of the j-th basis function is given by

$$z_j = \phi\left(d(\mathbf{x}, \mathbf{c}_j)\right), \tag{3}$$

subject to $\sum_{j=1}^{S} z_j = 1$, where $\mathbf{x} \in \mathbb{R}^{p+q}$ is the input vector, $d(\mathbf{x}, \mathbf{c}_j)$ is a distance function, with $\mathbf{c}_j$ denoting the center of the j-th basis function, $j = 1, \ldots, S$. The Euclidean distance function, $d(\mathbf{x}, \mathbf{c}_j) = \|\mathbf{x} - \mathbf{c}_j\|$, and the Gaussian basis function $\phi(u) = \exp(-0.5u^2/\sigma^2)$ are probably the most common choices, with σ denoting the radius (or width) of the Gaussian basis function.

The design of the RBF network basically involves the specification of the number S of basis functions, determination of their parameters $(\mathbf{c}_j, \sigma_j)$, $j = 1, \ldots, S$, and computation of the output weights. Usually, the widths $\{\sigma_j\}_{j=1}^{S}$ are treated as hyperparameters of the RBF network and must be computed in advance, i.e., before the estimation of the weights of the output layer. In this regard, we follow the approach introduced by [40], which consists of three stages executed sequentially. First, the positions of the S centers are found by means of a vector quantization algorithm, such as the K-means or the SOM network [28]. Second, heuristics are used for specifying the radius σ_j for the S basis functions. In this paper, σ_j is computed as half of the distance between the center $\mathbf{c}_j$ and the nearest center $\mathbf{c}_{j^*}$, $j^* \neq j$.

Finally, the third step requires estimating the output weights using the least mean squares (LMS) learning rule [58].

Initially, we develop the proposed neural models for the identification of a SISO system; hence we consider a single output neuron. Later on, we extended the models to handling MIMO systems. Thus, the estimate of the output of the RBFN is then computed as

$$\hat{y}(t) = \mathbf{w}^T(t)\mathbf{z}(t) = \sum_{j=1}^{S} w_j(t) z_j(t), \tag{4}$$

where $\mathbf{w}(t) = [w_1(t)\ w_2(t)\ \cdots\ w_S(t)]^T$ is the current weight vector of the output neuron and $\mathbf{z}(t) = [z_1(t)\ z_2(t)\ \cdots\ z_S(t)]^T$ is the response vector of the hidden neurons to the current input vector $\mathbf{x}(t)$. The normalized LMS rule [59] is the sequential learning algorithm used to adapt the output weights:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha(t) e(t) \frac{\mathbf{z}(t)}{\epsilon + \|\mathbf{z}(t)\|^2}, \tag{5}$$

where $e(t) = y(t) - \hat{y}(t)$ is the prediction error and $0 < \epsilon \ll 1$ is a small constant necessary to avoid division by zero. The squared norm of $\mathbf{z}(t)$ is usually implemented as $\|\mathbf{z}(t)\|^2 = \mathbf{z}^T(t)\mathbf{z}(t)$ for the sake of speed.

Remark 1 The RBFN-NARX model is closely related to the zero-order Takagi-Sugeno (TS) model [54], with the outputs of the S basis functions expressing the activations of the S rules and the consequent output values being represented by the output weights.

As mentioned in the introduction, the LMN model was introduced by [25] as a generalization of the RBF network for system identification [11]. It can be viewed as implementing a decomposition of the global nonlinear input-output mapping into a set of local submodels, which are then smoothly integrated by corresponding basis functions. The smoothing process allows a smaller number of local models to cover larger areas of the input space than the standard RBFN model.

More specifically, while the output of the RBF model is computed as in Eq. (4), the output of the LMN model associates a local function $f_j(\mathbf{x}; \mathbf{w}_j)$ to each basis function, namely,

$$\hat{y}(t) = \sum_{j=1}^{S} z_j(t) \hat{y}_j(t) = \sum_{j=1}^{S} z_j(t) f_j(\mathbf{x}, \mathbf{w}_j; t), \tag{6}$$

where $\hat{y}_j(t)$ is the prediction of the local f_j. A common choice for this function is the multiple linear regression function:

$$f_j(\mathbf{x}, \mathbf{w}_j; t) = \mathbf{w}_j^T(t)\mathbf{x}(t) + \gamma_j(t), \tag{7}$$

$$= w_{j1}(t)x_1(t) + \cdots + w_{j,p+q}(t)x_{p+q}(t) + \gamma_j(t), \tag{8}$$

where $\gamma_j(t) \in \mathbb{R}$ is the bias term of the j-th local function.

As the RBF-NARX model, LMN-NARX model has widespread use, with recent applications in system identification and control literature [4, 8, 14, 29, 36, 42].

Remark 2 The LMN-NARX model is closely related to 1st-order TS model, with the basis functions playing the role of the rules, and the local function $f_j(\mathbf{x}; \mathbf{w}_j) = \mathbf{w}_j^T\mathbf{x}$ being the rules' consequents. Rule weighting is governed by the normalized activations $z_j(t) = z_j(t) / \sum_{k=1}^{S} z_k(t)$.

3 Growing Models: A Brief Presentation

3.1 Incremental Local Linear Mapping

LLM (Local Linear Mapping) based approaches generally assume a fixed number of centers that are distributed in the input space by some vector quantization method. An incremental network model for supervised learning was proposed in [17]. In this approach, error information obtained during training is used to determine when and where to insert new units. The network theory GNG (growing neural gas) [16] was used to give an incremental character to the LLM model.

The GNG model described in [16] is unsupervised and inserts new units to reduce the distance between the positioning of the prototype vectors and the input vector. For this reason, distortion error is accumulated locally and new units are inserted close to the unit with the highest accumulated error.

The GNG model [16] is a vector quantization algorithm, but when used with Hebb's rule it is able to preserve the topology of the input data in a similar way to the SOM network. The GNG network incrementally builds a graphical representation of a given dataset that is n-dimensional. The GNG method distributes a set of centers in $\mathbb{R}^{n_x}$. This is partially done in adaptation stages, but mainly by interpolation of new centers among existing ones. Between two centers there may be a *link* indicating neighborhood indicating neighborhood in $\mathbb{R}^{n_x}$. These *links* are used for interpolation and are inserted according to the Hebb learning rules [38] during the adaptation steps.

The GNG model was created with the intention of being unsupervised and inserts new units to reduce the average error between the input vectors and the prototype vectors. [17] described how this principle could be used for supervised learning. First you have to define what the output of the network is (which was not necessary for unsupervised learning). Then, use is made of the difference between the actual output and the desired output to guide the insertions of new units. The problem at hand is to approximate a function $G(\cdot) : \mathbb{R}^{L_u+L_y} \rightarrow \mathbb{R}^{n_y}$, given input-output pairs, with the particular case where $n_y = 1$.

Associated with each neuron j in the network (positioned in $\mathbf{w}_j$ in the input space), there is an output $\hat{y}_j$ and a vector $\mathbf{a}_j$ of dimension $n_x \times 1$ associated with this output, where $n_x = L_u + L_y$. The scalar $\hat{y}_j$ is the output of the network for cases where the input vectors coincide with one of the centers, i.e. $\mathbf{x}(t) = \mathbf{w}_j$. For any input vector, the nearest center s_1 is determined and the network output $\hat{y}(t)$ is calculated as follows:

$$\hat{y}(t) = \hat{y}_{s_1}(t) + \mathbf{a}_{s_1}^T(t)(\mathbf{x}(t) - \mathbf{w}_{s_1}(t)), \tag{9}$$

where the term $\hat{y}_{s_1}$ is a local first approximation of the output with the second term providing a first-order correction, in the guise of a Taylor expansion around the point $\hat{y}_{s_1}$.

We have to make a slight change in the original GNG algorithm to allow the proper adjustment of the local linear models, since we are interested in reducing the mean squared error. We then change the original equation from the GNG model to

$$\Delta e_{s_1}(t) = |y(t) - \hat{y}(t)|^2, \tag{10}$$

where $|\cdot|$ denotes the absolute value.

Equation (10) means that now the error is accumulated locally with respect to the function to be approximated. New units are inserted when this approximation is considered bad.

The local linear mappings $\mathbf{a}_j$ associated with the network units are initially defined randomly. In each adaptation step, the input-output data pair is used twice: (i) the input vector

is used for the adaptation of the center and (ii) the entire pair is used to improve the input vector. coefficients $\mathbf{a}_{s_1}$ from the nearest center s_1. This is done using, once again, the LMS rule [58]:

$$\hat{y}_{s_1}(t+1) = \hat{y}_{s_1}(t) + \alpha e_{s_1}(t), \tag{11}$$

$$\mathbf{a}_{s_1}(t+1) = \mathbf{a}_{s_1}(t) + \alpha e_{s_1}(t)(\mathbf{x}(t) - \mathbf{w}_{s_1}), \tag{12}$$

where $0 < \alpha \ll 1$ is the learning rate.

When a new unit r is inserted, a local linear mapping is interpolated between neighbors q and f:

$$\hat{y}_r(t) = 0,5(\hat{y}_q(t) + \hat{y}_f(t)) \tag{13}$$

$$\mathbf{a}_r(t) = 0,5(\mathbf{a}_q(t) + \mathbf{a}_f(t)). \tag{14}$$

A stopping criterion must be defined to terminate the growth process. This can be chosen arbitrarily depending on the application. One possible choice is to observe the performance of the network on a validation set during training and stop when that performance starts to decline. Alternatively, the error in the training set can be used or simply the number of units in the network if for some reason a specific network size is desired.

The model proposed by [17] became a driving force for the incremental approach of local linear mappings. Some works found in the literature were proposed based on the presented model [18, 22, 33, 55]. In [55], the proposal is very similar to that of [17], at each iteration, the feedback information of the approximation error of the current model is used to make the decision of inserting new local models in the area of input with the biggest error. He et al. [22] proposed an adaptive incremental learning framework called *adaptive incremental learning* (ADAIN), which is able to learn from data, accumulating experience over time and using this knowledge to improve the future performance of learning and prediction. In [33], the authors present a new learning framework for linear mappings based on *deep learning* for image reconstruction. In this approach, the model is able to learn the mapping function without neglecting the multi-layered nature of some applications.

3.2 Adapted RAN-LMS Model

The resource allocation network (RAN) was proposed by [45] and is a two-layer network, similar to an RBF network, but with a growing structure. The first layer consists of hidden units that respond only to a local region of the input space, that is, their neurons have a localized receptive field. The second layer contains output units that aggregate outputs from these units and create the function that approximates input-output mapping across the entire space.

This network aims to build local representations of the underlying input-output mapping. As already mentioned, the hidden units of the RAN network respond only to a local region of the input space. The network learns by allocating new units and adjusting the parameters of existing units. If the network malfunctions for a given input vector, as measured by the output neuron's approximation error, a new hidden unit is allocated to correct the network's response to that input vector. If the network works fine for a given input vector, the parameters of existing output units will be updated using the standard LMS rule already described in previous sections.

The RAN model starts as a tabula rasa; that is, no input-output pattern is stored yet. As patterns are presented to it, the network chooses to store some of them. In other words, no defaults have been chosen to be stored yet. Input-output pairs are presented and the network

identifies a pattern that is not well represented at the moment and allocates a new hidden unit that memorizes this sample. The output of the new hidden unit extends to the output layer. After the new unit is allocated, the network output is equal to the desired output: $\hat{y}(t) = y(t)$. Let the index of this new unit be n. A formalization of the procedure for inserting a new hidden unit is provided according [45]

1. The center of the new unit is defined as the $\mathbf{c}_n = \mathbf{x}(t)$.
2. The weight connecting the new hidden unit to the output unit is defined with $w_n = y(t) - \hat{y}(t)$, where $\hat{y}(t)$ can be calculated according to the Equation (4) of the RBF model. It is worth mentioning that the RAN model resembles a growing model of the RBF network.
3. The radius of the new hidden unit is given by

$$\sigma_n(t) = \kappa \|\mathbf{x}(t) - \mathbf{c}_{nearest}(t)\|, \tag{15}$$

such that $\kappa > 0$ is an overlap factor and $\mathbf{c}_{nearest}$ is the center closest to $\mathbf{c}_n$.

To decide on the insertion of a new hidden unit, the RAN model checks two novelty detection conditions.

Condition 1 Check if the input vector is too far from the existing centers, which is checked by the following rule:

$$\|\mathbf{x}(t) - \mathbf{c}_{nearest}(t)\| > \delta(t), \tag{16}$$

where $\delta(t)$ is a decreasing threshold for novelty detection.

Condition 2 Check if the error between the desired output and the current network output is large, that is,

$$|y(t) - \hat{y}(t)| > \epsilon, \tag{17}$$

where $|\cdot|$ is the absolute value operator and ϵ is the desired accuracy for the approximation problem

Note 1: To stabilize the network growth process, the novelty threshold starts with a high value $\delta_{\max}$, decreasing with time until it reaches a minimum value $\delta_{\min}$, which is kept constant for the remainder of the learning process. In this regard, the decay function suggested in [45] is used for the novelty threshold:

$$\delta(t) = \max\left[\delta_{\max} \exp(-t/\tau), \delta_{\min}\right], \tag{18}$$

where τ is a decay constant that defines the speed of inserting new hidden units and therefore the number of hidden units that can be inserted.

When a new unit is not allocated, the centers of the existing hidden units and the output neuron weights and threshold are adjusted through variants of the LMS rule developed below. It is worth mentioning that these equations were developed in this thesis and differ greatly from the original RAN model equations. Therefore, it was chosen to put this variant as a proposal of this thesis. For the application of this rule, the global objective function of the model is the instantaneous squared error of the output neuron:

$$J_{LMS}(t) = \frac{1}{2}e^2(t) = \frac{1}{2}(y(t) - \hat{y}(t))^2, \tag{19}$$

$$= \frac{1}{2}\left(y(t) - \sum_{j=1}^{S} z_j(t) w_j(t)\right)^2, \tag{20}$$

where $w_j(t)$ is the weight associated with the j-th hidden unit and $z_j(t)$ is the output of the associated hidden unit. Therefore, to update the weight vector of the jth local function, the following update rule is used:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \alpha(t)\frac{\partial J_{LMS}(t)}{\partial \mathbf{w}(t)}, \tag{21}$$

$$= \mathbf{w}(t) - \alpha(t)\frac{\partial J_{LMS}(t)}{\partial e(t)}\frac{\partial e(t)}{\partial \hat{y}(t)}\frac{\partial \hat{y}(t)}{\partial \mathbf{w}(t)}, \tag{22}$$

$$= \mathbf{w}(t) - \alpha(t)e(t)\cdot(-1)\cdot\mathbf{z}(t), \tag{23}$$

$$= \mathbf{w}(t) + \alpha(t)e(t)\mathbf{z}(t), \tag{24}$$

where it can be seen that the update of the weight vectors of the local submodels is modulated by the outputs of the hidden units $z_j(t)$, $j = 1, \ldots, S$. This modulation gives a localized character to the parameter updating process. The greater the value of $z_j(t)$, the greater the variation in the weight vector $\mathbf{w}(t)$.

Upgrading the centers of hidden units follows a similar procedure. For this, the global objective function of the aforementioned model was used once again. In this case, the following rule applied:

$$\mathbf{c}_j(t+1) = \mathbf{c}_j(t) - \alpha(t)\frac{\partial J_{LMS}(t)}{\partial \mathbf{c}_j(t)}, \tag{25}$$

$$= \mathbf{c}_j(t) - \alpha(t)\frac{\partial J_{LMS}(t)}{\partial e(t)}\frac{\partial e(t)}{partial\hat{y}(t)}\frac{\partial \hat{y}(t)}{\partial z_j(t)}\frac{\partial z_j(t)}{\partial \mathbf{c}_j(t)}, \tag{26}$$

$$= \mathbf{c}_j(t) - \alpha(t)e(t)\cdot(-1)\cdot w_j(t)\cdot\left(\frac{z_j(t)}{\sigma_j^2}(\mathbf{x}(t) - \mathbf{c}_j(t))\right), \tag{27}$$

$$= \mathbf{c}_j(t) + \frac{\alpha(t)}{\sigma_j^2}z_j(t)e(t)w_j(t)(\mathbf{x}(t) - \mathbf{c}_j(t)), \tag{28}$$

where Eq. (28) is derived using the Gaussian basis function. Equations (25), (26) and (27) were developed for this paper seeking a better understanding of the updating of centers of the hidden units performed in the work of [45].

For the experiments developed in this thesis, better results were obtained using the normalized LMS rule [59]. To implement this variant of the LMS rule, replace the original learning rate $\alpha(t)$ with $\alpha'(t) = \alpha(t)/\|\mathbf{x}(t)\|^2$, where $\|\mathbf{x}(t)\|^2$ is the quadratic norm of the current input vector.

The ease of implementation and low computational cost are two major attractions of the RAN model. Pai et al. [44] applied RAN to estimate tool wear in milling operations. Acoustic emission signals, surface roughness parameters and cutting conditions (cut speed, feed) were used to formulate input patterns. The RAN performance was compared with the MLP network.

Mahanand et al. [35] presented a new approach for the identification of brain regions responsible for Alzheimer's disease using magnetic resonance imaging. The approach incorporated a version of the RAN, the SRAN (*self-adaptive resource allocation network*) [53] for the classification of Alzheimer's disease. The SRAN classifier uses a sequential learning algorithm, employing self-adaptive bounds to select the appropriate training samples and discarding redundant samples to avoid overtraining. These selected training samples are then used to efficiently develop the network architecture.

Bandyopadhyay [24] demonstrated the applicability of the RAN model through three examples from various domains: water cooling network, carbon constrained energy sector

planning and water allocation network. In [34], the RAN model is trained *offline* to determine an initial structure. From there, the initial RAN model is used for the *online* prediction of photovoltaic energy and is later updated. The simulation results showed that the two-stage RAN model can effectively improve the accuracy of PV output prediction.

Sun et al. [57] used the RAN model to estimate the parameters of a proportional-integral-derivative (PID) controller for automated guided vehicles (AGV). The simulation results show that the method has high performance, good tracking capability and high approach accuracy.

It can be noted that, despite the formulation having taken place in the early 1990s, recent applications of the RAN network are still found. This network will be used, with some modifications, for the development of the first proposal of this thesis in the next section.

4 The Proposed Approach

As we mentioned in the introduction, we are seeking an LMN-NARX model with the following features: growing online structure (i.e., no need to define the hidden units beforehand), fast (i.e., with few math operations) recursive updating rule, lightweight (i.e., small memory use), and outlier-robust. In this regard, efficiency in performance and simplicity of implementation are the key issues to bear in mind. Some of the alternative algorithms available today possess some of the features listed above, but not all of them, especially regarding the recursive updating rule. Most of them use the RLS algorithm, which requires much more memory space than the LMS algorithm due to the use of covariance matrices, one for each neuron in the network.

Growing structure This feature of the proposed approach is implemented with the help of the growing scheme of the RAN architecture [45], usually called nowadays as the novelty criterion. The RAN is a two-layered network, which resembles an RBFN but with a growing structure. The first layer consists of hidden units that respond to only a local partition of the input space; that is, its neurons have a localized receptive field. The second layer contains output units that aggregate outputs from these units and create the function that approximates the input-output mapping over the entire space.

The RAN model learns by allocating new hidden units and adjusting the parameters of existing ones. If the network performs poorly on a given input vector, as measured by the approximation error of the output neuron, then a new hidden unit is allocated to correct the network's response to that input vector. If the network performs well on a given input vector, then the parameters of existing output units are updated using the standard LMS updating rule.

4.1 A Growing LMN-NARX Model

The novelty criterion of the RAN network will be adapt here to develop a variant of the LMN-NARX model, which is capable of growing in time. The resulting algorithm is described step-by-step in the following.

The learning process The G-LMN-NARX model starts with a single hidden unit. As the input-output pairs starts to be presented, the network has to identify any pattern that is not currently well represented by the neural model and allocates a new hidden unit that memorizes that sample. Once the new unit is allocated, the network output is set to the target output: $\hat{y}(t) = y(t)$. Let the index of this new unit be n. Details on the insertion procedure of a new hidden unit is provided below.

1. The center of the new hidden unit is set to $\mathbf{c}_n = \mathbf{x}(t)$.
2. The weight connecting the new hidden unit to the output unit is set to $w_n = y(t) - \hat{y}(t)$.
3. The radius of the new hidden unit is given by

$$\sigma_n(t) = \kappa \|\mathbf{x}(t) - \mathbf{c}_{nearest}(t)\| \tag{29}$$

where $\kappa > 0$ is an overlapping factor and $\mathbf{c}_{nearest}$ is the closest center to $\mathbf{c}_n$.

To decide on the insertion of a new hidden unit, the G-LMN-NARX verifies two novelty criteria.

Criterion 1 checks how far is the input vector from existing centers by means of the following rule:

$$\|\mathbf{x}(t) - \mathbf{c}_{near}(t)\| > \delta(t), \tag{30}$$

where $\delta(t)$ is a decaying threshold for novelty detection.

Criterion 2 checks if the error between the target output and the current output of the network is large, i.e.

$$|y(t) - \hat{y}(t)| > \varepsilon, \tag{31}$$

where $|\cdot|$ is the absolute value operator and ε is the desired accuracy of the approximation problem.

Remark 3 To stabilize the network's growing process, Platt [45] suggests the following annealing function for the novelty threshold:

$$\delta(t) = \max\left[\delta_{\max} \exp(-t/\tau), \delta_{\min}\right], \tag{32}$$

where τ is a decay constant defining the speed of insertion of new hidden units.

If a new hidden unit is not inserted, the existing hidden units' centers, and the weights and bias term of the output neuron are adjusted through variants of the LMS rule. For this purpose, the global objective function of the model is the instantaneous squared error of the output neuron:

$$J_{LMS}(t) = \frac{1}{2}e^2(t) = \frac{1}{2}(y(t) - \hat{y}(t))^2, \tag{33}$$

$$= \frac{1}{2}\left(y(t) - \sum_{j=1}^{S} z_j(t)\hat{y}_j(t)\right)^2, \tag{34}$$

where $\hat{y}_j(t)$ is the output of the j-th local model and $z_j(t)$ is the output of the associated hidden unit.

Hence, for updating the weight vector of the j-th local function, the following update rule is used:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) - \alpha'(t)\frac{\partial J_{LMS}(t)}{\partial \mathbf{w}_j(t)}, \tag{35}$$

$$= \mathbf{w}_j(t) - \alpha'(t)\frac{\partial J_{LMS}(t)}{\partial e(t)}\frac{\partial e(t)}{\partial \hat{y}(t)}\frac{\partial \hat{y}(t)}{\partial \hat{y}_j(t)}\frac{\partial \hat{y}_j(t)}{\partial \mathbf{w}_j(t)}, \tag{36}$$

$$= \mathbf{w}_j(t) - \alpha'(t)e(t)\cdot(-1)\cdot z_j(t)\cdot \mathbf{x}(t), \tag{37}$$

$$= \mathbf{w}_j(t) + \alpha'(t)z_j(t)e(t)\mathbf{x}(t), \tag{38}$$

where $\alpha'(t) = \alpha(t)/(\epsilon + \|\mathbf{x}(t)\|^2)$, with $\|\mathbf{x}(t)\|^2$ denoting the squared norm of the current input vector. It should be noted that the updating of the weight vectors of the local submodels is modulated by the outputs of the hidden units $z_j(t)$, $j = 1, \ldots, S$. This modulation gives a localized character to the parameter updating process. The higher the values of $z_j(t)$, the higher is the change in the weight vector $\mathbf{w}_j(t)$.

The updating of the centers of the hidden units follows a similar procedure. In this case, the following rule applies:

$$\mathbf{c}_j(t+1) = \mathbf{c}_j(t) - \alpha'(t)\frac{\partial J_{LMS}(t)}{\partial \mathbf{c}_j(t)}, \tag{39}$$

$$= \mathbf{c}_j(t) - \alpha'(t)\frac{\partial J_{LMS}(t)}{\partial e(t)}\frac{\partial e(t)}{\partial \hat{y}(t)}\frac{\partial \hat{y}(t)}{\partial z_j(t)}\frac{\partial z_j(t)}{\partial \mathbf{c}_j(t)}, \tag{40}$$

$$= \mathbf{w}_j(t) - \alpha'(t)e(t)\cdot(-1)\cdot\hat{y}_j(t)\cdot\left(\frac{z_j(t)}{\sigma_j^2}(\mathbf{x}(t) - \mathbf{c}_j(t))\right), \tag{41}$$

$$= \mathbf{c}_j(t) + \frac{\alpha'(t)}{\sigma_j^2}z_j(t)e(t)\hat{y}_j(t)(\mathbf{x}(t) - \mathbf{c}_j(t)), \tag{42}$$

where Eq. (42) is derived by using the Gaussian basis function as defined in Eq. (3). The pseudocode of the G-LMN-NARX model is presented in Algorithm 1.

Algorithm 1 G-LMN-NARX model

Initialization:
Define: p, q, κ, ϵ, τ, δ_{max}, δ_{min}, σ_j, and α.
Set: $\delta_0 = \delta_{max}$, and $S = 1$.
Iterate: loop over presentations of input-output pairs
1. Prediction: For each input vector $\mathbf{x}(t)$, compute
1.1. $z_j(t) = \phi\left(d(\mathbf{x}(t), \mathbf{c}_j(t))\right); \quad j = 1, \ldots, S.$
1.2. $z_j(t) = z_j(t)/\sum_{r=1}^{S} z_r(t); \quad j = 1, \ldots, S.$
1.3. $f_j(\mathbf{x}, \mathbf{w}_j; t) = \mathbf{w}_j(t)^T\mathbf{x}(t) + \gamma_j(t); \quad j = 1, \ldots, S.$
1.4. $\hat{y}(t) = \sum_{j=1}^{S} z_j(t) f_j(\mathbf{x}, \mathbf{w}_j; t);$
2. Compute error:
2.1. $e(t) = y(t) - \hat{y}(t);$
3. Find the distance to nearest center:
3.1. $d_{near}(t) = \|\mathbf{x} - \mathbf{c}_{near}(t)\|;$
4. Decide between Growing or Updating:
IF $|e(t)| > \epsilon$ and $d_{near}(t) > \delta(t)$, **THEN**
{ % Grows
Allocate new unit $\mathbf{c}_n = \mathbf{x}(t)$ and $w_n = e(t)$;
IF this is the first unit to be allocated,
THEN $\sigma_n = \kappa\delta(t)$;
ELSE $\sigma_n = \kappa d_{near}(t)$.
}
ELSE
{ % Updates
Update the weight vectors using Eq. (38).
Update the centers using Eq. (42).
Update the novelty threshold using Eq. (32).
}

Remark 4 Our choice for the normalized LMS rule to update the parameters of the G-LMN-NARX model relies in three important aspects. Firstly, it is a light-weight rule for parameter estimation in comparison to the RLS rule, which is also a common alternative for recursive

parameter updating [37], since there is no need to estimate S inverse covariance matrices of dimension $(p+q)\times(p+q)$, one for each hidden unit, required to compute the associated Kalman gain vectors. Secondly, it has been demonstrated that the normalized LMS rule is an optimal filter in the H_∞ sense [21], meaning that it minimizes the maximum energy gain from the disturbances to the filtered errors. Finally, the third aspect is that an outlier-robust learning rule is straightforwardly obtained from the original LMS rule with no extra computational cost, as we show next.

5 An Outlier-Robust G-LMN-NARX Model

Real-world data are commonly contaminated with outliers, which are understood in a broad sense as observations differing markedly from other samples. It is often assumed that outliers follow a distribution other than the usual Gaussian noise, with heavy-tailed distribution (e.g., t-Student) being common choices. In this regard, standard estimation rules based on functions of squared errors, such as the OLS (batch) and the LMS (recursive), lead to predictive models with suboptimal performance since those rules are developed under the assumption of Gaussianity of the noise. This assumption is adequate for the usual measurement noise, but not for non-Gaussian noise.

Outliers' undesirable influence is even more pronounced in online applications, where an outlier-removal process is either harder or impossible to carry out in comparison to an offline scenario. In online applications, a recursive outlier-robust estimation technique is required for a good performance of the G-LMN-NARX model, primarily because the growing process in this model depends on the prediction error $y(t)-\hat{y}(t)$ (see Eq. (31)). If the model's prediction is biased by the influence of outliers, the growing process of the G-LMN-NARX is prone to be affected.

Robust estimation techniques have been developed which are resilient to outliers, such as those developed within the framework of M-estimators [23]. Based on this framework, robust recursive estimators are found in [62] and [10], who introduced simple modifications to the standard LMS-rule in order to improve its performance in outlier-rich data. The resulting rule was then called *least mean M-estimate* (LMM) and is as fast as the original LMS rule, in the sense that negligible computational burden is added to the estimation process.

In order to apply the underlying principle of the LMM rule to the updating rules of the weight vectors $\mathbf{w}_j$ and the centers $\mathbf{c}_j$ of the G-LMN-NARX model, we start by redefining the objective function in Eq. (33) to a more general expression given by

$$J_{LMM}(t) = \rho(e(t)) = \rho(y(t) - \hat{y}(t)) = \rho\left(y(t) - \sum_{j=1}^{S} z_j(t)\hat{y}_j(t)\right), \tag{43}$$

where the function $\rho(\cdot)$ computes the contribution of the current prediction error $e(t)$ to the objective function. The following requirements are imposed to the function:

1. $\rho(e) \geq 0$;
2. $\rho(0) = 0$;
3. $\rho(e) = \rho(-e)$;
4. $\rho(e(t_1)) \geq \rho(e(t_2))$, se $|e(t_1)| > |e(t_2)|$.

For the system identification task, two functions stand out, namely, Hampel function [62] and modified Huber (HM) [10]. In this thesis, we used the Modified Huber function described as

$$\varrho(e(t)) = \begin{cases} e(t)^2/2, & \text{se } |e(t)| < \beta, \\ \beta^2/2, & \text{otherwise,} \end{cases} \tag{44}$$

where β is the abnormality threshold, which defines the range in which the residual is considered to be within or outside the usual standards.

If $\rho(e(t)) = \frac{1}{2}e^2(t)$, one gets the instantaneous squared error objective function used by the LMS rule.

From the exposed, a robust variant of the updating rule of the weight vector of the j-th local function, shown in Eq. (38), is given by

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) - \alpha'(t)\frac{\partial J_{LMM}(t)}{\partial \mathbf{w}_j(t)}, \tag{45}$$

$$= \mathbf{w}_j(t) - \alpha'(t)\frac{\partial \rho(e(t))}{\partial e(t)}\frac{\partial e(t)}{\partial \hat{y}(t)}\frac{\partial \hat{y}(t)}{\partial \hat{y}_j(t)}\frac{\partial \hat{y}_j(t)}{\partial \mathbf{w}_j(t)}, \tag{46}$$

$$= \mathbf{w}_j(t) - \alpha'(t)q(e(t))e(t)\cdot(-1)\cdot z_j(t)\cdot \mathbf{x}(t), \tag{47}$$

$$= \mathbf{w}_j(t) + \alpha'(t)z_j(t)q(e(t))e(t)\mathbf{x}(t), \tag{48}$$

where $q(e(t)) = \frac{1}{e(t)}\frac{\partial \rho(e(t))}{\partial e(t)}$ is a function with the role of penalizing high values of the residuals $e(t)$ due to outliers.

It is important to highlight that the sole difference between Eqs. (48) and (38) is the weighting factor $q(e(t))$. In this paper, since we are focused on simplicity and speed of computation, our choice for $q(e(t))$ is the *Huber* function:

$$q(e(t)) = \begin{cases} \frac{\kappa}{|e(t)|}, & \text{if } |e(t)| > \beta, \\ 1, & \text{otherwise.} \end{cases} \tag{49}$$

where the constant $\beta > 0$ is a user-defined threshold.

If $|e(t)| \leq \beta$, then one gets $q(e(t)) = 1$, with Eq. (48) reducing to its original version shown in Eq. (38). If $|e(t)| > \beta$, $q(e(t))$ decreases exponentially to zero as $|e(t)| \to \infty$. This way, the LMM updating rule effectively reduces the effect of large errors mostly caused by outliers. It is recommended to use $\beta = 1.345\sigma$, where σ corresponds to the standard deviation of the residuals.

By the same token, we can derive a robust version for updating the center of the j-th hidden unit. Skipping the details, this is given by the following expression:

$$\mathbf{c}_j(t+1) = \mathbf{c}_j(t) + \frac{\alpha'(t)}{\sigma_j^2}z_j(t)q(e(t))e(t)\hat{y}_j(t)(\mathbf{x}(t) - \mathbf{c}_j(t)), \tag{50}$$

which, for the Huber function defined in Eq. (49), reduces to Eq. (42) whenever $|e(t)| \leq \beta$. The pseudocode of the *outlier-robust* G-LMN-NARX (ORG-LMN-NARX, for short) model is presented in Algorithm 2.

As mentioned in the introduction, as the first goal of this paper, we developed a growing LMN model for NARX-based online system identification. This goal led to the development of the G-LMN-NARX model. As the second goal, we aimed to add outlier-robustness to the G-LMN-NARX model by replacing the original LMS rules for updating the weight vectors of the local submodels and the centers of the hidden units. This goal led to the development of the ORG-LMN-NARX model, which is expected to perform better than the G-LMN-NARX model in the presence of outlier-rich scenarios. The computational experiments that test this hypothesis and the obtained results are presented from the next section onwards.

Algorithm 2 ORG-LMN-NARX model

Initialization:
Define: $p, q, \kappa, \epsilon, \tau, \delta_{max}, \delta_{min}, \sigma_j, \alpha$, and β.
Set: $\delta_0 = \delta_{max}$, and $S = 1$.
Iterate: loop over presentations of input-output pairs
1. Prediction: For each input vector $\mathbf{x}(t)$, compute
1.1. $z_j(t) = \phi\left(d(\mathbf{x}(t), \mathbf{c}_j(t))\right); \quad j = 1, \ldots, S.$
1.2. $z_j(t) = z_j(t) / \sum_{r=1}^{S} z_r(t); \quad j = 1, \ldots, S.$
1.3. $f_j(\mathbf{x}, \mathbf{w}_j; t) = \mathbf{w}_j(t)^T \mathbf{x}(t) + \gamma_j(t); \quad j = 1, \ldots, S.$
1.4. $\hat{y}(t) = \sum_{j=1}^{S} z_j(t) f_j(\mathbf{x}, \mathbf{w}_j; t);$
2. Compute error:
2.1. $e(t) = y(t) - \hat{y}(t);$
3. Compute weighting factor from Huber function:
IF $|e| > \beta$,
THEN $q(e) = \beta / |e|;$
ELSE $q(e) = 1$.
4. Find the distance to nearest center:
4.1. $d_{near}(t) = \|\mathbf{x} - \mathbf{c}_{near}(t)\|;$
5. Decide between Growing or Updating:
IF $|e(t)| > \epsilon$ and $d_{near}(t) > \delta(t)$, **THEN**
{ % Grows
Allocate new unit $\mathbf{c}_n = \mathbf{x}(t)$ and $w_n = e(t)$;
IF this is the first unit to be allocated,
THEN $\sigma_n = \kappa \delta(t)$;
ELSE $\sigma_n = \kappa d_{near}(t)$.
}
ELSE
{ % Updates
Update the weight vectors using Eq. (48).
Update the centers using Eq. (50).
Update the novelty threshold using Eq. (32).
}

6 The MIMO Case

The previous developments aimed at SISO systems, but they can be straightforwardly extended to the identification of MIMO dynamic systems. For this purpose, the regression vector is augmented to encompass information about all input and output variables of interest [32]. In this regard, we defined the input regression subvector $\mathbf{u}_a(t-1)$ associated with the a-th input as

$$\mathbf{u}_a(t-1) = [u_a(t-1) \;\; u_a(t-2) \;\; \cdots \;\; u_a(t-L_u)]^T, \quad a = 1, 2, \cdots n_u, \tag{51}$$

where L_u denotes the memory order of the allinput variables, kept the same for all variables. Likewise, we defined the output regression subvector $\mathbf{y}_b(t-1)$ associated with b-th output as

$$\mathbf{y}_b(t-1) = [y_b(t-1) \;\; y_b(t-2) \;\; \cdots \;\; y_b(t-L_y)]^T, \quad b = 1, 2, \cdots n_y, \tag{52}$$

where L_y denotes the memory order of the output variables, kept the same for all variables. Based on [6], the vector of regressors for the MIMO case is then given by

$$\mathbf{x}(t) = [\mathbf{y}_1(t-1) \;\; \mathbf{y}_2(t-1) \;\; \cdots \;\; \mathbf{y}_{n_y}(t-1), \;\; \mathbf{u}_1(t-1) \;\; \mathbf{u}_2(t-1) \;\; \cdots \;\; \mathbf{u}_{n_u}(t-1)] \tag{53}$$

where $\mathbf{x}(t) \in \mathbb{R}^{(n_y \cdot L_y + n_y \cdot L_u)}$ and $\mathbf{y}(t) = \mathbf{G}(\mathbf{x}(t))$ as illustrated in Fig. 2.

The nonlinear function $\mathbf{G}(\cdot) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ is considered to be unknown and $n_x = n_y \cdot L_y + n_u \cdot L_u$. The observed data in the form of multiple input and multiple output

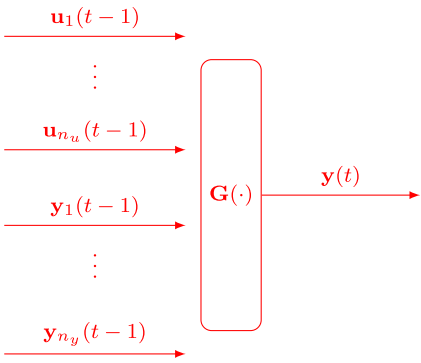

Fig. 2 Scheme of a NARX MIMO model, where $\mathbf{G}(\cdot)$ is a generic nonlinear function

time series is used to build an approximate model $\hat{\mathbf{G}}(\cdot)$ for the target function $\mathbf{G}(\cdot)$. For this purpose, we generalize the models described previously in this work to a MIMO system.

The G-LMN model will be described in more detail for the MIMO problem in the task of learning the dynamics of a given MIMO system using the input vectors as defined in Eq. (53). For this, we considered a neural model with an output layer with n_y neurons. Thus, the output of the G-LMN model associates a local vector function $\mathbf{f}_j(\mathbf{x}; \mathbf{w}_j)$ to each basis function, namely,

$$\hat{\mathbf{y}}(t) = \sum_{j=1}^{S} z_j(t)\hat{\mathbf{y}}_j(t) = \sum_{j=1}^{S} z_j(t)\mathbf{f}_j(\mathbf{x}, \mathbf{w}_j; t), \tag{54}$$

where $\hat{\mathbf{y}}_j(t) \in \mathbb{R}^{n_y}$ is the local multidimensional prediction associated with $\mathbf{f}_j \in \mathbb{R}^{n_y}$, whose components are given by

$$\begin{aligned}
\hat{y}_1(t) &= \sum_{j=1}^{S} z_j(t) f_{j,1}(\mathbf{x}, \mathbf{w}_j; t), \\
\hat{y}_2(t) &= \sum_{j=1}^{S} z_j(t) f_{j,2}(\mathbf{x}, \mathbf{w}_j; t), \\
&\vdots \qquad\qquad \vdots \\
\hat{y}_{n_y}(t) &= \sum_{j=1}^{S} z_j(t) f_{j,n_y}(\mathbf{x}, \mathbf{w}_j; t).
\end{aligned} \tag{55}$$

For this case, the learning process follows the same steps as in the SISO case. The formalization of the insertion of a new hidden unit is described below:

1. The center of the new hidden unit is given by $\mathbf{c}_n = \mathbf{x}(t)$.
2. The weight $\mathbf{w}_n$ connecting the new unit to the output layer is given by a matrix with n_x rows with each row being equal to $\mathbf{y}(t) - \hat{\mathbf{y}}(t)$.
3. The radius of the new unit is given by the Eq. (29).

For the MIMO case, the novelty criteria for inserting a new hidden unit are the same as the ones used in the SISO case, with the Criterion 2 in (31) adapted to the multi-ouput case

as follows:

$$\|\mathbf{y}(t) - \hat{\mathbf{y}}(t)\| > \varepsilon, \tag{56}$$

where $\|\cdot\|$ denotes the Euclidean norm. When a new unit is not allocated, the centers, weights and thresholds must be adjusted using the LMS rule or the LMM rule. To apply this rule to the MIMO case, the global objective function of the model, described in Eq. (33) for the SISO model, is replaced by its MIMO counterpart:

$$J_{MIMO}(t) = \frac{1}{2}\|\mathbf{e}(t)\|^2 = \frac{1}{2}\|\mathbf{y}(t) - \hat{\mathbf{y}}(t)\|^2, \tag{57}$$

$$= \frac{1}{2}\|\mathbf{y}(t) - \sum_{j=1}^{S} z_j(t)\hat{\mathbf{y}}_j(t)\|^2, \tag{58}$$

where $\hat{\mathbf{y}}_j(t) \in \mathbb{R}^{n_y}$ is the output of the j-th local model and $z_j(t)$ is the output of the unit associated hidden.

Therefore, to update the weight vector of the j-th local function, the updating rule in Eq. (38) is changed to

$$\begin{pmatrix} \mathbf{w}_{j,1}(t+1) \\ \vdots \\ \mathbf{w}_{j,n_y}(t+1) \end{pmatrix}_{n_y \times n_x} = \begin{pmatrix} \mathbf{w}_{j,1}(t) \\ \vdots \\ \mathbf{w}_{j,n_y}(t) \end{pmatrix}_{n_y \times n_x} - \alpha'(t) z_j(t) \begin{pmatrix} e_1(t) \\ \vdots \\ e_{n_y}(t) \end{pmatrix}_{n_y \times 1} \begin{pmatrix} x_1(t) \\ \vdots \\ x_{n_x}(t) \end{pmatrix}^T_{1 \times n_x}. \tag{59}$$

The updating rule for the centers of the hidden units, shown in Eq. (42) for the SISO case, receives similar modifications to take into account the multiple outputs:

$$\begin{pmatrix} c_{j,1}(t+1) \\ \vdots \\ c_{j,n_x}(t+1) \end{pmatrix}_{n_x \times 1} = \begin{pmatrix} c_{j,1}(t) \\ \vdots \\ c_{j,n_x}(t) \end{pmatrix}_{n_x \times 1} + \frac{\alpha'(t)}{\sigma_j^2} z_j(t) \begin{pmatrix} \hat{y}_1(t) \\ \vdots \\ \hat{y}_{n_y}(t) \end{pmatrix}^T_{1 \times n_y} \begin{pmatrix} e_1(t) \\ \vdots \\ e_{n_y}(t) \end{pmatrix}_{n_y \times 1} \left(\begin{pmatrix} x_1(t) \\ \vdots \\ x_{n_x}(t) \end{pmatrix} - \begin{pmatrix} c_{j,1}(t) \\ \vdots \\ c_{j,n_x}(t) \end{pmatrix} \right)_{n_x \times 1} \tag{60}$$

Once the predicted outpus for the MIMO case have been computed at a given time step t, the corresponding prediction errors due to each output are calculated by

$$\begin{aligned} e_1(t) &= y_1(t) - \sum_{j=1}^{S} z_j(t) f_{j,1}(\mathbf{x}, \mathbf{w}_j; t), \\ e_2(t) &= y_2(t) - \sum_{j=1}^{S} z_j(t) f_{j,2}(\mathbf{x}, \mathbf{w}_j; t), \\ &\vdots \\ e_{n_y}(t) &= y_{n_y}(t) - \sum_{j=1}^{S} z_j(t) f_{j,n_y}(\mathbf{x}, \mathbf{w}_j; t). \end{aligned} \tag{61}$$

Table 1 Summary of the evaluated data sets

Data set	Evaluated datasets			
	Estimation samples	Test samples	$\hat{L}_u$	$\hat{L}_y$
Hydraulic Actuator	512	512	4	4
pH	200	800	5	5
Silverbox	91,072	40,000	10	10

7 Results and Discussion

In this section, we report the results of a comprehensive performance comparison between the G-LMN-NARX and ORG-LMN-NARX models proposed in Sects. 4 and 5. The experiments' goal is to evaluate the ability of the proposed models in providing an incremental and accurate model for dynamical system identification in the presence of outliers. For the sake of completeness, a comparison with the fixed-size LMN-NARX model introduced in [5] is provided.

All the evaluated models were implemented from scratch using the script language of Octave,[1] and the corresponding codes are available upon request. We report results on three MISO benchmarking datasets (Hydraulic Actuator, *pH*, Silverbox), which are publicly available for download from the DaISy repository website at the *Katholieke Universiteit Leuven*.[2] Some important features of Hydraulic Actuator, *pH* and Silverbox datasets are summarized in Table 1.

The proposed models were also tested on a dataset originated from a MIMO system. For this, we selected the Industrial Dryer dataset described in [7] and [12]. This system has 3 inputs ($n_u = 3$), namely, raw material flow rate $u_1(t)$, fuel flow rate $u_2(t)$, and hot gas exhaust fan speed $u_3(t)$; and 3 outputs ($n_y = 3$): dry bulb temperature $y_1(t)$, wet bulb temperature $y_2(t)$ and raw material material moisture content $y_3(t)$. For better viewing of the amplitudes of the input u_1, they were divided by a factor of 17.

Accuracy of all the models is evaluated by the root mean square error: $RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N} e^2(t)}$. The training was carried out using one-step-ahead prediction mode, a.k.a as series-parallel training mode [48]. Outliers were artificially added only to training data in different proportions. For this, we followed the procedure described in [39]. The reported results correspond to the post-training evaluation of the model in outlier-free scenarios. The rationale for this approach is to assess how the outliers affect the parameter estimation process (training) and its impact in the model validation (testing) phase. Trained models were tested under the *free simulation* regime, in which predicted output values are fed back to the input regression vector.

During testing, the parameters of the models are not updated since our goal is to check if the actual dynamics of the system of interest has been learned even in the presence of outliers. We defined 100 epochs for training all models for the Actuator, *pH* and industrial drier data sets, with one epoch meaning the presentation of the whole training set. An online identification task is simulated only for the Silverbox data set because it is long enough to allowing this type of test. The number of hidden units is initially set to $S = 1$ for the growing models (G-LMN-NARX and ORG-LMN-NARX). For the fixed-size LMN-NARX

[1] www.gnu.org/software/octave/.

[2] http://homes.esat.kuleuven.be/~smc/daisy/.

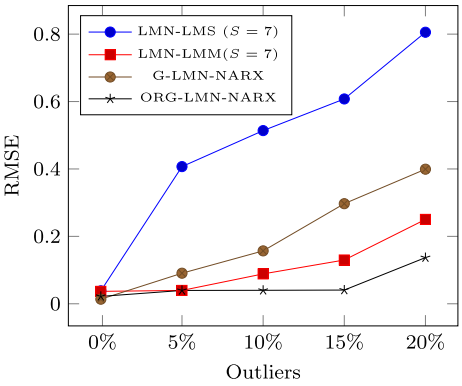

Fig. 3 RMSE values for test data as a function of the percentage of outliers in training data (Actuator dataset). The robust models perform better than the ones using the usual LMS rule. The proposed ORG-LMN-NARX model is the best performing one for this data set

models, this number is set to $S = 7$ as indicated by [5]. We trained the G-LMN-NARX model using the following values for its hyparameters: $\kappa = 0.9$, $\delta_{\max} = \max(\|\mathbf{x} - \mathbf{c}_0\|)$, $\delta_{\min} = \delta_{\max}/10$, $\tau = 10$, and $\epsilon = 10^{-4}$. The initial and final learning rates were set to $\alpha_0 = 0.5$ and $\alpha_T = 0.001$.

Results for the actuator data set

This data set consists of two time series (input: valve opening, output: oil pressure) each comprised of 1024 measurements collected from a hydraulic actuator on a crane [50]. We used 512 samples for model building and parameter estimation and the remaining half for model validation. The RMSE values as a function of the percentage of outliers are shown in Fig. 3. As expected, the robust fixed-size model (LMN-LMM) and its growing counterpart (ORG-LMN-NARX) performed better than the non-robust versions (those using the plain LMS rule). Furthermore, the ORG-LMN-NARX model was able to perform better than the fixed-size LMN-LLM model. The performance of the proposed ORG-LMN-NARX model is practically insensitive to the presence of outliers (i.e., the error curve is almost horizontal).

The improvement in performance achieved by the ORG-LMN-NARX model over its LMS-based counterpart for the Actuator data set can be visualized in Fig. 4. This figure shows typical predicted output time series in dashed red lines, while the actually observed output time series is shown in solid blue lines. The corresponding evolution of the number of local models for the two proposed models is shown in the upper row of the Fig. 5. This number has stabilized in $S = 7$ for both models. It should be noted that the number of local models grows faster for the robust ORG-LMN-NARX model, which uses the LMM learning rule. This rule suitably handles outliers by minimizing their negative influence in the parameter estimation process.

A positive side effect is that the outlier-detection mechanism of the LMM rule also signalizes to the adopted growing strategy that new local models should be included as soon as possible in order to deal with the outliers' negative influence. The effects of this interesting property of proposed ORG-LMN-NARX model can be seen in the faster rate of decay of the corresponding RMSE values, as shown in Fig. 6 (upper row). This decay rate is even more pronounced for the other two evaluated data sets, as will be shown next (Fig. 7).

The RMSE values as a function of the percentage of *outliers* are shown in Table 2. As expected, the robust models performed better than the non-robust versions. Furthermore, the ORG-LMN model was able to perform better than the RAN-LMM model. The performance of the proposed ORG-LMN model is practically insensitive to the presence of *outliers*. The performance of the growing models was generally better than the fixed-size models. This

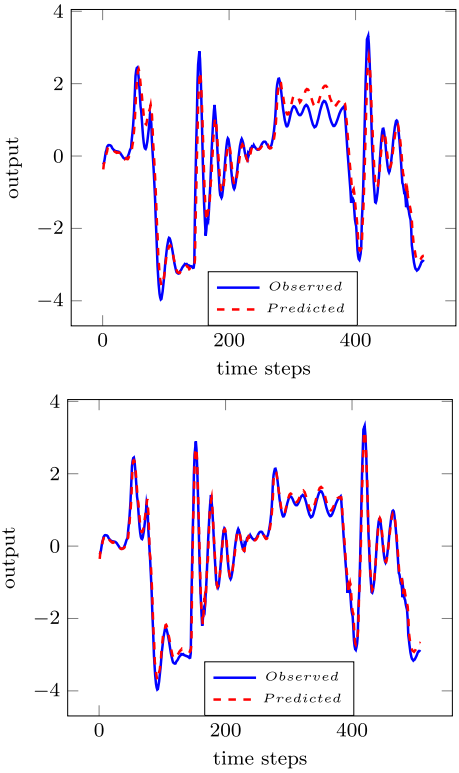

Fig. 4 Typical predictive performances of the proposed models, G-LMN-NARX (upper) and ORG-LMN-NARX (lower), after being trained with 15% of outliers (Actuator dataset)

confirms our hypothesis that growing models are better for dealing with a possible non-stationarity of the system.

Results for the pH dataset

The data samples come from a *pH* neutralization process in a constant volume stirring tank. The control input is the base solution flow and the output is the *pH* value of the solution in the tank. We use the first 200 samples for model building and parameter estimation and the next 800 samples for model validation.

A comparison in terms of mean RMSE values of the two fixed-size models (LMN-LMS and LMN-LMM) and the proposed growing models (G-LMN-NARX and ORG-LMN-NARX) for the *pH* dataset is shown in Fig. 9 for different scenarios of outlier contamination. As expected for the outlier-free scenario, the performances of the two fixed-size models (LMN-LMS and LMN-LMM) are basically the same. However, as the level of outlier contamination increases, one can again observe that the outlier-robust versions achieve better performance than those models using the original LMS-based learning rule. For this data set the best performance was achieved by the proposed ORG-LMN-NARX model.

Typical predicted output time series produced by the two proposed growing models for the *pH* data set are shown in Fig. 8. A difference between the predictive performances of the two models is hard to visualize, but the outlier-robust model (ORG-LMN-NARX) was capable of improving the RMSE on the prediction task even further, as can be observed in Table 3.

A comparison in terms of average RMSE values of the three models that use the LMS rule for updating and the three robust models for the *pH* dataset is shown in Fig. 9 for

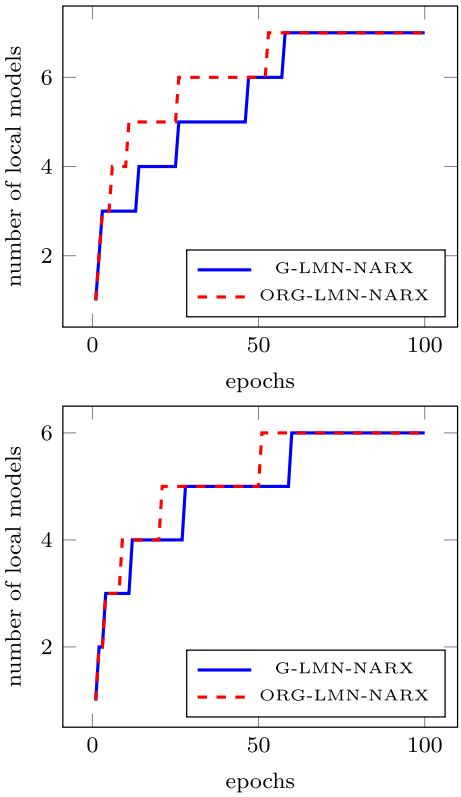

Fig. 5 Evolution of the number of local models associated with hidden units as a function of the training epochs for proposed models, G-LMN-NARX (solid blue line) and ORG-LMN-NARX (dashed red line) for the Actuator dataset (upper row), and the *pH* dataset (lower row). (Color figure online)

different contamination scenarios with *outliers*. As expected for the scenario free of *outliers*, the performance of the models is practically the same. For this scenario, the INC-LLM model (original and robust version) can be highlighted. These two models achieved lower RMSE values.

Results for the Silverbox dataset

As a final experiment, we evaluate the performance of the proposed G-LMN-NARX and ORG-LMN-NARX models on a large-scale data set. To this sake, we selected the *Silverbox* dataset [49]. The samples are obtained from an electrical circuit simulating a mass-spring-damper system, it corresponds to a nonlinear dynamical system with feedback, with a dominant linear behavior. This data set contains a total of 131,072 samples of each sequence u_n and y_n, where the first 40,000 samples of u_n and y_n were used for model building and the remaining 91,072 for model validation. Since the *Silverbox* data set is lengthy, training is not repeated for several epochs. Indeed, the model is trained in a fully online mode with recursive parameter estimation, requiring only one pass through the data in order to converge as shown in Fig. 10 for the the first 200 time steps. A closer look at this figure reveals that at the beginning of training, the prediction error is high so that the predicted time series is far from the observed series. As time goes by, the error values decrease and, hence, the predictive performance becomes better and better.

Typical examples of the prediction results achieved by the G-LMN-NARX and ORG-LMN-NARX models under free simulation regime for the *Silverbox* data set with 15% of

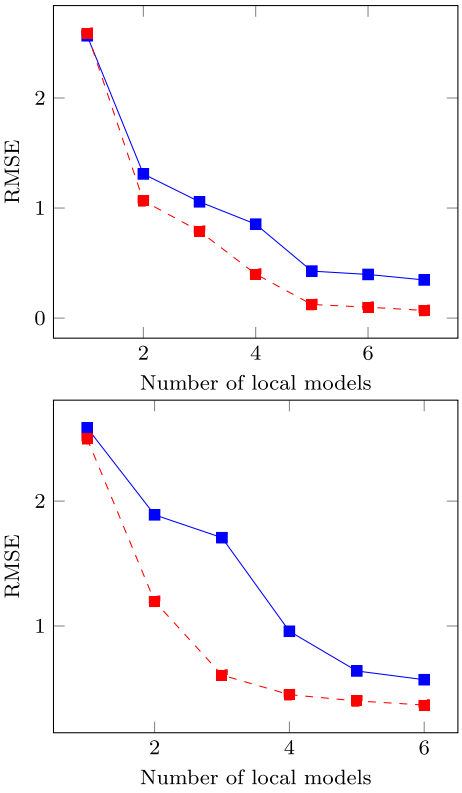

Fig. 6 The RMSE values as the number of local models increases for the proposed models, G-LMN-NARX (solid blue line) and ORG-LMN-NARX (dashed red line), for the Actuator dataset (upper row) and the *pH* dataset (lower row). (Color figure online)

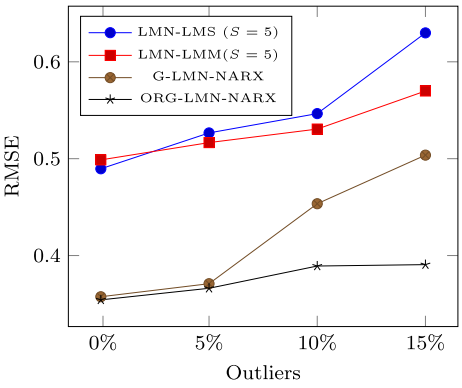

Fig. 7 RMSE values for test data as a function of the percentage of outliers in training data (*pH* dataset). The robust models perform better than the ones using the usual LMS rule. The proposed ORG-LMN-NARX model is the best performing one for this data set.

outlier contamination are shown in Fig. 11. For the sake of visualization, we show only the first 200 output samples of the validation data. For this large scale data set, for which the models were trained in an online fashion (i.e., no repeated training over several epochs), the better performance of the ORG-LMN-NARX model over its non-robust counterpart is clear.

A numerical comparison in terms of RMSE values of the two variants of the proposed model (G-LMN-NARX and ORG-LMN-NARX) for the *Silverbox* data set is shown in Table 4. As done previously, two scenarios were tested: the outlier-free scenario and one with 15% of outlier contamination. It can be observed that the deterioration of the performance

Table 2 RMSE in the test for the hydraulic actuator dataset for two levels of contamination from *outliers* (growing structures)

Models	0% Outliers		15% Outliers		Models	Epochs
	RMSE	STD	RMSE	STD		
INC-LLM-LMS	3.14E−02	1.08E−04	2.12E−01	3.65E−04	8	100
INC-LLM-LMM	2.06E−02	1.01E−04	6.06E−02	8.04E−04	7	100
RAN-LMS	4.63E−02	2.14E−05	6.45E−01	6.34E−05	7	100
RAN-LMM	2.38E−02	1.89E−05	1.18E−01	2.17E−05	7	100
G-LMN	1.36E−02	4.68E−05	2.97E−01	3.98E−05	7	100
ORG-LMN	2.19E−02	1.78E−05	4.07E−02	1.33E−05	7	100

with the increase in contamination levels is much smaller for the robust ORG-LMN-NARX model. The performances of the proposed models are also better than those achieved by fixed-size models (LMN-LMS and LMN-LMM).

Evaluation by the Kolmogorov–Smirnov (KS) test

In a sum, the KS-test computes a distance between the empirical cumulative distribution functions (CDF) of two sequences of residuals [51]. The null hypothesis to be tested is that

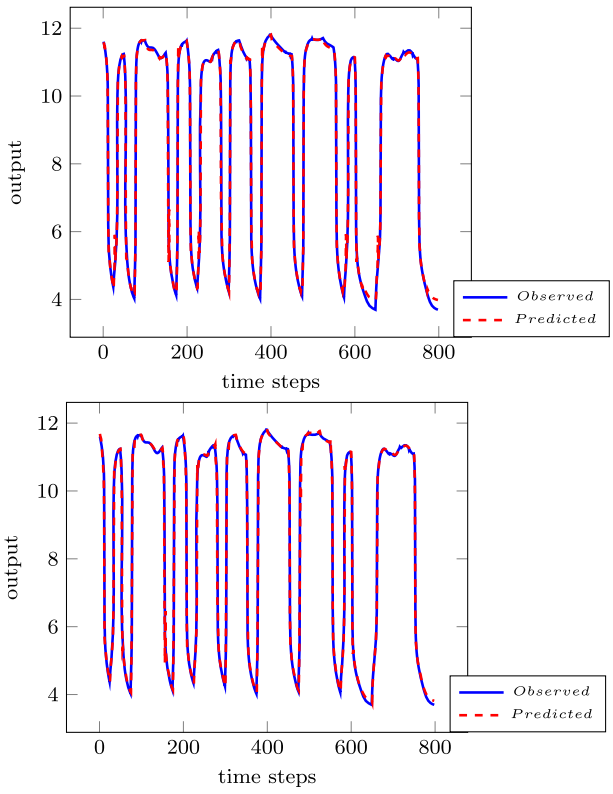

Fig. 8 Typical predictive performances of the proposed models, G-LMN-NARX (upper) and ORG-LMN-NARX (lower), after being trained with 15% of outliers (*pH* dataset)

Table 3 RMSE values for the LMN-LMM and ORG-LMN-NARX models on the *pH* data set for two levels of outlier contamination

Models	0% Outliers		15% Outliers		Models	Epochs
	RMSE	STD	RMSE	STD		
LMN-LMS (fixed-size, $S = 7$)	4.89E−01	2.64E−04	6.30E−01	1.42E−04	7	100
LMN-LMM (fixed-size, $S = 7$)	4.99E−01	1.23E−04	5.69E−01	2.96E−04	7	100
G-LMN-NARX (growing)	3.57E−01	7.56E−05	5.04E−01	8.81E−05	6	100
ORG-LMN-NARX (growing)	3.54E−01	6.69E−05	3.90E−01	7.01E−05	6	100

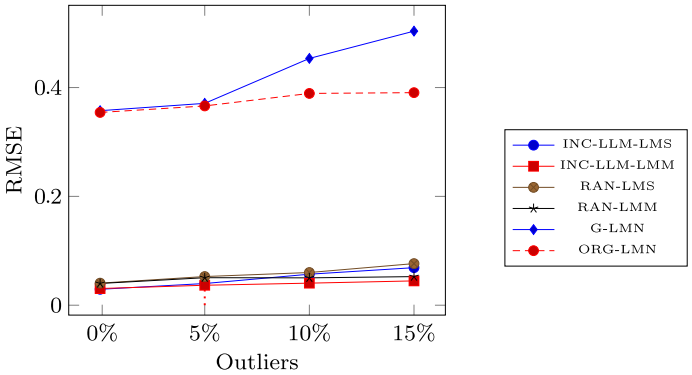

Fig. 9 RMSE values in the test as a function of the amount of *outliers* in the training data (*pH* neutralization dataset)

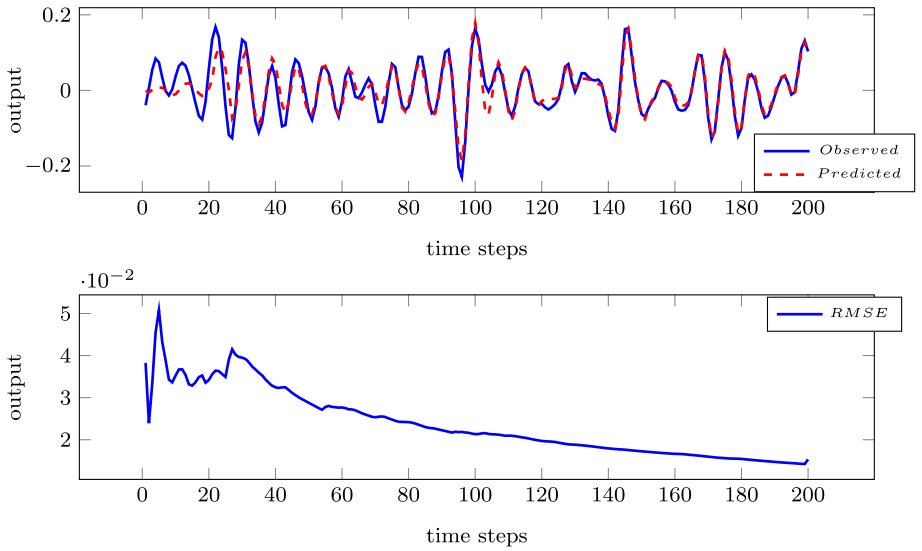

Fig. 10 Performance of the proposed model ORG-LMN-NARX during the first 200 training steps for the *Silverbox* data set. The upper figure shows the predicted output time series, while the lower figure shows the corresponding evolution of the RMSE values

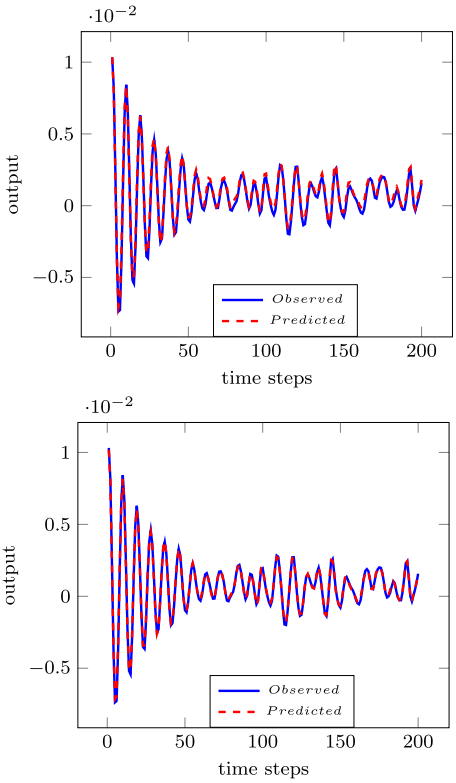

Fig. 11 Typical predictive performances of the proposed models, G-LMN-NARX (upper) and ORG-LMN-NARX (lower), after being trained with 15% of outliers (*Silverbox* data set)

Table 4 RMSE values for the G-LMN-NARX and ORG-LMN-NARX models on the *Silverbox* data set for two levels of outlier contamination

Models	0% Outliers		15% Outliers		Models	Epochs
	RMSE	STD	RMSE	STD		
LMN-LMS (fixed-size, $S = 5$)	3.98E−05	3.62E−07	5.34E−05	2.73E−07	5	1 (online)
LMN-LMM (fixed-size, $S = 5$)	3.24E−05	2.36E−07	3.59E−05	1.13E−07	5	1 (online)
G-LMN-NARX	1.50E−05	0	3.05 E−05	0	5	1 (online)
ORG-LMN-NARX	1.02E−05	0	1.18 E−05	0	5	1 (online)

the sequences are drawn from the same distribution [3]. The final set of experiments aims at evaluating the degree of similarity, from a statistical point of view, between the sequence of residuals generated by the models. The use of the KS-test in the present context is justified by the need of evaluating the difference between the performances of two models. If two allegedly different models generate statistically equivalent sequences of residuals (according to the KS-test), then the two models should be considered equivalent to each other in reality.

The application of the KS-test for the sequences of residuals produced by the two proposed models, G-LMN-NARX and ORG-LMN-NARX models, in the outlier-free scenario indicates that the models are statistically equivalent. This equivalence can be inferred qualitatively by comparing the empirical CDFs of the residuals produced by the two models in the outlier-free scenario, as shown in the upper row of Fig. 12. It is very hard to note any

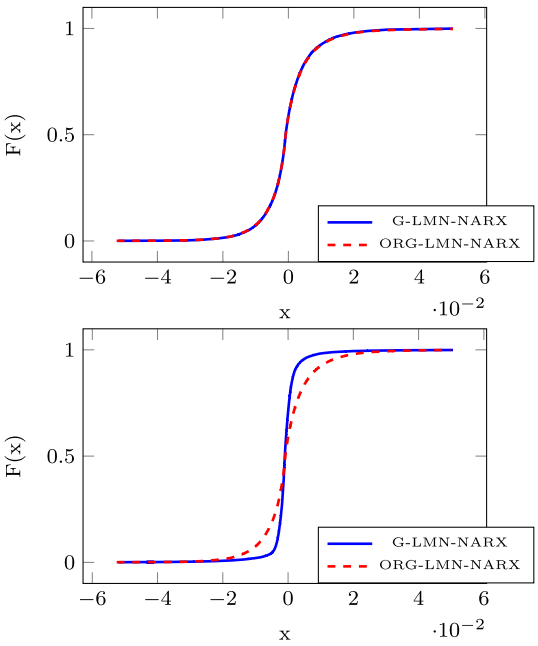

Fig. 12 Empirical CDFs of the residuals generated by the G-LMN-NARX model (solid blue line) and the ORG-LMN-NARX model (dashed red line) for the Siverbox data set. Outlier-free scenario (upper row). Outlier-contaminated scenario (lower row). (Color figure online)

Table 5 RMSE values for the G-LMN-NARX model on the Industrial Dryer (MIMO) data set for four levels of outlier contamination

Outliers (%)	G-LMN		
	y_1	y_2	y_3
0	2.35E−03 ± 0.00E0	4.43E−03 ± 0.00E0	2.27E−03 ± 0.00E0
5	2.44E−03 ± 0.00E0	4.49E−03 ± 0.00E0	2.29E−03 ± 0.00E0
10	4.54E−03 ± 0.00E0	6.40E−03 ± 0.00E0	3.03E−03 ± 0.00E0
15	9.48E−03 ± 0.00E0	7.30E−03 ± 0.00E0	5.21E−03 ± 0.00E0

Table 6 RMSE values for the ORG-LMN model on the Industrial Dryer (MIMO) data set for four levels of outlier contamination

Outliers (%)	ORG-LMN		
	y_1	y_2	y_3
0	2.27E−03 ± 0.00E0	4.27E−03 ± 0.00E0	2.26E−03 ± 0.00E0
5	2.41E−03 ± 0.00E0	4.41E−03 ± 0.00E0	2.31E−03 ± 0.00E0
10	2.49E−03 ± 0.00E0	4.50E−03 ± 0.00E0	2.39E−03 ± 0.00E0
15	2.98E−03 ± 0.00E0	4.68E−03 ± 0.00E0	2.59E−03 ± 0.00E0

difference between the two CDFs. However, for the outlier-contaminated scenario, the corresponding empirical CDFs of the residuals are shown in the lower row of Fig. 12, from which one can clearly observe a difference between the two CDFs. By this result, one can infer that the propsoed models present different predictive performances for the scenario with outlier contamination.

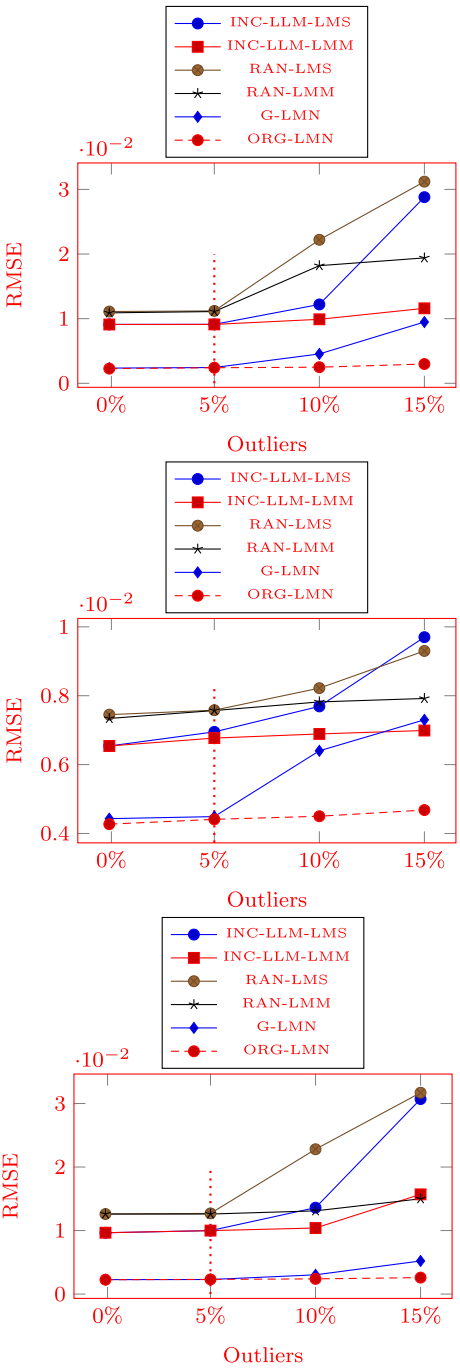

Fig. 13 RMSE values as a function of the amount of *outliers* of the training data for the output variables y_1, y_2 and y_3 (Industrial dryer MIMO data set)

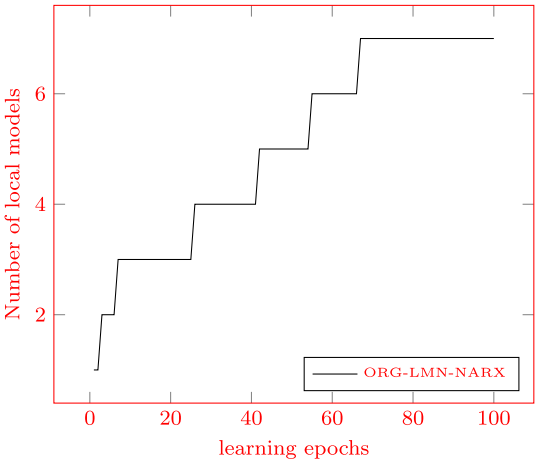

Fig. 14 Evolution of the number of local models as a function of training epochs for the proposed ORG-LMN-NARX for the industrial dryer MIMO data set

Results for the MIMO dataset

Finally, proposed approaches were also tested on a benchmarking dataset corresponding to a MIMO system. For this purpose, we selected the industrial dryer dataset described in [7] and [12] and available in the *DaISy* repository. For this data set, the values of the input variable u_1 are larger than those of the other two inputs ($u_1 \gg u_2$ and u_3); for this reason, we chose to normalize the data even before training for the range of values between 0 and 1. For the tests with this data set, the initial and final learning rates were defined as $\alpha_0 = 0.5$ and $\alpha_T = 0.001$ for all models. For the GLMN models, it was defined $\kappa = 0.9$, $\delta_{\max} = \max(\|\mathbf{x} - \mathbf{c}_0\|)$, $\delta_{\min} = \delta_{\max}/5$, $\tau = 5$, and $\epsilon = 10^{-4}$.

A numerical comparison was made in terms of RMSE values of the two proposed approaches (G-LMN-NARX and ORG-LMN-NARX). The results are shown in Tables 5 and 6. As before, different outlier-contamination scenarios were tested: one scenario free of *outliers*, the others with 5%, 10% and 15% of outlier-contamination, respectively. It can be seen that the performance deterioration with increasing contamination levels is much smaller for the ORG-LMN-NARX model for all output variables.

The performance deterioration with increasing contamination levels can also be seen in Fig. 13. The RMSE values for the output variable y_1 are shown in the upper figure, while for the output variables y_2 and y_3 the RMSE values are shown in the middle lower figures, respectively. The performance of the ORG-LMN-NARX model is practically insensitive to the presence of *outliers*. The INC-LLM-LMM model is also noteworthy regarding its sensitivity to *outliers*. In Fig. 14, we show the evolution of the number of local models as a function of training epochs for the ORG-LMN-NARX only.

Tests using the k-steps ahead prediction methodology were performed for the Industrial Dryer MIMO data set. The results for scenarios contaminated with 15% *outliers* for the ORG-LMN-NARX model only are reported in Table 7. It is possible to notice that the RMSE values for the free simulation scenario are of the same order of magnitude of those achieved for $k = 1$, 2, and 3 steps-ahead prediction, indicating that the model indeed learned the underlying dynamics of the identified system.

For the sake of completeness, typical time series for the output variables (y_1, y_2 and y_3) predicted by the ORG-LMN-NARX model in scenarios contaminated with 15% of *outliers* for the MIMO data set are shown in Fig. 15. A simpe visual inspection reveals a good matching between observed and predicted values. It should be emphasized that these results correspond

to time series which are being predicted under the free simulation regime, a considerably harder task than $k = 1$, 2, and 3 steps-ahead predictions.

Table 7 RMSE values achieved by the ORG-LMN-NARX model as a function of the prediction horizon during test phase in scenarios contaminated with 15% of *outliers*

Outliers	ORG-LMN-NARX		
	Output y_1	Output y_2	Output y_3
One-step ahead	1.58E−03 ± 0.00E0	2.17E−03 ± 0.00E0	1.12E−03 ± 0.00E0
Two-steps ahead	2.11E−03 ± 0.00E0	2.98E−03 ± 0.00E0	1.94E−03 ± 0.00E0
Three-steps ahead	2.59E−03 ± 0.00E0	3.17E−03 ± 0.00E0	2.04E−03 ± 0.00E0
Free simulation	2.98E−03 ± 0.00E0	4.68E−03 ± 0.00E0	2.59E−03 ± 0.00E0

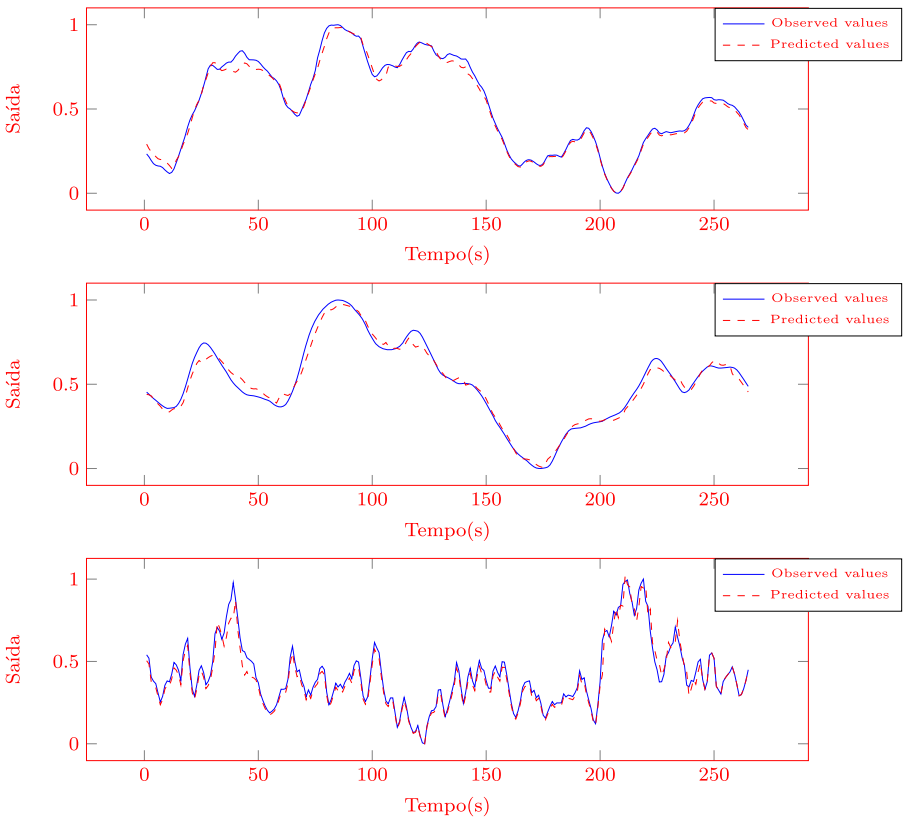

Fig. 15 Typical time series predicted under the free simulation regime by the proposed ORG-LMN-NARX model for the output variables (y_1, y_2 and y_3) in scenarios contaminated with 15% of *outliers* (industrial drier MIMO data set)

8 Conclusions and Further Work

In this paper, we introduced growing LMN-based models for online system identification in the presence of outliers. Growing models do not require the prior specification of the number of hidden units, with a new unit inserted only when required by the model's performance. We carefully addressed via a comprehensive set of numerical experiments how outliers affect the parameter estimation (i.e., learning) of the identified model. The simulations involved four bechmarking data sets (3 SISO, 1 MIMO) for dynamical system identification.

We developed the learning equations of two growing LMN-based model algorithms. This first model, named G-LMN-NARX, handles nonstationarity well but performs poorly in outlier-contaminated scenarios. Thus, an outlier-robust variant, named ORG-LMN-NARX, was then developed by introducing suitable modifications in all the learning rules. The modifications involved the replacement of the original LMS-like rule by a robust variant of it, known as the LMM rule. This simple but relevant modification was able to offer more resilience to outliers to the original G-LMN-NARX model. The performance comparison results with four benchmarking data sets revealed a considerable improvement in the performances of the proposed local models in outlier-contaminated scenarios.

We are currently evaluating the proposed ORG-LMN-NARX model as a parameterization method for the classification of different operating states of dynamical systems for fault detection purposes. This use is also being investigated to classify epileptic seizures from EEG signals and for motor imagery classification.

Acknowledgements This study was financed by the following Brazilian research funding agencies: CAPES (Finance Code 001), CNPq Grants No. 309379/2019-9 (2nd author) and 311211/2017-8 (3rd author).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Availability of data and materials The data used in the experiments reported in this paper are available for public use and can be freely downloaded for scientific purposes from the website indicated in the text.

Code availability The Octave/Matlab codes of the proposed approach are available for scientific purposes and can be provided upon request.

References

1. Angelov PP, Filev DP (2004) An approach to online identification of Takagi-Sugeno fuzzy models. IEEE Trans Syst Man Cybern Part B 34(1):484–497
2. Barreto GA, Araújo AFR (2004) Identification and control of dynamical systems using the self-organizing map. IEEE Trans Neural Netw 15(5):1244–1259
3. Barreto GA, Souza LGM (2016) Novel approaches for parameter estimation of local linear models for dynamical system identification. Appl Intell 44(1):149–165
4. Belz J, Münker T, Heinz TO, Kampmann G, Nelles O (2017) Automatic modeling with local model networks for benchmark processes. IFAC-PapersOnLine 50(1):470–475
5. Bessa JA, Barreto GA (2019) Recursive system identification using outlier-robust local models. IFAC-PapersOnLine 52(1):436–441
6. Billings SA (2013) Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. Wiley, London
7. Bittanti S, Picci G (1996) Identification, adaptation, learning: the science of learning models from data

8. Blažič S, Škrjanc I (2020) Incremental fuzzy C-regression clustering from streaming data for local-model-network identification. IEEE Trans Fuzzy Syst 28(4):758–767
9. Brunton SL, Kutz JN (2019) Data-driven science and engineering: machine learning, dynamical systems, and control. Cambridge Press, Cambridge
10. Chan S-C, Zhou Y (2010) On the performance analysis of the least mean M-estimate and normalized least mean M-estimate algorithms with gaussian inputs and additive gaussian and contaminated Gaussian noises. J Signal Process Syst 60(1):81–103
11. Chen S, Billings SA, Cowan CFN, Grant PM (1990) Non-linear systems identification using radial basis functions. Int J Syst Sci 21(12):2513–2539
12. Chou CT, Maciejowski JM (1997) System identification using balanced parametrizations. IEEE Trans Autom Control 42(7):956–974
13. Chuang C-C (2007) Fuzzy weighted support vector regression with a fuzzy partition. IEEE Trans Syst Man Cybern Part B 37(3):630–640
14. Costa TV, Fileti AMF, Oliveira-Lopes LC, Silva FV (2015) Experimental assessment and design of multiple model predictive control based on local model networks for industrial processes. Evol Syst 6(4):243–253
15. Foss BA, Johansen TA (1993) , On local and fuzzy modelling. In: Third International Conference on Industrial Fuzzy Control and Intelligent Systems (IFIS'93), pp 80–87
16. Fritzke B (1995) A growing neural gas network learns topologies. In: Advances in neural information processing systems 7. MIT Press, pp 625–632
17. Fritzke B (1995) Incremental learning of local linear mappings. In: Proceedings of the international conference on artificial neural networks (ICANN'95'), pp. 217–222
18. Fuangkhon P (2014) An incremental learning preprocessor for feed-forward neural network. Artif Intell Rev 41(2):183–210
19. Hampel FR, Ronchetti EM, Rousseeuw PJ, Stahel WA (2011) Robust statistics: the approach based on influence functions, vol. 114. Wiley
20. Han H, Chen Q, Qiao J (2010) Research on an online self-organizing radial basis function neural network. Neural Comput Appl 19:667–676
21. Hassibi B, Sayed AH, Kailath T (1996) H_∞ optimality of the LMS algorithm. IEEE Trans Signal Process 44(2):267–280
22. He H, Chen S, Li K, Xu X (2011) Incremental learning from stream data. IEEE Trans Neural Netw 22(12):1901–1914
23. Huber P (1964) Robust estimation of a location parameter. Ann Math Stat 35(1):73–101
24. Jain S, Bandyopadhyay S (2017) Resource allocation network for segregated targeting problems with dedicated sources. Ind Eng Chem Res 56(46):13831–13843
25. Johansen TA, Foss BA (1992) A NARMAX model representation for adaptive control based on local models. Model Identif Control 13(1):25
26. Jones RD, Lee YC, Barnes CW, Flake GW, Lee K, Lewis PS, Qian S (1990) Function approximation and time series prediction with neural networks. In: Proceedings of the 1990 international joint conference on neural networks (IJCNN'1990)', pp 649–665
27. Kiumarsi B, Lewis FL, Levine SD (2015) Optimal control of nonlinear discrete time-varying systems using a new neural network approximation structure. Neurocomputing 156:157–165
28. Kohonen T (2013) Essentials of the self-organizing map. Neural Netw 37:52–65
29. König O, Hametner C, Prochart G, Jakubek S (2014) Battery emulation for power-HIL using local model networks and robust impedance control. IEEE Trans Ind Electron 61(2):943–955
30. Kovačević B, Banjac Z, Kovačević IK (2016) Robust adaptive filtering using recursive weighted least squares with combined scale and variable forgetting factors. EURASIP J Adv Signal Process 37:1–22
31. Liu J, Djurdjanovic D, Marko K, Ni J (2009) Growing structure multiple model systems for anomaly detection and fault diagnosis. J Dyn Syst Meas Contr 131(5):1–13
32. Ljung L (1999) System identification: theory for the user, 2nd edn. Prentice Hall
33. Lu T, Pan L, Jiangs J, Zhang Y, Xiong Z (2017) Dlml: deep linear mappings learning for face super-resolution with nonlocal-patch. In: 2017 IEEE international conference on multimedia and expo (ICME), IEEE, pp 1362–1367
34. Lv C, Zhang T, Ma F, Yue D (2018) A very short-term online forecasting model for photovoltaic power based on two-stage resource allocation network. In: 2018 International joint conference on neural networks (IJCNN), IEEE, pp 1–6
35. Mahanand BS, Suresh S, Sundararajan N, Kumar MA (2012) Identification of brain regions responsible for Alzheimer's disease using a self-adaptive resource allocation network. Neural Netw 32:313–322
36. Maier CC, Schirrer A, Kozek M (2018) Real-time capable nonlinear pantograph models using local model networks in state-space configuration. Mechatronics 50:292–302

37. Mandic DP, Kanna S, Constantinides AG (2015) On the intrinsic relationship between the least mean square and Kalman filters. IEEE Signal Process Mag 32(6):117–122
38. Martinetz T (1993) , Competitive hebbian learning rule forms perfectly topology preserving maps. In: ICANN'93, Springer, pp 427–434
39. Mattos CLC, Dai Z, Damianou A, Barreto GA, Lawrence ND (2017) Deep recurrent gaussian processes for outlier-robust system identification. J Process Control 60:82–94
40. Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1(2):281–294
41. Moshou D, Ramon H (1997) Extended self-organizing maps with local linear mappings for function approximation and system identification. In: Proceedings of the 1st workshop on self-organizing maps (WSOM'97), pp 1–6
42. Münker T, Heinz TO, Nelles O (2017) , Hierarchical model predictive control for local model networks. In: Proceedings of the American Control Conference (ACC'2017), pp 5026–5031
43. Narendra K, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27
44. Pai PS, Nagabhushana T, Rao PR (2001) Tool wear estimation using resource allocation network. Int J Mach Tools Manuf 41(5):673–685
45. Platt J (1991) A resource-allocating network for function interpolation. Neural Comput 3(2):213–225
46. Poggio T, Girosi F (1990) Networks for approximation and learning. Proc IEEE 78(9):1484–1487
47. Prasad G, Swidenbank E, Hogg BW (1998) A local model networks based multivariable long-range predictive control strategy for thermal power plants. Automatica 34(10):1185–1204
48. Ribeiro AH, Aguirre LA (2018) Parallel training considered harmful?: Comparing series-parallel and parallel feedforward network training. Neurocomputing 316:222–231
49. Schoukens J, Nemeth JG, Crama P, Rolain Y, Pintelon R (2003) Fast approximate identification of nonlinear systems. Automatica 39(7):1267–1274
50. Sjöberg J, Zhang Q, Ljung L, Benveniste A, Delyon B, Glorennec P-Y, Hjalmarsson H, Juditsky A (1995) Nonlinear black-box modeling in system identification: a unified overview. Automatica 31(12):1691–1724
51. Soong TT (2004) Fundamentals of probability and statistics for engineers. Wiley, London
52. Stokbro K, Umberger DK, Hertz JA (1990) Exploiting neurons with localized receptive fields to learn chaos. Complex Syst 4(3):603–622
53. Suresh S, Dong K, Kim HJ (2010) A sequential learning algorithm for self-adaptive resource allocation network classifier. Neurocomputing 73(16–18):3012–3019
54. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modelling and control. IEEE Trans Syst Man Cybern 15(1):116–132
55. Vachkov G (2004) , Growing model algorithm for process identification based on neural-gas learning and local linear mapping. In: Fourth international conference on hybrid intelligent systems (HIS'04), IEEE, pp 222–227
56. Walter J, Ritter H, Schulten K (1990) , Nonlinear prediction with self-organizing maps. In: Proceedings of the IEEE international joint conference on neural networks (IJCNN'90), pp 589–594
57. Wang Q, Wang X, Sun B (2019) Resource allocation network and PID control based on automated guided vehicles. Acad J Manuf Eng 17(2)
58. Widrow B (2005) Thinking about thinking: the discovery of the LMS algorithm. IEEE Signal Process Mag 22(1):100–106
59. Widrow B, Lehr MA (1990) 30 years of adaptive neural networks: perceptron, madaline and backpropagation. Proc IEEE 78(9):1415–1442
60. Yan L, Sundararajan N, Saratchandran P (2000) Nonlinear system identification using Lyapunov based fully tuned dynamic RBF networks. Neural Process Lett 12(3):291–303
61. Zhao Z-L, Qiu Z-C, Zhang X-M, Han J-D (2016) Vibration control of a pneumatic driven piezoelectric flexible manipulator using self-organizing map based multiple models. Mech Syst Signal Process 70:345–372
62. Zou Y, Chan S-C, Ng T-S (2000) Least mean M-estimate algorithms for robust adaptive filtering in impulse noise. IEEE Trans Circuits Syst II: Analog Digit Signal Process 47(12):1564–1569
63. Zou Y, Chan SC, Ng TS (2001) Robust M-estimate adaptive filtering. IEE Proc Vis Image Signal Process 148(4):289–294