Nonlinear Dyn (2024) 112:5525–5540
https://doi.org/10.1007/s11071-024-09360-x

ORIGINAL PAPER

Online neuro-fuzzy model learning of dynamic systems with measurement noise

Wen Gu · Jianglin Lan · Byron Mason

Received: 3 February 2023 / Accepted: 22 December 2023 / Published online: 6 February 2024
© The Author(s) 2024

Abstract Model identification of nonlinear time varying dynamic systems is challenging because the system behaviours may vary significantly in different operational conditions. If the changes are insufficiently captured by training data, the trained model is unable to capture the system response well when the operational condition changes. The model performance may also be deteriorated in real-time implementation due to the noise in sensors or the environment. This paper presents a self-adaptive Neuro-Fuzzy (NF) modelling framework to address these challenges. The NF model, trained offline based on experimental data, combines the Auto-Regressive with eXogenous (ARX) models and Gaussian activation functions to capture the nonlinear system behaviours. During online implementation, the ARX model parameters are updated using new data through a recursive generalised least squares method, which embeds a noise model to eliminate effects of the noise. The online updating algorithm has a provable convergence guarantee and enables the proposed NF model to adapt to changes in system behaviours automatically. Efficacy of the algorithm is verified through two numerical examples and an experiment on a commercial automotive engine.

Keywords Neuro-Fuzzy model · Recursive identification · Online learning · Measurement noise

1 Introduction

Model identification of dynamic systems is challenging since most engineering systems are nonlinear and time-varying with dynamic uncertainties [6]. The problem becomes even more difficult for high dimensional and complex systems [14]. Neural network has been a popular tool for modelling since it is capable of representing the nonlinearity between inputs and outputs [8,28,42]. However, the lack of transparency and the curse of dimensionality restrict its implementation, particularly when it is used for controller designs [1,3]. Fuzzy models have gained popularity in dynamic system modelling [12,46,49], by combining prior human knowledge with information from data. This feature offers high modelling accuracy with interpretability and linguistic significance. The Neuro-Fuzzy (NF) model technique has been proposed in the literature to incorporate the merits of both neural network and fuzzy reasoning [4]. The basic idea is to firstly split the system input space, which describes the system operational conditions, into several regions. Then for each region, a local model is used to capture the local

W. Gu · B. Mason (✉)
Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough LE11 3TU, UK
e-mail: b.mason2@lboro.ac.uk

W. Gu
e-mail: wengu890327@gmail.com

J. Lan
James Watt School of Engineering, University of Glasgow, Glasgow G12 8QQ, UK
e-mail: jianglin.lan@glasgow.ac.uk

system behaviour accurately. Finally, nonlinear fuzzy activation functions are used to aggregate all the local models to capture the overall system behaviour. The NF technique has been shown to be highly promising not only in the area of nonlinear dynamic systems modelling, but also in various other application areas, such as observer design [29], control design [18,41,44] and fault detection [4].

Many NF methods have been proposed for modelling nonlinear systems, such as Takagi-Sugeno fuzzy neural networks [47], local model networks [37,38], wavelet fuzzy neural networks [20], local NF models [34] and adaptive neuro-fuzzy inference systems (ANFIS) [23,24]. However, challenges still exist when implementing the built NF models are in real-time systems operations and control. One challenge is overfitting, which refers to the phenomenon that growing the number of neurons can fit the training data as accurately as possible, but it may deteriorate the model performance when new data is imported [52]. Besides, since only partial data are used for offline model training, the model cannot cover the entire system operational conditions [13]. Hence, an important area of current research is developing NF models with autonomy to evolve their structures and parameters to capture changes in the system dynamics based on real-time measurement data. A further challenge is that measurement error or sampling noise, owing to the high intrinsic variability in the underlying data, will deteriorate model performance [22,39]. Therefore, this paper aims to advance the state-of-the-art NF modelling methods for nonlinear dynamic systems through developing an online NF model updating algorithm based on noisy sensor measurements.

Most of the popular NF modelling methods like the ANFIS are incapable of online learning [33,51]. The evolving learning algorithms for Takagi-Sugeno models are introduced in [2,13,27,30,32], where new fuzzy rules are generated through online clustering. However, in these evolving learning methods, all the fuzzy rules and even those irrelevant to the present measurement are overhauled in each update [31,52], which violates the minimum disturbance principle for learning [25,45]. The computational demand of evolving learning methods is also high [7]. Hence, online update of local parameters is more preferable than update of model structure in applications [9,16,50]. The forgetting factor technique [26] can update local parameters with fast convergence and small estimation errors, but it is limited to simple systems with weak nonlinearity. The work [11] proposes a multi-layer evolving NF model for complex systems and utilises error backpropagation to self-update the model parameters. The recursive least squares method [52] is used to update only the model parameters relevant to new measurements. However, all the above works do not consider the effect of noisy data on the effectiveness of the NF model updating.

The work [40] uses a filtered recursive least squares method to identify the NF model parameters by using noisy data with a known distribution. The generalised total least squares method can also be used for modelling single-input single-output systems [48] and multi-input single-output systems [43] with noisy inputs and outputs, but they are not for NF model updating. The generalised total least squares algorithm is extended in [21] for training an NF model with partial input signals corrupted by noise and also in [19] considering multiple types of noises. However, the method in [21] needs prior knowledge of the noise variances, while the method in [19] assumes that the noise distribution is known, which restricts their online application. Lughofer [31] proposes a recursive weighted total least squares method for estimating NF model parameters with noisy input data without requiring prior knowledge of the noise variances or distribution. However, their work needs an extra algorithm called Incremental Polynomial Kalman Smoother for recursively estimating the noise covariance matrix, which improves the design complexity. Moreover, all the existing methods [19,21,31,40] do not consider more complex systems with time-varying dynamics and changing operational conditions. Therefore, there is a need for new algorithms to update the NF model parameters for complex nonlinear systems based on noisy data, without requiring a prior knowledge or complex design for dealing with the noise.

To overcome the above limitations in the existing methods, this paper presents a novel NF model learning method. Compared to the state-of-the-art methods [19,21,31,40], the main contributions of this paper are summarised as follows:

(1) An online NF modelling strategy is developed for multi-input single-output dynamic systems based on noisy data. The strategy addresses modelling of nonlinear systems with time-varying dynamics and changing operational conditions, which is not

studied in [19,21,31,40]. The proposed method is directly applicable for multi-input multi-output systems by presenting them using a set of multi-input single-output models.

(2) A recursive generalised least squares (RGLS) algorithm is proposed for updating the NF model parameters online to adapt to the time-varying dynamics. Through embedding a noise model, the proposed RGLS design is robust to measurement noise that is non-stationary and heteroscedastic, without requiring the known covariances or distribution of noise as in [19,21,40], or an extra algorithm for estimating the noise covariances as in [31].

(3) An experimental testing on a commercial heavy-duty diesel engine is conducted to verify the efficacy and practical applicability of the proposed modelling method.

The rest of this paper is organised as follows. Section 2 describes the NF model. Section 3 presents the RGLS-based online learning algorithm, followed by its convergence analysis provided in Sect. 4. The NF model learning method is evaluated using two numerical examples in Sect. 5 and an experimental testing on a heavy-duty diesel engine in Sect. 6. Conclusions are made in Sect. 7.

2 Basics of Neuro-Fuzzy model

Consider a dynamic system described in discrete-time form as

$$y(k) = f(u, y, k) \tag{1}$$

where $y \in \mathcal{R}$ is the true system output without noise. $u \in \mathcal{R}^p$ and k denote the inputs and sampling index, respectively. This system can be represented by an NF model consisting of several neurons, where each neuron has a local ARX model associated with its fuzzy activation function. The architecture of the NF model is illustrated in Fig. 1, with each component detailed in the following subsections.

2.1 NF model description

Each part of the NF model is described as follows:

(i) *Input channels*:

$$Y(k) = \left[y(k-1), y(k-2), \cdots, y(k-d_y)\right] \tag{2}$$

$$U_i(k) = [u_i(k-1), u_i(k-2), \cdots, u_i(k-d_{ui})] \tag{3}$$

where u_i denotes the i-th input; $y(k-m)$, $m = 1, \ldots, d_y$, are the m-th delayed term of $y(k)$; $u_i(k-m)$, $m = 1, \ldots, d_{ui}$, are the m-th delayed term of $u_i(k)$. The positive integers d_y and d_{ui} are the corresponding maximum lags for $y(k)$ and $u_i(k)$, respectively. The vectors $Y(k)$ and $U_i(k)$, $i = 1, \ldots, p$, are fixed once the modelling framework is determined.

(ii) *Regressor*:

$$\varphi(k) = \left[1, \bar{y}(k), \bar{u}_1(k), \cdots, \bar{u}_i(k), \cdots, \bar{u}_p(k)\right] \tag{4}$$

$$\tilde{\varphi}(k) = [\tilde{y}(k), \tilde{u}_1(k), \cdots, \tilde{u}_i(k), \cdots, \tilde{u}_p(k)] \tag{5}$$

where $\varphi(k) \in \mathcal{R}^{1+r}$ and $\tilde{\varphi}(k) \in \mathcal{R}^{\tilde{r}}$ denote the regressors for the local ARX model and fuzzy activation function, respectively. The positive integer r is the number of entries in $\varphi(k)$ excluding the offset 1 and $\tilde{r}$ is the number of entries in $\tilde{\varphi}(k)$. The regressors satisfy $\bar{y}(k) \subseteq Y(k)$, $\bar{u}_i(k) \subseteq U_i(k)$, $\tilde{y}(k) \subseteq Y(k)$ and $\tilde{u}_i(k) \subseteq U_i(k)$. The entries in $\bar{y}(k)$, $\tilde{y}(k)$, $\bar{u}_i(k)$ and $\tilde{u}_i(k)$ can be chosen differently, and are fixed during modelling.

All the input channels in (2) and (3) having a linear effect on the process are gathered in $\varphi(k)$ to calculate the local model output. The input channels that influence the process in a nonlinear way are gathered in $\tilde{\varphi}(k)$ to determine the weight value of the local model output. Since only a part of the input channels are associated with the nonlinearity of the system [36], $\tilde{\varphi}(k)$ and $\varphi(k)$ can be chosen differently. As $\tilde{\varphi}(k)$ determines the system input space, the model complexity can be reduced by decreasing the number of entries $\tilde{r}$ in $\tilde{\varphi}(k)$.

(iii) *Neurons*:

(a) The NF model splits the input space of the dynamic system into several regions. Local ARX models are used to represent the dynamic behaviour of the system in each region. The output of the j-th, $j = 1, \ldots, N$, local model is represented as

$$\hat{y}_j(k) = \varphi^\top(k)\theta_j \tag{6}$$

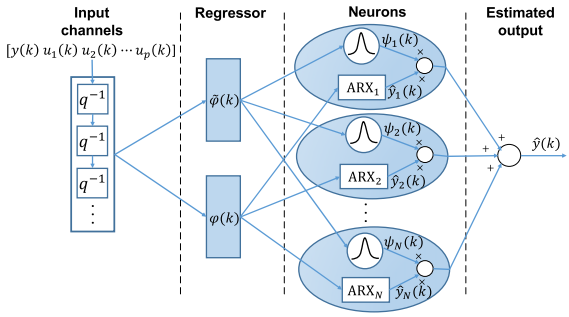

Fig. 1 Architecture of the NF model

where $\theta_j = [a_{j,0}, a_{j,1}, \cdots, a_{j,r}]^\top$ is the vector of local model parameters.

(b) Based on the location of the operating point within the input space, the fuzzy activation function $\psi_j(k)$ defines the contribution of the local model output $\hat{y}_j(k)$, $j = 1, \ldots, N$, to the overall NF model output. The activation functions satisfy

$$\sum_{j=1}^{N} \psi_j(k) = 1 \tag{7}$$

where N is the total number of neurons in the NF model.

(iv) *Estimated output*: The output of the NF model $\hat{y}(k)$ is calculated as a weighted sum of local model outputs:

$$\hat{y}(k) = \sum_{j=1}^{N} \psi_j(k)\hat{y}_j(k). \tag{8}$$

The processes of the space partition and local model construction are described in the following sections.

2.2 Fuzzy rules construction by space partition

A partitioning strategy is required to split the input space of the system such that the local model (6) represents the system behaviour in its corresponding region. Moreover, a smooth transition of local models among different regions is desired.

To avoid time-consuming nonlinear optimisation, the incremental tree-construction algorithm [10] is used to split the input space. The input space defined by the combination of input channels in $\tilde{\varphi}(k)$ can be treated as a hyperplane. Each input channel determines a potential plane splitting. The modelling algorithm iteratively

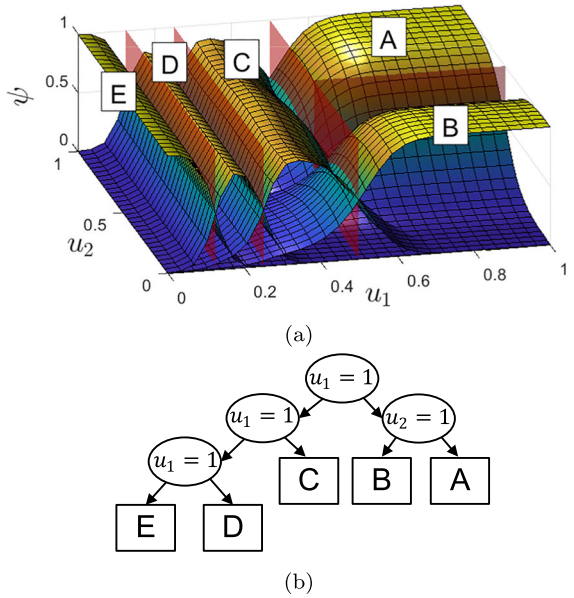

Fig. 2 Input space partitioning of two inputs system: **a** Fuzzy inference and **b** Incremental tree-construction algorithm

bisects each plane in the region holding the worst performance. In each division, the activation functions representing fuzzy rules are constructed first. Subsequently, the parameters for the two newly generated local models are estimated. The algorithm selects the best splitting result based on the overall model fidelity. To illustrate the principle of space partition, we use Fig. 2 to show the two-dimensional partition of the static system

$$y(k) = \frac{1}{0.1 + u_1(k)} + (2u_2(k))^2. \tag{9}$$

The values of five different activation functions describing the system nonlinearity are shown in Fig. 2a. The activation functions are denoted as A, B, C, D and E. The space partition process is illustrated in Fig. 2b. When $u_1 = 1$, the model accuracy with partition on u_1 dimension is the highest, and vice versa. By using the above space partition, the normalised Gaussian is chosen for the activation function in the j-th region and is defined as

$$\psi_j(k) = \frac{\mu_j(\tilde{\varphi}(k))}{\sum_{i=1}^{N} \mu_i(\tilde{\varphi}(k))} \tag{10}$$

with

$$\mu_j(\tilde{\varphi}(k)) = \exp\left(\sum_{i=1}^{\tilde{r}} -\frac{1}{2}\left(\frac{\tilde{\varphi}_i(k) - c_{j,i}}{\lambda_{j,i}}\right)^2\right) \tag{11}$$

where $\tilde{\varphi}_i(k)$ is the i-th entry in $\tilde{\varphi}(k)$. The corresponding centres and standard deviations of the Gaussian functions are denoted by $c_{j,i}$ and $\lambda_{j,i}$, respectively. The calculations of $c_{j,i}$ and $\lambda_{j,i}$ are based on the lower and upper boundary of $\tilde{\varphi}_i(k)$ in the j-th region, the details of which are referred to [36].

The incremental tree-construction algorithm iteratively improves the model accuracy until the termination criterion is met. The criterion can be the minimum modelling error or the maximum model complexity.

2.3 Local model parameters estimation

For fixed activation functions, estimation of the local ARX model parameters can be formulated as a linear optimisation problem. Two different approaches are recognised for optimising local model parameters: concurrent estimation and regional estimation. For the concurrent estimation, all the local model parameters are estimated simultaneously through the least squares (LS) method. The regional estimation neglects the interactions among the neighboured local models, and it estimates the parameters of each local model separately via the weighted least squares (WLS) method. The regional estimation can reduce the variance when the training data contains noise, resulting in an estimation with better robustness against the noise [38]. It is therefore adopted for model training in this paper.

The local model in (6) is only valid in the j-th region where the value of $\psi_j(k)$ is close to 1. It means that the operating point is close to the centre $c_{j,i},\ i = 1, \ldots, \tilde{r}$, of the activation functions. Therefore, the corresponding $\varphi(k)$ is highly relevant to the estimate θ_j. Consequently, the WLS is applied to estimate the local model parameters.

For a given set of training data with the length of M, the local ARX model parameters can be obtained by minimising the local weighted objective function

$$J(\theta_j) = \left[y - \varphi^\top \theta_j\right]^\top Q_j \left[y - \varphi^\top \theta_j\right] \tag{12}$$

with the output vector and the corresponding regression matrix given by

$$y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(M) \end{bmatrix},\ \varphi = \begin{bmatrix} \varphi^\top(1) \\ \varphi^\top(2) \\ \vdots \\ \varphi^\top(M) \end{bmatrix} \tag{13}$$

and the following $M \times M$ diagonal weighting matrix calculated from (10) and (11):

$$Q_j = \begin{bmatrix} \psi_j(1) & 0 & \ldots & 0 \\ 0 & \psi_j(2) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \psi_j(M) \end{bmatrix}. \tag{14}$$

The optimal solution θ_j to (12) satisfies $\frac{\partial J_i}{\partial \theta_j} = 0$ and can be obtained by applying a weighted pseudo-inverse [2]:

$$\theta_j = \left(\varphi Q_j \varphi^\top\right)^{-1} \varphi Q_j y. \tag{15}$$

The weighting factors $\psi_j(k),\ k = 1, \ldots, M$, given by (10) determines the relevancy of the training data to the j-th region. If the weighting factor is 1, the WLS algorithm performs the same as the LS estimation.

3 Recursive-generalised-least-squares (RGLS) based learning algorithm

The NF model should hold self-learning capability to guarantee its performance in real-time implementation. However, updating the complete model structure and parameters violates the minimum disturbance principle for learning [25,45]. Hence, the NF model structure is fixed during online learning. Nevertheless, the model still needs to be adaptive against noise and small changes such as encountering new operational conditions. Since the model can adapt to the local context rapidly by utilising the prior knowledge stored in the initial model, an initial model can be constructed offline first. When the initial model is deployed online, the RGLS-based learning algorithm is adopted to update the local model parameters.

The dynamics in (1) under noise are described by

$$y_{\text{noise}} = f(u, y, k) + w(k) \tag{16}$$

where $y_{\text{noise}}(k)$ is the measured system output with noise and $w(k)$ is the heteroscedastic noise which can be described as

$$w(k) = \frac{1}{c(z)} v(k) \tag{17}$$

while $v(k)$ is the stochastic white noise with zero mean and $c(z)$ is expressed as

$$c(z) = 1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_q z^{-q} \tag{18}$$

where z is the backshift operator moving the time index of an observation back by one sampling period, and

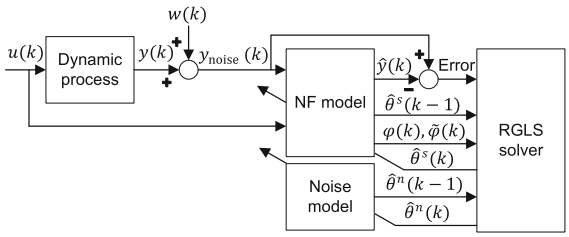

Fig. 3 Schematic of the RGLS-based online NF model learning

the positive integer q denotes the maximum delay. Since the type of noise considered in this paper is non-stationary and heteroscedastic, the proposed modelling method will be applicable to simpler noises such as stationary noise and homoscedastic noise. The generalisation of the method to a wider range of noise will be explored in future work.

The online learning framework is depicted in Fig. 3. The objective of the proposed framework is to update local model parameters $\hat{\theta}^s(k)$ of the NF model under the heteroscedastic noise at sampling index k. To correctly estimate the local model parameters, the effect of the noise is compensated by utilising the noise covariance provided by a noise model. The idea of embedding a noise model to compensate for its effect is inspired by the auxiliary model theory [53] without requiring a priori information (e.g., distribution and variances) of the noise. The noise model $\hat{\theta}^n(k)$ is identified online based on noisy measurements, where the characteristics of the noise are usually unknown and time-varying. To accurately compensate for the noise, the local model parameters of the NF model $\hat{\theta}^s(k)$ and the noise model parameters $\hat{\theta}^n(k)$ are updated simultaneously online using the proposed RGLS algorithm. Consequently, the updated NF model can describe the real system behaviour by removing the noise from measurements.

3.1 Offline NF model training

Noise-free training data is used for the offline model training, so the noise model is deactivated. The vectors $\bar{y}(k)$, $\bar{u}_i(k)$, $\tilde{y}(k)$ and $\tilde{u}_i(k)$, $i = 1, \ldots, p$, are chosen via trial-and-error and fixed in online learning. A set of training data is collected to learn the behaviour of the dynamic system (1). Based on Sects. 2.2 and 2.3, the offline NF model training is summarised in Algorithm 1.

Algorithm 1 Offline NF model training

1: Collect the training data and form $Y(k), U_i(k)$, $i = 1, \ldots, p$, $k = 1, \ldots, M$, using (2) and (3).
2: Construct regressors $\varphi(k)$ and $\tilde{\varphi}(k)$, $k = 1, \ldots, M$, using (4) and (5), respectively.
3: Generate the first neuron using (15).
4: Calculate the model output $\hat{y}$ using (8).
5: Calculate the global error $E = \sum_{k=1}^{M} (y(k) - \hat{y}(k))^2$.
6: Calculate the local error $e_j = \sum_{k=1}^{M} \psi_j(k)(y(k) - \hat{y}(k))^2$, where $j = 1$ since only one neuron exists, and $\psi_1(k) = 1$.
7: **while** $E > E_{\min}$ or $N < N_{\max}$ **do**
8: Bisect the worst neuron with the largest local error e_j into two neurons.
9: **for** $i = 1 : \tilde{r}$ **do**
10: Determine the activation function values ψ for two new neurons using (10) and (11).
11: Calculate the local model parameters θ for two new neurons using (15).
12: Calculate the model output $\hat{y}$ using (8).
13: Calculate the global error E_i.
14: **if** $i = 1$ or $E_i < E_{i-1}$, $i = 2, \ldots, \tilde{r}$, **then**
15: Save the results.
16: **else**
17: Go to step 7.
18: **end if**
19: **end for**
20: $N = N + 1$.
21: **end while**

▷ $E_{\min}$ and $N_{\max}$ are minimum modelling error and maximum number of neurons, respectively. E_i denotes the global error of the NF model with partition on the $\tilde{\varphi}_i$ dimension.

Define the parameter vector of all local ARX models as

$$\theta^s = \left[\theta_1^\top, \theta_2^\top, \cdots, \theta_N^\top\right]^\top. \tag{19}$$

The information vector consisting of the weighted regressor of each local model calculated using (4), (10) and (11) is given by

$$\begin{aligned} \varphi_j(k) &= \psi_j(k)\,\varphi(k), \\ \varphi^s(k) &= \left[\varphi_1^\top(k), \varphi_2^\top(k), \cdots, \varphi_N^\top(k)\right]^\top. \end{aligned} \tag{20}$$

By using (19) and (20), the NF model (8) is reformulated as

$$\hat{y}(k) = \left(\varphi^s(k)\right)^\top \theta^s. \tag{21}$$

3.2 Online NF model learning

As discussed in Sect. 2.3, although regional estimation enhances the NF model's robustness against noise, bias in the estimation of the model parameters is increased if the input space partitioning is poor. When the model

structure is well chosen in the initial training, the concurrent estimation is more accurate than the regional estimation [35]. Therefore, the concurrent estimation method is adopted for online model learning.

For a set of time-series experimental data, correlations exist between observations. Therefore, an autoregressive (AR) model is chosen to identify modelling errors and noise online. The measured system output $y_{\text{noise}}(k)$ in (16) is estimated by a NF model combined with a noise model as follows:

$$\hat{y}_{\text{noise}}(k) = \hat{y}(k) + \hat{w}(k) \tag{22}$$

where the estimated noise is characterised by an AR model

$$\hat{w}(k) = -\sum_{i=1}^{n_c} c_i \hat{w}(k-i) = \left(\varphi^n(k)\right)^\top \theta^n \tag{23}$$

with n_c being the order of the model. The AR model is chosen due to its simplicity and low computational requirement, compared to the moving average model or autoregressive moving average model [5].

Define the noise model parameter vector as

$$\theta^n = \left[c_1, c_2, \cdots, c_{n_c}\right]^\top \tag{24}$$

and the information vector as

$$\varphi^n(k) = [-w(k-1), -w(k-2), \cdots, -w(k-n_c)]^\top. \tag{25}$$

Finally, the system (22) can be rewritten as

$$\begin{aligned} \hat{y}_{\text{noise}}(k) &= \sum_{j=1}^{N} \hat{y}_j(k) + \hat{w}(k) \\ &= \left(\varphi^s(k)\right)^\top \theta^s + \left(\varphi^n(k)\right)^\top \theta^n \\ &= \varphi^\top(k)\theta \end{aligned} \tag{26}$$

where

$$\theta = \begin{bmatrix} \theta^s \\ \theta^n \end{bmatrix}, \ \varphi(k) = \begin{bmatrix} \varphi^s(k) \\ \varphi^n(k) \end{bmatrix}. \tag{27}$$

To estimate the unknown parameter θ, consider a quadratic cost function

$$J(\theta) = \sum_{i=1}^{k} \left[y_{\text{noise}}(i) - \varphi^\top(i)\theta\right]^2. \tag{28}$$

The cost function can be minimised by satisfying $\frac{\partial J(\theta)}{\partial \theta} = 0$, which gives the LS estimate of the optimal θ as

$$\hat{\theta}(k) = \left[\sum_{i=1}^{k} \varphi(i)\varphi^\top(i)\right]^{-1} \left[\sum_{i=1}^{k} \varphi(i) y_{\text{noise}}(i)\right]. \tag{29}$$

The optimal estimate $\hat{\theta}(k)$ can be computed recursively as

$$\hat{\theta}(k) = P(k)\xi(k) \tag{30}$$

where the covariance matrix $P(k)$ and the vector $\xi(k)$ are given by

$$\begin{aligned} P^{-1}(k) &= \sum_{i=1}^{k} \varphi(i)\varphi^\top(i) \\ &= P^{-1}(k-1) + \varphi(k)\varphi^\top(k) \end{aligned} \tag{31}$$

$$\begin{aligned} \xi(k) &= \sum_{i=1}^{k} \varphi(i) y_{\text{noise}}(i) \\ &= \xi(k-1) + \varphi(k) y_{\text{noise}}(k). \end{aligned} \tag{32}$$

By applying the matrix inverse lemma [17]: $(A + BC)^{-1} = A^{-1} - A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1}$ to (31), with $A = P^{-1}(k-1)$, $B = \varphi(k)$ and $C = \varphi^\top(k)$, then it yields

$$P(k) = P(k-1) - \frac{P(k-1)\varphi(k)\varphi^\top(k)P(k-1)}{1 + \varphi^\top(k)P(k-1)\varphi(k)}. \tag{33}$$

However, the recursive algorithm in (30)–(33) cannot be implemented because $w(k-i)$, $i = 1, \ldots, n_c$, in $\varphi(k)$ are unknown. To address this, the noise model is identified by comparing the measured output $y_{\text{noise}}(k)$ with the NF model output $\hat{y}(k)$. Replacing $w(k-i)$ in (25) with $\hat{w}(k-i)$ gives

$$\hat{w}(k-i) = y_{\text{noise}}(k-i) - \hat{y}(k-i). \tag{34}$$

Thereafter $\hat{\varphi}(k)$ can be constructed. By replacing $\varphi(k)$ in (32)–(33) with its estimate $\hat{\varphi}(k)$, the proposed RGLS-based algorithm for identifying $\hat{\theta}(k)$ is summarised in Algorithm 2. The order of the noise model is determined before launching the online learning algorithm. It is observed that a higher-order AR model generates a non-positive definite covariance matrix, which is more likely to cause unstable prediction. Therefore, a low-order AR model is used in this work.

4 Convergence of the proposed algorithm

This section analyses convergence of the RGLS-based learning algorithm, i.e., the use of (30)–(33) to estimate the parameter $\hat{\theta}(k)$.

To facilitate the analysis, (30) is reformulated as

$$\begin{aligned}\hat{\theta}(k) &= P(k)\xi(k)\\ &= \hat{\theta}(k-1) + L(k)[y_{\text{noise}}(k) - \hat{\varphi}^{\top}(k)\hat{\theta}(k-1)]\end{aligned} \tag{35}$$

where

$$L(k) = P(k)\hat{\varphi}(k) = \frac{P(k-1)\hat{\varphi}(k)}{1+\hat{\varphi}^{\top}(k)P(k-1)\hat{\varphi}(k)}. \tag{36}$$

Before proceeding to the convergence analysis, we provide some useful concepts as below:

(i) Let $\{v(k), \mathcal{F}_k\}$ be a martingale difference sequence defined in a probability space $\{\Omega, \mathcal{F}, P\}$, where $\mathcal{F}_k$ denotes the algebra sequence generated by the noise signal $\{v(k)\}$. The noise sequence $\{v(k)\}$ satisfies

$$\mathrm{E}[v(k)|\mathcal{F}_{t-1}] = 0 \tag{37}$$

$$\mathrm{E}[v(k)^2|\mathcal{F}_{t-1}] = \sigma^2(k) \le \overline{\sigma}^2 < \infty \tag{38}$$

where $\mathrm{E}[\cdot]$ is the expectation and σ denotes the noise variance with the upper boundary $\overline{\sigma}$.

(ii) For $g(k) \ge 0$, we write $f(k) = O(g(k))$, if there exists finite positive constants δ and k_0 such that $|f(k)| \le \delta g(k)$ for $k \ge k_0$.

The convergence property of the proposed RGLS learning algorithm is stated in Theorem 1.

Theorem 1 *For the system (22), assume that (37) and (38) hold. Then for any $c > 1$, the parameter estimation error of the RGLS algorithm in (30)–(33) satisfies*

$$\|\hat{\theta}(k) - \theta\|^2 = O\left(\frac{[\ln|P^{-1}(k)|]^c}{\lambda_{min}[P^{-1}(k)]}\right). \tag{39}$$

Proof See Appendix A. □

Theorem 1 shows that for the noise sequence $\{v(k)\}$ with a bounded variance, the convergence rate of the RGLS algorithm is proportional to the ratio between the logarithm of the maximum eigenvalue and the minimum eigenvalue of the covariance matrix $P^{-1}(k)$.

5 Numerical examples

Two numerical examples are used to evaluate effectiveness of the proposed online learning algorithm. The modelling results are compared with those of the offline trained model. Since numerical examples are used, the model output is compared with the true system output. The model performance is quantified by the normalised root mean square error (NRMSE) defined as

$$\text{NRMSE} = \sum_{i=1}^{k}\sqrt{\frac{y(i)-\hat{y}(i)}{y(i)-\mathrm{E}[y]}}. \tag{40}$$

Algorithm 2 RGLS-based NF model learning

1: **Initialise:** Calculate $\xi(0)$ and $P(0)$ using (31) and (32) respectively with the noise-free training data.
2: Assign modelling error in the training phase to $\hat{\varphi}^n(0)$ using (34).
3: **for** $k = 1, 2, \cdots, M$ **do**
4: Collect measurements $u(k)$ and $y_{\text{noise}}(k)$, and construct $\varphi(k)$ and $\tilde{\varphi}(k)$ using (2) - (5) accordingly.
5: Construct $\varphi^s(k)$ and $\hat{\varphi}^n(k)$ using (10), (11), (20) and (25).
6: Form $\hat{\varphi}(k)$ using (27).
7: Calculate $\xi(k)$ using (32) and $P(k)$ using (33).
8: Update $\hat{\theta}(k)$ using (30).
9: Estimate $\hat{w}(k-i)$, $i = 1, \ldots, n_c$, using (21) and (34).
10: **end for**
▷ M denotes the number of online measurements.

Noise-free data is used for offline model training. During the training process, the model expands to fit the training data. However, the increased model complexity will deteriorate the prediction accuracy due to over-fitting. In order to find a suitable offline model, the training data is divided into two parts: the first 65% is for training and the last 35% is for validation. The model with the lowest NRMSE in fitting validation data is chosen as the offline model.

Case 1 Model learning encountering new operational conditions with measurement noise

This case demonstrates the capability of the proposed algorithm in capturing new operational conditions of the dynamic system under noise. The online measurements in some operational conditions are not covered by the training data. The following nonlinear dynamic system is tested [15]:

$$\begin{aligned}y_{\text{noise}}(k+1) = {} & 0.72y(k) + 0.025y(k-1)u(k-1)\\ & + 0.01u^2(k-2) + 0.2u(k-1)\\ & + w(k)\end{aligned} \tag{41}$$

with the noise

$$w(k) = \frac{1}{1+0.11z^{-1}}v(k) \tag{42}$$

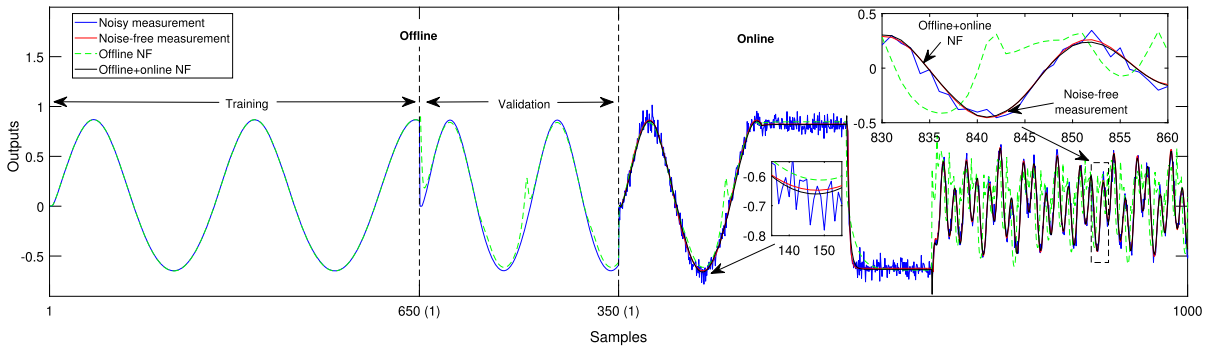

Fig. 4 Comparison of model prediction for offline training, offline validation and online testing: Case 1

where $v(k)$ is a random noise signal with zero mean and $\sigma^2 = 0.001$. In total of 1000 training data is generated by a sinusoidal signal $u(k) = 1.05\sin(k/45)$ with the initial outputs $y(i) = 0,\ i = 1, \ldots, 4$. When generating the training data, the noise term $w(k)$ is deactivated. There are 1000 online noisy measurements produced using the input signal

$$u(k) = \begin{cases} 1.05\sin(\pi k/30)\,, & 1 < k < 250 \\ 1.0, & 250 \le k < 400 \\ -1.0, & 400 \le k < 550 \\ 0.3\sin(\pi k/25) + 0.1\sin(\pi k/32) & \\ \quad + 0.6\sin(\pi k/10) & 550 \le k \le 1000. \end{cases}$$

An offline model with 9 neurons is built using the training data. The model fits the training data and the validation data accurately with NRMSE = 0.001483 and NRMSE = 0.0013, respectively. The modelling results of the offline model and online RGLS-based NF model are illustrated in Fig. 4. The offline model performs poorly when it is implemented online under noise, even though it has a low NRMSE during training. As shown by the zoomed in plot, the offline NF model cannot deal with the noisy signal. Overshoots and biases are observed in the model output. The proposed online learning model estimates the noise signal online with a second-order AR model. The online NF model reduces spikes in the prediction. Meanwhile, the prediction is closer to the actual dynamic response of the system.

The performance of the offline and online RGLS-based NF models is compared in Fig. 5. It is seen that the online RGLS-based NF model overwhelms the offline trained model under noise. The overall prediction error of the offline NF model is NRMSE = 0.4685, while the prediction error of the online RGLS-based

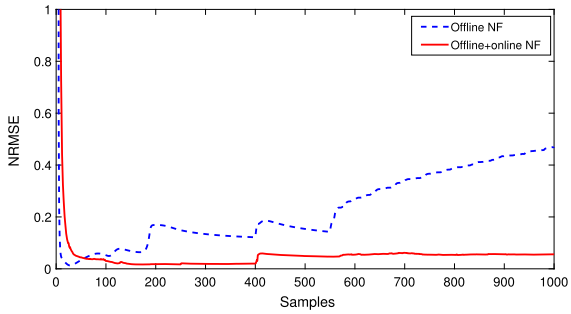

Fig. 5 Comparison of online modelling performance: Case 1

NF model is NRMSE = 0.05295. For the online noisy measurements including new operational conditions ($550 \le k \le 1000$), the offline model is less capable in predicting the dynamic system response since it does not fully cover the system behaviour. Therefore, the model has poor accuracy and does not converge. However, the proposed algorithm can reflect the real system response accurately. Moreover, the proposed algorithm converges quickly within 100 samples.

Case 2 Model learning of a time-varying system with measurement noise

In this case, the proposed algorithm is used to identify a time-varying dynamic system. The example system is modified from (41) by only replacing one constant parameter with a time varying parameter as follows:

$$\begin{aligned} y_{\text{noise}}(k+1) =& 0.72y(k) + \beta(k)y(k-1)u(k-1) \\ &+ 0.01u^2(k-2) \\ &+ 0.2u(k-1) + w(k) \end{aligned} \tag{43}$$

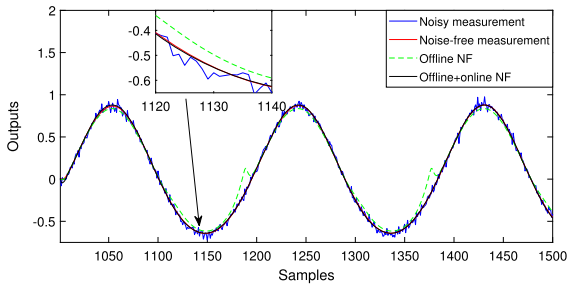

Fig. 6 Comparison of online model prediction: Case 2

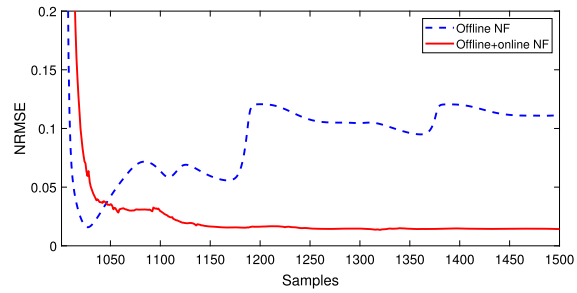

Fig. 7 Comparison of online modelling performance: Case 2

with a time-varying parameter

$$\beta(k) = \begin{cases} 0.025, & 1 \le k \le 1000 \\ 0.025 + 0.00001k, & 1000 < k \le 1500 \end{cases}. \tag{44}$$

The noise term $w(k)$ is the same as (42) in Case 1. For the offline training data generated during $1 \le k \le 1000$, the input patterns in Case 1 are utilised. Therefore, the offline NF model in Case 2 is identical to the offline NF model in Case 1. To generate online noisy measurements, the noise term $w(k)$ is activated. A set of 500 online noisy measurements is generated using the input signal $u(k) = 1.05\sin(\pi k/30)$, $1001 \le k \le 1500$.

The online predictions of the offline NF model and the proposed RGLS-based NF model are compared in Fig. 6. The offline NF model fits the training data accurately by using 9 neurons, with NRMSE = 0.001483 for the training data and NRMSE = 0.0013 for the validation data. However, the online performance of the offline trained NF model is poor with an overall NRMSE = 0.1111, as shown in Fig. 7. Contrarily, the RGLS-based NF model estimates the noise accurately with a second-order AR model and removes the noise from the measured system output. The NF is then updated using the data describing the actual system behaviour. Moreover, when the system behaviour changes online, the accuracy of the RGLS-based NF model is improved continuously using the incoming measurements with the overall NRMSE = 0.01432.

6 Experimental study

The performance of the proposed online model learning method is further evaluated on an electrified turbocharged diesel engine shown in Fig. 8. The engine is

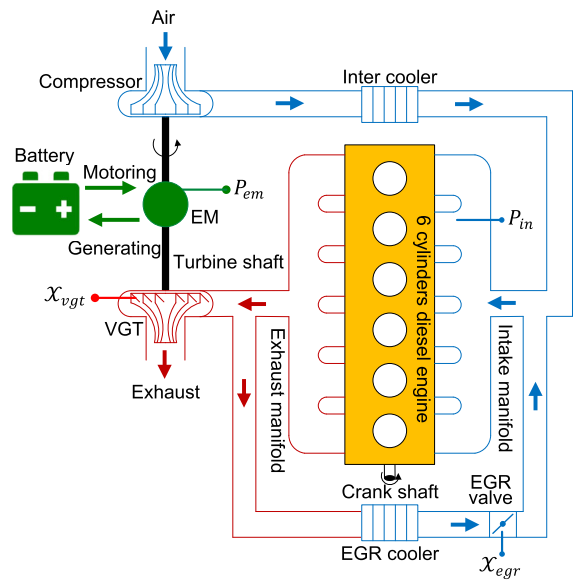

Fig. 8 Electrified turbocharged diesel engine, where EGR means high-pressure exhaust gas recirculation, VGT means variable geometry turbocharger, and EM means electrical motor

a Cat C7.1 ACERT heavy-duty engine equipped with a high-pressure exhaust gas recirculation (EGR) and an electrical turbocharger assist (ETA). The engine produces peak power 205 kW at 2200 r/min and peak torque 1257 Nm at 1400 r/min.

The ETA is developed to overcome the transient limitation of a downsized engine. The ETA consists of a variable geometry turbocharger (VGT) and an electrical motor (EM). The engine exhausts gas before the EGR valve passes through the VGT. The expansion of the exhaust gas produces mechanical power through the VGT turbine. The power is then used to drive the compressor via the turbine shaft. Thereafter, more fresh air is pumped into the intake manifold which increases the intake air pressure P_{in}. The ratio of modes converting

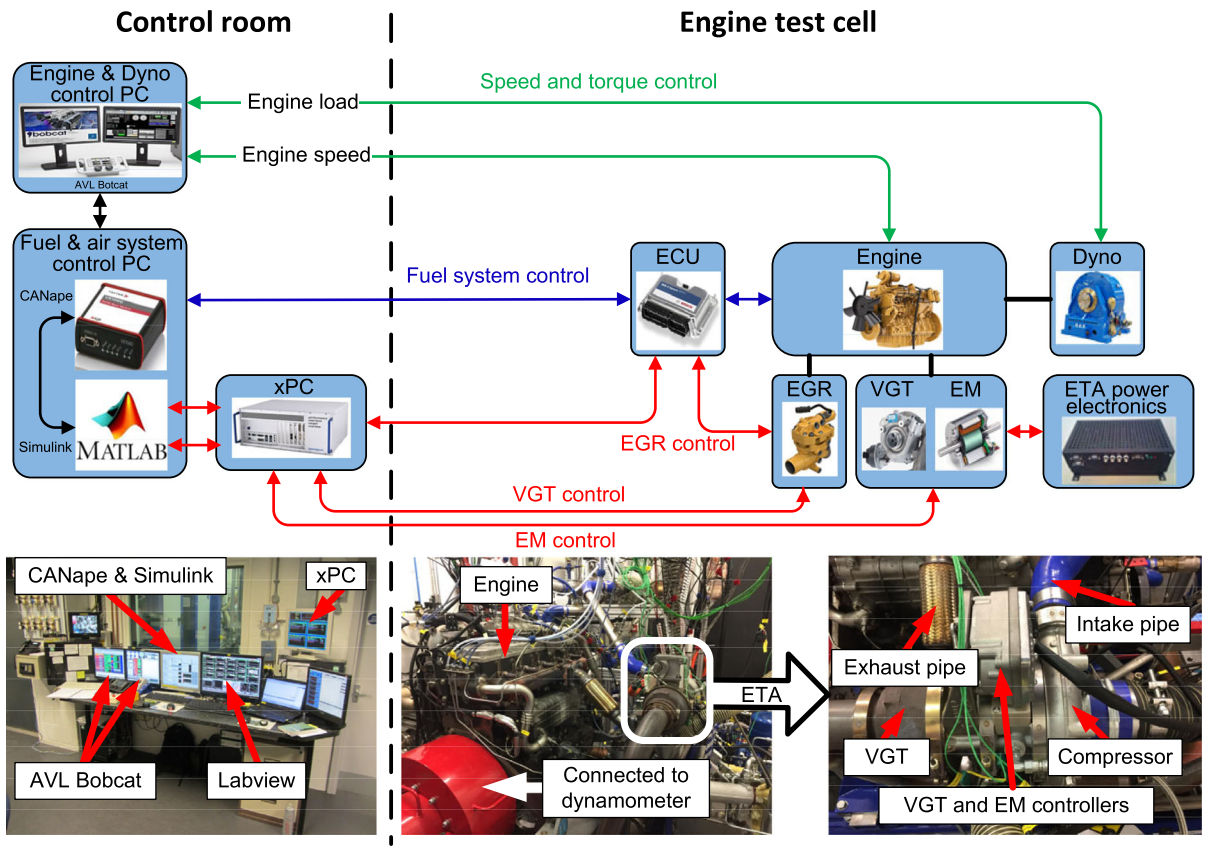

Fig. 9 Testing platform of the electrified turbocharged diesel engine, where EGR means high-pressure exhaust gas recirculation, VGT means variable geometry turbocharger, EM means electrical motor, and ETA means electrical turbocharger assist

mechanical power into kinetic energy of intake air is determined by the VGT actuator position $\mathcal{X}_{\text{vgt}}$. The EM can work in either motoring or generating. In motoring, the EM extracts power from the battery to provide more air into the engine. In generating, the EM converts mechanical power into electrical power and stores the electricity in the battery. Consequently, P_{in} is reduced. The electric power extracting/storing from/in the battery is denoted as ETA power P_{em}. The EGR delivers part of the exhaust gas into the intake manifold to dilute the oxygen for NO_x reduction. The amount of recirculated exhaust gas can be changed by tuning the EGR actuator $\mathcal{X}_{\text{egr}}$. Since both the VGT and EGR are mounted on the exhaust manifold, they are strongly coupled and highly nonlinear in P_{in}. This motivates the use of NF model to represent the dynamics of the system.

The model of electrified turbocharged diesel engine intake air path is identified as the following structure:

$$\begin{aligned} P_{\text{in}}(k+1) = f\Big(&\mathcal{X}_{\text{vgt}}(k), \mathcal{X}_{\text{vgt}}(k-1), \mathcal{X}_{\text{vgt}}(k-2),\\ &\mathcal{X}_{\text{egr}}(k), \mathcal{X}_{\text{egr}}(k-1), \mathcal{X}_{\text{egr}}(k-2),\\ &P_{\text{em}}(k), P_{\text{em}}(k-1), P_{\text{em}}(k-2),\\ &P_{\text{in}}(k), P_{\text{in}}(k-1)\Big). \end{aligned}$$

A set of experimental data is generated and filtered offline for model training and validation. The amplitude modulated pseudo random binary signal (AMPRBS) is applied for the input signal excitation at the operating condition (1800 r/min, 260 Nm). The experimental data continues for 100 s at 10 Hz sampling rate. The ranges of $\mathcal{X}_{\text{vgt}}$, P_{em} and $\mathcal{X}_{\text{egr}}$ are limited to [0.1, 0.95], [−1.5 kW, 1.5 kW] and [0.05, 0.6], respectively. The modelling accuracy for training and validation is NRMSE = 0.0228 and NRMSE = 0.0375, respectively. The model is implemented online with the proposed learning algorithm.

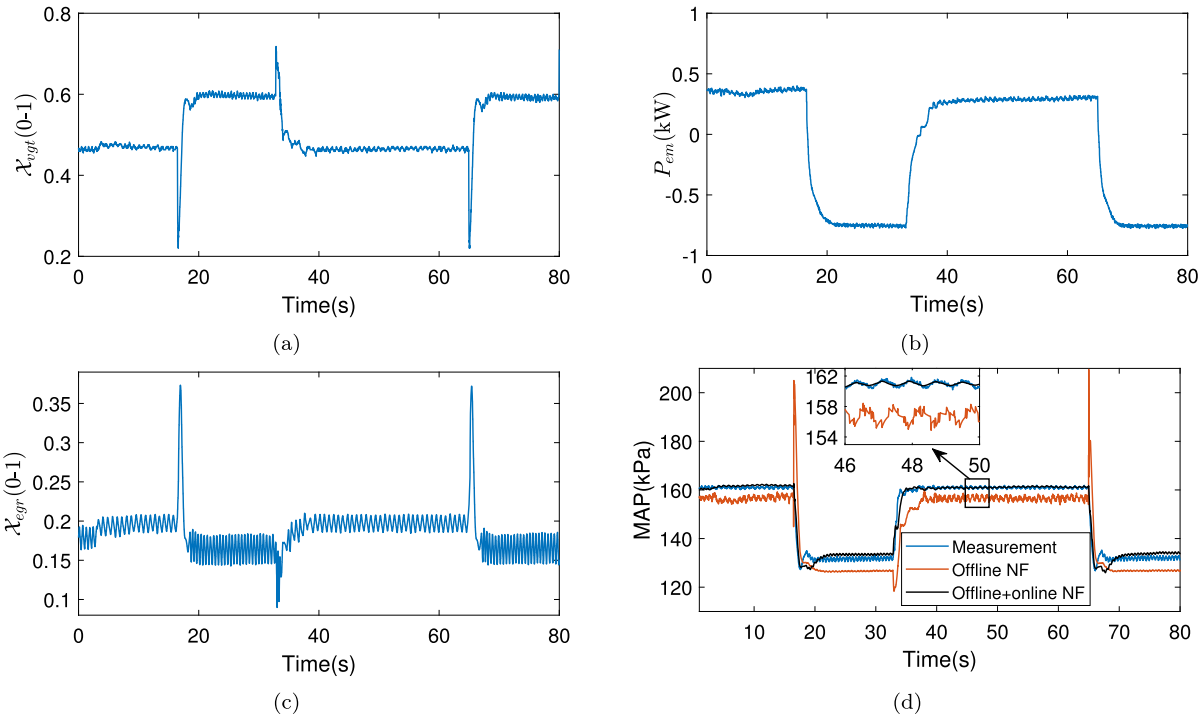

Fig. 10 Modelling performance test at (1800 r/min, 120 Nm) at steady state: **a** $\mathcal{X}_{\text{vgt}}$ action, **b** P_{em} action, **c** $\mathcal{X}_{\text{egr}}$ action and **d** Model prediction results

6.1 Experiments setup

The test platform is illustrated in Fig. 9. The test engine is connected to a dynamometer. The engine torque and speed are controlled by the AVL Bobcat system. The fuel injection system is controlled by the engine control module (ECM). The EGR input command is delivered to the actuator via ECM, whereas the rest of the input signals are sent via xPC. The xPC is connected to the host PC with an Ethernet cable, enabling real time communication between MATLAB and CANape. The data exchange between the control room and the engine cell is via the CAN bus. The test data is acquired by the Vector CANape and visualised in Labview.

6.2 Experimental results

6.2.1 Modelling performance at steady state

The testing results at (1800 r/min, 260 N·m) are illustrated in Fig. 10. From 0 s to 80 s, the input variables $\mathcal{X}_{\text{vgt}}$, P_{em} and $\mathcal{X}_{\text{egr}}$ change periodically and are then maintained until the system is stabilised. The offline model cannot capture the system behaviour, although it does fit the training data well with NRMSE = 0.0228. During the test, the offline model introduces steady state error and has spikes when the input signals change. The proposed online RGLS-based NF model tracks the system response accurately in steady state and reduces spikes in the prediction during transients.

6.2.2 Robustness against fast transient

The modelling performance when input signals change rapidly during transients are illustrated in Fig. 11. The input variables $\mathcal{X}_{\text{vgt}}$ and $\mathcal{X}_{\text{egr}}$ are set to maintain the engine at a steady condition. P_{em} is designed to change rapidly to maintain the exhaust pressure. The offline NF model completely fails to provide predictions, as shown in Fig. 11d. The fluctuation of the offline model prediction is aggressive. After 60 s, the prediction becomes unstable and goes to a very large value. The online NF model extracts the actual system output from the noisy measurements and updates the model parameters

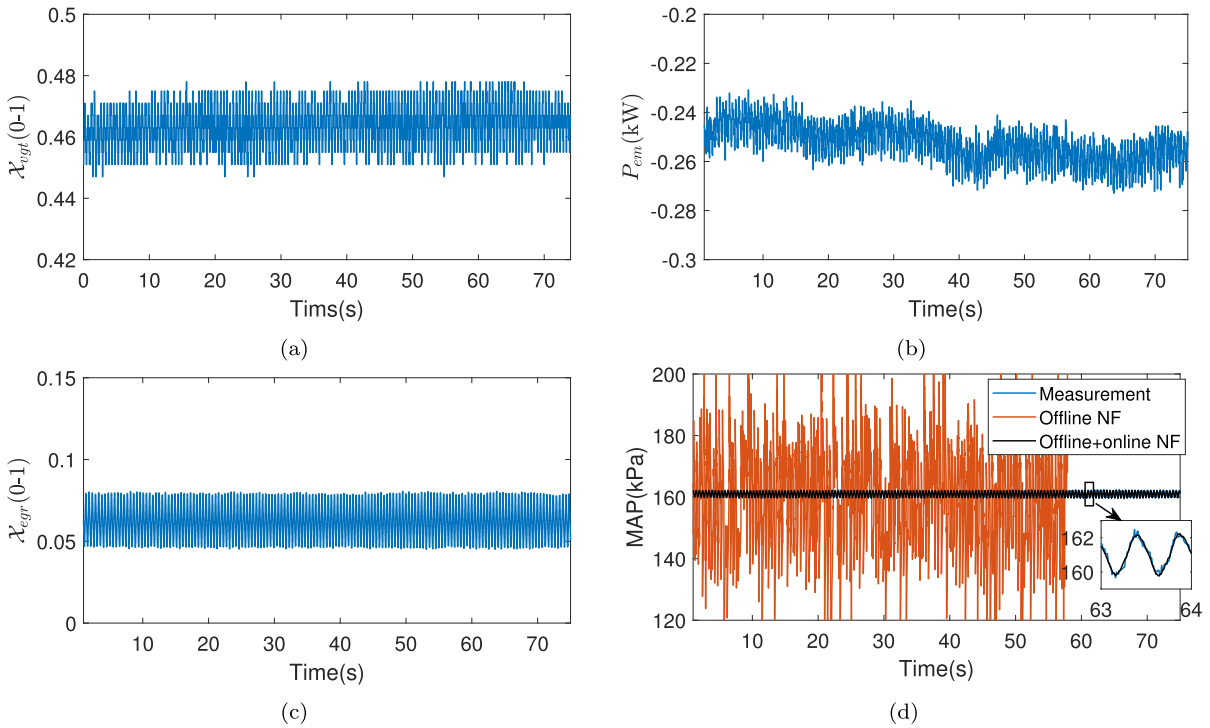

Fig. 11 Modelling performance test at (1800 r/min, 120 Nm) under fast transient: **a** $\mathcal{X}_{\text{vgt}}$ action, **b** P_{em} action, **c** $\mathcal{X}_{\text{egr}}$ action, and **d** Model prediction results (After 60 s, the offline NF model prediction goes to 10^{15}, hence it is not depicted here)

accordingly. This enables the online model to capture the system behaviour accurately.

7 Conclusion

This paper presents an online learning algorithm for improving the NF model performance in noisy environment. An NF model with ARX plus Gaussian function is introduced for identifying dynamic systems. An enhanced RGLS-based local learning algorithm is proposed for online implementations when measurement noise is present, which updates the NF model to capture the actual dynamic system behaviours. The proposed learning algorithm is proved to have a convergence guarantee. The results of two numerical examples show that the RGLS-based NF model obtains accurate fitting, robustness against noise and fast convergence speed. Efficacy of the proposed algorithm is further confirmed by applying it to modelling the complex nonlinear dynamics of a commercial heavy-duty diesel engine. Future work will consider integrating the online learning model with adaptive control to establish a model-learning-based control strategy for complex nonlinear systems.

Funding Wen Gu was supported by the EPSRC Centre for Doctoral Training in Embedded Intelligence under Grant EP/L014998/1. Jianglin Lan was supported by a Leverhulme Trust Early Career Fellowship under Award ECF-2021-517.

Data availability Data will be made available from the corresponding author upon reasonable request.

Declarations

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

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view

a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

A Proof of Theorem 1

The following notations are defined for further deduction. For a square matrix $\boldsymbol{X}$, the symbol $\lambda_{\min}[\boldsymbol{X}]$ represents its minimum eigenvalues and $|\boldsymbol{X}| = \det[\boldsymbol{X}]$ denotes its matrix determinant.

Define the parameter estimation error as

$$\tilde{\theta}(k) = \hat{\theta}(k) - \theta. \tag{45}$$

Note that $P(k)$ is a symmetric positive definite matrix. Using (35) and (45) gives

$$\begin{aligned}
&\tilde{\theta}(k)\\
&=\tilde{\theta}(k-1) + P(k)\hat{\varphi}(k)\left[\varphi^{\top}(k)\theta + \nu(k) - \hat{\varphi}^{\top}(k)\hat{\theta}(k-1)\right]\\
&= \tilde{\theta}(k-1) + P(k)\hat{\varphi}(k)\Big[- \hat{\varphi}^{\top}(k)\tilde{\theta}(k-1) +\\
&\quad (\varphi(k) - \hat{\varphi}(k))^{\top}\theta + \nu(k)\Big]\\
&=\tilde{\theta}(k-1) + P(k)\hat{\varphi}(k)\left[-\tilde{y}(k) + \Delta(k) + \nu(k)\right]
\end{aligned} \tag{46}$$

where $\tilde{y}(k) = \hat{\varphi}^{\top}(k)\tilde{\theta}(k-1)$, $\Delta(k) = \left[\varphi(k) - \hat{\varphi}(k)\right]^{\top}\theta$.

Consider the Lyapunov function

$$V(k) = \tilde{\theta}^{\top}(k)P^{-1}(k)\tilde{\theta}(k). \tag{47}$$

Substituting (32) and (46) into (47) gives

$$\begin{aligned}
V(k) &= \left\{\tilde{\theta}(k-1) + P(k)\hat{\varphi}(k)\left[-\tilde{y}(k) + \Delta(k) + \nu(k)\right]\right\}^{\top}\\
&\quad P^{-1}(k)\\
&\quad \left\{\tilde{\theta}(k-1) + P(k)\hat{\varphi}(k)\left[-\tilde{y}(k) + \Delta(k) + \nu(k)\right]\right\}\\
&= V(k-1) - \tilde{y}^2(k) + 2\tilde{y}(k)\left[\Delta(k) + \nu(k)\right]\\
&\quad +\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)\{\tilde{y}^2(k) + \nu^2(k) + \Delta^2(k)\\
&\quad -2\tilde{y}(k) - 2\tilde{y}(k)[\Delta(k) + \nu(k)] + 2\Delta(k)\nu(k)\}\\
&= V(k-1) - \left[1 - \hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)\right]\tilde{y}^2(k)\\
&\quad +2\left[1 - \hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)\right]\tilde{y}(k)[\Delta(k) + \nu(k)]\\
&\quad +\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)[\nu^2(k) + \Delta^2(k) + 2\Delta(k)\nu(k)].
\end{aligned}$$

In view of the relation

$$\begin{aligned}
1 - \hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k) &= 1 - \frac{\hat{\varphi}^{\top}(k)P(k-1)\hat{\varphi}(k)}{1 + \hat{\varphi}^{\top}(k)P(k-1)\hat{\varphi}(k)}\\
&= \left[1 + \hat{\varphi}^{\top}(k)P(k-1)\hat{\varphi}(k)\right]^{-1}\\
&> 0,
\end{aligned}$$

we have the following inequality

$$\begin{aligned}
V(k) \leq\ & V(k-1)\\
&+ 2\left[1 - \hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)\right]\tilde{y}(k)[\Delta(k) + \nu(k)]\\
&+ \hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)\left[\nu^2(k) + \Delta^2(k) + 2\Delta(k)\nu(k)\right].
\end{aligned} \tag{48}$$

Assume that $\Delta(k)$ is bounded as $\Delta^2(k) \leq \varepsilon$, where ε is the upper boundary. Since $\nu(k)$ is a white noise sequence with zero mean and variance σ^2, the terms $\tilde{y}(k)$, $\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)$ and $\Delta(k)$ are uncorrelated to the noise signal. Under the assumptions (37) and (38), taking the conditional expectation of (48) with respect to $\mathcal{F}_{t-1}$ gives

$$\begin{aligned}
\mathrm{E}[V(k)|\mathcal{F}_{t-1}] \leq\ & V(k-1)\\
&+\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)[\overline{\sigma}^2 + \varepsilon].
\end{aligned} \tag{49}$$

It follows from (47) that

$$\|\tilde{\theta}(k)\|^2 \leq \frac{\tilde{\theta}^{\top}(k)P^{-1}(k)\tilde{\theta}(k)}{\lambda_{\min}[P^{-1}(k)]} = \frac{V(k)}{\lambda_{\min}[P^{-1}(k)]}. \tag{50}$$

Define the variable $Z(k)$ as

$$\mathrm{Z}(k) = \frac{V(k)}{[\ln|P^{-1}(k)|]^c}. \tag{51}$$

According to (31), $P^{-1}(k) = P^{-1}(k-1) + \varphi(k)\varphi^{\top}(k)$. Since both the covariance matrix $P^{-1}(k-1)$ and the matrix $\varphi(k)\varphi^{\top}(k)$ are positive semidefinite, it holds that $|P^{-1}(k)| \geq |P^{-1}(k-1)|$ and thus $\ln|P^{-1}(k)|$ is non-decreasing. Substituting (51) into (49) and taking the conditional expectation of both sides leads to

$$\begin{aligned}
&\mathrm{E}[\mathrm{Z}(k)|\mathcal{F}_{t-1}]\\
&\leq \frac{V(k-1)}{[\ln|P^{-1}(k)|]^c} + \frac{\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)}{[\ln|P^{-1}(k)|]^c}[\overline{\sigma}^2 + \varepsilon]\\
&\leq \mathrm{Z}(k-1) + \frac{\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)}{[\ln|P^{-1}(k)|]^c}[\overline{\sigma}^2 + \varepsilon].
\end{aligned} \tag{52}$$

By using (31), $P^{-1}(k-1)$ can be obtained as

$$P^{-1}(k-1) = P^{-1}(k) - \hat{\varphi}(k)\hat{\varphi}^{\top}(k).$$

Taking the determinant of both sides and using $|I + AB| = |I + BA|$ gives

$$\begin{aligned}
|P^{-1}(k-1)| &= |P^{-1}(k)||I - P(k)\hat{\varphi}(k)\hat{\varphi}^{\top}(k)|\\
&= |P^{-1}(k)|[1 - \hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)].
\end{aligned}$$

Hence, we have that

$$\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k) = \frac{|P^{-1}(k)| - |P^{-1}(k-1)|}{|P^{-1}(k)|}.$$

As $|P^{-1}(k)|$ is a non-decreasing function of k, summing the second term on the right hand side of (52) from $k = 1$ to ∞ yields

$$\begin{aligned}
&\sum_{k=1}^{\infty} \frac{\hat{\varphi}^{\top}(k)P(k)\hat{\varphi}(k)}{[\ln|P^{-1}(k)|]^c} \\
&= \sum_{k=1}^{\infty} \frac{|P^{-1}(k)| - |P^{-1}(k-1)|}{|P^{-1}(k)|[\ln|P^{-1}(k)|]^c} \\
&= \sum_{k=1}^{\infty} \int_{|P^{-1}(k-1)|}^{|P^{-1}(k)|} \frac{\mathrm{dx}}{|P^{-1}(k)|[\ln|P^{-1}(k)|]^c} \\
&\leq \int_{|P^{-1}(0)|}^{|P^{-1}(\infty)|} \frac{\mathrm{dx}}{|P^{-1}(k)|[\ln|P^{-1}(k)|]^c} \\
&\leq \int_{\ln|P^{-1}(0)|}^{\ln|P^{-1}(\infty)|} \frac{1}{[\ln|P^{-1}(k)|]^c} \mathrm{d}[\ln|P^{-1}(k)|] \\
&\leq \frac{[\ln|P^{-1}(\infty)|]^{1-c} - [\ln|P^{-1}(0)|]^{1-c}}{1-c} + \rho I \\
&< \infty
\end{aligned} \tag{53}$$

where ρ is a positive constant.

By using the martingale convergence theorem, we have

$$\lim_{k\to\infty} Z(k) = \lim_{k\to\infty} \frac{V(k)}{[\ln|P^{-1}(k)|]^c} + \rho I < \infty.$$

This indicates that

$$V(k) = O([\ln|P^{-1}(k)|]^c). \tag{54}$$

From (50), (52) and (54), $\|\tilde{\theta}(k)\|^2$ can be derived as

$$\|\tilde{\theta}(k)\|^2 = O\left(\frac{[\ln|P^{-1}(k)|]^c}{\lambda_{\min}[P^{-1}(k)]}\right).$$

This proves that the parameter estimation error will converge to a small neighbourhood of the origin when k goes to infinity. This completes the proof.

References

1. Adeniran, A.A., El Ferik, S.: Modeling and identification of nonlinear systems: a review of the multimodel approach-part 1. IEEE Trans. Syst. Man Cybern. Syst. **47**(7), 1149–1159 (2016)
2. Angelov, P.P., Filev, D.P., Member, S.: An approach to online identification of Takagi-Sugeno fuzzy models. IEEE Trans. Syst. Man. Cybern. B Cybern. **34**(1), 484–498 (2004)
3. Ažman, K., Kocijan, J.: Dynamical systems identification using gaussian process models incorporated local models. Eng. Appl. Artif. Intell. **24**(2), 398–408 (2011)
4. de Campos Souza, P.V.: Fuzzy neural networks and neuro-fuzzy networks: a review the main techniques and applications used in the literature. Appl. Soft Comput. **92**, 106275 (2020)
5. Ding, F., Wang, Y., Ding, J.: Recursive least squares parameter identification algorithms for systems with colored noise using the filtering technique and the auxilary model. Digit. Signal Process. A Rev. J. **37**(1), 100–108 (2015)
6. Dinh, H.Q., Aubert, N., Noman, N., Fujii, T., Rondelez, Y., Iba, H.: An effective method for evolving reaction networks in synthetic biochemical systems. IEEE Trans. Evol. Comput. **19**(3), 374–386 (2015)
7. Du, H., Zhang, N.: Application of evolving Takagi-Sugeno fuzzy model to nonlinear system identification. Appl. Soft Comput. J. **8**(1), 676–686 (2008)
8. Feng, S., Philip, C.L.: Fuzzy broad learning system: a novel neuro-fuzzy model for regression and classification. IEEE Trans. Cybern. **50**(2), 414–424 (2020)
9. Gao, H., He, W., Zhou, C., Sun, C.: Neural network control of a two-link flexible robotic manipulator using assumed mode method. IEEE Trans. Neural Netw. Learn. Syst. **29**(2), 755–765 (2018)
10. Gu, W., Zhao, D., Mason, B.: Real-time modelling and parallel optimisation of a gasoline direct injection engine. In: 2019 American Control Conference. pp. 5544–5549. IEEE, Philadelphia (2019)
11. Gu, X., Angelov, P., Han, J., Shen, Q.: Multilayer evolving fuzzy neural networks. IEEE Trans. Fuzzy Syst. (2023). https://doi.org/10.1109/TFUZZ.2023.3276263
12. Gu, X., Han, J., Shen, Q., Angelov, P.P.: Autonomous learning for fuzzy systems: a review. Artif. Intell. Rev. 1–47 (2022)
13. Hametner, C., Jakubek, S.: Local model network identification for online engine modelling. Inf. Sci. (Ny) **220**, 210–225 (2013)
14. Han, H., Wu, X.L., Qiao, J.F.: Nonlinear systems modeling based on self-organizing fuzzy-neural-network with adaptive computation algorithm. IEEE Trans. Cybern. **44**(4), 554–564 (2014)
15. Han, H.G., Lin, Z.L., Qiao, J.F.: Modeling of nonlinear systems using the self-organizing fuzzy neural network with adaptive gradient algorithm. Neurocomputing **266**, 566–578 (2017)
16. He, W., Dong, Y.: Adaptive fuzzy neural network control for a constrained robot using impedance learning. IEEE Trans. Neural Netw. Learn. Syst. **29**(4), 1174–1186 (2018)
17. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. SIAM, Philadelphia (2002)
18. Hsu, C.F.: Nonlinear system control using a self-organizing functional-linked neuro-fuzzy network. Nonlinear Dyn. **73**(3), 1631–1643 (2013)
19. Huang, H., Rong, H.J., Yang, Z.X., Vong, C.M.: Recursive least mean dual p-power solution to the generalization of evolving fuzzy system under multiple noises. Inf. Sci. **609**, 228–247 (2022)
20. Hung, Y.C., Lin, F.J., Hwang, J.C., Chang, J.K., Ruan, K.C.: Wavelet fuzzy neural network with asymmetric membership function controller for electric power steering system via

improved differential evolution. IEEE Trans. Power Electron. **30**(4), 2350–2362 (2014)
21. Jakubek, S., Hametner, C.: Identification of neurofuzzy models using GTLS parameter estimation. IEEE Trans. Syst. Man Cybern. Part B Cybern. **39**(5), 1121–1133 (2009)
22. Jakubek, S., Hametner, C., Keuth, N.: Total least squares in fuzzy system identification: an application to an industrial engine. Eng. Appl. Artif. Intell. **21**(8), 1277–1288 (2008)
23. Jang, J.S.R.: ANFIS: adaptive-network-based fuzzy inference systems. IEEE Trans. Syst. Man Cybern. Part B Cybern. **23**(3), 665–685 (1993)
24. Jang, J.S.R., Sun, C.T.: Neuro-fuzzy modelling and control. Proc. IEEE **83**(3), 378–406 (1995)
25. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-fuzzy and Soft Computing-A Computational Approach to Learning and Machine Intelligence. Prentice-Hall Inc, Upper Saddle River, NJ, USA (1997)
26. Leung, C.S., Young, G.H., Sum, J., Kan, W.K.: On the regularization of forgetting recursive least square. IEEE Trans. Neural Netw. **10**(6), 1482–1486 (1999)
27. Lin, Y.Y., Chang, J.Y., Lin, C.T.: Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. IEEE Trans. Neural Netw. Learn. Syst. **24**(2), 310–321 (2013)
28. Liu, C., et al.: Neural-network-based sliding-mode control of an uncertain robot using dynamic model approximated switching gain. IEEE Trans. Cybern. **51**(5), 2339–2346 (2021)
29. Liu, S., Liu, Y., Liang, X., Wang, N.: Uncertainty observation-based adaptive succinct fuzzy-neuro dynamic surface control for trajectory tracking of fully actuated underwater vehicle system with input saturation. Nonlinear Dyn. **98**(3), 1683–1699 (2019)
30. Liu, Y.T., Lin, Y.Y., Wu, S.L., Chuang, C.H., Lin, C.T.: Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network. IEEE Trans. Neural Netw. Learn. Syst. **27**(2), 347–360 (2015)
31. Lughofer, E.: Improving the robustness of recursive consequent parameters learning in evolving neuro-fuzzy systems. Inf. Sci. **545**, 555–574 (2021)
32. Lughofer, E.: Evolving fuzzy and neuro-fuzzy systems: fundamentals, stability, explainability, useability, and applications. In: Handbook on Computer Learning and Intelligence: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation, pp. 133–234. World Scientific, Singapore (2022)
33. Mehrasa, M., Babaie, M., Zafari, A., Al-Haddad, K.: Passivity ANFIS-based control for an intelligent compact multilevel converter. IEEE Trans. Industr. Inform. **17**(8), 5141–5151 (2021)
34. Miranian, A., Abdollahzade, M.: Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction. IEEE Trans. Neural Netw. Learn. Syst. **24**(2), 207–218 (2013)
35. Murray-Smith, R., Johansen, T.A.: Local learning in local model networks. In: Fourth International Conference of Artificial Neural Networks. IET (1995)
36. Nelles, O.: Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, 2nd edn. Springer, New York (2013)
37. Nelles, O.: Nonlinear System Identification. Springer, New York (2020)
38. Nelles, O., Fink, A., Isermann, R.: Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. In IFAC Proceedings Volumes. vol. 33(15), pp. 845–850. (2000)
39. Ning, H., Qing, G., Tian, T., Jing, X.: Online identification of nonlinear stochastic spatiotemporal system with multiplicative noise by robust optimal control-based kernel learning method. IEEE Trans. Neural Netw. Learn. Syst. **30**(2), 389–404 (2019)
40. Ožbot, M., Lughofer, E., Škrjanc, I.: Evolving neuro-fuzzy systems based design of experiments in process identification. IEEE Trans. Fuzzy Syst. **31**(6), 1995–2005 (2023)
41. Rafiei, H., Akbarzadeh-T, M.R.: Reliable fuzzy neural networks for systems identification and control. IEEE Trans. Fuzzy Syst. **31**(7), 2251–2263 (2022)
42. Ren, Y., Zhao, Z., Zhang, C., Yang, Q., Hong, K.S.: Adaptive neural-network boundary control for a flexible manipulator with input constraints and model uncertainties. IEEE Trans. Cybern. **51**(10), 4796–4807 (2021)
43. Rhode, S., Bleimund, F., Gauterin, F.: Recursive generalized total least squares with noise covariance estimation. In IFAC Proceedings Volumes. vol. 47, pp. 4637–4643. IFAC (2014)
44. Rigatos, G.G.: A differential flatness theory approach to observer-based adaptive fuzzy control of MIMO nonlinear dynamical systems. Nonlinear Dyn. **76**(2), 1335–1354 (2014)
45. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Propagation. California University, San Diego (1985)
46. Sa'ad, H.H.Y., Isa, N.A.M., Ahmed, M.M., Sa'd, A.H.Y.: A robust structure identification method for evolving fuzzy system. Expert Syst. Appl. **93**, 267–282 (2018)
47. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **SMC–15**(1), 116–132 (1985)
48. Vandersteen, G.: On the use of compensated total least squares in system identification. IEEE Trans. Automat. Contr. **43**(10), 1436–1441 (1998)
49. Varshney, A.K., Torra, V.: Literature review of the recent trends and applications in various fuzzy rule-based systems. Int. J. Fuzzy Syst. 1–24 (2023)
50. Wai, R.J., Muthusamy, R.: Fuzzy-neural-network inherited sliding-mode control for robot manipulator including actuator dynamics. IEEE Trans. Neural Netw. Learn. Syst. **24**(2), 274–284 (2013)
51. Yang, J., Shang, C., Li, Y., Li, F., Shen, Q.: ANFIS construction with sparse data via group rule interpolation. IEEE Trans. Cybern. **51**(5), 2773–2786 (2021)
52. Yeh, J.W., Su, S.F.: Efficient approach for RLS type learning in TSK neural fuzzy systems. IEEE Trans. Cybern. **47**(9), 2343–2352 (2017)
53. Yu, F., Mao, Z., Jia, M., Yuan, P.: Recursive parameter identification of Hammerstein–Wiener systems with measurement noise. Signal Process. **105**, 137–147 (2014)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.