Implementation of an Evolving Fuzzy Model (eFuMo) in a Monitoring System for a Waste-Water Treatment Process

Dejan Dovžan, Vito Logar, and Igor Škrjanc

***Abstract*—Increasing demands on effluent quality and loads call for an improved control, monitoring, and fault detection of waste-water treatment plants (WWTPs). Improved control and optimization of WWTP lead to increased pollutant removal, a reduced need for chemicals as well as energy savings. An important step toward the optimal functioning of a WWTP is to minimize the influence of sensor faults on the control quality. To achieve this, a fault-detection system should be implemented. In this paper, the idea of using an evolving method as a base for the fault-detection/monitoring system is tested. The system is based on the evolving fuzzy model method. This method allows us to model the nonlinear relations between the variables with the Takagi–Sugeno fuzzy model. The method uses basic evolving mechanisms to add and remove clusters and the adaptation mechanism to adapt the clusters' and local models' parameters. The proposed fault-detection system is tested on measured data from a real WWTP. The results indicate the potential improvement of the WWTP's control during a sensor malfunction.**

***Index Terms*—Adaptation of fuzzy model, evolving fuzzy model (eFuMo), evolving mechanisms, fault detection, soft-sensor, waste-water treatment plant (WWTP).**

I. Introduction

INCREASING demands on effluent quality and increasing loads call for an improved control, monitoring, and fault detection of waste-water treatment plants (WWTPs). Applied research in automatic control is one of the important tools in achieving an overall high performance of the plant. The improved control and optimization of a WWTP can lead to an increased pollutant removal, a reduced need for chemicals as well as energy savings. One step toward an optimal WWTP control is the implementation of a monitoring system [1]. An efficient monitoring system should be able to detect the fault, isolate it, and manage it in such a way that the quality of the control is not compromised [2]. In [3], a model-predictive-control algorithm was designed and tested on a WWTP. From the measured responses, it clear that the quality of the control is compromised by the failure of a sensor. It was pointed out in the paper that the sensor's failure caused an undesired peak in the process output. To improve the control during the sensor failures, a monitoring system should be implemented, which is able to detect the sensor's failure and estimate its output during the failure period.

Over the years, the monitoring system design for WWTPs has received a lot of attention. They usually implement a soft sensor for estimating the variables that cannot be measured online. In [4], a soft sensor for estimating the phosphor concentration is proposed. The design is based on different approaches, such as partial least squares, multiple linear regression, and principal component regression. In [5]–[9], a soft sensor for estimating the oxygen-dissolving rate and the respiration of microorganisms is proposed. The two variables are estimated from the dissolved oxygen in the aerobic tank and the air flow. In the papers, different algorithms are proposed: modified Luenberg's algorithm [5], generalized damped least squares, [6], support vector machine [7], and Kalman filter [8], [9].

In [10]–[12], the online estimation of ammonia and nitrogen oxide is studied. A soft sensor is designed based on the online dissolved oxygen, pH, and oxidation redox potential measurements. In [10], a soft sensor is based on GMDH-type polynomial neural networks. To improve the estimation accuracy, they propose an additional fuzzy compensator based on empirical rules. In [11], a soft sensor is based on the Kalman filter. In [12], a hybrid neural network [neural network in combination with a principal component analysis (PCA)] is proposed for estimating the Kjeldahlov nitrogen concentration.

In [13], a monitoring system for a WWTP is presented. They use soft sensors for estimating the unmeasurable variables (chemical oxygen demand, biochemical oxygen demand, and nitrogen and phosphate concentration) and for the fault detection of the online measurable variables (temperature, flow, redox potential, pH, conductivity, and the opacity of the waste water in the tanks). The models used in soft sensors are built using neural networks with the ARX local model structure. The fault detection of the online measurable variables is based on the adaptive PCA models. The adaptation of the PCA model is achieved by implementing the sliding-window concept. For each online sensor, a sensor-validity index is calculated. If the index threshold is breached, the alarm is raised, and the sensor output is estimated with an appropriate model. An interval observer for monitoring the variables of the WWTP is proposed in [14].

In [1], [15] and [16], a soft sensor based on PCA for monitoring the online measurements is proposed. The abnormal behavior is detected by T^2 and Q statistics. In [15], the pH, dissolved oxygen level, and oxidation reduction potential are monitored. In [14], the ammonia and NO_x are monitored.

Manuscript received June 5, 2014; revised September 9, 2014; accepted October 28, 2014. Date of publication December 9, 2014; date of current version October 2, 2015.

The authors are with the Laboratory of Autonomous Mobile Systems, Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia (e-mail: dejan.dovzan@fe.uni-lj.si; vito.logar@fe.uni-lj.si; igor.skrjanc@fe.uni-lj.si).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2014.2379252

A more detailed overview of the state-of-the-art soft sensors for WWTPs can be found in [17]. More general information about soft sensors and fault-detection approaches can be found in [18] and [19].

In this paper, we propose a monitoring system for a WWTP based on a novel evolving fuzzy model (eFuMo) method. Since the WWTP is a nonlinear and time-varying process, the theoretical models are hard to tune and may not produce accurate results [14]. It can be seen in [3] that simplified theoretical models produce approximations that differ substantially from the measured values. Therefore, we propose to use an eFuMo method for building the models used in the monitoring system. The method utilizes the Takagi–Sugeno (T–S) fuzzy model to describe the relations between the model input and the output variables. It builds the model online by adding new or removing old local models, if necessary, and adapting the parameters of the local models and clusters. The advantage of online fuzzy model identification is that the model is able to adapt to the current process behavior. This is useful, especially for processes with changing dynamics, such as WWTPs. The proposed system is not meant to replace physical sensors; it can only be used for a short time prediction of the sensor's output, when the sensor is offline or faulty.

Depending on the learning abilities, the online fuzzy-identification methods can be categorized into: *adaptive methods* (e.g., ANFIS [20], GANFIS [21], rFCM [22], rGK [23]), where the initial structure of the fuzzy model must be given. The number of space partitions/clusters does not change over time, only the parameters of the membership functions and local models are adapted; *incremental methods* (e.g., RAN [24], SONFIN [25], SCFNN [26], NeuroFAST [27], DENFIS [28], eTS [29], FLEXFIS [30], PANFIS [31]), where only adding mechanisms are implemented; *evolving methods* (e.g., SAFIS [32], SOFNN [33], GAP-RBF [34], EFuNN [35], [36], D-FNN [37], GD-FNN [38], ENFM [39], eTS+ [40], ENFM [39], FLEXFIS++ [41], AHLTNM [42], SOFMLS [43]) which, besides an adding mechanism, implement removing and some of them also merging and splitting mechanisms. More on evolving methods can be found in [44] and [45], where concepts and open issues regarding these methods are presented.

This paper is organized in the following order. First, the problem is defined, then follows the description of the methodology, where the used learning method is briefly described and the idea of the monitoring system is given. In Section IV, the results are given, accompanied with some comments. At the end, some conclusions are drawn.

II. Problem Statement

The monitoring system proposed in this paper was designed and tested on measured data from a pilot WWTP, shown in Fig. 1 and an operational WWTP stationed near Ljubljana. The pilot plant consists of two anoxic reactors, two aerobic reactors, and an additional reactor, where the water is collected before returning as an internal recycle or passing down to the settler. To ensure the homogeneity, the waste water is mixed by mixers in the anoxic reactors and by air flow in the aerobic reactors.

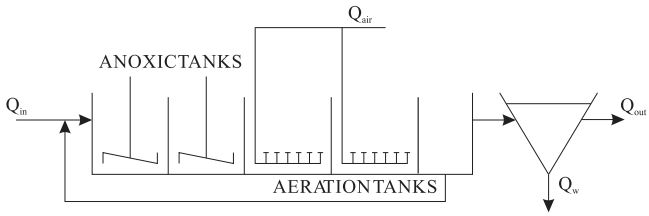

Fig. 1. Scheme of the MBBR.

The measurements made available to us were flow rate and ammonia concentration at the influent, air-flow rate in the aeration system, oxygen concentration in the first and second aerobic reactors, ammonia concentration in the last aerobic reactor, and waste-water temperature at the effluent. From the operational WWTP, we obtained the measurements of flow rate, temperature, total organic carbon (TOC) concentration, total nitrogen (TN) concentration, suspended solids, nitrates and ammonia concentration at the effluent, and TN, TOC, and ammonia concentration at the influent. The data for both WWTPs were sampled with a sampling time of 20 s. The measurement of the TN and TOC were refreshed every 5 min by an inline analyzer. The influent to the plant is the waste water after a mechanical primary.

The purpose of the control is to keep the effluent ammonia concentration in a specified bound. The control scheme consists of three controllers. The Model predictive controller controls the ammonia by changing the oxygen-concentration set point of the last aerobic reactor. The inputs to the controller are the specified set point of the effluent ammonia, the influent ammonia concentration, the waste-water temperature, and the effluent ammonia concentration. The oxygen proportional-integral (PI) controller controls the air-flow set point. The inputs to the controller are the oxygen set point and the oxygen concentration in the last aerobic reactor. The second PI controller controls the air flow by opening the air valve. The inputs to the controller are valve opening and air flow. Only the total air flow into the aerobic reactors can be manipulated. Around half of the total air flow goes in the first aerobic reactor and the other half in the second aerobic reactor. Detailed description of the process and control can be found in [3].

As stated in Section I, the problem with the WWTPs is that some key sensors often fail. Such a sensor failure affects the quality of the WWTP's control. In [3], where a model-predictive-control algorithm was tested on a WWTP, this issue was pointed out. The situation is shown in Fig. 2. It is clear that the malfunction of the influent ammonia sensor causes a peak in the effluent ammonia concentration. The fault is usually a consequence of the sensor-cleaning procedure. The sensors are periodically cleaned and, therefore, turned OFF. However, judging by the description of the control scheme in [3], we believe that the reported failures might be caused by a broken internet link. A monitoring system that can detect the sensor failures of key variables used in a control algorithm and estimate the sensor output would minimize the effects of the sensor failures on the control quality. Although it will take a lot of time and additional research in order to implement such a system for

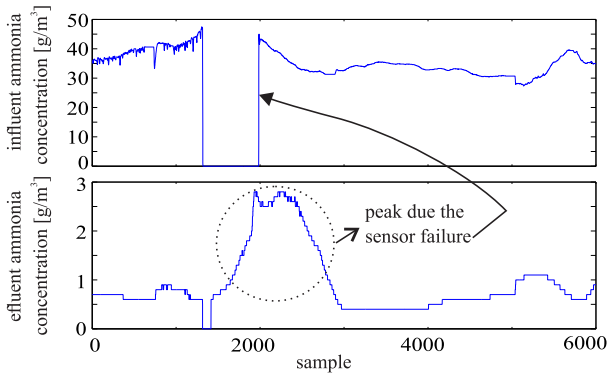

Fig. 2. Effect of sensor failure.

everyday use in the control loop, there is a possible immediate use for it in the postprocessing of the data. Namely, each year, the WWT companies have to prepare an annual report of the functioning of the WWT process. The report includes the trends of the measured variables for each day. They must also specify the intervals when the sensors were offline or faulty. The postprocessing of the data is currently conducted manually. The presented monitoring system could help speed up the process by automatically detecting intervals when the sensors were offline and providing estimated values for that time.

The monitoring system presented in this paper is able to detect and estimate the values of the following signals: air flow, oxygen concentration, and influent ammonia concentration.

III. Methodology and Materials

In order to estimate the sensor values during the sensor failure, the monitoring/fault-detection system (FDS) should be based either on a process model or on a model that describes the relations among the monitored variable and the other measured variables. Environmental systems, such as WWTPs, possess several characteristics that make their modeling and control difficult: They involve interactions between physical–chemical and biological processes, they are stochastic, and they are very often periodic in time, complex, and evolve over time [46]. It is also very difficult to identify the true simulation model of the process due to the control loop. Therefore, we propose to use the eFuMo method to model the relations between the process variables. The eFuMo method is based on the T–S fuzzy model, which is a very powerful engineering tool to approximate nonlinear processes within a required accuracy, provided that enough regions are given [47]. The method implements algorithms for the online learning of the fuzzy model, such as different adaptation algorithms, algorithms for adding new clusters and local models, removing them when they are not valid, merging them, and splitting them. The used eFuMo algorithms are described in the next sections.

A. Evolving Fuzzy Model Method

The eFuMo method can be divided into three major parts: the *central decision logic (CDL)* that is responsible for monitoring the method procedure and executing the calls of the necessary algorithms (evolving and adaptation mechanisms); the *evolving mechanisms* that calculate the conditions for the adding, removing, splitting, and merging of clusters; and the *adaptation mechanism* that is responsible for adapting the clusters' fuzzy covariance matrices and centers.

The inputs to the eFuMo identification method are the clustering vector ($\mathbf{x}_f$), the regression vector ($\mathbf{x}$), the output of the process (y), and the number of the current sample (i). The CDL first checks the current sample number (i), the sample number when the last change in the cluster number was made, and the user-defined time delay N_{wait}. If the sum of these two values is smaller than the current sample number, the evolving mechanisms are called. Otherwise, the CDL skips the call to the evolving mechanisms and continues with the call to the adaptation mechanisms. This was implemented to allow the fuzzy model to adapt the new structure to the data, before making any decisions about the removing and adding of clusters. This approach was also used in [27].

The CDL first calls the adding mechanism, then the removing mechanism, follows the merging mechanism, and at the end, the CDL calls the splitting mechanism. If one of the mechanisms changes the fuzzy structure, other evolving mechanisms that follow are not called and the eFuMo continues with the adaptation algorithm. The adaptation algorithm is called for every sample. The CDL block also calculates the variances and means of the input and output variables and updates the correlation coefficient used by the supervised merging mechanism. The detailed workflow of the method can be found at `http://msc.fe.uni-lj.si/efumo.asp`.

More detailed descriptions of the mechanisms are given in the following sections.

B. Adaptation Mechanisms

The adaptation algorithm adapts the clusters (its centers and fuzzy covariance matrix) and the local models' parameters. For the clustering, the online Gustafson–Kessel (GK) clustering is used. For the local linear models' parameter identification, the weighted least-squares method is used. A detailed derivation of the adaptation mechanism can be found in [23]. The adaptation mechanism first performs online clustering. For every new clustering vector, first the distance to the existing clusters is calculated using the following equation:

$$\begin{aligned} \mathbf{A_i} &= \det(\mathbf{F}_i(k-1))^{\frac{1}{z}}\mathbf{F}_i^{-1}(k-1) \\ d_{ik} &= \sqrt{(\mathbf{x}_f(k)-\mathbf{v}_i(k-1))^T\,\mathbf{A}_i\,(\mathbf{x}_f(k)-\mathbf{v}_i(k-1))} \end{aligned} \tag{1}$$

where d_{ik} is the distance of the kth sample from the ith cluster, $\mathbf{x}_f(k)$ is the clustering vector, $\mathbf{F}_i$ is the fuzzy covariance matrix of the ith cluster, $\mathbf{F}_i^{-1}$ is the inverse fuzzy covariance matrix, $\det(\mathbf{F}_i)$ is the fuzzy covariance matrix determinant, $\mathbf{v}_i$ is the cluster center, and z is the size of the clustering vector. This is the default distance for the GK clustering algorithm; however, one can also use the standard Mahalanobis distance ($\mathbf{A}_i = \mathbf{F}_i^{-1}(k-1)$).

Next, the membership degrees of the clustering vector are calculated as

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}}\right)^{\frac{2}{\eta-1}}} \tag{2}$$

follows the adaptation of the sum of the past membership degrees s_i:

$$s_i(k) = \gamma_v s_i(k-1) + \mu_{ik}^{\eta}. \tag{3}$$

Then, the new cluster positions are calculated as

$$\Delta \mathbf{v}_i(k) = \frac{\mu_{ik}^{\eta}\left(\mathbf{x}(k) - \mathbf{v}_i(k-1)\right)}{s_i(k)} \tag{4}$$

$$\mathbf{v}_i(k) = \mathbf{v}_i(k-1) + \Delta \mathbf{v}_i(k). \tag{5}$$

At the end of the procedure, the adaptations of the fuzzy covariance matrix, its inverse, and the determinant are made:

$$\begin{aligned}
&\mathbf{B}_i = \frac{\mu_{ik}^{\eta}}{\mathbf{s_i(k)}}\left(\mathbf{x(k)} - \mathbf{v}_i\mathbf{(k)}\right)\left(\mathbf{x(k)} - \mathbf{v}_i\mathbf{(k)}\right)^{\mathbf{T}} \\
&\mathbf{F}_i(k) = \gamma_c \frac{s_i(k-1)}{s_i(k)}\mathbf{F}_i(k-1) + \mathbf{B}_i \\
&\mathbf{C}_i = \mathbf{F}_i^{-1}\mathbf{(k-1)}\left(\mathbf{x(k)} - \mathbf{v}_i\mathbf{(k)}\right)\left(\mathbf{x(k)} - \mathbf{v}_i\mathbf{(k)}\right)^{\mathbf{T}} \\
&\times \mathbf{F}_i(k-1)^{-1}\mathbf{D}_i = \gamma_c \frac{s_i(k-1)}{\mu_{ik}^{\eta}} + \left(\mathbf{x}(k) - \mathbf{v}_i(k)\right)^{T} \\
&\times \mathbf{F}_i(k-1)^{-1}\left(\mathbf{x}(k) - \mathbf{v}_i(k)\right) \\
&\mathbf{F}_i^{-1}(k) = \frac{1}{\gamma_c}\frac{s_i(k)}{s_i(k-1)}\left[\mathbf{F}_i^{-1}(k-1) - \frac{\mathbf{C}_i}{\mathbf{D}_i}\right]
\end{aligned} \tag{6}$$

$$\mathbf{E}_i = \frac{\mathbf{1}}{\gamma_\mathbf{c}}\frac{\mu_{\mathbf{ik}}^{\eta}}{\mathbf{s_i(k)}}\left(\mathbf{x(k)} - \mathbf{v}_i\mathbf{(k)}\right)^{\mathbf{T}}\mathbf{F}_i^{-1}\mathbf{(k-1)}\left(\mathbf{x(k)} - \mathbf{v}_i\mathbf{(k)}\right) \tag{7}$$

$$\det\left(\mathbf{F}_i(k)\right) = \left(\gamma_c \frac{s_i(k-1)}{s_i(k)}\right)^{z+1} \det\left(\mathbf{F}_i(k-1)\right)\left(1 + \mathbf{E}_i\right). \tag{8}$$

Since this algorithm is used for an online identification, we do not have control of the sample order. It can happen that for a longer period of time, only the data belonging to one cluster will arrive to the algorithm. In this case, other clusters would move to that area. In order to prevent that, a *membership-cut* criterion was introduced. This means that the membership degrees that are lower than a user-specified constant are set to zero. This way, the clustering vector does not have an effect on the distend clusters. By means of previous experimentation and testing, it was found that the value of the constant should be set somewhere between 0.1 and 0.3 (in this paper, the value 0.3 was used).

After the clustering, the local models' parameters are adjusted. First, the input membership degrees are calculated. In this paper, we use radial membership functions (Gaussian membership functions) defined as

$$\mu_{ik_j} = e^{-\frac{\left(x_{f_j}(k) - v_{i_j}(k)\right)^2}{2\eta_m F_{i_{jj}}(k)}}, \; j = 1, 2, ...z-1 \;\; i = 1, 2, ..., c \tag{9}$$

where x_{f_j} is the jth element of the clustering vector, v_{i_j} is the jth component of the ith cluster center, η_m is the overlapping factor, and $F_{i_{jj}}$ is the jth diagonal element of the fuzzy covariance matrix. The membership degrees under (9) are calculated for each element of the clustering vector except the last element. The last element of the clustering vector is the output of the process. To obtain the membership degree of the clustering vector to cluster the component, the membership degrees are combined as

$$\beta_{ik} = \prod_{j=1}^{z-1} \mu_{ik_j}. \tag{10}$$

Then, the membership degrees are normalized:

$$\beta_{ik} = \frac{\beta_{ik}}{\sum_{j=1}^{c}\beta_{jk}}, \quad i = 1, 2, ..., c. \tag{11}$$

One can also use the ellipsoidal Gaussian membership functions [48]. The next step is constructing the regressors for each local model:

$$\boldsymbol{\psi}_i(k) = \beta_{ik}\,\mathbf{x}(k), \;\; y_i(k) = \beta_{ik}\,y(k), \quad i = 1, 2, ..., c. \tag{12}$$

In addition, the final step is applying the recursive least-squares method for each local model:

$$\begin{aligned}
\mathbf{P}_i(\mathbf{k}) &= \frac{1}{\lambda_r}\left(\mathbf{P}_i(k-1) - \frac{\mathbf{P}_i(k-1)\boldsymbol{\psi}_i(k)\boldsymbol{\psi}_i^T(k)\mathbf{P}_i(k-1)}{\lambda_r + \boldsymbol{\psi}_i^T(k)\mathbf{P}_i(k-1)\boldsymbol{\psi}_i(k)}\right) \\
\boldsymbol{\theta}_i(k) &= \boldsymbol{\theta}_i(k-1) + \mathbf{P}_i(k)\boldsymbol{\psi}_i(k)\left(y_i(k) - \boldsymbol{\psi}_i^T(k)\boldsymbol{\theta}_i(k-1)\right).
\end{aligned} \tag{13}$$

Different least-squares techniques can be used for adapting the local model parameters (see, e.g., [29], [39], [49], and [50]).

C. Evolving Mechanisms

Evolving mechanisms are called to upgrade the fuzzy model structure. The eFuMo method implements different adding, removing, merging, and cluster-splitting algorithms. Next, the used evolving mechanisms will be described.

1) Adding Mechanism: It is one of the most important mechanisms. It adds new clusters to the fuzzy model structure and improves the fuzzy model performance. In the literature, there are several different conditions for adding new clusters based on the model output error, the distance of the current sample to the existing cluster, and the ϵ-completeness, which is based on the current sample's membership degree to existing clusters.

In this paper, the distance condition is used for the cluster adding. With the eFuMo method, different distances can be used. In this application, the normalized Mahalanobis distance was used. When using the normalized Mahanalobis distance, the normalization vector is formed from diagonal elements of

the fuzzy matrix:

$$\mathbf{s}_{i_{\text{norm}}} = \left[\sqrt{f_{i_{11}}}\ \sqrt{f_{i_{22}}} \cdots \sqrt{f_{i_{pp}}}\right]^T. \tag{14}$$

The normalized distance is then calculated as

$$d_{i_{\text{norm}}} = \frac{((\mathbf{x}_f(k) - \mathbf{v}_i)^T \mathbf{F}_i^{-1}(\mathbf{x}_f(k) - \mathbf{v}_i))^{0.5}}{k_n(\mathbf{s}_{i_{\text{norm}}}^T \mathbf{F}_i^{-1} \mathbf{s}_{i_{\text{norm}}})^{0.5}} \tag{15}$$

where k_n is a user-defined constant. The cluster satisfies the distance-adding criterion if the normalized distance is greater than 1.

A new cluster center is positioned to the current clustering vector ($\mathbf{v}_i = \mathbf{x}_f(k)$). The fuzzy covariance matrix of a new cluster is initialized as follows:

$$\mathbf{F}_i = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_z^2 \end{bmatrix} \tag{16}$$

where the diagonal elements of the matrix depend on the distance to the nearest cluster and the allowed influence zone. The influence zone ϵ_β is defined by the user and represents the membership degree of a nearest cluster to the newly created one. The diagonal elements are calculated as

$$\sigma_j^2 = -\frac{d_j^2}{2\eta_m \ln(\epsilon_\beta)} \tag{17}$$

where d_j is the distance between the jth element of the new cluster center and the cluster center nearest to it. If the distance d_j is zero, we take the jth diagonal element of the fuzzy covariance matrix of the nearest cluster.

The parameters of the new local model are initialized as the weighted mean:

$$\theta_{i+1,j} = \frac{\sum_{i=1}^{c} \omega_{ij}\theta_{ij}}{\sum_{i=1}^{c} \omega_{ij}}, \quad j = 1, \ldots z+1 \tag{18}$$

where the weights are a combination of the membership degrees β_i and the local model parameter's variance $P_{i_{jj}}$:

$$\omega_{ij} = \beta_i \frac{1}{P_{i_{jj}}}. \tag{19}$$

The parameters for the first local model are initialized with zero. The initial position of the first cluster center is the same as the first clustering vector ($\mathbf{x}_f(1)$). The fuzzy covariance matrix of the first cluster is initialized in the same manner as explained above. Except that instead of the distance to the closest cluster, the expected range of the input and output variables and the expected number of clusters are used to calculate the distance d_j.

It is also important to mention that if a certain clustering vector satisfies the distance-adding criterion, a new cluster will not necessarily be created. In order for a new cluster to be created, at least N_c previous samples must satisfy the distance-adding criterion. This reduces influence of the outliers [51].

2) Removing Mechanism: It is for removing old clusters and clusters created based on outliers. There are two conditions for removing: a minimum existence condition and a support-age ratio condition [40]. The minimum existence condition removes the clusters that in a certain period after creation (k_{delay}) do not receive enough support samples (N_{s_i}). Support is the number of samples (clustering vectors) belonging to the cluster. The sample always belongs to the closest cluster. The support-age condition is based on the clusters' supports (N_{s_i}) normalized with the clusters' age [see (20)]. The cluster with a ratio lower than a percentage (ε) of the mean ratio is deleted. Age (a_i) is defined as the number of samples from the cluster's creation k_i to the current sample k:

$$a_i = k - k_i, \ \ S_{n_i} = \frac{N_{s_i}}{a_i}. \tag{20}$$

Both conditions for removing can be written as

$$\textbf{IF } S_{n_i} < \varepsilon \text{ mean}(S_n) \textbf{ OR } \big(N_{s_i} < N_{s_{\text{trh}}} \textbf{ AND } k > k_i + k_{\text{delay}}\big) \textbf{THEN} \text{ remove } i \text{ th cluster.} \tag{21}$$

3) Splitting Mechanism: The eFuMo's splitting mechanism is based on the relative model error that clusters gather over time. The error is updated every time the splitting mechanism is called, and the current sample does not satisfy the distance-adding condition. First, the relative model error is calculated:

$$e(k) = \frac{|y_m(k) - y(k)|}{n_\sigma \sigma_y} \tag{22}$$

where y is the real output, and y_m is the model output. σ_y represents the current standard deviation of the process output and is calculated as

$$\begin{aligned} \bar{y}(k) =& \ \frac{1}{k}\left((k-1)\bar{y}(k-1) + y(k)\right) \\ \sigma_y^2(k) =& \ \frac{1}{k}\left((k-1)(\sigma_y^2(k-1) + \bar{y}(k-1)^2) + y(k)^2\right) - \\ & - \frac{1}{k^2}\left((k-1)\bar{y}(k-1) + y(k)\right)^2. \end{aligned} \tag{23}$$

The factor n_σ was set as 3.4. The value was established by experimentation. In most cases, by multiplying the standard deviation with this factor, the whole range of the variable is obtained. The error is then divided among the existing clusters and added to the previous cluster error:

$$e_{\text{sum}_i}(k) = e_{\text{sum}_i}(k-1) + \beta_i e(k) \tag{24}$$

where β_i is the membership degree of the current sample to the ith cluster. The splitting mechanism checks the cluster with the largest error. If its support from the last change (N_{sl_i}) is greater than a threshold ($N_{sl_{\text{trh}}}$) and its mean relative error is larger than a threshold value, the cluster is split. The error threshold is set by the user, specifying the maximum ($e_{\max}$) and minimum ($e_{\min}$) error threshold and the decay constant (T). The current threshold is calculated as

$$e_{\text{trh}} = \max(e_{\max}\exp(-N_e/T), e_{\min}) \tag{25}$$

where N_e are the number of samples that are used for the error calculation and the decay constant, respectively. If a cluster is

added to or removed from the structure, N_{sl_i}, N_e, and e_{sum_i} are set to zero.

The positions of the split clusters are calculated using the diagonal elements (vector $\mathbf{s}_{i_{\text{norm}}}$) of the fuzzy covariance matrix:

$$\mathbf{v}_{i1} = \mathbf{v}_i + 0.5\mathbf{s}_{i_{\text{norm}}}, \;\; \mathbf{v}_{i2} = \mathbf{v}_i - 0.5\mathbf{s}_{i_{\text{norm}}} \tag{26}$$

where i is the index of the cluster that is split. The new center positions can also be calculated using the singular value decomposition, as in [52]. The fuzzy covariance matrix, the support, and the sum of the past membership degrees are set to half of their original values for both clusters. The time of the cluster creation is initialized as the creation time of the original cluster for both clusters.

4) Merging Mechanism: There are two types of merging algorithms implemented in eFuMo: supervised and unsupervised. The eFuMo unsupervised merging merges the clusters that are close together. The similarity and the vicinity of the two clusters are measured using the normalized distance:

$$d_{ik}^2 = \; (\mathbf{v}_i - \mathbf{v}_k)^T F_i^{-1} (\mathbf{v}_i - \mathbf{v}_k), \; i,k = 1,...,c \; i \neq k \tag{27}$$

$$d_{\text{norm}_{ik}} = \sqrt{\frac{d_{ik}^2}{2\mathbf{s}_{i_{\text{norm}}}^T F_i^{-1} \mathbf{s}_{i_{\text{norm}}}}}. \tag{28}$$

The distances are calculated only for clusters that have larger support N_{sl_i} than a user-defined threshold $N_{sl_{\text{trh}}}$. The clusters are considered for merging if both normalized distances $d_{\text{norm}_{ik}}$ and $d_{\text{norm}_{ki}}$ are shorter than the logarithm of the predefined threshold ϵ_{β_m}. If this criterion is satisfied, the distance ratio is checked. If the ratio is above the user-defined threshold $k_{d_{\text{merge}}}$, clusters are merged. The rule for merging can be written as

$$\begin{aligned} &\textbf{IF } d_{\text{norm}_{ik}} < \sqrt{-\ln(\epsilon_{\beta_m})} \textbf{ AND } d_{\text{norm}_{ki}} < \sqrt{-\ln(\epsilon_{\beta_m})} \\ &\textbf{AND} \frac{1 - \min(d_{\text{norm}_{ik}}, d_{\text{norm}_{ki}})}{\max(d_{\text{norm}_{ik}}, d_{\text{norm}_{ki}})} < k_{d_{\text{merge}}} \\ &\textbf{THEN} \text{ merge } i\text{th and } k\text{th cluster.} \end{aligned} \tag{29}$$

The parameters of the new cluster are initialized as a weighted mean. The fuzzy covariance and the centers are initialized as proposed in [39]. The new sum of the past membership degrees is calculated as the weighted mean, where the weights are the clusters' support. The support of the merged cluster and the time of creation are calculated as the weighted mean, where weights are the sums of the past membership degrees (s_i, s_k). The parameters of the new merged local model are also calculated as the weighted mean using a combination of clusters' support (Ns_i) and the variance of each parameter as the weights.

The supervised merging considers the difference between the local models. It is meant to merge the neighborhood clusters that have approximately the same shape and local model but are not close enough for the unsupervised merging to merge them. In order to detect these clusters, three different measures are used. The algorithm uses the angles between the local models' parameters (angle-merging condition), the correlation between the membership degrees (correlation-merging condition), and the distance ratio (distance-ratio-merging condition). The correlation coefficient is calculated based on the monitoring of the membership degrees and their products $\beta_{ij}(k) = \beta_{ij}(k-1) + \beta_i(k)\beta_j(k)$, $\beta_{ii}(k) = \beta_{ii}(k-1) + \beta_i(k)\beta_i(k)$ and is calculated as

$$C_{ij}(k) = \frac{\beta_{ij}}{\beta_{ii}^{0.5}\beta_{jj}^{0.5}}. \tag{30}$$

If the coefficient $C_{ij}(k)$ is above the user-defined threshold, the clusters i and j are considered for merging.

The distance-ratio criterion for merging is the same as in (29). The clusters are considered for merging if the distance ratio is lower than a user-defined threshold $k_{d_{\text{merge}_s}}$ and the correlation coefficient is at least half of the threshold defined for the correlation-merging condition.

The angle-merging criterion is based on local models' angles. The idea is to compare the local models by the contribution of each input to the system output. First, the parameters are normalized with the highest absolute value:

$$\theta_{ni_k} = \frac{\theta_{i_k}}{\max\limits_{1 \leq q \leq c} |\theta_{q_k}|}. \tag{31}$$

The normalized parameters of the two local models i and j are then compared by means of the angle that they represent in a one-input one-output space:

$$\alpha_{ij_k} = |\arctan(\theta_{ni_k}) - \arctan(\theta_{nj_k})| \tag{32}$$

where k is the parameter index. The clusters are considered for merging if all the angles α_{ij_k}, $k = 1,..,z$, where z is the number of the local model's parameters, are below the user-defined threshold and the correlation coefficient is at least half of the threshold defined for the correlation-merging condition.

After the eFuMo identifies the possible merging pairs with the correlation, the angle, and the distance-ratio conditions, it then estimates the error made when merging the local models:

$$\begin{aligned} e_1 &= \left|\theta_i^T \mathbf{x}_1 - \theta_j^T \mathbf{x}_1\right| \\ e_2 &= \sum_{r=1}^{z-1} |\theta_{i_r}(x_{1_r} + 2\sigma_{u_{r-1}}) - \theta_{j_r}(x_{1_r} + 2\sigma_{u_{r-1}})| \\ e_3 &= \sum_{r=1}^{z-1} |\theta_{i_r}(x_{1_r} - 2\sigma_{u_{r-1}}) - \theta_{j_r}(x_{1_r} - 2\sigma_{u_{r-1}})| \\ e &= \frac{1}{3n_\sigma \sigma_y} \sum_{r=1}^{3} e_r \end{aligned} \tag{33}$$

where $\mathbf{x}_1 = [1, \bar{u}_1, ..., \bar{u}_{z-1}]^T$, $\bar{u}$ is the mean value of a certain input variable, $\sigma_{u_{r-1}}$ is its standard deviation, σ_y is the standard deviation of the process output, $z-1$ is the number of inputs, and θ_{i_r} is the rth parameter of the ith local model. If the clustering vector is the same as the regression vector, the algorithm checks for the model error only for the region where the two local models are valid. The mean value of the inputs and their standard deviations are calculated from the cluster center vector and the fuzzy covariance matrix. The pair with the lowest error below the threshold is merged. Note that two cluster are only merged if no other clusters are between them.

The center of the merged cluster is positioned in the middle between the maximum and minimum borders of both clusters:

$$\begin{aligned}\mathbf{v_{new}} &= \mathbf{v}_j + \mathbf{d}_2 \\ \mathbf{d}_1 &= \mathbf{v}_i - \mathbf{v}_j,\ \ \mathbf{d}_2 = \frac{\mathbf{v}'_i - \mathbf{v}'_j}{2} \\ \mathbf{v}'_i &= \mathbf{v}_i + \text{sign}(\mathbf{d}_1)\mathbf{s}_{i_{\text{norm}}},\ \mathbf{v}'_j = \mathbf{v}_j - \text{sign}(\mathbf{d}_1)\mathbf{s}_{j_{\text{norm}}}. \quad (34)\end{aligned}$$

The fuzzy covariance matrix of a new merged cluster is initialized as the sum of both clusters' fuzzy covariance matrices; in the same way, the cluster's support is initialized. The local model parameters are initialized as the mean of both local models' parameters. The creation time is initialized to the creation time of the oldest cluster in the pair. The sum of the past membership degrees is initialized to the maximum sum of the past membership degrees of both clusters.

D. About the Parameters

The evolving methods usually have a number of parameters that need to be tuned properly in order to achieve good models. The number of parameters is an inherent problem of all self-adjusting, adaptive, or learning algorithms. On the one hand, the parameters make the methods very flexible to a variety of data and problems, while on the other hand, the tuning of the parameters can be a very time-consuming task. In order to help tune the eFuMo method, some guidelines for the parameters are given in the next sections.

1) Adaptation Mechanism: For the adaptation mechanisms, the user must set the following parameters: the fuzziness factor η. With this parameter, we control the overlapping of the membership functions under (2). The usual value for this factor is 2 [53]. Higher factors mean smoother transitions between the clusters. However, it is not recommended to set the factors too high. When setting the factor to 1, crisp transitions between the clusters are obtained. The overlapping factor η_m has the same effect on the membership functions defined by (11). The standard value in this case is 1. Both factors also affect the accuracy of the model. In general, you will get more accurate local models when setting η_m to lower values (more crisp transitions between the clusters). In [54], a study of the fuzziness factor and overlapping factor on the fuzzy model's accuracy is presented. The study also deals with the impact of the forgetting factors. The factors are introduced to handle the concept drift [55]. The forgetting factors γ_c and γ_v should be set to the same value. If the γ_c is higher than γ_v, the fuzzy covariance matrix is larger than it should be. If γ_c is lower than γ_v, the fuzzy covariance matrix is smaller than it should be. When setting the forgetting factors, we can use the equation introduced in [56]: $\lambda = 1 - \frac{2}{N}$, where λ is the forgetting factor, and N is the width of the forgetting window (the number of samples that have an effect on the estimate). With the forgetting factors, we control the speed of the adaptation. The initial value of s_i controls the initial speed of the adaptation. Theoretically, the initial value for s_i should be 1. However, when setting it higher, the initial speed of the adaptation can be lowered. The equation for the center adaptation can be written in the following form:

$$\mathbf{v}_i(k) = \left(1 - \frac{\mu_{ik}^{\eta}}{s_i(k)}\right)\mathbf{v}_i(k-1) + \frac{\mu_{ik}^{\eta}}{s_i(k)}\mathbf{x}_f(k). \quad (35)$$

The equation represents a first-order model with a variable time constant and gain. The speed of the adaptation is governed by the term $\left(1 - \frac{\mu_{ik}^{\eta}}{s_i(k)}\right)$. The term is indirectly controlled by the forgetting factor γ_v and by the initial value of s_i. The initial covariance matrix is set as $\boldsymbol{P}_i(0) = \alpha \boldsymbol{I}$, where $\boldsymbol{I}$ is the unity matrix, and α is a large positive constant [29]. The higher the constant α, the faster the initial adaptation of the local models. However, if α is set too high (in our experiences higher than 100), spikes in the model prediction can be observed when a new cluster is added.

2) Adding Mechanism: The user-specified parameters for the adding condition are as follows: the distance-adding normalization constant k_n. The factor is usually set between 2 and 3. These values come from a 1-D problem, where the distances are usually compared to some reference distance. This distance is, in most cases, 2σ or 3σ of the data, where σ represents the standard deviation. In the case of the presented adding mechanism, the reference distance was defined as the Mahalanobis distance from the cluster center to a reference point. The distance between the reference point and the cluster center in each dimension is the same as the standard deviation of the data around the cluster center for that dimension. The result of higher k_n values is a fuzzy model with a smaller number of clusters. If k_n is set too low, the fuzzy model might have too many clusters. The parameter ϵ_β is important for the initialization of a new fuzzy covariance matrix. ϵ_β influences the spread of the membership functions. As described in [31], if the membership function is too narrow, it can lead to overfitting; too wide membership functions lead to too much averaging. ϵ_β defines the membership degree of the nearest cluster to that newly created in one dimension. The total membership degree is then ϵ_β^{z-1}. The idea is to define the variance so that the new cluster center is out of the 2σ zone. The default setting for this parameter is 0.1. However, when setting the parameter, one should also consider the dimensionality of the problem [31]. The parameter N_c is set depending on the expected duration of the outliers. If the number is set high the delay of adding a new cluster is increased, but the system is more resistant to outliers. In [40], the dynamical adaptation of N_c is proposed. The eFuMo method has a fixed threshold.

3) Removing Mechanism: The user must specify three parameters: $N_{s_{\text{trh}}}$ and k_{delay} for the minimum existence condition and ε for the support-age condition. The minimum existence condition is used for removing the clusters created based on a faulty measurement. They should be set based on the data distribution and the length of the outliers. If the data come randomly from random clusters, this criterion should be turned OFF (also N_c should be set to 0). Like with the parameter N_c, the user must estimate how many samples does it take to detect the creation of a new cluster. In [40], the values 3 and 10 are proposed for $N_{s_{\text{trh}}}$ and k_{delay}, respectively. The support-age condition removes the inactive clusters. The parameter ε is usually set to 0.01, meaning

that the clusters with a ration lower than 1% of the mean are deleted. It should be noted that this condition should be turned OFF when dealing with a process that is at one working point for a long period of time. In such a case, the nonactive clusters' ratio will decrease, and the clusters will be removed from the structure, even though they might still be valid.

4) Splitting Mechanism: It is meant for fine tuning of the fuzzy model. The adaptive error threshold is meant to allow first the adding mechanism to add clusters and cover the problem space. After that, the splitting mechanism is responsible for creating new clusters with the splitting procedure. $e_{\max}$ is usually set to 0.5 and $e_{\min}$ to 0.05. This means that we allow each cluster to have a 50% error at the start, and at the end, each cluster should have a mean error less than 5% . If the maximum and minimum thresholds are set too low, there is a possibility of overfitting the model. With the decay constant T, the threshold is slowly decreasing to allow the model adaptation and trustworthy estimation of the cluster errors. Usually, T is set to at least 100. Meaning that after 231 samples, the error threshold is equal to $e_{\min}$ (when the proposed threshold values are used). The threshold $N_{sl_{\text{trh}}}$ is a safety parameter that does not allow splitting of the newly created clusters and that ensures that at least some representative data samples are used for the cluster error calculation. Higher threshold values delay the splitting procedure, but the error estimation is more accurate. The default value is 10.

5) Merging Mechanism: The unsupervised merging mechanism has three tuning parameters. $N_{sl_{\text{trh}}}$ is the same as with the splitting mechanism and ensures that newly created clusters have at least some time to adapt. The distance ratio is used to measure the similarity of the clusters. In an ideal case, the distance ratio between two similar clusters would be 1. In practice, this is almost never true; therefore, a tolerance bound $k_{d_{\text{merge}}}$ was introduced. We usually allow the distances to differ by 10%. For unsupervised merging, the vicinity of the clusters is also important. The vicinity is measured by the distance. A threshold vicinity is given by ϵ_{β_m} . The parameter was designed in such a way that it reflects the membership degree of one cluster to the other cluster. It should be set between 0.8 and 0.9, which in terms of membership degrees means a high overlapping between two clusters. In our case, this means that the clusters that are closer than 0.67 ($\epsilon_{\beta_m} = 0.8$) or 0.45 ($\epsilon_{\beta_m} = 0.9$) of their normalized distance are merged together. The supervised merging mechanism has four parameters that need tuning: the correlation threshold C_{trh}, the distance-ratio threshold $k_{d_{\text{merge}_s}}$, the angle-difference threshold α_{trh}, and the error threshold e_{merge}. The correlation coefficient is usually set to 0.8. Lower values mean that more cluster pairs will be checked for error, slowing down the algorithm. Higher values of the parameter will result in less detected pairs, leaving some unnecessary clusters in the structure. The distance ratio $k_{d_{\text{merge}_s}}$ can be set to the same value as with the unsupervised merging. In our applications, we usually allow 2–4° of angle difference. Higher values of the distance ratio and the angle threshold have the same effect as lower values of correlation threshold. For the error threshold e_{merge}, it makes sense to set it at least to $e_{\min}$ or lower.

TABLE I
RESULTS FOR THE DYNAMICAL SYSTEM

Method	Rules	RMSE
SAFIS [32]	8	0.0116
MRAN [59]	10	0.0129
RANEKF [60]	11	0.0184
simpl_eTS [61]	18	0.0122
eTS [29]	19	0.0082
SONFIN [25]	10	0.013
SAFIN [57]	13	0.007
eFuMo [48]	12	0.0035

TABLE II
RESULTS FOR TSCD

Method	Rules	RMSE
DENFIS [28]	17	0.0510
eFuMo [48]	12	0.0543
ANYA [62]	13	0.0543
ELM [63]	20	0.1564
eHFN [64]	13	0.0936
eNNEL [58]	13	0.0761

TABLE III
RESULTS FOR BOX–JENKINS GAS FURNACE DATA WHERE ALL DATA ARE USED FOR TRAINING AND VALIDATION

Method	Rules	RMSE
DENFIS [28]	12	0.0190
eFuMo [48]	3	0.0337
ANYA [62]	7	0.0393
ELM [63]	20	0.0232
eHFN [64]	7	0.0245
eNNEL [58]	7	0.0354

E. Comparison of the eFuMo to Other Similar Methods

To show that the used evolving fuzzy model method can achieve a similar model performance to other existing online learning methods, some results on benchmark problems are given in this section. In Table I, the results are given for the modeling of the dynamical system given by (36). The parameters a, b, and c were set to 1. The input to the model is defined as $u(k) = \sin\left(2\pi\frac{k}{100}\right)$. According to [57], the learning set included 50 000 data points, and 200 data points were used for the testing

$$y(k+1) = \frac{ay(k)}{(1+by(k)^2)} + cu(k)^3. \tag{36}$$

In Table II, the results for the benchmark problem of the time-series concept drift (TSCD) are presented. The process model is given by (36). The difference compared with the previous example is that here the parameters a, b, and c are not fixed, but change over time according to the equations in [58], where the detailed description of experiment is also given.

In Tables III and IV, the results for the Box–Jenkins gas-furnace data are presented. The two tables demonstrate two different experimental setups. In Table III, the experiment is

TABLE IV
RESULTS FOR BOX–JENKINS GAS-FURNACE DATA WHERE 200 DATA ARE USED FOR TRAINING

Method	Rules	RMSE
eTS [29]	5	0.0490
simpl_eTS [61]	3	0.0485
SOFNN [33]	4	0.0480
SOFMLS [43]	5	0.0474
eFuMo [48]	4	0.0433

TABLE V
RESULTS FOR MACKEY–GLASS TIME SERIES

Method	Rules	RMSE
DENFIS [28]	58	0.0628
DENFIS [28]	27	0.0920
exTS [65]	10	0.0754
eTS+ [40]	10	0.0892
eTS [29]	113	0.0217
rGK [23]	58	0.0481
rGK [23]	10	0.0862
rFCM [22]	10	0.1039
rFCM [22]	58	0.0702
rFCM [22]	100	0.0285
eFuMo [48]	21	0.0753
eFuMo [48]	41	0.0316
eFuMo [48]	68	0.0224

set up as described in [58]. All the data are used for testing and validation. The error is calculated for the next sample. In Table IV, the experiment is set up as described in [43]. The first 200 data points are used for the learning, and the remaining 90 data points are used for the validation.

Table V presents the results for an 85-step prediction of a Mackey–Glass time series. The setup of the experiment is described in [29].

The presented results show that the eFuMo achieves a comparable degree of accuracy with that of other evolving methods. In some cases, it can even produce more accurate models than other online learning methods. In general, the eFuMo method gives us better results than its predecessors rFCM and rGK, which do not have any evolving mechanisms. The advantage of the evolving methods over the adaptive methods is also that the user does not have to define the number of clusters and initial model structure. The different number of clusters generated by the eFuMo method was achieved by setting the adding-distance normalization constant. The higher the constant, the lower the number of clusters. The detailed differences among the eFuMo settings can be found in the files provided under the previously given download link.

F. Monitoring System

As stated in Section I, the monitoring system is based on the presented eFuMo method, which is responsible for the learning and adapting the fuzzy model. Two FDSs were designed. The first was for monitoring the air flow and oxygen concentration; the second was designed for monitoring the influent ammonia concentration. Both systems are based on the same principles. They use the T–S fuzzy model to estimate the relations between the variables. To design the monitoring system, the whole WWTP was divided into three subprocesses: the air-flow process, the oxygen concentration process, and the influent ammonia process. The inputs to the FDS are the inputs to the monitored subprocess and their outputs, as shown in Fig. 3. The air flow is estimated using the information about the valve opening and the previous measurement of the air flow; oxygen concentration is estimated from the air flow, the temperature in the reactor, and the previous measurement of the oxygen concentration. For the air flow and the oxygen concentration, first-order local models were chosen. The influent ammonia is estimated using measurements of the TN concentration at the effluent and influent, the temperature of the reactor, the ammonia concentration at the effluent, and the flow rate of the effluent water. For the influent ammonia, static local models were chosen. It should be noted that the obtained models are not real process simulation models. They are used for modeling the relations between the monitored variable and other correlated variables and are only valid for short prediction horizon. The signals used as FDS inputs are shown in Fig. 4. The temperature for the first dataset ranged from 11 °C to 15 °C and for the second dataset from 14 °C to 16 °C. The inputs were selected by a backward selection. The idea is to detect the error in the process output based on the presented inputs. The outputs of the FDS are $y_{\mathrm{soft}}(k)$ and $alarm(k)$. The output $alarm(k)$ indicates the presence of the fault in the measured signal ($alarm(k) = 1$: fault detected). The output $y_{\mathrm{soft}}(k)$ is the process output with the removed fault. If there is no fault detected, the output $y_{\mathrm{soft}}(k)$ is equal to the process output $y(k)$. If the fault is detected, the output $y_{\mathrm{soft}}(k)$ is calculated based on a fuzzy model that describes the proper relations between the input signals and the monitored signals. The FDS determines the fault based on the internal fuzzy model of the signal relations. For each monitored signal, the three models are kept in the FDS's memory: a full eFuMo, an adaptive fuzzy model (parameters of clusters and local models are adapted), and a fuzzy model with fixed parameters that holds the information about the last good known parameters. The learning of the fuzzy models is delayed for 200 samples. The delay was introduced for future research to cope with slower faults. The data sample is used for learning if there was no fault detected. For each sample and each model, the relative prediction error is calculated. The calculated error (its absolute value) is assigned to the model. The prediction error assigned to the fuzzy model is combined

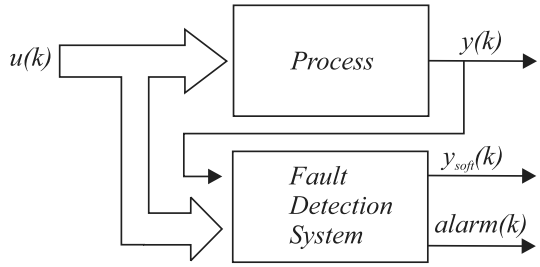

Fig. 3. FDS attached to the process.

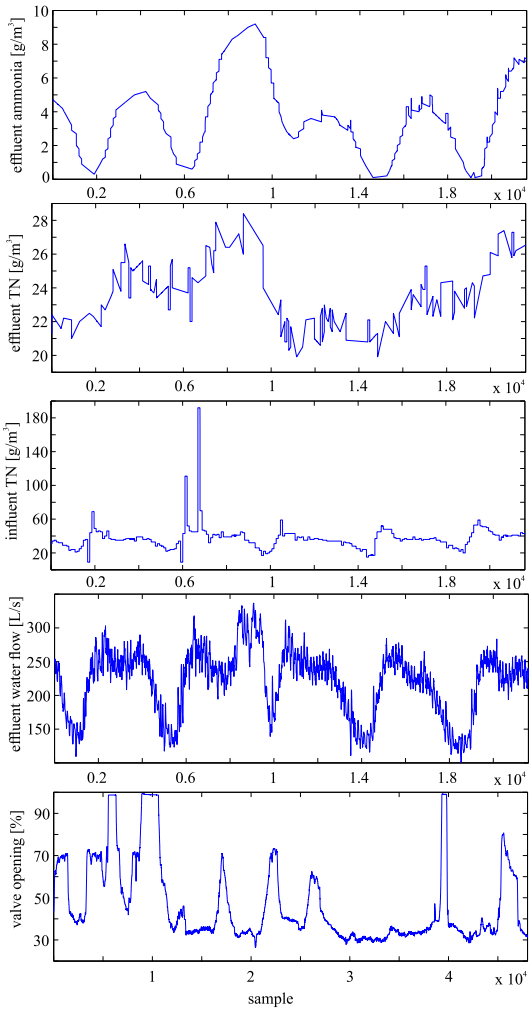

Fig. 4. Inputs to FDS.

with the simulation error, which is calculated periodically on every 200th sample using the 200 samples in the buffer. The prediction error is also used for learning the prediction-error fuzzy model. Namely, each model that describes the signal relations is accompanied by the error model. The error model is used to calculate the allowed difference between the estimated and measured signals. For estimating the sensor output during the failure, the model with the lowest assigned error is used.

The adaptive and fixed model structure and parameters can be replaced when a cluster is added or removed from the eFuMo's structure. Before the number of cluster changes, the error of each model is checked. If the evolving model has the smallest error, the adaptive and fixed model structure is replaced by the evolving model's structure. In addition, their error models are replaced. The simplified diagram of the procedure is shown in Fig. 5.

The variances denoted as $\sigma\_evolving$, $\sigma\_adaptive$, and σ_fixed are calculated from the error model:

$$\sigma = \sum_{i=1}^{c} \beta_i \sqrt{F_{i_{r,r}}} \tag{37}$$

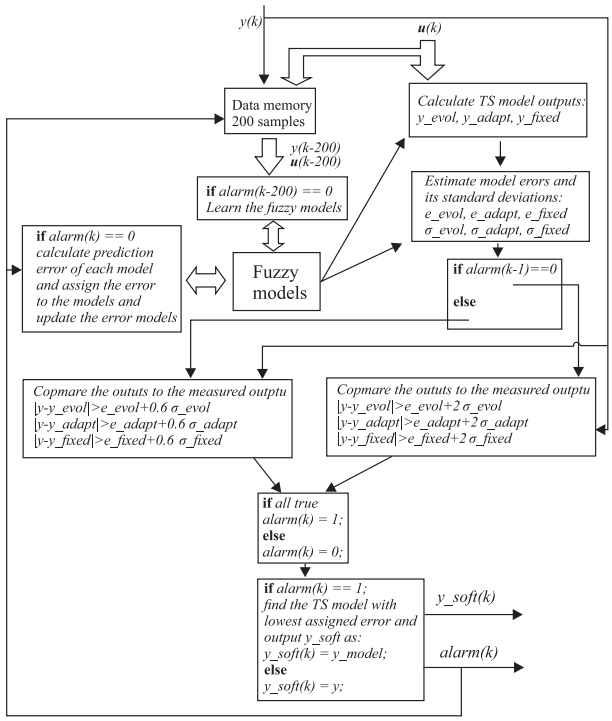

Fig. 5. Scheme of the FDS for a subprocess.

where $F_{i_{r,r}}$ is the last diagonal element of the error fuzzy model's cluster i. This element represents the variance of the error. As seen in Fig. 5, the alarm is raised if the difference between the estimated output and the measured output is higher than the maximum allowed difference. Note that the alarm is turned OFF when for at least 30 consecutive samples the difference is below the defined threshold for turning OFF the alarm. To ensure a smooth transition from the estimated output to the measured output, when the alarm is turned OFF, a filter was implemented that calculated the output of the FDS as

$$y_{\text{soft}} = \frac{((30 - k_{\text{alarm}})y_{\text{model}} + k_{\text{alarm}}\, y)}{30} \tag{38}$$

where k_{alarm} is the number of samples from the sample when the condition for turning the alarm OFF was reached. The maximum number of k_{alarm} is 30, and its value is reset to 0 every time a new alarm is raised.

G. Detecting the False Alarms Due to Manual Calibration

During the design phase, it was noticed that manual tuning of the oxygen concentration was performed. This is seen on the upper graph in Fig. 6. The drift of the sensor was manually reduced by the operator, causing the FDS to report an error. It can be seen that the shapes of the estimated and measured outputs are practically the same. However, due to an offset of the signal, the FDS detects the error. To automatically turn OFF such alarms, an additional algorithm was implemented to the FDS. This algorithm is turned ON when a new alarm is detected. Two different algorithms were tested. Both performed very similarly, as will be seen in Section IV. The first one is based on the correlation between the estimated and measured outputs, and

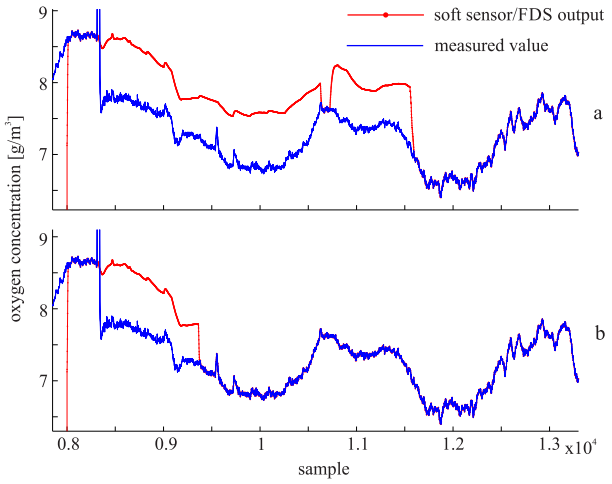

Fig. 6. Effect of sensor calibration.

the other one is based on the variance of the measured, the estimated, and the error signals.

1) Correlation Procedure: When the alarm is raised, the algorithm starts calculating the mean values of the measured ($\bar{y}$) and the estimated sensor outputs ($\bar{\hat{y}}$) using (23). After a certain number of samples (in our case 500), the algorithm stops refreshing the mean values and starts calculating the products for calculating Pearson's correlation coefficient:

$$\begin{aligned} cc_y(k) &= c_y(k-1) + (\bar{y} - y(k))^2 \\ c_{\hat{y}}(k) &= c_{\hat{y}}(k-1) + (\bar{\hat{y}} - \hat{y}(k))^2 \\ c_{y\hat{y}}(k) &= c_{y\hat{y}}(k-1) + (\bar{\hat{y}} - \hat{y}(k))(\bar{y} - y(k)). \end{aligned} \quad (39)$$

After a certain number of samples (in our case 1000), the algorithm starts to check the correlation coefficient:

$$C_{y\hat{y}} = \frac{c_{y\hat{y}}(k)}{\sqrt{c_{\hat{y}}(k)}\sqrt{c_y(k)}}. \quad (40)$$

The algorithm turns OFF the alarm automatically when the correlation coefficient rises above the user-defined threshold (in our case 0.9) and stays above it for at least 50 consecutive samples. The algorithm is turned OFF when, for at least a certain number of consecutive samples (in our case 100), the difference between the estimated and the measured outputs is below the maximum allowed difference. The algorithm is also turned OFF if its maximum functioning time is reached. In our case, the maximum functioning time was set to 700 samples, counting from the sample when the correlation coefficient reached the threshold. These values were determined ad-hoc and they depend on a specific problem and on the adaptation ability of the fuzzy model.

2) Variance Procedure: With this procedure, the algorithm starts to calculate the variances of the estimated output, the measured output, and the variance of their difference when the alarm is raised. The idea behind this solution is that the variance of the estimated and measured output (if they are only shifted) should be higher than the variance of their difference, under the assumption that the model used for estimating the output is not biased and the process output changes (there is an excitation present). The variances are calculated recursively with (23). When the variance of the difference between the estimated and measured outputs falls under the variance of both the estimated and the measured outputs the raised alarm is turned OFF. The algorithm starts to check this condition after the alarm is present for some time (in our case 300 samples). The ending of the algorithm is defined in the same manner as with the correlation procedure.

TABLE VI
SETTINGS FOR THE LEARNING METHOD AND FUZZY MODEL

Parameter	air flow	concentration O_2	concentration NH_4
N_{wait}	100	100	30
Adaptation mechanism and model			
η / η_m	2 / 1	2 / 0.25	2 / 2
γ_v / γ_m	1	0.9996	0.9999995
λ_r	1	0.9998	0.9999995
mem. deg. cut	0.3	0.3	0.3
$s_i(0)$	1	1	50
α	1	1	1
Adding mechanism			
k_n	2	1.6	2
N_c	5	5	4
ϵ_β	0.1	0.1	0.1
Removing mechanism			
k_{delay}	20	20	20
$N_{s_{trh}}$	10	10	10
ε	0.01	0.01	0.01
Splitting mechanism			
e_{max}	0.5	0.5	0.5
e_{min}	0.05	0.05	0.05
T	1000	1000	1000
$N_{sl_{trh}}$	10	10	15
Unsupervised merging mechanism			
$N_{sl_{trh}}$	10	10	10
k_{dmerge}	0.1	0.1	0.1
$\epsilon_{\beta m}$	0.8	0.8	0.8
Supervised merging mechanism			
C_{trh}	0.9	0.9	0.9
α_{trh}	2	2	2
$k_{dmerges}$	0.05	0.05	0.05
e_{merge}	0.05	0.02	0.05

IV. RESULTS AND DISCUSSION

The presented system was tested on real data gathered from a local WWT plant. The presented data were then centered and normalized. To estimate the performance of the system during a sensor's malfunction, a failure was simulated on a known part of the data. Note that the duration of the simulated fault was exaggerated in order to test the system. The simulated faults lasted for about 7000 samples (around 39 h). Usually, the faults last from about a few minutes up to 6 h. The settings of the evolving method were obtained based on trial and error. They are given in Table VI. For clustering with the air-flow model, the distance under (1) was used. For the other two models, the Mahanalobis distance was used. The membership degrees for the air flow and O_2 were radial [see (9)] and for the NH_4 model they were ellipsoidal [48].

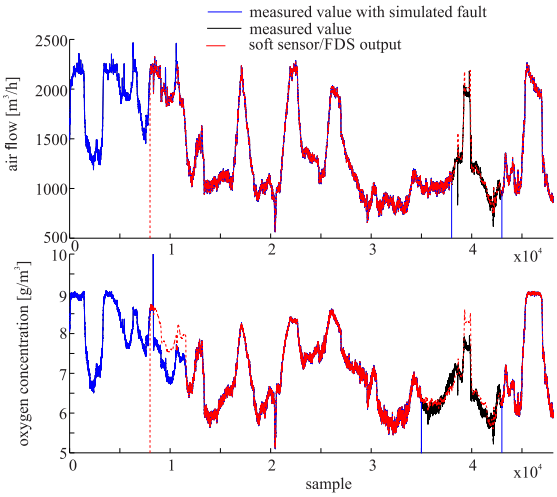

Fig. 7. Air-flow and oxygen-concentration fault detection.

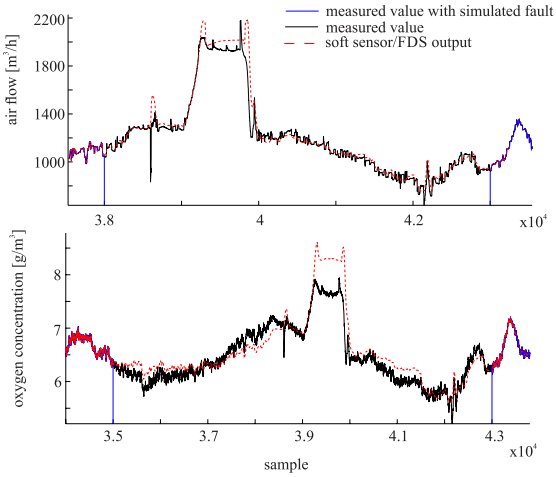

Fig. 8. Closeup of simulated air-flow and oxygen-concentration signal fault.

A. Oxygen and Air-Flow Fault Detection

For the air-flow signal, the fault was simulated between samples 38 000 and 43 000. For the oxygen-concentration signal, the fault was simulated between the samples 35 000 and 43 000. Note that with the oxygen concentration, two faults occur simultaneously. The first is on the input to the oxygen-concentration FDS (on an air-flow signal) and the second is on the oxygen-concentration signal itself. The whole experiment is shown in Fig. 7. The first 8000 data points were used for the initial learning of the fuzzy model. The learning was performed using the eFuMo method. The alarm signal and the number of fuzzy model clusters are shown in Fig. 9, while Fig. 8 shows a closeup of the simulated fault for both the air-flow and oxygen signals. Besides the simulated fault, the system also detected some faults that were not added to the signals. These faults were caused by sudden spikes in the monitored signals, and therefore, the detection of the fault seems justified. When identifying the valve opening and the air-flow relation, it was noticed that when the valve is fully opened very rapidly, the air flow first increases then decreases, and then slowly increases again. This phenomenon introduces an additional dynamic into the process, which was

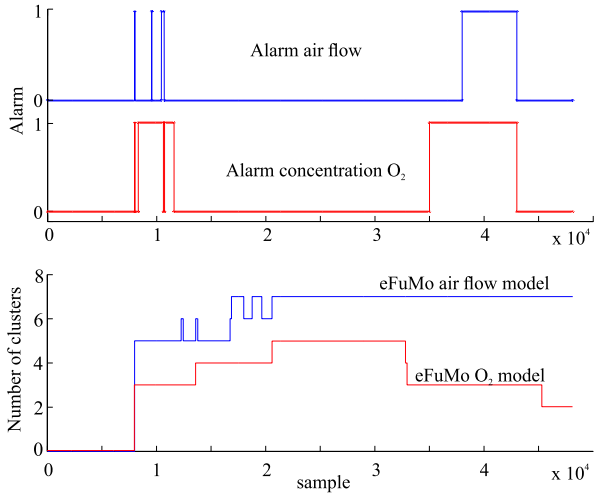

Fig. 9. Alarm signal and number of clusters over the experiment.

TABLE VII
ESTIMATION ERROR DURING THE SIMULATED FAULT

Estimation error	air flow	concentration O_2	concentration NH_4
NDEI	0.223	0.488	0.411
min. abs.	0.005 [L/s]	2.83e-5 [g/m^3]	3.69e-5 [g/m^3]
max. abs.	531.83 [L/s]	1.347 [g/m^3]	5.074 [g/m^3]
avg. abs.	39.97 [L/s]	0.189 [g/m^3]	1.367 [g/m^3]
signal range	1562 [L/s]	2.72 [g/m^3]	18.395 [g/m^3]
min. rel.	2.99e-6	1.04e-5	2e-6
max. rel.	0.341	0.495	0.276
avg. rel.	0.026	0.0695	0.0744
faulty samples	38–43 [$\times 10^3$]	35–43 [$\times 10^3$]	15–18.5 [$\times 10^3$]

hard to model with the chosen regressors and method, probably also because the occurrences of this phenomenon were not very frequent. This can be seen in Fig. 7 at around 0.5×10^4, 1×10^4, and in Fig. 8 at sample 3.9×10^4. The model is slightly less accurate when this happens. If the phenomenon occurred more frequently, the performance of the model in this part of the input–output space would improve.

Even though the estimated signal is not entirely covering the measured signal, we believe that the estimation accuracy is still good enough for the control purposes of the WWTP. The error between the measured and estimated signal during the fault is given in Table VII. This table also includes the NIDE index; the minimum, maximum, and mean absolute error; the signal range for the faulty samples; the minimum, maximum, and mean relative error; and the samples where the fault was simulated are given. It is clear that even if both signals (air-flow and oxygen-concentration) fail simultaneously, the system is able to produce a reasonably good estimation of the sensor signal. The crucial signal on which the oxygen and air-flow FDS depend is the valve opening. If the fault was to occur on the valve-opening signal, the FDS would fail. The system would raise the alarm for the air-flow signal. In order to be able to detect faults on the valve-opening signal an additional logic should be implemented. The logic should check for the alarm of the oxygen concentration based on the measured air flow and the estimated air flow. If the alarm is raised for the second case and not for the first case, it

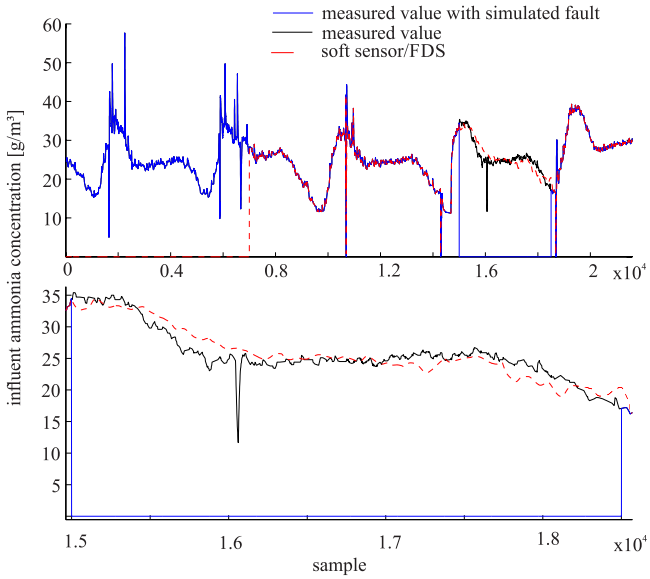

Fig. 10. Effluent ammonia fault detection.

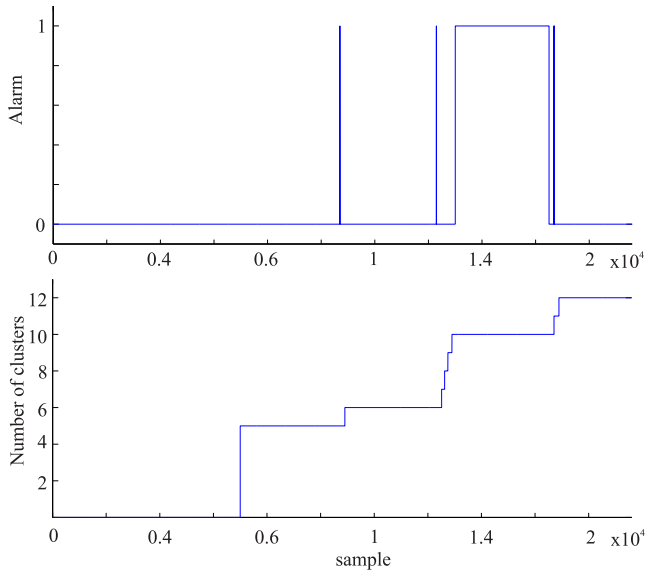

Fig. 11. Alarm and number of clusters over the ammonia experiment.

is highly likely that the fault was on the valve-opening signal. However, the fault of the valve-opening signal is unlikely to happen, since this is the controller output, and therefore, this logic was omitted in our FDS.

B. Influent Ammonia Concentration

For the ammonia concentration, the fault was simulated between samples 15 000 and 18 500. The first 7000 samples were chosen for the initial model learning using the eFuMo method. The results are shown in Fig. 10. The upper graph displays the whole experiment, and the lower graph displays an interval with the simulated fault. The alarm signal and the number of clusters are shown in Fig. 11. The estimation error is given in the last column of Table VII. The peak at around sample 16 000 was not considered in the error calculation.

As with the previous experiment, it can be seen that the accuracy of the estimated output is fairly good. The FDS detected some faults that were not simulated. These faults were caused by spikes in the output signal. Judging from the signals, these spikes could be caused by the faulty measurement. Since the influent ammonia signal is normally used for the gain scheduling of the main controller, the accuracy of the estimations is not so crucial. In contrast with the air-flow and oxygen-concentration signals, where the estimation of both signals depends only on the valve-opening signal, which is very unlikely to be faulty, the monitoring of the influent ammonia concentration depends on four signals that may also be faulty. Therefore, it can happen that the alarm is raised on the influent ammonia signal, even though it is not faulty. This will happen if a fault occurs on an input signal, e.g., the influent TN signal. Out of the four input signals, this one is the most likely to be faulty. In future works, an additional fault-isolation algorithm should be designed that is able to distinguish between a real fault on an output signal and a fault caused by an input signal. This will probably include some statistical modeling of the signals (e.g., a PCA or a time-series approximation of the influent ammonia).

C. False Alarms Due to Manual Calibration Detection

As can be seen on the upper graph in Fig. 6, the manual tuning creates an offset of the measured signal, resulting in the detection of a fault. At around sample 8400, a real fault occurs, which then quickly vanishes. Later on, the measured signal is shifted. The FDS detects the alarm. Because the signal is shifted after the fault, the alarm is still present. The alarm is finally turned OFF at sample 11 500, when the measured signal comes into the allowed difference zone and stays there long enough for the fuzzy model to adapt itself to the signal shift. On the lower graph in Fig. 6, the false-alarm detection was implemented. The plotted response shows the correlation procedure. The response of the variance procedure is practically the same and was, therefore, not plotted. It can be seen that the signal shift is successfully detected, and the alarm is turned OFF more quickly than without the implemented false-alarm detection algorithm.

On the upper graph in Fig. 12, the course of the correlation factor and, on the lower graph, the variances are shown. The correlation factor first reaches the threshold value at sample 8840. The alarm raised based on the allowed maximum difference of the measured, and estimated output is overridden from sample 9361 on. For this sample, the conditions are met for overriding the original alarm. The last raised alarm based on the output differences is raised at sample 9670. Therefore, the correlation procedure is switched OFF at sample 9770.

With the variance procedure, the variance of the difference (between the estimated and measured signal) falls under the measured signal's variance very quickly. This is partly because the initial fault of the measured signal is included in the variance calculation. The variance of the difference falls under the variance of the estimated signal at sample 9475. With this, the conditions for overriding the original alarm are met. The last alarm based on the output differences is raised at sample 9740.

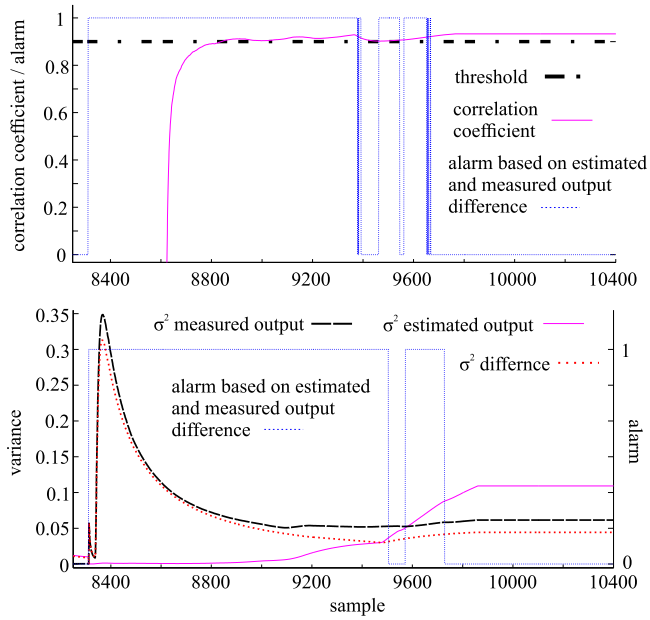

Fig. 12. Course of the correlation coefficient and variances.

Therefore, the variance procedure is switched OFF at sample 9840.

Both procedures successfully detected the signal offset caused by the manual calibration. However, we believe that the variance procedure is easier to implement and to use since it has fewer design and tuning parameters than the correlation procedure.

V. Conclusion

This paper has presented the possible construction of an FDS for a WWTP. The obtained results on measured data gathered from a real plant show that the evolving methods could be successfully used in such systems. We believe that the control of the WWTP could be greatly improved with the proposed FDS. The obtained estimation results during the sensor failure are reasonably accurate. With such a system, we could achieve smooth functioning of the control, even if some of the sensors drop out. The proposed approach can also be used on other types of processes.

It was shown that a complex and time-varying relations between variables, such as found in the WWTP, can be modeled with evolving methods and provide reasonable accuracy. However, there are still some issues that need our attention. One of the objectives of our future work will certainly be the construction of an additional algorithm for checking the faults on the input signals (e.g., for estimating the influent ammonia) as they might also be faulty. This algorithm should also work in an online manner. The disadvantage of the presented approach is that it is not capable of detecting the faults if the measured signals are within the calculated bounds. Therefore, it should be combined with approaches that track the statistical properties of the signal. We also believe that some additional research efforts will have to be made before connecting the proposed system in the real closed loop.

Regarding the evolving method, there is the problem of tuning. The evolving methods usually have a number of parameters that need to be tuned properly in order to achieve good models. This procedure can be time consuming, at least for the nonexpert.

References

[1] F. Baggiani and S. Marsili-Libelli, "Real-time fault detection and isolation in biological wastewater treatment plants," *Water Sci. Technol.*, vol. 60, no. 11, pp. 2949–2961, 2009.

[2] R. Isermann, *Fault-Diagnosis Systems: An Introduction From Fault Detection to Fault Tolerance*. Heidelberg, Germany: Springer-Verlag, 2006.

[3] D. Vrečko, N. Hvala, and M. Stražar, "The application of model predictive control of ammonia nitrogen in an activated sludge process." *Water Sci. Technol.*, vol. 64, no. 5, pp. 1115–1121, 2011.

[4] S. Graziani, N. Pitrone, M. Xibilia, and N. Barbalace, "Robust BOD_5 soft sensor design using local learning," in *Proc. Int. Conf. Comput. Appl. Syst. Modeling*, Taiyuan, China, Oct. 22–24, 2010, vol. 3, pp. 584–589.

[5] G. Bavdazž, N. Hvala, J. Kocijan, and D. Juričič, "Nelinearni algoritem za estimacijo stanj in identifikacijo parametrov šaržnega biološkega procesa," *Elektrotehniški vestnik*, vol. 68, pp. 57–63, 2001.

[6] C. K. Yoo and I.-B. Lee, "Soft sensor and adaptive model-based dissolved oxygen control for biological wastewater treatment processes," *Environ. Eng. Sci.*, vol. 21, no. 3, pp. 331–340, 2004.

[7] H. Ye, F. Luo, and Y. Xu, "Dissolved oxygen generic model control of wastewater treatment process based on immune optimization least squares support vector machine," in *Proc. 8th World Congr. Intell. Control Autom.*, 2010, pp. 5082–5086.

[8] U. Holmberg, G. Olsson, and B. Andersson, "Simultaneous DO control and respiration estimation," *Water Sci. Technol.*, vol. 21, pp. 1185–1195, 1989.

[9] C. F. Lindberg, "Control and estimation strategies applied to the activated sludge process," Ph.D. dissertation, Dept. Mater. Sci. Syst. Control Group, Uppsala Univ., Stockholm, Sweden, 1997.

[10] Y. Kim, H. Bae, K. Poo, J. Kim, T. Moon, S. Kim, and C. Kim, "Soft sensor using PNN model and rule base for wastewater treatment plant," in *Advances in Neural Networks* (Lecture Notes in Computer Science Series). Berlin, Germany: Springer, 2006, vol. 3973, pp. 1261–1269.

[11] D. Cecil and M. Kozlowska, "Software sensors are a real alternative to true sensors," *Environ. Modelling Softw.*, vol. 25, pp. 622–625, 2010.

[12] D.-J. Choi and H. Park, "A hybrid artificial neural network as a software sensor for optimal control of a wastewater treatment process," *Water Res.*, vol. 35, no. 16, pp. 3959–3967, 2001.

[13] M. W. Lee, S. H. Hong, H. Choi, J.-H. Kim, D. S. Lee, and J. M. Park, "Real-time remote monitoring of small-scaled biological wastewater treatment plants by a multivariate statistical process control and neural network-based software sensors," *Process Biochem.*, vol. 43, no. 10, pp. 1107–1113, 2008.

[14] V. Alcaraz-Gonzales, J. Harmand, A. Rapaport, and J. P. Steyer, "Software sensors for highly uncertain WWTPs: A new approach based on interval observers," *Water Res.*, vol. 36, pp. 2515–2524, 2002.

[15] E. P. Tao, W. H. Shen, T. L. Liu, and X. Q. Chen, "Fault diagnosis based on PCA for sensors of laboratorial wastewater treatment process," *Chemometrics Intell. Lab. Syst.*, vol. 128, pp. 49–55, 2013.

[16] M. J. Fuente, D. Garcia-Alvarez, G. I. Sainz-Palmero, and P. Vega, "Fault detection in a wastewater treatment plant based on neural networks and PCA," in *Proc. 20th Mediterranean Conf. Control Autom.*, Jul. 2012, pp. 758–763.

[17] H. Haimi, M. Mulas, F. Corona, and R. Vahala, "Data-derived soft-sensors for biological wastewater treatment plants: An overview," *Environ. Modelling Softw.*, vol. 47, pp. 88–107, 2013.

[18] S. X. Ding, *Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools* (Advances in Industrial Control Series), 2nd ed. Berlin, Germany: Springer, 2013.

[19] J. Korbicz, J. M. Koscielny, Z. Kowalczuk, and W. Cholewa, *Fault Diagnosis—Models, Artificial Intelligence and Applications*. Berlin, Germany: Springer Verlag, 2004.

[20] J. Shing and R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.

[21] M. F. Azeem, H. Hanmandlu, and N. Ahmad, "Structure identification of generalized adaptive neuro-fuzzy inference systems," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 5, pp. 666–681, Oct. 2003.

[22] D. Dovžan and I. Škrjanc, "Recursive fuzzy c-means clustering for recursive fuzzy identification of time-varying processes," *ISA Trans.*, vol. 50, no. 2, pp. 159–169, 2011.

[23] D. Dovžan and I. Škrjanc, "Recursive clustering based on a Gustafson–Kessel algorithm," *Evolving Syst.*, vol. 2, no. 1, pp. 15–24, 2011.
[24] J. Platt, "A resource allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.
[25] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, Feb. 1998.
[26] F. J. Lin, C. H. Lin, and P. H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 5, pp. 751–759, Oct. 2001.
[27] S. G. Tzafestas and K. C. Zikidis, "NeuroFAST: On-line neuro-fuzzy ART-based structure and parameter learning TSK model," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 5, pp. 797–802, Oct. 2001.
[28] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.
[29] P. P. Angelov and D. P. Filev, "An approach to on-line identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 484–497, Feb. 2004.
[30] E. Lughofer and E. P. Klement, "FLEXFIS: A variant for incremental learning of Takagi-Sugeno fuzzy systems," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Reno, NV, USA, May 22–25, 2005, pp. 915–920.
[31] M. Pratama, S. G. Anavatti, P. Angelov, and E. Lughofer, "PANFIS: A novel incremental learning machine," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 1, pp. 55–68, Jan. 2014.
[32] H. J. Rong, N. Sundararajan, G. B. Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets Syst.*, vol. 157, no. 9, pp. 1260–1275, 2006.
[33] G. Leng, G. Prasad, and T. M. McGinnity, "An on-line algorithm for creating self-organizing fuzzy neural networks," *Neural Netw.*, vol. 17, pp. 1477–1493, 2004.
[34] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A recursive growing and pruning RBF (GAP-RBF) algorithm for function approximations," in *Proc. 4th Int. Conf. Control Autom.*, Montreal, QC, Canada, Jun. 2003, pp. 10–12.
[35] N. K. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised on-line knowledge-based learning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 6, pp. 902–918, Dec. 2001.
[36] N. Kasabov, "Evolving fuzzy neural networks – Algorithms, applications and biological motivation," in *Proc. Methodologies Conceptation, Des. Appl. Soft Comput.*, 1998, pp. 271–274.
[37] S. Wu and M. J. Er, "Dynamic fuzzy neural networks: A novel approach to function approximation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 2, pp. 358–364, Apr. 2000.
[38] S. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 578–594, Aug. 2001.
[39] H. Soleimani-B, C. Lucas, and B. N. Araabi, "Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling," *Evolving Syst.*, vol. 1, no. 1, pp. 59–71, 2010.
[40] P. Angelov, "Evolving Takagi-Sugeno fuzzy systems from streaming data (eTS+)," in *Evolving Intelligent Systems: Methodology and Applications*. Hoboken, NJ, USA: Wiley, 2010, pp. 21–50.
[41] E. Lughofer, "Flexible evolving fuzzy inference systems from data streams (FLEXFIS++)," in *Learning in Non-Stationary Environments: Methods and Applications*. New York, NY, USA: Springer, pp. 205–245.
[42] A. Kalhor, B. Araabi, and C. Lucas, "An online predictor model as adaptive habitually linear and transiently nonlinear model," *Evolving Syst.*, vol. 1, no. 1, pp. 29–41, 2010.
[43] J. Rubio, "SOFMLS: Online self-organizing fuzzy modified least-squares network," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 6, pp. 1296–1309, Dec. 2009.
[44] E. Lughofer, "On-line assurance of interpretability criteria in evolving fuzzy systems - achievements, new concepts and open issues," *Inf. Sci.*, vol. 251, pp. 22–46, 2013.
[45] E. Lughofer, *Evolving Fuzzy Systems—Methodologies, Advanced Concepts and Applications* (Studies in Fuzziness and Soft Computing series), vol. 266. New York, NY, USA: Springer, 2011.
[46] I. Rodrigez-Roda, M. Sanchez-Marre, J. Comas, J. Baeza, J. Colprim, J. Lafuente, U. Cortes, and M. Poch, "A hybrid supervisory system to support WWTP operation: Implementation and validation," *Water Sci. Technol.*, vol. 45, nos. 4/5, pp. 289–297, 2002.
[47] C. L. Hwang and L. J. Chang, "Fuzzy neural-based control for nonlinear time-varying delay systems," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 37, no. 6, pp. 1471–1485, Dec. 2007.
[48] D. Dovžan, V. Logar, and I. Škrjanc, "Solving the sales prediction problem with fuzzy evolving methods," in *Proc. IEEE Congr. Evolutionary Comput.*, Brisbane, Australia, Jun. 2012, pp. 1–8.
[49] O. Nelles, "Local linear model tree for on-line identification of time invariant nonlinear dynamic systems," in *Proc. Int. Conf. Artif. Neural Netw.*, Bochum, Germany, 1996, pp. 115–120.
[50] J. de Jesús Rubio, "Stability analysis for an online evolving neuro-fuzzy recurrent network," in *Evolving Intelligent Systems Methodology and Applications*. Hoboken, NJ, USA: Wiley, 2010, pp. 173–199.
[51] L. Hartert, M. S. Mouchaweh, and P. Billaudel, "A semi-supervised dynamic version of fuzzy *k*-nearest neighbours to monitor evolving systems," *Evolving Syst.*, vol. 1, pp. 3–15, 2010.
[52] I. Škrjanc, B. Hartmann, O. Banfer, O. Nelles, A. Sodja, and L. Teslič, "Supervised hierarchical clustering in fuzzy model identification," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 6, pp. 1163–1176, Dec. 2011.
[53] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer, 1981.
[54] D. Dovžan and I. Škrjanc, "Recursive fuzzy model identification," in *Proc. Advances Comput. Sci. Eng.*, Sharm El Sheikh, Egypt, Mar. 15–17, 2010, pp. 1–6.
[55] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2057–2068, 2011.
[56] K. J. strom and B. Wittenmark, *Adaptive Control*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1995.
[57] S. W. Tung, C. Quek, and C. Guan, "SaFIN: A self-adaptive fuzzy inference network," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1928–1940, Dec. 2011.
[58] R. Rosa, F. Gomide, D. Dovžan, and I. Škrjanc, "Evolving neural network with extreme learning for system modeling," in *Proc. IEEE Conf. Evolving Adaptive Intell. Syst.*, Linz, Austria, Jun. 2–4w, 2014, pp. 11–34.
[59] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural networks," *Neural Comput.*, vol. 5, no. 6, pp. 954–975, 1993.
[60] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function (RBF) neural networks," *Neural Comput.*, vol. 9, pp. 461–478, 1997.
[61] P. Angelov and D. Filev, "Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models," in *Proc.. IEEE Int. Conf. Fuzzy Syst.*, Reno, NV, USA, May 22–25, 2005, pp. 1068–1073.
[62] P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," *Int. J. General Syst.*, vol. 41, no. 2, pp. 163–185, 2011.
[63] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2004, pp. 985–990.
[64] R. Rosa, R. Ballini, and F. Gomide, "Evolving hybrid neural fuzzy network for system modeling and time series forecasting," in *Proc. IEEE Int. Conf. Mach. Learning Appl.*, Miami, FL, USA, Dec. 4–7, 2013, pp. 378–383.
[65] P. Angelov and Z. Xiaowei, "Evolving fuzzy systems from data streams in real-time," in *Proc. Int. Symp. Evolving Fuzzy Syst.*, 2006, pp. 29–35.

Dejan Dovžan received the B.Sc. degree from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2008. In his bachelor thesis, he covered self-tuning algorithms for the PFC controller and proposed solutions for better disturbance rejection of the PFC. In 2013, he received the Ph.D. degree also from the University of Ljubljana.

In his Ph.D. degree, he was covering the aspects of online and evolving methods for control and modeling purposes. He is currently an Assistant Professor with the Laboratory for Autonomous and Mobile Systems, University of Ljubljana.. His main research interests include fuzzy model learning methods, recursive and evolving fuzzy model identification, modeling from data, and model-predictive control.

Dr. Dovžan received the Prešeren Award for bachelor thesis. For the Ph.D. thesis, he received the Vodovnik award for exceptional Ph.D.

Vito Logar received the B.Sc. and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2004 and 2009, respectively.

He is currently an Assistant Professor with the University of Ljubljana. His main research interests include modeling of the industrial systems and optimization of the electric arc furnace processes.

Igor Škrjanc received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical and Computer Engineering, University of Ljubljana, Ljubljana, Slovenia, in 1988, 1991, and 1996, respectively.

He is currently a Professor of automatic control with the Faculty of Electrical Engineering, University of Ljubljana, and the Head of the research program Modeling, Simulation and Control. His main research interests include intelligent, predictive control systems, and autonomous mobile systems.

Dr. Škrjanc received the Highest Research Award from the Faculty of Electrical Engineering, University of Ljubljana, in 2007, and the Highest Award of the Republic of Slovenia for Scientific and Research Achievements in 2008. He also received the Zois Award for outstanding research results in the field of intelligent control. He also received the Humboldt Research Fellowship for Experienced Researchers for the period between 2009 and 2011.