Evolving Fuzzy Model (eFuMo) Method for on-line Fuzzy Model Learning with Application to Monitoring System

Dejan Dovžan*, Vito Logar, Igor Škrjanc

Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia;
*dejan.dovzan@fe.uni-lj.si

SNE Simulation Notes Europe SNE 26(4), 2016, 205-220
DOI: 10.11128/sne.26.tn10352
Received: November 15, 2016
Accepted: December 5, 2016 (Special Issue Review)

Abstract. With evermore complex system the monitoring and fault detection is becoming a crucial part of control systems. They allow fast and effective fault diagnosis and can decrease the cost of system maintenance. Modelling of processes plays a crucial part when designing a monitoring system. In this paper an on-line approach for modelling of fuzzy model is presented (Evolving Fuzzy Model - eFuMo). As demonstrated in the paper, the method can be used in the design of model based fault detection system.

Introduction

Increasing demands of productivity and reliability call for extending the ability of a common SCADA systems with the monitoring and fault detection systems. There are several approaches for designing the fault detection system. In our paper the monitoring system is based on a process model. The model is based on a evolving fuzzy model method (eFuMo). The presented method is able to build Takagi-Sugeno fuzzy model (TS) from scratch, starting with one cluster and a local model. The TS fuzzy models are a powerful practical engineering tool for modelling and control of complex systems. They expand and generalize the well-known concept of gain scheduling. They utilize the idea of linearization in a fuzzily defined region of the state space. Due to the fuzzy regions (clusters), the nonlinear system is decomposed into a multi-model structure consisting of linear local models [1].

This enables the T-S fuzzy model to approximate virtually any nonlinear system within a required accuracy, provided that enough regions are given [2].

The eFuMo method is an on-line learning method that is also able to adapt models during the functioning of the system. Depending on the learning abilities, the on-line fuzzy-identification methods can be divided into: *Adaptive methods* (e.g., ANFIS [3], GANFIS [4], rFCM [5], rGK [6]), 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 [7], SONFIN [8], SCFNN [9], NeuroFAST [10], DENFIS [11], eTS [12], FLEXFIS [13], PANFIS [14]), where only adding mechanisms are implemented; *Evolving methods* (e.g., SAFIS [15], SOFNN [16], GAP-RBF [17], EFuNN [18, 19], D-FNN [20], GD-FNN [21], ENFM [22], eTS+ [23], ENFM [22], FLEXFIS++ [24], AHLTNM [25], SOFMLS [26]) which, besides an adding mechanism, implement removing and some of them also merging and splitting mechanisms. More on evolving methods can be found in [27] and [28], where concepts and open issues regarding these methods are presented.

The paper is organized in the following order. First, the eFuMo learning method is described, next the monitoring problem is given followed by results and conclusions.

1 eFuMo Structure

The eFuMo method has two types of mechanisms for identifying the fuzzy model: the adaptation algorithm and the evolving mechanisms. The first is responsible for parameter adaptation, such as cluster centers and lo-

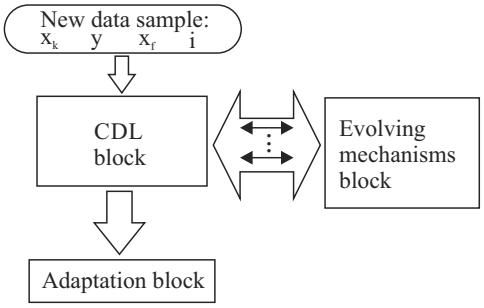


Figure 1: The eFuMo top scheme.

cal models' parameters; the second is responsible for structure update: adding, removing, merging and splitting of clusters. A central decision logic (CDL) decides which type of mechanism will be used at current sample. The block scheme is presented on Figure 1. In the following subsection, the adaptation and evolving mechanisms will be presented and at the end the CDL will be described.

1.1 Adaptation mechanisms

In order to build the TS fuzzy model clusters and local linear models must be identified. Adaptation mechanisms are responsible for identifying clusters' and local models' parameters and for their adaptation. To partition input-output data space recursive clustering algorithm is used and for identifying the local models' parameters the fuzzy recursive least squares is used.

Space partitioning. For data space partitioning, the cluster centers and fuzzy covariance matrix must be calculated. The centers are adapted with the following equation:

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

$$\Delta\mathbf{v}_i(k) = \frac{\mu_i(k)^\eta \left(\mathbf{x}_f(k) - \mathbf{v}_i(k)\right)}{s_i(k)} \tag{2}$$

where η is fuzziness factor, $\mathbf{v}_i$ is the center position vector $\mathbf{x}_f$ is clustering vector, μ_i is membership degree of the current clustering vector to the i-th cluster also called the firing degree of the i-th cluster and $s_i(k+1)$ is the sum of past membership degrees / firing levels of the i-th cluster:

$$s_i(k) = \lambda_c s_i(k-1) + \mu_i(k)^\eta. \tag{3}$$

where λ_c was introduced as a forgetting factor to enable the adaptation of centers. The membership degrees μ_i can be calculated as in equation 4 (c is the number of existing clusters), either based on rFCM [5] (equation 5), rGK [6] (equation 6) or Mahalanobis distance (equation 7).

$$\mu_i(k) = \begin{cases} \dfrac{1}{\sum_{j=1}^{c}\left(\frac{d_i(k)}{d_j(k)}\right)^{\frac{2}{\eta-1}}} & \text{if } \mathbf{x}_f(k) \neq \mathbf{v}_i; i = 1,...,c \\ 1 & \text{if } \mathbf{x}_f(k) = \mathbf{v}_i \\ 0 & \text{if } \mathbf{x}_f(k) = \mathbf{v}_j; i \neq j \end{cases} \tag{4}$$

$$d_i(k) = \left(\left(\mathbf{x}_f(k) - \mathbf{v}_i(k)\right)^T \left(\mathbf{x}_f(k) - \mathbf{v}_i(k)\right)\right)^{0.5} \tag{5}$$

$$d_i(k) = \left(\left(\mathbf{x}_f(k) - \mathbf{v}_i(k)\right)^T \det(\mathbf{F}_i)^{\frac{1}{p}} \mathbf{F}_i^{-1} \left(\mathbf{x}_f(k) - \mathbf{v}_i(k)\right)\right)^{0.5} \tag{6}$$

$$d_i(k) = \left(\left(\mathbf{x}_f(k) - \mathbf{v}_i(k)\right)^T \mathbf{F}_i^{-1} \left(\mathbf{x}_f(k) - \mathbf{v}_i(k)\right)\right)^{0.5} \tag{7}$$

To get the area of cluster influence the fuzzy covariance matrix $\mathbf{F}_i$ is calculated. The recursive equation for $\mathbf{F}_i$ is the following:

$$\mathbf{F}_i(k+1) = \gamma_c \frac{s_i(k-1)}{s_i(k)} \mathbf{F}_i(k) + \frac{\mu_i(k)^\eta}{s_i(k)} \mathbf{D}_{F_i}(k)$$
$$\mathbf{D}_{F_i}(k) = \left(\mathbf{x}(k) - \mathbf{v}_i(k)\right)\left(\mathbf{x}(k) - \mathbf{v}_i(k)\right)^T. \tag{8}$$

where γ_c is the forgetting factor. To be able to calculate the Gustafson-Kessel clustering distance (equation 6) the inverse and determinant of fuzzy covariance matrix must be calculated. The recursive equation for the inverse matrix is obtained by using the Woodbury matrix identity lemma. The equation is following:

$$\left[\mathbf{F}_i(k+1)\right]^{-1} = \frac{1}{\gamma_c} \frac{s_i(k)}{s_i(k-1)} \left[\left[\mathbf{F}_i(k)\right]^{-1} - \frac{\mathbf{B}}{C}\right] \tag{9}$$

$$\mathbf{B} = \left[\mathbf{F}_i(k)\right]^{-1} \mathbf{D}_{F_i} \left[\mathbf{F}_i(k)\right]^{-1} \tag{10}$$

$$C = \gamma_c \frac{s_i(k-1)}{\mu_i(k)^\eta} + d_{F_i}^T \left[\mathbf{F}_i(k)\right]^{-1} d_{F_i} \tag{11}$$

$$d_{F_i} = \mathbf{x}_f - \mathbf{v}_i(k). \tag{12}$$

The determinant is obtained using determinant lemma (equation 13):

$$\det(\mathbf{A}+\mathbf{u}\mathbf{v}^{\mathrm{T}}) = (1+\mathbf{v}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{u})\det(\mathbf{A}). \tag{13}$$

The recursive equation for determinant calculation is:

$$\det(\mathbf{F}_i(k+1)) = \left(\gamma_c \frac{s_i(k-1)}{s_i(k)}\right)^p \det(\mathbf{F}_i(k))(1+A) \tag{14}$$

$$A = \frac{1}{\gamma_c}\frac{\mu_i(k)^\eta}{s_i(k-1)}\left(d_{F_i}^T[\mathbf{F}_i(k)]^{-1}d_{F_i}\right), \tag{15}$$

where p is the number of rows/colmuns of fuzzy covariance matrix. The detailed derivations of equations are given in [6]. The eFuMo method implements the method for stopping the cluster parameters adaptation if the clusters firing level is below a certain user defined threshold $\beta_{cut_{trh}}$. This prevents clusters, that are far from current clustering vector, to converge to that area. The clusters' firing levels 4 that are below the threshold are set to zero. The rest of the firing levels are then normalized.

Local models' parameters identification.

Each cluster has a linear local model, that is valid in that area. The output of the local model is calculated as:

$$y_{m_i}(k) = \theta_i^T[1 \;\; \mathbf{x}_k(k)^T]^T, \tag{16}$$

where $\mathbf{x}_k(k)$ is the regression vector and θ_i^T are the local model parameters. The regression vector is usually the input part of the clustering vector:

$$\mathbf{x}_f(k) = [\mathbf{x}_k(k)^T y(k)]^T, \tag{17}$$

where y is the process output. However unlike many existing on-line fuzzy identification methods the eFuMo method allows the clustering vector to be different than the regression vector.

The eFuMo has different fuzzy least squares based identification methods included ([12], [22], [5] and [29]). Usually best results are obtained using local fuzzy weighted least squares presented in [12]:

$$\begin{aligned}\mathbf{x}_e(k) &= [1 \;\; \mathbf{x}_k(k)^T]^T \\ \mathbf{P}_i(k+1) &= \frac{1}{\lambda_r}\left(\mathbf{P}_i(k) - \frac{\beta_i\mathbf{P}_i(k)\mathbf{x}_e(k)\mathbf{x}_e^T(k)\mathbf{P}_i(k)}{\lambda_r+\beta_i\mathbf{x}_e^T(k)\mathbf{P}_i(k)\mathbf{x}_e(k)}\right) \\ \theta_i(k+1) &= \theta_i(k)+\mathbf{P}_i(k)\beta_i\mathbf{x}_e(k)\left(y(k)-\mathbf{x}_e^T(k)\theta_i(k)\right)\end{aligned} \tag{18}$$

where i is the cluster index, θ is the vector of local model's parameters and β is the firing level of cluster and the y is the process output. The firing levels are calculated on the input space. Usually the methods use Gaussian functions equation 19 or equation 20:

$$\begin{aligned}\mu_i(k) &= e^{-\frac{(x_{f_k}-v_{i_k})^2}{2\eta_m\mathbf{F}_{i_{k,k}}}} \quad k=1,2,...p-1 \;\; i=1,2,...,c \\ \beta_i &= \prod_{k=1}^{p-1}\mu_i(k)\end{aligned} \tag{19}$$

$$\begin{aligned}\beta_i &= e^{\frac{-D_{ik}^2}{2\eta_m}} \quad i=1,2,...,c, \\ D_{ik}^2 &= \left(\mathbf{x}_{f_{in}}(k)-\mathbf{v}_{i_{in}}\right)^T\mathbf{F}_{i_{in}}^{-1}\left(\mathbf{x}_{f_{in}}(k)-\mathbf{v}_{i_{in}}\right).\end{aligned} \tag{20}$$

where η_m is the overlapping factor usually set to 1, $\mathbf{F}_{i_{k,k}}$ is diagonal element k of fuzzy covariance matrix, x_{f_k} and v_{i_k} are the k-th element of clustering vector and k-th element of i-th cluster center vector, respectively. The $\mathbf{F}_{i_{in}}$ is the input fuzzy covariance matrix, $\mathbf{x}_{f_{in}}$ is the clustering vector containing only the input variables and $\mathbf{v}_{i_{in}}$ is the cluster center in an input space. The obtained firing levels are then normalized:

$$\beta_i = \frac{\beta_i}{\sum_{k=1}^c \beta_k} \quad i=1,2,...,c. \tag{21}$$

One can also use the same equation for firing level calculation as with clustering algorithm. However, with Gussian functions the transitions between local models (clusters) are more smooth.

When building the simulation model, the model parameters can be identified more accurately using the instrumental version of least squares [30]. The instrumental variable adaptation algorithm for equation 22 can be written as:

$$\begin{aligned}\mathbf{x}_e(k) &= [1 \;\; \mathbf{x}_k(k)^T]^T \\ \mathbf{x}_{e_m}(k) &= [1 \;\; \mathbf{x}_{k_m}(k)^T]^T \\ \mathbf{P}_i(k+1) &= \frac{1}{\lambda_r}\left(\mathbf{P}_i(k) - \frac{\beta_i\mathbf{P}_i(k)\mathbf{x}_{e_m}(k)\mathbf{x}_e^T(k)\mathbf{P}_i(k)}{\lambda_r+\beta_i\mathbf{x}_e^T(k)\mathbf{P}_i(k)\mathbf{x}_{e_m}(k)}\right) \\ \theta_i(k+1) &= \theta_i(k)+\mathbf{P}_i(k)\beta_{i_m}\mathbf{x}_{e_m}(k)\left(y(k)-\mathbf{x}_e^T(k)\theta_i(k)\right)\end{aligned} \tag{22}$$

where $\mathbf{x}_{e_m}(k)$ is the regression vector where the delayed process outputs were replaced with model outputs and β_{i_m} is the membership degree of vector $\mathbf{x}_{f_m}(k)$, which is the clustering vector, where delayed process outputs

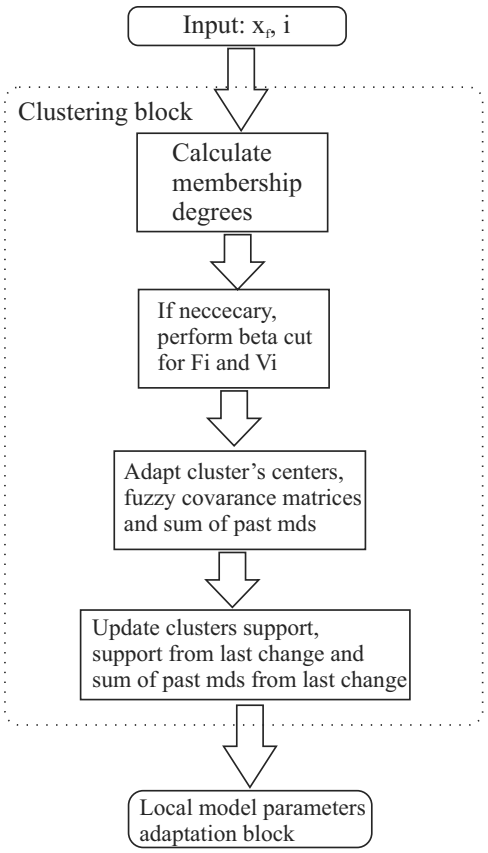


Figure 2: The clustering algorithm.

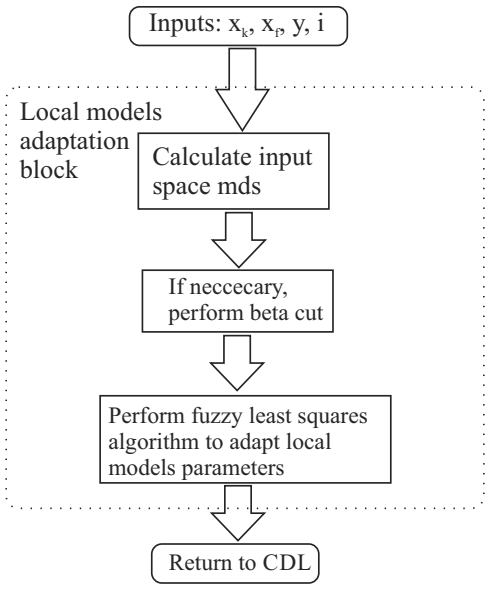


Figure 3: The parameter adaptation algorithm.

were replaced with model outputs:

$$\begin{aligned}\mathbf{x}_f(k) &= [u(k-n)\ \ldots u(k)\ \ y(k-r)\ \ \ldots\ \ y(k-1)]\\ \mathbf{x}_{f_m}(k) &= [u(k-n)\ \ldots u(k)\ \ y_m(k-r)\ \ \ldots\ \ y_m(k-1)]\end{aligned} \tag{23}$$

where y_m is the model output and y is the real output. In both cases the dead zone for adaptation can be considered [31].

The adaptation procedure can be represented by the diagrams as shown on figure 2 and figure 3. In figure 2 the clustering procedure is represented and in figure 3 the local model parameters identification algorithm is presented.

1.2 Evolving mechanisms

To upgrade the fuzzy model structure evolving mechanisms, such as adding and removing the clusters is implemented in the eFuMo method.

Adding mechanism. This 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 of adding new clusters based on model output error, distance of current sample to existing cluster and ε-completeness which is based on current samples membership degree to existing clusters.

In [32] (DFKNN) a cluster adding is based on Euclidian distance to the existing cluster centers and the change of variance that the new sample brings to the closest cluster. The distance and variance change must be greater than the predefined threshold. A new cluster is added if a certain number of sequential samples satisfy this condition. In [11] (DENFIS) adding is based on an Euclidian distance. If the distance of current sample to closest cluster is grater than two times the threshold a new cluster is added. In [20] (D-FNN) and [21] (GD-FNN) adding is based on model error and distance of new sample vector to closest cluster. If both are grater than a user defined threshold the cluster is added. The threshold is decreasing with time. In [17] (GAP-RBF) and [15] (SAFIS) a new cluster is added if the model error and distance of the current sample to existing clusters is over a threshold. They calculate the decrease in error if current sample would be taken for a new cluster. If the decrease is large enough new cluster is created. In [18, 19] (EFuNN) the adding is based on sensitivity calculated based on normalized fuzzy differ-

ence distances. The eTS method [12] adds a new cluster when the potential of current sample is higher than a potential of existing clusters and if it is distanced enough from the nearest cluster. In [33] (NFCN), [22] (ENFM), [8] (SONFIN), [9] (SCFNN), [16] (SOFNN) adding is based on ε-completeness principle. In [13] (FLEXFIS) the adding is based on distance and vicinity quotient.

In practice, the distance conditions work best. Therefore, the eFuMo implements two conditions for adding: the distance conditions and the consequent samples conditions. Both conditions must be satisfied in order for a new cluster to be added. The consequent samples condition is to prevent a new cluster being created based on outlier sample. This condition means that several consecutive samples must satisfy the distance condition before a new cluster is added. The condition is explained in [32].

The distance adding condition is based on a normalized distance. There is an option of choosing the component distances or Mahanalobis distance. The normalized component distances are calculated as:

$$d_{ij} = \frac{|x_{f_j}(k) - v_{i_j}|}{k_n \sqrt{f_{i_{jj}}}}, \quad j = 1,...,p \;\; i = 1,...,c \qquad (24)$$

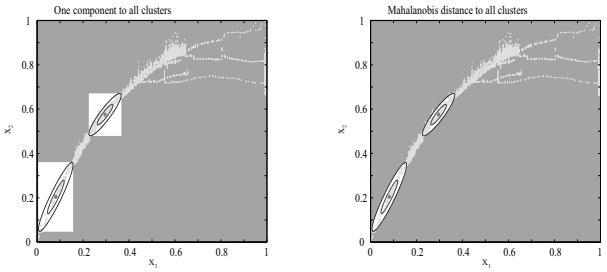
where $x_{f_j}(k)$ is the j-th element of clustering vector, v_{i_j} is the j-th component of i-th cluster center, p is the length of clustering vector, c is the number of clusters, $f_{i_{jj}}$ is the j-th diagonal element of i-th cluster's fuzzy covariance matrix and k_n is the user defined constant, usually set to 2. When using normalized Mahanalobis distance the normalization vector is formed from diagonal elements of fuzzy matrix:

$$\mathbf{s}_{i_{norm}} = [\sqrt{f_{i_{11}}} \;\; \sqrt{f_{i_{22}}} \;\; \cdots \;\; \sqrt{f_{i_{pp}}}]^T, \qquad (25)$$

The normalized distance is then calculated as:

$$d_{i_{norm}} = \frac{((\mathbf{x}_f(k) - \mathbf{v}_i)^T \mathbf{F}_i^{-1} (\mathbf{x}_f(k) - \mathbf{v}_i))^{0.5}}{k_n (\mathbf{s}_{i_{norm}}^T \mathbf{F}_i^{-1} \mathbf{s}_{i_{norm}})^{0.5}} \qquad (26)$$

With the first condition, a cluster can be added is any of the component distance equation 24 is larger than 1. The same component distance must be larger than 1 for all existing clusters. With the second condition a cluster can be added if the distances equation 26 to all clusters are larger than 1. Figures 4(a) show the possible adding space for the component distance conditions and figure 4(b) for the Mahanalobis distance condition. Both figures show the possible adding space (orange) for two



(a) Component adding distance.

(b) Mahalanobis adding distance.

Figure 4: Different adding distance conditions.

dimensional space. When a cluster is added, the parameters of the cluster must be initialized. The center of a new cluster is set at the position of current clustering vector. The fuzzy covariance matrix is initialized as diagonal matrix where the distances to closest cluster are considered. The diagonal elements are defined as:

$$f_{new_{jj}} = -\frac{d_{ij}^2}{2\eta_m \ln(\varepsilon_\beta)}, \qquad (27)$$

where ε_β is a user defined constant, normally set to 0.15. If the distance d_{ij} is smaller than standard deviation ($\sqrt{f_{i_{jj}}}$), then this diagonal element is equal to a diagonal element of the closest cluster's fuzzy covariance matrix ($f_{new_{jj}} = f_{i_{jj}}$).

The first cluster is added at the position of the first clustering vector. Its fuzzy covariance is initialized in the similar manner considering the input-output space boundary and expected number of clusters:

$$d_{max_j} = \max(x_j) - \min(x_j), \quad j = 1,...,p \qquad (28)$$

where d_{max_j} is an expected range of j-th element of clustering vector. The influence zone of the j-th component is then calculated as:

$$d_{influence_j} = \frac{d_{max_j}}{2\,c}, \quad j = 1,...,p \qquad (29)$$

where c is the expected number of clusters. The diagonal j-th element of fuzzy covariance matrix is then calculated as:

$$\sigma_j^2 = -\frac{d_{influence_j}^2}{2\eta_m \ln(\varepsilon_\beta)} \;\; j = 1,...,p \qquad (30)$$

The fuzzy covariance is built with σ_j^2 as:

$$\mathbf{F}_i = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_p^2 \end{bmatrix} \tag{31}$$

The parameters of new local model can be initialized using weighted mean:

$$\theta_{i+1_j} = \frac{\sum_{i=1}^{c} \omega_{i_j} \theta_{i_j}}{\sum_{i=1}^{c} \omega_{i_j}} \tag{32}$$

where i is the index of cluster and j is the parameter index. Weights ω_{i_j} can be equal to normalized firing levels of clusters, or equal to normalized firing levels of clusters combined with parameters variances:

$$\omega_{i_j} = \beta_i \frac{1}{\sigma^2_{P_{i_{jj}}}}, \tag{33}$$

where $\sigma^2_{P_{i_{jj}}}$ is the j-th diagonal element of least squares covariance matrix of i-th cluster.

Removing mechanism. It is meant to remove old clusters and clusters created based on outliers. In eFuMo method, this mechanism is not so important as the method incorporates the forgetting factors that ensure the adaptation of the structure to the new data. However, it may happen that a cluster is created in a partition of input-output space that doesn't have much samples and is not very important for the model accuracy. This mechanism ensures that these clusters are removed from the model structure. In literature, different ideas are presented. In [34] the cluster is removed if in a certain time the cluster doesn't receive any support sample. Cluster receives a support sample if it has greater firing level than other clusters. This might be a problem with industrial processes, where it might happen that the process is in one working point for a longer period of time. In this case, other clusters, that describe different working points, might be removed from the structure. In [20], [21], [17], [15] in [16] (D-FNN, GD-FNN, GAP-RBF, SAFIS in SOFNN) the removing is based on model error. In [20] (D-FNN) the *error reduction* ratio is introduced. The amount of error, that a certain cluster brings to the overall model error is calculated. If this is small, the cluster is considered as redundant and is therefore deleted. Similar concept is used in [21] (GD-FNN), where *sensitivity index* is introduced. In [15] (SAFIS) an equation is introduced to estimate the error change if a certain cluster is removed from the structure. If this change is small, the cluster is removed. In [16] (SOFNN) removing is based on *optimal brain surgeon approach* [35, 36]. In [37] (Neural gas) the clusters are removed based on their age. All clusters that are older than an user defined age are removed from the structure. In [38, 39] (exTS) the removing is based on cluster's support and cluster's age. The clusters are removed based on support-age ratio. Similar conditions are introduced in +eTS [23], where also the *utility* condition is added. This condition is based on the ratio of sum of firing levels and age of cluster. The threshold values are defined with standard deviation and mean values of the ratios. In general, this is not adequate, since there is usually small number of clusters; therefore, using standard deviation and mean value are not really representative. In +eTS also minimal existence condition is introduced. With this condition, a newly created cluster must gather a certain amount of support samples in a certain time period after creation. If the gathered support is lower than a predefined threshold, the cluster is removed from the structure. In [18, 19] (EFuNN) the removing is based on cluster's age and sum of cluster's firing levels. In [32] (DFKNN) removing is based on minimal support and time period. If the cluster has lower support than an user defined threshold the cluster is deleted. The cluster is also removed if in certain time period after cluster's creation, no support sample is assigned to it.

The proposed eFuMo method has two conditions for removing: A minimal existence condition and support-age ratio condition. The minimal existence condition is the same as in [23]. It simply removes clusters that in certain period after creation (k_{delay}) don't receive enough support samples (Ns_i). The time period (k_{delay}) and support threshold (Ns_{trh}) are user defined constant usually set to 20 and 10, respectively. The support-age ratio condition is based on clusters' supports Ns_i normalized with clusters' age (equation 35). Cluster with the ratio lower than a percent ε of mean ratio is deleted. Age a_i is defined as a number of samples from the cluster's creation k_i and current sample k:

$$a_i = k - k_i \tag{34}$$

$$S_{n_i} = \frac{Ns_i}{a_i}. \tag{35}$$

Both conditions for removing can be written as:

$$\begin{aligned} &\textbf{IF } S_{n_i} < \varepsilon \text{ mean}(S_n) \\ &\textbf{OR } \left(Ns_i < Ns_{trh} \textbf{ AND } k > k_i + k_{delay}\right) \\ &\textbf{THEN } \text{remove } i\text{-th cluster.} \end{aligned} \tag{36}$$

Splitting mechanism. It is in our case meant for fine tuning the evolving fuzzy model. It can add clusters in input-output space, where the output model error is higher than predefined threshold. The concept of splitting was used in the on-line incremental learning of Gaussian Mixture Models in [40], where the Chernoff bound is used and in [41], where fidelity measure is used. It is argued in [42] that these methods are slow and don't produce good results. Therefore they propose an integrating a joint incremental on-line split-and-merge scenario, that should overcome under- and over-clustered partitions. The splitting is based on a BIC value. The BIC is a combination of Gaussian density function function and cluster overlapping. The clusters that are split are found using trail and error procedure. In [10] (NeuroFAST) clusters are split based on mean squared error (MSE). The error is checked every P step. The cluster that has the highest MSE and is at least P-times activated is split.

The eFuMo's splitting mechanism is based on relative model error, that clusters gather over time. The error is updated every time the splitting mechanism is called and the current sample doesn't satisfy the distance adding condition. First the relative model error is calculated:

$$e(k) = \frac{|y_m(k) - y(k)|}{3.4\sigma_y}, \tag{37}$$

where y is the real output and y_m is the model output. The σ_y is calculated by CDL block and represents current standard deviation of the process output. The error is then divided among the existing clusters and added to the previous error:

$$e_{sum_i}(k) = e_{sum_i}(k-1) + \beta_i e(k), \tag{38}$$

where β_i is the firing level of i-th cluster. The splitting mechanism checks the cluster with the highest error. If its support from the last change in cluster number till now is higher than a threshold (usually set to 20) and its error normalized with N (number of samples used to calculate the error) is larger than a threshold value, the cluster is split. The error threshold is set by the user, specifying the maximal and minimal error threshold and the decay constant. The current threshold is calculated as:

$$e_{trh} = max(e_{max} exp(-N/T), e_{min}), \tag{39}$$

where e_{trh} is the current threshold, e_{max} is the maximal error threshold, e_{min} is the minimal error threshold, N and T are the number of samples that are used for error calculation and decay constant, respectively.

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

$$\begin{aligned} \mathbf{v}_{i1} &= \mathbf{v}_i + 0.5\mathbf{s}_{i_{norm}} \\ \mathbf{v}_{i1} &= \mathbf{v}_i - 0.5\mathbf{s}_{i_{norm}} \end{aligned} \tag{40}$$

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 [43]. The fuzzy covariance matrix, support and sum of past membership degrees are set to half of their original value for both clusters. The time of cluster creation is for both clusters initialized as the creation time of the original cluster.

Merging mechanism. There are two types of merging algorithms implemented in eFuMo: supervised and unsupervised. In literature different concept of merging techniques can be found. In [32] (DFKNN), the center positions are monitored. If the centers are converging to the same area the clusters are merged. The used similarity measure is based on samples membership degrees and is similar to the correlation between clusters firing levels. It is presented in detail in [44]. In [18] (EFuNN), the merging is done based on clusters' firing levels correlation. The method merges neighborhood clusters, where after merging the total radius does not exceed the predefined maximal radius. In [22] (ENFM),the clusters are merged if the membership degree of the first cluster to the second and vice versa is higher than a predefined threshold. In [16] (SOFNN) clusters are merged if the cluster centers of the two clusters are the same. The possibility of using similarity measure from [45] is mentioned. In [46] (FLEXFIS+), the merging based on membership function intersections is proposed and the overlapping index is calculated. If this index for the two clusters is higher than a predefined threshold and the angles between the local models' parameters are small the clusters are merged.

The eFuMo unsupervised merging is based on most commonly used principle of merging. It merges clusters

that are close together. The similarity and the vicinity of the two clusters are measured by the normalized distance:

$$d_{ik}^2 = (\mathbf{v}_i - \mathbf{v}_k)^T F_i^{-1}(\mathbf{v}_i - \mathbf{v}_k),\ \ i,k = 1,...,c\ \ i \neq k. \quad (41)$$

$$d_{norm_{ik}} = \sqrt{\frac{d_{ik}^2}{2\mathbf{s}_{i_{norm}}^T F_i^{-1} \mathbf{s}_{i_{norm}}}} \quad (42)$$

The distances are calculated only for clusters that have higher support from last change in cluster number than an user defined threshold (usually set to 20 for both values). The clusters are considered for merging if both normalized distances $d_{norm_{ik}}$ and $d_{norm_{ki}}$ are shorter than the predefined threshold ε_β:

$$d_{norm_{ik}} < \sqrt{-\ln(\varepsilon_\beta)} \quad (43)$$

If this criterion is satisfied, the distance ratio is checked:

$$\left| 1 - \frac{min(d_{norm_{ik}}, d_{norm_{ki}})}{max(d_{norm_{ik}}, d_{norm_{ki}})} \right| < k_{d_{merge}} \quad (44)$$

if the ratio is above the user defined threshold $k_{d_{merge}}$ (usually 10 percent) clusters are merged.

The parameters of new cluster are initialized as a weighted mean. The fuzzy covariance as proposed in [22]:

$$\begin{aligned}\mathbf{F}_{new} = \frac{1}{(Ns_i + Ns_k)^3}((Ns_i^3 + 2Ns_i^2 Ns_k + Ns_i Ns_k^2)\mathbf{F}_i + \\ +(Ns_k^3 + 2Ns_k^2 Ns_i + Ns_k Ns_i^2)\mathbf{F}_k + \\ +(Ns_i^2 Ns_k + Ns_i Ns_k^2)(\mathbf{v}_i - \mathbf{v}_k)(\mathbf{v}_i - \mathbf{v}_k)^T)\end{aligned} \quad (45)$$

The new center is calculated as:

$$\mathbf{v}_{new} = \frac{Ns_i \mathbf{v}_i + Ns_k \mathbf{v}_k}{Ns_i + Ns_k} \quad (46)$$

In the same manner a the new sum of past membership degree is calculated. New support of the cluster 47 and time of creation are calculated as weighted mean where weights are sum of past membership degrees (s_i, s_k):

$$Ns_{new} = \frac{Ns_i s_i + Ns_k s_k}{s_i + s_k}. \quad (47)$$

The local linear model parameters are calculated as weighted mean:

$$\theta_{new_j} = \frac{\omega_{i_j}\theta_{i_j} + \omega_{k_j}\theta_{k_j}}{\omega_{i_j} + \omega_{k_j}} \quad j = 1,...p, \quad (48)$$

where weights ω are the cluster supports Ns combined with a variance of the parameters.

The supervised merging considers the prediction model error. The supervised merging has three different measures to detect the clusters that could be merged together. It uses angles between local models' parameters (angle merging condition), correlation between clusters firing levels (correlation merging condition) and distance ratio (distance ratio merging condition). Only clusters that gathered higher support and sum of past membership degrees than a predefined threshold can be considered for supervised merging. The correlation coefficient is calculated based on monitoring of firing levels 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:

$$C_{ij}(k) = \frac{\beta_{ij}}{\beta_{ii}^{0.5}\beta_{jj}^{0.5}} \quad (49)$$

If the coefficient $C_{ij}(k)$ is above user-defined threshold (usually set to 0.9) the clusters i and j are considered for merging.

The distance ratio criterion for merging is similar than with the unsupervised merging. The distance ratio is calculated as:

$$\begin{aligned}d_{ik} = \sqrt{(\mathbf{v}_i - \mathbf{v}_k)^T \det(F_i)^{\frac{1}{p}} F_i^{-1}(\mathbf{v}_i - \mathbf{v}_k)} \\ K_d \frac{|1 - min(d_{ik}, d_{ki})|}{max(d_{ik}, d_{ki})}\end{aligned} \quad (50)$$

The clusters are considered for merging if the distance ratio K_d is lower than an user defined threshold (usually 0.05) and the correlation coefficient is at least half of the threshold defined for correlation merging condition.

The angle merging criterion is based on local models' angles. First the parameters are normalized. The algorithm sweeps all local models' parameters to find the vector of the largest absolute value of parameters. Then the parameters of local models are normalized with this vector. The angles for the two clusters for all parameters are calculated:

$$\alpha_{ij_k} = |\arctan(\theta_{i_k}) - \arctan(\theta_{j_k})| \quad (51)$$

where k is the parameter index. The clusters are consid-

ered for merging if all angles $\alpha_{ij_k}, k = 1,..,r$, where r is the number of local model's parameters, are below the user-defined threshold (usually set to 2 degrees) and the correlation coefficient is at least half of the threshold defined for correlation merging condition.

After the eFuMo identifies the possible merging pairs with the correlation, angle and distance ratio merging conditions it then checks the local models for the error:

$$\begin{aligned} \mathbf{x}_1 &= [1, \bar{u}_1, ..., \bar{u}_{p-1}]^T \\ e_1 &= |\theta_i^T \mathbf{x}_1 - \theta_j^T \mathbf{x}_1| \\ e_2 &= \sum_{r=1}^{p} |\theta_{i_r}(x_{1_r} + 2\sigma_{u_{r-1}}) - \theta_{j_r}(x_{1_r} + 2\sigma_{u_{r-1}})| \\ e_3 &= \sum_{r=1}^{p} |\theta_{i_r}(x_{1_r} - 2\sigma_{u_{r-1}}) - \theta_{j_r}(x_{1_r} - 2\sigma_{u_{r-1}})| \\ e &= \frac{1}{10.2\ \sigma_y} \sum_{r=1}^{3} e_r \end{aligned} \tag{52}$$

where $\bar{u}$ 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, $p-1$ is the number of inputs, j and i are the cluster indexes θ_i is the i-th cluster's local model parameter vector and θ_{i_r} is the r-th parameter of the i-th local model.

The pair that has the lowest error and the error is below the threshold is merged. The center of the merged cluster is positioned in the middle between maximum and minimum border of both clusters:

$$\begin{aligned} \mathbf{d}_1 &= \mathbf{v}_i - \mathbf{v}_j \\ \mathbf{v}_i' &= \mathbf{v}_i + \text{sign}(\mathbf{d}_1)\mathbf{s}_{i_{norm}} \\ \mathbf{v}_j' &= \mathbf{v}_j - \text{sign}(\mathbf{d}_1)\mathbf{s}_{j_{norm}} \\ \mathbf{d}_2 &= \frac{\mathbf{v}_i' - \mathbf{v}_j'}{2} \\ \mathbf{v}_{new} &= \mathbf{v}_j + \mathbf{d}_2 \end{aligned} \tag{53}$$

The fuzzy covariance matrix and support of a new merged cluster is initialized as a sum of both clusters' fuzzy covariance matrices and supports, local model parameters are initialized as a mean of both local models' parameters and the creation time is initialized to the creation time of the oldest cluster. The sum of past membership degrees is initialized to the max sum of past membership degrees of both clusters.

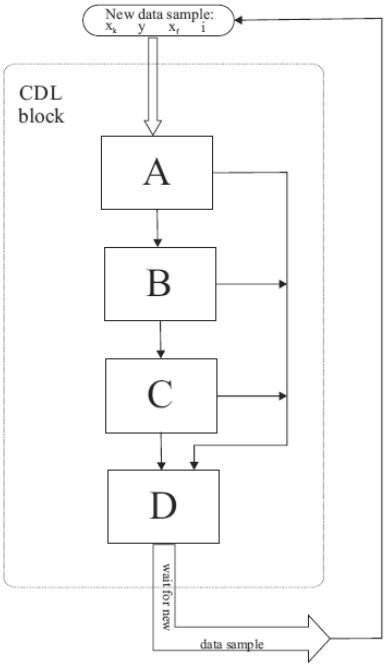


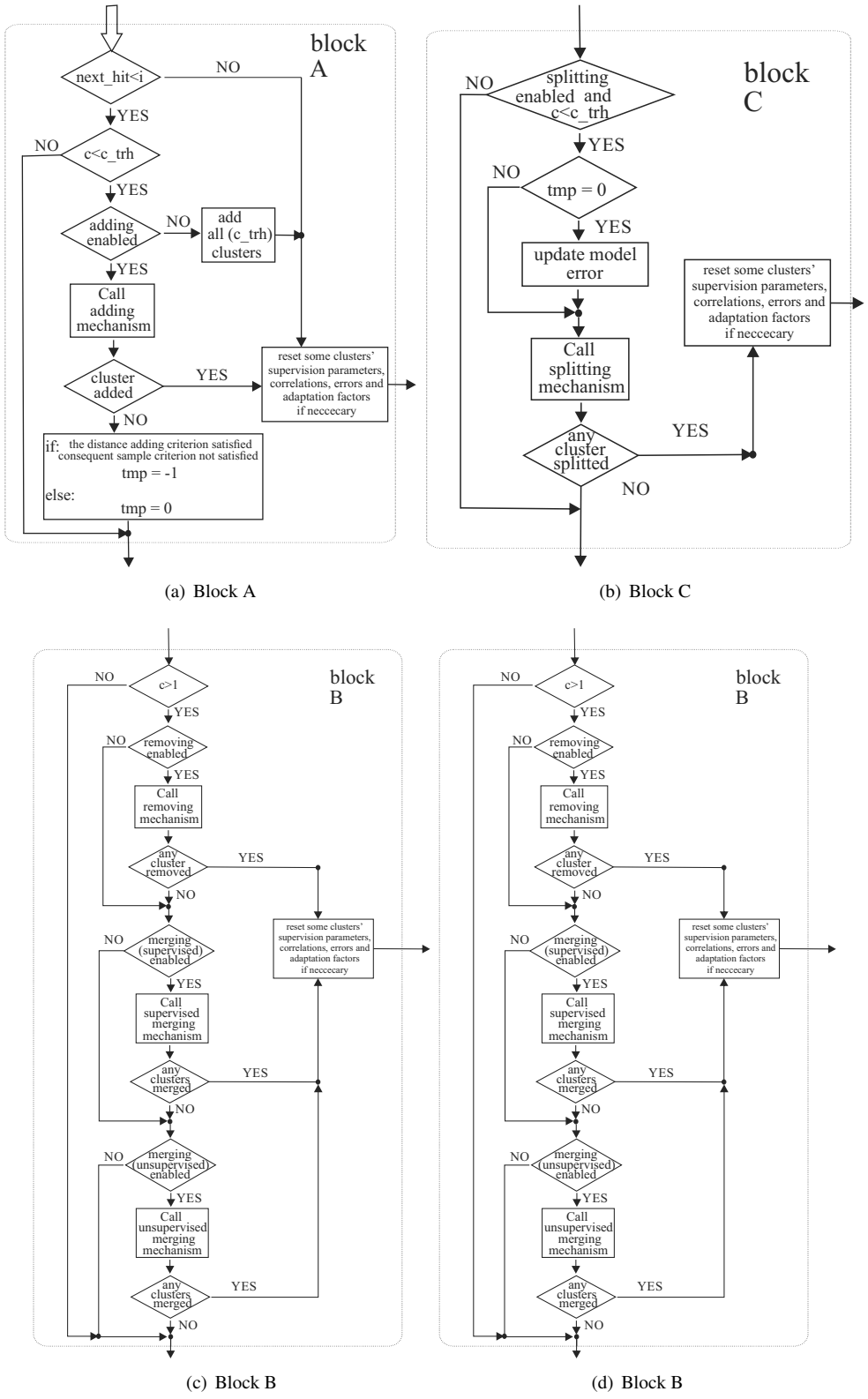
Figure 5: Scheme of the CDL.

1.3 Central decision logic

The CDL is responsible for proper flow of the operations. It controls the calls to evolving mechanisms and adaptation mechanisms. It also calculates the mean and standard deviations of the inputs and output of the process, that is identified with eFuMo. The scheme of the CDL block is shown on figure 5 and the sub-blocks are shown on figure 6.

The input to the eFuMo identification method are clustering vector ($\mathbf{x}_f$), regression vector ($\mathbf{x}_k$), output of the process (y) and number of current sample (i). The CDL block first checks the current sample number (i) to the sample number when the last change in cluster number was made and the user defined time delay. If the sum of these two values are smaller than a current sample number, the evolving mechanisms may be called. Otherwise the CDL skips the call to evolving mechanisms.

The CDL first calls the adding mechanism, then the removing mechanism, follows the supervised merging mechanism and unsupervised merging mechanism and at the end the CDL calls the splitting mechanism. If one of the mechanisms changes the cluster number other evolving mechanisms that follow are not called and the eFuMo continues with the adaptation algorithm.



(a) Block A (b) Block C

(c) Block B (d) Block B

Figure 6: The CDL scheme blocks. The c is the number of clusters, c_trh is the maximal allowed number of clusters, age_trh is the age threshold for minimal existence condition and $last\_change$ is the sample number when the last change in cluster number occurred.

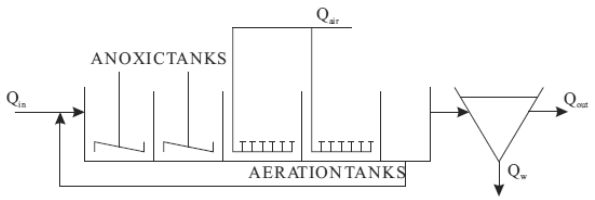


Figure 7: Scheme of the MBBR.

The CDL algorithm is also responsible for calculating the variance and mean of the input variables and output. The variance σ^2 is calculated on line by the following equation:

$$\sigma_x^2(k) = \frac{1}{k}\left((k-1)(\sigma^2(k-1)+\bar{x}(k-1)^2)+x(k)^2\right) - \\ -\frac{1}{k^2}\left((k-1)\bar{x}(k-1)+x(k)\right)^2 \tag{54}$$

where x is the variable and $\bar{x}$ is the mean of it, calculated as:

$$\bar{x}(k) = \frac{1}{k}\left((k-1)\bar{x}(k-1)+x\right) \tag{55}$$

If the splitting is enabled, the CDL also calls the error update algorithm. The algorithm updates a cluster error equation 38. This algorithm is only called if the current data sample doesn't satisfy the distance adding condition. The CDL also calculates the clusters' firing levels products used to calculate the correlation coefficient 49.

2 Monitoring Example

2.1 Monitoring system idea

The monitoring system that includes the evolving fuzzy model was tested on measured data from a pilot wastewater treatment plant, shown in figure 7. 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. In this example the monitoring of oxygen concentration in anoxic reactors will be done. The monitoring system is based on Takagi-Sugeno (TS) fuzzy model that estimates the relations between the input and output variables. The oxygen concentration is estimated from the air flow, the temperature in the reactor and the previous measurement of the oxygen concentration. First order local models are used. The inputs were selected by a backward selection. The idea is to detect the error in the process output based on the inputs. The outputs of the FDS are $y_{soft}(k)$ and $alarm(k)$. The output $alarm(k)$ indicates the presence of the fault in the measured signal ($alarm(k) = 1$: fault detected). The output $y_{soft}(k)$ is the process output with the removed fault. If there is no fault detected the output $y_{soft}(k)$ is equal to the process output $y(k)$. If the fault is detected, the output $y_{soft}(k)$ is calculated based on a fuzzy model that describes the proper relations between the input signals and the monitored signal.

The FDS determines the fault based on the internal fuzzy model of the signal relations. For monitored signal, three models are kept in the FDS's memory: a full evolving fuzzy model, 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 with the simulation error, which is calculated periodically on every 200-th 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 evolving fuzzy model'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 figure 8.

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

$$\sigma = \sum_{i=1}^{c} \beta_i \sqrt{F_{i_{r,r}}}, \tag{56}$$

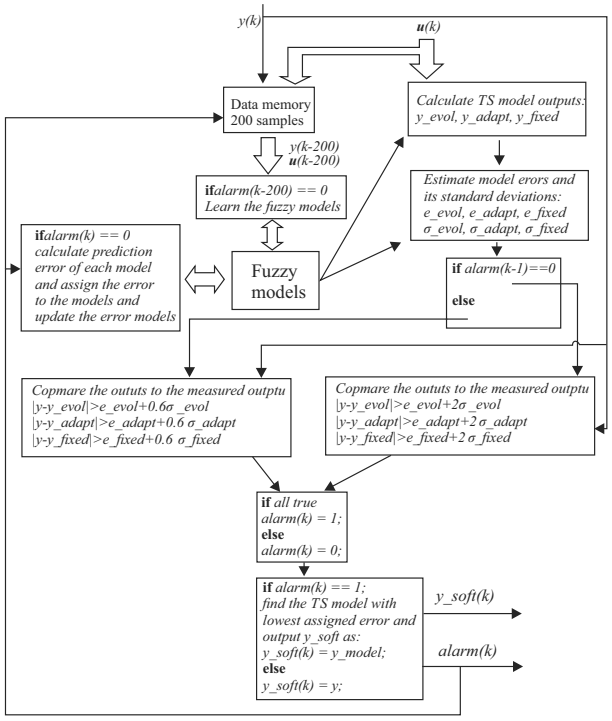


Figure 8: Scheme of the FDS for a subprocess.

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 figure 8, 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:

$$y_{soft} = \frac{((30 - k_{alarm})y_{model} + k_{alarm}y)}{30}, \qquad (57)$$

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.

2.2 Detecting the false alarms due to manual calibration

Manual tuning and offset repairs of the oxygen concentration signal is performed every few months. This is seen on the upper graph in figure9. 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. 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 equations 55 and 54. 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 algorithm is turned off when, for at least a certain number of consecutive samples (in our case 100), the variance of error is below the model and process variance. The algorithm is also turned off if its maximum functioning time is reached.

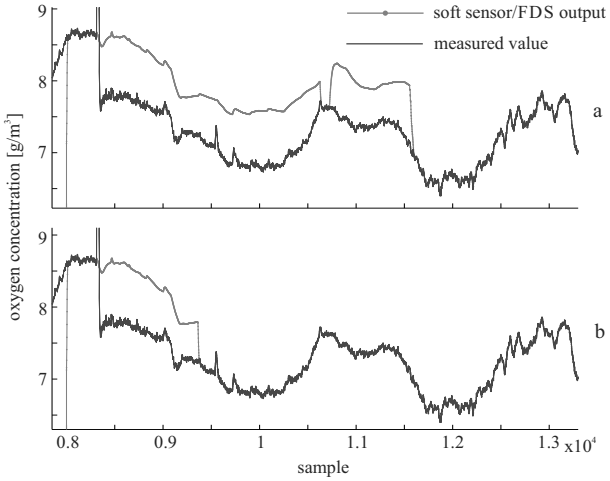


Figure 9: Effect of sensor calibration.

3 Results and Discussion

The presented idea was tested on real data. To estimate the performance of the system during a sensor's

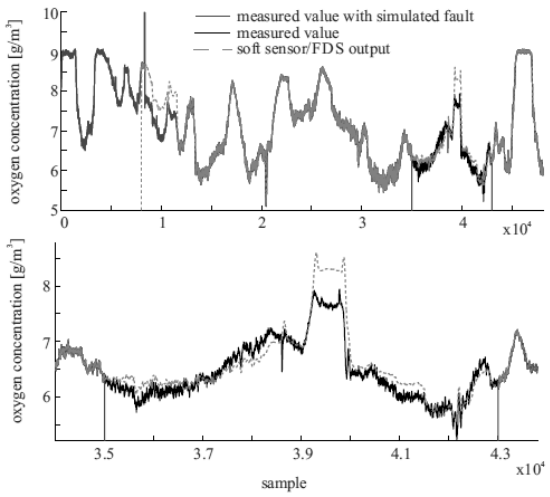


Figure 10: Oxygen-concentration fault detection.

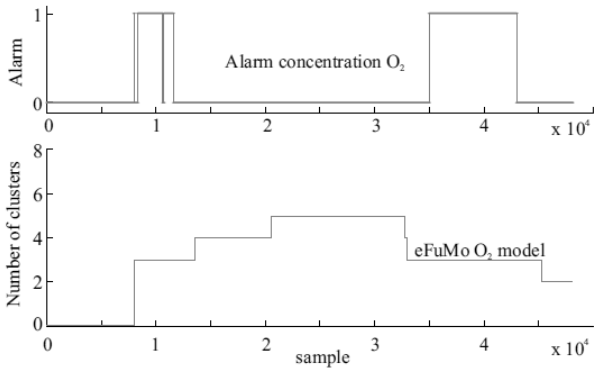


Figure 11: Alarm signal and number of clusters over the experiment.

Estimation Error	Concentration O_2
NDEI	0.488
min. abs.	2.83e-5 $[g/m^3]$
max. abs.	1.347 $[g/m^3]$
avg. abs.	0.189 $[g/m^3]$
signal range	2.72 $[g/m^3]$
min. rel.	1.04e-5
max. rel.	0.495
avg. rel.	0.0695
faulty samples $[10^3]$	$35-43$

Table 1: Estimation error during the simulated fault.

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 hours). Usually, the faults last from about a few minutes up to 6 hours. The settings of the evolving method were obtained based on trail and error. The fault was simulated between the samples 35000 and 43000. The whole experiment is shown in figure 10. 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 figure 11. 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.

Even though the estimated signal is not entirely covering the measured signal, we believe that the estimation accuracy is still good enough. The error between the measured and estimated signal during the fault is given in Table 1. 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.

As can be seen on the upper graph in figure 9, 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 11500, 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 figure 9, the false-alarm detection was implemented. 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 figure 12, the course of variances are shown. 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

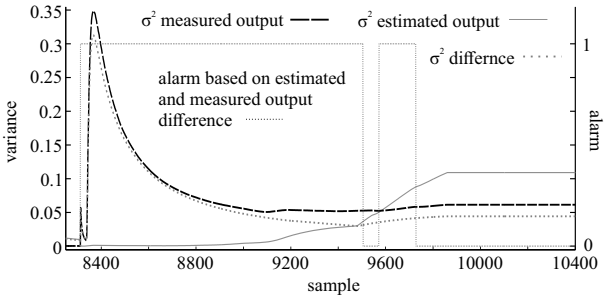


Figure 12: The course of variances.

this, the conditions for overriding the original alarm are met. The last alarm based on the output differences is raised at sample 9740. Therefore, the variance procedure is switched off at sample 9840. The procedure successfully detected the signal offset caused by the manual calibration.

4 Conclusion

In this paper an evolving fuzzy model method for on-line learning of fuzzy models was presented. The method is useful when dealing with nonlinear time-varying processes. The method was used in an example of fault detection system. The presented results show that the approach can be successfully used for such tasks. The only issue of the method and all such methods is in its tuning. There are a number of parameters that need to be tuned. Their tuning highly depends on a problem and require an expert to tune them. Further research will be focused on lowering the number of tuning parameters and on self tuning of the method.

References

[1] Johanson TA, Murray-Smith R. *Operating regime approach to nonlinear modeling and control*. UK: Taylor Francis. 1981.

[2] Hwang CL, Chang LJ. Fuzzy neural-based control for nonlinear time-varying delay systems. *IEEE Trans Syst Man Cyber part B*. 2007;37(6):1471–1485.

[3] Shing J, Jang R. ANFIS : Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans Syst Man Cyber*. 1993;23(3):665–685.

[4] Azeem MF, Hanmandlu H, Ahmad N. Structure identification of generalized adaptive neuro-fuzzy inference systems. *IEEE Trans on Fuzzy Syst*. 2003; 11(5):666–681.

[5] Dovzan D, Skrjanc I. Recursive fuzzy c-means clustering for recursive fuzzy identification of time-varying processes. *ISA Transactions*. 2011; 50(2):159 – 169.

[6] Dovzan D, Skrjanc I. Recursive clustering based on a Gustafson-Kessel algorithm. *Evolving Systems*. 2011; 2:15–24.

[7] Patt J. A resource allocating network for function interpolation. *Neural Computat*. 1991;3(2):213–225.

[8] Juang CF, Lin CT. An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Trans on Fuzzy Syst*. 1998;6(1):12–32.

[9] Lin FJ, Lin CH, Shen PH. Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Trans on Fuzzy Syst*. 2001;9(5):751–759.

[10] Tzafestas SG, Zikidis KC. NeuroFAST: On-line neuro-fuzzy ART-based structure and parameter learning TSK model. *IEEE Trans Syst Man Cyber Part B*. 2001;31(5):797–802.

[11] Kasabov NK, Song Q. DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction. *IEEE Trans on Fuzzy Syst*. 2002;10(2):144–154.

[12] Angelov PP, Filev DP. An approach to on-line identification of Takagi-Sugeno fuzzy models. *IEEE Trans Syst Man Cyber Part B*. 2004;34(1):484–497.

[13] Lughofer E, Klement EP. FLEXFIS: A Variant for Incremental Learning of Takagi-Sugeno Fuzzy Systems. In: *Proceedings of The 2005 IEEE International Conference on Fuzzy Systems*. Reno, Nevada, USA. 2005; pp. 915 – 920.

[14] M Pratama PA S G Anavatti, Lughofer E. A novel incremental learning machines. *IEEE Transactions on*

Neural Networks and Learning Systems. 2014; 25(1):55–68.

[15] Rong HJ, Sundararajan N, Huang GB, Saratchandran P. Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. *Fuzzy Sets and Syst*. 2006;157(9):1260–1275.

[16] Leng G, Prasad G, McGinnity TM. An on-line algorithm for creating self-organizing fuzzy neural networks. *Neural Networks*. 2004;17:1477 – 1493.

[17] Huang GB, Saratchandran P, Sundararajan N. A Recursive Growing and Pruning RBF (GAP-RBF) Algorithm for Function Approximations. In: *Proceedings of The Fourth International Conference on Control and Automation (ICCA-03)*. Montreal, Canada. 2003; pp. 10 – 12.

[18] Kasabov NK. Evolving fuzzy neural networks for supervised/unsupervised on-line knowledge-based learning. *IEEE Trans Syst Man Cyber Part B*. 2001; 31(6):902–918.

[19] Kasabov N. Evolving fuzzy neural networks-Algorithms, applications and biological motivation. In: *Methodologies for the Conceptation, Design and Application of Soft Computing*. Japan. 1998; pp. 271–274.

[20] Wu S, Er MJ. Dynamic fuzzy neural networks - A novel approach to function approximation. *IEEE Trans Syst Man Cyber Part B*. 2000;30(2):358–364.

[21] Wu S, Er MJ, Gao Y. A fast approach for automatic generation of fuzzy rules by generalized dynamicc fuzzy neural networks. *IEEE Trans on Fuzzy Syst*. 2001;9(4):578–594.

[22] Soleimani-B H, Lucas C, Araabi BN. Recursive Gath-Geva Clustering as a Basis for Evolving Neuro-Fuzzy Modeling. *Evolving Systems*. 2010; 1(1):59 – 71.

[23] Angelov P. *EVOLVING INTELLIGENT SYSTEMS: Methodology and Applications*, chap. Evolving Takagi-Sugeno Fuzzy Systems From Streaming Data (eTS+), pp. 21 – 50. New Jersey: Wiley. 2010;.

[24] Lughofer E. *Learning in Non-Stationary Environments: Methods and Applications*. Springer.

[25] Kalhor A, Araabi B, Lucas C. An online predictor model as adaptive habitually linear and transiently nonlinear model. *Evolving Systems*. 2010;1(1):29–41.

[26] Rubio J. SOFMLS: Online self-organizing fuzzy modified least-squares network. *IEEE Transactions on Fuzzy Systems*. 2009;17(6):1296–1309.

[27] Lughofer E. On-line assurance of interpretability criteria in evolving fuzzy systems - achievements, new concepts and open issues. *Information Sciences*. 2013; 251:22 – 46.

[28] Lughofer E. *Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications*, vol. 266 of *Studies in Fuzziness and Soft Computing*. Springer.

[29] de Jesús Rubio J. *EVOLVING INTELLIGENT SYSTEMS Methodology and Applications*, chap. STABILITY ANALYSIS FOR AN ONLINE EVOLVING NEURO-FUZZY RECURRENT NETWORK, pp. 173–199. Hoboken, New Jersey: John Wiley & Sons. 2010;.

[30] Blažič S, Škrjanc I, Gerkšič S, Dolanc G, Strmčnik S, Hadjiski MB, Stathaki A. Online fuzzy identification for an intelligent controller based on a simple platform. *Engineering Applications of Artificial Intelligence*. 2009;22:628–638.

[31] Dovžan D, Škrjanc I. Predictive functional control based on an adaptive fuzzy model of a hybrid semi-batch reactor. *Control Engineering Practice*. 2010;18(8):979 – 989.

[32] Hartert L, Mouchaweh MS, Billaudel P. A semi-supervised dynamic version of Fuzzy K-Nearest Neighbours to monitor evolving systems. *Evolving Systems*. 2010;1:3–15.

[33] Lin CT. A neural fuzzy control system with structure and parameter learning. *Fuzzy Sets and Syst*. 1995; 70:183–212.

[34] Crespoa F, Weberb R. A methodology for dynamic data mining based on fuzzy clustering. *Fuzzy Sets and Systems*. 2005;150:267–284.

[35] Hassibi B, Stork DG. Second-order derivatives for network pruning: Optimal brain surgeon. *Advances in Neural Information Processing*. 1993;4:164–171.

[36] Leung CS, Wong KW, Sum PF, Chan LW. A pruning method for the recursive least squared algorithm. *Neural Networks*. 2001;14:147–174.

[37] Fritzke B. A growing neural gas network learns topologies. *Adv Neural Inform Processing Syst*. 1995; 7:845–865.

[38] Asif M, Angelov P, Ahmed H. An Approach to Real-time Color-based Object Tracking. In: *Evolving Fuzzy Systems, 2006 International Symposium on*. 2006; pp. 86 –91.

[39] Angelov P, Zhou X. Evolving Fuzzy Systems from Data Streams in Real-Time. In: *Evolving Fuzzy Systems, 2006 International Symposium on*. 2006; pp. 29 –35.

[40] Hall P, Hicks Y. A method to add Gaussian mixture models. *Tech. rep.*, Department of Computer Science, University of Bath. 2004.

[41] Declercq A, Piater J. Online learning of gaussian mixture models - a two-level approach. In: *Proceedings of the 3rd international conference on computer vision theory and applications VISAPP*. Funchal, Portugal. 2008; pp. 605–611.

[42] Lughofer E. A dynamic split-and-merge approach for evolving cluster models. *Evolving Systems*. 2012;DOI: 10.1007/s12530-012-9046-5.

[43] Skrjanc I, Hartmann B, Banfer O, Nelles O, Sodja A, Teslic L. Supervised Hierarchical Clustering in Fuzzy Model Identification. *Fuzzy Systems, IEEE Transactions on*. 2011;PP(99):1.

[44] Frigui H, Krishnapuram R. A robust algorithm for automatic extraction of an unknown number of clusters from noisy data. *Pattern Recognition Letters*. 1996; 17:1223–1232.

[45] Wang WJ. New similarity measures on fuzzy sets and on elements. *Fuzzy Sets and Systems*. 1997;85:305–309.

[46] Lughofer E, Bouchot JL, Shaker A. On-line elimination of local redundancies in evolving fuzzy systems. *Evolving Systems*. 2011;(2):165–187.