2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A new Unsupervised Approach to Fault Detection and Identification

Bruno Sielly Jales Costa
Campus Natal – Zona Norte
IFRN
Natal, RN, Brazil
bruno.costa@ifrn.edu.br

Plamen Parvanov Angelov
School of Comput. and Communications
InfoLab21, Lancaster University
Lancaster, United Kingdom
p.angelov@lancaster.ac.uk

Luiz Affonso Guedes
Department of Comput. Engineering
UFRN
Natal, RN, Brazil
affonso@dca.ufrn.br

***Abstract*—In this paper, a new fully unsupervised approach to fault detection and identification is proposed. It is based on a two-stage algorithm and starts with the recursive density estimation (RDE) in the feature space. The choice of the features is important and in the real world process that we consider these are control and error related variables. The basis of the proposed approach is the fully unsupervised evolving classifier AutoClass which can be seen as an extension of the earlier one, but is using data clouds and data density information. It has to be stressed that the density in the data space is not the same as the well-known and widely used in statistics probability density function (pdf) although it looks similar. The density in the data space, D is pivotal and instrumental for anomaly detection. It can be calculated recursively, which makes it very efficient in terms of memory, computational power and, thus, applicable to on-line applications. Importantly, the proposed method not only can detect anomalies, but also can identify and diagnose the fault during the second stage of the process. While the first stage is centred around RDE, the second stage is based on the evolving fuzzy rule-based (FRB) classifier AutoClass. A key advantage of AutoClass is that it is fully unsupervised (there is no need to pre-specify the fuzzy rules, number of classes) and can start learning "from scratch". AutoClass can be initialised with some prior knowledge (assuming that it does exists) and evolve/develop it further, but that is not mandatory. This new approach is generic, but in this paper without limiting the concept it is validated on a lab based control kit. In this particular example, the features are the control and error signals. The results significantly outperform alternative methods, which is in addition to the advantages that the approach is autonomous.**

Keywords*—Fault detection, fault identification, recursive density estimation, evolving classifiers, autonomous learning.

I. INTRODUCTION

Fault detection and identification (FDI) field of research has received considerable attention in the past four decades. It has been shown that human errors are, by far, the main reason for accidents in industrial environments [1]. One of the main goals in industrial research is the detection of faults while the system is still operating in a controllable region. Early detection is responsible for preventing or, at least, reducing productivity losses and health risks.

The process of abnormal event management (AEM) is usually divided into a series of stages, which is often called diagnosis scheme. The first stage addresses what is known as fault or anomaly detection, and is responsible for identifying whether the system is working properly (normal state of operation) or is in a faulty state. The subsequent stage, known as fault identification or classification, is reached when a fault is detected during the first stage, and refers to the determination of type, location and time of detection of the fault.

Each of these stages presents its own challenges and they are often solved independently. Many authors have addressed FDI in the literature. We can mention, for example, the observer-based [2], analytical redundancy-based [3], fuzzy model-based [4], neural network-based [5], and so on. Unfortunately, most of the above mentioned approaches require either previous knowledge or empirical observation about the model or behaviour of the system, need extensive computational efforts or too many thresholds or problem-specific parameters to be pre-defined in advance, hampering their use in on-line applications. Ergo, these technical features make difficult their adoption in real industrial problems.

In order to overcome these problems, in this paper we propose an on-line and fully unsupervised two-stage FDI algorithm, in which detection and identification are presented as sequential and independent procedures.

The detection algorithm is based on the Recursive Density Estimation (RDE) [6], which allows building, accumulating, and self-learning a dynamically evolving information model of "normality" based on the process data for particular specific plant based on the normal/accident-free cases only. Similarly to Statistical Process Control (SPC) [7], RDE is an on-line statistical technique, however, it does not require that the process parameters follow Gaussian/normal distributions nor makes other prior assumptions.

The identification procedure, on the other hand, is based the new self-learning fuzzy rule-based (FRB) classification technique, called *AutoClass*. It builds upon the concept of evolving FRB classifiers, such as eClass0, eClass1 Simpl_eClass [8], [9], or the evolving classifiers reported in [10]. While the referred approaches are based on the traditional concept of clusters, AutoClass works with the so called *data clouds*, which are structures with no specific

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

shapes or boundaries, expanding, this way, the idea of the AnYa FRB systems introduced in [11].

Both stages of the proposed FDI approach can start "from scratch", from the very first data sample acquired, with no previous knowledge about the plant model or dynamics, training or complex user-defined parameters or thresholds. The generated fuzzy rules have no specific parameters or shapes for the membership functions and the approach is entirely data-driven.

Thus, it is fully autonomous, does not require type or number of faults to be known beforehand or to have a dedicated training phase. Yet it outperforms the best known statistical process control approach

The remainder of the paper is organised as follows: in Section II, the new fault detection algorithm is presented; Section III details the new fault identification procedure; Section IV addresses the experimental setup, the obtained results and the results analysis; finally, in section V, the main conclusions are presented.

I. Fault Detection Stage

The first stage of the FDI proposal is the fault detection. The FD algorithm is based on the RDE approach [6], [12]. RDE is a very suitable approach for outlier detection and has been recently used in many practical applications [13]. The main idea of RDE is the estimation of the proximity in the *n*-dimensional data space. The referred density is based on a Cauchy function and, thus, can be calculated recursively, which makes the FD algorithm, memory- and computational-efficient, since it does not request the storage of previous data samples. It is also data-driven and free of complex user-defined parameters.

Let all measurable physical variables form the vector $x \in R^n$ and are divided into several data clouds, Λ. Then, for any vector $x \in R^n$, its Λ-th local density value is calculated for Euclidean type distance as [6]:

$$d_\Lambda = \frac{1}{1 + \frac{1}{N_\Lambda}\sum_{i=1}^{N_\Lambda}\|x_k - x_i\|^2} \quad (1)$$

where d describes the local density of the cluster Λ; N_Λ denotes the number of data samples associated with the cluster Λ; x_k represents the data vector measured at the time instant k.

According to [12], [6], the global density, D can be derived as an exact (not approximated or learned) quantity as

$$D(x_k) = \frac{1}{1 + \|x_k - \mu_k\|^2 + \mathrm{X}_k - \|\mu_k\|^2} \quad (2)$$

where both, the mean, μ_k, and the scalar product, X_k can be updated recursively as follows:

$$\mu_k = \frac{k-1}{k}\mu_{k-1} + \frac{1}{k}x_k\,, \quad \mu_1 = x_1 \quad (3)$$

$$\mathrm{X}_k = \frac{k-1}{k}\mathrm{X}_{k-1} + \frac{1}{k}\|x_k\|^2\,, \quad X_1 = \|x_1\|^2 \quad (4)$$

Local density, d_Λ can also be updated similarly to the global density, but applied to over the data points associated with the respective data cloud only (not all the data).

The proposed FD procedure starts with the initialisation of the step counter $k = 1$, which tracks total number of data samples read so far, and the counter $ks = 1$, which tracks the number of iterations within the current status ("normal" or "fault"). The *status* is, then, initialised with the value "normal", as we assume that the detection process will always start from a fault-free state of operation.

The first data sample x_k is read from one of the available interfaces (data files, data bases, industrial protocols and so on). The density $D(x_k)$ is calculated and the mean, μ_k, and scalar product, X_k, are recursively updated by the equations (2), (3), and (4), respectively. We, then, calculate the density variation Δ_D from the time step *k-1* to k by $\Delta_D = abs(D(x_k) - D(x_{k-1}))$. This measure will be used to determine if there is a considerable change in the density behaviour to update the status of the system.

The mean of the density, μ_D, is now recursively calculated and updated as:

$$\mu_D(ks) = \left(\frac{ks-1}{ks}\mu_D(ks-1) + \frac{1}{ks}D(x_k)\right)(1 - \Delta_D) + D(x_k)\Delta_D \quad (5)$$

The equation (5) also does not need storing any previous values in the memory, which is appropriate for an on-line approach. The mean of density is based on the principles of the equation (3), although is not as conservative. This measure is sensitive to abrupt changes, since it considers the length of the density variation while updating the mean.

In general, three scenarios can occur:

a) IF the current *status* of the system is "normal" and the density $D(x_k)$ is less than the mean of density μ_D for the last ε_1 seconds, the *status* is changed to "fault", or
b) ELSE if the current *status* of the system is "fault" and the density $D(x_k)$ is greater than the mean of density μ_D for the last ε_2 seconds, the *status* is changed to "normal", or
c) ELSE, do nothing.

In both cases a) and b), the counter ks is re-initialised ($ks = 0$). Note also, that we use two very intuitive "status change" thresholds (2 and 8 seconds). They are used to determine whether the system will switch from "normal" to "fault" state, which happens when the density is below the mean for ε_1 consecutive seconds, and from "fault" to "normal" states, which occurs when the density is above the mean for ε_2 consecutive seconds, regardless of the sampling period of the process. Recommended values for the the parameters ε_1 and ε_2 are 2 and 8 (seconds). These values

represent a good trade-off between response time and robustness of the detection system.

Finally, the process is terminated and starts from the reading of the next data sample and the time step k and counter ks are incremented. The full detection algorithm is presented in the flowchart in Figure 1.

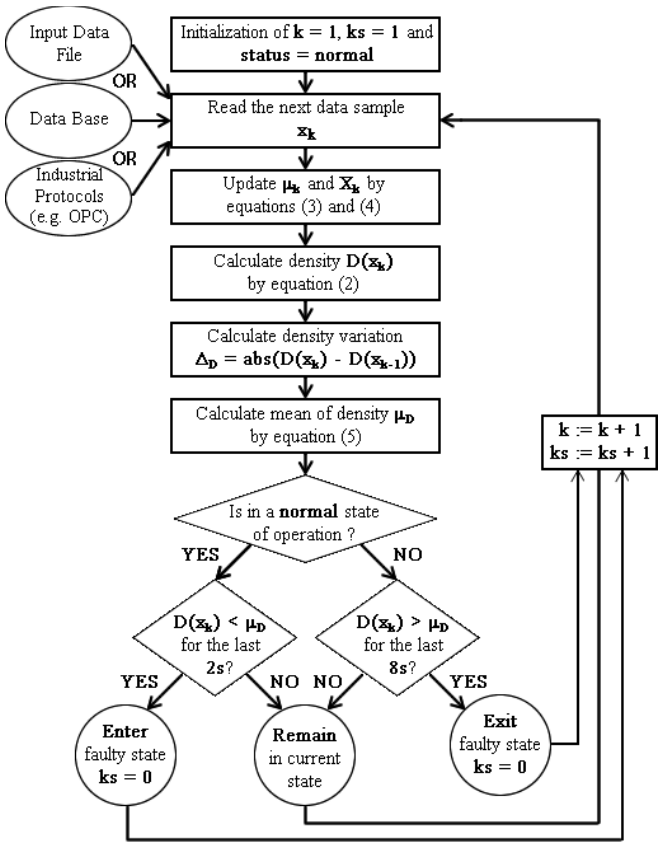


Fig. 1. Flowchart of the proposed fault detection algorithm

II. Fault Identification Stage

Once a fault is detected by the procedure described in Section II, the fault identification system is enabled (it remains disabled during a normal operation). It is important to stress that both detection and identification stages are fully independent and can be used separately with other existing approaches.

The proposed fault identification algorithm is based on the self-learning and fully unsupervised evolving classifier algorithm called *AutoClass* which is an AnYa-like FRB classifier and, unlike the traditional FRB systems (e.g. Mamdani, Takagi-Sugeno), AnYa does not require the definition of membership functions [11]. The antecedent part of the inference rule uses the concepts of data clouds [11] and relative data density, representing exactly the real distribution of the data.

A data cloud is a collection of data samples in the *n*-dimensional space, similar to the well-known data clusters, however, it is different since a data cloud is non-parametric and it does not have a specific shape or boundary [6] instead it follows the exact real data distribution. A given data sample can belong to all the data clouds with a different degree $\gamma \in [0;1]$, thus the fuzzy aspect of the model is preserved.

The consequent of the inference rule in AutoClass is a zero-order Takagi-Sugeno crisp function, i.e. a class label $L_i = [1, K]$. The inference rules follow the construct of an AnYa FRB system [11]:

$$\Re^i: \textbf{IF } (\vec{x} \sim \aleph^i) \textbf{ THEN } (L^i)$$

where $\Re^i$ is the *i*-th rule, $\vec{x} = [x_1, x_2, \dots, x_n]^T$ is the input data vector, $\aleph^i \in \mathcal{R}^n$ is the *i*-th data cloud and $\sim$ denotes the fuzzy membership expressed linguistically as "is associated with" or "is close to".

The inference in AutoClass is produced using the well-known "winner takes all" rule [8]:

$$L = L^{i*}, \;\; i^* = argmax_{i=1}^{n}(\gamma^i) \tag{6}$$

where γ^i represents the degree of membership of the input vector x_k to the data cloud X^i, defined as the relative local density and recursively calculated as

$$\gamma_k^i = \frac{1}{1 + \|x_k - \mu_k\|^2 + \mathrm{X}_k - \|\mu_k\|^2} \tag{7}$$

where μ_k and X_k are updated by equations (3) and (4).

The classification process starts with the definition of the initial zone of influence (ZI) by the user. The concept of ZI is similar to the radius in data clusters, although data clouds do not have specified boundaries. Initial values of $ZI \in [0.3; 0.5]$ can be recommended [6]. The time step k is also initialised ($k = 1$).

The first data sample x_k is then read. Since the algorithm does not require any *a priori* information about the process or any kind of training, the fuzzy rule basis and cloud set is empty at this point. AutoClass will, then, create the first data cloud, which will be associated with one point (the first data sample), with its ZI equal to the initial ZI defined by the user and its focal prototype point, FP, equal to x_k. The newly created data cloud is, then, added to the vector data clouds and a label "Class 1" as the consequent part will compose the first inference rule:

$$\Re^1: \textbf{IF } (\vec{x} \sim cloud^1) \textbf{ THEN } ("Class\ 1")$$

The time step k is, then, incremented by one and *AutoClass* continues by the reading the next data sample x_k. From $k = 2$ onwards, at each execution run and data sample x_k being read, three scenarios can occur:

a) x_k is close to an existing data cloud:

If the current data sample is within two times the zone of influence of an existing data cloud, x_k will be associated with that cloud. The focal point of th(i/e)s(e) data cloud(s) is/are

updated as a weighted sum of the current focal point (mean) and x_k and the zone of influence of the referred cloud is updated as the weighted sum of the current ZI and the Euclidean distance from x_k to the focal point of the data cloud. Finally, the number of points, NP, associated with that cloud is incremented by one. Note, that there is no need to store any read data samples, since all the information about the data cloud is recursively updated, which is a crucial concern when developing on-line applications.

After the update of all close data clouds, the time step k is incremented by one and the procedure continues by reading the next data sample x_k.

b) x_k is not close to an existing data cloud:

In this case, the data sample x_k is, firstly, classified as an outlier and added to the vector of *outliers*. This vector is limited in size and, thus, in stack memory, in order not to compromise the performance of the on-line algorithm. In this work, the maximum number of data samples to be stored is 100. When the vector reaches its full capacity, the older data sample is removed.

After the *outliers* vector is updated, the procedure analyses all *potential data clouds* to be created from stored outliers that are close to each other, considering again the premise of two times the zone of influence. Let's define nP as the number of points of the more populated *potential cloud*, $minP$ as 15% of the points of the less populated *existing cloud* , but not less than 3. Let's also define HD as the density of the more populated *potential cloud* and $D^{mean} = (\sum_{i=1}^{R} D^i)/R$ (where R is the number of rules so far) the average density of all *existing clouds*. If nP is greater than $minP$ and HD is greater than D^{mean}, a new cloud nc will be created. This condition considers both the size (number of points) of the potential cloud to be created, and its density. We consider that, in order to form a cloud, the set needs, at least, 3 points (this value will be increased to 15% of the less populated cloud) and density higher than the average of all existing clouds. This way, we try to avoid major discrepancies among the generated clouds.

If the conditions above are satisfied, the newly created cloud nc will be associated with nP points, its ZI will be equal to the mean of the ZIs of all existing clouds and the initial ZI, defined by the user, and its focal point, FP, will be equal to the mean of all data samples associated with nc. The newly created cloud is, then, added to the vector clouds (and all data samples associated with nc removed from the vector *outliers*) and a label L^{R+1} as the consequent part will compose the *(R+1)*-th inference rule:

$$\Re^{R+1}: \textbf{IF } \left(x_k \sim \aleph^{R+1}\right) \textbf{ THEN } (L^{R+1})$$

After the creation of the new cloud, or if the conditions of number and density are not satisfied, the time step k is incremented by 1 and AutoClass continues by reading the next data sample x_k. The full FI algorithm is presented in the flowchart in Figure 2.

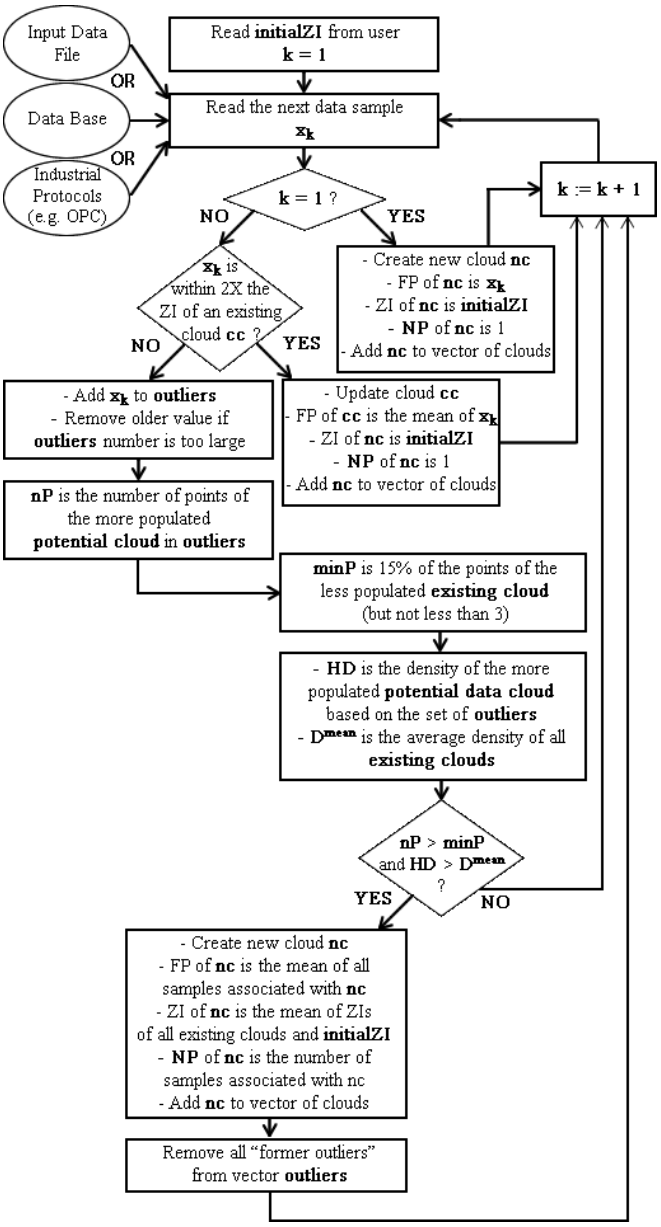


Fig. 2. Flowchart of the proposed fault identification algorithm

III. Experiment and Results

In order to validate the proposal, we developed a liquid level control application, using a Laboratory pilot plant for industrial process control, developed by DeLorenzo do Brasil [14]. The plant consists of two pressurised tanks, T1 and T2, two valves, V1 and V2, and a centrifugal pump, all connected by a piping system, which enables the flow between the two vessels, in both directions [15].

The plant is controlled by a multistage fuzzy controller [16], developed in JFuzZ [17] software tool through an OPC (OLE for Process Control) interface [18]. The behaviour generated by the controller, when the process has reached the regime state and the control action and error signals are

stabilised (with no considerable oscillation), represent the "normal" state of operation of the plant.

A set of 16 different faults were generated physically and by software, is the object of study of this experiment. The referred faults are divided in 4 groups, depending on the nature of the fault: actuator, leakage, stuck valves and disturbance-related.

Each group contains experiments with different patterns and levels. In the "actuator" group, there are 6 levels of offsets in the centrifugal pump ($F_1 = +2\%$, $F_2 = +4\%$, $F_3 = +8\%$, $F_4 = 2\%$, $F_5 = 4\%$ and $F_6 = 8\%$); in the 'leakage' group there are 3 levels ($F_7 = 33\%$, $F_8 = 66\%$ and $F_9 = 100\%$) of open drain, which simulates a physical leakage in the tank T1; in the "stuck valves" group there are 3 levels of jamming of each valve (F_{10}, F_{11}, F_{12}, F_{13}, F_{14} and F_{15}); and in the "disturbance" group there is 1 environment disturbance (F_{16}).

Both in the detection and identification stages each data sample x_k is a point of a n-dimensional feature space. For the detection stage, in this application we chose to use two features – the control action (u) and error (e), calculated by $e = r$ (reference) – y (output), thus, $x = \{u, e\}$ – as they are very representative signals and meet the basic requirement of being constant when the plant is operating normally and oscillatory in the presence of a fault.

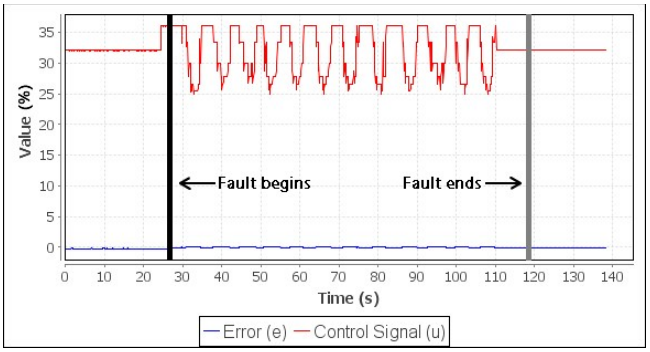
For the identification stage, we also chose to use two features – here called Feature 1 and Feature 2, thus $x = \{Feature\ 1,\ Feature\ 2\}$ – where the first one is the period and the second one is the amplitude of the control signal. It has been noticed that, in most generated faults, the control signal assumes a periodic behaviour, with nearly constant frequency. While the period is useful to distinguish between different classes of faults, the amplitude is also important to define the level of the fault.

We, then, divided the experiment in two separate stages: i) detection and ii) identification/classification. In the first one, we analysed the data collected from the operating plant, sequentially, in the presence of the 16 types of fault. The results obtained with the proposed detection algorithm for the fault F_{11} are shown in Figure 3.

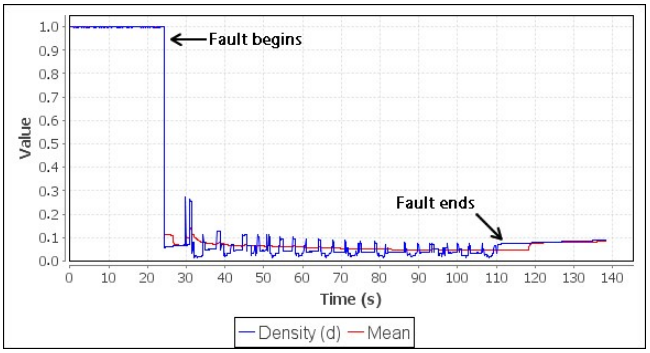
Note, that the algorithm was able to almost immediately detects the beginning of and the end of the fault, as seen in Figure 3(a). A fault is detected when there is a considerable drop in the density signal, which, in the case of fault F_{11}, occurs around 28s of execution, as seen in Figure 3(b). The exit of a faulty state, on the other hand, occurs when the density signal starts to considerably rise, which means that there is no oscillation, and is around 119s for the fault F_{11} example.

For comparison purposes, the same experiment was performed both with the proposed fault detection and SPC approaches. SPC is a well-known algorithm for outlier detection in industrial processes. The procedure details were exhaustively presented in literature [7].

After all 16 experiments, the proposed detection algorithm obtained a hit rate of 86.97%, considering the right classification of faulty and normal states. The SPC technique, on the other hand, using similar setups, obtained a hit rate of 55.37%. Individually, the proposed application resulted in 9 out of 16 experiments with a hit rate in excess of 95%, which demonstrated the efficiency of the algorithm. The results for all 16 experiments with the SPC and the proposed detection algorithm are detailed in Table 1.



(a) Control behaviour chart



(b) Density chart

Fig. 3. Results for fault F_{11} with the proposed fault detection application

TABLE I. RESULTS FROM FAULT DETECTION ALGORITHMS

Fault	Hit Rate *SPC*	Hit Rate *Proposed Algorithm*
F_1	64.13 %	**97.84 %**
F_2	71.46 %	**98.48 %**
F_3	50.23 %	**98.63 %**
F_4	56.66 %	**96.7 %**
F_5	61.41 %	**96.46 %**
F_6	74.76 %	**98.48 %**
F_7	50.29 %	48.36 %
F_8	45.46 %	75.59 %
F_9	61.64 %	**95.46 %**
F_{10}	45.78 %	**98.93 %**
F_{11}	62.4 %	88.61 %
F_{12}	52.3 %	77.14 %
F_{13}	48.25 %	66.92 %
F_{14}	33.62 %	90.31 %
F_{15}	46.21 %	**98.75 %**
F_{16}	61.3 %	64.86 %

In the identification/classification stage, AutoClass is executed after a fault is detected and stops when the system exits the faulty state. This stage is quite unique in the sense that it autonomously and in a completely unsupervised manner

(without any pre-training or prior knowledge and information) identifies the types of faults.

For this experiment, we applied 4 sequential fault data streams to AutoClass (F_2, F_4, F_1 and F_9). The progress of execution and behaviour of the system are illustrated in the next charts. Similarly to the previous figures, black bars indicate the moment when the fault is detected and grey bars indicate the moment when the system exits a faulty state.

Figure 4 shows the status of the identification after 2,150 data samples. At that instant, the fault $F_2$was already initiated and terminated, as we can see in Figures 4(a) and 4(b), and the fault F_4 was recently identified, as we can see in Figure 4(c).

Note, that after the very first faulty data sample read, AutoClass created a class of fault, automatically named "Class 1", which, in this case, represents the fault F_2, and all subsequent data samples from the referred stream, due to the spatial closeness in terms of the two features, were assigned to this class.

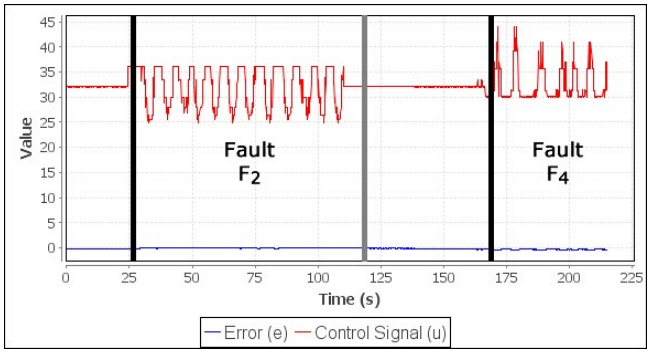
After the detection of the second fault, *AutoClass* was instantly able to identify that the newly arrived data samples did not belong to the existing class of fault. Until the required conditions were satisfied, all data outside the existing data cloud was classified as outliers, as we can see in Figure 4(c). Once there was enough close outliers to create a new cloud, *AutoClass* defined a new class of fault, automatically named "Class 2", which, in this case, represents the second fault stream, fault F_4as we can see in the same Figure.

The results obtained after reading of all 5,600 data samples, representing the sequential reading of 4 different fault streams, are shown in Figure 5.

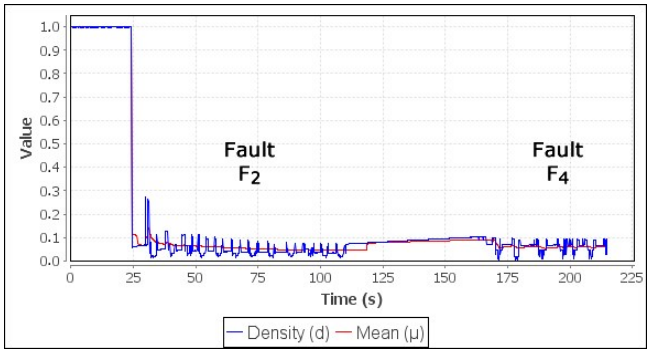
As can be seen in Figures 5 (a) and 5(b), fault F_4 was terminated around 275s, fault F_1 was initiated around 320s and terminated around 380s, and fault F_9 was initiated around 430s and terminated around 540s.

Also from Figure 5 we can see that Class 1 represents Faults 1 and 2; Class 2 represents Fault 4 and Class 3 – Fault 9, but this is extracted information autonomously and there is no need to provide this information a priory or train AutoClass in order this to be later identified. This is the major difference of the newly proposed fully unsupervised autonomous FDI approach and all existing so far which is demonstrated here based on a Laboratory based real plant with induced faults.

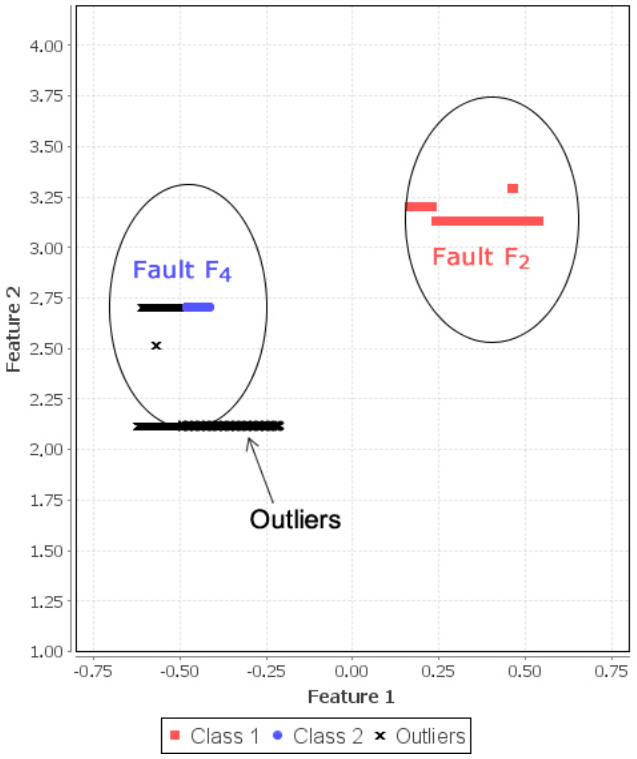
Note, in Figure 5(c) that the faults F_1 and F_2, which were not read sequentially in the experiment, were classified in the same data cloud which after the automatic labelling becomes a class, "Class 1", and activated by the same inference rule. This is entirely logical/expected, because they actually belong to the same fault type, although with different levels/amplitudes. Faults F_4 and F_9, represented by the labels "Class 2" and "Class 3", respectively, are also clearly indicated in the same figure.



(a) Detection stage – Input signals



(b) Detection stage – Density



(c) Identification stage – system status for k = 2,150

Fig. 4. : Fault detection and identification

It is important to highlight that, even though the system was able to distinguish faults F_1 and F_2, which are positive offset of the actuator, from fault F_4, which is negative offset of the actuator, they are still close together, because both faults concern the actuator. Note also, that faults F_2 and F_4 are also close to each other in terms of Feature 2, albeit one is negative Feature 1, while the other is positive. Fault F_9, on the other hand, concern structural changes and, in Figure 5(c), is further from faults F_1 and F_2, but, since leakage is logically closer to a negative change, F_9 is close to F_4. The fuzzy rule based classifier autonomously generated from the data stream which consists of 5,600 data samples, is presented as follows:

$$\Re^1: \textbf{IF } \left(\vec{x} \sim \aleph^1\right) \textbf{ THEN } (\text{"}Class\ 1\text{"})$$
$$\Re^2: \textbf{IF } \left(\vec{x} \sim \aleph^2\right) \textbf{ THEN } (\text{"}Class\ 2\text{"})$$
$$\Re^3: \textbf{IF } \left(\vec{x} \sim \aleph^3\right) \textbf{ THEN } (\text{"}Class\ 3\text{"})$$

with

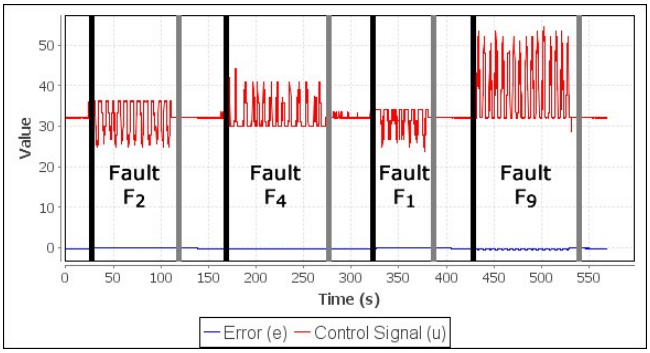
$$\aleph^1: c_1 = [0.416, 3.316] \text{ and } ZI_1 = [0.251, 0.756]$$
$$\aleph^2: c_2 = [-0.513, 2.706] \text{ and } ZI_2 = [0.250, 0.601]$$
$$\aleph^3: c_3 = [-0.416, 1.491] \text{ and } ZI_3 = [0.197, 0.451]$$

where c_i is the focal point and Z_i is the zone of influence of the cloud.

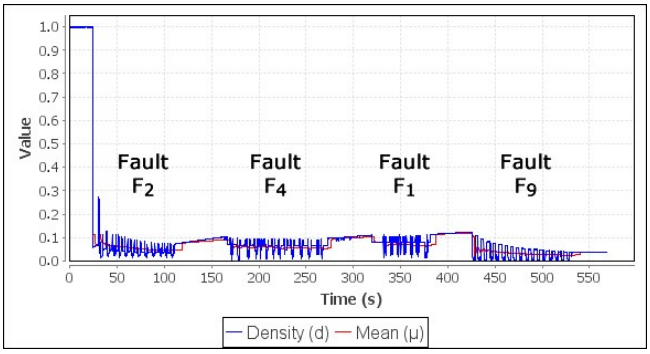
The fault identification procedure is quite unique in the sense that it autonomously and in a completely unsupervised manner (automatic labels) identifies the types of faults. Therefore, it is difficult to compare this new approach with any existing alternative approach directly.

IV. Conclusion

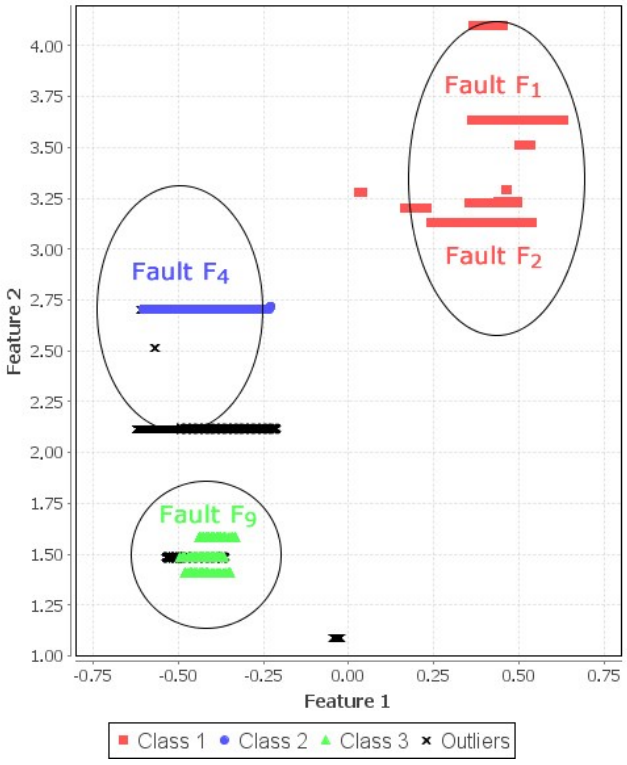
A two-stage FDI approach is presented in this paper. With a detection algorithm based on RDE and an identification procedure based on the recently introduced *AutoClass*, the proposed system is able to perform FDI on-line, starting "from scratch", since the very first data sample acquired, **in a fully unsupervised manner**. The proposed algorithm does not require any *a priori* information about the plant, its mathematical models, complex user-defined parameters or thresholds. This proposal differs from the traditional approaches as it works with the concept of data clouds, which are structures with no specific shape, boundaries, centre, parametric function to describe them and yet they are represented by an aggregated measure (data density). A number of experiments with different fault data streams were performed on-line using a real industrial hydraulic pilot plant. The results have shown a high rate of success on both detection and classification stages, with a very limited computational effort. When compared with a well-known fault detection technique, the proposed algorithm demonstrated considerable superior results, in terms of both global and individual experiments.



(a) Detection stage – Input signals



(b) Detection stage – Density



(c) Identification stage – system status after the reading of all 5,600 data samples

Fig. 5. Fault detection and identification

REFERENCES

[1] Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N., 2003. "A review of process fault detection and diagnosis, Part I: Quantitative model-based methods". Computers & Chemical Engineering, vol. 27, 293 – 311.

[2] Chen, W., Saif, M., 2007. "Observer-based strategies for actuator fault detection, isolation and estimation for certain class of uncertain nonlinear systems". Control Theory Applications, IET 1, 1672–1680.

[3] Anwar, S., Chen, L., 2007. "An analytical redundancy-based fault detection and isolation algorithm for a road-wheel control subsystem in a steer-by-wire system". IEEE Transactions on Vehicular Technology, vol. 56, 2859–2869.

[4] El-Shal, S., Morris, A., 2000. "A fuzzy expert system for fault detection in statistical process control of industrial processes". IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 30, 281–289.

[5] Bernieri, A., Betta, G., Liguori, C., 1996. "On-line fault detection and diagnosis obtained by implementing neural algorithms on a digital signal processor". IEEE Transactions on Instrumentation and Measurement, vol. 45, 894–899.

[6] Angelov, P., 2012. Autonomous Learning Systems: From Data to Knowledge in Real Time. John Willey and Sons.

[7] Cook, G., Maxwell, J., Barnett, R., Strauss, A., 1997. "Statistical process control application to weld process". IEEE Transactions on Industry Applications, vol. 33, 454–463.

[8] Angelov, P., Zhou, X., 2008. "Evolving fuzzy-rule-based classifiers from data streams", IEEE Transactions on Fuzzy Systems, vol. 16, 1462–1475.

[9] Angelov, P., Baruah, R.D., Andreu, J., 2011. "Simpl_eClass: simple potential-free evolving fuzzy rule-based on-line classifiers", in: Proceedings of 2011 IEEE International Conference on Systems, Man and Cybernetics, SMC 2011, Anchorage, Alaska, USA, 7-9 Oct, 2011, IEEE. pp. 2249–2254.

[10] Lemos, A., Caminhas, W., Gomide, F., 2013. "Adaptive fault detection and diagnosis using an evolving fuzzy classifier". Information Sciences, vol. 220, 64 – 85.

[11] Angelov, P., Yager, R., 2012. "A new type of simplified fuzzy rule-based systems". International Journal of General Systems, vol. 41, 163–185.

[12] Angelov, P., 2012. "Anomalous system state identification", patent GB1208542.9, priority date 15 may 2012.

[13] Kolev, D., Angelov, P., Markarian, G., Suvorov, M., Lysanov, S., 2013. "ARFA: Automated real-time flight data analysis using evolving clustering, classifiers and recursive density estimation", in: Proceedings of the IEEE Symposium Series on Computational Intelligence SSCI-2013, Singapore. pp. 91–97.

[14] DeLorenzo, 2009. DL 2314BR – "Didactic process control pilot plant". Catalog. DeLorenzo Italy. Italy.[12] Costa, B., Skrjanc, I., Blazic, S., Angelov, P., 2013. "A practical implementation of self-evolving cloud-based control of a pilot plant", in: Proceedings of 2013 IEEE International Conference on Cybernetics, Lausanne, Switzerland, pp. 7–12.

[15] Costa, B., Skrjanc, I., Blazic, S., Angelov, P., 2013. "A practical implementation of self-evolving cloud-based control of a pilot plant", in: Proceedings of 2013 IEEE International Conference on Cybernetics, Lausanne, Switzerland, pp. 7–12.

[16] Costa, B., Bezerra, C., Guedes, L., 2012. "A multistage fuzzy controller: Toolbox for industrial applications", in: Proceedings of the 2012 IEEE International Conference on Industrial Technology (ICIT), pp. 1142–1147.

[17] Costa, B., Bezerra, C.G., Guedes, L.A., 2010. "Java fuzzy logic toolbox for industrial process control", in: Proceedings of the 2010 Brazilian Conference on Automatics (CBA), Brazilian Society for Automatics (SBA), Bonito-MS, Brazil, pp. 207–214.

[18] Liu, J., Lim, K.W., Ho, W.K., Tan, K.C., Tay, A., Srinivasan, R., 2005. "Using the OPC standard for real-time process monitoring and control". IEEE Software, vol. 22, 54–59.