Tendências em Matemática Aplicada e Computacional, **19**, N. 2 (2018), 327-345
© 2018 Sociedade Brasileira de Matemática Aplicada e Computacional
www.scielo.br/tema
doi: 10.5540/tema.2018.019.02.0327

A New Scheme for Fault Detection and Classification Applied to DC Motor

L.I. SANTOS[1], R.M. PALHARES[2], M.F.S.V. D'ANGELO[3*], J.B. MENDES[3], R.R. VELOSO[3] and P.Y. EKEL[4]

Received on October 19, 2017 / Accepted on March 12, 2018

ABSTRACT. This study presents an approach for fault detection and classification in a DC drive system. The fault is detected by a classical Luenberger observer. After the fault detection, the fault classification is started. The fault classification, the main contribution of this paper, is based on a representation which combines the Subctrative Clustering algorithm with an adaptation of Particle Swarm Clustering.

Keywords: Fault Detection and Classification 1, Luenberger Observer 2, Particle Swarm Clustering 3.

1 INTRODUCTION

Fault detection and analysis is a very important strategy that is commonly employed in the industry with the purpose of allowing a cost-effective maintenance policy, keeping productivity standards and ensuring safety. The fault analysis gives support for the design of corrective actions, system redundancies, and safety policies in order to mitigate the effects of a fault [19]. In this paper, a fault diagnosis procedure is divided into two tasks: i) fault detection, indicating the occurrence of some fault in a monitored system; and ii) fault classification, establishing the type and/or location of the fault.

The literature presents several classes of strategies to deal with fault detection and isolation (FDI) [7]. These strategies can be, in general, divided in approaches based on quantitative models [36] and on qualitative models [34], [35].

Considering the quantitative model-based approaches (used for fault detection), many works with different emphases have been published over the past years. Among them, the main approaches focus on knowledge of mathematical models of the plant, and are based on observers

*Corresponding author: Marcos Flávio Silveira Vasconcelos D'Angelo – E-mail: marcos.dangelo@unimontes.br
[1]IFNMG Campus Montes Claros – Rua Dois, 300, Village do Lago I, 39404-058, Montes Claros - MG - Brasil. E-mail: laercio.santos@ifnmg.edu.br
[2]Departamento de Engenharia Eletrônica, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, 31270-901, Belo Horizonte - MG - Brasil. E-mail: rpalhares@ufmg.br
[3]Departamento de Ciências da Computação - UNIMONTES, Av. Rui Braga, sn, Vila Mauricéia, 39401-089, Montes Claros - MG - Brasil. E-mail: marcos.dangelo@unimontes.br; joao.mendes@unimontes.br; rene.veloso@unimontes.br
[4]Programa de Graduação em Engenharia Elétrica, Pontifícia Universidade Católica de Minas Gerais, Av. Dom Jose Gaspar, 500, 30535-610, Belo Horizonte, Brasil. E-mail: pekel@superig.com.br

[7], [4], [32], [31], [8], [28]. On the other hand, considering the qualitative model-based approaches (used for fault classification), focusing on the pattern analysis of the historic process data, the main related approaches are: signed directed graphs [24], [9], [2], fault trees [16], fuzzy systems [17],[29], [15], qualitative trend analysis [25], [18], [14], [13], mutual information [38], neural networks [3], [11] (neural networks also can be used as observer [33], [26]), artificial immune systems [22], [23], [30], Bayesian networks [39], [37] and the combination of techniques [21], [12].

In this paper, an approach for fault detection (first step) and classification (second step) is presented. The classical Luenberger observer is used in the first step, and this information is used for the classification system start. The idea of the second step, the main contribution of this paper, is to deal with the fault classification in a new way, using an adaptation of Particle Swarm Clustering. To illustrate the efficiency of the proposed methodology, the problem of fault detection in a DC Motor Benchmark Model [5] has been solved. An overview of the FDI framework proposed in this paper is illustrated in Figure 1.

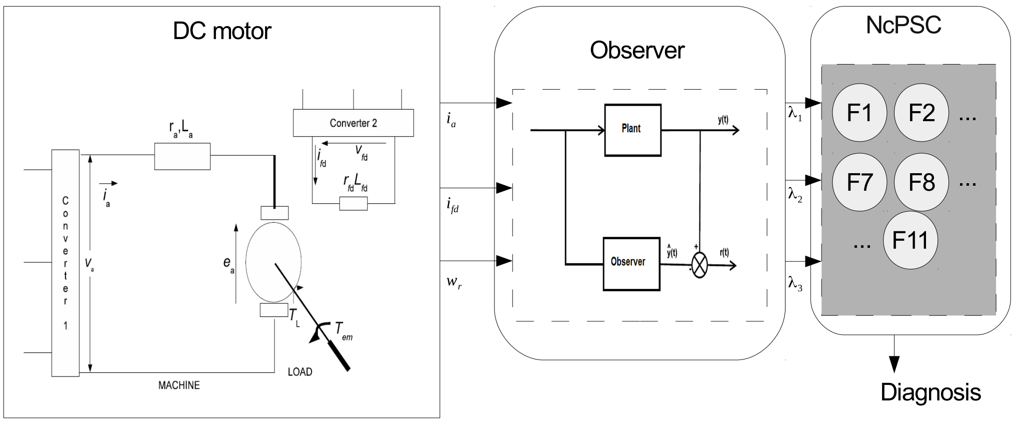

Figure 1: Framework for fault detection and classification.

1.1 Contributions

In this paper is proposed an algorithm, adapted from *cPSC* ([27]) and denoted as *New cPSC (NcPSC)*, which optimizes the hit rate and the total number of groups (classes) of a data set. It is a supervised algorithm which combines the following functionalities:

- A routine, detailed in subsection 3.2, was developed for generating the initial particle set;
- While the *cPSC* algorithm implements the cosine distance for computing the similarity measure, the *NcPSC* uses the Euclidean distance for computing the similarity between a particle and the input data set.

- A particle growing procedure, presented in Algorithm 2, was developed;
- A particle stagnation mechanism, described in Algorithm 3, was implemented.

It is important to note that the proposed functionalities may be adapted in several metaheuristics.

Paper organization. Section 2 presents and analyzes the DC Motor modeling considering the case of different types of faults and the observer design for fault detection. Section 3 describes the adaptive methodology based on Particle Swarm Clustering, the main contribution of this paper, for fault classification. Section 4 shows the new proposed approach applied to the DC motor fault classification problem. Finally, section 5 presents the concluding remarks.

2 LUENBERGER OBSERVER DESIGN FOR DC MOTOR

The DC motor benchmark model, evaluated in [5], is descripted as a drive system which consists of two power supplies, controlled static converters, a DC motor and a mechanical load. The system can be represented as shown in Figure 2.

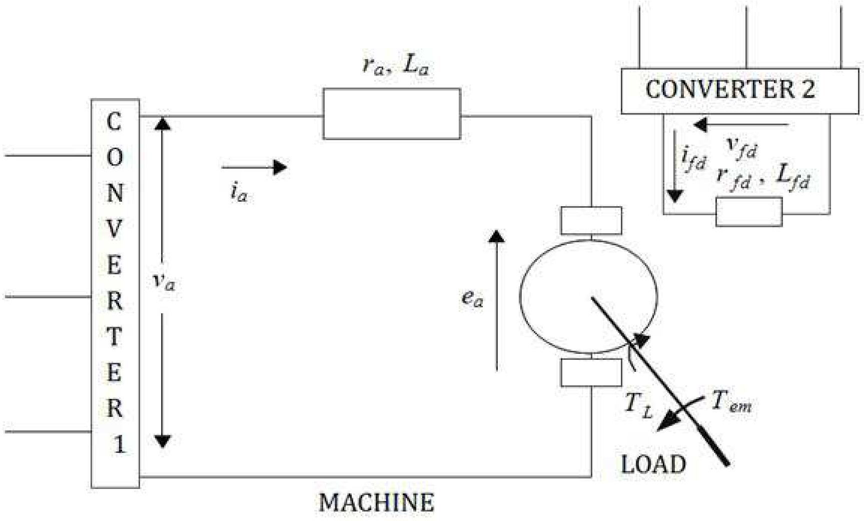

Figure 2: Representation of the DC drive system.

In Figure 2, each variable as the following:

- v_a is the armature voltage,
- v_{fd} is the field voltage,
- i_a is the armature current,
- i_{fd} is the field current.

- ω_r represents mechanical rotation speed in rad/s,
- r_a is the armature resistance,
- L_a is the armature inductance,
- r_{fd} is the field resistance,
- L_{fd} is the field inductance,
- e_a is the counter electromotive force and is dependent of L_{afd} (mutual inductance).

A discrete model for the DC motor is given in (2.1) and presented in [5], where $i_a = x_1$, $i_{fd} = x_2$ and $\omega_r = x_3$. All state variables are measured, i.e., $y(t) = \mathbf{I}x(t)$.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & a_2(k) & 0 \\ 0 & a_3 & 0 \\ a_4(k) & 0 & a_5 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & d_1 \end{bmatrix} \begin{bmatrix} v_a(k) \\ v_{fd}(k) \\ T_L \end{bmatrix} \tag{2.1}$$

As the following: $a_1 = a_1(r_a, L_a) = e^{-\frac{r_a}{L_a}h}$;
$a_3 = a_3(r_{fd}, L_{fd}) = e^{-\frac{r_{fd}}{L_{fd}}h}$;
$a_5 = a_5(B_m, J_m) = e^{-\frac{B_m}{J_m}h}$;
$a_2(k) = a_2(r_a, L_a, r_{fd}, L_{fd}, x_3(k)) = \frac{1}{r_{fd}La - r_aL_{fd}}[L_{afd}L_{fd}(a_3 - a_1)x_3(k) + r_aL_{fd}a_1 - r_{fd}L_aa_3]$;
$a_4(k) = a_5(B_m, J_m, x_2(k)) = L_{afd}\frac{(1-a_5)}{B_m}x_2(k)$;
$b_1 = b_1(r_a, L_a) = \frac{1-a_1}{ra}b_2 = b_2(r_{fd}, L_{fd}) = \frac{1-a_3}{r_{fd}}d_1$;
$d_1 = d_1(B_m, J_m) = -\frac{1-a_5}{B_m}$.

where B_m is the coefficient of viscous friction, J_m is the moment of inertia and T_L is the mechanical torque of the load.

Faults on the DC drive system may occur in: actuators (armature and field converters), plant or process (DC Machine) and sensors (current meters and speed). The faults on actuators are: armature converter disconnection, field converter disconnection, armature converter short circuit and field converter short circuit. The faults in DC Machine are: armature turns short-circuit, field turns short-circuit, ventilation system fault and bearing lubrication fault. The faults on sensors are: armature current sensor fault, field current sensor fault and machine speed sensor fault. Considering these fault types (see Table 1), the complete model is described in (2.2). For more details see the full modeling in [5] and [30].

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} k_{aa}a_1{}^f & k_{aa}a_2{}^f(k) & 0 \\ 0 & k_{afd}a_3{}^f & 0 \\ a_4{}^f(k) & 0 & a_5{}^f \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} b_1{}^f & 0 & 0 \\ 0 & b_2{}^f & 0 \\ 0 & 0 & d_1{}^f \end{bmatrix} \begin{bmatrix} k_{aa}k_{cca}v_a(k) \\ k_{afd}k_{ccfd}v_{fd}(k) \\ T_L \end{bmatrix}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \end{bmatrix} = \begin{bmatrix} k_{i_a}{}^f & 0 & 0 \\ 0 & k_{i_{fd}}{}^f & 0 \\ 0 & 0 & k_{\omega_r}{}^f \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} \tag{2.2}$$

where: $r_a{}^f = k_{ca}{}^{r_a}k_{fv}{}^{r_a}r_a$, $L_a{}^f = k_{ca}{}^{L_a}L_a$, $r_{fd}{}^f = k_{cfd}{}^{r_{fd}}k_{fv}{}^{r_{fd}}r_{fd}$, $L_{fd}{}^f = k_{cfd}{}^{L_{fd}}L_{fd}$, $B_m{}^f = k_{fl}B_m$.

Table 1: Summary of DC motor system faults.

Fault Index	Fault Type	Affected Variables	Fault Indicator Parameters	Parameter Values
1	Armature converter disconnection	$i_a = 0$	k_{aa}	$\{0,1\}$
2	Field converter disconnection	$i_{fd} = 0$	k_{afd}.	$\{0,1\}$
3	Armature converter short circuit	$v_a = 0$	k_{cca}	$\{0,1\}$
4	Field converter short circuit	$v_{fd} = 0$	k_{ccfd}.	$\{0,1\}$
5	Armature turns short-circuit	r_a and L_a	$k^{r_a}{}_{ca}$ and $k^{L_a}{}_{ca}$	$[0,1]$
6	Field turns short-circuit	r_{fd} and L_{fd}	$k^{r_{fd}}{}_{cfd}$ and $k^{L_{fd}}{}_{cfd}$	$[0,1]$
7	Ventilation system fault	r_a and r_{fd}	$k_{fv}{}^{r_a}$ and $k_{fv}{}^{r_{fd}}$	$[1,\infty)$
8	Bearing lubrication fault	B_m	k_{fl}	$[1,\infty)$
9	Armature current sensor fault	i_a	$k_{i_a}{}^f$	$\{0,1\}$
10	Field current sensor fault	i_{fd}	$k_{i_{fd}}{}^f$	$\{0,1\}$
11	Machine speed sensor fault	ω_r	$k_{\omega_r}{}^f$	$\{0,1\}$

Simulations were made to show some situations of DC motor operation mode. The normal operation of the machine and the four faults in the actuators were simulated and the values of the variables are shown in Figs 3 to 7.

We obtained 4000 points (corresponding to a test of 4 seconds, as each 1s corresponds to 1000 points) and each fault occurs 2s after the beginning of the tests. In these scenarios, it is possible to see which variables are affected after the occurrence of faults.

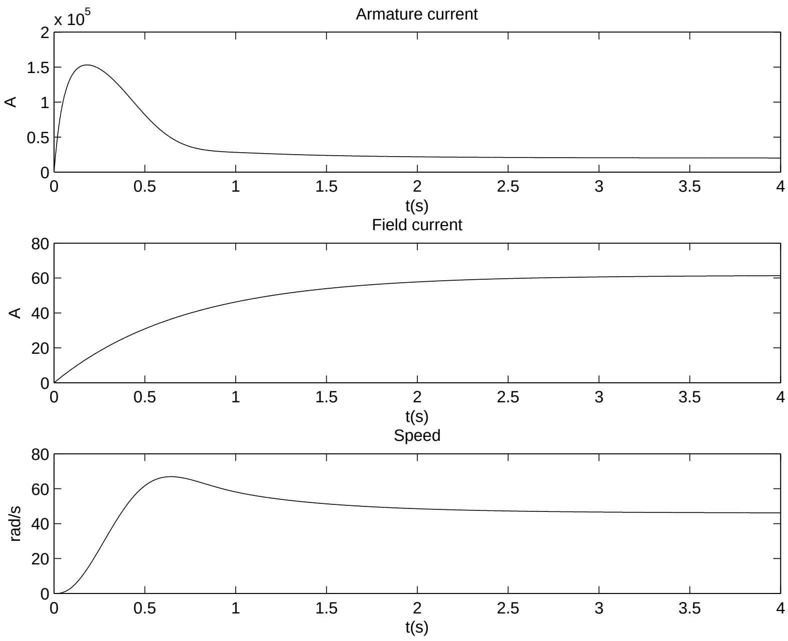

Figure 3: Simulation of the DC motor in normal scenario.

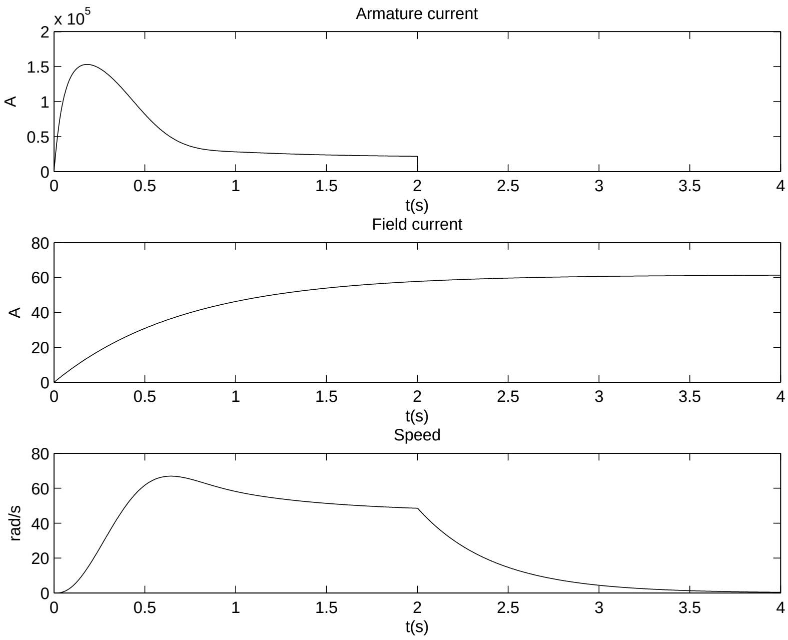

Figure 4: Simulation of the DC motor with disconnection of the armature converter.

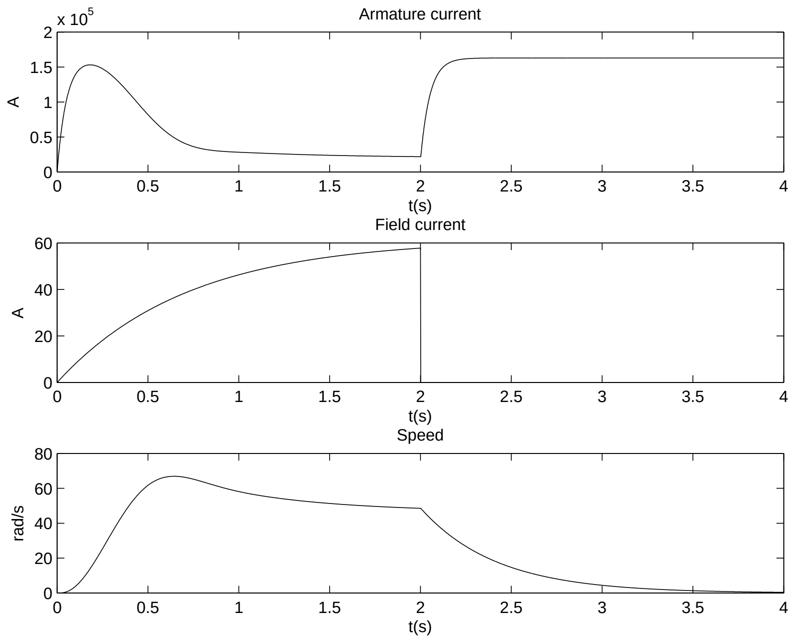

Figure 5: Simulation of the DC motor with disconnection of the field converter.

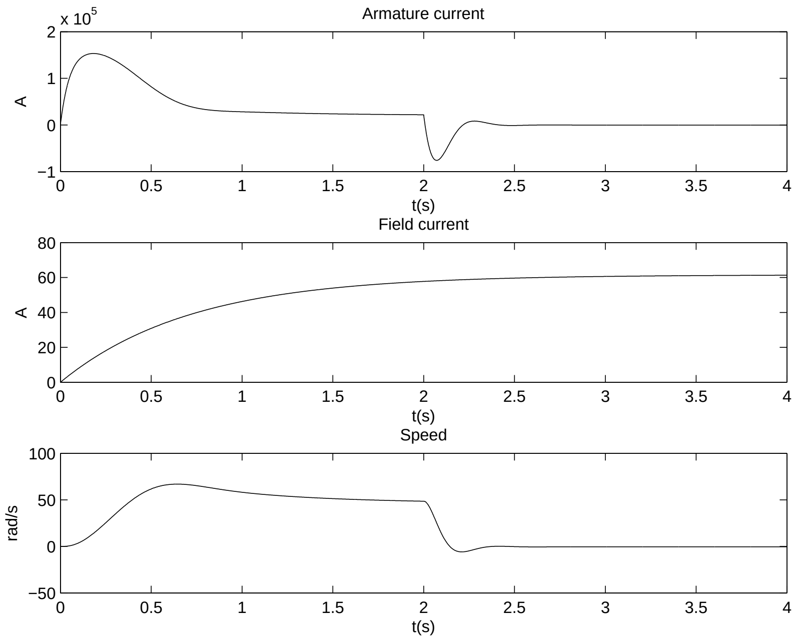

Figure 6: Simulation of the DC motor with short circuit in the armature converter.

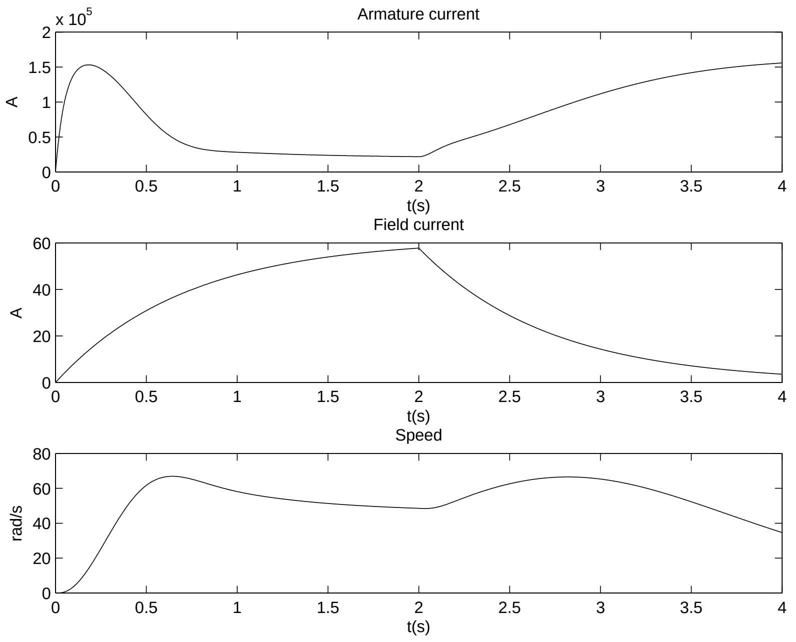

Figure 7: Simulation of the DC motor with short circuit in the field converter.

2.1 Observer-based fault detection

The aim of the observer-based fault detection method is to generate a residual, used for fault indication. Considering the state space model:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.3}$$

$$y(t) = Cx(t) \tag{2.4}$$

where $u(t)$ is the input ,$x(t)$ is the state and $y(t)$ is the output.

The observer can be designed as follows to provide the system observability:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + Le(t) \tag{2.5}$$

$$e(t) = y(t) - C\hat{x}(t) \tag{2.6}$$

$\hat{x}$ is the estimated system state, L is the matrix of the observer feedback gains that is designed to provide the required performance of the observer and $e(t)$ is the output error.

Replacing (2.6) in (2.5):

$$\dot{\hat{x}}(t) = [A - LC]\hat{x}(t) + Bu(t) + Ly(t) \tag{2.7}$$

The state error is given by:

$$\dot{\tilde{x}}(t) = \dot{x}(t) - \dot{\hat{x}}(t) \tag{2.8}$$

Replacing (2.3) and (2.7) in (2.8):

$$\dot{\tilde{x}}(t) = [A - LC]\tilde{x}(t) \tag{2.9}$$

The observer design results in:

$$\lim_{t \to \infty} \tilde{x}(t) = 0$$

Considering the matrices A and C of the discrete system (2.1) to $A - LC$, and imposing pole placement to gain L observer, of the form:

$$A - LC = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

the observer gain is obtained by:

$$L = \begin{bmatrix} a_1 - \lambda_1 & a_2(k) & 0 \\ 0 & a_3 - \lambda_2 & 0 \\ a_4(k) & 0 & a_5 - \lambda_3 \end{bmatrix}$$

Note: for discrete systems, the pole placement of $A - LC$ is inside unit circle, $|\lambda_i| < 1$.

Figure 8 shows residuals for observer gain with $\lambda_1 = \lambda_2 = \lambda_3 = 0.3$ for armature converter disconnection.

3 ADAPTIVE APPROACH FOR FAULT CLASSIFICATION

In this section, we present the *NcPSC* algorithm which is an adaptation of the *cPSC* algorithm. In [6] a new unsupervised mechanism based on the particle swarm optimization (PSO)[20] algorithm, called *Particle Swarm Clustering - PSC*, is proposed for solving clustering problems. The PSC creates a population of individuals (particles), initially randomly, where each particle represents a candidate solution which moves through the search space. Specifically for data clustering problems, a particle identifies a cluster (or a prototype for a cluster). However, having to pass the value of k (cluster number) to the algorithm may not be interesting. Thus, [27] developed the $cPSC(ConstructivePSC)$ (an adaptation of PSC) algorithm which identifies automatically the total of groups in a set. The *cPSC* implements a dynamic mechanism for particles growthing (cloning) and pruning to compute the total of groups in a set. The cloning mechanism works as follows: i) Firstly, it computes how many data from input data set each particle represents. If it is equal zero (or a fixed iteration number) the particle is removed from the swarm. Otherwise, it is a candidate particle for cloning; ii) Also, if the similarity between a particle x and an input data is greater than a constant ε, x will be cloned. The cosine metric is used to calculate the similarity between a particle and an input data. The clone particle is positioned between the cloned particle and the most similar input data item. Finally, the *NcPSC* algorithm The NcPSC introduces some functionality in the cPSC to insert supervised characteristics, since the problem deal with classification. Next we detail each functionality implemented by *NcPSC* algorithm.

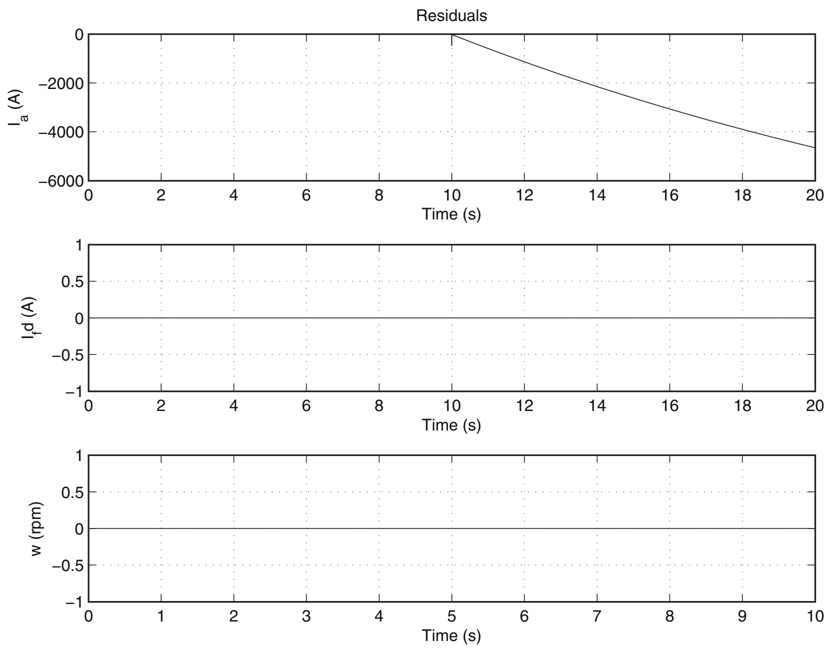

Figure 8: Residual for observer gain with $\lambda_1 = \lambda_2 = \lambda_3 = 0.3$.

3.1 Concentration level and hit rate

The *NcPSC* implements a mechanism for growing and pruning particles based on its concentration level and hit rate. The concentration level of a particle p (p_{cl}) is computed as:

Algorithm 1: Computing_CL (p, X).

```
1 begin
2     $p_{cl} \leftarrow 0$;
3     for each Input_Data $i_x \in X$ do
4         if $p$ is the most similar particle to $i_x$ then
5             $p_{cl} \leftarrow p_{cl} + 1$; Associate $i_x$ to $p$; $p.I_x \leftarrow p.I_x \cup i_x$;
6         end
7     end
8 end
```

$p.I_x$ is a list associated to every particle p used for storing input data.

3.2 Generating initial particle set

The initial population plays a key role in terms of convergence rate and may affect the success of an EA in finding high quality or satisfactory solutions [1]. For this reason, the *NcPSC* algorithm also implements a simple mechanism for generating initial particles set: Firstly it generates a swarm with *N* particles. The initial position of each particle is determined using the following mechanism: It calculates *N* centroids for the training data set using the *subtractive clustering (SC)* method[10].

3.3 Particle growing mechanism

The growing procedure implemented by *NcPSC* was proposed for generating new particles and it is based on concentration level and hit rate values of a particle. The proposed method, detailed in Algorithm 2, clones a particle *p*, resulting in a clone particle *p*'. A particle *p* is cloned if its concentration level ($p.cl$) value is greater than ε_1 and its hit rate ($p.hh$) value is less than γ_1 - it means that *p* concentrates many input data from more than one class (ε_1 and γ_1 are parameters of *NcPSC*). It is noted that a particle with low hit rate worsens (or deteriorates) the final hit rate of the swarm.

Finally, a simple mutation is performed over a dimension (chosen randomly) of the clone particle. It is expected that clone particles present better hit rates values.

Algorithm 2: Particle_growing (p, ε_1, γ_1).

```
begin
    if $p.cl > \varepsilon_1$ and $p.hh < \gamma_1$ then
        p' ← p.clone();
        RETURN p'.mutation();
    end
end
```

3.4 Particle stagnation

A mechanism for particle stagnation was proposed for *NcPSC* because when choosing the most similar particle to the input data, generally more particles move to the crowded regions of the search space. If these crowded regions concentrate several classes, the hit rate values of the attracted particles may be reduced.

The stagnation routine, presented in Algorithm 3, checks the concentration level and hit rate values of a particle. It means if a particle concentration level value is above ε_2 and its hit rate value is above γ_2, the particle will not move in current *NcPSC* iteration (ε_2 and γ_2 are parameters of *NcPSC*).

Algorithm 3: Particle_stagnation (p, ε_2, γ_2).

```
1 begin
2     $p.stagned \leftarrow False$;
3     if $p.cl > \varepsilon_2$ and $p.hh > \gamma_2$ then
4         $p.stagned \leftarrow True$;
5     end
6     RETURN $p$;
7 end
```

3.5 General structure of the *NcPSC* algorithm

The proposed *NcPSC*, detailed in algorithm 4, is an adaptation of *cPSC* algorithm and it has a set of input parameters presented in the following:

- *Number of particles*: Variable;
- *Input Data*: Dataset for clustering;
- $\varepsilon_1 = 0.60, \varepsilon_2 = 0.90$, $\gamma_1 = 21$ and $\gamma_2 = 15$
- $\omega = 0.20$;
- *Stop condition*: 20 (twenty) iterations.

The *NcPSC* works as: Firstly, it generates a swarm with N particles. In the next, the following steps are executed in an ordered and repetitive manner until a termination criterion is found:

- All particles which concentration level value greater than ε_2 and hit rate value greater than γ_2 are marked as stagnated (line 6);
- It identifies the most similar particle from the swarm for each item in the input data. This particle is denoted as the winner particle (X_w). The following operations are applied over each winner particle (It is *moved*):
  - Its velocity is updated through expression 3.1:

$$\begin{aligned} V_w(t+1) = \omega V_w(t) + \varphi_1 \otimes (PBest_{wj}(t) - X_w(t)) + \\ \varphi_2 \otimes (GBest_j(t) - X_w(t)) + \varphi_3 \otimes (Y_j(t) - X_w(t)) \end{aligned} \tag{3.1}$$

    where $V_w(t)$ is the particle velocity in iteration t, ω is the inertia component and φ_1, φ_2 and φ_3 are random number vectors. While *PBest* denotes the best (most similar) position of a particle in relation to an input data, *GBest* identifies the best (most similar) particle position (from all *PBest* particles) in relation to an input data.

- Its position is updated using equation:

$$X_w(t+1) = X_w(t) + V_w(t+1) \tag{3.2}$$

- Its p_best and p_best components are updated;

- It removes particles from swarm: All particles with concentration level equal zero are removed from the swarm;
- It clones particles from swarm: All particles with concentration level greater than ε_1 and with hit rates less than γ_1 are cloned.

Algorithm 4: NcPSC_algorithm (ID, ε_1, γ_1, ε_2, γ_2).

```
1  begin
2      S ← generate_initialSwarm(ID);
3      while stop condition not satisfied do
4          for each particle $p \in S$ do
5              $p.cl$ ← COMPUTE concentration level of $p$; $p.hh$ ← COMPUTE hit rate of $p$;
                 Particle_stagnation($p, \varepsilon_2, \gamma_2$);
6          end
7          for each item $Y_j \in ID$ do
8              Find $X_w \in S$ such that $X_w$ is the most similar particle to $Y_j$;
9              if NOT ($X_w.stagned$) then
10                 Update $V_w$ using equation 3.1; Update $X_w$ using equation 3.2; Update
                     $Pbest_{wj}$ e $Gbest_j$;
11             end
12         end
13         S.remove_particles();
14         for each particle $p \in S$ do
15             Particle_growing($p, \varepsilon_1, \gamma_1$);
16         end
17     end
18 end
```

4 PROPOSED APPROACH APPLIED TO THE FAULT CLASSIFICATION IN DC MOTOR

To investigate the main functionalities of the proposed methodology, we analyze the performance of several algorithms which implement some functionalities (isolated or combined) of the *NcPSC* algorithm. The developed algorithms with their implemented functionalities are:

- *cPSC*: The original *cPSC* algorithm proposed in [27];
- $cPSC_{ED}$: A version of *cPSC* algorithm which implements the Euclidean distance for similarity measure;
- $cPSC_{PG}$: An adaptation of *cPSC* algorithm which develops only the *Particle_growing* routine;
- $cPSC_{EP}$: A Version of $cPSC_{ED}$ which combines both the Euclidean distance for similarity measure and the *Particle_growing* mechanism;
- $cPSC_{PS}$: An adaptation of $cPSC_{EP}$ algorithm which develops the *Particle_stagnation* mechanism;
- *NcPSC*: The proposed *NcPSC* algorithm which implements all mentioned functionalities.

To assess the performance of the developed algorithms they were applied to a clustering problem of a data set with 252 operation points representing 11 (eleven) fault points (Problem with 11 classes). Each algorithm was executed 100 times and each execution performed 20 (twenty) iterations. Some data (Larger, Smaller and average values of hit rate and total of particles) were computed for each algorithm and the obtained results are shown in Table 2.

Table 2: Performance results (Hit rate and Number of particles) for the problem generated by each algorithm.

Description	Hit Rate (%)			Number of particles		
	Larger	Smaller	Average	Larger	Smaller	Average
cPSC	88.10	41.67	73.19	172	9	54.41
$cPSC_{ED}$	83.73	73.02	80.99	155	76	125.76
$cPSC_{PG}$	90.48	65.48	81.19	25	16	20.66
$cPSC_{EP}$	96.04	76.98	86.37	24	15	19.90
$cPSC_{PS}$	91.67	54.37	78.91	18	10	14.56
NcPSC	100	82.14	91.30	20	11	13.43

Some comments about the performance of the 06 algorithms are presented in the following:

- The $cPSC_{ED}$ and the *cPSC* algorithms produced the worst hit rate results. Their greater and average hit rate value are smaller than the values of the remaining algorithms. Also, their greater and average number of particles (classes) values are worse than the values generated by the others algorithms. In summary, the Euclidean distance mechanism is not sufficient to produce good results in solving the proposed clustering problem;

- When comparing the three algorithms ($cPSC_{PG}$, $cPSC_{EP}$ and $cPSC_{PS}$), it is noted that the $cPSC_{EP}$ produced better hit rate results than $cPSC_{PG}$ and $cPSC_{PS}$ algorithms. On the other hand, the $cPSC_{PS}$ produced best number of particles (near or equal to the number of classes) results than the other two algorithms ($cPSC_{PG}$ and $cPSC_{EP}$ algorithms). In addition, the number of classes found by $cPSC_{EP}$ is a little better than $cPSC_{PG}$;

- *NcPSC* was capable of providing a better clustering than all algorithms. Its hit rate (greater, smaller and average) values better than the others algorithms. Its number of particles (smaller and average) values are better than the results produced by the remaining algorithms. It is important to comment that the *NcPSC* algorithm found the exact number of classes of the proposed problem. Despite its greater number of classes value is greater than the value found by $cPSC_{PS}$ algorithm, its average value is closer to the real value than the average value produced by $cPSC_{PS}$ algorithm.

The results generated by the six algorithms are presented in a boxplot perspective shown in Figure 9. Firstly, the hit rate results, illustrated in Figure 9(a), evidence the good performance of the $cPSC_{PG}$, $cPSC_{EP}$, $cPSC_{PS}$ and *NcPSC* algorithms when compared with *cPSC* and $cPSC_{ED}$ methods.

Also, this perspective emphasizes the good results (both hit rate and particles number) presented by the *NcPSC* algorithm when compared to the others implementations. The lower boundary of its central box is above the upper boundary of the central box of all others implementations. Figure 9(a) reveals that $cPSC_{ED}$, $cPSC_{EP}$, $cPSC_{PS}$ and *NcPSC* (they developed the euclidian distance as similarity metric) have a more homogeneous behavior than *cPSC* and $cPSC_{EG}$ algorithms (both implemented the cossin distance as similarity metric). Furthermore, the symmetry is more pronounced in $cPSC_{EP}$ and *NcPSC* implementations. The results generated by $cPSC_{PS}$ algorithm, which only implements the particle stagnation mechanism, are worser than $cPSC_{EP}$ results. However, a version which implements all funcionalities (*NcPSC* algorithm) produced better results than all versions presented in this paper. Also, the total of particles found by each algorithm is presented in a boxplot style (See Figure 9(b)). Again, the figure illustrates the good results produced by algorithms $cPSC_{PG}$, $cPSC_{EP}$, $cPSC_{PS}$ and *NcPSC* and $cPSC_{PG}$, $cPSC_{EP}$, $cPSC_{PS}$ and *NcPSC* are more homogenic and generated better results than *cPSC* and $cPSC_{ED}$ algorithms. The total of particles (classes) found by both implementations are closer to the real context than the others algorithms.

5 CONCLUSION

In this paper, a new strategy for fault detection and classification is proposed. The strategy is based on the classical Luenberger observer, for fault detection, associated with an adaptive approach, for fault classification. The adaptive approach for fault classification, the main contribution of this paper, is based on the Particle Swarm Clustering algorithm. This methodology has been successfully applied to the fault detection and classification problem in a DC motor. The simulation results presented illustrate the effectiveness of the proposed method.

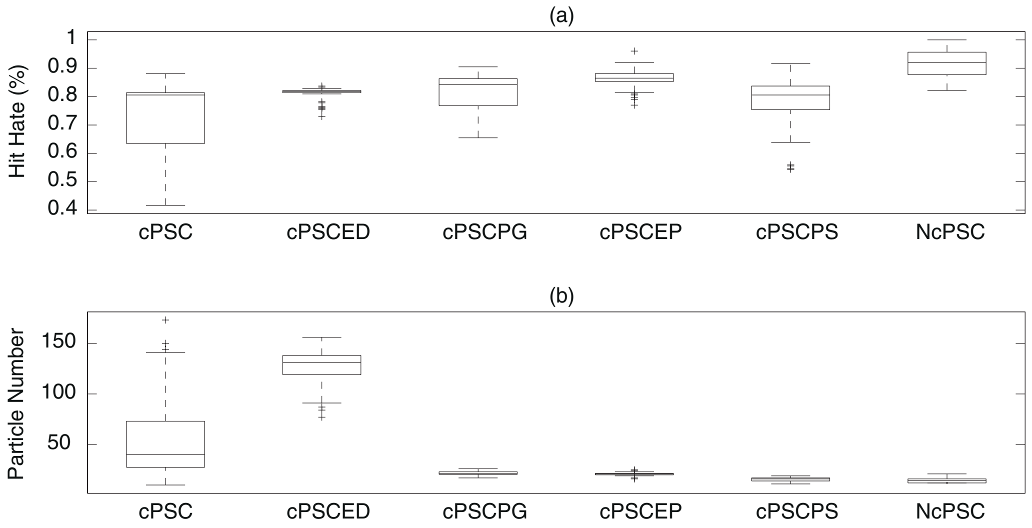

Figure 9: Boxplot illustrating the performance of the six implemented algorithms.

ACKNOWLEDGMENT

This work has been supported in part by the Brazilian agencies CNPq and FAPEMIG.

REFERENCES

[1] D. Bajer, G. Martinović & J. Brest. A population initialization method for evolutionary algorithms based on clustering and Cauchy deviates. *Expert Systems with Applications*, **60** (2016), 294–310.

[2] T. Boukhobza, F. Hamelin & S. Canitrot. A graph-theoretic approach to fault detection and isolation for structured bilinear systems. *International Journal of Control*, **81**(4) (2008), 661–678.

[3] J.M.F. Calado, J. Korbicz, K. Patan, R.J. Patton & J.M.G.S. da Costa. Soft computing approaches to fault diagnosis for dynamic systems. *European Journal of Control*, **7**(2-3) (2001), 248–286.

[4] W.M. Caminhas & R.H.C. Takahashi. Dynamic system failure detection and diagnosis employing sliding mode observers and fuzzy neural networks. In "Proceedings of the Joint 9th IFSA and 20th NAFIPS". Vancouver (2001), pp. 304–309.

[5] W.M. Caminhas & R.H.C. Takahashi. Dynamic system failure detection and diagnosis employing sliding mode observers and fuzzy neural networks. In "Joint nineth IFSA world congress and 20th NAFIPS international conference" (2001), pp. 304–309.

[6] L.N. Castro & S.C.M. Cohen. Data Clustering with Particle Swarms. *International Conference on Evolutionary Computation, Proceedings of World Congress on Computational Intelligence*, (2006), 1792–1798.

[7] J. Chen & R.J. Patton. "Robust model-based fault diagnosis for dynamic systems", volume 3. Springer Science & Business Media (2012).

[8] W. Chen & M. Saif. Observer-based strategies for actuator fault detection, isolation and estimation for certain class of uncertain nonlinear systems. *IET Control Theoty and Applications*, **1**(6) (2007), 1672–1680.

[9] H. Cheng, M. Nikus & S. Jamsa-Jounela. Fault diagnosis of the paper machine short circulation process using novel dynamic causal digraph reasoning. *Journal of Process Control*, **18**(7–8) (2008), 676–691.

[10] S.L. Chiu. Method and software for extracting fuzzy classification rules by subtractive clustering. *Biennial Conf. North American Fuzzy Information Processing Soc.*, (1996), 461–465.

[11] M.F.S.V. D'Angelo & P.P. Costa. Detection of shorted turns in the field winding of turbogenerators using the neural network MLP. In "Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics". Tucson (2001), pp. 1930–1935.

[12] M.F.S.V. D'Angelo, R.M. Palhares, L.B. Cosme, L.A. Aguiar, F.S. Fonseca & W.M. Caminhas. Fault detection in dynamic systems by a Fuzzy/Bayesian network formulation. *Applied Soft Computing*, **21** (2014), 647–653.

[13] M.F.S.V. D'Angelo, R.M. Palhares, R.H.C. Takahashi & R.H. Loschi. Fuzzy/Bayesian change point detection approach to incipient fault detection. *IET Control Theory & Applications*, **5**(4) (2011), 539–551.

[14] M.F.S.V. D'Angelo, R.M. Palhares, R.H.C. Takahashi, R.H. Loschi, L.M.R. Baccarini & W.M. Caminhas. Incipient fault detection in induction machine stator-winding using a fuzzy-Bayesian change point detection approach. *Applied Soft Computing*, **11**(1) (2011), 179–192.

[15] K.P. Detroja, , R.D. Gudi & S.C. Patwardhan. A possibilistic clustering approach to novel fault detection and isolation. *Journal of Process Control*, **16**(10) (2006), 1055–1073.

[16] Y. Dutuit & A. Rauzy. Approximate estimation of system reliability via fault trees. *Reliability Engineering & System Safety*, **87**(2) (2005), 163–172.

[17] S.M. El-Shal & A.S. Morris. A fuzzy expert system for fault detection in statistical process control of industrial processes. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, **30**(2) (2000), 281–289.

[18] K.F. Fong, A.P. Loh & W.W. Tan. A frequency domain approach for fault detection. *International Journal of Control*, **81**(2) (2008), 264–276.

[19] R. Isermann & P. Balle. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, **5**(5) (1997), 707–719.

[20] J. Kennedy & R. Eberhart. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks (1995), pp. 1942–1948.

[21] C.A. Laurentys, C.H.M. Bomfim, B.R. Menezes & W.M. Caminhas. Design of a pipeline leakage detection using expert system: A novel approach. *Applied Soft Computing*, **11**(1) (2011), 1057–1066.

[22] C.A. Laurentys, R.M. Palhares & W.M. Caminhas. Design of an artificial immune system based on Danger Model for fault detection. *Expert Systems with Applications*, **37**(7) (2010), 5145–5152.

[23] C.A. Laurentys, R.M. Palhares & W.M. Caminhas. A novel Artificial Immune System for fault behavior detection. *Expert Systems with Applications*, **38**(11) (2011), 6957–6966.

[24] M.R. Maurya, R. Rengaswamy & V. Venkatasubramanian. A signed directed graph-based systematic framework for steady-state malfunction diagnosis inside control loops. *Chemical Engineering Science*, **61**(6) (2006), 1790–1810.

[25] M.R. Maurya, R. Rengaswamy & V. Venkatasubramanian. Fault diagnosis using dynamic trend analysis: A review and recent developments. *Engineering Applications of Artificial Intelligence*, **20**(2) (2007), 133–146.

[26] K. Patan & T. Parisini. Identification of neural dynamic models for fault detection and isolation: the case of a real sugar evaporation process. *Journal of Process Control*, **15**(1) (2005), 67–79.

[27] A.K.F. Prior & L.N. Castro. cPSC: Um algoritmo de enxame construtivo para agrupamento de dados. In "Anais do XVIII Congresso Brasileiro de Automática" (2010), pp. 3300–3307.

[28] V. Puig, A. Stancu, T. Escobet, F. Nejjari, J. Quevedo & R. Patton. Passive robust fault detection using interval observers: Application to the DAMADICS benchmark problem. *Control Engineering Practice*, **14**(6) (2006), 621–633.

[29] J. Ragot & D. Maquin. Fault measurement detection in an urban water supply network. *Journal of Process Control*, **16**(9) (2006), 887–902.

[30] G.C. Silva, R.M. Palhares & W.M. Caminhas. Immune inspired Fault Detection and Diagnosis: A fuzzy-based approach of the negative selection algorithm and participatory clustering. *Expert Systems with Applications*, **39**(16) (2012), 12474–12486.

[31] R.H.C. Takahashi, R.M. Palhares & P.L.D. Peres. Discrete-time Singular Observers: $\mathscr{H}_2/\mathscr{H}_\infty$ Optimality and Unknown Inputs. *International Journal of Control*, **72**(6) (1999), 481–492.

[32] R.H.C. Takahashi & P.L.D. Peres. Unknown input observers for uncertain systems: A unifying approach. *European Journal of Control*, **5**(2–4) (1999), 261–275.

[33] F.J. Uppal, R.J. Patton & M. Witczak. A neuro-fuzzy multiple-model observer approach to robust fault diagnosis based on the DAMADICS benchmark problem. *Control Engineering Practice*, **14**(6) (2006), 699–717.

[34] V. Venkatasubramanian, R. Rengaswamy & S.N. Kavuri. A review of process fault detection and diagnosis – Part II: Qualitative models and search strategies. *Computers and Chemical Engineering*, **27**(3) (2003), 313–326.

[35] V. Venkatasubramanian, R. Rengaswamy, S.N. Kavuri & K. Yin. A review of process fault detection and diagnosis – Part III: Process history based methods. *Computers and Chemical Engineering*, **27**(3) (2003), 327–346.

[36] V. Venkatasubramanian, R. Rengaswamy, K. Yin & S.N. Kavuri. A review of process fault detection and diagnosis – Part I: Quantitative model-based methods. *Computers and Chemical Engineering*, **27**(3) (2003), 293–311.

[37] S. Verron, J. Li & T. Tiplica. Fault detection and isolation of faults in a multivariate process with Bayesian network. *Journal of Process Control*, **20**(8) (2010), 902–911.

[38] S. Verron, T. Tiplica & A. Kobi. Fault detection and identification with a new feature selection based on mutual information. *Journal of Process Control*, **18**(5) (2008), 479–490.

[39] B.G. Xu. Intelligent fault inference for rotating flexible rotors using Bayesian belief network. *Expert Systems with Applications*, **39**(1) (2012), 816–822.