Engineering Applications of Artificial Intelligence 79 (2019) 67–86

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

State space neural networks and model-decomposition methods for fault diagnosis of complex industrial systems☆

Belarmino Pulido [a,*,1,3], Jesús M. Zamarreño [b,c,2,3], Alejandro Merino [d,2], Anibal Bregon [a,1,3]

[a] Departamento de Informática, University of Valladolid, Valladolid, 47011, Spain
[b] Departamento de Ingeniería de Sistemas y Automática, University of Valladolid, Valladolid, Spain
[c] Institute of Sustainable Processes, University of Valladolid, Valladolid, Spain
[d] Departamento de Ingeniería Electromecánica, University of Burgos, Burgos, Spain

ARTICLE INFO

Keywords:
Intelligent fault diagnosis
Model-based diagnosis
State space neural networks
Data-driven modelling
Grey-box models

ABSTRACT

Reliable and timely fault detection and isolation are necessary tasks to guarantee continuous performance in complex industrial systems, avoiding failure propagation in the system and helping to minimize downtime. Model-based diagnosis fulfils those requirements, and has the additional advantage of using reusable models. However, reusing existing complex non-linear models for diagnosis in large industrial systems is not straightforward. Most of the times, the models have been created for other purposes different from diagnosis, and many times the required analytical redundancy is small. The approach proposed in this work combines techniques from two different research communities within Artificial Intelligence: Model-based Reasoning and Neural Networks. In particular, in this work we propose to use Possible Conflicts, which is a model decomposition technique from the Artificial Intelligence community to provide the structure (equations, inputs, outputs, and state variables) of minimal models able to perform fault detection and isolation. Such structural information is then used to design a grey box model by means of state space neural networks. In this work we prove that the structure of the Minimal Evaluable Model for a Possible Conflict can be used in real-world industrial systems to guide the design of the state space model of the neural network, reducing its complexity and avoiding the process of multiple unknown parameter estimation in the first principles models. We demonstrate the feasibility of the approach in an evaporator for a beet sugar factory using real data.

1. Introduction

Integrated Systems Health Management (ISHM) is an engineering discipline that seeks to improve the operation of complex systems in the presence of faults and degradations. The development of efficient and reliable ISHM methodologies is fundamental to guarantee continuous operation, to comply with safety requirements and to reduce costs in large industrial systems. In particular, fault diagnosis of industrial systems is an essential technology to avoid fault effects propagation, to prevent failures, and to minimize downtime. The process of fault diagnosis includes recognizing that a fault has occurred in the system (i.e., fault detection), determining the root causes of the fault (i.e., fault isolation), and estimating the fault size (i.e., fault identification). In this work we focus on model-based approaches for on-line fault detection, isolation, and identification (FDII) in complex industrial systems.

Model-based diagnosis uses analytical models of the system to estimate the proper behaviour and to compare with current observations in order to detect anomalies. These models may come in different formats: based on Physics first principles, usually as collection of Ordinary or Differential Algebraic Equations, state-space models, or based on empirical data and relating input and output measurements. In the last three decades model-based diagnosis has been approached by two different communities: DX (Hamscher et al., 1992) –using Artificial Intelligence techniques–, and FDI (Gertler, 1998; Blanke et al.,

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2018.12.007.

* Corresponding author.
E-mail addresses: belar@infor.uva.es (B. Pulido), jesusm@autom.uva.es (J.M. Zamarreño), alejandromg@ubu.es (A. Merino), anibal@infor.uva.es (A. Bregon).

[1] B. Pulido and A. Bregon's work has been partially supported by Spanish Ministery of Economy and Competitiveness under grant DPI2013-45414-R.

[2] J.M. Zamarreño and A. Merino's work has been partially supported by Spanish Ministery of Economy and Competitiveness/FEDER under grants DPI2015-67341-C2-2-R and DPI2015-70975-P. J.M. Zamarreño also acknowledges support by the regional government of Castilla y León and the EU-FEDER (CLU 2017-09).

[3] The authors acknowledge personnel from "ACOR" for the data provided for these experiments.

https://doi.org/10.1016/j.engappai.2018.12.007
Received 25 May 2018; Received in revised form 13 November 2018; Accepted 24 December 2018
Available online 9 January 2019

2006; Patton et al., 2000) –based on Systems Theory and Control. Both communities provide different but complementary techniques, as demonstrated by recent works (Cordier et al., 2004; Bregon et al., 2014). This work elaborates on the similarities of both approaches and focus on consistency-based diagnosis using numerical models (Pulido et al., 2001).

Consistency-based diagnosis proceeds in three stages: first, fault detection is performed by detecting minimal conflicts in the system (minimal set of equations or components involved in predicting a discrepancy); second, fault isolation is achieved by computing the minimal hitting-sets of the conflicts; third, fault identification requires using fault models to predict the faulty behaviour (Reiter, 1987; Dressler and Struss, 1996), and rejecting those fault modes that are not consistent with current observations. The main advantage of model-based diagnosis techniques is the possibility to reuse existing models, but, when dealing with complex industrial systems, this is also its main handicap. For these systems, models are usually created for purposes different from fault diagnosis (such as operator training), their calibration is a rather complex problem, and the required analytical redundancy in the system is small (due to the high price of including additional sensors, and because their allocation is related to process control). Both problems exist in large industrial systems where complexity comes from the highly non-linear models required to mimic system performance. Consequently, reusing existing non-linear models for diagnosis in those systems is not straightforward.

Contrary to the model-based solutions, data-driven techniques have been widely applied in industrial systems for monitoring and diagnosis purposes. The basic idea behind these schemes is a direct construction of a fault diagnosis system using a set of collected process data without explicitly identifying an analytical system model, thus avoiding all the complexity of the model generation and calibration. However, in order to have an accurate and efficient solution to fault diagnosis using data-driven techniques, very large amounts of nominal and faulty data is necessary to correctly train the diagnosis system. This is specially relevant for industrial systems exhibiting complex, non-linear dynamics.

In this work, we propose a hybrid solution where model decomposition techniques can be used to analyse the relations among the system model variables to design the basic structure of reduced grey-box models that are then implemented using data-driven approaches. Thus, we will not need complex analytical models specifically designed for diagnosis purposes and we will not need large amounts of data due to the smaller number of system variables interacting within the reduced grey-box models. In particular, in this work we use Possible Conflicts (PCs) (Pulido and Alonso-González, 2004) and state space Neural Networks (ssNN) (Zamarreño and Vega, 1998). PCs define the complete set of minimal redundant models that can become conflicts and are computed off-line. PCs provide the structural model – equations, input, output, and state variables – that can be used for fault diagnosis. SSNN is a great tool for modelling non-linear processes as shown in several cases (González Lanza and Zamarreño, 2002; Zamarreño et al., 2000).

The main contribution of this work is a framework to use the structural information in each PC to design a different kind of executable model. In this work, where precise analytical models are difficult to handle, we propose to build grey-box models based on a state space neural network architecture derived from that structural information in the PC, which links measurements with equations, and consequently with parameters related to faulty behaviour. The ssNN are used to track the system behaviour, but once a fault detection is confirmed we go back to the structural information in the models and the consistency-based diagnosis paradigm to perform fault isolation.

Preliminary results in an evaporation unit for a beet sugar factory in Spain using real data show the feasibility of the approach. The system has slow dynamics and due to the high costs of the start-up mode, it should work for weeks uninterrupted. Main difficulty in the existing models comes from the number of unknown parameters to be identified in the model, and the presence of non-linearities that requires expert manipulation in order to derive diagnosis-oriented models. An additional problem to test any approach is that there is few information about faults actually happening. Hence, any feedback from the model-based diagnosis system will be very helpful for the system operators.

The organization of this paper is as follows: first we introduce the Possible Conflicts technique used to find minimal models. Second, we introduce the state space neural network approach to obtain grey-box models for the Possible Conflicts. Next, we introduce the case study which we later use for designing the grey-box models and test them on nominal and faulty data, finally drawing some conclusions.

2. Possible Conflicts for structural model decomposition

In this section we introduce the basic definitions to understand the model-based diagnosis approach using Possible Conflicts.

2.1. Possible Conflicts

The computation of the set of Possible Conflicts (PCs) (Pulido et al., 2001; Pulido and Alonso-González, 2004) is a system model decomposition method from the DX community, which searches for the whole set of submodels with minimal analytical redundancy (the number of equations in the submodel equals the set of unknown variables plus one) from a given system model to perform fault diagnosis. PCs are computed off line based on the analysis of the available system model, and each PC is represented as a set of equations, together with its known and unknown variables.

In this work we will focus our analysis on continuous systems, with only one nominal operation mode, whose nominal behaviour can be described as a set Σ of Differential Algebraic Equations (DAEs). The model of our system will be the required system description to perform model-based diagnosis:

Definition 1 (*Model*)**.** The system model can be defined as $M(\Sigma, U, Y, X, \Theta)$, where: Σ is a set of DAEs, defined over a collection of known and unknown variables: U is a set of inputs, Y a set of outputs, X a set of state and/or intermediate, i.e., unknown variables, and Θ is the set of model parameters.

In our system model we have opted for not including explicitly in Σ the observational model. In that sense, any variable in U or in Y is the potential source for a sensor fault. To cope with these faults explicitly we should introduce one equation for each input ($\hat{u}_i = u_{i_{mes}}$) or output ($y_{j_{mes}} = \hat{y}_j$), where we can introduce disturbance or sensor fault models. For the sake of simplicity, instead of introducing one different equation for each element in $U \cup Y$, we will use the notation m^* to denote that variable m is computed directly from a sensor, either for input or as output.

Off line PCs computation requires three steps:.

1. To generate an abstract representation of the system as a hypergraph. The nodes of the hypergraph are system variables and the hyperarcs represent constraints between these variables. These constraints are abstracted from the equations in Σ, because we only need the information about the measured and unknown variables in each model equation. Thus each equation $c_i \in \Sigma$ will provide one structural constraint: $(c_i : S_i, X_i)$, where $S_i \subseteq U \cup Y$ accounts for the measured input or output variables, and $X_i \subseteq X$ accounts for the unknown (state or intermediate variables in c_i).
2. To derive *Minimal Evaluation Chains* (MECs), which are minimal over-constrained subsystems. The existence of a MEC is a necessary condition for analytical redundancy to exist. MECs have the potential to be solved using local propagation (solving one equation in one unknown) from the set of available measurements. Each MEC will define a model for a subsystem: $MEC_m = (\Sigma_m, S_m, Y_m, X_m, \Theta_m) \subseteq M$ where $\Sigma_m \subseteq \Sigma, S_m \subseteq (U \cup Y), Y_m \subseteq Y, X_m \subseteq X, \Theta_m \subseteq \Theta$, S_m has at least one element, and Y_m has exactly one element.

3. To generate *Minimal Evaluation Models* (MEMs) by assigning causality[4] to the constraints in the set of MECs, if possible.
   Each constraint $(c_i : S_i, X_i) \in \Sigma_m$ can be solved in one or more causal ways. Hence, for each c_i there are $(c_{i_j} : S_{i_j}, X_{i_j})$ where $j \geq 1$.
   Each MEM is a directed hypergraph derived from a MEC_m, $H_{mem} = \{V_{mem}, R_{mem}\}$, where $V_{mem} = U_m \cup Y_m \cup X_m$ and $R_{mem} \subset V_{mem} \times \cdots \times V_{mem}$. The leaves in the hypergraph are the variables in U_m, the intermediate nodes are the variables in X_m, and the root is $\hat{y}_m \in Y_m$. Each hyperarc $r_i \in R_{mem}$ in the MEM represents one causal assignment $(c_{i_j} : S_{i_j}, X_{i_j}) \in \Sigma_m$ for c_i in MEC_m. That is, the MEM specifies the order in which equations in a MEC should be locally solved starting from input measurements U_m to generate the subsystem output $\hat{y}_m$, which is later checked against $y_m \in Y_m$. There will be zero or more MEMs for each MEC. If there are zero, it means that the MEC cannot be solved using local propagation. Otherwise, there is at least one MEM that represents an ordering about how the set of equations can be solved to estimate the redundant variable $\hat{y}_m$, and such ordering can be used to build an executable model.
   Given that we have opted for an implicit observational model, the presence of a measurement in the graphical description of a MEM, expressed as m^*, represents two different equations given its place in the graphical model. If m^* is a leaf in the MEM, it represents a value from U_m, and the node $u_i^* \in U_m$ must be interpreted as $\hat{u}_i = u_{i_{mes}}$. If m^* is the root, it is the only output value in the MEM, and represents a value in Y_m. The node $y_m^* \in Y_m$ must be interpreted as $y_{m_{mes}} = \hat{y}_m$.

Although these steps are done off line, the executable model provided by the MEM (in the form of simulation or state-observer, as it will be explained later) can be used afterwards on line to estimate nominal behaviour, thus allowing to track the system and computing a residual for each MEM, being the difference between the estimated and the measured variables: $\hat{y}_m - y_{m_{mes}}$.

In consistency-based diagnosis (Reiter, 1987; de Kleer and Williams, 1987) a conflict arises given a discrepancy between observed and predicted values for the same variable.[5] Hence, conflicts are the result of the fault detection stage. PCs have been designed to find off line those subsystems capable to become conflicts online. This notion leads to the definition of a Possible Conflict:

Definition 2 (*Possible Conflict (PC)*)**.** The set of constraints in a MEC that give rise to at least one MEM.

Different works have demonstrated the relation between the DX and FDI approaches (Cordier et al., 2004), between PCs/MECs, Analytical Redundancy Relations (ARRs), and other structural model decomposition methods for static systems (Armengol et al., 2009), and linear dynamic systems (Bregon et al., 2014). Because we are interested in diagnosing continuous dynamic systems, we need to include additional information to make more explicit this similarity.

2.2. Inclusion of temporal information in the models

The common framework for DX and FDI diagnosis stated by Cordier et al. (2004) and later extended by Armengol et al. (2009) works just for static systems. There is no general theory for dynamic systems in the DX world that can extend the Reiter formalization for consistency-based diagnosis (Reiter, 1987). For that reason, it is necessary to provide *ad-hoc* extensions to consistency-based diagnosis for dynamic systems (Mosterman and Biswas, 1999; Travé-Massuyès and Milne, 1997; Bregon et al., 2014).

The main difference between static and dynamic system models is the presence of equations or constraints linking state variables and their derivatives (Dressler, 1996; Frisk et al., 2003; Pulido et al., 2010). As a consequence, there are two kinds of constraints in the model: *Differential* constraints, those used to model dynamic behaviour, and *instantaneous* constraints, those used to model static or instantaneous relations between system variables. Differential constraints represent a relation between a state variable and its first derivative $(x(t), \dot{x}(t))$. As shown by Frisk et al. (2003) the fact that this equation is explicitly represented or not does not introduce genuinely new structural information. In the PCs approach this representation is always explicit.

Differential constraints in the MEMs can be interpreted in two ways, depending on the selected causality assignment. In integral causality, and assuming that we have a discretized model and a time sampling equal to 1, this constraint is solved as $x(t) = x(t-1) + \int_{t-1}^{t} \dot{x}(t)dt$. In derivative causality, $\dot{x}(t) = \frac{dx(t)}{dt}$ assumes that the derivative can be computed based on present and past samples for x. Integral causality usually implies using simulation, and it is the preferred approach in the DX field. Derivative causality is the preferred approach in the FDI approach. Both have been demonstrated to be equivalent for numerical models, assuming adequate sampling rates and precise approximations for derivative computation are available, and assuming initial conditions for simulation are known (Chantler et al., 1996). PCs can easily handle both types of causality, since they only represent a different causal assignment while building MEMs (Pulido et al., 2010).

Special attention must be paid to loops in the MEM (set of equations that must be solved simultaneously). Loops containing differential constraints in integral causality are allowed, because under integral causality the time indices are different to both sides of the differential constraint (Dressler, 1994, 1996). It is generally accepted that loops containing differential constraints in derivative causality cannot be solved (Blanke et al., 2006).

Summarizing, each MEM for a PC represents how to build an executable model to monitor the behaviour of the subsystem defined by the PC, just traversing the MEM structure from leaves to the root (which is $\hat{y_m}$). Such executable model can be implemented as a simulation model or as a state-observer (Pulido et al., 2010; Bregon et al., 2014). However, building such model for complex non-linear systems is not a trivial task.

2.3. From DAEs to state-space representation in PCs

Our PCs are made up of a collection of DAEs: $\Sigma_m \subseteq \Sigma$, and when we include differential constraints they contain state variables. Can we obtain a state-space representation equivalent to the DAE form? We will explain that in this subsection.

Let us consider a dynamic nonlinear system described by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t)) \quad (1)$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)) \quad (2)$$

where $\mathbf{x} \in \Re^{n_x}$, $\mathbf{u} \in \Re^{n_u}$, $\mathbf{y} \in \Re^{n_y}$ are the state, input, and output vectors, respectively, $\mathbf{v} \in \Re^{n_v}$ $\mathbf{w} \in \Re^{n_w}$ represents the process and measurement noise vectors, respectively, and $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are nonlinear functions. The dimension of a vector a is denoted by n_a. The general model described by Eqs. (1) and (2) can be implemented as a simulation or a state observer model as follows:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{v}(t)) + \mathbf{k} \cdot (\mathbf{y}(t) - \hat{\mathbf{y}}(t)) \quad (3)$$

$$\hat{\mathbf{y}}(t) = \mathbf{g}(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{w}(t)) \quad (4)$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the estimated state and output variables, respectively. $\mathbf{k}$ is the observer gain, which filters out the difference between the measured and estimated variables, minimizing the error, $\mathbf{e_y}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t)$, considering a given design criterion. Depending on the computed value for $\mathbf{k}$, if $\mathbf{k} = 0$ we have the simulation model for the system, but if

[4] In this context, by causality assignment we mean every possible way one variable in one equation can be solved assuming the remaining variables are known.

[5] In FDI terminology a conflict arises when the residual deviates significantly from zero.

$\mathbf{k} \neq 0$ we have a general state observer model for the same system (Puig et al., 2006).

A MEM is built in such way that guarantees structural observability, according to Staroswiecki (2007):

Definition 3 (*Structural Observability*)**.** A system in integral causality is structurally observable if there exists a matching[6] that is complete on the unknown variables.

As mentioned before in this work we have imposed integral causality on Possible Conflicts calculation. Moreover, each PC defines a strictly overdetermined set of constraints, i.e. a MEC. By construction each MEC defines a bipartite graph, made up of the constraints and the unknown variables, with a complete matching in the unknowns[7] (Pulido and Alonso-González, 2004). Consequently, once differential equations are introduced, each MEM defines a subsystem that is structurally observable (Bregon et al., 2014) if it contains at least one state variable and its differential constraint.

Theorem 1. *Given a PC, if its MEM describes a dynamic subsystem, this subsystem is structurally observable.*

Given a Minimal Evaluation Model, MEM_i, $H_{mem_i} = \{V_{mem_i}, R_{mem_i}\}$, its state space representation can be expressed in a general way by the tuple $(\mathbf{x_i}, \mathbf{u_i}, y_i, \mathbf{f_i}, g_i)$, where:

- $\mathbf{x_i} = \langle x_{i_1}, x_{i_2}, \dots, x_{i_n} \rangle$ is the state vector of the system described by MEM_i,
- $\mathbf{u_i} = \langle u_{i_1}, u_{i_2}, \dots, u_{i_m} \rangle$ is the input vector of the system described by MEM_i,
- y_i is the output of the system described by MEM_i
- $\mathbf{f_i}$ is the state function of the system described by MEM_i,
- g_i is the output function of the system described by MEM_i.

with $\{x_{i_1}, x_{i_2}, \dots, x_{i_n}, u_{i_1}, u_{i_2}, \dots, u_{i_m}, y_i\} \subseteq V_{mem_i}$, $\{\dot{x}_{i_1}, \dot{x}_{i_2}, \dots, \dot{x}_{i_n}\} \subseteq V_{mem_i}$, y_i is the discrepancy node of MEM_i, $\{u_{i_1}, u_{i_2}, \dots, u_{i_m}, y_i\}$ are the only observed variables of MEM_i and

$$\dot{\mathbf{x}}_i(t) = \mathbf{f_i}(\mathbf{x_i}(t), \mathbf{u_i}(t)) \tag{5}$$

$$y_i(t) = g_i(\mathbf{x_i}(t), \mathbf{u_i}(t)) \tag{6}$$

When MEM_i has no algebraic loops, each component of the state function, f_{i_j}, is obtained from H_{mem_i} by the following procedure:

- Build subgraph $H_{f_{i_j}} \subseteq H_{mem_i}$, traversing H_{mem_i} from the occurrence of $\dot{x}_j$ to the first occurrence of either an input or a state variable.
- Compose the equations that label the arcs of $H_{f_{i_j}}$ from inputs and state variables to $\dot{x}_j$.

Similarly, the output function g_i is obtained from H_{mem_i} by the procedure:

- Built subgraph $H_{g_i} \subseteq H_{mem_i}$, traversing H_{mem_i} from the output y_i to the first occurrence of either an input or state variable.
- Compose the equations that label the arcs of H_{g_i} from inputs and state variables to y_i.

By construction, the causal matching in each MEM guarantees that $\forall i, j$, $H_{f_{i_j}}$ and H_{g_i} can be built for any MEM_i and state variable x_{i_j} in MEM_i. Consequently, when MEM_i has no algebraic loops, the analytical expression of f_{i_j} and g_i can always be obtained from MEM_i.

If MEM_i has an algebraic loop, we cannot obtain the analytical expression of f_{i_j} and/or g_i. Nevertheless, we still can built $H_{f_{i_j}}$ and H_{g_i}, which provide the structural description of f_{i_j} and g_i, respectively. From these structural descriptions an external solver can compute the value of all the unknown variables in state space formulation.

2.4. Fault detection and isolation capabilities for PCs

Each *PC* can be used to track a part of the system and consequently is sensitive to faults in that part of the system. In this work we have considered both additive and parametric faults related to output sensor faults and faults in components. Each fault will affect a parameter in one model equation. Regarding faults in components we will consider for instance mass or heat leakages, together with broken pipes or blocked valves.

If we decide to include explicitly faults in sensors, we should add one column for each input in S_m, and the output Y_m in the model for each PC.

Using this structural information we can build the Theoretical Fault Signature Matrix for the set of PCs, linking measurements and parametric faults with their corresponding equations. This is the information required to perform fault isolation.

Next section shows the fundamentals for the type of neural network model used in this work, and later we will illustrate all these concepts in our case study.

3. State space neural networks for behaviour estimation

3.1. Description

State space Neural Networks (ssNN) (Zamarreño and Vega, 1998) is a great tool for modelling non-linear processes as shown in several cases (González Lanza and Zamarreño, 2002; Zamarreño et al., 2000); even in the sugar industry (Zamarreño and Vega, 1997). Main advantages of such modelling approach are its ability for representing any non-linear dynamics, and what is called a *parallel model*. This model represents the cause–effect process dynamics without considering past inputs and/or past outputs. The dynamic relation is modelled by the state layer, which calculates the internal state of the network using just current inputs of the model and internal state values from the previous time step.

The architecture of the ssNN (see Fig. 1) consists of five blocks, and each block represents a neural network layer. From left to right, the number of neurons (or processing elements) at each layer is n, h, s, $h2$ and m. The third layer represents the state of the system (the dynamics). As it can be seen in the figure, there is a feedback from the state layer to the previous layer, which means that the current state depends (in a non-linear way) on the state at the previous time step. The second and fourth layers model the non-linear behaviour: from the inputs to the states and from the states (and possibly inputs) to the outputs, respectively. The first and the fifth layers provide linear transformations from inputs and to outputs, respectively.

The ssNN is implemented by the following mathematical representation:

$$\hat{\vec{x}}(t+1) = W^h \cdot f_1(W^r \cdot \hat{\vec{x}}(t) + W^i \cdot \vec{u}(t) + B^h) \tag{7}$$

$$\hat{\vec{y}}(t) = W^o \cdot f_2(W^{h2} \cdot \hat{\vec{x}}(t) + W^u \cdot \vec{u}(t) + B^{h2}) \tag{8}$$

where the parameters are weight matrices, W, and bias vectors, B:

- W^i, W^h, W^r, W^{h2}, W^u, W^o are matrices with dimension h x n, s x h, h x s, $h2$ x s, $h2$ x n and m x $h2$, respectively.
- B^h and B^{h2} are bias vectors with h and $h2$ elements respectively.

[6] In the structural approach defined by Staroswiecki, the structural model defines a bipartite graph for the constraints and the unknown variables in the system. The matching in the definition refers to a matching in that bipartite graph. The reader can find additional information in those structural issues in the work by Blanke et al. (2006).

[7] This is easy to understand because MECs are built adding one constraint in each step to determine an unknown variable, mimicking how the CBD computational paradigm works on-line (de Kleer and Williams, 1987).

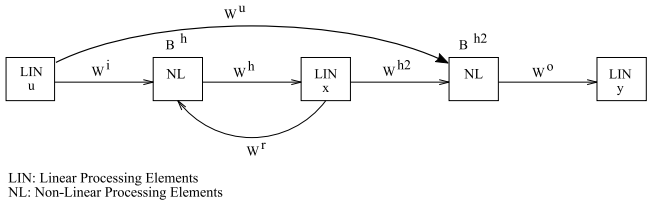

Fig. 1. Generic state space Neural Network architecture.

- f_1 and f_2 are two functions (non-linear, in general) which are applied elementwise to a vector or matrix. They are usually of sigmoid type.

Eqs. (7) and (8) represent a discrete nonlinear state space model where the nonlinear functions f_1 and f_2 map the nonlinearities of the states and outputs, respectively. In this sense, Eqs. (7) and (8) can be seen as a neural discrete form of the state space continuous representation given by Eqs. (5) and (6).

For some processes, where some *a priori* knowledge about the first principle equations can be obtained, a *black box* model could be too generic to obtain good results. But this knowledge can be used to restrict the *architecture* of the model, so we end up with a *grey box* model that can be better adjusted to mimic the process. We will apply this procedure at Section 5.1, when developing the neural models for the evaporation case study.

As seen at Section 2, a MEM represents how the equations must be used to compute the output, using measurements as inputs, and how the inputs are used to compute the intermediate unknown variables. At each MEM, the states can be easily identified as they appear as a differential constraint (a directed arc between the derivative of the state and the state itself). From this representation, a rather automatic procedure can be derived to obtain a custom ssNN model that represents the process characteristics in a better way. The steps to derive a custom ssNN model from a MEM are very similar to those described in Section 2.3 to obtain a state-space representation for a MEM. These steps are as follows:

- Identify the states. This gives us the number of states s for the ssNN model.
- Obtain the dependency between each state and the measurements (inputs) by analysing f_i (Eq. (5)). This relationship establish a simplified weight matrix (W^i) where some connections are missing which results in a structured matrix with some zero submatrices. Moreover, as each state is modelled independently, the second layer at Fig. 1 can be split into s independent nonlinear blocks, which also simplifies W^h and W^r as some connections would not be present.
- Obtain the dependency between the output and the states by analysing g_i (Eq. (6)). This relationship could simplify W^{h2} if the output does not depend on all states.

As mentioned before, these neural models can also been obtained through the state-space representation deduced at Section 2.3. This procedure will be applied to the case study at Section 5.1 and explained in detail at Section 5.1.4.

3.2. Training

Training is the process of modifying the parameters (weights and bias) of the neural network to adjust its output to the process output. Error between the neural network output and process output has to be minimized, so the training procedure is an optimization task where some index, Sum Squared Error (SSE) in our case, has to be minimized.

A feedforward network is quite easy to train, using the backpropagation method or some of its variants. But a recurrent neural network (such as ssNN) is more difficult to train due to the recurrent connections. Stochastic methods are an alternative for this kind of neural network, which results in easier to implement training algorithms. The Modified Random Optimization Method (Solis and Wets, 1981) has been selected in this work, but with some modifications to improve convergence as shown in González-Lanza and Zamarreño (2002).

Summarizing, our proposal consist on analysing the system description in a given model finding the set of PCs, then using ssNN to implement the MEMs just considering the causal relations between variables. Then, the ssNN must be trained to track the system behaviour using just nominal data. The residual obtained from the difference between the expected and real behaviour will be used for fault detection.

4. Case study: an evaporation section in a beet sugar factory

4.1. Description of the evaporation unit

The proposed approach will be tested using real data corresponding to an evaporation station of a beet sugar factory. Sugar production involves various sections such as diffusion, purification, evaporation and crystallization (van der Poel et al., 1998). Sucrose is extracted from sugar beets in the diffusion section and then purified to remove impurities. At the end of those two stages, a juice with low sucrose concentration is obtained. Evaporation is one of the main parts of the plant; in the evaporation section the water contained in that juice is evaporated in order to obtain a higher concentration. Finally, the resulting syrup is used to obtain sugar crystals in a set of vacuum pans. Fig. 2 shows the main elements in an evaporation plant: the evaporation units. In evaporation sections, in order to use steam efficiently, various evaporators are arranged in a way that the steam produced in some evaporators is used by others working at lower pressure, while the juice flows from one effect to another increasing the sugar concentration. A group of one or more evaporators working at the same pressure is called an "effect". Only the first effect is fed with boilers steam and purified juice. In the last effect, the evaporated steam escapes from the juice chamber to a barometric condenser.

The case study plant corresponds to a sugar factory in Olmedo (Spain) that has a set of six effects interconnected through pipelines and valves. The total amount of measurements in the plant exceeds 850 sensors.

In sugar factories various types of evaporators are used. Usually, in the first effects, Robert evaporators are used, while in the last effects, due to the lower temperature difference between the steam and the juice, falling film evaporators are used (Asadi, 2007). In this case study, the first effect of the evaporation is analysed, so a Robert evaporator is used. Robert evaporators have two chambers: the juice chamber and the steam chamber. The steam chamber surrounds a set of vertical tubes that contains boiling juice. A flow of steam enters the steam chamber and transfers heat to the juice providing the energy needed for boiling. The steam condenses around the tubes and leaves the evaporator as condensate. Boiling juice overflows the tubes bundle and leaves the evaporator through a tube, called "central well". The inner tubes, where the heat transmission takes place is called "calandria" and together with head space form the "juice chamber". The steam produced from the water evaporation reaches the upper space and leaves the juice chamber by a pipe at the top.

In Fig. 3, a scheme of a Robert evaporation is displayed. The location of the variables used in the next sections to evaluate the PCs and the SSNN is also shown. The nomenclature of the variables used in this paper for the evaporator case study can be found in Table A.1 at Appendix.

4.2. The experimental data-set

Real data were collected from the 2015 campaign, from December 5th, 2014 to January 19th, 2015 with 30 s as sampling time in the sugar factory located at Olmedo (Valladolid). Collected variables were (see Fig. 3 for their location):

Fig. 2. Six evaporation units for the evaporation section in a beet sugar factory in Spain.

- Op: Opening of the control valve that regulates the juice coming into the evaporator (%)
- L: Level of the evaporator (%)
- T_{ji}: Juice temperature at the input (°C)
- T_{jo}: Juice temperature at the output (°C)
- T_s: Temperature of the steam entering the steam chamber (°C)
- W_s: Steam flow entering the steam chamber (T/h)
- P_s: Pressure of the steam entering the steam chamber (bar)
- T_v: Temperature of the generated steam (°C)
- W_{ji}: Juice flow at the input (m^3/h)
- W_{jo}: Juice flow at the output (m^3/h) (estimated from other process variables)
- P_v: Pressure of the generated steam (bar)
- P_{vn}: Pressure at the next evaporator (bar). This variable does not appear in Fig. 3, but it is important because the steam flow leaving the evaporator depends on this pressure.

c_{ji} and c_{jo} correspond to the sugar concentration at the input and the output of the evaporator, respectively. The value of c_{ji} is obtained by chemical analysis periodically but its dynamics are smaller than the rest of variables analysed here. Hence, we consider c_{ji} as an input and constant through the simulation process. On the other hand c_{jo} is not measured until the end of the last effect.

LC is the acronym for a level controller existing in the evaporator which is a Proportional Integral Derivative (PID) regulator that measures the level of the evaporator L and calculates the value for the valve opening Op for the juice entering the evaporator, see Eqs. (20), (40) and (41). The juice flow leaving the evaporators W_{jo} is not directly measured in the plant. Nevertheless, this flow can be easily estimated using other plant measurements and applying the equation for the calculation of the flow of an incompressible fluid through a valve:

$$W_{jo} = Op_n \cdot Cv_n \cdot sgn(\Delta P_n) \cdot \sqrt{|\Delta P_n|} \tag{9}$$

being ΔP_n the pressure difference between effect 1 and 2:

$$\Delta P_n = (Pv + \rho \cdot g \cdot h) - (Pv_n + \rho_n \cdot g \cdot h_n) \tag{10}$$

4.3. The simulation models

In process industry, there are usually available simple stationary models used by plant engineers to make calculations about the process performance. Recently, more elaborate dynamic models have been developed in process plants to implement Data validation and Reconciliation, Real Time Optimization (RTO), Economic Model Predictive Controllers, training simulators, etc. (Engell, 2007). For the case study described in this paper, there exists a training simulator, and stationary models for RTO (Merino et al., 2005; Acebes et al., 2009).

Rigorous modelling of an evaporator is quite complex. Besides classical mass and heat balances, models should include complex flow patterns, evaporation calculations and thermodynamic equations. If those models are also going to be used in fault detection, extra relations to represent the faulty behaviour should be included. Even with the existence of complex dynamic models for other purposes, the reuse of those models for diagnosis is not straightforward.

On the one hand, the available set of measurements in the process together with the set of equations will determine the available analytical redundancy in the system, which will be the basis for our model-based diagnosis proposal.

On the other hand, there is no need of a match between the physical causality of the modelled system and the causality imposed by the

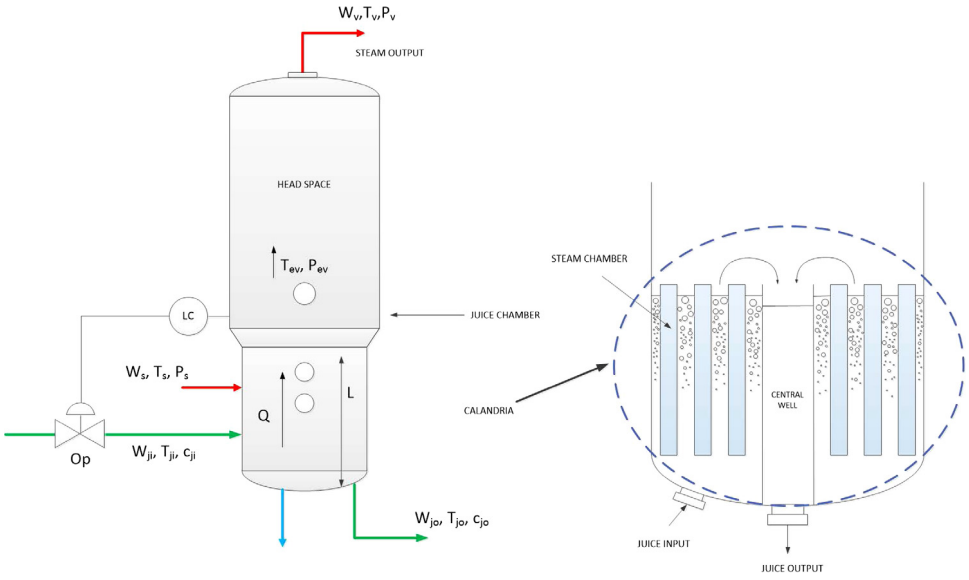

Fig. 3. Schematic of an evaporator.

Table 1
RMSE for simulated variables using the first principles mathematical model.

Variable	RMSE	Output mean	Relative error
Wji	16.62 m^3/h	540 m^3/h	3%
Tv	0.74 °C	124 °C	0.6%
Pv	0.048 bar	1.4 bar	3.4%

availability of the measures. This involves the symbolic manipulation of the mathematical model, which is usually complex, even when using object oriented modelling languages that allow non-causal modelling. For example, in the case of evaporation, the juice level is a measured variable. This variable, from the point of view of the physical modelling, is a state variable that is calculated by numerical integration. In the case of fault diagnosis, this variable is a measurement that cannot be calculated by the model by integration without appearing a high index problem (Pantelides, 1988). This makes necessary to manipulate the model so that the present binding disappears. In the simple case described that includes only one effect of an evaporation section, six parameters need to be estimated, and the assumptions and simplifications made in the models can cause mismatches that can induce false detection.

To develop this work two different models of one evaporator have been used. The first one is a simple model, provided in Section 4.4, similar to the basic mass and energy balances available in many industries. This model is enough for the proposed methodology because it provides the relations between process variables necessary to implement PCs, avoiding the intensive work related with detailed modelling development.

The second model was developed to simulate the behaviour of the evaporator when failures occur and was necessary due to the difficulty to obtain real plant data with the failures needed to perform the tests for this study. In this case, the model is a detailed rigorous complex dynamic mathematical model that uses first principles equations. This model is based on previous models created for a training simulator carried out at the University of Valladolid, Spain (Merino et al., 2005). The model has been developed using the EcosimPro (Empresarios Agrupados Internacional, 2012) simulation language, by the aggregation of libraries of elemental units in an object oriented modelling approach (Merino, 2008). The model includes the evaporator and other auxiliary equipment of the plant, resulting in 144 equations and 10 state variables and is able to represent accurately the plant performance.[8]

Fig. 4 shows the comparison between the simulated and the real measured variables for two main variables on the evaporation process. The upper graph shows the juice flow entering the juice chamber; such flow is regulated by a valve controlling the level. The lower graph shows the evolution of the juice temperature leaving the juice chamber. The model also includes equations to simulate several malfunctions that can eventually occur in evaporation sections. Due to the lack of failures in the collected data, the simulator is going to be used to generate data corresponding to the expected malfunctions, as described in Section 6.

Table 1 shows the RMSE (Root-mean-square error) for some plant measurements compared with the values obtained with the mathematical model.

4.4. Model equations for the evaporator

In this section, the model used to perform the different MEMs for the evaporator is detailed. This model is a reduced version of the model used for the generation of failures. The complete model is quite complex and will not be described here. For details about the complete model (Merino, 2008) can be consulted. The models are presented attending to the different parts of a Robert evaporator as described before: the juice chamber and the steam chamber. As it was mentioned before, the nomenclature used for the evaporation case study is detailed in Table A.1 at Appendix.

The general assumptions for the model described here are the following:

- Inside all chambers perfect mixing is considered, so properties are homogeneous inside the whole chamber and equal to the properties at the outlet.
- Steams are considered always saturated, so no sensible heat is considered.
- Energy losses are neglected.
- Gases are considered as ideal.
- Liquid and gas phases are perfectly separated.

Specific assumptions for the different parts of the evaporator are included in the next subsections.

4.4.1. Juice chamber

The juice chamber includes the calandria, in the lower part, in which the heat is transferred from the steam to the juice, and the head space where the produced steam is separated from the juice.

4.4.2. Calandria

Assumptions made for the juice in the calandria are described next.

- Dynamic is considered for the mass and energy balances.
- For the calculation of properties, juice is considered a technical sucrose solution.
- Purity does not change, so purity will be considered constant and equal to 0.92.
- Juice density is considered constant, so the level can be considered proportional to the mass of juice inside the evaporator.
- The calandria is considered as a unique chamber without distinction of central well, tubes, etc.

Equations for the juice in the calandria are the following. Thermodynamic relations f_1, f_2, f_3 and f_4 are detailed in Section 4.4.5.

$$\frac{dm_j}{dt} = W_{ji} - W_{jo} - W_{ev} \tag{11}$$

$$\frac{d(m_j \cdot c_{jo})}{dt} = W_{ji} \cdot c_{ji} - W_{jo} \cdot c_{jo} \tag{12}$$

$$\frac{d(m_j \cdot H_{jo})}{dt} = W_{ji} \cdot H_{ji} - W_{jo} \cdot H_{jo} - W_{ev} \cdot H_{ev} + Q \tag{13}$$

$$L = \frac{m_j}{m_{jmax}} \tag{14}$$

$$H_{ji} = f_1(T_{ji}, c_{ji}, Pu) \tag{15}$$

$$H_{jo} = f_1(T_{jo}, c_{jo}, Pu) \tag{16}$$

$$P_{ev} = f_2(T_{jo}, c_{jo}) \tag{17}$$

$$T_{ev} = f_3(P_{ev}) \tag{18}$$

$$H_{ev} = f_4(T_{ev}) \tag{19}$$

The juice entering the juice chamber, can be estimated using Eq. (21) that calculates the flow through a valve as a function of the valve opening Op that is calculated from the control law of the regulator f_{PID}.

$$Op = f_{PID}(L) \tag{20}$$

$$W_{ji} = f_L(Op) \tag{21}$$

4.4.3. Steam phase

- Dynamic is considered for the mass and energy balances.
- Generated steam is saturated, so Antoine equation (Antoine, 1888) (Eq. (24)) is used to relate temperature and pressure of the vapour.

[8] The complete Ecosim model for only the evaporation section of the factory contains more than 2500 equations and more than 250 state variables.

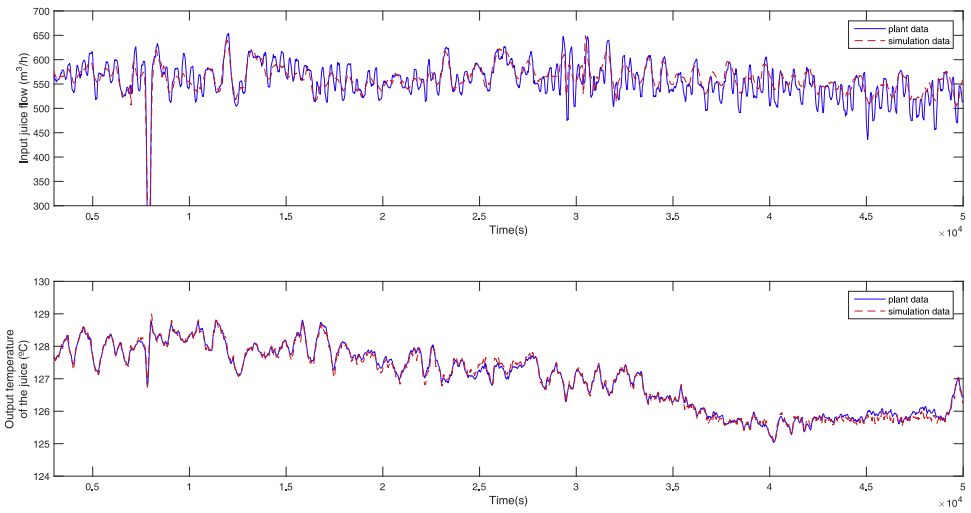

Fig. 4. Comparison of real plant data with simulation results for W_{ji} and T_{jo} measurements.

- Pressure of the steam vapour leaving the chamber is equal to the pressure inside the juice chamber.

$$\frac{dm_v}{dt} = W_{ev} - W_v \tag{22}$$

$$\frac{d(m_v \cdot H_v)}{dt} = W_{ev} \cdot H_{ev} - W_v \cdot H_v \tag{23}$$

$$T_v = f_3(P_v) \tag{24}$$

$$H_v = f_4(T_v) \tag{25}$$

$$P_v \cdot V = \frac{m_v}{0.0018} \cdot 8.31 \cdot 10^{-5} \cdot (T_v + 273.15) \tag{26}$$

$$W_{ev} = k_{w1}\sqrt{P_{ev}^2 - P_v^2} \tag{27}$$

$$W_v = k_{w2}\sqrt{P_v^2 - P_{vn}^2} \tag{28}$$

4.4.4. Steam chamber

For the case of the steam chamber, the model is very simple. The following assumptions have been made.

- Perfect mixing is considered for the steam.
- No dynamic in the energy balances is considered for the steam chamber.
- The steam entering the chamber not saturated, but sensible heat is neglected.
- Neither accumulation of steam nor condensed water is considered, so mass balances are not needed and the flow of condensed water is equal to the steam flow entering the chamber.
- Non-condensable gases are not considered.
- The heat transfer coefficients are considered constant and independent of any other variable.
- Pressure in the steam chamber is considered equal to the pressure of the entering steam.

With this modelling assumptions, the model equations for the steam chamber are:

$$Q = U \cdot S \cdot (T_s - T_{jo}) \tag{29}$$

$$\lambda_s = f_5(P_s) \tag{30}$$

$$W_s = \frac{Q}{\lambda_s} \tag{31}$$

4.4.5. Auxiliary functions

Auxiliary functions, named with the letter f and a sub-index are detailed next. f_1 to f_5 correspond with physico-chemical properties and have been obtained from (Bubnik et al., 1995).

- f_1 is the relation between the enthalpy, brix and temperature of the juice and can be calculated with the following equation.

$$H_j = T_j \cdot (4.1868 - c_j \cdot (0.0297 - 4.6 \cdot 10^{-5} \cdot Pu_j) + 3.75 \cdot 10^{-5} \cdot c_j \cdot T_j) \tag{32}$$

- f_2 is the thermodynamic calculation of the vapour pressure of the juice; this correlation can be evaluated using the following equations:

$$P = 10^{\left(\frac{-2147}{T_x + 273.15} + 5.7545\right)} \tag{33}$$

$$T_x = \frac{-k_2 + \sqrt{k_2^2 - 4 \cdot k_3 \cdot (k_1 - T_j)}}{2 \cdot k_3} \tag{34}$$

$$k_1 = -0.2 + e^{\left(-1.5254 + 0.022962 \cdot c_j + 0.0001161 \cdot c_j^2\right)} \tag{35}$$

$$k_2 = 0.9985 + 0.01 \cdot e^{\left(-3.2021 + 0.066743 \cdot c_j + 0.0002163 \cdot c_j^2\right)} \tag{36}$$

$$k_3 = 0.0001 + e^{\left(-1.4276 + 0.024362 \cdot c_j + 0.0006047 \cdot c_j^2\right)} \tag{37}$$

- f_3 is the relation between the pressure and temperature of saturated steam.

$$T = \left(\frac{3816.44}{11.68346 - \log(P + 43.73)}\right) - 273.15 \tag{38}$$

- f_4 is the relation between the temperature and enthalpy for saturated steam, it can be calculated interpolating in a table. The table can be found in Bubnik et al. (1995) in pages from 283 to 289.
- f_5 corresponds with the calculation of the latent heat released from the condensing steam as a function of the condensing pressure P:

$$\lambda_s = \left(5.9893 \cdot 10^{-2} + f_3(P_s) \cdot \left(-6.1910^{-1} + f_3(P_s) \cdot \left(6.82 \cdot 10^{-4} + f_3(P_s) \cdot \left(-4.86 \cdot 10^{-6}\right)\right)\right)\right) \cdot 4.184 \tag{39}$$

- f_{PID} corresponds to the control law of Proportional Integral Derivative (PID) regulator.

$$f_{PID} = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{d}{dt}e(t)\right) \tag{40}$$

$$e(t) = L_{ref} - L \tag{41}$$

- f_L corresponds with the formula to calculate the flow of an incompressible fluid through a valve.

$$W_{ji} = Op_{ji} \cdot Cv_{ji} \cdot \sqrt{Pv_{ji} - (Pv + \rho \cdot g \cdot h)} \tag{42}$$

This equation is similar to Eq. (9) but applied to the incoming fluid.

Table 2
PCs for the evaporation unit model. Relations between PCs and equations.

PC	Equation																									
	(11)	(12)	(13)	(15)	(16)	(17)	(18)	(19)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)	(st_1)	(st_2)	(st_3)	(st_4)	(st_5)	12_{aux}	13_{aux}	23_{aux}
1									1																		
2												1															
3	1	1				1									1					1	1				1		
4	1	1				1				1				1	1	1					1		1		1		
5	1	1	1	1	1	1	1	1							1		1				1	1			1	1	
6	1	1				1	1	1		1	1		1		1	1					1		1	1	1		1
7																	1	1	1								
8	1	1	1	1	1	1	1	1							1			1	1		1	1			1	1	

Additionally to these equations that are required to estimate the behaviour of the different subsystems, as it was mentioned in Section 2, it is necessary to introduce in our models the relation between the state variables in the system and their derivatives. In our case study there are five state variables: $\{c_{jo}, H_v, m_j, H_{jo}, m_v\}$. Consequently, we need to introduce five additional equations that relate these variables and their derivatives in our discretized model as follows:

$$m_j(t) = m_j(t-1) + \int_{t-1}^{t} \dot{m}_j dt \qquad (st_1)$$

$$c_{jo}(t) = c_{jo}(t-1) + \int_{t-1}^{t} \dot{c}_{jo} dt \qquad (st_2)$$

$$H_{jo}(t) = H_{jo}(t-1) + \int_{t-1}^{t} \dot{H}_{jo} dt \qquad (st_3)$$

$$m_v(t) = m_v(t-1) + \int_{t-1}^{t} \dot{m}_v dt \qquad (st_4)$$

$$H_v(t) = H_v(t-1) + \int_{t-1}^{t} \dot{H}_v dt \qquad (st_5)$$

Finally, we have used three auxiliary equations to compute Possible Conflicts that were required to ease the EcosimPro model analysis while computing equations of the form:

$$d(x_i \cdot x_j)/dt = x_i \cdot d(x_j)/dt + x_j \cdot d(x_i)/dt \qquad (x_{aux})$$

These auxiliary equations could be removed if there was no previous simulation model. We have three auxiliary equations in our model: $(12_{aux}, 13_{aux}, 23_{aux})$ related to Eqs. (12), (13), and (23), respectively.

5. Grey-box models design for fault diagnosis

5.1. PCs for the evaporation unit and their associated ssNN

As mentioned in Section 4.1, the evaporation section of the sugar factory is made up of six effects working sequentially to increase the sugar concentration in the syrup. All the evaporation units in the same effect share the same steam output conduit, and provide the steam for the next effect, thus partially coupling the behaviour of all the units. For our tests we have focused on the first evaporation unit in the first effect.

There are several assumptions that must be made in order to simplify the original model used in the training simulator, and to use those first principles equations for diagnosis. In our case, we simplified the dynamic processes actually happening inside the evaporation chamber, and we assumed the system was in only one operation mode. The dynamic processes considered in the evaporation unit were: conservation law for the amount of sugar and no-sugar products, global mass balance in the evaporation chamber, sugar balance, level in the chamber, energy balance, steam volume balance, interchanged heat, and pressures in the chamber.

As a result of this simplification process, our model was made up of **27** equations based on first principles of physics, with **21** unknown variables, and **13** measured variables. Only five of these equations were used to model the evolution of state variables: c_{jo}, H_v, m_j, H_{jo}, m_v, corresponding to the sugar concentration of the juice leaving the juice chamber, the steam enthalpy (measured through the steam temperature), the juice mass in the juice chamber, the enthalpy of the juice leaving the juice chamber (measured through the juice temperature flowing out of the chamber), and the mass of steam in the steam chamber (measured through the generated steam), respectively. As previously mentioned, c_j is the sucrose concentration of the syrup, being its dynamic much slower than the rest of variables in the system. As a consequence, the input value c_{ji} is assumed to be known and constant during the diagnosis experiments.

The set of variables that can be measured is fixed by the set of sensors: inputs, $\{T_{j_i}, T_s, Op\}$, and outputs, $\{P_v, T_v, L, T_{jo}, W_{ji}, W_s, P_s, P_{vn}\}$. Finally, we have an estimation of the input flow in the evaporator that can be used as both input or output $\{W_{jo}\}$.

Further details about the equations, the variables, and the assumptions used in this case study have been presented at Section 4.4.

The algorithms used to compute the set of PCs provided 162 MECs, but only twelve MEMs. The total number of PCs (MECs with at least one MEM) in this system was eight. The relation between the PCs and the original model equations is represented in Table 2, where an entry of 1 in row i and column j represents that equation j is part of the subsystem described by PC_i.

The reason behind the small number of PCs, with respect to the number of MECs, is that in the original model there are several equations containing partial derivatives, and several non-linear functions. These equations usually force only one causal way for solving the unknown variables in the equation. For that reason, most of the generated MECs have no MEM. Additionally some of the MEMs can be hardly implemented, because of those non-linearities, although it is analytically possible. The problem we faced at that point was to select and implement the relevant MEMs for the set of PCs, because it would be necessary to write down by hand each simulation model. The process needs to be supervised by the modelling expert, thus producing a bottleneck in the development of the diagnosis system.

Table 3 represents the relation between the different PCs and their sets of inputs and output variables. The column residual in the table represents the root in the MEM, which is the system observation to be tracked and it is used to build the residual for fault detection, while the remainder set of variables in Table 3 must be considered as input variables (they are the leaves in the MEM). As mentioned in Section 2, we do not explicitly include in the set of equations the observational model. Hence, the presence of variables such as m^* in the MEMs related to the PCs must be interpreted in a different way depending if it is found as a leaf or as the root of the directed graph. In both cases, the presence of those measurements in the PCs could be also considered as a potential source of sensor faults in the Theoretical Fault Signature Matrix.

In the following we analyse the semantics of the **eight** Possible Conflicts found and how they were used for fault diagnosis in the evaporation unit.

Table 3
PCs for the evaporation unit model. Relations between PCs and observations.

PC	Residual	Measured Variables used in the PC											
		C_{ji}	Wji	Wjo	Tji	Tjo	Op	L	Ts	Tv	Pv	Pvn	Ws	Ps
1	W_{ji}		1				1	1						
2	T_v									1	1			
3	L	1	1	1		1		1			1			
4	P_v	1	1	1		1		1		1	1	1		
5	T_{jo}	1	1	1	1	1		1	1		1			
6	T_v	1	1	1		1		1		1	1	1		
7	W_s					1				1			1	1
8	T_{jo}	1	1	1	1	1		1			1		1	1

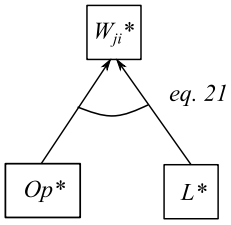

Fig. 5. Minimal Evaluable Model schematics for Possible Conflict PC_1. The estimated variable is W_{ji}.

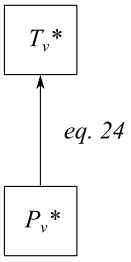

Fig. 6. Minimal Evaluable Model schematics for Possible Conflict PC_2. The estimated variable is T_v.

5.1.1. Possible Conflict 1: residual generation for W_{ji}

The first PC being modelled is PC_1, whose MEM is graphically described in Fig. 5. This subsystem contains only one equation, two input measurements –L and Op–, and one observed output variable, W_{ji}.

In this case, as the relationship is static, it is not necessary to use a ssNN, as it is oriented towards representation of dynamic systems. The equation that relates these three variables is well known in the literature as the flow through a valve is proportional to the opening percentage Op and also depends on the pressure drop which is proportional to the squared root of the pressure difference in the two ends of the valve (Eq. (43)). So, for this simple case, a parametric equation has been chosen to represent this PC, where its parameters, like Cv_{n-1}, have to be adjusted to the real data. This situation is also true for two other PCs: PC_2 that estimates T_v, and PC_7 that estimates W_s.

$$W_{ji} = Op_{n-1} \cdot Cv_{n-1} \sqrt{P_{v_{n-1}} - \rho \cdot g \cdot \frac{L \cdot h_m}{100}} \tag{43}$$

Subindex $n-1$ refers to the conditions previous to the first effect, and h_m is the maximum height of the juice level in the evaporator (m).

5.1.2. Possible Conflict 2: residual generation for T_v

The second PCs being modelled is PC_2, whose MEM is graphically described in Fig. 6. In a similar way to the previous PC, this subsystem contains only one equation, one input measurement –P_v–, and one observed output variable, T_v. The equation corresponds to Antoine's equation for saturated steam. This equation has been corrected including a small bias to adjust real plant data. The equation can be found at Section 4.4, Eq. (24) (and Eq. (38) provides its mathematical expression).

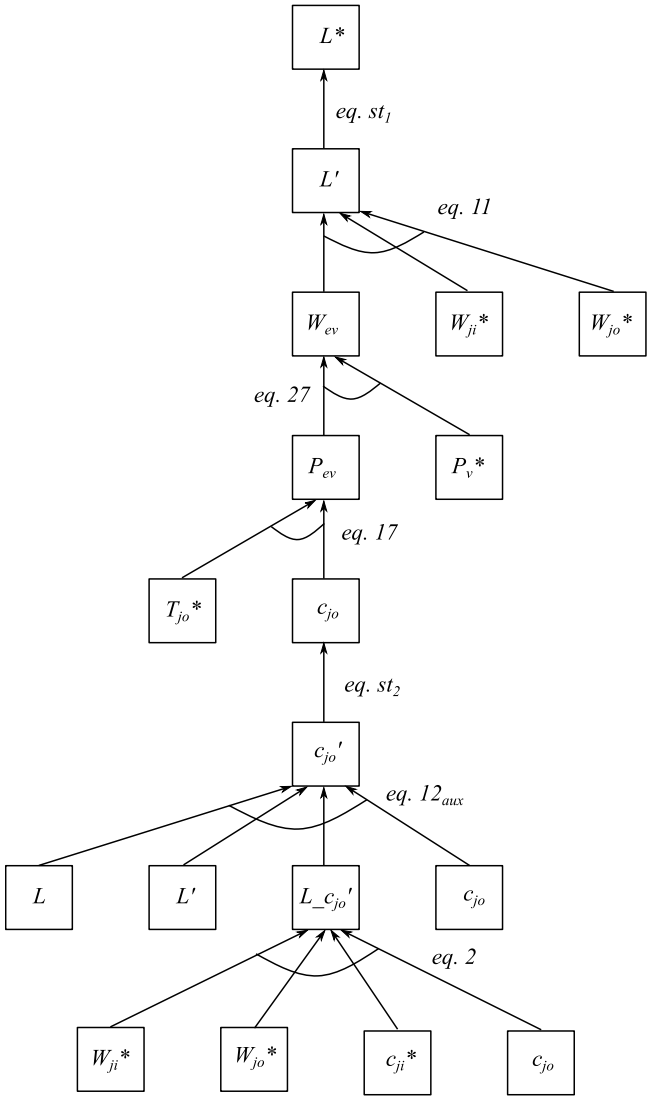

Fig. 7. Minimal Evaluable Model schematics for PC_3. The estimated variable is L.

5.1.3. Possible Conflict 3: residual generation for L

With respect to PC_3, shown at Fig. 7, and whose output is L, we are not able to train the associated ssNN since L is a controlled variable in the factory so it is always maintained at an almost constant value and no model is able to predict a constant output as a function of the inputs (except an obvious constant function). If we try to train a neural network to give a constant output, the obvious solution is to make zero the weights that connect to the inputs. So, this model is not going to give us any information about faults. In fact, as controlling this variable is critical (the evaporator tubes can be severely damaged if there is

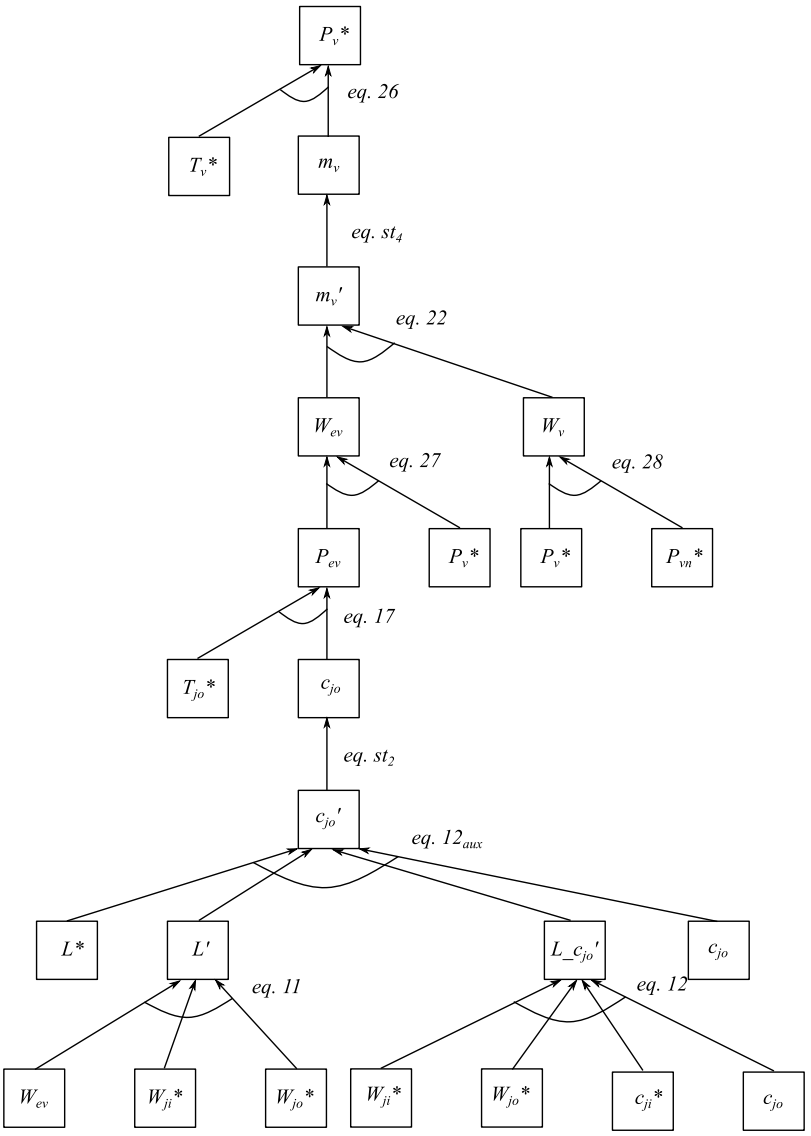

Fig. 8. Minimal Evaluable Model schematics for PC_4. The estimated variable is P_v.

not enough juice in the chamber), if the level at a tank previous to the evaporators is lower than a certain value, the control system of the factory automatically feeds water into the tank to guarantee the presence of fluid in the juice chambers of the evaporators. For those reasons, this PC is no longer useful for model-based diagnosis using our models.

5.1.4. Possible Conflict 4: residual generation for P_v

PC_4 is the next PC being modelled. Its related MEM, used to estimate P_v, is in Fig. 8, which graphically describes the relation between its inputs, state and intermediate variables, and its unique output. The subsystem contains 10 equations, one output measurement P_v, six input measurements –L, T_{jo}, T_v, P_{vn}, W_{ji}, and W_{jo}–, two state-variables –m_v and c_{jo}–.

Taking into account the procedure stated at Section 3, the ssNN architecture can be customized to represent the process characteristics in a better way, as described in Fig. 9. Looking at that figure, the second non-linear layer representing the non-linear relationship between P_v and m_v and T_v corresponds to Eq. (26). The first non-linear layer representing the non-linear relationship between the state m_v and the inputs T_{jo}, T_v and P_{vn} and the state c_{jo} corresponds to dependencies obtained from Eqs. (17), (22), (26), (27), (28). At the same layer, the non-linear

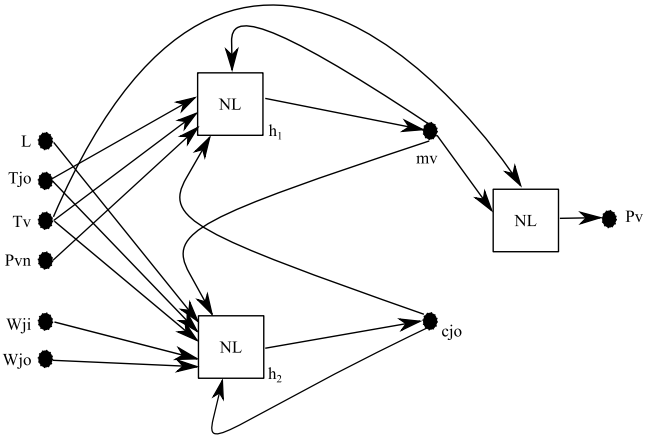

Fig. 9. Simplified ssNN representing the Minimal Evaluable Model MEM_5 for PC_4.

relationship between the state c_{jo} and the inputs L, T_{jo}, T_v, W_{ji} and W_{jo} and the state m_v corresponds to dependencies obtained from equations

$$W^u = \begin{bmatrix} 0 & 0 & u_{1,3} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & & \\ 0 & 0 & u_{h2,3} & 0 & 0 & 0 \end{bmatrix}$$

Fig. 10. Simplified weight matrix W^u for the ssNN implementing PC_4.

(12_{aux}), (11), (12), (26), (27). As it can be seen, the non-linear (hidden) layer is split into two parts, and each part (represented by NL inside a square) has a number of neurons (h_1, h_2) that must be adjusted by trial and error to represent the nonlinear dynamics of each state.

This simplified ssNN architecture can be viewed as removing some of the weights between layers, or setting zeros in some specific elements of the weight matrices. The structure of these matrices will be clearly seen for the next PC in detail. The main novelty of this model is the existence of matrix W^u whose structure is represented as shown in Fig. 10.

5.1.5. Possible Conflict 5: residual generation for T_{jo}

MEM_7 related to PC_5 is graphically described in Fig. 11. The MEM is a directed hypergraph which represents how the equations must be used to perform a double estimation of the variable H_{jo} using Eq. (16) and (st_3). However, since H_{jo} is not directly measurable, we can estimate T_{jo} which is measured by inverting Eq. (16). Now, we have an estimation of a measured variable T_{jo}, using just measurements as inputs, and how the inputs are used to compute the intermediate unknown variables.

The reader should notice that T_{jo} appears both as a leaf and as the root of the directed graph. Nodes labelled T_{jo} in the leaves represent past and previously estimated values for the variable, while T_{jo} as the root represents the new and actual estimation of the MEM, $\hat{T_{jo}}$, which is then used to compute the residual by comparison with sensor value T^*_{jo}.

This subsystem contains fourteen equations, six input measurements –L, P_v, W_{ji}, W_{jo}, T_s, and T_{ji}–, and two state-variables –c_{jo} and H_{jo}–. The observed output variable is T_{jo}.

MEM_7 graphically specifies in Fig. 11 the relations between the inputs, and the states, to estimate the output of the model, T_{jo}. Taking into account the procedure stated at Section 3, the ssNN architecture can be customized to represent the process characteristics in a better way,

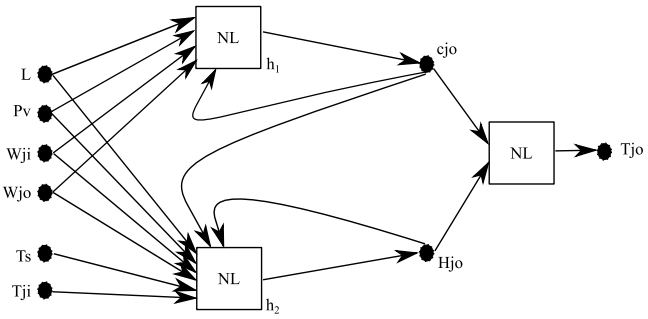

Fig. 12. Simplified ssNN representing MEM_7 for PC_5.

$$W^i = \begin{bmatrix} \begin{bmatrix} i_{1,1} & i_{1,2} & i_{1,3} & i_{1,4} & 0 & 0 \\ \vdots & \vdots & \vdots & & & \\ i_{h_1,1} & i_{h_1,2} & i_{h_1,3} & i_{h_1,4} & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} i_{h_1+1,1} & i_{h_1+1,2} & i_{h_1+1,3} & i_{h_1+1,4} & i_{h_1+1,5} & i_{h_1+1,6} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ i_{h_1+h_2,1} & i_{h_1+h_2,2} & i_{h_1+h_2,3} & i_{h_1+h_2,4} & i_{h_1+h_2,5} & i_{h_1+h_2,6} \end{bmatrix} \end{bmatrix}$$

$$W^h = \begin{bmatrix} \begin{bmatrix} d_{1,1} & \cdots & d_{1,h1} \\ 0 & \cdots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \cdots & 0 \\ d_{2,h1+1} & \cdots & d_{2,h1+h2} \end{bmatrix} \end{bmatrix}$$

$$W^r = \begin{bmatrix} \begin{bmatrix} r_{1,1} & 0 \\ \vdots & \vdots \\ r_{h1,1} & 0 \end{bmatrix} \\ \begin{bmatrix} r_{h_1+1,1} & r_{h_1+1,2} \\ \vdots & \vdots \\ r_{h_1+h_2,1} & r_{h_1+h_2,2} \end{bmatrix} \end{bmatrix}$$

Fig. 13. Simplified weight matrices W^i, W^h, and W^r for the ssNN implementing PC_5.

as described in Fig. 12, obtained analysing the dependencies in the same way that the previous PC. The non-linear (hidden) layer is split into two parts, and each part (represented by NL inside a square) has a number

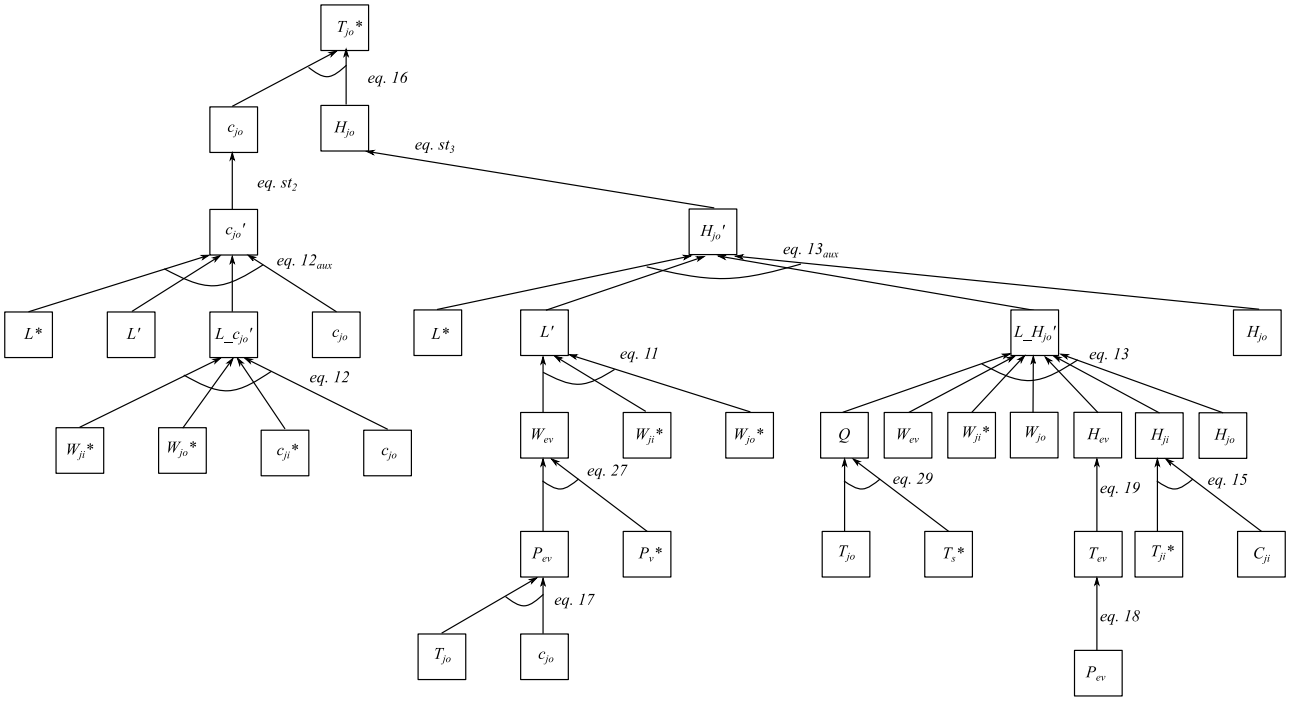

Fig. 11. Minimal Evaluable Model schematics for PC_5. The estimated variable is Tjo.

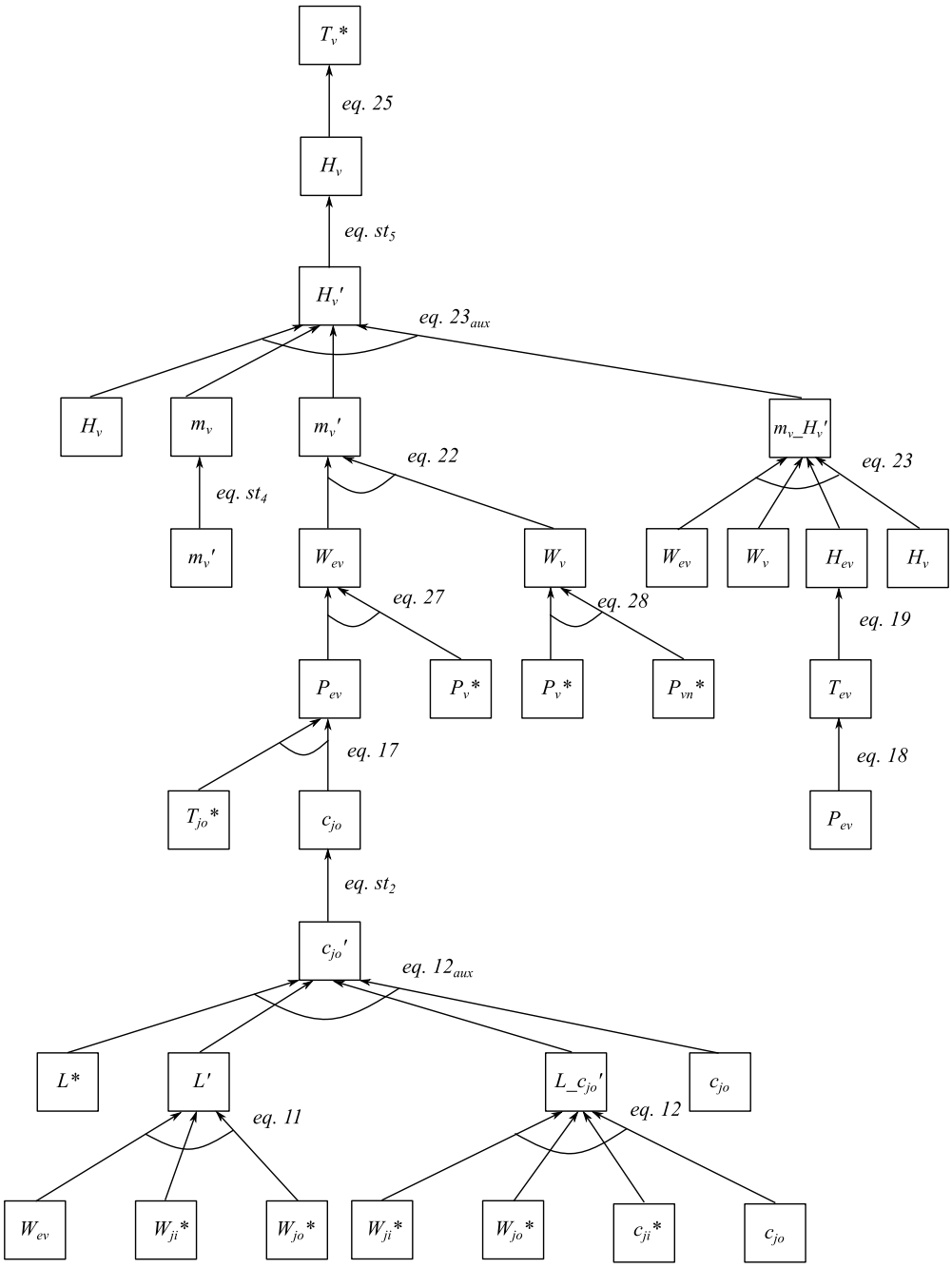

Fig. 14. Minimal Evaluable Model schematics for Possible Conflict PC_6. The estimated variable is T_v.

of neurons (h_1, h_2) that must be adjusted by trial and error to represent the nonlinear dynamics of each state.

This simplified ssNN architecture can be viewed as removing some of the weights between layers, or setting zeros in some specific elements of the weight matrices (the matrices can be seen in Fig. 13). Dimension of ad hoc matrices W^i, W^h, and W^r is $(h_1 + h_2) \times 6$, $2 \times (h_1 + h_2)$, and $(h_1 + h_2) \times 2$, respectively. The rest of the matrices (W^{h2}, W^o) maintain their original structure.

As the ssNN structure is more suited to the underlying equations governing the dynamics of the system, training of the neural network is carried out in an easier way, with excellent results as it will be seen later (Fig. 19 at Section 6).

5.1.6. Possible Conflict 6: residual generation for T_v

The next PC being modelled is PC_6, whose MEM is graphically described in Fig. 14.

This subsystem contains 15 equations, 6 input measurements –L, T_{jo}, P_v, P_{vn}, W_{ji} and W_{jo}–, and three state-variables –H_v, m_v and c_{jo}–. The observed output variable is T_v.

PC_6 graphically specifies in Fig. 14 the relations between the inputs (L, T_{jo}, P_v, P_{vn}, W_{ji}, and W_{jo}) and the states (H_v, m_v and c_{jo}). The output of the model is T_v. Again, taking into account the procedure stated at Section 3, the ssNN architecture can be customized to represent the process characteristics in a better way, as described in Fig. 15. The nonlinear (hidden) layer is split into three parts, and each part (represented by NL inside a square) has a number of neurons (h_1, h_2, h_3) that must be adjusted by trial and error to represent the nonlinear dynamics of each state.

Again, this simplified ssNN architecture can be viewed as removing some of the weights between layers, or setting zeros in some specific elements of the weight matrices. The simplified structure of these matrices could be represented in the same way as in the previous model.

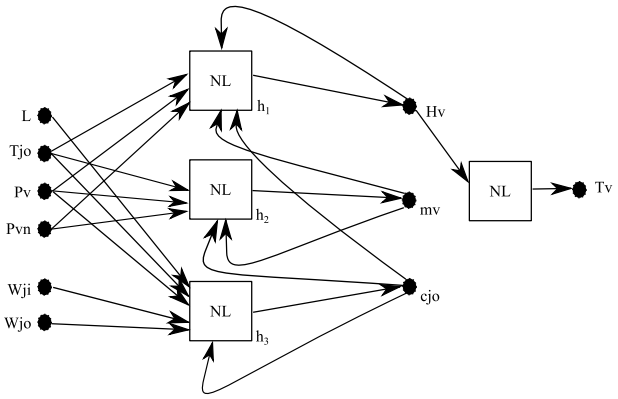

Fig. 15. Simplified ssNN representing the Minimal Evaluable Model MEM_8 for PC_6.

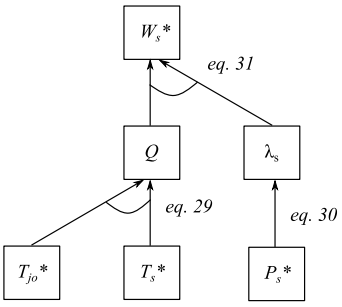

Fig. 16. Minimal Evaluable Model schematics for Possible Conflict PC_7. The estimated variable is W_s.

5.1.7. Possible Conflict 7: residual generation for W_s

The seventh PC being modelled is PC_7, whose MEM is graphically described in Fig. 16. This PC contains only three equations, with three input measurements –T_{jo}, T_s, P_s–, and one estimated variable W_s.

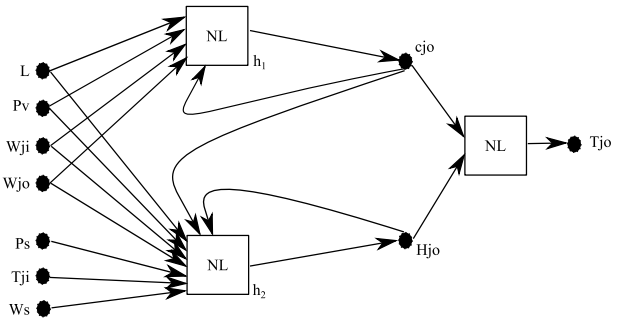

Fig. 18. Simplified ssNN representing PC_8.

This static relationship is similar to the PC_1 and PC_2 cases, where the equations are known (see Eq. (29), (30) and (31) at Section 4.4) and only some parameters have to be tuned to the real data.

5.1.8. Possible Conflict 8: residual generation for T_{jo}

The last PC being modelled is PC_8. The MEM described graphically in Fig. 17 is almost identical to the MEM used to compute PC_5. This strong similarity can be also confirmed looking at Table 2 and 3. PC_8 contains 15 equations, with seven input measurements –L, P_v, W_{ji}, W_{jo}, T_{ji}, P_s, and W_s–, and two state-variables –c_{jo} and H_{jo}–. The observed output variable is T_{jo}. The main difference with PC_5 is the way the intermediate variable Q is estimated. In PC_5 we used Eq. (29), with inputs T_{jo} and T_s, while in this case we use Eqs. (30) and (31), with inputs W_s and P_s, hence covering different physical components.

Accordingly, the simplified ssNN used to implement this MEM is very similar to that used for PC_5, as can be seen in Fig. 18.

6. Results on the case study

6.1. ssNN training

Collected data (see Section 4.2) from the real factory were used for adjusting the parameters of the neural models obtained at Section 5.1.

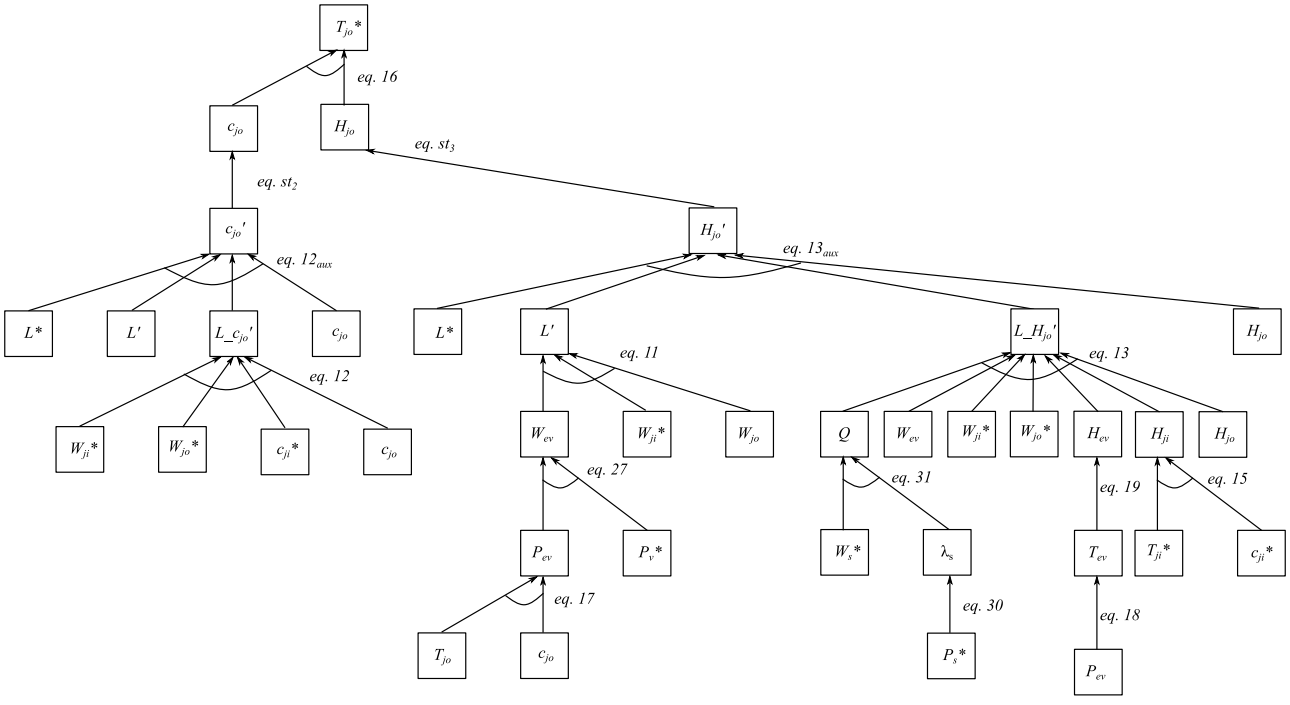

Fig. 17. Minimal Evaluable Model schematics for Possible Conflict PC_8. The estimated variable is T_{jo}.

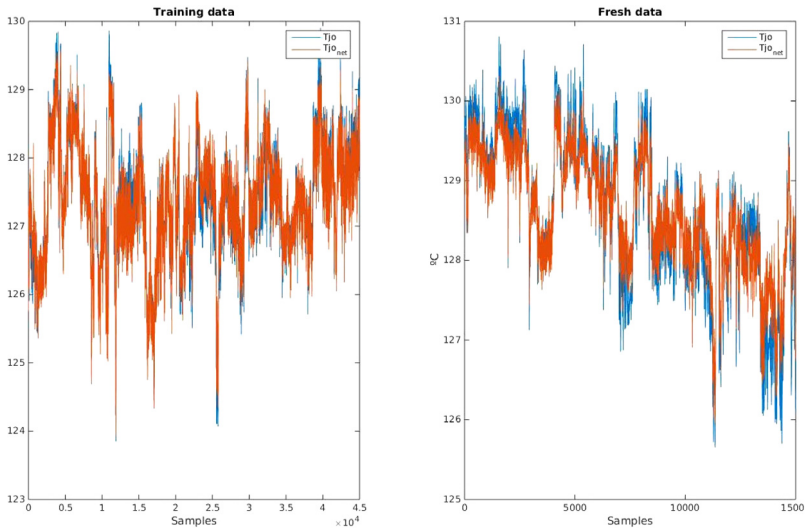

Fig. 19. Adjustment of one of the models (T_{jo} as output) to training (left) and validation (right) data.

Table 4
RMSE for training and validation datasets for each neural model. For PC_1, PC_2 and PC_7, static equations were manually adjusted to real data.

Model	Output variable	RMSE		Output mean	Relative error
		Training	Validation		
PC_1	W_{ji}	22.05 m^3/h		540 m^3/h	4%
PC_2	T_v	0.21 °C		124 °C	0.2%
PC_4	P_v	0.025 bar	0.038 bar	1.4 bar	(1.8, 2.7)%
PC_5	T_{jo}	0.27 °C	0.34 °C	127 °C	(0.2, 0.3)%
PC_6	T_v	0.10 °C	0.12 °C	124 °C	(0.08, 0.1)%
PC_7	W_s	7.49 m^3/h		140 m^3/h	5%
PC_8	T_{jo}	0.14 °C	0.14 °C	127 °C	0.1%

From these data, only part of them were used for training and validation. For each of the four neural models, the first 45000 data samples were used for training and 15000 intermediate data samples were used for validation on fresh data.

As an example, for one of the models, Fig. 19 shows the adjustment of the neural model to the training data as well as the performance when fresh data are fed to the model. For the rest of the models, similar results were achieved. As a summary, Table 4 shows the RMSE (Root-mean-square error) for training and validation datasets for each of the four neural models as well as the RMSE for PC_1, PC_2 and PC_7 static equations.

6.2. Faulty behaviour in the case study

Most of the real data available from the factory were related to nominal behaviour. As a consequence we have almost no real data corresponding to abnormal or faulty behaviour. But, our model-based diagnosis approach is based on using only models for correct behaviour and to provide an early alarm to operators even for unexpected faults. Hence, we need to feed the PCs with both nominal and faulty scenarios to test the validity of the approach.

To overcome the problem of the lack of real data from faulty situations we decided to test our proposal using the highly complex simulator (described in Section 4.3), whose behaviour had been tuned for representing the real factory. To obtain the faulty behaviour we fed real data as inputs to the model and we injected several faults in the simulator code. 5500 samples were generated with the simulator and various experiments were performed introducing two typical faults at the 3000 sample. To generate the faulty scenarios, we have computed the residual between the estimated nominal and faulty behaviours. That residual was later added to real output data after the 3000 time instant, thus simulating the effect of the fault on real data.

Take into account that the neural models have been trained on real data, not on the simulator, so the performance hopefully would be even better in case of real faults in the real plant.

Two faults were simulated: **Steam leak** –F_1– and **Sensor failure** –F_2. Additionally to these simulated faulty scenarios, we found a fault in the real data, related to measurement P_v that was acknowledged by plant operators as clearly abnormal, but they were unable to find the root cause of such behaviour. Consequently, we have introduced such data as our F_3 fault, labelled as **Unknown fault on** P_v.

- F_1 - **Steam leak**: Sometimes, steam can leak from the juice chamber. This can be produced for various reasons. Sometimes tubes can crack due to thermal or mechanical stress. In the case of evaporators of Robert type, the central tube can be emptied, and the steam can flow from one juice chamber to the next one. The effect of this failure in the process is that for the same energy provided by the heating steam and the same juice flow and temperature, the vapour pressure in the juice chamber will decrease (see Fig. 20).
- F_2 - **Sensor failure at** P_v: Sometimes, due to different factors such as mechanical or electrical failures, sensors can provide wrong measurements. In this case a bias in the sensor for variable P_v is artificially introduced.
- F_3 - **Unknown fault on** P_v: This corresponds to a detected change in the sensor dynamics on the real plant whose behaviour suggests a real fault has been found on the original data; no simulation bias is involved on this fault. According to Fig. 21, this effect is clearly seen from sample 5640. Plant operators were not able to find a source of the abnormal behaviour. Consequently, we labelled the fault as "abnormal evolution of P_v" signal dynamics with unknown root cause.

These faults introduce changes in the dynamics of the system because they affect different equations. In these scenarios we have the following dependencies, obtained by analysing the presence of fault parameters in the equations:

- F_1 - *Steam leak*: it is related to the computation of variable W_v, which is located in Eq. (28) at Section 4.4.
- F_2 - *Sensor failure in* P_v: as a sensor failure, every PC should be sensitive to this fault if P_v is an input or an output. Additionally, Eq. (26) at Section 4.4 is used to estimate P_v, hence all PCs containing this equation will be sensitive to this fault.

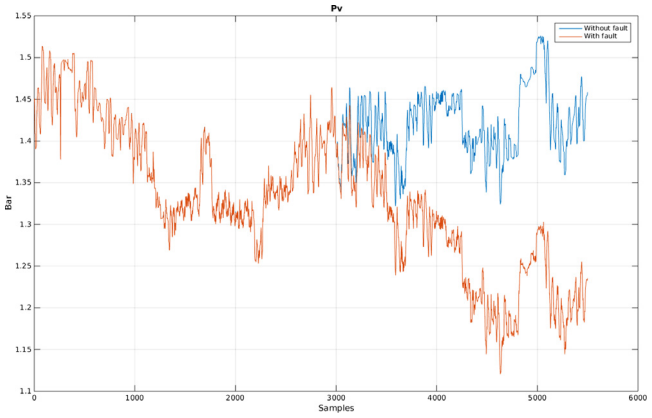

Fig. 20. Pv signal: original and affected by steam leak fault.

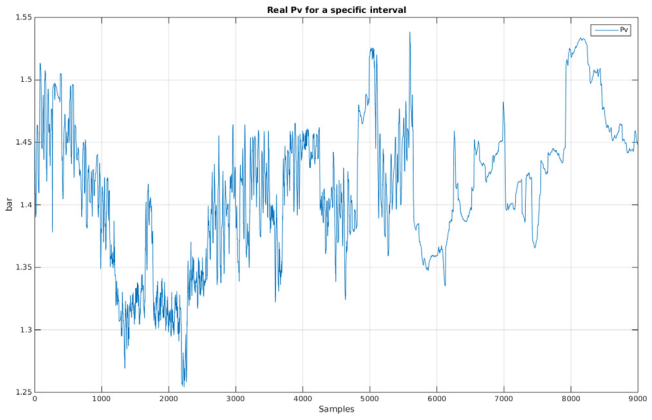

Fig. 21. Pv sensor output from real data.

Table 5
Relation between faults and equations in the model.

Fault	Equations
F_1	(28)
F_2	(26)

- We cannot link fault F_3 to any equation, because of its unknown origin. We can only guess that those PCs containing P_v as an input or as an output could be sensitive to this fault. Due to this uncertainty, we propose to mark with X's the corresponding entries in the Theoretical Fault Signature Matrix (TFSM), instead of the classical 1 which represents certainty.

Table 5 summarizes the relation between faults and equations in the system:

As a consequence, the relation between faults and PCs as represented in the TFSM can be seen in Table 6. An entry $tfsm_{i,j} = 0$ in that table means that PC_j is not sensitive to fault F_i. Conversely, an entry $tfsm_{i,j} = 1$ in that table means that PC_j must be sensitive to fault F_i. Finally, an entry $tfsm_{i,j} = X$ in that table means that PC_j could be sensitive to fault F_i or not. Hence, both 1 or 0 values are allowed.

6.3. Results

The performance of each PC for each one of these faults is shown next, computing the value of the residuals, and finally applying a statistical test to determine if there was a fault detection or not.

Table 6
Relation between faults and PCs: the Theoretical Fault Signature Matrix.

Faults/Pcs	PC_1	PC_2	PC_4	PC_5	PC_6	PC_7	PC_8
F_1	0	0	1	0	1	0	0
F_2	0	1	1	1	1	0	1
F_3	0	X	X	X	X	0	X

Table 7
Real Fault Signature Matrix for the case study showing the activation times for each PC and each fault. FN indicates a false negative and FP a false positive.

Faults/Pcs	PC_1	PC_2	PC_4	PC_5	PC_6	PC_7	PC_8
F_1	0	0	3405	4000 (FP)	3460	0	3108 (FP)
F_2	0	3019	3019	3062	0 (FN)	0	3145
F_3	0	5699	5757	0	0	0	0

For each of the faults, activated at sample 3000, a residual is computed as the difference between the output of the PC model and the output of the process (whose output has been modified as a consequence of the fault as predicted by the simulator for faults F_1 and F_2).

In this work, for robust fault detection, we used the statistical Z-test with a set of sliding windows. In the Z-test, a small window, W_2, is used to estimate the current mean of a residual signal. The variance of the nominal residual signal is then computed using a large window W_1 preceding W_2, by a buffer W_{delay} that is meant to ensure that W_1 contains no samples after fault occurrence. A user-specified confidence level determines the bounds for a two-sided Z-test that are used to compute the fault detection thresholds. The parameters W_1, W_2, W_{delay}, and the z bounds must be tuned to optimize performance. The W_1 window must be large enough to accurately compute the nominal residual variance, but constrained by memory requirements. The W_2 window must be large enough to accurately compute the mean, but small enough to respond quickly to faults. This technique is described in further detail in Biswas et al. (2003). For the case study discussed in this paper, we empirically determined the window sizes as 2500 for W_1, 100 for W_2, and 50 for W_{delay}. The confidence level is 5%. These window values ensure a robust fault detection decision, considering the slow dynamics of the evaporation system.

Table 7 shows the results on the three fault scenarios for our proposal. The values in the table correspond to the activation times (0 if there is no activation) returned after applying one Z-test for each PC on each faulty scenario. Consequently, Table 7 represents the real Fault Signature Matrix. We discuss these results for each faulty scenario.

6.3.1. Steam leak

This fault is simulated starting at sample 3000. As compared with Table 6, Table 7 shows good performance although two false positive cases appear, corresponding to PC_5 and PC_8. This two PCs are related because both of them predict T_{jo} as output and, as seen at Section 5.1.8, they share a strong similarity. The reason for these false positive comes from the fact that the steam leak fault forces the process to work in an operating point where the neural network has not been designed for. For example, W_s variable goes up to 230 m^3/h while the neural network training dataset included only values less than 160 m^3/h. It is a big difference that explains the early false positive in PC_8.

As for the other PCs that correctly trigger a fault (PC_4 and PC_6), for instance, Fig. 22 shows the output of the neural model corresponding to PC_6 as compared with data from the process, as well as the residual. The effect of the fault is clear just looking at the drift in the residual. This deviation could also be seen on PC_4. For these two models, the Z-test was able to detect the steam leak at $t = 3405$ sample (PC_4) and at $t = 3460$ sample (PC_6).

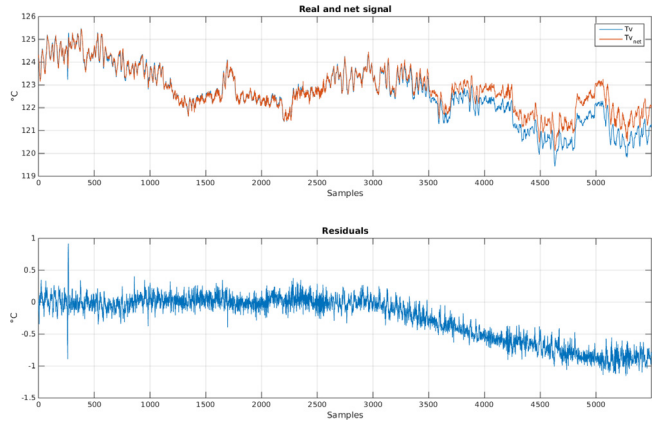

Fig. 22. Comparison between the neural model corresponding to PC_6 and process data. Steam leak starts at sample 3000.

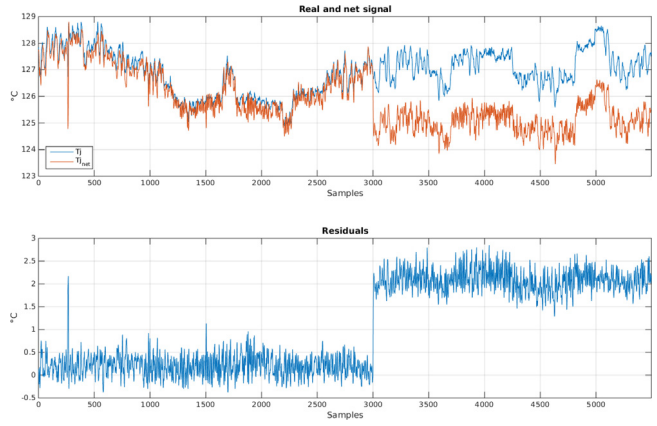

Fig. 23. Comparison between the neural model corresponding to PC_5 and process data for T_{jo}. Sensor failure occurs at sample 3000.

6.3.2. Sensor failure

A bias in the P_v sensor is simulated from sample 3000, just subtracting 0.2 bar to the real signal. As Table 7 reflects, this affects all PCs except PC_1 and PC_7. For every model, detection is rather fast. As an example, Fig. 23 shows the output of the neural model corresponding to PC_5 as compared with process data. In this particular case, the Z-test was able to detect the sensor fault at $t = 3062$ sample.

Table 7 also shows (F_2 row) that a false negative appears for PC_6 model. This false negative can be justified analysing the sensitivity of the model output (T_v) with respect to the inputs: a 1% increase in the P_v input causes an increment of only 0.13 °C in T_v. In contrast, the input with more correlation for this model is T_{jo}: a 1% increase in T_{jo} causes an increment of 1.12 °C in T_v. For this reason, the responsiveness of PC_6 to changes in P_v is almost null and it explains the false negative in this case; theoretically P_v is an input to the model, but in practice its influence is negligible for this model.

6.3.3. Unknown fault on P_v

As it was mentioned before, an anomaly was detected on the dynamics of the P_v sensor (see Fig. 21) for the real data collected at the factory. As Table 7 reflects (F_3 row), it is detected by PC_2 and PC_4.

PC_2 is sensitive to faults in Eq. (24), and sensors T_v and P_v.

PC_4 is sensitive to faults in Eqs. (11), (12), (17), (22), (26), (27), and (28), and faults in sensors T_v, P_v, W_{ji}, W_{jo}, T_{jo}, and L.

Since we have provided X as entries in the TFSM we analyse the results using the DX approach for fault isolation. In that approach, residuals are analysed row-wise, instead of column-wise (Cordier et al., 2004). Once residual activation is detected, the diagnosis approach computes the potential fault candidates given the structural information related to each PC. Using a minimal-hitting set algorithm, it is possible to automatically provide the set of multiple fault candidates. In our case study, simultaneous activation points towards faults in sensors T_v or P_v as the only single fault candidates. Any other cause for activation can be explained as double faults (combination of fault in (24) and any fault from PC_4): there are 11 double fault candidates. It is a common procedure to prefer single-fault over double-faults diagnoses, hence faults in P_v or T_v sensors would perfectly fit as explanations for these abnormal data.

Nevertheless, a more detailed analysis would be necessary to find the causes of the anomaly in the sensor and evaluate if it is important enough for triggering an alarm to the operator.

6.4. Discussion of results in the case study

Looking at the results in Table 4 and comparing the RMSE in Table 1 we can conclude that the proposal to use structural and behavioural information to automatically derive information to develop grey-box neural models works well. These results have shown that our proposal allows to develop smaller models, in terms of unknown parameters to be estimated, while keeping the accuracy in the estimation. Regarding fault detection capabilities, in our case study we have found that our proposal works well in 18 out of 21 cases (85.7%), with only 2 false positives (9.5%) and one false negative (4.8%).

The presence of false positives in our approach, specially for F_1, due to the neural model working in a new operation point is due to the well known limitation for neural networks, that they are not able to extrapolate a new behaviour; they only shows good performance in the range in which they have been trained. A possible solution is to get data for a wider range of operating points, but this is quite complicated in a real continuous process where the operators always try to maintain all the variables in a narrow range because that is the optimal way of operating the process.

The presence of a false negative for F_2 makes clear that it is necessary to perform a sensitivity analysis for each potential fault and each PC. This would benefit the fault isolation process, changing the binary information in the TFSM for another entry that explains each residual sensitivity, thus leading to some kind of enhanced Fault Signature Matrices (Koscielny, 1995; Puig et al., 2005).

Finally, even for these three diagnosis scenarios, the diagnosis proposal has shown that it does not provide false alarms in absence of faults (there is no false residual activation before the 3000 time point in any of the 21 cases). And even more important, it is able to detect unexpected faults due to the usage of models of correct behaviour.

7. Related work

Previous works on fault diagnosis of an evaporation section were developed using knowledge-based approaches (Alonso González et al., 1998; Alonso Gonzalez et al., 2001) given the amount of experience available. However, these approaches cannot be easily generalized and have a strong dependency on continuous knowledge model update. On the other hand, consistency-based diagnosis techniques have the advantage of requiring only models of correct behaviour that are more easily parameterizable and they can be used in more than one factory, just by reusing the models (Hamscher et al., 1992).

In this work we have proposed to combine a model-compilation technique to find the complete set of minimal analytically redundant subsystems, using the Possible Conflict concept, with ssNNs, which provide grey-box type models capable to track dynamic system behaviour.

Possible Conflicts have been used before with first-principle models to track and to perform model-based diagnosis in dynamic systems: using simulation models (Pulido et al., 2001), state-space observers, and a combination of both (Bregon et al., 2014). In all the cases explicit

causal assignments for each equation in the model was required to build the executable model capable to track the system. Main difference with this approach is that using ssNN we can overcome this limitation and just use the structural model to build the ssNN model.

Possible Conflicts were also combined with statistical methods based on Principal Components Analysis to perform fault detection and isolation (Garcia-Alvarez et al., 2011) of dynamic systems. While in that work we used PCs to determine subsets of measurements that were physically and structurally dependants, there were no additional information between the models and the measurements. In this work we use the explicit information about structure and causality in the original equations to design a ssNN which resembles a grey-box model for a state-space observer.

The dynamic neural network used in this work, i.e. ssNN, was presented in Zamarreño and Vega (1998) and used for modelling of non-linear processes in Zamarreño and Vega (1999) and Zamarreño et al. (2000). However, in those works, no structural information about the processes were used in the design of the ssNN besides the number of inputs, outputs and eventually number of states. In this sense, the neural network was considered as a black box model and specific relationships between states and inputs and between outputs and states were deduced indirectly in the training process. A high computation cost arises from this lack of knowledge, resulting in long training times. In this work, the structural information of the model has been included in the structure of the ssNN, making the training process easier to be carried out by the training algorithm and using these known relationships for obtaining a grey-box model more suitable for representing the process.

State space neural networks for fault diagnosis has been presented by other authors, i.e. Czajkowski et al. (2014) and Czajkowski and Patan (2011), but as they use the model like a black box, a simplified structure is considered where a linear relationship is assumed in the observation equation. In our case, we prefer to use a non-linear relationship in the observation equation because it describes in a better way the inner structure of the model as reflected by the first principles model. As it was justified in the previous paragraph, if the neural network model structure follows the structure of the physical model, training would be easier and its generalization capability improved as a grey-box model is a more precise representation of the system.

Recent works have also used different kinds of black-box models, including Recurrent Neural Networks (RNN) (Sorrentino et al., 2014; Pohjoranta et al., 2015), or Supervised Self-Organizing Maps (Wu and Liu, 2015), to perform fault diagnosis on dynamic systems. These techniques are used alone, or in combination with other model-based or statistical/machine-learning approaches, for fault isolation. Main difference with our proposal is the presence of structural information derived from the analytical models to link faults and models automatically, allowing fault isolation, and also providing precise guidelines for building the internal model for the state space Neural Network.

For dealing with model uncertainty and measurement noise, a great issue when working with real processes, Witczak et al. (2015) use a robust state space GMDH neural network that improves fault detection in sensor and actuator faults. Based on Marcin (2013), a dynamic neuron model is built to represent the dynamic nature of the process. In contrast, our proposal deals with model uncertainty assuming an expert knowledge of the structure of the physical model in the form of DAE equations.

The usage of grey-box models mixing structural models and parameters learnt from data, as proposed in this work, is an alternative to avoid model linearization or to rely upon intervalar models to deal with parameter uncertainty as proposed by Rotondo et al. (2016) and references therein.

The combination of model-based diagnosis and data-driven models is different from the typical combination of model-based diagnosis methods coupled with machine-learning techniques (Alonso et al., 2004; Costamagna et al., 2016). In this case we use the data from the process to learn the parameters of grey-box models, conceptually closer to parameter estimation in FDI, than to learn classification models that can be used to refine the fault candidates once the model-based approach is used for fault-detection. With that said, if enough data are available for using these techniques, they could be also coupled with our approach, as long as they are used to confirm the results from the model-based fault detection and isolation stages.

Finally, the current work extends a former version presented in Pulido et al. (2012) in almost every aspect. First, we provide a complete detailed model of the evaporator used to build the diagnosis model. We also have extended the description of the Possible Conflict concept, including a precise characterization of the model structure to ease the definition of the algorithm required to design the state space Neural Networks. The new diagnostic model provides a new set of minimally analytical redundant subsystems which are completely described, together with their corresponding state space Neural Networks. We have also included a more robust statistical fault-detection method. Finally, the real data used in this work are different, both for the nominal and the faulty situations. In this case we have generated two synthetic faults using real data as inputs because the system is tightly controlled. But additionally we have found an unexpected fault, acknowledged by plant operators, within the real data.

8. Conclusions

In this work we have proposed to use Possible Conflicts to decompose a large system model into smaller models with minimal redundancy for fault detection and isolation. Possible Conflicts provide the structural models (equations, inputs, outputs, and state variables) required for model-based fault detection and isolation, which can be later implemented as simulation or state-observer executable models. Since deriving such models for complex non-linear models is not straightforward and requires the participation of modelling experts, we have proposed to use the structural information in the model to design a neural network grey-box model using a state space architecture.

The main conclusion is that the structure of the Minimal Evaluable Model for a Possible Conflict can guide the design of the state space model of the neural network, reducing its complexity and avoiding the process of multiple unknown parameter estimation in the first principles models. Comparing results of this approach in an evaporation unit of a beet sugar factory we have observed that the ssNN is able to obtain similar or even better results than a simulation model manually derived by an expert. Both types of models were used to successfully monitor the process and to detect faults.

The ssNN models have proven to be capable to detect small/synthetic faults, but also capable to detect anomalies that were undetected by the plant operators. The main drawback found so far are sensitivity problems that should be analysed in future works. Obtaining real data in different operation points for assuring good representation of the system in the overall working range is an issue for any kind of neural model, but we can still argue that we have had promising results, obtaining correct detection results in 85.7% of the cases. The difficulties found in the fault detection stage are due to the use of real data in a slow dynamics continuous process that is very well controlled, where it is really difficult to find real measurements related to faulty situations. We plan to test the ssNN models on a larger experiment data-set; hopefully with richer behaviours since one of the main handicaps we had to cope with was the low variability of the real data sets, a common problem when working with data from real continuous processes. Another option would be to provide a method to analyse those cases where the PCs do provide a false positive and use those data, usually related to new operation modes, to re-train the networks.

Since the proposed method relies upon structural information about the models and data availability to tune the grey-box models provided by the PCs, it is clear that it is a generalizable proposal and it can be used in many different domains.

As future work we plan to develop grey-box ssNN models for those PCs without a MEM. Those models has no global valid causal assignment

Table A.1
Nomenclature.

Symbol	Description	Units
$_{ji}$	Sub-index used for juice entering the juice chamber	–
$_{jo}$	Sub-index used for juice leaving the juice chamber	–
$_{ev}$	Sub-index used for steam evaporated from the juice	–
$_{v}$	Sub-index used for steam leaving the juice chamber	–
$_{n}$	Sub-index used for steam in the next evaporator	–
$_{s}$	Sub-index used for steam entering the steam chamber	–
m	Mass of juice	kg
W	Mass flow	kg/s
c	Brix of the juice	weight %
Pu	Purity of the juice	weight %
T	Temperature	°C
P	Pressure	bar
V	Volume	m^3
H	Enthalpy	kJ/kg
Q	Heat flow from the steam entering the steam chamber to the juice	kW
m_{jmax}	Maximum mass of juice in the juice chamber	kg
U	Overall heat transfer coefficient	$W/(m^2\ K)$
S	Heat transfer surface	m^2
λ_s	Latent heat of the condensing steam	kJ/kg
Op	Valve opening	%
Cv	Flow coefficient for the valve	$m^3/s\ bar^{-1}$
k_{w1}, k_{w2}	Adjustment parameters	$kg/s\ bar^{-1}$
L	Juice level	%
L_{ref}	Reference for the juice level	%
$e(t)$	Level controller output error	%
t	time	s
τ	Variable for time integration	s
K_p	Proportional coefficient of the PID regulator	output units · input $units^{-1}$
T_i	Integral time constant of the PID regulator	s
T_d	Derivative time constant of the PID regulator	s^{-1}
ρ	Juice density	kg/m^3
g	Gravitational constant	m/s^2
h	Height of juice in the evaporator	m

for every equation in a MEC. We need to derive a new set of guidelines for building the ssNN structure.

Additionally we plan to use the structural information from the PCs to analyse the sensitivity of the different residuals to different faults. This would produce an enhanced Fault Signature Matrix as in Koscielny (1995) or Puig et al. (2005), and also it could help to tailor the detection process for each PC. In this initial proposal we have used the same values for the Z-test used in the fault detection stage.

We need to test the approach at different times of the season, because this is a very slow evolving process whose parameters vary over time. Moreover, we can test more abstract models that will produce fewer PCs, but still containing the same structural information, for instance using another set of modelling assumptions. Finally, once we introduce larger data sets, we will use statistical tests to perform fault detection, and to determine the threshold to guarantee a minimum percentage of false positives and false negatives, where the sensitivity of the inputs would be one of the factors to take into account for effective detection.

Appendix. Notation

See Table A.1.

References

Acebes, L., Merino, A., Alves, R., de Prada, C., 2009. Online energy diagnosis of sugar plants (in spanish in the original). RIAI - Rev. Iberoam. Autom. Inform. Ind. 6 (3), 68–75.

Alonso, C.J., Rodríguez, J.J., Pulido, B., 2004. Enhancing consistency based diagnosis with machine learning techniques. In: Current Topics in Artificial Intelligence. Springer, pp. 312–321.

Alonso González, C., Pulido Junquera, B., Acosta, G., 1998. On line industrial diagnosis: an attempt to apply artificial intelligence techniques to process control. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, pp. 804–813.

Alonso Gonzalez, C., Pulido Junquera, B., Lazo, G.A., Bello, C.L., 2001. On-line industrial supervision and diagnosis, knowledge level description and experimental results. Expert Syst. Appl. 20 (2), 117–132.

Antoine, C., 1888. Tensions des vapeurs; nouvelle relation entre les tensions et les températures. C. R. Séances Acad. Sci. 107, 681–684, 778–780, 836–837.

Armengol, J., Bregon, A., Escobet, T., Gelso, E., Krysander, M., Nyberg, M., Olive, X., Pulido, B., Travé-Massuyès, L., Minimal structurally overdetermined sets for residual generation: a comparison of alternative approaches, in: Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS09, Barcelona, Spain, pp. 1480–1485.

Asadi, M., 2007. Wiley InterScience (Online service). Beet-sugar handbook. Wiley-Interscience, p. 866.

Biswas, G., Simon, G., Mahadevan, N., Narasimhan, S., Ramirez, J., Karsai, G., 2003. A robust method for hybrid diagnosis of complex systems. In: Proceedings of the 5th Symposium on Fault Detection, Supervision and Safety for Technical Processes, pp. 1125–1131.

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., 2006. Diagnosis and Fault-Tolerant Control. Springer.

Bregon, A., Alonso-González, C.J., Pulido, B., 2014. Integration of simulation and state observers for online fault detection of nonlinear continuous systems. IEEE Trans. Syst. Man Cybern. A 44 (12), 1553–1568.

Bregon, A., Biswas, G., Pulido, B., Alonso-González, C., Khorasgani, H., 2014. A common framework for compilation techniques applied to diagnosis of linear dynamic systems. IEEE Trans. Syst. Man Cybern. A 44 (7), 863–876. http://dx.doi.org/10.1109/TSMC.2013.2284577.

Bubnik, Z., Kadlec, P., Urban, D., M., B., 1995. In: Bartens (Ed.), Sugar technologists manual: Chemical and physical data for sugar manufacturers and users.

Chantler, M., Daus, S., Vikatos, T., Coghill, G., 1996. The use of quantitative dynamic models and dependency recording engines. In Proceedings of the Seventh International Workshop on Principles of Diagnosis, DX96, Val Morin, Quebec, Canada. pp. 59–68.

Cordier, M., Dague, P., Lévy, F., Montmain, J., Travé-Massuyès, M.S.L., 2004. Conflicts versus analytical redundancy relations: a comparativeanalysis of the model-based diagnosis approach from the artificial intelligence and automatic control perspectives. IEEE Trans. Syst. Man Cybern. B 34 (5), 2163–2177.

Costamagna, P., De Giorgi, A., Gotelli, A., Magistri, L., Moser, G., Sciaccaluga, E., Trucco, A., 2016. Fault diagnosis strategies for SOFC-based power generation plants. Sensors 16 (8), http://dx.doi.org/10.3390/s16081336.

Czajkowski, A., Patan, K., 2011. Robust fault detection and accomodation of the boiler unit using state space neural networks. In: Pomiary, Automatyka, Kontrola, Vol. 57. pp. 1428–1435.

Czajkowski, A., Patan, K., Szymaki, M., 2014. Application of the state space neural network to the fault tolerant control system of the PLC-controlled laboratory stand. Eng. Appl. Artif. Intell. 30, 168–178. http://dx.doi.org/10.1016/j.engappai.2014.01.017.

Dressler, O., 1994. Model-based diagnosis on board: Magellan-MT inside. Working Notes of the International Workshop on Principles of Diagnosis, DX94, Goslar, Germany.

Dressler, O., 1996. On-line diagnosis and monitoring of dynamic systems based on qualitativemodels and dependency-recording diagnosis engines. In: Proceedings of the Twelfth European Conference on Artificial Intelligence, ECAI-96. John Wiley and Sons, New York, pp. 461–465.

Dressler, O., Struss, P., 1996. The consistency-based approach to automated diagnosis of devices. In: Brewka, G. (Ed.), Principles of Knowledge Representation. CSLI Publications, Standford, pp. 269–314.

Empresarios Agrupados Internacional, 2012. EcosimPro, Madrid, Spain. http://www.ecosimpro.com/.

Engell, S., 2007. Feedback control for optimal process operation. J. Process Control 17 (3), 203–219. http://dx.doi.org/10.1016/j.jprocont.2006.10.011.

Frisk, E., Dustegor, D., Krysander, M., Cocquempot, V., 2003. Improving fault isolability properties by structural analysis of faulty behavior models: application to the damadics benchmark problem. In: Proceedings of SAFEPROCESS-2003, Washington, DC, USA.

Garcia-Alvarez, D., Bregon, A., Fuente, M.J., Pulido, B., 2011. Improving parameter estimation using minimal analytically redundant subsystems. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on. IEEE, pp. 7788–7793.

Gertler, J., 1998. Fault detection and diagnosis in Engineering Systems. Marcel Dekker, Inc., Basel.

González-Lanza, P., Zamarreño, J., 2002. A hybrid method for training a feedback neural network. In: First International ICSC-NAISO Congress on Neuro Fuzzy Technologies NF 2002, Havana - Cuba.

González Lanza, P., Zamarreño, J., 2002. A short-term temperature forecaster based on a state space neural network. Eng. Appl. Artif. Intell. 15 (5), 459–464. http://dx.doi.org/10.1016/S0952-1976(02)00089-1.

Hamscher, W., Console, L., de Kleer, J. (Eds.), 1992. Readings in Model based Diagnosis. Morgan-Kaufmann Pub., San Mateo.

de Kleer, J., Williams, B., 1987. Diagnosing multiple faults. Artif. Intell. 32, 97–130.

Koscielny, J.M., 1995. Fault isolation in industrial processes by the dynamic table of states method. Automatica 31 (5), 747–753. http://dx.doi.org/10.1016/0005-1098(94)00147-B.

Marcin, M., 2013. An unscented Kalman filter in designing dynamic GMDH neural networks for robust fault detection. In: amcs, vol. 23. p. 157. http://dx.doi.org/10.2478/amcs-2013-0013.

Merino, A., 2008. Librería de modelos del cuarto de remolacha de una industria azucarera para un simulador de entrenamiento de operarios (Ph.D. thesis), Universidad de Valladolid.

Merino, A., Alves, R., Acebes, L., 2005. A training simulator for the evaporation section of a beet sugar production process. In: Proceedings of the 2005 European Simulation and Modelling conference.

Mosterman, P.J., Biswas, G., 1999. Diagnosis of continuous valued systems in transient operating regions. IEEE Trans. Syst. Man Cybern. A 29 (6), 554–565.

Pantelides, C., 1988. The consistent initialization of differential-algebraic systems. SIAM J. Sci. Stat. Comput. 9 (2), 213–231. http://dx.doi.org/10.1137/0909014.

Patton, R.J., Frank, P.M., Clark, R.N., 2000. Issues in Fault Diagnosis for Dynamic Systems. Springer Verlag, New York.

van der Poel, P.W., Schiweck, H.M., Schwartz, T.K., 1998. Sugar Technology: Beet and Cane Sugar Manufacture. URL https://books.google.es/books/about/Sugar_Technology.html?id=nwjwAAAACAAJ&pgis=1.

Pohjoranta, A., Sorrentino, M., Pianese, C., Amatruda, F., Hottinen, T., 2015. Validation of neural network-based fault diagnosis for multi-stack fuel cell systems: Stack voltage deviation detection. In: 69th Conference of the Italian Thermal Engineering Association, ATI 2014. Energy Proc. 81, 173–181.

Puig, V., Quevedo, J., Escobet, T., Meseguer, J., 2006. Toward a better integration of passive robust interval-based FDI algorithms. In: Proceedings of the 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS062006, Beijing, China.

Puig, V., Quevedo, J., Escobet, T., Pulido, B., 2005. On the integration of fault detection and isolation in model-based fault diagnosis. In: Proceedings of the 16th International Workshop on Principles of Diagnosis (DX-05). pp. 227–232.

Pulido, B., Alonso-González, C., 2004. Possible conflicts: a compilation technique for consistency-based diagnosis. IEEE Trans. Syst. Man Cybern. B 34 (5), 2192–2206.

Pulido, B., Alonso-González, C., Acebes, F., 2001. Consistency-based diagnosis of dynamic systems using quantitative models and off-line dependency-recording. In 12th International Workshop on Principles of Diagnosis (DX-01), Sansicario, Italy, pp. 175–182.

Pulido, B., Bregon, A., Alonso-González, C., 2010. Analyzing the influence of differential constraints in possible conflict and ARR computation. In: Current Topics in Artficial Intelligence, CAEPIA 2009 Selected Papers. Springer-Verlag Berlin.

Pulido, B., Zamarreño, J.M., Merino, A., Bregón, A., 2012. Using structural decomposition methods to design gray-box models for fault diagnosis of complex industrial systems: a beet sugar factory case study. In: Proceedings of European Conference of the Prognostics and Health Management Society, PHME12.

Reiter, R., 1987. A theory of diagnosis from first principles. Artificial Intelligence 32, 57–95.

Rotondo, D., Fernandez-Canti, R.M., Tornil-Sin, S., Blesa, J., Puig, V., 2016. Robust fault diagnosis of proton exchange membrane fuel cells using a Takagi-Sugeno interval observer approach. Int. J. Hydrogen Energy 41 (4), 2875–2886. http://dx.doi.org/10.1016/j.ijhydene.2015.12.071.

Solis, F., Wets, R.J.-B., 1981. Minimization by random search techniques. Math. Oper. Res. 6, 19–30.

Sorrentino, M., Marra, D., Pianese, C., Guida, M., Postiglione, F., Wang, K., Pohjoranta, A., 2014. On the use of neural networks and statistical tools for nonlinear modeling and on-field diagnosis of solid oxide fuel cell stacks. In: ATI 2013 - 68th Conference of the Italian Thermal Machines Engineering Association. Energy Proc. 45, 298–307. http://dx.doi.org/10.1016/j.egypro.2014.01.032.

Staroswiecki, M., 2007. A structural view of fault-tolerant estimation. In: Proc. of IMechE. Part I: J. Systems and Control Engineering, vol. 221.

Travé-Massuyès, L., Milne, R., 1997. Gas-turbine condition monitoring using qualitative model-based diagnosis. IEEE Expert 12 (3), 22–31.

Witczak, M., Mrugalski, M., Korbicz, J., 2015. Towards robust neural-network-based sensor and actuator fault diagnosis: Application to a tunnel furnace. Neural Process. Lett. 42 (1), 71–87. http://dx.doi.org/10.1007/s11063-014-9387-0.

Wu, X., Liu, H., 2015. Fault diagnosis of solid oxide fuel cell based on a supervised self-organization map model. J. Fuel Cell Sci. Technol. 12 (3), http://dx.doi.org/10.1115/1.4029070.

Zamarreño, J., Vega, P., 1997. Identification and predictive control of a melter unit used in the sugar industry. Artif. Intell. Eng. 11 (4), 365–373. http://dx.doi.org/10.1016/S0954-1810(96)00055-6.

Zamarreño, J., Vega, P., 1998. State space neural network. Properties and application. Neural Netw. 11 (6), 1099–1112.

Zamarreño, J., Vega, P., 1999. Neural predictive control. Application to a highly non-linear system. Eng. Appl. Artif. Intell. 12 (2), 149–158. http://dx.doi.org/10.1016/S0952-1976(98)00055-4.

Zamarreño, J., Vega, P., García, L., Francisco, M., 2000. State-space neural network for modelling, prediction and control. Control Eng. Pract. 8 (9), 1063–1075. http://dx.doi.org/10.1016/S0967-0661(00)00045-9.