A FRAMEWORK FOR ON-LINE TREND EXTRACTION AND FAULT DIAGNOSIS

Mano Ram Maurya [*] **Raghunathan Rengaswamy** [**]
Venkat Venkatasubramanian [*,1]

[*] *Laboratory for Intelligent Process Systems, School of Chemical Engineering, Purdue University, West Lafayette, IN 47907, USA.*
[**] *Department of Chemical Engineering, Clarkson University, Potsdam, NY 13699-5705, USA.*

Abstract: Qualitative trend analysis (QTA) is a process-history-based data-driven technique that works by extracting important features (trends) from the measured signals and evaluating the trends. QTA has been widely used for process fault detection and diagnosis. Recently, Dash *et al.* (2001, 2003) presented an interval-halving-based algorithm for off-line automatic trend extraction from a record of data, a fuzzy-logic based methodology for trend-matching and a fuzzy-rule-based framework for fault diagnosis (FD). In this article, an algorithm for on-line extraction of qualitative trends is proposed. A framework for on-line fault diagnosis using QTA also has been presented. Some of the issues addressed are - (i) development of a robust and computationally efficient QTA-knowledge-base, (ii) fault detection, (iii) estimation of the fault occurrence time, (iv) on-line trend-matching and (v) updating the QTA-knowledge-base when a novel fault is diagnosed manually. Some results for FD of the Tennessee Eastman (TE) process using the developed framework are presented. *Copyright© 2003 IFAC.*

Keywords: Qualitative Trend Analysis, On-Line, Fault Detection, Fault Diagnosis.

1. INTRODUCTION

Modern plants are data rich and information poor. Vast amount of process data is available which can be used to assess the process state by utilizing the important features present in the measured data. Qualitative trends (e.g. increasing, constant etc.) are the most natural representation of features that have been widely used for FD. Every diagnostic system that uses process trends to achieve fault classification has three components: (i) a language for trend representation such as triangular episodes (Cheung and Stephanopoulos, 1990), primitive-based language (Janusz and Venkatasubramanian, 1991) and piecewise-linear elements (Mah *et al.*, 1992), (ii) a methodology to extract the trends such as wavelet-based method (Bakshi and Stephanopoulos, 1994*a*), use of wavelet, neural networks and B-Splines-based method (Vedam, 1999) and (iii) a classification methodology and a knowledge-base to map the sensor-trends into faults such as decision trees (Bakshi and Stephanopoulos, 1994*b*), weighted symptom trees (WST) (Oh *et al.*, 1997) and fault or sensor centric trees (Vedam, 1999). Recently Dash *et al.* (2001) proposed an interval-halving-based algorithm for automatic trend extraction. The primitive-based language (Janusz and Venkatasubramanian, 1991) is used for the representation of the qualitative trends. The seven primitives, *viz.* A(0, 0), B(+, +), C(+, 0), D(+,

[1] Author to whom all correspondences should be addressed, E-mail: venkat@ecn.purdue.edu, phone: (765) 494 0734, fax: (765) 494 0805.

-), E(-, +), F(-, 0), G(-, -) where the signs are of the first and second derivatives, respectively, are shown in Figure 1. The primitives B, D, E and G are nonlinear primitives. Dash *et al.* (2003) also developed a fuzzy-logic-based framework for trend-matching and fault diagnosis. The overall activity of trend-extraction and trend-matching is called qualitative trend analysis (QTA).

While the interval-halving algorithm, the trend-matching methodology and their application for FD have been discussed in detail (Dash *et al.*, 2001; Dash *et al.*, 2003), on-line implementation has not been discussed. Though the research work by Vedam (1999) dealt with some of the issues involved in on-line implementation of trend-similarity-based FD, little has been discussed in the published literature. In this article, we discuss most of the important issues that are involved in on-line fault diagnosis. In particular, we present an algorithm for on-line trend extraction using the interval-halving technique to achieve computational efficiency and robustness. A framework for on-line FD using QTA is also presented. The organization of this article is as follows.

In the next section, an overview of QTA is presented. In section 3, we motivate the need for an on-line variant of the interval-halving algorithm and discuss some of the challenges. We also list the activities that are carried out in on-line FD using QTA. In section 4, the design of the algorithm for the on-line variant is discussed. The framework for on-line FD is discussed in sections 5 and 6. Section 5 deals with development of a knowledge-base (KB) for QTA. Section 6 deals with the use of the QTA-KB for on-line FD. In section 7, a succinct discussion on the development of a prototype diagnostic system in Matlab® is presented. Sample results of FD in the TE process are also presented. Finally we conclude this article with discussion on future work.

2. OVERVIEW OF QTA

There are two subtasks in QTA- (i) trend extraction (Dash *et al.*, 2001) and (ii) trend matching (Dash *et al.*, 2003). A brief discussion follows.

Trend extraction by using the interval-halving algorithm: The algorithm works by fitting either a constant, a first order or a second order polynomial (in that order) to the data and halving the interval if the fit error is significant as compared to the noise present in the signal (as dictated by the F-test) even for the quadratic function. Once the polynomial is fitted over a certain interval, a primitive is assigned based upon the sign of the first and second derivatives (t-test is used to check the significance of the derivatives).

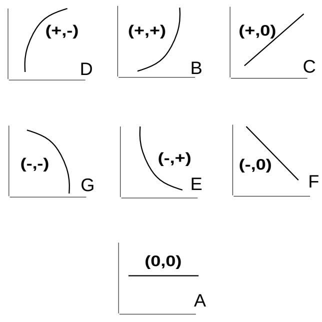

Fig. 1. Fundamental language: primitives

Then the interval-halving procedure is applied to the remaining data till the entire signal is transformed into a sequence of primitives.

Fuzzy trend-matching: Trend-matching involves calculation of the following: (a) fuzzy similarity match between two primitives, (b) similarity measure between two trends (the time-weighted average of the similarity match between the primitives involved in different time intervals) for the same sensor and (c) multivariate (overall) similarity measure or confidence index (C.I.) between two scenarios (given by $C.I. = min(S_1,\ S_2,\ \ldots,\ S_n)$ where n is the number of sensors and S_k is the similarity measure between the trends of the k^{th} sensors in the two scenarios). To perform fault diagnosis by using QTA, the faults stored in the QTA-KB are rank ordered in decreasing order of their C.I. The fault with the highest C.I. is the fault that has most likely occurred. A low value of C.I. (say below 0.50) for all the faults indicates the occurrence of a novel fault.

3. ON-LINE FAULT DIAGNOSIS USING QTA: MOTIVATION AND ISSUES

Our ultimate aim is to implement the QTA-based fault diagnosis in real plants. In off-line trend extraction, the interval halving algorithm is applied on the entire data. During on-line implementation, more and more data is available from sensor measurements. The primitives obtained at current time may no longer correctly represent the trend at a future time. Thus a key feature of an algorithm for on-line trend extraction is that the primitives should be updated as more data becomes available. On one hand, trend extraction cannot be performed on all the data available so far at every sample time. On the other hand, if one were to find trends corresponding to only the new data, one would end up assigning only ‘A’ primitives since no useful trend is contained in few samples. Thus the data set for trend extraction should comprise of some of the past data and the newly available data. This poses the question of how should one choose the data segment for trend extraction? Some related questions are- (i) should the data segments chosen for two consecutive trend extractions overlap?, (ii) can one calculate an average primitive in the overlapping region?, etc. These issues are discussed in the next section. Further, for real-time FD, we need to consider the following- (i) building a knowledge-base of fault-symptom signatures, (ii) detecting the occurrence of an abnormal event (fault detection), (iii) extracting the relevant portion of trend from an arbitrarily long sequence of primitives (does fault occurrence time play a vital role?), (iv) time-efficient computation of similarity measure and (v) learning- updating the QTA-KB if a novel fault is manually diagnosed by the operator. Discussion on on-line trend extraction follows.

4. ON-LINE TREND EXTRACTION

Given sensor data, the basic interval-halving algorithm can be applied on the entire data in off-line fashion to extract trends. From the above discussion it is clear that the trend extraction has to be performed over a window of data, that the window should move as more data becomes available and that the trend extracted in the current window has to be intelligently combined with the already extracted trends. Before trying to device an algorithm to carry out additional preprocessing (and possibly post-processing) for on-line trend extraction, let us briefly analyze the off-line and on-line implementation of another methodology for trend-extraction *viz.* B-Spline based trend extraction (Vedam, 1999).

To implement the B-Spline based algorithm (which extracts linear primitives) for on-line trend extraction by using a sliding-window approach, off-line algorithm is applied on a window containing 2^k+1 samples (k is a positive integer). After measuring p more samples, the window slides by p data points (the window size does not change). This is similar to the sliding window approach for on-line denoising using wavelet analysis. Now to get a consolidated list of linear primitives till the current time, the primitives in the current window are combined with the old list of consolidated linear primitives. Since the primitives to be combined are linear, an average value can be used at the starting point of the current window. After this time instant, the old primitives are replaced by the primitives in the current window. Every time trends are extracted, after updating the list of linear primitives, they are concatenated to get higher order primitives. The higher order primitives are not updated directly. Now let us analyze the key features of the off-line interval-halving algorithm for trend extraction.

The basic interval-halving algorithm is capable of extracting nonlinear primitives directly and no concatenation is needed. This means that if the interval-halving algorithm identifies a nonlinear primitive then there is no need to allow this nonlinear primitive to evolve further except when the primitive length is very small. This is a very good feature since concatenation requires a parameter (magnitude threshold) for every sensor and is not transparent.

Given that one should preserve the ability to extract nonlinear primitives and avoid concatenation, unlike on-line implementation of B-Spline based methodology, averaging at the starting point of the current window is not an option. In fact, averaging would render the polynomial coefficients (which can be used for data compression) useless. Similarly, concatenation is not allowed. Yet, the primitives should evolve as more data comes in. Two key ideas to achieve this effect are explained below.

Unrestricted evolution of the last primitive of the current window: By the nature of the interval-halving algorithm, as more samples are available, only the last primitive evolves. This assumption may be violated for very small time intervals (where one of the primitives before the last primitive could be an ‘A’ primitive) due to parameter-less and ‘fit the simplest-primitive’ nature of the algorithm. Such local violations do not have much effect on similarity measure.

If the last primitive is a linear primitive (‘A’, ‘C’ or ‘F’) then allow it to evolve until it becomes a nonlinear primitive or it becomes very long (so that further evolution would require too much computation). If the last primitive is a nonlinear primitive then it should not be allowed to evolve except when its length is too small.

Parameters and selection of the window: Three important parameters are the default window length, the length of the shortest primitive (minNLP_len) and the length of the longest linear primitive (maxCFlen). For choosing the window, the endpoint of the current window should coincide with the last sample available. The starting point of the window can be chosen to coincide with the starting point of the last primitive. With this rule in effect, the window size changes adaptively. One exception to this rule is that if the last primitive is nonlinear and is very long then one can choose the starting point of the window so as to keep the window size equal to the default window size. Depending upon how window is selected, one should keep a record of the primitives that would not evolve anymore so that the most updated list of primitives can be obtained simply by appending the primitives in the current (or latest) window to the existing list of non-evolving primitives. The above procedure of window selection is schematically shown in Figure 2. A flowchart for the overall

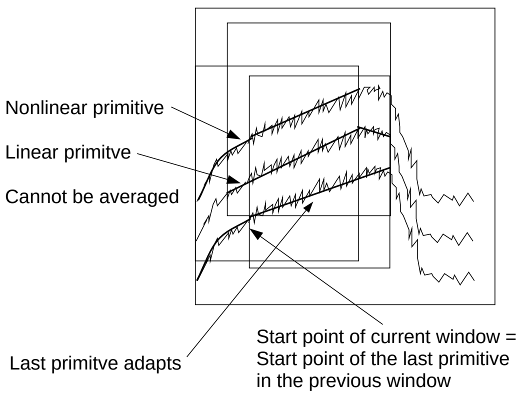
Nonlinear primitive
Linear primitve
Cannot be averaged
Last primitve adapts
Start point of current window = Start point of the last primitive in the previous window

Fig. 2. Adaptive selection of the window

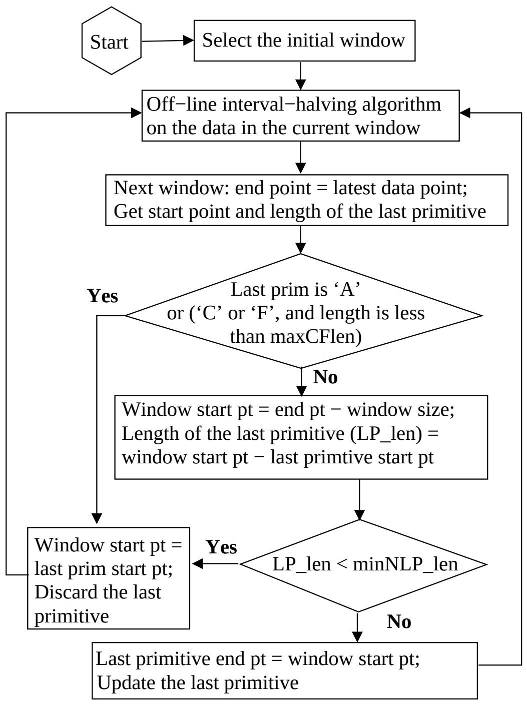
Start
Select the initial window
Off−line interval−halving algorithm on the data in the current window
Next window: end point = latest data point; Get start point and length of the last primitive
Last prim is ‘A’ or (‘C’ or ‘F’, and length is less than maxCFlen)
Yes
No
Window start pt = end pt − window size; Length of the last primitive (LP_len) = window start pt − last primtive start pt
LP_len < minNLP_len
Yes
No
Window start pt = last prim start pt; Discard the last primitive
Last primitive end pt = window start pt; Update the last primitive

Fig. 3. Flowchart for on-line trend-extraction

algorithm is shown in Figure 3. Figure 4 shows a snapshot of trend-extraction in two consecutive windows for a sample signal. If one were to use the interval-halving algorithm on a sliding window of fixed length (65), the last primitive (at the current time) would still have been ‘A’. Also, unnecessary computation would have been performed over a certain portion of the first primitive.

5. BUILDING QTA KNOWLEDGE-BASE

Development of the QTA-KB primarily involves trend-extraction for all known faults. Some of the parameters (which are universal for a given plant) related to QTA-KB are- description of the faults stored (including whether the fault truly corresponds to an abnormal scenario), description of the sensors which are used for trend extraction *viz.* sample rate, normal value, noise etc., the start and endpoints of the data used for trend

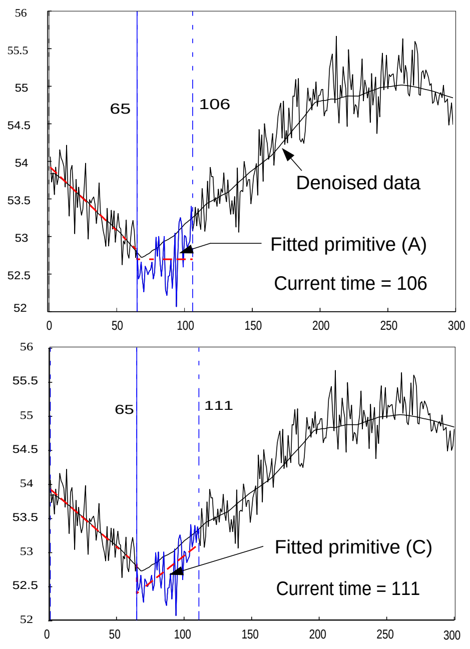

Fig. 4. Snapshots during two consecutive windows

extraction for each sensor in every fault scenario, the global parameters related to on-line trend extraction such as default window length etc. For small-scale plants, where the number of sensors is small (say, less than 100), one may be able to compute similarity measure for all the sensors with all the fault scenarios in real-time but for large plants, it may be infeasible. For large plants, one must consider the issue of computational complexity. Apart from computational infeasibility, it is expected that the sensors that are physically located near the fault origin would show departure from their normal operating region (NOR) before the sensors that are located far away. Hence, such sensors are useful in detecting the fault occurrence and they should be chosen for estimating the similarity measure. Thus optimal selection of sensors (that should be used for calculation of similarity measure) corresponding to each fault is very important. Other issues in optimal selection of sensors are- (i) consistency among similar faults: the sensors should show similar trends for multiple manifestations of the same fault and (ii) discrimination from other faults: the sensors should be chosen so that they provide maximum discrimination from other (different) faults. These ideas have been earlier discussed and implemented by Vedam (1999). Further, when lesser (but sufficient) number of sensors are chosen, most likely the chosen sensor would show fast evolution. This would result in robustness in the calculation of similarity measure, particularly during the incipient stage of fault evolution, the duration in which one is interested in diagnosing the fault. Since the confidence index assigned to a fault is the minimum of the similarity measure for the sensors dedicated to diagnose the fault, if all the sensors are chosen blindly for assessing every fault then correct diagnosis would be delayed. To summarize these ideas, the sensors dedicated for diagnosing various faults should be chosen with respect to three criteria: (i) consistency among similar faults, (ii) fast dynamics for the fault and (iii) discrimination from other faults. The procedure for optimal screening of sensors is discussed below-

[1] Extract trends for all the sensors for all the fault scenarios.

[2] Compute a global similarity matrix containing the similarity measure for each sensor for all pairs of faults. Perform the steps 3-7 for every fault.

[3] Identify the set of faults that are similar to this fault, and the set of faults that are different.

[4] Corresponding to the set of similar faults, for each sensor, extract the similarity measures from the similarity matrix and take the minimum. Rank the sensors in decreasing order of the minimum. Thus the sensor that shows maximum similarity would be on the top of the list (list 1).

[5] Corresponding to the pairs of this fault with the different faults, extract the similarity measures for all sensors, take the maximum for each sensor, and rank the sensors in increasing order of the maximum. Thus the sensors that shows least similarity measure (maximum discrimination) would be on the top of this list (list 2).

[6] Rank the sensors in decreasing order of speed of evolution (the sensor that evolves the fastest should be on the top) (list 3).

[7] Prepare a new list by selecting sensors from list 1 and list 3 (rank them according to a weighted criterion). This new list is a ranked list of sensors in decreasing order of consistency and speed of evolution. If the value of the weighted criterion is equal for two or more sensors in this list then sort them according to list 2 (decreasing discrimination). Now select sensors from the new list one after another till the fault can be resolved from all other (different) faults.

Of course, in the above procedure, a fail-safe approach should be adopted so that sufficient number of sensors are chosen to ensure discrimination from other faults. In some rare cases, where the data used for developing the knowledge-base is not collected properly, consistency with similar faults and discrimination from different faults could be in conflict. In such cases, a robust and reliable knowledge-base cannot be developed.

6. ON-LINE FAULT DIAGNOSIS

During online implementation, the primitives are continuously extracted. When a fault occurs, the sensors start deviating from their normal values. Thus the first step is fault detection. In QTA-based fault detection, the presence of a non-

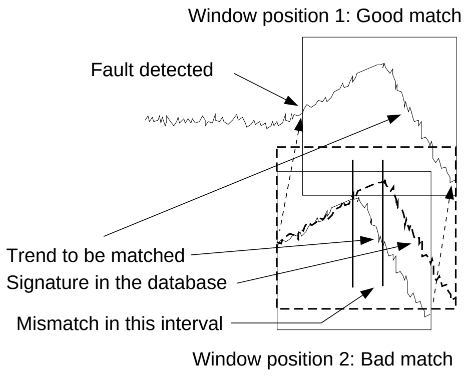
Window position 1: Good match
Fault detected
Trend to be matched
Signature in the database
Mismatch in this interval
Window position 2: Bad match

Fig. 5. Importance of fault occurrence time

A primitive and departure from the NOR indicates presence of a fault. A suitable multivariate methodology also can be used for fault detection. After detection, it is very important to estimate the time at which the fault occurred so that an appropriate portion from the infinite sequence of primitives can be extracted. If the fault occurrence time is not estimated properly then poor similarity measure may be obtained (see Figure 5) even for very similar trends. Due to excessive computational complexity, similarity measures with shifted trends cannot be evaluated in real-time and hence, good estimate of fault occurrence time is required. The methodology used for the estimation of the fault occurrence time is called *backtracking*. As shown in Figure 6, once a fault is detected, we try to fit an ‘A’ primitive in the last interval. If an ‘A’ can be fitted then there is little variation which means that the fault occurred sometime before this interval. So the estimation window is stretched backwards and this procedure is repeated over the stretched window till an ‘A’ primitive cannot be fitted *i.e.* there is enough variation in the data in the estimation window. This procedure always terminates since last primitive is a non-A primitive. Trends are recalculated for the data after the fault occurrence time. It can be seen that this method does not take into account the time-delay but that is not a problem because, to ensure robust estimation of similarity measure, the same methodology can be used during the development of the QTA-KB. As the fault evolves, more and more sensors deviate from their NOR. These sensors are used for the estimation of similarity measure and C.I. for various faults. This ensures that similarity measure for a slowly evolving sensor or a sensor that is not showing enough deviation would not result in incorrect C.I. for the actual fault. To summarize, the main activities involved in on-line FD are:

- *Fault detection*
- *Estimation of the fault occurrence time*
- *Computationally efficient trend matching*
- *Learning and updating the QTA-KB*

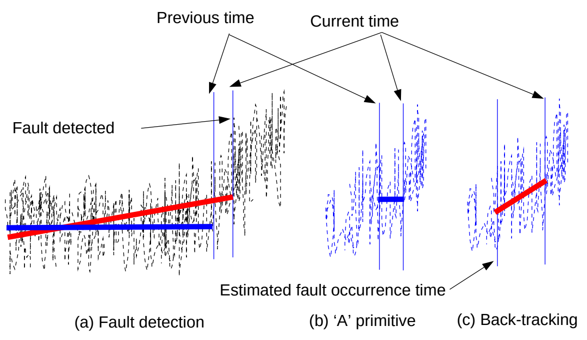

Fig. 6. Estimation of the fault occurrence time

7. A PROTOTYPE DIAGNOSTIC SYSTEM

A prototype diagnostic system to implement the framework discussed above has been designed in Matlab®. To give a brief idea about the important features of the framework, some of the components are succinctly discussed below:

Development of QTA-KB: Trends for various fault scenarios are extracted and sensors to be used for estimating the C.I. for various faults are identified. *On-line trend extraction:* The adaptive trend-extraction algorithm is used.
Fault detection: Presence of a fault is triggered if a sensor deviates from its NOR and shows a non-A primitive.
Fault occurrence time: Backtracking methodology is used. The relevant trends are re-calculated.
Fault diagnosis: C.I. for various faults are calculated. The operator is informed about the abnormal sensors and the known or an unknown event.
Learning and updating the QTA-KB: If a novel event occurs and the operator diagnoses the fault, it is added to the QTA-KB.

The diagnostic system has been tested on the TE process (sample time is 0.001 hr for all 14 sensors). No optimal sensor assignment has been carried out for this case study. Out of the 33 possible faults, 26 faults are used to build the QTA-KB. Results for three test scenarios are as follows:

(1) Fault 1: correct diagnosis.
(2) Fault 25: faults 17, 25 and 18 (located in the same control loop).
(3) Fault 28 (novel fault): fault 20 (located in the same control loop as fault 28).

Thus the prototype diagnostic system is quite robust with respect to correct FD and detection of novel events. Interaction with the operator, learning and maintenance is also easily facilitated. Another important characteristic of the overall approach is that most of the parameters used can be easily tuned and that reasonable variation in them does not degrade the overall performance. For example, in the TE process, characteristic time is few hours. The normal window size is 255 (1/4th of the number of samples per hour). maxCFlen and minNLP_len are 1000 and 50, respectively. The window shift length is 15 (to allow the necessary computations every minute).

8. CONCLUSIONS AND FUTURE SCOPE

Various issues in on-line trend extraction and fault diagnosis have been discussed. An algorithm for the same has been presented. A framework for on-line fault diagnosis has been presented. Important components have been discussed. A prototype for FD has been developed in Matlab®. Some results for FD of the TE process have been discussed. Our future work will include improvement in on-line trend extraction (to further reduce computation time) and multiple fault diagnosis.

REFERENCES

Bakshi, B. R. and G. Stephanopoulos (1994*a*). Representation of process trends – III. Multi-scale extraction of trends from process data. *Comput. & Chem. Engng.* **18**(4), 267–302.

Bakshi, B. R. and G. Stephanopoulos (1994*b*). Representation of process trends – IV. Induction of real-time patterns from operating data for diagnosis and supervisory control. *Comput. & Chem. Engng.* **18**(4), 303–332.

Cheung, J. T. and G. Stephanopoulos (1990). Representation of process trends – Part I. A formal representation framework. *Comput. & Chem. Engng.* **14**(4/5), 495–510.

Dash, S., R. Rengaswamy and V. Venkatasubramanian (2001). A novel interval halving algorithm for process trend identification. In: 4th *IFAC Workshop on On-Line Fault Detection & Supervision in the Chemical Process Industries, Korea.* pp. 155–160.

Dash, S., R. Rengaswamy and V. Venkatasubramanian (2003). Fuzzy-logic based trend classification for fault diagnosis of chemical processes. *Comp. & Chem. Eng.* **27**(3), 347–362.

Janusz, M. and V. Venkatasubramanian (1991). Automatic generation of qualitative description of process trends for fault detection and diagnosis. *Engng Applic. Artif. Intell.* **4**(5), 329–339.

Mah, R. S. H., A. C. Tamhane, S. H. Tung and A. N. Patel (1992). Process trending with piecewise linear smoothing. *Comput. & Chem. Engng.* **19**(2), 129–137.

Oh, Y. S., K. J. Moo, E. S. Yoon and J. H. Yoon (1997). Fault diagnosis based on weighted symptom tree and pattern matching. *Ind. Eng. Chem. Res.* **36**, 2672–2678.

Vedam, H. (1999). OP-AIDE: An intelligent operator decision support system for diagnosis and assessment of abnormal situations in process plants. PhD thesis. Purdue University.