PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 41, NO. 2, PP. 101–121 (1997)

EVENT RECOGNITION VIA LINEAR STATE AND PARAMETER MODEL

Mohamed M. ELMISURATI

Department of Process Control
Technical University of Budapest
H–1521 Budapest, Hungary
E-mail: Elmisu@bagira.fsz.bme.hu

Received: Nov. 26, 1996

Abstract

A new method based on linearization of the system equations has been proposed to recognize malfunctions, which can be associated with system parameters in dynamical systems. The main features of this method are the short computation time, and the ability to filter out those events caused by disturbances. In addition, the employed linear state and parameter model makes possible to compute the solution with a recursive algorithm, which is fast enough for on-line application, consequently, one can avoid false malfunction recognition due to operational dynamic transients.

To illustrate the method, recognition of malfunctions in the operation of a gravitational water tower is presented.

Notation

$\mathbf{A}_p$ = cross-sectional area of the transport pipe
$\mathbf{A}_T$ = cross-sectional area of the vertical cylindrical tank
F_0 = volumetric inflow rate
F = volumetric outflow rate
g = the acceleration due to gravity
h = height of water in the vertical cylindrical tank
K_f = flow resistance coefficient
K_p = proportional gain of the controller
L = length of the transport pipe
p_a = ambient pressure
p_c = consumption pressure
t = independent time variable
v = velocity of mass of water in the transport pipe
ρ = water density
α = leakage coefficient
λ = relaxation coefficient
$d()/dt$= derivative

The process state variables at the steady state operating condition and the process coefficients had the following values for the simulations:

State variables and their values at normal operating conditions:
v_s= velocity of mass of water in the transport pipe, 4.975 m/s
h_s= height of water in the vertical cylindrical tank, 1.6 m

Process coefficients:
L = length of the transport pipe, 900 m
g = the acceleration due to gravity, 9.81 m/s^2
K_f = flow resistance coefficient, 1.61 kg/m^2
ρ = water density, 1000 kg/m^3
$\mathbf{A}_p$ = cross-sectional area of the transport pipe, 0.785 m^2
$\mathbf{A}_T$= cross-sectional area of the vertical cylindrical tank, 12.56 m^2
α = leakage coefficient 0 (no leakage)
F_{os}= volumetric inflow rate, 3.905 m^3/s
K_p = proportional gain of the controller, 0.6 m^2/s
p_a = ambient pressure, 10^5 N/m^2
p_c = consumption pressure, 7×10^4 N/m^2

Abbreviations

$LPME1$	Linear Plant Model Event 1
$NNE1$	Neural Network Event 1
$LSAPME1$	Linear State And Parameter Model Event 1
$E1L4$	Event 1/ Level 4
$E2L1$	Event 2/ Level 1

Introduction

Physical systems are often subjected to unexpected changes due to variations in operating conditions or component failures, that tend to degrade overall system performance. It is important that changes be promptly detected and identified so that appropriate remedies can be applied. Over the past decades numerous approaches to the problem of failure detection and identification (FDI) in dynamical systems have been developed [1]; detection filters [2], [3]; the generalized likelihood ratio (GLR) method [4]; and the multiple model method [5], [6] are some examples. All these analytical methods require that a dynamical process model of some sorts be given. The main objective of FDI is to detect faults, disturbing patterns or other changes in order to prevent catastrophic failures in the system. This is extremely necessary in advanced technologies, where failures can endanger human beings or the natural environment. In case of fault situations failure detection systems may either command control systems to switch off the abnormal process, accommodate the fault in any way, e.g. using fault tolerant control. For the sake of clarity and further understanding, expressions and definitions commonly used in the theory of FDI are defined next.

Change: A change is understood as any variation of process variables from a constant value. Change is a natural property of dynamical systems (e.g. dynamical change of inputs, outputs, state variables or certain process coefficients).

Fault: A fault is understood as any kind of unallowed deviation of at least one process variable from its nominal value. It is an abnormal change that leads to an unacceptable anomaly in the overall system performance. Such faults may occur in sensors, actuators, or in components of a process.

Malfunction: A malfunction is the abnormal operation of at least one part of a system caused by one or more faults. It is a state and can be handled through fault accommodation so that it will be temporary.

Failure: A failure is understood as the disability of at least one functional unit of the system to be operated. It is an event that can be handled only through switching off the failed units. *Fig. 1* illustrates the above definitions.

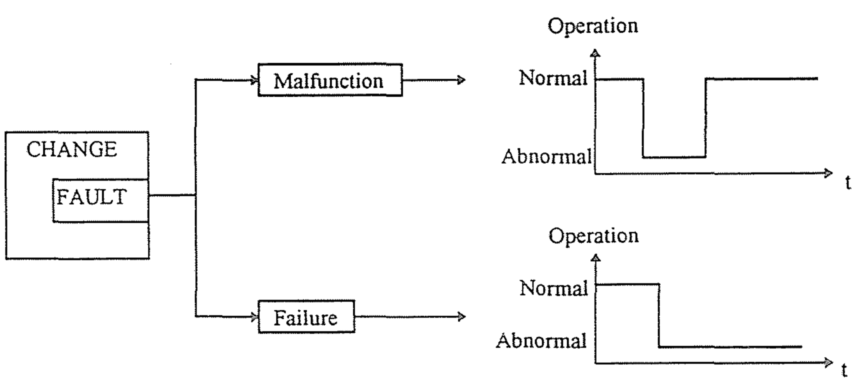

Fig. 1. Definitions of FDI

It is clear from the above illustration that though a detected change does not necessarily correspond to a fault, the faulty operation of the system is always preceded by changes in the dynamics. Therefore, the major task is the early detection of all significant changes in the dynamics of the plant behavior: *change detection*.

Once a change is detected, one has to decide if it is significant from fault detection point of view or not. In other words, changes have to be validated: *change validation.* In order to achieve efficient diagnosis, the isolation of the origin, functional or spatial locations of changes have to be determined. This problem is referred to as *change isolation.* It is noteworthy that the last two stages of the detection process are sometimes cited as *change identification* [7] as illustrated in *Fig. 2.* In practical situations, the levels of normal and abnormal operations are decided on the basis of the actual process technology.

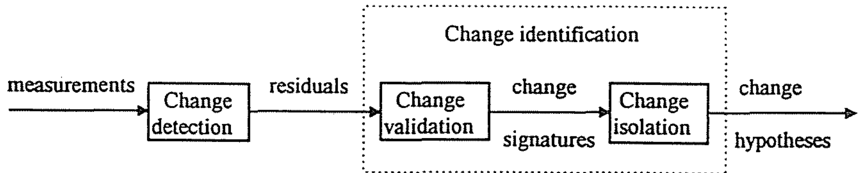

Fig. 2. Computational stages of the change detection process

Linear Plant Model

This method is based on identification of the trajectories of system variables [8]. A nonlinear process is approximated by a linear multivariable model where the control variables $\mathbf{u}(t)$ represent the so-called event indicators:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \tag{1}$$

The event indicators can be computed as the inverse solution of the linear model. Using a time discretized form, the solution (when the number of events to be recognized is equal to the number of measured state variables) is given by

$$\mathbf{u}^{n+1} = \frac{1}{\Delta T}\mathbf{B}^{-1}\mathbf{x}^{n+1} - \frac{2}{\Delta T}\mathbf{B}^{-1}\Phi\mathbf{x}^{n} - \mathbf{B}^{-1}\Phi\mathbf{B}\mathbf{u}^{n}, \tag{2}$$

where $\Phi = \exp(\mathbf{A}\Delta T)$, and $\mathbf{x}^n = \mathbf{x}(n\Delta T)$ *Eq.* (2) can be utilized for detecting events (faults) resulting in measurable malfunction trajectories $\mathbf{x}(t)$. The main feature of this method is that the event to be detected is directly related to the trajectories of the state variables. In this way its robustness is considerably increased. In addition, the employed linear plant model makes it possible to compute the inverse solution with a recursive algorithm, which is fast enough for on-line application. The method works well even in the presence of system and measurement noises, as long as the number of measured state variables is greater than or equal to the number of events to be recognized. One has to keep in mind that for a highly-nonlinear process the validity of the linear approximation is limited, although the global linearization employed for identifying this model is somewhat more robust than a linearization around an equilibrium point, and in the case of a new operational point, generally a new identification is needed or else one has to be satisfied with only qualitative results.

In the presence of disturbances, however, this method gives false event indication.

Neural Network

Artificial neural networks can identify and learn correlative patterns between sets of input data and corresponding target values. Once trained, such networks can be used predictively to forecast outcomes from new input data. The roots of these ideas lie in the simplified explanations of the functioning of human and animal brains.

Artificial Neuron

An artificial neuron is a simple processing element that serves as a transfer function mapping a multidimensional input received from other artificial neurons or external stimuli to a one-dimensional output which is distributed to other artificial neurons through weighted connections. The transfer function of the j-th artificial neuron in the 1-th layer in *Fig. 3* is specified by a sigmoidal function

$$x_j^{(l)} = \frac{1}{1 - e^{-u_j(l)}}, \tag{3}$$

with $u_j^{(1)}$ being usually (but not always) a linear sum of the weighted connection strengths being fed to the node plus a threshold:

$$u_j^{(l)} = \left\{ \sum_{i=1}^{N-1} w_{ji}^{(l)} x_i^{(l-1)} + \theta_j^{(l)} \right\}, \tag{4}$$

where variable $x_j^{(l)}$ is the output or the activity level of j-th node in the l-th layer and variable $x_i^{(l-1)}$ is the output of i-th node in the $(l-1)$th layer, when the input pattern p is fed to the network; $\theta_j^{(l)}$ is the threshold of the j-th node in l-th layer.

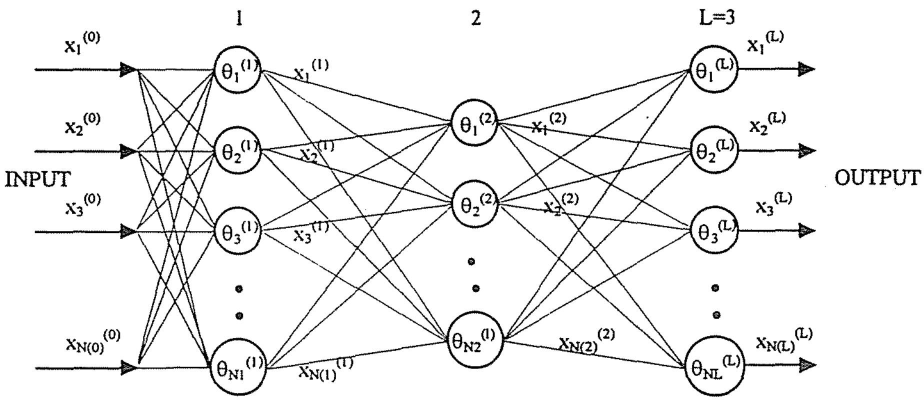

Fig. 3. Schema of a multilayer neural network

Network Topology

Fig. 3 shows a standard multilayer feedforward artificial neural network with one or more so-called hidden layers ('hidden' because such layers do not communicate directly with the external environment.) The arcs that connect the artificial neurons are unidirectional feedforward connections. Let the number of neurons in layer j be N_l. The arc from i-th node in the $(l-1)$th layer to j-th node in j-th layer has an associative weight $w_{ji}^{(l)}$ which multiplies the signal from i-th node in $(l-1)$th layer. Knowledge in artificial neural networks is distributed among the connections and the weights $w_{ji}^{(l)}$ and not stored at a single computer address. For fault detection each node in the output layer would represent a particular fault.

Processing in a Multilayer Artificial Neural Network

Each node in the input layer receives input from an external stimulus that is either scaled prior to introduction into the respective input node or scaled by the node. The output of each node in the input layer is passed on to all the nodes in the next layer.

Each artificial neuron in the next layer computes an output (activity level) that is a function of its inputs. The computations within a layer are asynchronous and thus may be performed in parallel. The output of one node is distributed to all the other artificial neurons in the subsequent layer through the weighted connections. This arrangement is repeated in a feedforward manner until the output layer is reached. Thus, computations between layers are synchronous.

Learning (Training)

Learning is nothing more than adjusting the weights associated with connections between the nodes of the network. An input vector of process measurements associated with a fault pattern is introduced into the input layer of the network. A corresponding output pattern composed of output layer node activities is calculated. An error is generated for each output node based on the difference from a target value for the node or a goal. For fault detection, an output node target would be either 0 (no fault) or 1 (a particular fault). The neural network learns a target output pattern by adjustment of the weights in the network; therefore, after a sequence of presentations of input vectors, the network generates the desired output pattern for its associated input measurement vector. To adjust the weights, the backpropagation procedure [9] is used in which the objective function:

$$E_p = \frac{1}{2}\sum_{j=1}^{N_L}\{t_{pj}^{(L)} - x_{pj}^{(L)}\}^2 \tag{5}$$

is minimized for a given input pattern p, where $t_{pj}^{(L)}$ is the target output (activity) of the j-th node in the output layer for pattern p. The same input vector may be used periodically during the learning process.

Learning via backpropagation involves two phases. In the first phase, the inputs are propagated in a feedforward manner through the network to produce output values that are compared to the target values, resulting in the error signal for each of the output nodes. In the second phase, the errors are propagated backward through the network and used to adjust the weights. The error signals for the output layer are calculated first, and these error signals are used recursively to calculate the needed adjustments layer by layer until the weights for all of the connections are recalculated.

Pattern Recognition (Fault Detection)

Once the weights on the connections are finally adjusted in the training phase, news of sensor measurements can be sent to the input nodes of the network and classified. Very little computation time is needed for this step. The degree of misclassification that occurs is a function of how well the knowledge stored in the connections and weights can represent perturbations from its training set of data.

The neural network method is relatively fast and requires less computational power as compared with state estimation and identification procedures. Another advantage of this method is that it can be utilized in situations where the number of events to be recognized exceeds the number of measured state variables. This method also works well in the presence of system and measurement noises as well as different deterioration levels for each fault [10]. However, in the presence of disturbances this method also gives misleading or false event indication.

Linear State and Parameter Model

To avoid the misleading results and false event indication, in the presence of disturbances, we suggest the following new model:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{p}(t) + \mathbf{C}\mathbf{d}(t), \tag{6}$$

where $\mathbf{x}(t)$ is an N-dimensional vector,
$\mathbf{p}(t)$ is an M-dimensional parameter vector,
$\mathbf{d}(t)$ is an L-dimensional disturbance vector.

Based on the fact that changes of faults of a dynamical system may be reflected in the physical parameters, the choice of components of the parameter vector $\mathbf{p}$ are directly related to the M events to be recognized. The components of $\mathbf{d}$ correspond to possible disturbances.

Let u_i represent the i-th event, then

$$u_i = \sum_{j=1}^{M} \alpha_{ij} p_j + \alpha_{io} \tag{7}$$

so that abnormalities in the system can be detected if a certain parameter does not remain within the bounds of normal operation as shown in *Fig. 4*.

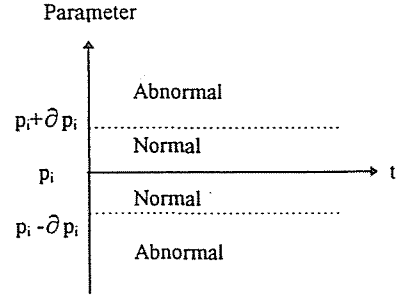

Fig. 4. Admissible bounds of the i-th parameter

Building Up the Model

(a) No model case:

When there are only measurements, it would be very difficult to identify the $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ matrices.

(b) Model-based approach:

When a mathematical model of the nonlinear process is available, then a linearized version can be obtained by expanding the model equations into a Taylor series around some normal operating points $(\overline{\mathbf{x}}, \overline{\mathbf{p}}, \overline{\mathbf{d}})$ as follows:

Let the nonlinear process be represented by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{p}, \mathbf{d}) \ , \tag{8}$$

then the i-th equation of the linearized process is given by

$$\dot{\mathbf{x}}_i = \mathbf{f}_i(\overline{\mathbf{x}}, \overline{\mathbf{p}}, \overline{\mathbf{d}}) + \sum_{j=1}^{N} \frac{\partial f_i}{\partial x_j}(x_j - \overline{x}_j) + \sum_{k=1}^{M} \frac{\partial f_i}{\partial p_k}(p_k - \overline{p}_k) +$$

$$+ \sum_{q=1}^{L} \frac{\partial f_i}{\partial d_q}(d_q - \overline{d}_q) \ , \tag{9}$$

where the high-order terms were neglected, since we are considering only a small perturbation near the normal operating point (equilibrium point).

Algorithm for Parameter Identification

The solution of *Eq.* (6) for discrete time steps ΔT can be written in the form:

$$\mathbf{x}^{n+1} = \Phi \mathbf{x}^n + \frac{\Delta T}{2}[\Phi(\mathbf{B}\mathbf{p}^n + \mathbf{C}\mathbf{d}^n) + (\mathbf{B}\mathbf{p}^{n+1} + \mathbf{C}\mathbf{d}^{n+1})] \ , \tag{10}$$

where use of the trapezoidal rule for integration over the interval ΔT has been made, and

$$\Phi = \exp(\mathbf{A}\Delta T) \ , \quad \mathbf{x}^n = \mathbf{x}(n\Delta T) \ .$$

When the $\mathbf{B}$ matrix has an inverse, and $N = M$ (that is, the number of measured state variables is equal to the number of events to be recognized), expressing $\mathbf{p}^{n+1}$ from *Eq.* (1) we get:

$$\mathbf{p}^{n+1} = \frac{2}{\Delta T}\mathbf{B}^{-1}\mathbf{x}^{n+1} - \frac{2}{\Delta T}\mathbf{B}^{-1}\Phi\mathbf{x}^{n} - \mathbf{B}^{-1}\Phi\mathbf{B}\mathbf{p}^{n} - \mathbf{B}^{-1}\Phi\mathbf{C}\mathbf{d}^{n} - \mathbf{B}^{-1}\mathbf{C}\mathbf{d}^{n+1} . \tag{11}$$

Eq. (11) is a recursive formula for determining $\mathbf{p}$, and consequently the event indicators. To enhance the accuracy of the resulting $\mathbf{p}$ further a moving average (MA) may be used:

$$MA(p_k) = \frac{\sum_{k=0}^{i} p_k}{i} . \tag{12}$$

Illustration

To illustrate the use of the above suggested model, the well known linear plant model, and the neural network approach, we considered an application example of a gravitational water tower process.

Process Model

The schematic diagram of the process is shown in *Fig. 5*.

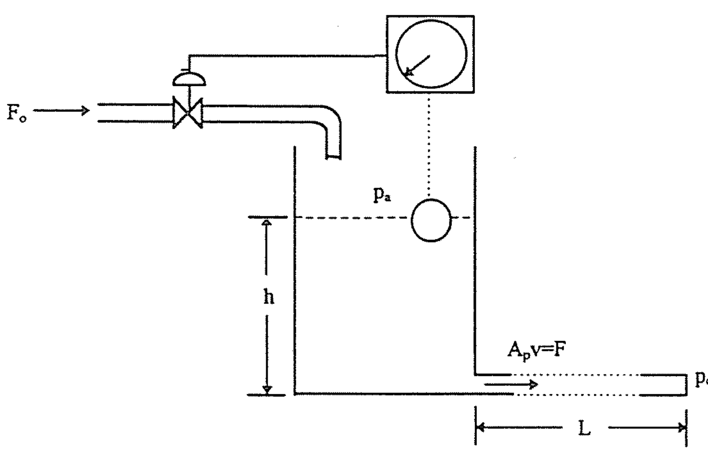

Fig. 5. Water tower system to be diagnosed

It is a proportional-feedback-controlled process of a gravitational water tower, into which water is pumped at a variable volumetric rate F_0. The controlled variable is the water level of the tower h, and the controller regulates this level to the specified value regardless of the change in the consumption pressure p_c.

The part of this process that is described by a force balance or more eloquently the conservation of momentum is the water flowing through the transport pipe [11]. The mathematical model of the water tower, and controller that were used to simulate various events are:

Momentum equation:

$$LA_p\rho\frac{dv(t)}{dt} = (p_a + h(t)\rho g - \frac{K_f L}{A_p}v^2(t) - p_c)A_p \ , \tag{13}$$

which can be written as

$$\frac{dv(t)}{dt} = \frac{p_a - p_c}{L\rho} + \frac{g}{L}h(t) = -\frac{K_f}{\rho A_p}v^2(t) \ . \tag{14}$$

Continuity equation:

$$A_T\frac{dh(t)}{dt} = F_o - A_p v(t)(1+\alpha) \ . \tag{15}$$

Proportional controller:

$$F_o = F_{os} - K_p(h(t) - h_s) \ . \tag{16}$$

Combining *Eqs.* (15), and (16) we obtain:

$$\frac{dh(t)}{dt} = \frac{[F_{os} - K_p(h(t) - h_s) - A_p v(t)(1+\alpha)]}{A_T} \ . \tag{17}$$

Initial conditions are:

$$v(0) = v_o = v_s \quad \text{and} \quad h(0) = h_o = h_s \ . \tag{18}$$

Event Recognition

To compare the three different methods in diagnosing events, we categorized two possible causes of faults as follows:

Event 1

Leakage in the transport pipe, due to hole corrosion (slow), cracking of welding seam (abrupt), or pipe burst, leads to an increase in the leakage factor α.

Event 2

Partial plugging of the pipe line leads to an increase in the frictional force opposing the flow (due to water viscosity), and hence an increase in the friction coefficient K_f (frictional force $= K_f L v^2$).

We discriminated among and / or diagnosed the existence of the above two events from measurements of velocity v of mass of water in the transport pipe, and the water level of the cylindrical tower h. The fault data were generated by increasing the parameters α, and K_f. We picked examples of the two different events cited above and four different levels of deterioration for each of the two events. The number of the events designates the label assigned to each of the two different events, and the level of the events corresponds to the degree of deterioration. Events in level 1 are slight and thus incipient, events in level 2 and 3 are medium, and an event in level 4 is the most severe (see *Table 1*).

The neural network employed in this comparative task comprises three layers. It has two input nodes corresponding to the two measured state variables, a middle layer with five nodes, and two output nodes corresponding to the two events.

The dynamical trajectories shown in *Fig. 6* were used (after normalization) for teaching the network. The training pattern consists of the deviations from steady state of the two process state variables resulting from simulation of the tower over a period of 800 seconds, so that 400 samples corresponding to each of the two variables (for every event at level 1) are used as the input teaching data, together with the normal case (no faults).

The network was trained via the backpropagation procedure with the input data normalized between 0.1 and 0.9. The number of iterations of a pattern set to learn the knowledge of the level 1 events was 10,000.

Analysis

In the following Figures, indices i and j refer to various sampling instants of time. *Fig. 6* displays the deviations of the state variables with respect to the two events at the lowest level (level 1).

Now the recognition of the events at level 1 is performed via the three different methods { the linear plant model (LPM), the neural network

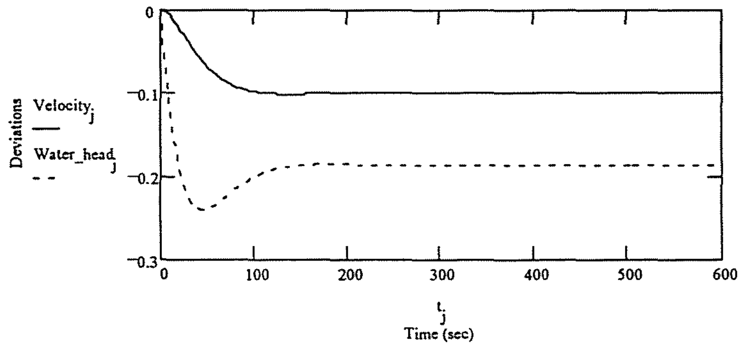

Fig. 6a. Deviations with respect to event 1 / level 1

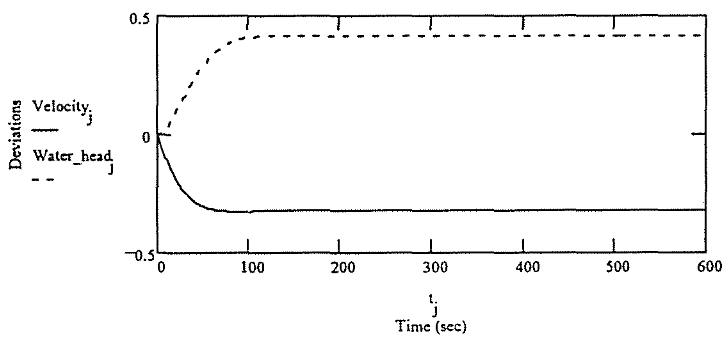

Fig. 6b. Deviations with respect to event 2 / level 1

Table 1
Causes of events

Event No./Level	Cause
1 / 1	$\alpha = 0.05$
2	$\alpha = 0.1$
3	$\alpha = 0.15$
4	$\alpha = 0.2$
2 / 1	$K_f = 2$
2	$K_f = 2.3$
3	$K_f = 2.6$
4	$K_f = 3$

(NN), and the suggested linear state and parameter model (LSAPM) } as shown in *Fig. 7*. It is evident that all methods gave satisfactory results even though the suggested LSAPM is a little bit better. We also see that in *Fig. 7a* the indicator vector corresponding to event 2 (in the case of the neural network, and the suggested models coincide with each other) remains approximately zero, which means that event 2 did not occur.

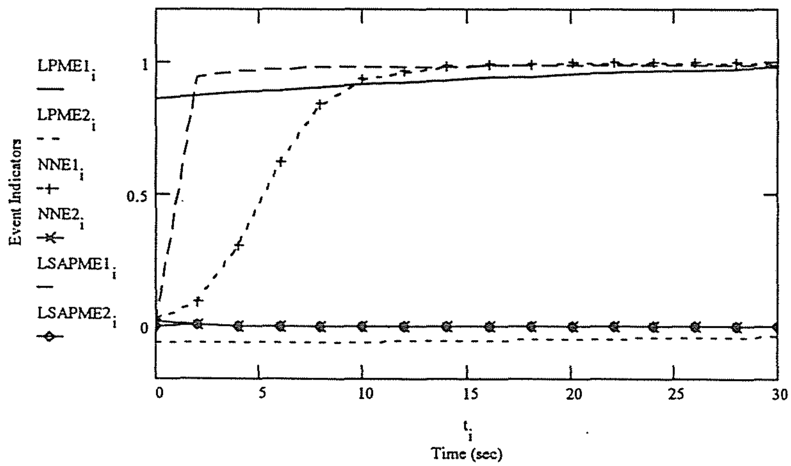

Fig. 7a. Recognition of event 1 / level 1

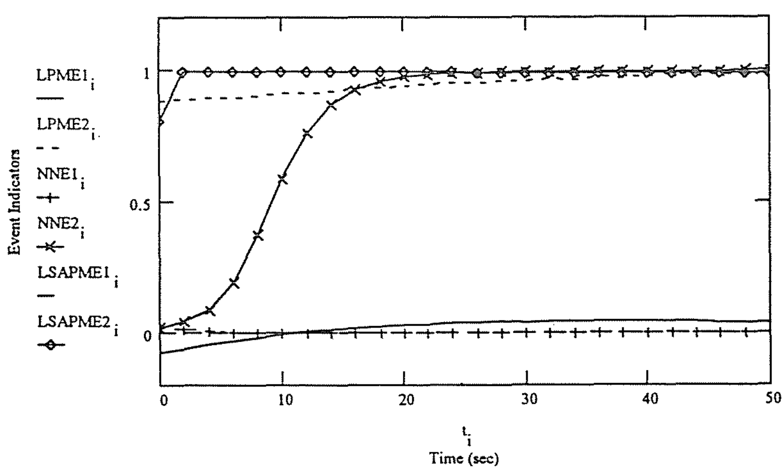

Fig. 7b. Recognition of event 2 / level 1

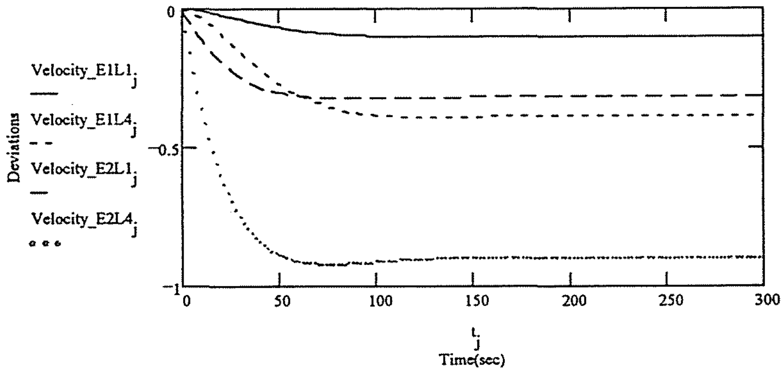

Fig. 8. Deviations of velocity

The deviations of velocity with respect to both events at lowest and highest levels of deterioration are shown in *Fig. 8.* (Here, EIL1 means event 1 / level 1, and E2L4 means event 2 / level 4).
Next, the recognition of events at level 4 is executed, and the results are displayed in *Fig. 9.* We see that both the neural network and the suggested model gave satisfactory quantitative event indication, while the linear plant model gave only a qualitative result. This is due to the limited validity of the linear approximation, and in the case of a new operational point (level 4), a new identification of the linear plant model parameters is necessary.

Next, we added 10% internal noise in the consumption pressure p_c together with 10% measurement noise in order to compare the versatility of the three methods. The deviations of the state variables with respect to events at level 1 in the presence of such noise are shown in *Fig. 10.*

To enhance the accuracy of the event recognition in the presence of noise, utilization of the moving average *Eq.* (2) has been made, and the effect of its use on recognizing event 1 / level 1 in the case of the linear plant model is illustrated in *Fig. 11b.* In contrast, *Fig. 11a* displays the same event recognition but without using the moving average.

Besides the moving average, we also made use of the relaxation method in the case of the suggested model.

The relaxation method was used in the following way:

$$\tilde{\mathbf{p}}^{n+1} = \lambda \mathbf{p}^{n+1} + (1-\lambda)\mathbf{p}^{n} , \tag{19}$$

with the relaxation coefficient $\lambda = 0.5$.
The combined effect of their use on recognizing event 1 / level 1 is illustrated in *Fig. 11d*, while *Fig. 11c* shows the same event recognition only with the moving average being utilized.

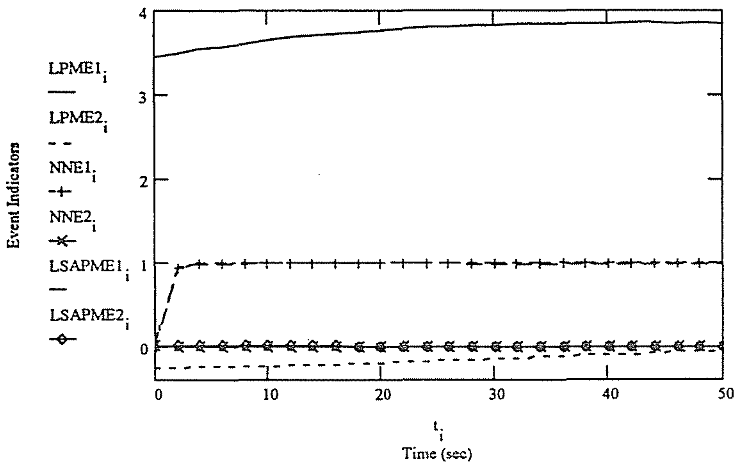

Fig. 9a. Recognition of event 1 / level 4

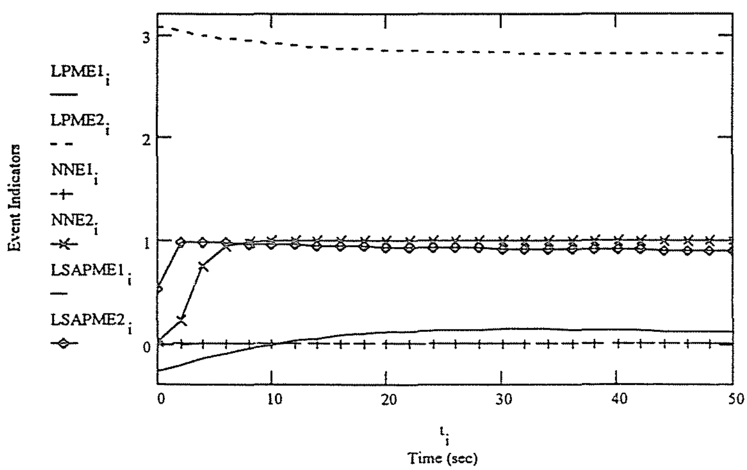

Fig. 9b. Recognition of event 2 / level 4

The recognition of events at level 1 in the presence of the above mentioned noise is shown in *Fig. 12.* It is clear from the figure that all three methods provided reasonable results.

Next, under normal operating conditions we applied an abrupt (step) disturbance in the consumption pressure p_c of 30% magnitude. The corresponding deviations of the state variables are displayed in *Fig. 13.*

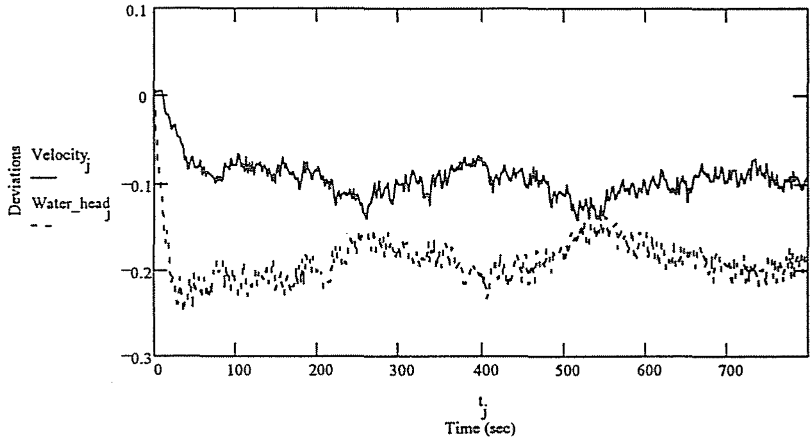

Fig. 10. a Deviations with respect to event 1 / level 1

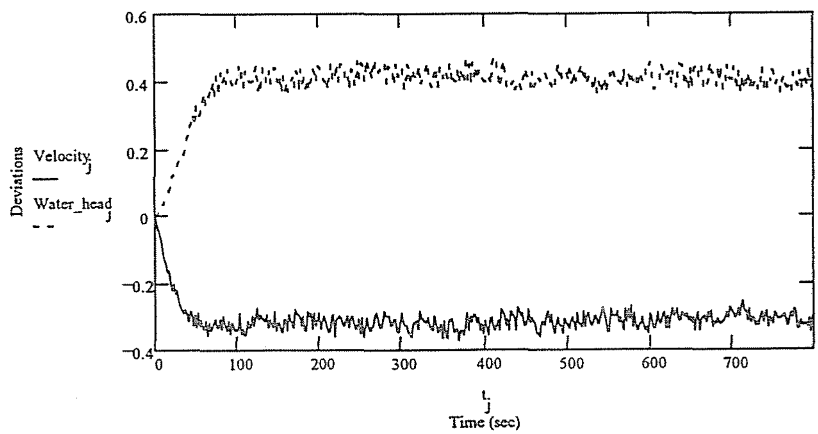

Fig. 10. b Deviations with respect to event 2 / level 1

The response of the three methods to such a disturbance is shown in *Fig. 14*. It is crystal clear from this figure that both the linear plant model, and the neural network gave false (misleading) event indication. In contrast, only the suggested model was able to produce the correct (no malfunction) result.

Finally, to test the suggested model further, we examined the situation when the two events at level 1 are delayed in time. The results of the recognition in this case are shown in *Fig. 15*. It is obvious that both the

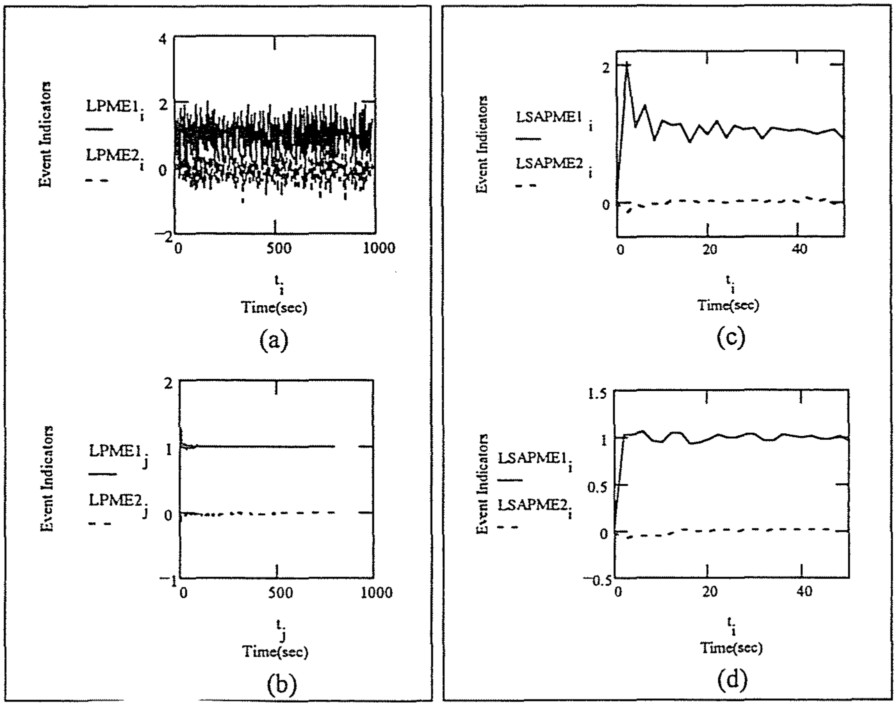

Fig. 11. Recognition of event 1 / level 1

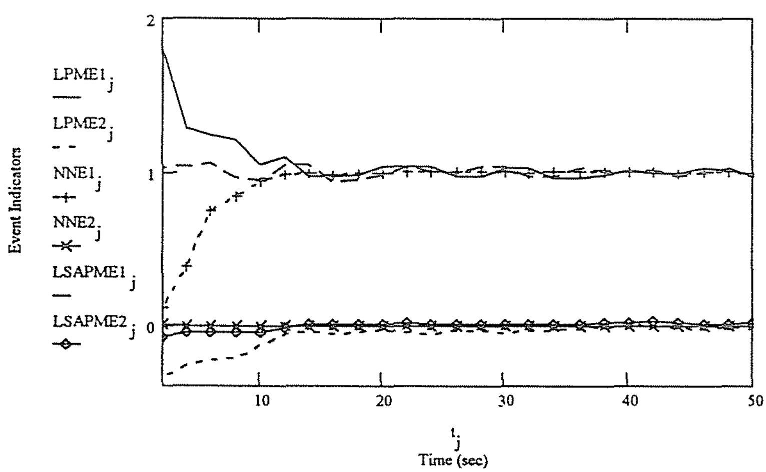

Fig. 12. a Recognition of event 1 / level 1

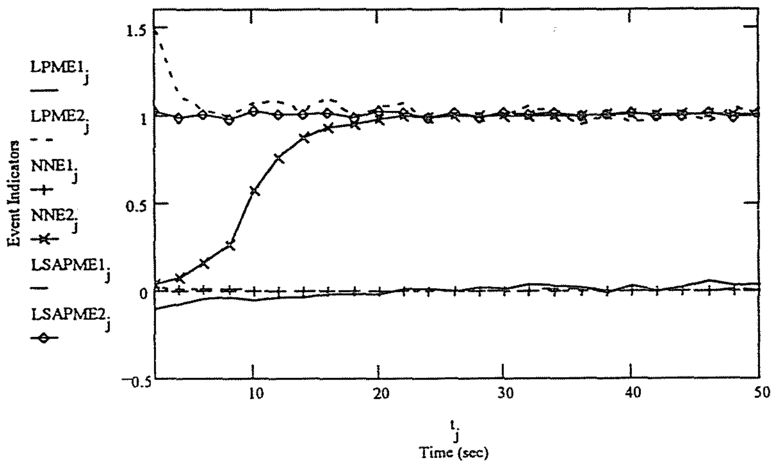

Fig. 12. b Recognition of event 2 / level 1

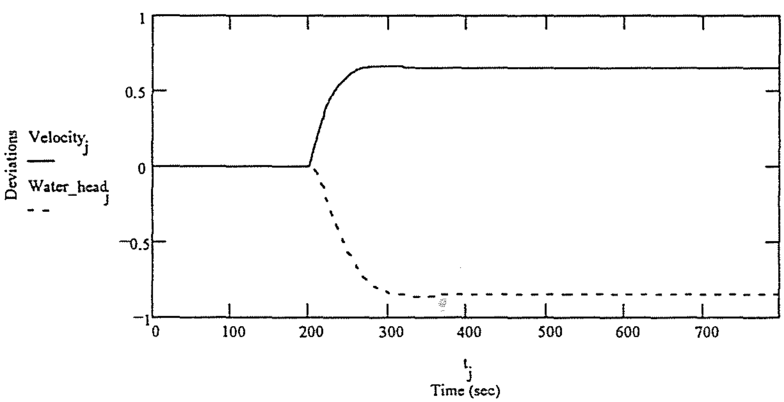

Fig. 13. Deviations with respect to jump disturbance in p_c

linear plant model and the suggested model gave satisfactory results, while the neural network indicated the events correctly one at a time, but not jointly together. This is due to the fact that such a situation of delayed events was not included in the teaching pattern for the neural network.

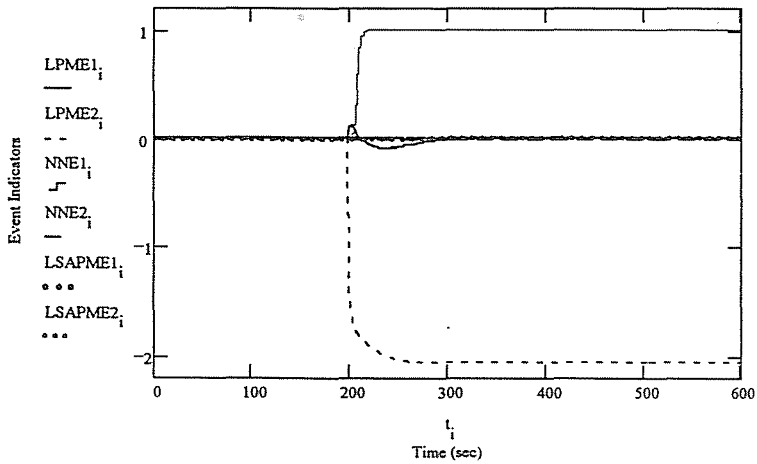

Fig. 14. Response to jump disturbance

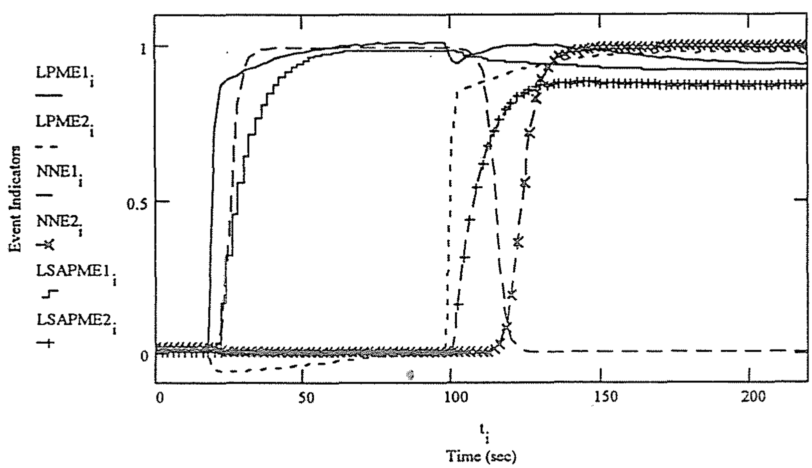

Fig. 15. Recognition of two events delayed in time

Conclusions

In this paper, a modified linear state and parameter model was suggested for recognition in dynamical systems. Based on the results of our investigation, the method can be used on-line as well as to distinguish malfunctions from operational dynamic transients.

The method is robust with respect to noise and level change of events, also it is applicable to simultaneous events delayed in time.

Further improvement of this method can be achieved by considering matrices with time-varying entries (elements), instead of time-invariant ones, and hence the resulting model will be suitable for even a broader class of highly nonlinear systems.

Acknowledgement

The author wishes to thank his supervisors Dr. Z. Benyó and Dr. B. Paláncz for their valuable guidance and encouragement in the realization of this article.

References

1. WILLSKY, A. S. (1976): A Survey of Design Methods for Failure Detection in Dynamic Systems, *Automatica*, Vol. 12, pp. 601–611.
2. ASACHENKOV, A. – MARCHUK, G. – MOHLER, R. – ZUEV, S. (1994): Disease Dynamics, Birkhäuser, Boston, first-edition.
3. BASSEVILLE, M. (1988): Detecting Changes in Signals and Systems – a Survey, *Automatica*, Vol. 24, No. 3, pp. 309–326.
4. BASTIN, G. – DOCHAIN, D. (1986): On-line Estimation of Microbial Specific Growth Rates, *Automatica*, Vol. 22, No. 6, pp. 705–709.
5. BASTIN, G. – GEVERS, M. (1988): Stable Adaptive Observers for Nonlinear Time Varying Systems, *IEEE Transactions on Automatic Control*, Vol. 33, pp. 650–658.
6. BEARD, R. V. (1971): Failure Accommodation in Linear Systems through Self-reorganization, PhD Thesis, MIT, Cambridge, MA, USA.
7. EDELMAYER, A. (1993): Robust Detection Filter Design in Uncertain Dynamical Systems, PhD Thesis, Systems and Control Lab., Hungarian Acad. Sci., Budapest, Hungary.
8. BENYÓ, Z. – BENYÓ, I. – BENEDEK, S. – PALÁNCZ, B. (1990): Use of an Event Recognition Method for Determining Illnesses of Physiological Systems, *Proceedings of the 10th Annual Conference of the IEEE/EMBS*, pp. 1198–1199, Philadelphia, USA.
9. RUMELHART, D.E. – MCCLELLAND, J. L. (1987): Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, Massachusetts.
10. ELMISURATI, M. (1996): Dynamical Approach for Event Recognition in Chemical Processes via Neural Networks, *Proceedings of the 10th European Simulation Multiconference*, pp. 641–645, Budapest, Hungary.
11. LUYBEN, W.L.: Process Modeling, Simulation, and Control for Chemical Engineers, New York, McGraw-Hill, pp. 105–106.