IEICE TRANS. FUNDAMENTALS, VOL.E90–A, NO.11 NOVEMBER 2007

PAPER Special Section on Concurrent/Hybrid Systems: Theory and Applications

Fault Detection and Diagnosis of Manipulator Based on Probabilistic Production Rule

Shinkichi INAGAKI[†a)], **Koudai HAYASHI**[†*], *Nonmembers*, *and* **Tatsuya SUZUKI**[†], *Member*

SUMMARY This paper presents a new strategy to detect and diagnose fault of a manipulator based on the expression with a Probabilistic Production Rule (PPR). Production Rule (PR) is widely used in the field of computer science as a tool of formal verification. In this work, first of all, PR is used to represent the mapping between highly quantized input and output signals of the dynamical system. By using PR expression, the fault detection and diagnosis algorithm can be implemented with less computational effort. In addition, we introduce a new system description with Probabilistic PR (PPR) wherein the occurrence probability of PRs is assigned to them to improve the robustness with small computational burden. The probability is derived from the statistic characteristics of the observed input and output signals. Then, the fault detection and diagnosis algorithm is developed based on calculating the log-likelihood of the measured data for the designed PPR. Finally, some experiments on a controlled manipulator are demonstrated to confirm the usefulness of the proposed method.

key words: *probabilistic production rule, fault detection, fault diagnosis, manipulator*

1. Introduction

The demand for design of reliable control system is rapidly growing. The significant function to realize the reliable control system is to develop a fault detection and diagnosis (FDD) procedure, and this problem has been addressed by many researchers [1]–[5].

In the model based FDD for continuous systems, the observer based approaches [1], [3], [4] are well-known and effective in the case that a precise information of the system structure and the interaction with the environment are available. However, these model based approaches often result in high cost systems because of their high computational burden. Furthermore, the application of these approaches are generally limited to a linear system and its extensions. On the other hand, an information processing based approaches have been also developed. The typical ideas are expert systems and neural networks [5]. However, the expert systems depend strongly on the expert knowledge, and the neural networks have lots of difficulty in the design of the network structure due to the lack of explicit relationship between the performance of the FDD and the structure.

One of the interesting approaches of the FDD for the continuous systems is to quantize the system and develop the FDD based on the highly quantized system description. Since the computational effort depends on the quantization level, i.e., the number of alphabets — wherein each alphabet corresponds to each quantized symbol, the quantization based idea leads to the flexible design of the FDD balancing the performance and the computational burden. Once the system is quantized, several ideas of the FDD developed for the discrete event systems become available [6]–[9]. Although these approaches are very useful when the event occurrence and the system model are clearly defined in advance, the direct application to the complex continuous systems has not been addressed at all.

From this viewpoint, J.F. Martins et al. have developed an interesting FDD strategy applicable to the continuous systems based on a production rule (PR) based system description [10]. The advantages of PR based approach are (1) this approach requires only the input-output data strings, and no explicit model of the system structure is required at all (i.e., applicable to nonlinear complex dynamical systems), (2) the performance and computational burden of the FDD can be controlled by appropriately designing the quantization level.

One of the significant drawback of PR based approach, however, is that the number of PRs tends to be unreasonably large in proportion to the increase of complexity of the system and/or the given task. In order to overcome this problem, we introduce the probabilistic production rule (PPR), and exploit it for the design of the FDD of the continuous dynamical systems. By exploiting the PPR, the number of PRs can be reduced, and the flexible fault detection and diagnosis can be realized by evaluating the likelihood value of the observed input and output data calculated from the PPR.

This paper is organized as follows: In Sect. 2, the PR is briefly reviewed and the PPR is derived by assigning probabilities to each production rule. Then a vector quantization is introduced in order to quantize the input and output data, and alphabets are created based on the quantized results in Sect. 3. In Sect. 4, the FDD process based on the PPR description is proposed. In Sect. 5, the usefulness of the proposed method is verified by some experiments using a manipulator which has strong nonlinearity in its dynamics.

Manuscript received March 30, 2007.
Manuscript revised June 16, 2007.
Final manuscript received July 17, 2007.

[†]The authors are with the Dept. of Mechanical Science and Engineering, Nagoya University, Nagoya-shi, 464-8603 Japan.
[*]Presently, with NEC Corporation.
a) E-mail: inagaki@nuem.nagoya-u.ac.jp

DOI: 10.1093/ietfec/e90–a.11.2488

2. Description of the Dynamical System by Probabilistic Production Rule

2.1 Description of the Dynamical System by Production Rule (PR)

The system is supposed to be described by the following input-output dynamics:

$$\begin{aligned}\tilde{y}_{k+1} &= g_0(\tilde{U}_k)\\ \tilde{y}_{k+1} &= g_1(\tilde{y}_k, \tilde{U}_k)\\ \tilde{y}_{k+1} &= g_2(\tilde{y}_k, \tilde{y}_{k-1}, \tilde{U}_k)\\ &\vdots\\ \tilde{y}_{k+1} &= g_n(\tilde{y}_k, \tilde{y}_{k-1}, \cdots, \tilde{y}_{k-(n-1)}, \tilde{U}_k). \end{aligned} \tag{1}$$

where $\tilde{y}_k$ is the output value at time index k, and $\tilde{U}_k$ is the input value at time index k. This model implies that the one step ahead output $\tilde{y}_{k+1}$ is generated by the past outputs, and current input and output. This system can be represented by the following production rule (PR) [10] together with an appropriate quantization of input and output (see Sect. 3).

$$y_{k-(n-1)} \cdots y_k U_k \rightarrow y_{k-(n-1)} \cdots y_k y_{k+1} \delta \tag{2}$$

where y_k and U_k are the output and input alphabets quantized from $\tilde{y}_k$ and $\tilde{U}_k$ respectively. Also, δ denotes a special symbol given by

$$\begin{cases} \delta \rightarrow U_{k+1} \quad \text{or} & \text{(3a)}\\ \delta \rightarrow \lambda & \text{(3b)} \end{cases}$$

where λ denotes an 'empty' symbol. Equation (2) represents that the y_{k+1} is generated by the past n output alphabets and current input alphabet. In the following, the generation algorithm of the PRs based on the observed input and output is described. Note that the structure of PR which includes past n samples like (2) is referred by n-type structure.

1. Whenever a new alphabet is observed, this is embedded as 0-type structure.

$$U_k \rightarrow y_k \tag{4}$$

2. If more than one n-type rules (n=0,1,2,$\cdots$) contradict with each other for the observation, then these rules are expanded to $(n+1)$-type rules. For example, consider the case wherein the following two contradicting rules are generated based on the observation:

$$y_{k-(n-1)} \cdots y_k\ U_k \rightarrow y_{k-(n-1)} \cdots y_k\ y_{k+1} \tag{5}$$

$$y_{k-(n-1)} \cdots y_k\ U_k \rightarrow y_{k-(n-1)} \cdots y_k\ y^*_{k+1} \quad (y_{k+1} \neq y^*_{k+1}) \tag{6}$$

In this case, these two rules are expanded to the following $(n+1)$-type rules:

$$y_{k-n} y_{k-(n-1)} \cdots y_k\ U_k \rightarrow y_{k-n}\, y_{k-(n-1)} \cdots y_k\, y_{k+1} \tag{7}$$

$$y^*_{k-n} y_{k-(n-1)} \cdots y_k\ U_k \rightarrow y^*_{k-n}\, y_{k-(n-1)} \cdots y_k\, y^*_{k+1} \quad (y_{k-n} \neq y^*_{k-n}). \tag{8}$$

If $k - n < 0$, the conflicting n-type productions are all deleted because there is no sufficient information before the initial time.

Generally speaking, if the system is known to have an order of n, the system can be modelled by n-type PRs. From viewpoint of the computational burden, the restriction of the number of the type must be imposed in order to avoid the generation of unreasonably large number of rules. In many practical situations, however, many contradicting rules like (5) and (6) may be generated because of an ignored fast dynamics and/or measurement noise in the system. As the result, the maximum number of type of the PR and the number of PRs tends to be unreasonably large. Furthermore, many input-output strings may be observed in the detection/diagnosis phase, which does not coincide with any PRs developed in the learning phase. In other word, PRs are not always generated sufficiently because of the insufficiency in the training data. In this paper, the assignment of probability and smoothing strategy are introduced to overcome these problems in the following:

2.2 Assignment of Probability to PRs

First of all, the maximum number of type of the PR is supposed to be restricted to $n+1$. If the system is known to have an order of n, $(n+1)$-type would be enough in the PPR with consideration of the observation noise or the nonlinearity of the system. In this case, some contradicting PRs may exist in the $(n+1)$-type PRs by following the procedure in Sect. 2.1. Note that there are no contradiction in the 0- to n-type PRs. Therefore, the 0- to n-type PRs become deterministic PRs. Next, we divide the $(n+1)$-type PRs into the "contradicting PRs" and the "non-contradicting PRs." Then, probability 1 is assigned to all the non-contradicting PRs. These PRs also lead to deterministic PRs. Finally, we consider the contradicting $(n+1)$-type PRs, that is, the PRs with same antecedent. The occurrence probability is assigned to these PRs as follows:

$$\begin{aligned}&\text{PR } 1 \quad y_{k-n} \cdots y_k U_k \rightarrow y_{k-n} \cdots y_k (y_{k+1})_1 : P_1\\ &\text{PR } 2 \quad y_{k-n} \cdots y_k U_k \rightarrow y_{k-n} \cdots y_k (y_{k+1})_2 : P_2\\ &\qquad\vdots\\ &\text{PR } r \quad y_{k-n} \cdots y_k U_k \rightarrow y_{k-n} \cdots y_k (y_{k+1})_r : P_r \end{aligned} \tag{9}$$

where $(y_{k+1})_j$ denotes an output symbol at $(k+1)$th time instant appearing in the PR j, and must satisfy $(y_{k+1})_g \neq (y_{k+1})_h$ when $g \neq h$. Furthermore, P_i $(i = 1, \cdots, r)$ is an assigned probability for PR i, which is statistically calculated from the observed frequency of the corresponding PR in the

learning phase. Therefore,

$$\sum_{i=1}^{r} P_i = 1 \tag{10}$$

must hold.

2.3 Smoothing of the Probability

In order to overcome the problem of "insufficiency in the training data," the following smoothing procedure is introduced. First of all, for $(y_{k+1})_j$ in (9), a new PR is generated by replacing $(y_{k+1})_j$ with its neighboring output symbol. Note that if the replaced rule already exists, then it will be eliminated. For example, assuming that we have an output alphabet $\{a, b, c, d, e, f, g, h\}$, the neighboring symbols of c are b and d, the neighboring symbol of h is g only. By applying this procedure, the following new PRs are generated.

$$\begin{aligned}
&\text{PR } r+1 \quad y_{k-n}\cdots y_k U_k \to y_{k-n}\cdots y_k (y_{k+1})_{r+1} : P_{r+1}\\
&\text{PR } r+2 \quad y_{k-n}\cdots y_k U_k \to y_{k-n}\cdots y_k (y_{k+1})_{r+2} : P_{r+2}\\
&\qquad\vdots\\
&\text{PR } r+l \quad y_{k-n}\cdots y_k U_k \to y_{k-n}\cdots y_k (y_{k+1})_{r+l} : P_{r+l}.
\end{aligned} \tag{11}$$

If the number of new generated PRs is l, then the total number of the PRs contradicting with each other comes to $(r+l)$.

As the last step, the assigned probabilities are updated by the following equation with assuming the initial value of $P_{r+i}(i = 1, \cdots, l)$ to be 0. For $j = 1 \cdots r + l$,

$$P'_j = \frac{P_j + \alpha \max(P_1, P_2, \cdots P_{r+l})}{1 + (r+l)\,\alpha \max(P_1, P_2, \cdots P_{r+l})}. \tag{12}$$

where P'_j is the updated probability of PR j, α indicates the strength of smoothing and was empirically set to be 0.1 in this work.

The probability of PRs which does not show up after the above procedure is assumed to be small constant ϵ in order to prevent the zero likelihood in the detection/diagnosis phase. In this work, ϵ was set to be 0.0001.

3. Vector Quantization

3.1 Competitive and Selective Learning Quantization

In order to make PRs from the observed input and output data, the data must be quantized. The quantizing process is called vector quantization, which plays an essential role to grasp the dynamic characteristics of the system. In this paper, the competitive and selective learning (CSL) quantization [12] is adopted. The CSL quantization is known to be robust to the initialization of the neurons. In Appendix A and B, the detail of the competitive learning (CL) quantization and the CSL quantization are explained. The CL quantization is an original version of the CSL quantization. The convergence threshold of CSL algorithm in Appendix B was set to be 10^{-10} which was derived empirically.

3.2 Decision of Number of Quantization Levels

The number of the quantization levels, that is, the resolution of the system description is positively correlated to the number of PRs. When the number of PRs becomes larger, the computational burden and necessary memory storage also become larger. As the result, the system will may have some problems in real-time execution.

In the case that the number of the quantization levels is not enough, even when a fault occurs in the target system, it is unlikely that the fault can be detected and diagnosed because of lack of the resolution for the description of the characteristics of the system. On the other hand, if the system is described with too much resolution, wrong detection and diagnosis is likely to occur because the PRs become too much sensitive to the measurement noise. Therefore, the number of the quantization levels has to be decided by taking the balance between the accuracy and the sensitivity. In Sect. 6, an appropriate number of the quantization levels is discussed via an experiment.

4. Fault Detection and Diagnosis

The FDD is composed of two phases, a learning phase and detection/diagnosis phase. In the following sections, the details are explained.

4.1 Learning Phase

In the learning phase, which is an offline process, the PPRs are created from the input-output data sequence of normal and expected faulty operating conditions by the following algorithm:

Learning Algorithm:

At first, the set of expected operating conditions is defined as Ω which consists of normal and expected faulty operating conditions.

1. Initialize: $\Omega' := \{\phi\}$.
2. Choose $\exists\omega \in \{\omega | \omega \in \Omega,\ \omega \notin \Omega'\}$.
3. Observed input and output sequences under the operating condition ω are quantized by using CSL algorithm, wherein the reference vector $\boldsymbol{Y}^{\omega}$ is obtained.
4. Make PRs by the procedure in Sect. 2.1, and assign probabilities to the created PRs by the procedure in Sect. 2.2.
5. Smooth PPRs by the smoothing procedure in Sect. 2.3. Then, the set of PPRs, $\mathcal{R}^{\omega}$ is created.
6. $\Omega' := \Omega' \cup \{\omega\}$ and if $\Omega' = \Omega$ then the learning algorithm terminates, otherwise go to step 2.

4.2 Detection/Diagnosis Phase

The detection/diagnosis phase is executed online, wherein the log-likelihood values of all the operating conditions are calculated and compared.

Detection/diagnosis algorithm:

First of all, the log-likelihood value for the operating condition $\omega \in \Omega$ at time index k is defined as L_k^ω.

1. Initialize: $k := 0$, $L_k^\omega := 0\ \forall\omega \in \Omega$ and observe initial input $\tilde{U}_0$ and output $\tilde{y}_0$.
2. $k := k + 1$
3. Observe the current input $\tilde{U}_k$ and output $\tilde{y}_k$.
4. $\forall\omega \in \Omega$, quantize $\tilde{U}_k$ and $\tilde{y}_k$ to U_k^ω and y_k^ω respectively by the reference vectors $\boldsymbol{Y}^\omega$ and find a PPR $r_k^\omega \in \mathcal{R}^\omega$ which meets the quantized input U_{k-1}^ω and the quantized outputs $y_k^\omega,\ y_{k-1}^\omega, \cdots$.
5. Update log-likelihood values:

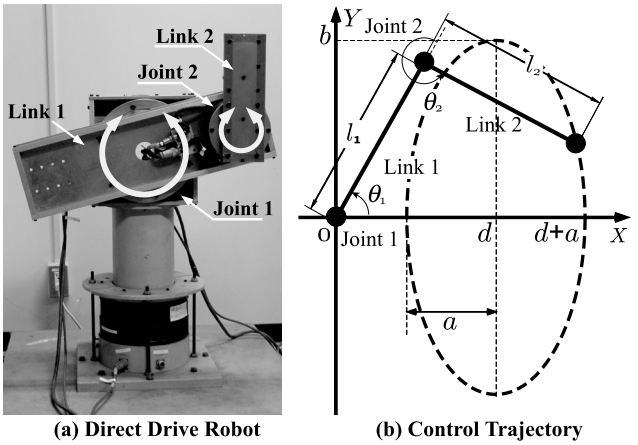
$$L_k^\omega := L_{k-1}^\omega + \log\left(P(r_k^\omega)\right), \tag{13}$$

   where $P(r)$ means the probability of the PPR r.
6. When the log-likelihood value of the data of the normal condition is below a threshold value, some fault is considered to occur.
7. In addition, if the log-likelihood value of the data over some specified faulty condition exceeds the threshold value, the corresponding fault is considered to occur. Otherwise, the current condition is considered to be one under an unexpected failure.
8. Go to step 2.

5. Experimental Results

5.1 Experimental Setup and Faulty Conditions

A two-degree-of-freedom robot with direct drive motors (DD-Robot) used for the experiment is shown in Fig. 1(a). Although the analytical dynamical model of this type of robot is already known, the FDD based on the nonlinear dynamical model often requires much computation. Furthermore, the existence of unmodelled dynamics, such as the coulomb friction, may degrade the accuracy of the model.



(a) Direct Drive Robot (b) Control Trajectory

Fig. 1 Direct Drive Robot (DD-Robot) and control trajectory of DD-Robot ($l_1 = l_2 = 0.24$ [m]).

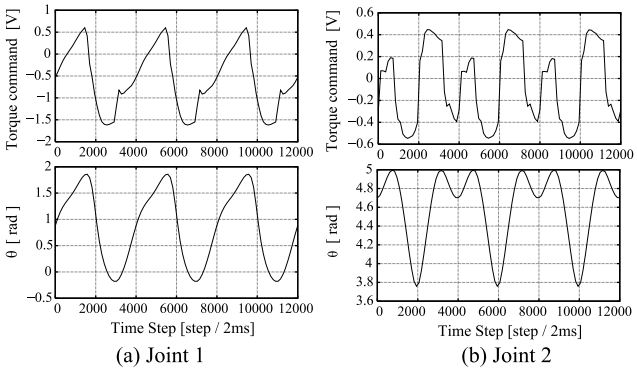
On the other hand, the proposed PPR-based system description can take a balance between the accuracy and the computational complexity. This is a significant advantage of the proposed model. The DD-Robot was supposed to operate so as to follow the trajectory

$$\begin{cases} X = d + a\cos t' \\ Y = b\sin t' \\ t' = 2\pi t/C \end{cases} \tag{14}$$

where C is cycle time and was set to be 8 [sec]. This reference trajectory is shown in Fig. 1(b). In this experiment, $a = 0.1$ [m], $b = 0.3$ [m] and $d = 0.24$ [m]. Each joint was controlled by the PD control with sampling interval 1 [msec]. Figure 2 shows the input and output data acquired from actuators and sensors. The input of the system is a torque [N·m] to actuate each joint, and the output is an angle [rad] of each joint. The input and output data were acquired every 2 [msec], and as the result, 4,000 input and output data were stored for one cycle, respectively. In order to execute the CSL quantization and create PPRs (i.e., the learning phase), data of 50 cycles, that is, 200,000 data were used. On the other hand, in the detection/diagnosis process, the log-likelihood value is calculated at each sampling instant by integrating the probability of corresponding PPR over the past 4,000 samples. The number of quantized symbols of the inputs and outputs was set to be 12 including the upper and lower limit values, and the maximum order of the PPRs was assumed to be 3. In the CSL algorithm, the upper and lower limit values are fixed. The number of quantized symbols were determined based on the average distortion given by (A· 6), the storage capacity, and the computational burden. Table 1 shows a part of PPRs generated under the normal condition (faultless case). The input alphabet consists of 12 symbols $\{A,B,C,\cdots,L\}$, and the output alphabet also consists of 12 symbols $\{a,b,c,\cdots,l\}$. Some samples of the obtained PPRs are shown in Table 1. Note that the PPRs in Table 1 are not smoothed.

PPRs were created under the following four conditions in each joint control system independently:

1. Normal (faultless) condition



(a) Joint 1 (b) Joint 2

Fig. 2 Input and output data profile of DD-Robot under normal condition (3 cycles).

Table 1 Generated PPRs of the joint 1 under normal condition without smoothing: Step 4 of Sect. 4.1.

Rule			Prob.	Rule			Prob.
1-type rule				2-type rule			
bB	→	bb	1.0	cdE	→	cdd	1.0
cD	→	cc	1.0	dcB	→	dcc	1.0
gD	→	gg	1.0	edB	→	edd	1.0
⋮		⋮		gfC	→	gff	1.0
3-type rule (deterministic)				3-type rule (probabilistic)			
cddE	→	cddd	1.0	bbbD	→	bbbb	0.982
dccB	→	dccc	1.0	bbbD	→	bbbc	0.018
eddB	→	eddd	1.0	eeeF	→	eeee	0.986
gffC	→	gfff	1.0	eeeF	→	eeef	0.014
				hhhD	→	hhhg	0.023
				hhhD	→	hhhh	0.977
				⋮			⋮

2. 30% torque fluctuation in the motor of the joint 1
3. Change of the viscous resistance in the joint 1
4. Change of mass in the Link 1

The faulty condition 2 is a case that some power devices in a motor driver become failure. This was imitated by multiplying the torque reference by $(1+0.3\sin(2\pi t))$. The faulty condition 3 is a case of bearing burnoff. This was imitated by adding virtual frictional torque to the torque reference. The faulty condition 4 is a case that unexpected exogenous force is added to a link. This was imitated by hanging 1 kg mass on the tip of the Link 1. These faulty conditions are considered to be incident and reasonable enough to demonstrate the proposed method. PPRs with smoothing were created for the above conditions, respectively. The set of the created PPRs and the corresponding probabilities are labeled by J1F1, J1F2, J1F3, J1F4, J2F1, J2F2, J2F3 and J2F4 respectively. For example, J2F1 means the set of the PPRs and the probabilities of the joint 2 in the case of the faulty condition 1.

The threshold to detect and diagnose a fault in the steps 6 and 7 in Sect. 4.2 was set to be −2000. The threshold should be decided according to the number of quantization levels. Conditions to establish the threshold are discussed in Sect. 6.

5.2 Fault Detection

Figure 3 shows the result in the case that the faulty condition 2 has occurred. The vertical axis represents the log-likelihood value.

In Fig. 3, the fault has occurred at the time step 16000, i.e., the end of the 4th cyclic motion. In this case, the log-likelihood value of J1F1 decreased as shown in Fig. 3(a). This obviously indicates that some fault has occurred. Although the log-likelihood value of J1F1 changed sharply, the log-likelihood value of J2F1 does not show rapid change after the fault occurred. This is because the fault in the joint 1 made little influence on the observation as for the joint 2.

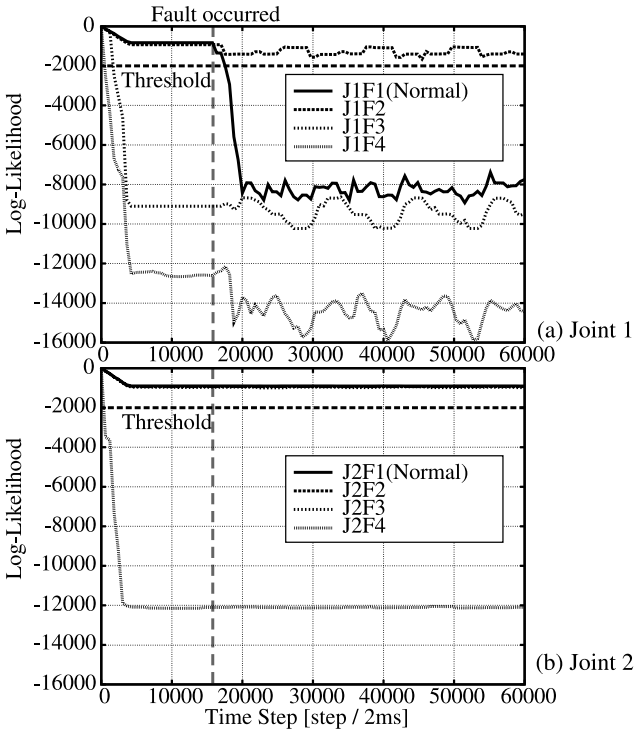


Fig. 3 Log-likelihood when the faulty condition 2 occurred.

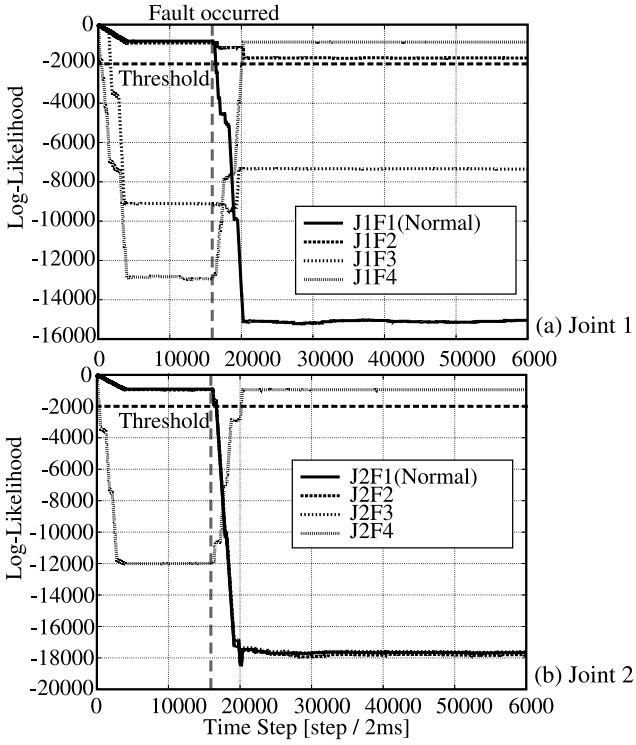


Fig. 4 Log-likelihood when the faulty condition 4 occurred.

5.3 Fault Diagnosis

In Fig. 3, the profiles of the log-likelihood values for each

faulty condition are also depicted in the case that the faulty condition 2 has occurred. The log-likelihood values of J2 (Fig. 3(b)) do not show any symptom that a fault has occurred because the log-likelihood value of the normal condition keeps being high value after the fault has occurred in the joint 1. However, as shown in Fig. 3(a), the condition 2 can be considered as the strong candidate of the fault from the fact that the log-likelihood value of J1F2 keeps being higher than the others after the fault has occurred.

Figure 4 shows the result of the fault diagnosis in the case that the faulty condition 4 has occurred at the time step 16000. In this case, the fault in the Link 2 made influence on the observations of the both joints. Therefore, in Fig. 4(a)(b), the log-likelihood values of J1F1 and J2F1 decreased sharply after the fault had occurred, and then those of J1F4 and J2F4 increased. These results imply that the condition 4 can be considered as the first candidate of the fault. In addition, in Fig. 4(a), the log-likelihood values of J1F2 also keeps being high value. Therefore, the condition 2 is considered as the second candidate. These results reveal that the fault diagnosis is successfully realized.

6. Discussion

In this section, the number of quantization levels is discussed from the viewpoints of the computation burden and the performance of the FDD.

Restriction of the number of quantization levels (i.e., the number of symbols) is an important matter to reduce the computational burden for making PPRs, in particular, in the CSL quantization. Table 2 shows the average distortion and the computation time for the CSL quantization in three cases of different number of the quantization levels. This procedure appears in step 4 of Learning algorithm in Sect. 4.1. When the number of quantization levels is large, the accuracy of the system description becomes higher thanks to the decrease of the average distortion, while the computation time increases. However, the computational time evolves almost linear to the increase of the number of quantization levels.

Table 3 also shows the number of generated PPRs and the computation time to acquire PPRs from the quantized input-output data. This procedure appears in step 4 and 5 of Learning algorithm in Sect. 4.1. When the number of quantization levels is large, the number of generated PPRs becomes higher due to the high resolution of the system description. Although the computation time increases according to increase of the number of quantization levels, the evolution of the computational time is almost linear to it again.

The performance of the FDD also depends on the number of quantization levels. Figure 5 shows profiles of the log-likelihood values of the joint 1 in the faulty condition 1 (J1F1) for various numbers of quantization levels in the case that the faulty condition 2 has occurred at the time step 16000. In this case, the log-likelihood value must become sufficiently small after the fault occurred. As shown in Fig. 5, when the number of quantization levels is larger,

Table 2 Average distortion and computation time of the CSL quantization: The computation time was necessary at the stage 2 in Fig. 1(a), and was evaluated by Pentium4 2.8 GHz.

quantization level	Ave. distortion of the joint 1		Ave. distortion of the joint 2		comp. time
	input	output	input	output	
4	0.0548	0.0629	0.0508	0.0518	1h41m
10	0.0225	0.0238	0.0182	0.0203	2h06m
40	0.0054	0.0058	0.0049	0.0051	7h29m

Table 3 Number of PPRs and computation time to acquire PPRs from quantized input-output data: The computation time was necessary at the stages 3-5 in Sect. 4.1, and was evaluated by PentiumM 1.5 GHz. Note: Type 3/3* is deterministic/probabilistic 3-type PR.

quantization level	Joint	type				comp. time
		1	2	3	3*	
4	1	5	2	2	12	66s
	2	6	3	3	14	
10	1	19	4	4	36	75s
	2	24	15	15	48	
40	1	82	29	31	168	115s
	2	146	111	112	332	

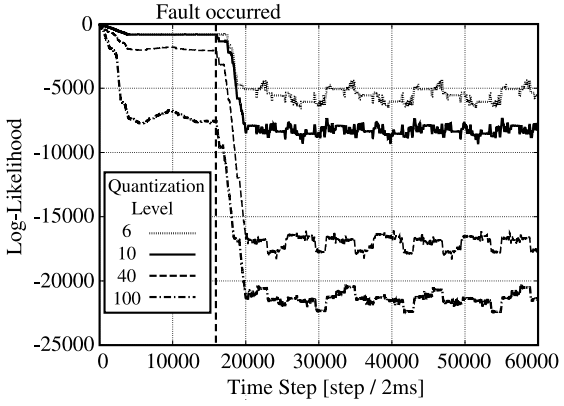


Fig. 5 Log-likelihood of the normal condition for various quantization levels in the case that the faulty condition 2 has occurred.

the log-likelihood value reaches to smaller value after the fault occurred. However, as for the duration before the fault had occurred, the log-likelihood value is relatively smaller when the number of quantization levels is larger. This can be explained as follows: When the number of quantization levels is excessively large, wrong detection and diagnosis is likely to occur because the PRs become too much sensitive to the measurement noise. Additionally, from this reason, the threshold of the detection/diagnosis procedure should be decided appropriately according to the number of quantization levels.

As the result, the number of quantization levels must be decided appropriately taking into account both performance and computational burden.

7. Conclusion

In this paper, Probabilistic Production Rule (PPR) has been

introduced and exploited for the fault detection and diagnosis. Some experiments on the manipulator control confirmed the usefulness of the proposed method. The significant advantage of the PR model based approach is that the designer can take a balance between the accuracy of the system description and the model complexity by appropriately specifying the resolution of the system description. The proposed methodology is applicable to many other kinds of dynamical systems including nonlinear, stochastic, discrete-event, or hybrid systems. A quantitative comparison with other existing methods, and application to other practical systems are our future works.

References

[1] R. Isermann, "Process fault detection based on modeling and Estimation methods," Automatica, vol.20, no.4, pp.387–404, 1984.
[2] R. N. Clark, D. C. Fosth, and V. M. Walton, "Detecting instrument malfunction in control system," IEEE Trans. Aerosp. Electron. Syst., vol.AES-11, no.4, pp.465–473, 1975.
[3] V. Venkatasubramanian, R. Rengaswamy, S.N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis Part I: Quantitative model-based methods," Computer and Chemical Engineering, vol.27, pp.293–311, 2003.
[4] V. Venkatasubramanian, R. Rengaswamy, S.N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis Part II: Qualitative models and search strategies," Computer and Chemical Engineering, vol.27, pp.313–326, 2003.
[5] V. Venkatasubramanian, R. Rengaswamy, S.N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis Part III: Process history based methods," Computer and Chemical Engineering, vol.27, pp.327–346, 2003.
[6] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," IEEE Trans. Autom. Control, vol.40, no.9, pp.1555–1575, 1995.
[7] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis, "Failure diagnosis using discrete-event models," IEEE Trans. Control Syst. Technol., vol.4, no.2, pp.105–124, 1996.
[8] J. Lunze, "Diagnosis of quantized systems based on a timed discrete-event model," IEEE Trans. Syst., Man Cybern., A, Syst. Humans, vol.30, no.3, pp.322–335, 2000.
[9] M. Saito, T. Suzuki, S. Inagaki, and T. Aoki, "Fault diagnosis of event-driven control systems based on timed Markov model with maximum entropy estimation," Proc. International Symposium on Mathematical Theory of Networks and Systems, pp.2197–2202, Kyoto, 2006.
[10] J. F. Martins, J. A. Dente, A. J. Pires, and R. Vilela Mendes, "Language identification of controlled systems: Modeling, control, and anomaly detection," IEEE Trans. Syst., Man Cybern. C, Apple. Rev., vol.31, no.2, pp.234–242, 2001.
[11] N. Ueda and R. Nakano, "A new competitive learning approach based on an equidistortion principle for designing optiomal vector quantizers," Neural Netw., vol.7, no.8, pp.1211–1227, 1994.
[12] N. Ueda and R. Nakano, "A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers," Neural Netw., vol.7, no.8, pp.1211–1227, 1994.

Appendix A: Competitive Learning (CL) Quantization

In the CL quantization, the input vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_k)$ is presented to neurons. Each neuron i is specified by a reference vector $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \cdots, y_{ik}) \in \boldsymbol{R}^k$, and computes the distortion $d(\boldsymbol{x}, \boldsymbol{y}_i)$ between its reference vector and an input vector. The distortion is defined as the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}_i$:

$$d(\boldsymbol{x}, \boldsymbol{y}_i) = \sqrt{\sum_{j=1}^{k} \left(x_j - y_{ij}\right)^2}. \quad (\mathrm{A}\cdot 1)$$

Then a winning neuron c which gives the minimum distortion is selected.

$$d(\boldsymbol{x}, \boldsymbol{y}_c) \le d(\boldsymbol{x}, \boldsymbol{y}_j) \qquad \text{for all } j \ne c. \quad (\mathrm{A}\cdot 2)$$

To reduce the distortion of the winning neuron c, its new reference vector $\boldsymbol{y}_c(t+1)$ is then slightly adjusted toward the current input vector $\boldsymbol{x}(t)$:

$$\boldsymbol{y}_c(t+l) = \boldsymbol{y}_c(t) + \eta(t)[\boldsymbol{x}(t) - \boldsymbol{y}_c(t)]. \quad (\mathrm{A}\cdot 3)$$

The reference vectors of the other losing neurons are unchanged. The nonnegative parameter $\eta(t)$ ($\ll 1.0$), which represents the learning rate for the adjustment of reference vector, decreases monotonically to zero as learning progresses. Thus, the above algorithm practically converges in a finite time. In practice, the reference vectors are modified by a finite number of training vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T\}$. That is, these training vectors are repeatedly utilized for updating the reference vectors until the algorithm converges. With the above algorithm, the resulting performance strongly depends on the distribution of the initial reference vectors.

Appendix B: Competitive and Selective Learning (CSL) Quantization

When a finite number of training data, $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T\}$, is available, it is repeatedly utilized to adjust the reference vectors of neurons until the algorithm converges. Then, in addition to the adjustment, the selection is performed every T learning times. Note that to practically guarantee the convergence of the selection itself, the number of neurons that are subject to the selection must monotonically decrease after every selection. Thus, the selection always ends in a finite number of times. The details of the selection algorithm are given in the following:

CSL Algorithm:

1. Initialization: Given N (the number of neurons), a training set $\boldsymbol{X}$, convergence threshold ζ, and initial reference vectors of neurons $\boldsymbol{Y}^{(0)}$. Set $m = 1$ and $D(0) = \infty$. Let $s(m)$ be the number of neurons that are subject to the selection at the mth iteration in the CSL algorithm.
2. Training: For all $\boldsymbol{x} \in \boldsymbol{X}$, perform the conventional competitive learning algorithm described in Appendix A.
3. Selection: If $s(m) > 2$, choose $s(m)$ neurons alternately from both the first and the last rank of the sub-distortion D_j given by (A·4) and perform selection algorithm (see later) for the chosen neurons and replace m with $m + 1$ and go to step 2. Otherwise go to step 4.

4. Convergence test: Compute the average distortion $D(m)$ given by (A·6) using the final reference vectors obtained in step 2. If $[D(m-1) - D(m)]/D(m) \leq \zeta$, terminate with $\boldsymbol{Y}^{(m)}$ as the final reference vectors. Otherwise replace m with $m+1$ and go to step 2.

The sub-distortion D_j in step 3 is computed by

$$D_j = \frac{1}{T} \sum_{\boldsymbol{x} \in S_j} d(\boldsymbol{x}, \boldsymbol{y}_j). \tag{A·4}$$

Note that the divisor is not t_j but T. S_j denotes the *Voronoi region* belonging to $\boldsymbol{y}_j$. That is,

$$S_j = \bigcap_{k,k \neq j} \{\boldsymbol{x} | d(\boldsymbol{x}, \boldsymbol{y}_j) \leq d(\boldsymbol{x}, \boldsymbol{y}_k)\}. \tag{A·5}$$

The average distortion in step 4 at the mth iteration is calculated by

$$D(m) = \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}} \min_{\boldsymbol{y} \in \boldsymbol{Y}^{(m)}} d(\boldsymbol{x}, \boldsymbol{y}), \tag{A·6}$$

where $\boldsymbol{Y}^{(m)}$ is the final reference vectors after step 2. The number $s(m)$ is forced to decrease monotonically as m increases.

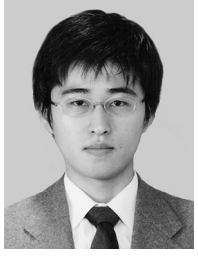
The selection algorithm for the $s(m)$ neurons is given as follows. Note that for simplicity, in the following algorithm, the indices of these chosen $s(m)$ neurons are rearranged and referred to as $j = 1, 2, \cdots, s(m)$.

Selection Algorithm:

1. Compute a normalized fitness measure using the sub-distortion D_j:

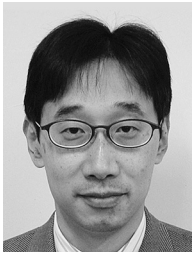
$$g_j = D_j^{\gamma} \Big/ \sum\nolimits_{j=1}^{s(m)} D_j^{\gamma}, \tag{A·7}$$

   where γ (< 1) is a nonnegative constant.
2. Determine the number of neurons to be reproduced: Compute $[g_j s(m)] (\equiv u_j)$ for all j where $[a]$ denotes the largest integer less than or equal to a. For top $s(m) - \sum_{j=1}^{s(m)} u_j$ neurons with respect to the value of $g_j s(m) - u_j$, add one to each u_j.
3. For every j, reproduce u_j neurons with adding random perturbation vectors δ_{jl} ($l = 1, 2, \cdots, u_j$) to $\boldsymbol{y}_j$, where $\|\delta_{jl}\| \ll \|\boldsymbol{y}_j\|$. If $u_j = 0$, the neuron is eliminated. Terminate.



Shinkichi Inagaki was born in 1975. He received the Ph.D. degrees from Tokyo University, JAPAN in 2003. Currently, he is an assistant professor of the Department of Mechanical Science and Engineering of Nagoya University. Also, he is a visiting researcher of Bio-Mimetic Control Research Center, RIKEN in Nagoya.

Koudai Hayashi was born in 1981. He received the M.E. degrees from Nagoya University, JAPAN in 2006. Currently, he is with NEC Corporation.



Tatsuya Suzuki was born in 1964. He received the Ph.D. degrees from Nagoya University, JAPAN in 1991. From 1998 to 1999, he was a visiting researcher of the Mechanical Engineering Department of U.C.Berkeley. Currently, he is a professor of the Department of Mechanical Science and Engineering of Nagoya University. He won the paper award from IEEJ in 1995.