Construction of Fuzzy Inference Rules by NDF and NDFL

Isao Hayashi, Hiroyoshi Nomura, Hisayo Yamasaki, and Noboru Wakami

Matsushita Electric Industrial Co., Ltd., Moriguchi, Osaka, Japan

ABSTRACT

Whereas conventional fuzzy reasoning lacks determining membership functions, a neural network driven fuzzy reasoning (NDF) capable of determining membership functions uniquely by an artificial neural network is formulated. In an NDF algorithm the optimum membership function in the antecedent part of fuzzy inference rules is determined by a neural network, while in the consequent parts an amount of reasoning for each rule is determined by other plural neural networks. On the other hand, we propose a new algorithm that can adjust inference rules to compensate for a change of inference environment. We call this algorithm a neural network driven fuzzy reasoning with learning function (NDFL). NDFL can determine the optimal membership function and obtain the coefficients of linear equations in the consequent parts by the searching function of the pattern search method. In this paper, inference rules for making a pendulum stand up from its lowest suspended point are determined by the NDF algorithm for verifying its effectiveness. The NDFL algorithm is formulated and applied to a simple numerical example to demonstrate its effectiveness.

KEYWORDS: ***fuzzy reasoning, fuzzy logic, neural network, membership functions, learning function***

INTRODUCTION

Extensive applications of fuzzy reasoning for various control problems have been reported (Hirota [1]). However, in these cases, fuzzy reasoning is generally involved with tuning problems (Lee [2]). That is, the form of the

Address correspondence to Isao Hayashi, Matsushita Electric Industrial Co., Ltd., Moriguchi, Osaka, 570, Japan.

International Journal of Approximate Reasoning 1992; 6:241-266

655 Avenue of the Americas, New York, NY 10010 0888-613X/92/$5.00

fuzzy number of antecedent and consequent parts of fuzzy inference rules has to be adjusted to minimize the difference between estimation of fuzzy reasoning and output data for a given input data.

A neural network driven fuzzy reasoning (NDF for short) (Hayashi et al. [3], Takagi and Hayashi [4]) by which inference rules are constructed from the learning function of neural networks (Anderson and Rosenfield [5], Tank and Hopplied [6]) for solving tuning problems was previously reported. NDF is a type of fuzzy reasoning having an error backpropagation type of neural network (Rumlhart et al. [7]) that represents fuzzy sets in its antecedent, while another plural error backpropagation type of neural network represents a relationship between input and output data of the consequent of each rule. NDF can obtain the optimal membership function and inference rules from the observed input-output data. However, NDF is unable to alter its inference rule when an environment for constructing that rule is dynamically changing. Thus, we propose a new algorithm that can adjust its inference rules in response to changes in the inference environment. We call this algorithm neural network driven fuzzy reasoning with learning function (NDFL). NDFL can determine the optimal membership function in the same way as NDF by a learning function of the error backpropagation type of neural network and obtain the coefficients of linear equations (Sugeno and Kang [8]) in the consequent parts by the searching function of the pattern search method (Hooke and Jeeves [9]).

In this paper, an algorithm for constructing inference rules based on NDF is introduced first, and an experimental verification of its effectiveness is performed taking as an example an inverted pendulum system. Furthermore, the NDFL algorithm is formulated and applied to a simple numerical example to demonstrate its effectiveness. Since the fuzzy set of the antecedent and the input-output relationship between consequent parts can be determined by means of NDF and NDFL without fine tuning of inference rules by utilizing the neural network learning function acquired from the input-output data, it is advantageous to solve tuning problems of fuzzy reasoning.

NEURAL NETWORK DRIVEN FUZZY REASONING (NDF)

In NDF, the membership function in the antecedent part is determined in multidimensional space. For example, the conventional fuzzy inference rules for representing a fuzzy model (Sugeno and Kang [8]) shown below are considered.

$$R_1\text{: IF } x_1 \text{ is } F_{SL} \text{ and } x_2 \text{ is } F_{SL}, \quad \text{THEN } y_1 = a_{10} + a_{11}x_1 + a_{12}x_2 \tag{1a}$$

$$R_2\text{: IF } x_1 \text{ is } F_{SL} \text{ and } x_2 \text{ is } F_{BG}, \quad \text{THEN } y_2 = a_{20} + a_{21}x_1 + a_{22}x_2 \tag{1b}$$

$$R_3\text{: IF } x_1 \text{ is } F_{BG}, \qquad \text{THEN } y_3 = a_{30} + a_{31}x_1 \qquad (1c)$$

wherein x_1 and $x_2, \ldots, x_{31}$ are input variables; y_1, y_2, y_3 are output variables; a_{10} and others are coefficients; and F_{SL} and F_{BG} are fuzzy numbers where SL and BG mean small and big, respectively. Since the antecedent of fuzzy inference rule R_1 means both x_1 and x_2 are small, the fuzzy set $F_1 = F_{SL} \times F_{SL}$ can be constructed in a partial space of input as shown in Figure 1. Fuzzy sets $F_2 = F_{SL} \times F_{BG}$ and $F_3 = F_{BG}$ for R_2 and R_3 can be obtained similarly. Since the boundary between partial spaces is vague, the boundary is shown by a hatched line. This means that an input space consisting of x_1 and x_2 is divided into individual partial spaces by a number of fuzzy rules, and the fuzzy sets of the antecedent of each inference rule are constructed in each partial space. The NDF algorithm determines these fuzzy sets of antecedent parts constructed in a partial space of input by utilizing the backpropagation type of network. In NDF, the fuzzy inference rules are represented by the IF-THEN format.

$$\begin{aligned} R_s\text{: IF } \mathbf{x} &= (x_1, x_2, \ldots, x_n) \text{ is } A_s, \\ \text{THEN } y_s &= NN_s(x_1, x_2, \ldots, x_m), \\ s &= 1, 2, \ldots, r; \quad m \le n \end{aligned} \qquad (2)$$

The number of inference rules employed here is expressed by r, A_s represents a fuzzy set of the antecedent part of each inference rule, and $NN_s(x_1, x_2, \cdots, x_m)$ denotes a structure of model function that is characterized by and M-layer backpropagation neural network for a given input $(x_1, x_2, \ldots, x_m)$ and output y. The degree of attribution of input $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to the antecedent part of the sth inference rule is derived from the membership value of fuzzy sets A_s to the input $\mathbf{x}$. Furthermore, the amount of operations y_s of consequent parts is estimated for a case where a combination of input variables $(x_1, x_2, \ldots, x_m)$ is substituted in the input layer of the backpropagation neural network. The number of variables employed is m according to a

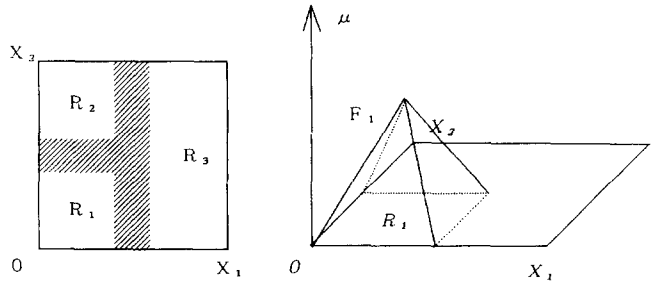


Figure 1. Conventional fuzzy partition of rules.

method for selecting the optimum model employing a backpropagation type of neural network (Takagi and Hayashi [4]). The estimated value is obtained by a calculation of the center of gravity for y_s (Sugeno and Kang [8]). A typical rule division performed by NDF is shown in Figure 2, which is a nonlinear division different from the rectangular divisions shown in Figure 1.

The backpropagation type of neural network is constructed by a general type of processing units found in the neural system, and the processing unit in a neural network shares some of the physical properties of real neurons; the processing unit is called a neuron here. Figure 3 shows an example of fundamental layered backpropagation neural networks containing four layers, where the first layer is called the input layer, the fourth layer is the output layer, and the other layers are called intermediate layers. The structure of model function $\mathrm{NN}(x_1, x_2, \cdots, x_m)$ is characterized by M layers $[u_1 \times u_2 \times \cdots \times u_M]$, where $u_i, i = 1, 2, \ldots, M$, are the numbers of neurons within the input, intermediate, and output layers, respectively. Figure 3 shows the structure of a backpropagation type of neural network consisting of four layers $[3 \times 2 \times 2 \times 2]$.

Next, let's explain how to determine membership functions in the antecedent part by using the neural network $\mathrm{NN}_{\mathrm{mem}}$ shown in Figure 4. The fundamental considerations made on the membership functions in the antecedent part are shown in Figure 4. For example, we consider the backpropagation type of neural network of which input and output layers are input the ith data (x_{i1}, x_{i2}), $i = 1, 2, \ldots, N$, and the data attribution to the rule expressed by (R_1, R_2, R_3), respectively. The estimated values of the backpropagation neural network are considered the membership values of fuzzy sets in the antecedent part because the estimated value represents the attribution of data to each rule.

For an explanation for the NDF algorithm, refer to the NDF block diagram shown in Figure 5. The algorithm used to obtain the inference rules and the estimate y_i^* for the ith input data $\mathbf{x}_i$, $i = 1, 2, \cdots, N$ is as follows.

STEP 1 Select input variables $x_1, x_2, \ldots, x_n$, which are related to the control

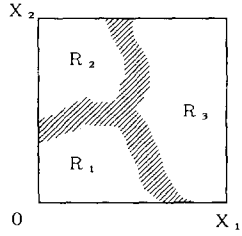


Figure 2. Proposed fuzzy partition of rules.

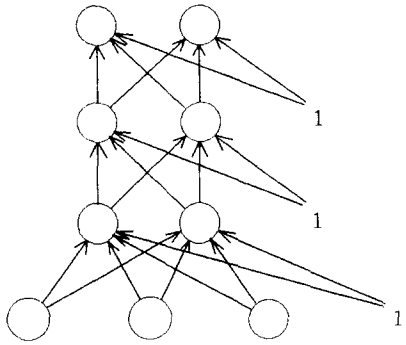


Figure 3. Example of neural network.

value y. This is for an assumed case where the ith input-output data $(y_i, \mathbf{x}_i) = (y_i, x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, 2, \ldots, N$, are obtained and the input data x_{ij}, where $j = 1, 2, \ldots, n$, are the ith data of input variable x_j.

STEP 2 Divide input-output data into r classes of R_s, where $s = 1, 2, \ldots, r$. As mentioned before, each partition is regarded as an inference rule R_s, and the input-output data for each R_s are expressed by $(y_i^s, \mathbf{x}_i^s)$, where $i = 1, 2, \ldots, N_s$, provided that N_s is a number of input-output data for each R_s.

STEP 3 Determine membership functions in the antecedent part by using the neural network $\mathrm{NN}_{\mathrm{mem}}$ shown in Figure 5 provided that the structure of the backpropagation neural network is M-layered $[n \times u_2 \times \cdots \times u_{M-1} \times r]$.

STEP 4 Determine models in the consequent part by using neural networks $\mathrm{NN}_1, \mathrm{NN}_2, \ldots, \mathrm{NN}_r$ shown in Figure 5 provided that the structure of each backpropagation type of network NN_s is M-layered $[k \times u_2 \times \cdots \times u_{M-1} \times 1]$, $k = n, n-1, \ldots, 1$, and select the optimum model for each NN_s.

We propose the stepwise procedure for utilizing the backpropagation type of neural network for determining input variables in the consequent part. The stepwise procedure for utilizing the backpropagation type of neural network is described as follows.

STEP 4-1 Setting a condition at $k = n$, the input data $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$, $i = 1, 2, \ldots, N$, are assigned for the input layer of each NN_s, and the output data y_i is assigned for the output layer of each NN_s, where the input variables assigned for the input and output layers are expressed respectively by

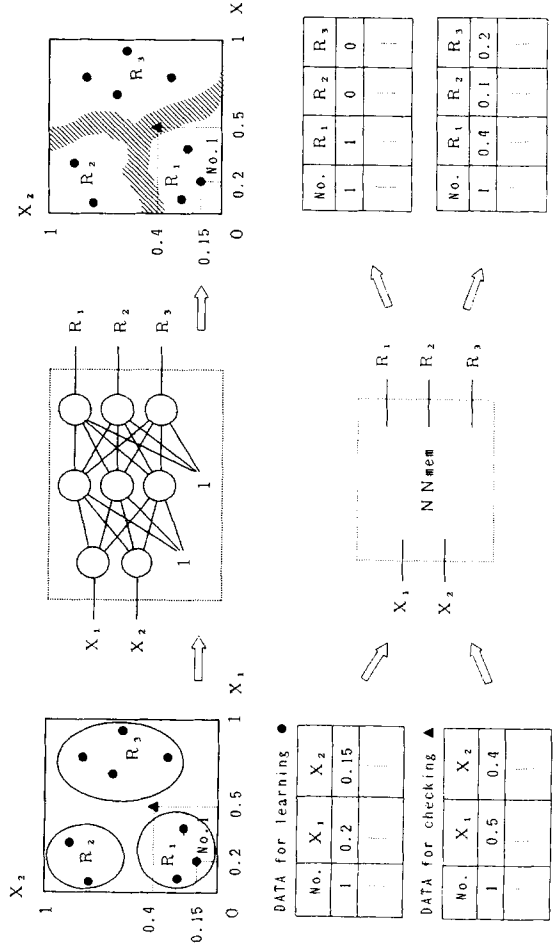


Figure 4. Decision of membership function in antecedent parts of rules.

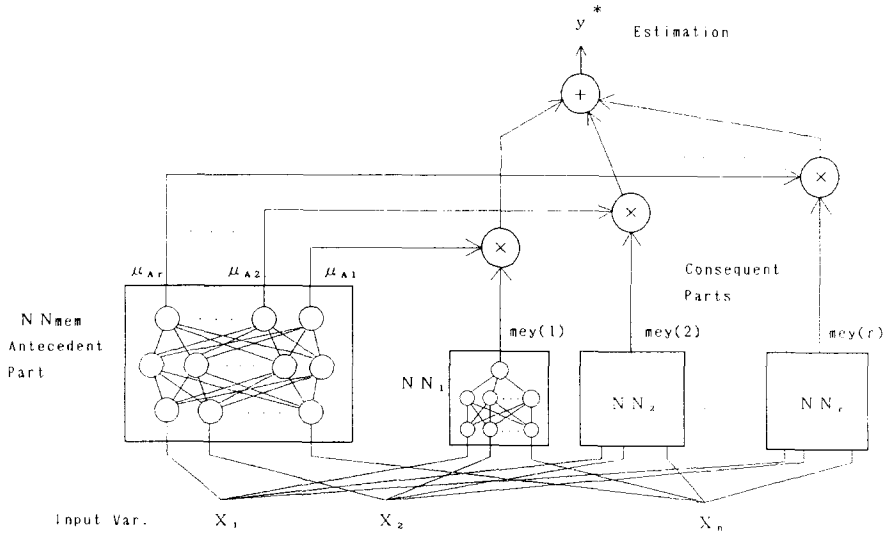


Figure 5. Block diagram of neural network driven fuzzy reasoning.

$$Q_s = \{x_1, x_2, \ldots, x_k\} \tag{3}$$

and

$$T_s = \{y\} \tag{4}$$

where Q_s represents a set of input variables assigned for the input layer of each backpropagation type of neural network NN_s, and T_s represents a set of output variables assigned for the output layer of NN_s.

STEP 4-2 An estimation of ey_i for the input data $x_{i1}, x_{i2}, \ldots, x_{ik}$ can be obtained after repeated learnings made on the backpropagation neural network NN_s. However, the number of learnings is set at approximately 3000. Then the sum of mean squared errors of output data y_i and the estimate ey_i is calculated to obtain an evaluation value of J_k^s required for determining the input variables.

$$J_k^s = \left[\sum_{i=1}^{N} (y_i - ey_i)^2\right]/N, \qquad s = 1, 2, \ldots, r \tag{5}$$

STEP 4-3 In order to determine the correlation of input variables x_j to the output variables y, the input variable x_j is temporarily removed from the set of input variables $x_1, x_2, \ldots, x_k$. The input data from which the input variable x_j is removed, $x_{i1}, \ldots, x_{ij-1}, x_{ij+1}, \ldots, x_{ik}$, where $i = 1, 2, \ldots, N$, are assigned to the input layer of the M-layer backpropagation neural network $[k - 1 \times u_2 \times \cdots \times u_{M-1} \times 1]$, and the output data y_i are

assigned to the output layer. Then the estimate ey_i' for the input data $x_{i1}, \ldots, x_{ij-1}, x_{ij+1}, \ldots, x_{ik}$ can be obtained after the backpropagation learning. An evaluation value J_{k-1}^{sj} required for determining the input variables is derived by calculating the sum of mean squared errors of output data y_i for this estimate ey_i:

$$J_{k-1}^{sj} = \left[\sum_{i=1}^{N} \left(y_i - ey_i' \right)^2 \right] / N, \qquad s = 1, 2, \ldots, r \tag{6}$$

The same calculations are carried out for the input variables other than x_j to obtain the evaluations $J_{k-1}^{s1}, J_{k-1}^{s2}, \ldots, J_{k-1}^{sj}, \ldots, J_{k-1}^{sk}$. The minimum value, J_{k-1}^{sc}, can be obtained by calculating

$$J_{k-1}^{sc} = \min_j J_{k-1}^{sj}, \qquad j = 1, 2, \ldots, k \tag{7}$$

Equation (7) shows that the evaluation of J_{k-1}^{sc} obtained by removing the input variables x_c from the set of input variables takes a minimum value among equations $J_{k-1}^{s1}, J_{k-1}^{s2}, \ldots, J_{k-1}^{sj}, \ldots, J_{k-1}^{sk}$.

STEP 4-4 Comparing the value of J_{k-1}^{sc} of Eq. (7) to the value of J_k^s of Eq. (5), the set of variables Q_s is altered as follows.

$$Q_s = \{x_1, x_2, \ldots, x_{c-1}, x_{c+1}, \ldots, x_k\}, \qquad \text{if } J_{k-1}^{sc} < J_k^s \tag{8}$$

$$Q_s = \{x_1, x_2, \ldots, x_k\}, \qquad \text{if } J_{k-1}^{sc} \geq J_k^s \tag{9}$$

When Eq. (8) is established, the sum of the mean squared errors can be decreased by removing the input variables x_c, and this means that the estimate ey_i' represents y_i better than ey_i. Thus, the correlation of input variables x_c to the output variables y is considered weak, and the input variables can be removed from the input variable sets Q_s. As a result of this, a set of newly established input variables consists of $k - 1$ input variables.

On the other hand, the effectiveness obtained by removing input variables temporarily cannot be attained when Eq. (9) is established. This fact means that the input variables x_c are strongly correlated to the output variables y, and the number of set of input variables Q_s is left unchanged as k.

In cases where the input variables can be reduced, k is altered to $n - 1$, $n - 2, \ldots, 1$; step 4 is repeated until Eq. (9) can be established; and the procedures for reducing the input variables of the backpropagation type of neural network NN_s are completed until Eq. (9) can be established.

Thus the backpropagation type of neural network NN_s having the final set of input variables expressed as $Q_s = \{x_1, x_2, \ldots, x_m\}$ obtained at the time the procedure is completed becomes an optimum backpropagation type of neural

network representing the structure of the consequent part of the rule R_s. The same step procedure is conducted for each NN_s to determine the consequent parts of all the inference rules.

STEP 5 The estimate y_i^* can be derived from the equation

$$y_i^* = \frac{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, \ldots, x_{in}) \times mey_i(s)}{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, \ldots, x_{in})}, \qquad i = 1, 2, \ldots, N \quad (10)$$

where $mey_i(s)$ is an estimate obtained by the optimum backpropagation type of neural network NN_s derived in step 4.

Figure 5 shows that the estimate y_i^* can be derived from the results obtained by conducting product operations between the membership values of the antecedent of each inference rules, i.e., $\mu_{As}(x_{i1}, x_{i2}, \ldots, x_{in})$ and the estimate of the consequent, i.e., $mey_i(s)$, and by continuously conducting summation operations between each pair of rules. Figure 5 shows, however, a case where a condition of $\mu_{As}(x_{i1}, x_{i2}, \ldots, x_{in}) = 1$ is established.

Although it is also possible to determine an overall nonlinear relationship by using only one backpropagation type of neural network, the determination of the overall input-output relationship by applying a backpropagation type of neural network for each partial space is considered more advantageous than employing only one backpropagation neural network for better clarification of overall nonlinear relationship.

APPLICATION TO AN INVERTED PENDULUM SYSTEM

The NDF we propose is capable of forming inference rules automatically, that is, the function is self-autotuning, and shown here is an inverted pendulum system to which a learning function that uses NDF is applied. In the algorithm employed for this experiment, four inputs and one output data are acquired by observing manual operating controls, and fuzzy inference rules and membership functions are then automatically constructed from the acquired data by using an NDF algorithm.

Figure 6 shows the structure of an inverted pendulum system that consists of four elements:

1. cart, which runs on a rail.
2. pendulum that can rotate freely around an axis on the cart.
3. Motor that drives the cart.
4. Fixed pulleys and a belt system for combining the three parts listed above.

The pendulum angle apart from the perpendicular θ degree and the distance from the original cart position are detected by the potentiometers a and b, respectively, shown in Figure 6. These values are digitized by an A/D converter and are fed to a personal computer wherein the velocities of inverted pendulum angle and the cart distance are derived from the four variables, pendulum angle, angular velocity, cart distance, and cart velocity by using an NDF algorithm. As the motor control signal derived by the personal computer takes a digital form, it is converted into an analog value through a D/A converter.

The configuration of the inverted pendulum system and the control computer are as follows.

Body of the inverted pendulum system	Length 1410 mm, width 400 mm, height 880 mm
Pendulum	Length 400 mm, weight 40 g, diameter 4 mm
Drive force	25-W dc motor with gear ratio of 12.5:1
Sensors	Potentiometer to measure the distance from the original position of the cart, and another to measure the pendulum angle.
Microcomputer	CPU 80286
Program	C language, 21K bytes

The inverted pendulum system has two control areas—a linear control area where the pendulum stands upright, and a nonlinear control area where the pendulum falls. We constructed an inverted pendulum system in the linear

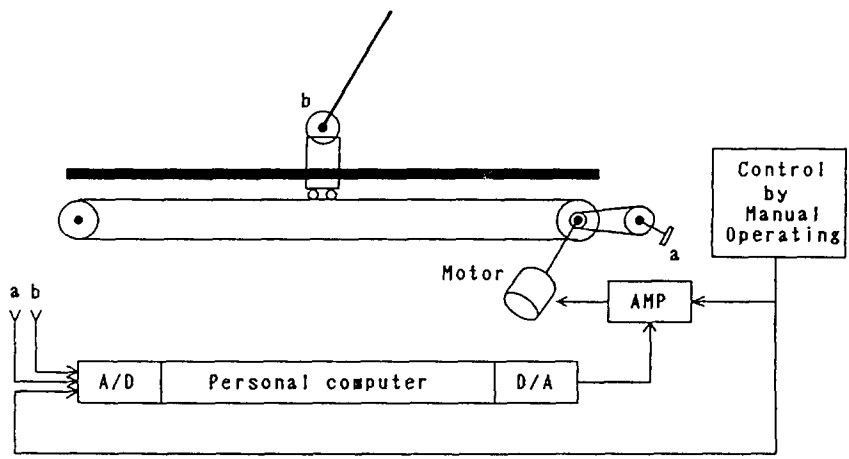


Figure 6. Structure of inverted pendulum system.

control area by using a conventional fuzzy control, and a control model in the nonlinear control area by utilizing NDF.

Control rules applicable to the inverted pendulum were formulated according to an algorithm developed for constructing the inference rules by applying NDF described in the following.

STEP 1 Prepare input-output data. This is performed by an operator who tries to swing up the pendulum by moving the cart in the right or left direction on the rail by pressing either of the corresponding controller buttons until the pendulum is brought to its inverted position, and the following input-output data with a sampling period of 4 ms are recorded.

OUTPUT VARIABLE

y Motor control signal (V)

INPUT VARIABLES

x_1 Distance from the original cart position
x_2 Velocity of x_1 (cm/s)
x_3 Pendulum angle (deg)
x_4 Velocity of x_3 (deg/s)

Approximately 1000–3000 data were acquired from these manual operations, and from these, the 98 input-output data shown in Table 1 applicable to NDF were extracted.

Table 1. Input and Output Data of Inverted Pendulum System

No.	Input Data				Output Data
	x_1 (cm)	x_2 (cm/s)	x_3 (deg)	x_4 (deg/s)	y [V]
1	−1.1482	0.0000	178.5074	0.0000	0.7597
2	−0.0201	8.5486	180.9129	34.5660	0.7421
3	3.2197	29.9073	185.6439	34.5660	0.7617
4	7.8338	38.4472	188.4660	0.0000	0.0039
5	10.9510	4.2697	182.7386	−121.0536	−0.7168
6	9.2718	−21.3586	165.3085	−155.6554	−0.7968
7	5.3319	−38.4560	151.5283	−69.1678	−0.7519
8	0.1432	−42.7261	150.9847	51.8840	−0.7519
⋮					
93	7.9980	42.7261	85.663	−380.4464	−0.0136
94	11.7713	4.2701	199.1743	−639.8375	−0.7265
95	10.6843	−17.0885	117.4961	−622.5536	−0.7519
96	7.0135	−42.7261	56.6514	−345.8786	−0.7519
97	1.9891	−38.4560	27.0170	−138.3357	0.0039
98	−2.9937	−34.1774	16.6419	−51.8822	−0.0019

STEP 2 Set two rules for the input-output data containing the data distributions.

STEP 3 Determine the membership functions of the antecedent part. A three-layer [4 × 6 × 2] backpropagation type of neural network is employed here for determining the antecedent part construction, and the number of learnings is set at 1500.

STEP 4 Determine the consequent part structure. A three-layer [$k \times 6 \times 1$], $k = 4, 3, 2, 1$, backpropagation type of neural network for determining the consequent part structure is employed here, and the number of learnings of each backpropagation type of network is set at 3500.

By using a stepwise procedure for utilizing backpropagation type neural networks, we obtain

$$J_4^1 = 0.016 \tag{11}$$

$$J_3^{11} = \min_j J_s^{1j} (= 0.007), \qquad j = 1, 2, 3, 4 \tag{12}$$

Therefore,

$$J_3^{11} < J_4^1 \tag{13}$$

By removing the input variables x_1, we obtained $Q_s = \{x_2, x_3, x_4\}$. As for $Q_s = \{x_2, x_3, x_4\}$, the following can be obtained.

$$J_3^{11} = 0.007 \tag{14}$$

$$J_2^{13} = \min J_2^{1j} (= 0.021), \qquad j = 1, 2, 3, 4 \tag{15}$$

This means

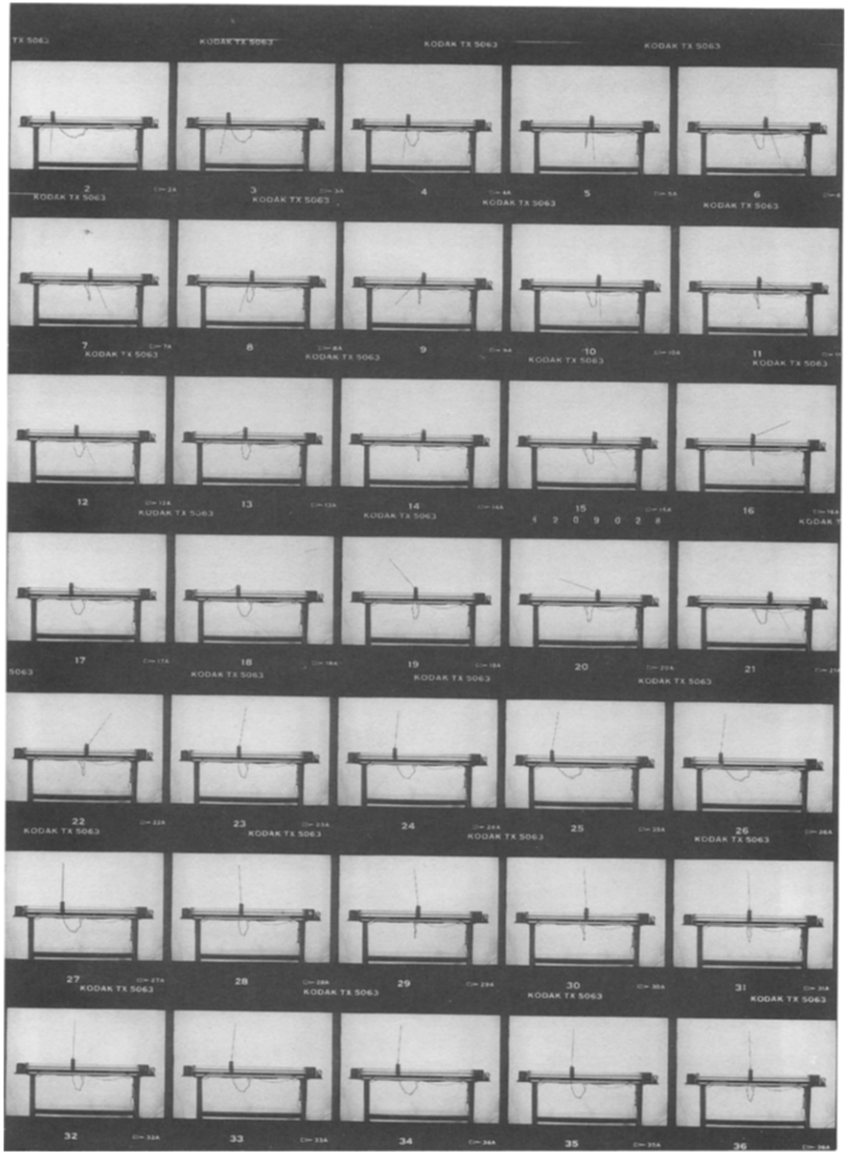
$$J_2^{13} > J_3^{11} \tag{16}$$

Thus, the number of input variables is not reduced, and the algorithm for rule 1 is completed by the second calculation process. The algorithm for rule 2 is completed by the second calculation process in the same way. The inference rules consequently obtained by these are

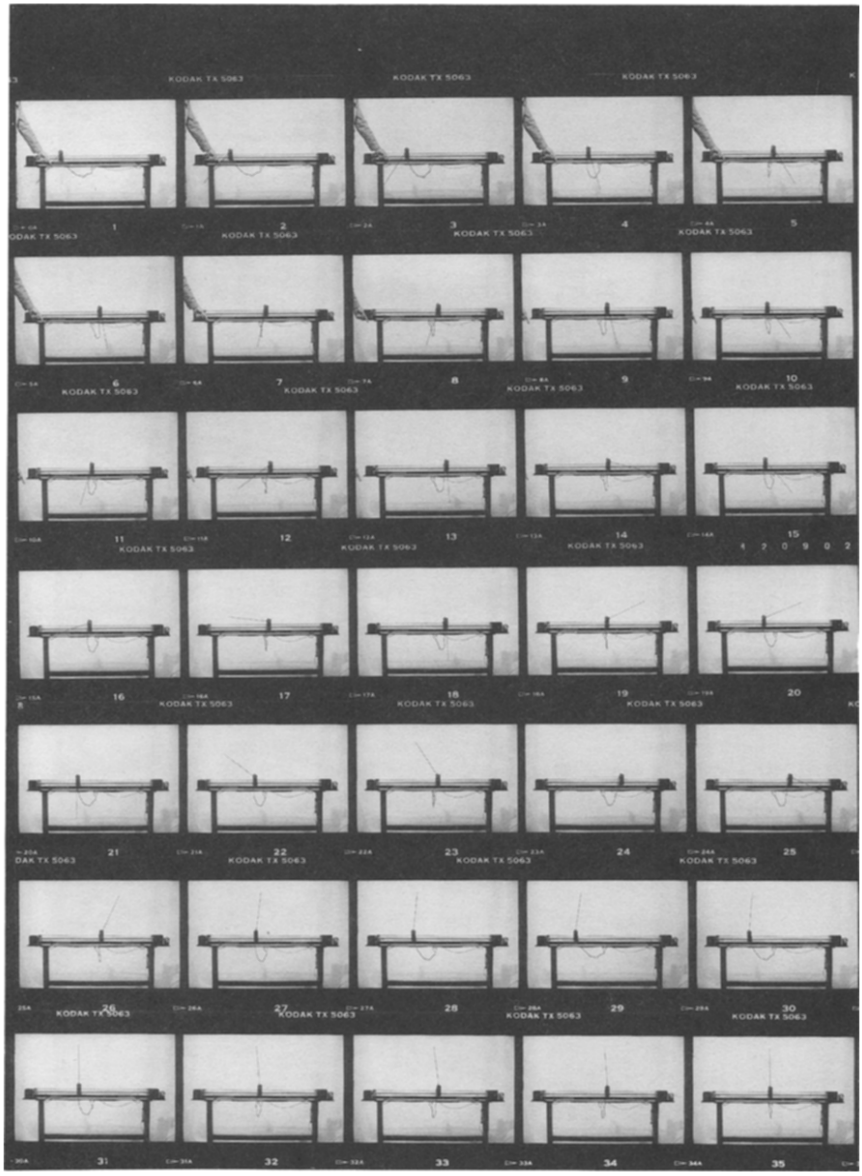
$$R_1\text{: IF } \mathbf{x} = (x_1, x_2, x_3, x_4) \text{ is } A_1, \quad \text{THEN } y_1 = \mathrm{NN}_1(x_2, x_3, x_4) \tag{17a}$$

$$R_2\text{: IF } \mathbf{x} = (x_1, x_2, x_3, x_4) \text{ is } A_2, \quad \text{THEN } y_2 = \mathrm{NN}_2(x_1, x_2, x_4) \tag{17b}$$

Photographs 1 and 2 show the swing-up motions of the pendulum controlled by fuzzy inference rules expressed by Eqs. (17a) and (17b). Photograph 1 shows sequential motions as the pendulum swung from its stable equilibrium state to an inverted standstill state. The estimate y_i^* can be derived from Eq.



Photograph 1. Control of inverted pendulum system No. 1.



Photograph 2. Control of inverted pendulum system No. 2.

(10). The pendulum can be brought to its inverted position regardless of the cart position on the rail. Photograph 2 shows the other control of swing-up motion for given pendulum angles.

An experimental study of the robustness of control performed by NDF was carried out by changing the parameters that govern the dynamic characteristics of the controlled object, and the length of pendulum was taken as a parameter governing the dynamic characteristics of the pendulum here. The initial position of the cart was set at the center position of the belt on which the inverted pendulum device is mounted, and the pendulum angle was set at 0° when it was hung down initially and ±180° is specified when the pendulum was at an inverted position. The angle was incremented for clockwise rotation and decremented for counterclockwise rotation.

The inference rule was constructed for a case where the pendulum length was set at 40 cm, and Figure 7 shows the response of such a pendulum. Figures 8, 9, and 10 show the responses of pendulums 20, 30, and 50 cm long, respectively. These inference rules were constructed for a case where the pendulum length was set at 40 cm. The shifts of pendulum angle are shown by solid lines, and the changes of angular velocity are shown by broken lines in Figures 8–10. However, only the changes of pendulum angle and angular velocity until the pendulum comes to an inverted position, and no response after completion of inversion, are shown there. As for the learning of the inverted pendulum, the learning of the swing-up process was made for constructing an inference rule applicable to the process of a pendulum starting from a downward-hanging position and proceeding to a nearly inverted position. The inverted position is defined as a pendulum angle close to ±180°, and its angular velocity is nearly zero at that time.

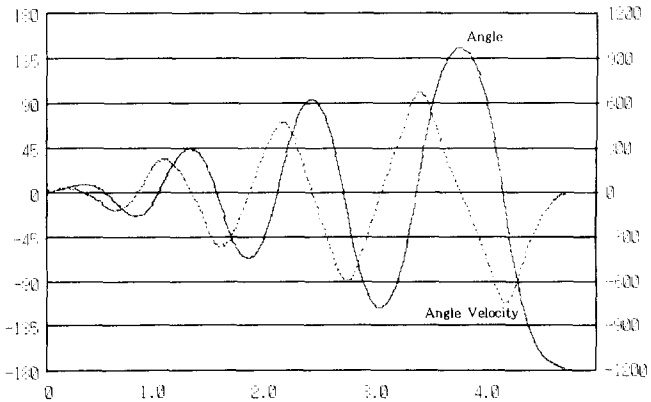


Figure 7. Angle and angle velocity of 40-cm long pendulum.

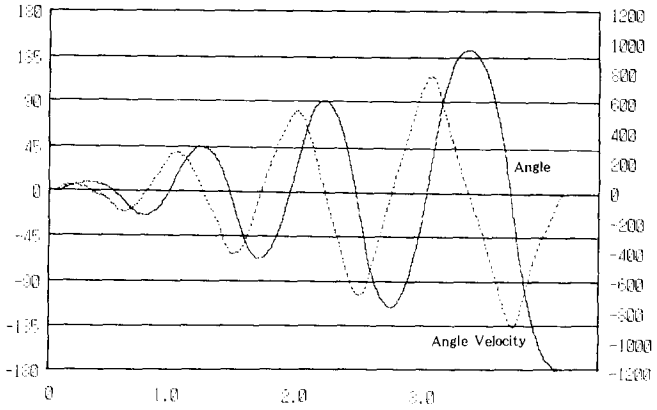


Figure 8. Angle and angle velocity of 30-cm long pendulum.

As shown in Figure 7, the pendulum reached the $-180°$ position in 4.6 s after the start of control, attaining an angular velocity of about 0°/s, and the pendulum stood still at the inverted position. This is a natural consequence because the inference rules were established for a 40-cm long pendulum.

Figure 8 shows a transient response of a pendulum 30 cm long. The pendulum was brought to its inverted position, showing a response similar to that obtained with the 40-cm pendulum, but the angle reached $-180°$ in 3.9 s. The overall controllable characteristics were similar to those of the 40-cm pendulum.

In a case where the length of pendulum was set at 20 cm as shown in Figure 9, a large velocity change was observed, and the angle became 180° in 6.2 s, attaining an angular velocity of about 0°/s. Although the pendulum reached

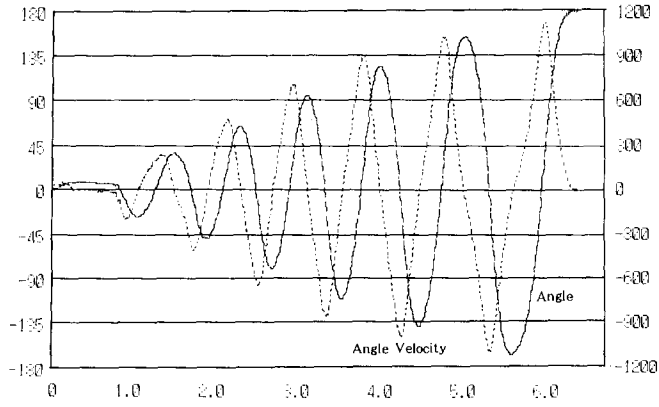


Figure 9. Angle and angle velocity of 20-cm long pendulum.

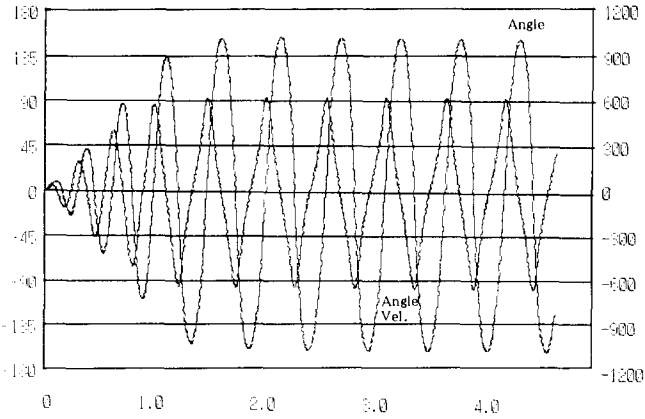


Figure 10. Angle and angle velocity of 50-cm long pendulum.

the inverted position and stayed there, the angular velocity was greater, and a longer lead-in time was required.

Figure 10 shows the transient response obtained with a 50-cm pendulum, which could not be brought to its inverted position. As seen in Figure 10, the pendulum angle could not be brought to its $\pm 180°$ position despite a longer lead-in time. The correlation between the dynamic characteristics of the pendulum and its length can be summarized as follows.

1. By applying NDF to a pendulum system whose length is varied from 40 to 20 cm, a stable operation to bring the pendulum to its inverted position became feasible despite the lead-in time required for its motion. In other words, the robustness of NDF is higher for shorter pendulum lengths.
2. For longer pendulums, however, the suppression of deviations of the control system cannot be attained, and this means that relearning or additional learning is necessary for NDF applied to a longer pendulum.

NEURAL NETWORK DRIVEN FUZZY REASONING WITH LEARNING FUNCTION (NDFL)

Neural network driven fuzzy reasoning is unable to alter its inference rule when the environment for constructing its inference rule is dynamically changing. Thus, we propose a new algorithm that can adapt to adjust its inference rules for a change of environment. We call this algorithm a neural network driven fuzzy reasoning with learning function (NDFL). NDFL can obtain the optimal coefficients of linear equations in consequent parts by using

the pattern search method. The pattern search method is divided into two parts: the exploratory movement and the pattern movement (Hooke and Jeeves [9]). The exploratory movement is a process to determine the direction for an optimal solution by calculating values of the evaluation function f. The pattern movement is a process to move the searching point to the direction needed for an optimal solution.

Now, we assume that the number of searching parameters is m, the coordinates terminated the $p-1$th exploratory movement at the kth pattern search are expressed by $W_p^k = (w_0^k, w_1^k, \ldots, w_p^k, \ldots, w_m^k)$, and a unit vector that the value of pth is one is expressed by d_p, where $p = 1, 2, \ldots, m$. Let's explain both movements, i.e., exploratory movement and pattern movement.

1) EXPLORATORY MOVEMENT

The following procedure is taken from $p = 1$. If

$$\int \left(W_p^k + \alpha_p^k d_p\right) < f\left(W_p^k\right), \ \alpha_p^k > 0 \tag{18}$$

is obtained, the pth exploratory movement is a success, and $W_{p+1}^k = W_p^k + \alpha_p^k d_p$ is established for the $p+1$th exploratory movement. If this fails, $\alpha_p^k = -\alpha_p^k$ is set, and the same operation is repeated. If both fail, $W_{p+1}^k = W_p^k$ is established. Howver, α_p^k shows a range of search.

2) PATTERN MOVEMENT

The coordinates terminated to mth exploratory movement at the kth pattern search are expressed by $W_B^k = (w_0^k, w_1^k, \ldots, w_p^k, \ldots, w_m^k)$. W_B^k is called a base point at the kth pattern search. Pattern movement means that the following shift from the base point W_B^k to W^k

$$W^k = W_B^k + \left(W_B^k - W_B^{k-1}\right) \tag{19}$$

A tuning of the operation of the consequent part of the inference rule performed for minimizing the evaluation function f by applying a pattern search method is illustrated here by employing the inference rule R_s,

$$\begin{aligned} R_s\text{: IF } \mathbf{x} &= (x_1, \ldots, x_n) \text{ is } A_s \\ \text{THEN } y_s^k &= w_{s0}^k + w_{s1}^k x_1 + \cdots + w_{sn}^k x_n, \\ s &= 1, 2, \ldots, r \end{aligned} \tag{20}$$

where A_s represents a fuzzy set in the input space of the antecedent, and $w_{s0}^k, \ldots, w_{sn}^k$ are coefficients for adjusting by the pattern search method.

Given the ith input data $X_i = (x_{i1}, \ldots, x_{in})$, the estimate y_i^* can be derived from the equation

$$y_i^* = \frac{\sum_{s=1}^{r} \mu_{As}(x_{i1}, \ldots, x_{in}) \times (w_{s0} + w_{s1}x_{i1} + \cdots + w_{sn}x_{in})}{\sum_{s=1}^{r} \mu_{As}(x_{i1}, \ldots, x_{in})}$$

$$i = 1, 2, \ldots, N \tag{21}$$

A fundamental structure of this method is shown in Figure 11. This shows that the form of fuzzy set A_s of the antecedent is determined by a backpropagation type of neural network model $\mathrm{NN}_{\mathrm{mem}}$, and a pattern search is made to improve the amount of operation $w_{s0}, \ldots, w_{sn}$ for the evaluation function f to attain an optimum value.

The steps of this algorithm are explained below.

STEP 1 Divide input-output data $x_i = (x_{i1}, \ldots, x_{in})$ into r classes of R_s, where $s = 1, 2, \ldots, r$. The input-output data for each R_s are expressed by $(y_i^s, x_i^s) = (y_i^s, x_{i1}^s, x_{i2}^s, \ldots, x_{in}^s)$, where $i = 1, 2, \ldots, N_s$, provided that N_s is a number of input-output data for each R_s. $\mathrm{NN}_{\mathrm{mem}}$ in Figure 11 conducts backpropagation learning in the same way as step 2 of the NDF algorithm. After conducting a backpropagation learning, the estimated value of the output represents the membership value of fuzzy sets A_s of each rule.

STEP 2 The initial values $w_{s0}^0, \ldots, w_{sn}^0$ of the coefficients of equations in the consequent parts required for the search are set.

STEP 3 The kth base point in the pattern search method is represented by $W_B^k = (w_0^k, w_1^k, \ldots, w_p^k, \ldots, w_m^k) = (w_{10}^k, w_{11}^k, \ldots, w_{1n}^k, w_{20}^k, \ldots, w_{2n}^k, \ldots, w_{r0}^k, \ldots, w_{rn}^k)$. The kth exploratory movement is taken from $W_1^k = W_B^k$.

STEP 4 If whole exploratory movements fail, an absolute value of α_p^k is reset to a smaller value. However, if

$$\max_p |\alpha_p^k| \leqq \varepsilon \tag{22}$$

is obtained, a procedure of algorithm is terminated, where ϵ is a threshold to stop the algorithm.

STEP 5 The pattern movement from the base point W_B^k to W^k is taken by using Eq. (19). After the pattern movement, the exploratory movement is taken

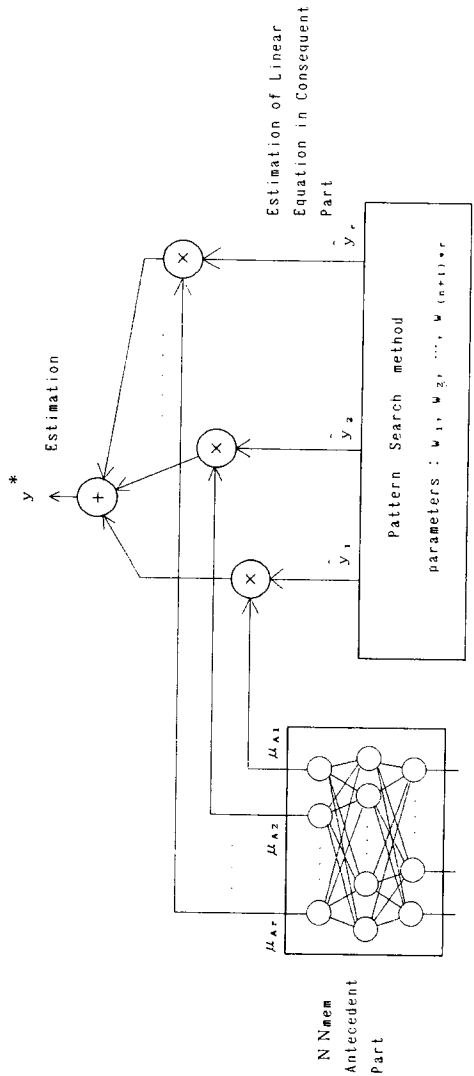


Figure 11. Block diagram of neural network driven fuzzy reasoning with learning function.

from $W_1^k = W^k$, and $W_B^{k+1} = W_{m+1}^k$ is set after the exploratory movement.

STEP 6 If

$$f(W_B^{k+1}) < f(W_B^k) \tag{23}$$

is obtained, go to step 5 as $k = k + 1$. If it fails, go to step 4.

AN APPLICATION TO SECONDARY FUNCTION IDENTIFICATION

To verify the effectiveness of the NDFL algorithm, an identification of a simple secondary function is conducted for a case where the input-output data are both the input and output of (x, y) shown in Figure 12.

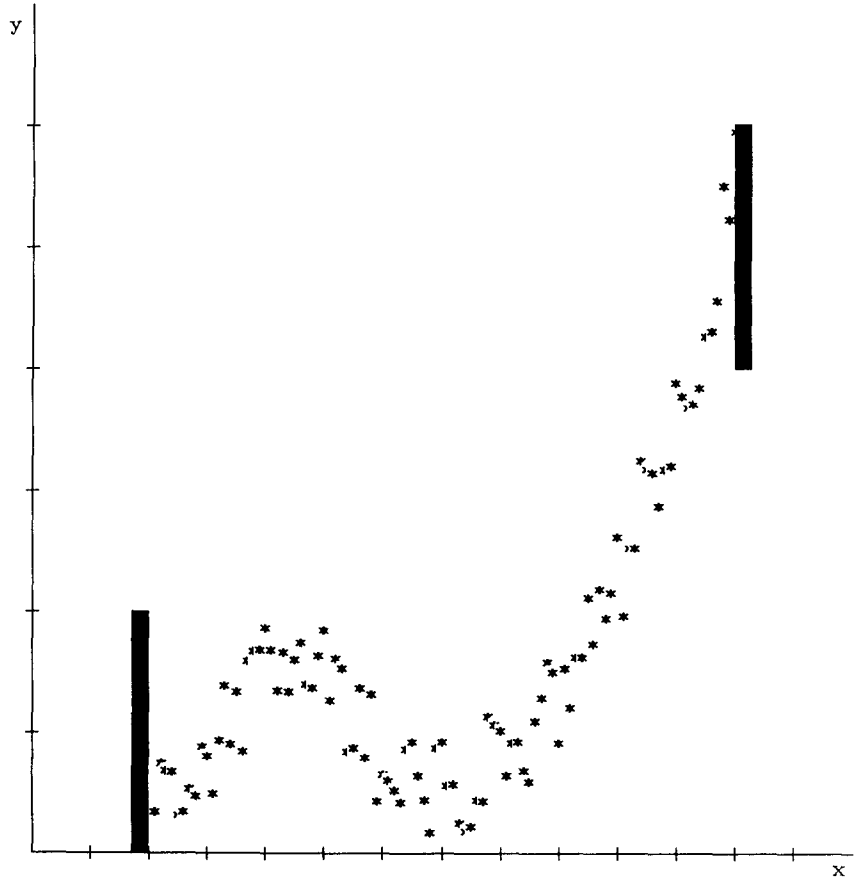


Figure 12. Input and output data for secondary function identification.

The following assumptions are made for NDFL algorithm application.

1. Based on the distributions of x-axis input data, four inference rules are set.
2. The inference rules are set as follows.

$$\text{Rs}: \text{IF } X \textit{ is } As \text{ THEN } y_s^k = w_{s0}^k + w_{s1}^k x$$

$$s = 1, 2, \ldots, 4 \tag{24}$$

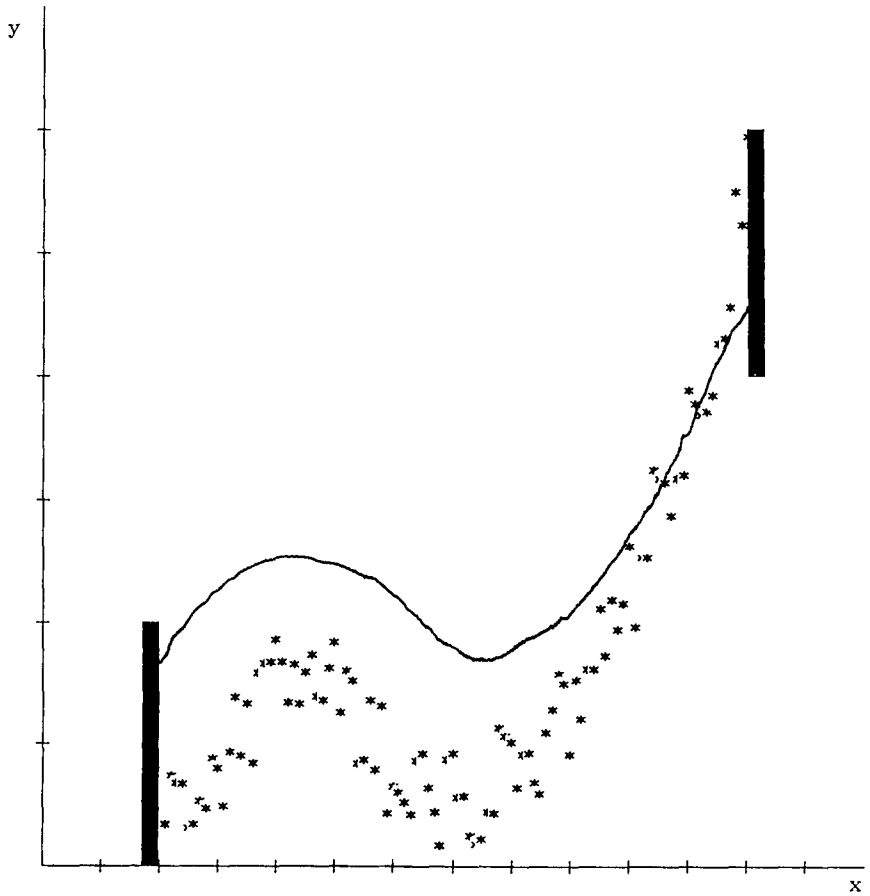


Figure 13. Teaching based on teacher's data.

3. The neural network for determining the antecedent part of the fuzzy inference rule is set as a three-layer [1 × 5 × 4], and the number of learnings is set at 550.
4. d_p^k is reduced from 1.0 by using $d_p^k = (1/2)^g$, $g = 0, 1, 2, \ldots$, and a threshold ε is set as $\varepsilon = 0.01$.
5. The evaluation function f is set as a sum of mean squares of errors between output value and estimated value.

As shown in Figure 13, a teaching based on the teacher's data (x_t^s, y_t^s), where $t = 1, 2, \ldots, 25$ is made first. The solid line in Figure 13 shows the teaching of the teacher's data. The initial values w_{s0}^0, w_{s1}^0 of the coefficients of equations in the consequent parts are derived by using the following set of equations.

$$w_{s1}^0 = \sum_{t=1}^{24} \left(\frac{y_{t+1}^2 - y_t^s}{x_{t+1}^2 - x_t^s} \right) \Big/ 24 \tag{25a}$$

$$w_{s1}^0 = \sum_{t=1}^{24} \left(y_t^s - \frac{y_{t+1}^s - y_t^s}{x_{t+1}^2 - x_t^s} x_t^s \right) \tag{25b}$$

$$s = 1, 2, \ldots, 4, \quad t = 1, 2, \ldots, 25$$

where the treacher's data (x_t^s, y_t^s) are obtained from the solid line in Figure 13. The search is commenced by assigning coefficients derived from the teacher's data to the initial values of search variables of the consequent of the fuzzy inference rule. Figure 14 shows an estimated curve obtained by this search, with reasonable estimated values representing the given input-output data.

The inference rules consequently obtained are

$$R_1: \text{IF } x \text{ is } A_1$$
$$\text{THEN } y_1 = 2.09 + 2.87x \tag{26a}$$

$$R_2: \text{IF } x \text{ is } A_2$$
$$\text{THEN } y_2 = 20.61 - 3.76x \tag{26b}$$

$$R_3: \text{IF } x \text{ is } A_3$$
$$\text{THEN } y_3 = -12.13 + 3.12x \tag{26c}$$

$$R_4: \text{IF } x \text{ is } A_4$$
$$\text{THEN } y_4 = -41.39 + 6.81x \tag{26d}$$

When part of the given input-output data are altered assuming a case where a change is made in the inference environment, NDFL is still applicable for constructing an inference rule. Figure 15 shows an estimated curve obtained by

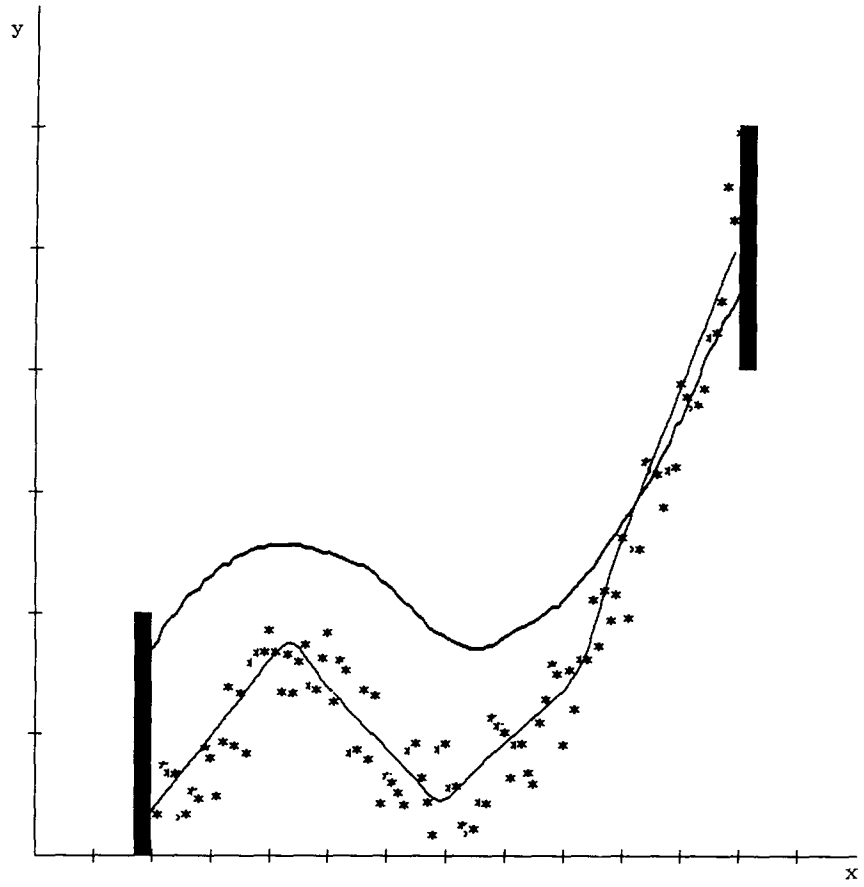


Figure 14. Curve estimated by NDFL.

NDFL after the input-output data are altered, and also shows a reasonable estimated curve as in the case illustrated in Figure 14.

As shown in the above, NDFL can be used to construct inference rules quickly and precisely even if the environment for constructing an inference rule is dynamically changing.

CONCLUSION

Whereas conventional fuzzy reasoning is associated with inherent tuning problems, NDF and NDFL are, when input-output variables are given, capable

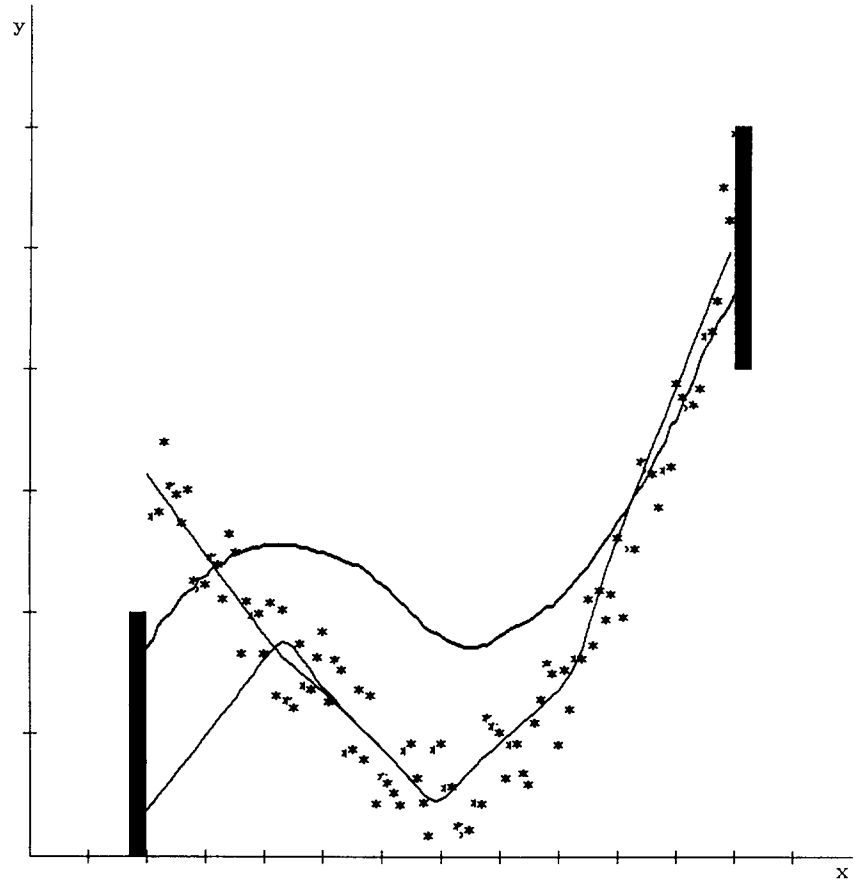


Figure 15. Curve estimated by NDFL after input-output data are altered.

of determining optimum inference rules and membership functions by utilizing the nonlinearity of a backpropagation neural network and its learning capabilities.

To verify the effectiveness of NDF, it was applied to an experimental pendulum system wherein the pendulum was brought to its inverted standstill position starting from a downward-hanging position. The length of the pendulum was altered to confirm its effects on the NDF control characteristics. Furthermore, NDFL was applied to identify a simple secondary function to verify its usefulness, and a satisfactory result was obtained.

Since this method is capable of deriving an inference rule by using a learning function of a backpropagation type of neural network, it was possible to introduce a learning function into fuzzy reasoning.

References

1. Hirota, K., Robotics and automation industrial applications in Japan, *3rd IFSA Congress*, Seattle, 229-230, 1989.

2. Lee, C. C., A self-learning rule-based controller with approximate reasoning, Memo No. UCB/ERL, M89/84, Univ. Berkeley, 1989.

3. Hayashi, I., Nomura, H., and Wakami, N., Artificial neural network driven fuzzy control and its application to the learning of inverted pendulum system, *3rd IFSA Congress*, Seattle, 610-613, 1989.

4. Takagi, H., and Hayashi, I., NN-driven fuzzy reasoning, *Int. J. Approx. Reasoning*, **5**(3), 191-212, 1991.

5. Anderson, J., and Rosenfield, E., *Neurocomputing*, MIT Press, Cambridge, Mass., 1988.

6. Tank, D., and Hopplield, J., Collective computations in neuronlike circuits, *Sci. Am.* 104-114, 1987.

7. Rumlhart, D. E., Hinton, G. E., and Williams, R. J., Learning representations by backpropagation errors, *Nature* **323**(9), 533-536, 1986.

8. Sugeno, M., and Kang, G. T., Structure identification of fuzzy model, *Fuzzy Sets Syst.* **28**(1), 15-33, 1988.

9. Hooke, R., and Jeeves, T. A., Direct search solution of numerical and statistical problems *J. ACM* **8**, 212-221, 1961.