A.1.5.4

MATHEMATICAL and INTELLIGENT models in system simulation
R. Hanus, P. Kool, S. Tzafestas (editors)
J.C. Baltzer AG, Scientific Publishing Co. © IMACS, 1991
pp. 287-292

INTELLIGENT SELF-ORGANIZING CONTROLLERS AND THEIR APPLICATION TO THE CONTROL OF DYNAMIC SYSTEMS+

M. De NEYER[1], D. STIPANICEV[2] and R. GOREZ[1]
[1]*Laboratoire d'Automatique, UCL, Louvain-la-Neuve, Belgium*
[2]*University of Split, Split, Yugoslavia*

ABSTRACT

A self-organizing controller is a fuzzy controller which learns its control strategy through experience. Two types of self-organizing controllers (SOC) are described and compared about their mechanisms and performances. Theoretical aspects are illustrated by some simulation runs of the control of an inverted pendulum. The concept of a self-tuning SOC is briefly introduced.

1. INTRODUCTION

Some complex industrial processes cannot be satisfactorily controlled by conventional automatic controllers. Those processes are characterized either by a non-linear or time-varying behavior, or by poor available measurement and as consequence by a model deficiency. On the other hand, there is an important amount of qualitative informations in those processes and they are often well controlled by human operators. An alternative approach to conventional control systems may be the investigation and the duplication of the control strategies which are employed by the human operator. The human operator develops a control strategy according his experience and intuition of the process behavior. Extraction of this knowledge is possible by discussion and observation. The obtained result is a qualitative strategy expressed linguistically as a set of imprecise and heuristic rules [1]. The fuzzy logic and the fuzzy set theory allows one to handle this qualitative information in a rigorous way [7] and then to reproduce the operator control strategy in the form of a control rule base. This rule base, integrated in a automatic control system , yields a fuzzy controller [1][2].

One of the difficulties in building such controllers is to obtain the set of control rules. There are several ways of acquiring control rules:
- manually by trial and error,
- by extracting the control strategy from an expert in the control of the particular process as proposed above,
- on the basis of a fuzzy model of the process to be controlled,
- by automatic learning.

This last possibility was investigated by Procyk in his self-organizing controller [3][4].

Section 2 of the paper describes two types of self-organizing controllers (SOC). The following sections compare their internal mechanisms and their performances first by logical analysis, second through some simulation runs. Section 5 proposes a way for self-tuning of the controllers. Conclusions are given in section 6.

2. DESCRIPTION

The self-organizing controller is a fuzzy controller which acquires control rules through experience in order to obtain a predetermined closed-loop control performance. Usually it is realized as a hierarchical rule-based controller with two layers : the first one is a rule-based fuzzy controller and the second a learning module which generates and modifies the rules [4][6].

2.1 Fuzzy controller

A fuzzy controller infers a fuzzy value for its output variable from fuzzy values of the input variables and from the rule base. The rule base is a collection of rules expressed as "If situation then action" statements. For example, a rule can be

If Error is A_i then control increment is B_i

+ Research supported by the Belgian National incentive-program for fundamental research in Artificial Intelligence, Prime Minister's Office - Science Policy Programming. The scientific responsibility is assumed by its authors.

where A_i and B_i are labels of fuzzy sets representing linguistic values like "small", "high"or fuzzy numerical values like "about 2". A rule is represented by a *fuzzy implication.*

Given an actual situation, "Error is A", each rule gives a contribution to the output value depending on the matching between the actual situation and the hypothetical situation carried by the rule. This contribution is calculated by means of a *inference law.* Then contributions from all the rules are aggregated in one global value for the output variable. Obviously if the fuzzy controller is applied to the control of a process in a non fuzzy environment, interfaces for converting fuzzy values to precise ones and vice-versa are needed, as shown in figure 1.

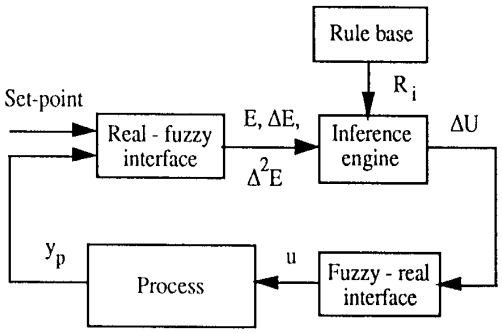

Figure 1.

In this study, we consider two types of controllers, both of them having 3 inputs variables and one output variable: E, the error, i.e the difference between the actual process output and the desired one, ΔE, the variation of E, Δ^2E, the variation of ΔE, are the input variables; ΔU, control increment, is the output variable of the fuzzy controller (see figure 1). The control rules have the following form:

If (E is A_i and ΔE is B_i and Δ^2E is C_i) then ΔU is D_i.

where A_i, B_i, C_i and D_i are labels of fuzzy sets representing values of the previous variables. These fuzzy sets are characterized by their membership functions μ defined on their respective "universes of discourse", X, Y, Z, V.

Controller n°1.

The i-th rule is represented by its fuzzy implication $R_i = A_i \times B_i \times C_i \times D_i$ defined by

$$\mu_{R_i}(x,y,z,v) = \min\{\mu_{A_i}(x), \mu_{B_i}(y), \mu_{C_i}(z), \mu_{D_i}(v)\},$$

with $x \in X$, $y \in Y$, $z \in Z$, $v \in V$. From an actual situation "E is A' and ΔE is B' and Δ^2E is C' " and from the i-th rule, one can infer a value of ΔU represented by the fuzzy set $D'_i = A' \bullet B' \bullet C' \bullet R_i$ defined by

$$\mu_{D'_i}(v) = \max_x \min\left\{\mu_{A'}(x), \max_y \min\{\mu_{B'}(y), \max_z \min\{\mu_{C'}(z), \mu_{R_i}(x,y,z,v)\}\}\right\},$$

symbol • denoting a compositional rule of inference. The contributions of all the rules are aggregated in one global value represented by the fuzzy set $D' = \bigcup_i D'_i$ such that

$$\mu_{D'}(v) = \max_i \mu_{D'_i}(v).$$

However one needs an interface between the real world and the fuzzy controller. First, the actual values of the input variables are precise, non-fuzzy: error is measured, the other variables are derived, all the variables being scaled and quantized as follows:

$$e_0 = Q(G_E * e(t)),$$
$$\Delta e_0 = Q(G_{\Delta E} * (e(t) - e(t-1))) = Q(G_{\Delta E} * \Delta e(t)),$$
$$\Delta^2 e_0 = Q(G_{\Delta^2 E} * (\Delta e(t) - \Delta e(t-1)) = Q(G_{\Delta^2 E} * \Delta^2 e(t)),$$

where $e(t) = r - y_p(t)$, G_E, $G_{\Delta E}$, $G_{\Delta 2E}$ are non-linear scale factors, Q(x) is the integer value which is the nearest to x; r is the set-point and y_p, the process output. This non-linear scaling tries to imitate the perception of a human being [6]. Our sensibility is relative; we feel more the difference between 0 and 1 than between 99 and 100.

In a fuzzy domain, a precise value is represented by a fuzzy singleton whose membership function is null everywhere except in one point (corresponding to the precise value where the grade of membership is equal to 1). Thus values of A', B' C' are described by fuzzy singletons such that $\mu_{A'}(e_0) = 1$, $\mu_{B'}(\Delta e_0) = 1$ and $\mu_{C'}(\Delta^2 e_0) = 1$. The use of non-fuzzy values simplifies the calculation of the inferences.

Conversely, the fuzzy value D' of the output variable is 'de-fuzzified' by the "mean of maxima" method:

$$\Delta u = R\left(\frac{\Delta u\max + \Delta u\min}{2}\right),$$

where Δumin and Δumax are respectively the lowest and highest elements in the universe V whose membership value $\mu_{D'}$ is maximum, R rounding off to the nearest integer value. Finally the actual control variable is updated, $u(t) = G_{\Delta U} * \Delta u(t) + u(t-1)$ and applied to the process, $G_{\Delta U}$ being a constant scale factor.

Controller n°2

Multiplication is used instead of the "minimum" operator for the fuzzy implication R_i

$$\mu_{R_i}(x,y,z,v) = \mu_{A_i}(x) * \mu_{B_i}(y) * \mu_{C_i}(z) * \mu_{D_i}(v),$$

with $x \in X$, $y \in Y$, $z \in Z$, $v \in V$. In the same way, the

inference law $D'_i = A' \circ B' \circ C' \circ R_i$ where A', B', C' are fuzzy sets representing the actual values of the inputs variables, is defined by

$$\mu_{D'_i}(v) = \max_x \{\mu_{A'}(x) * \max_y \{\mu_{B'}(y) * \max_z \{\mu_{C'}(z) * \mu_{R_i}(x,y,z,v)\}\}\}.$$

Now the aggregation $D' = \bigcup_i D'_i$ is defined by

$$\mu_{D'}(v) = \sum_i \mu_{D'_i}(v).$$

Here the values of $\mu_{D'}$ obtained through this aggregation procedure can be greater than 1. In order to allow their interpretation as membership function values, a scaling would be required; in fact, it is not necessary due to the 'de-fuzzification' procedure described below.

The real to fuzzy interface is similar to that described above except in one point, the variables being not quantized:

$e_0 = G_E * e(t)$,
$\Delta e_0 = G_{\Delta E} * (e(t) - e(t-1)) = G_{\Delta E} * \Delta e(t)$,
$\Delta^2 e_0 = G_{\Delta^2 E} * (\Delta e(t) - \Delta e(t-1)) = G_{\Delta^2 E} * \Delta^2 e(t)$,

In the fuzzy to real interface, the fuzzy value D' is 'de-fuzzified' by the "center of gravity" method:

$$\Delta u = \frac{\sum_v v * \mu_{D'}(v)}{\sum_v \mu_{D'}(v)}, \text{ where } v \in V.$$

The actual control variable u(t) is updated as explained for the first case.

2.2 Learning module

A learning mechanism allows the acquisition and the modification of the control rules in order to obtain a desired closed-loop response. To reach this goal, it is necessary to have a procedure for assessing the control performance and one for modifying the control strategy according to the value of a performance index (figure 2).

A performance index (PI) express how far is the process state from the desired one. In fact, this PI corresponds to the designer's idea about a minimum tolerable response [4] and is not specific to a given process. It can be described by rules as

If (E is "about 6" and ΔE is "about 0") then PI is "about 6",

or by a table (table 1). If PI = 0, the process behavior is as desired. The set of rules with PI = 0 defines a tolerance band in which the process output has to be contained (figure 3). If PI ≠ 0, the larger is its absolute value, the further is the process state from the desired one. Its value is a quantitative information related to the magnitude of the correction which is required at the process output.

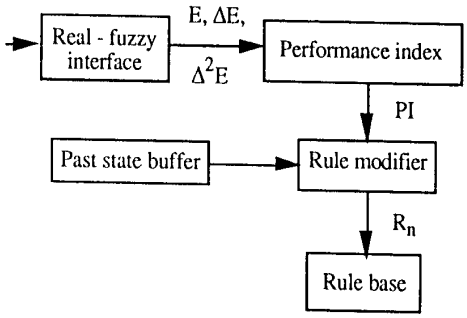

Figure 2.

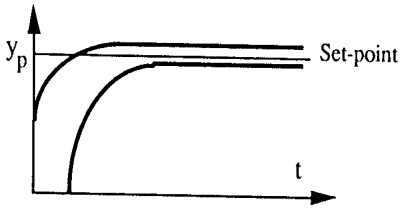

Figure 3.

If the actual values of the input variables are quantized, the performance index can be represented by a table the dimension of which is equal to the number of input variables; table 1 shows an example of performance index table for a two inputs controller [6].

e \ Δe	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-6	-6	-6	-6	-6	-6	-6	-6	-4	-3	-2	-1	0	0
-5	-6	-6	-6	-5	-5	-4	-4	-3	-2	-1	0	0	0
-4	-6	-6	-5	-5	-4	-3	-3	-2	-1	0	0	0	0
-3	-6	-5	-5	-4	-3	-2	-2	-1	0	0	0	0	1
-2	-6	-5	-4	-3	-2	-1	-1	0	0	0	0	1	2
-1	-5	-4	-3	-2	-1	-1	-1	0	0	0	1	2	3
0	-4	-3	-2	-1	-1	0	0	0	1	1	2	3	4
1	-3	-2	-1	0	0	0	1	1	1	2	3	4	5
2	-2	-1	0	0	0	0	1	1	2	3	4	5	6
3	-1	0	0	0	0	1	2	2	3	4	5	5	6
4	0	0	0	0	1	2	3	3	4	5	5	6	6
5	0	0	0	1	2	3	4	4	5	5	6	6	6
6	0	0	1	2	3	4	6	6	6	6	6	6	6

Table 1.

Rule modification is based on the assumption that the controller output at n samples in the past is responsible for the present state of the process and should be changed: n, called "delay in reward" is related to the process delay. Usually, for a single input-single output process, the

amplitude of the correction is taken equal to the value of the performance index.

To the two controllers considered above correspond two roughly similar learning parts. The performance index table is the same and is described by a three dimensional array (13x13x3) whose indices are quantized values of E, ΔE and $\Delta^2 E$.

Rule modification method is slightly different for each case. Let be R_i, the rule corresponding to the process state at time t-nT. For the first case, the rule modification is :

$$D_i(t) = F(\Delta u(t-nT) + r(t))$$

with D_i, the value of the control variable of i-th rule, Δu(t-nT), the controller output at n previous samples in the past, r(t), the correction equal to the performance index and T, the sampling interval; F(x) is a fuzzification procedure which consists in building a fuzzy set with a predefined spread around a precise element x. In the second case, several rules can be modified: R_i and rules corresponding to neighboring states at time t-nT

$$D_k(t) = F(\Delta u_k(t-nT) + w(k) * r(t))$$

with Δu_k(t-nT) is the central value of D_k(t-nT), w(k) is a weighting factor inversely proportional to the distance between the state at time t-nT and the state carried by rule R_k.

Thus, any SOC contains several parameters which must be determined:

- scaling factors: G_E, $G_{\Delta E}$, $G_{\Delta^2 E}$, $G_{\Delta U}$,
- performance index table,
- delay in reward n.

3. COMPARATIVE ANALYSIS

The basic characteristics of the two SOC above described will be compared and some conclusions about control performances and quality will be given. The two controllers have different grades of complexity, the first one using 'minimum' and 'maximum' operations with integer numbers and the second using multiplication and addition operations with real numbers. The inference law and fuzzy implication using product (case 2) instead of 'minimum' operator (case 1) give better discrimination between the rules which are relevant to a given input situation and those which are not [5].

The 'maximum' operator for aggregation merges the small contributions of rules in the big ones whereas the addition gives a weight to all the contributions according to the ratio of the values of membership functions [5].

The "mean of maxima" technique used for "de-fuzzification "in the first case gives discontinuous and abrupt changes in the controller output. This method is such that at most 2 groups of rules (often 1 rule) contribute to the precise value of the controller output; one group for Δumin and one for Δumax. On the other hand, the "center of gravity" method gives smoother transients, because several rules contribute to the precise value of the output [5]. That explains why in the learning part for the second case, it is possible to modify more than one rule.

Quantization of the values of the inputs variables brings (among other things) dead-zones around zero, the controller output being the same (normally zero) whatever the actual values of the variables may be near to zero. These dead-zones are inversely proportional to the scaling factors. (Increasing these factors gives a larger rise-time, oscillations or difficulties in the convergence of the learning procedure [3]). The final value of the error depends on the $G_{\Delta U}$ scaling factor, its value defining the smallest value of the control increment. In order to obtain small steady state errors one can decrease $G_{\Delta U}$, but small $G_{\Delta U}$ restricts the output controller range so that the rise-time decreases. If continuous values are used for the variables in conjunction with the "center of gravity" method, dead-zones vanish; finer control is possible, more or less independently of $G_{\Delta U}$. The constraint on $G_{\Delta U}$ does no longer exist since continuous values are possible for Δu. Note that when the values of the input variables are quantized, one can simply represent the control strategy by a decision table which gives the precise value of output variable Δu with respect to the quantized values of the inputs variables. The use of this table decreases the computation time and is interesting for real time control.

The learning mechanism in the first case combined with the "mean of maxima" method gives after learning a control strategy where for a given controller output, often one rule only contributes; in fact nearly for each quantized process state visited during the learning phase, the learning mechanism creates a rule. If a rule was created for each visited process state, the control algorithm would be roughly simple; the precise controller output would be equal to the rule action corresponding to a given process state. In such a case, any control rule can be restricted to the following form (with one input variable):

If (quantized value of the input variable is equal to x) then Δu is equal to v, where $x \in X, v \in V$.

The following general facts have been observed during simulation runs on several processes:

1) on average, the number of generated rules is greater in the first controller. It can be explained by the one-to-one relationship between the process states and precise controller outputs (see above).
2) Convergence is slower for the first SOC: the number of

iterations for obtaining convergence of the control strategy is greater (the control strategy is said to have converged, if there are no longer changes in the rule base).

3) The ranges of the parameters values which lead to good performances are wider for the second type of SOC.

4 SIMULATION RESULTS

In this section, some points described above are illustrated by simulation of the control of a non-linear process: an inverted pendulum. This process can be viewed as a basic model of a manipulator link, it is unstable and non-linear. The process is described by the following equation:

$$ml^2\ddot{\theta} = mgl \sin\theta - k\dot{\theta} + C$$

where θ, angular position of the pendulum (with respect to a vertical line),
C, input torque,
m = 1 [kg], mass,
2l = 2 [m], length,
k = 1, viscous coefficient,
g = 9.81, gravitational constant.

Fuzzy sets are defined on the interval [-6, 6] for E, ΔE, ΔU and on [-1, 1] for Δ^2E; their membership functions μ have a same triangular shape centered on an integer value x_0 between -6 and 6 (figure 4) or between -1 and 1. There are 13 possible values for A_i, B_i and D_i and only 3 for C_i.

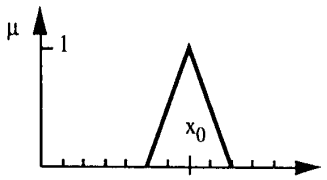

Figure 4.

A scaled value x_g is obtained as follows:
$x_g = G_X * x(t) = f(G_{X0} * x(t))$ where G_{X0} is a constant gain and f, a piecewise linear function (figure 5).

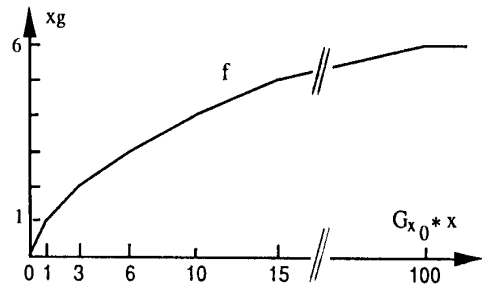

Figure 5.

The process and the control system have been simulated using C language on an IBM PS2.

The procedure for choosing the parameters of the SOC will not be described in details (see next section). They were chosen in view of a rise-time around 2 seconds, an overshoot lower than 1% and an steady state error smaller than 1% (table 2).

	G_E	$G_{\Delta E}$	$G_{\Delta^2 E}$	$G_{\Delta U}$	n
SOC 1	100	2000	1900	0.1	2
SOC 2	100	2000	1500	0.15	2

Table 2

Performances ratios are shown in table 3 where 'aae' is the average absolute error in steady state and 'iae' = integral of the absolute error with respect to time. Both of the controllers meet the specifications but in this case the second one has clearly better performances. Figures 6 (SOC 1) and 7 (SOC 2) show the actual control variable u and the process output y_p (set-point = 0.5; initial y_p = 0). The control signal u varies very abruptly for the first SOC (figure 6). The number of rules generated by the SOC is relatively high (165 and 108) because many states are visited during the learning due to the facts that the controllers start without any rules, the process controlled is unstable, and the tolerance band is narrow (high value of gains). However the number of rules used for control is smaller (28 and 44).

	SOC 1	SOC 2
rise-time	1.95	1.9
overshoot	0.87%	0.08%
5% settling time	3.1	2.75
1% settling time	4.45	5.95
iae	23.79	14.49
aae	0.24%	0.0004%
number of rules	165	108
convergence after m iterations	14	5

Table 3

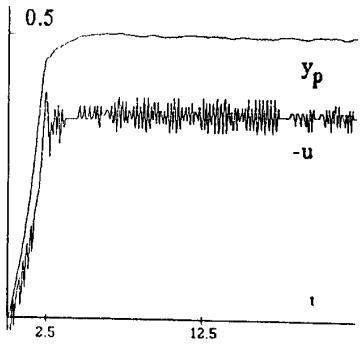

Figure 6.

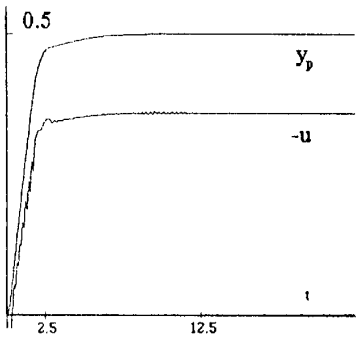

Figure 7.

Simulation runs were done with different values of the process parameters l, m and k using the two controller rule bases obtained through learning with the controller parameters tuned as above. Steady state performances are conserved with process parameter variations up to 40 or 50% for both controllers even further in some cases for the first one. Stability of the closed-loop is obtained for wider process parameter ranges with the first controller.

5. SELF-TUNING SOC

In the SOC here described, the choice of the parameter values is done manually, being guided by heuristic rules such that: increasing G_E yields smaller steady state error and lower rise-time ,but it may result in overshoot and oscillations in steady state; increasing $G_{\Delta E}$ increases the rise-time and decreases the overshoot and the oscillation amplitudes in steady state; high values for G_E or/and $G_{\Delta E}$, gives a very sensitive performance index and a too narrow tolerance band so that convergence of the control strategy is difficult and many of rules are generated; the steady state error is proportional to $\frac{1}{2G_E}$ (for the first SOC), etc...

Tuning could be done through an automatic procedure implemented in a self-tuning SOC. The latter is based on the following ideas: the knowledge about the influences of some parameters on the control and learning performances is collected in a rule base. Those rules would express the changes that must be brought to the parameters with respect to performance ratios (steady state error, rise-time, convergence...). In fact this rule base will form a third control layer, which will allow the automatic adjustment of the parameters for good performances.

6. CONCLUSIONS

Two types of self-organizing controllers have been described and compared, the first one, simpler in its mechanisms, uses quantized variables and the "mean of maxima" method, the second one uses continuous variables and the "center of gravity" method.

They can have similar control performances but the variations of the actual control signal are smoother in the second case, and in steady state, the control is better with the second SOC often resulting in a smaller error and it is more oscillating in the first case. The first SOC seems to present a better robustness to the process parameter variations.

On the other hand, the learning performances are different: more rules are generated, slower convergence is achieved and the parameter ranges which allows convergence are smaller for the first controller.

To the simplicity and the less good performances of the first SOC correspond a greater complexity and better performances of the second.

REFERENCES

[1] King P.J., Mamdani E.H., The application of fuzzy control systems to industrial processes, Automatica, Pergamon Press, Great Britain, 1977, Vol.13, pp.235-242.

[2] Mamdani E.H., Application of fuzzy logic to approximate reasoning using linguistic synthesis, IEEE Trans.Computers, 1977, vol.c26, n°12, pp.1182-1191.

[3] Procyk T.J., A self-organising controller for dynamic processes, Research report n°6a, Queen Mary college, London, 1977.

[4] Procyk T.J., Mamdani E.H. A linguistic self-organising controller, Automatica, Pergamon Press, Great Britain, 1979, Vol.15, pp.15-30.

[5] Sugiyama K., Analysis and synthesis of the rule-based self-organising controller, PhD thesis, Queen Mary College, London, 1986.

[6] Sugiyama K., Rule-based self-organising controller, Fuzzy computing: Theory of hardware and applications - Ed: M.M.Gupta && T.Yamakawa, Elsevier Science Publishers, 1988, pp.341-353.

[7] Zadeh L., A Theory of Approximate Reasoning, Memorandum n° UCB/ERL M77/58, University of California, 1977.