Carolina State University

Telerobotic Control of a Mobile Coordinated Robotic Server

NAG-1-1283-2

Box 7921
Raleigh, NC 27695-7921
(919) 515-5931

INTERIM REPORT

EXECUTIVE SUMMARY

This interim report continues with the research effort on advanced adaptive controls for space robotic systems. In particular, previous results developed by the principle investigator and his research team centered around fuzzy logic control (FLC) in which the lack of knowledge of the robotic system as well as the uncertainties of the environment are compensated for by a rule base structure which interacts with varying degrees of belief of control action using system measurements. An on-line adaptive algorithm was developed using a single parameter tuning scheme. In the effort presented in this report, the methodology is further developed to include on-line scaling factor tuning and self-learning control as well as extended to the multi-input, multi-output (MIMO) case. Classical fuzzy logic control requires tuning input scale factors off-line through trial and error techniques. This is time-consuming and can not adapt to new changes in the process. The new adaptive FLC includes a self-tuning scheme for choosing the scaling factors on-line. Further the rule base in classical FLC is usually produced by soliciting knowledge from human operators as to what is good control action for given circumstances. This usually requires full knowledge and experience of the process and operating conditions, which limits applicability. A self-learning scheme is developed which adaptively forms the rule base with very limited knowledge of the process. Finally, a MIMO method is presented employing optimization techniques. This is required for application to space robotics in which several degrees-of-freedom links are commonly used. Simulation examples are presented for terminal control - typical of robotic problems in which a desired terminal point is to be reached for each link. Future activities will be to implement the MIMO adaptive FLC on an INTEL microcontroller-based circuit and to test the algorithm on a robotic system at the Mars Mission Research Center at North Carolina State University.

1. Introduction

A brief summary of the classical fuzzy logic control structure is presented here for completeness.

The structure of a fuzzy logic controller is shown in Figure 1. It is composed of fuzzifier, rule base, and defuzzifier. The computation of the control action consists of the following stages:

1) Compute current error(E) and its rate of change(CE).
2) Convert numerical E and CE into fuzzy E and CE.
3) Evaluate the control rules using the fuzzy logic operations.
4) Compute the deterministic input required to control the process.

The fuzzifier includes scaling part and membership function part as shown in Figure 2.

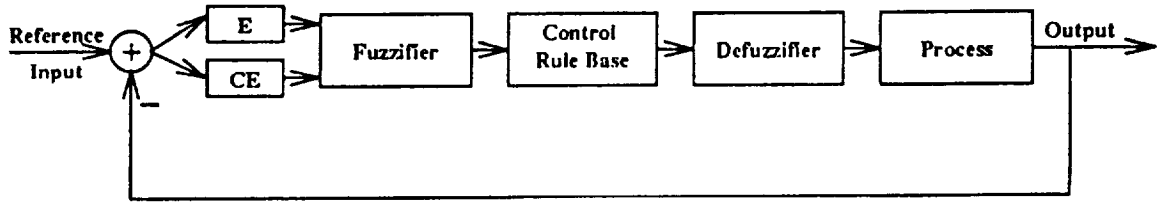

Figure 1: Structure of a classical fuzzy logic controller

The scaling factors can be nonlinear; also the membership functions can have other shapes, like bell, trapezoidal, sinusoidal shapes and etc. The fuzzifier converts numerical E and CE, such as 100.01, -0.93, etc., into fuzzy E and CE, such as SN (small negative), ZE (zero), MP (medium positive), etc., with grades of membership μ(E) and μ(CE) from 0 to 1.

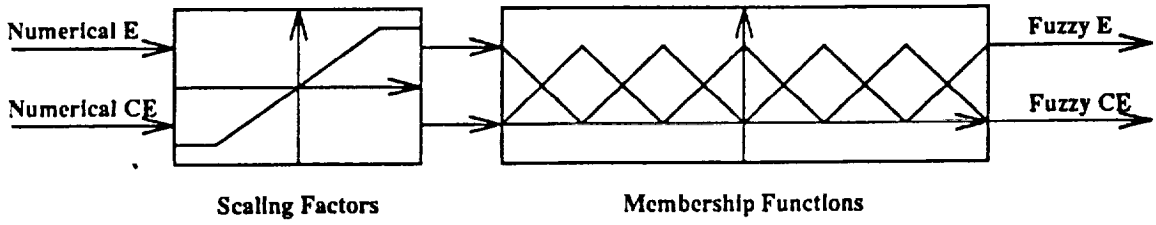

Figure 2: Structure of the fuzzifier

The rule base contains the control rules which are *if...then...* statements. Figure 3 gives the structure. These rules are evaluated to determined the fuzzy process control input. The control rules can also be represented in the three dimensional space.

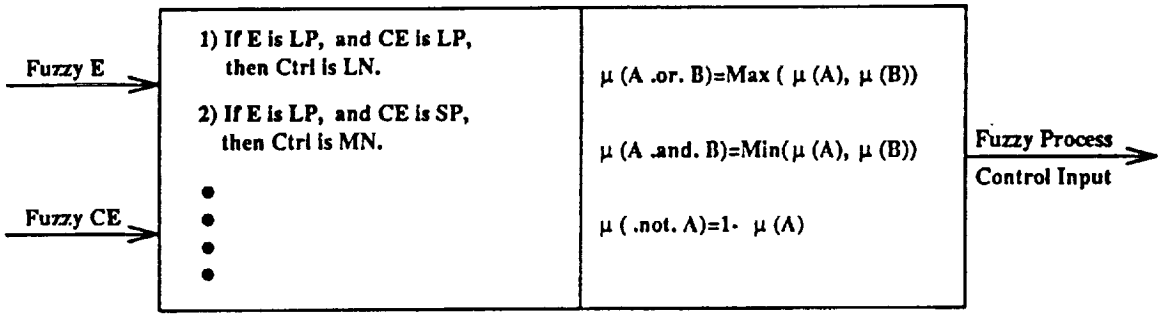

Figure 3: Structure of the rulebase

The control rules can also represented as a surface in the three dimensional space shown in Figure 4.

The defuzzifier is the inverse of the fuzzifier. It converts fuzzy process control inputs obtained through rule evaluation into numerical deterministic process control inputs. Many algorithms can be used here; the center of gravity method is the most popular one.

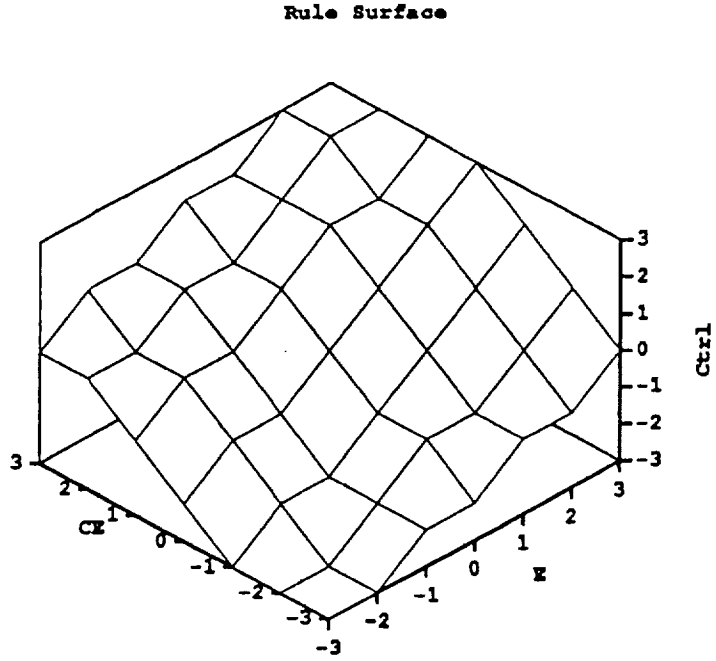

Figure 4: Control rule surface with (-3:LN) (-2:MN) (-1:SN) (0:ZE) (1:SP) (2:MP) (3:LP)

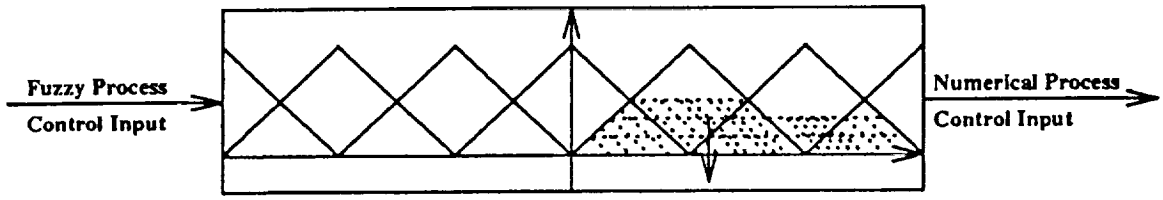

Figure 5: Structure of the defuzzifier

2. The Self-Tuning Scheme

From the structure of a fuzzy logic, one can find that the scaling factors of fuzzifier affect the performance of the controller to a large extent. In general, if the scaling factors are too small, the control response would be slow. If the scaling factors are too large, the control response would have large overshoot. There are two common ways to decide the scaling factors, both by trial and error:

1) Compare ideal step response with actual step response, adjust the scaling factors till the ideal step response and the actual response are almost identical. This is done off-line after each simulation or experiment. It takes a lot of time.

2) Check the error phase portraying factors till the portrait smoothly and quickly converges to zero. This is also done off-line.

Here, a self-tuning scaling scheme is given based on error phase portrait. With this scheme, the tedious off-line scaling adjustments can be avoided and since the self-tuning scheme is on-line, it can adapt to the new changes of the process.

The general idea behind the scheme is to adjust the scaling factors so that the augmented error which is the weighted distance in the error phase portrait decreases.

The scheme can be illustrated through Figure 6. The variables t_n and t_{n+1} are sampling times, L_n and L_{n+1} are weighed distances to the origin (or weighted augmented errors),

$$L_i = \sqrt{w_e E_i^2 + w_{ce} CE_i^2}$$

Further w_e and w_{ce} are the weights.

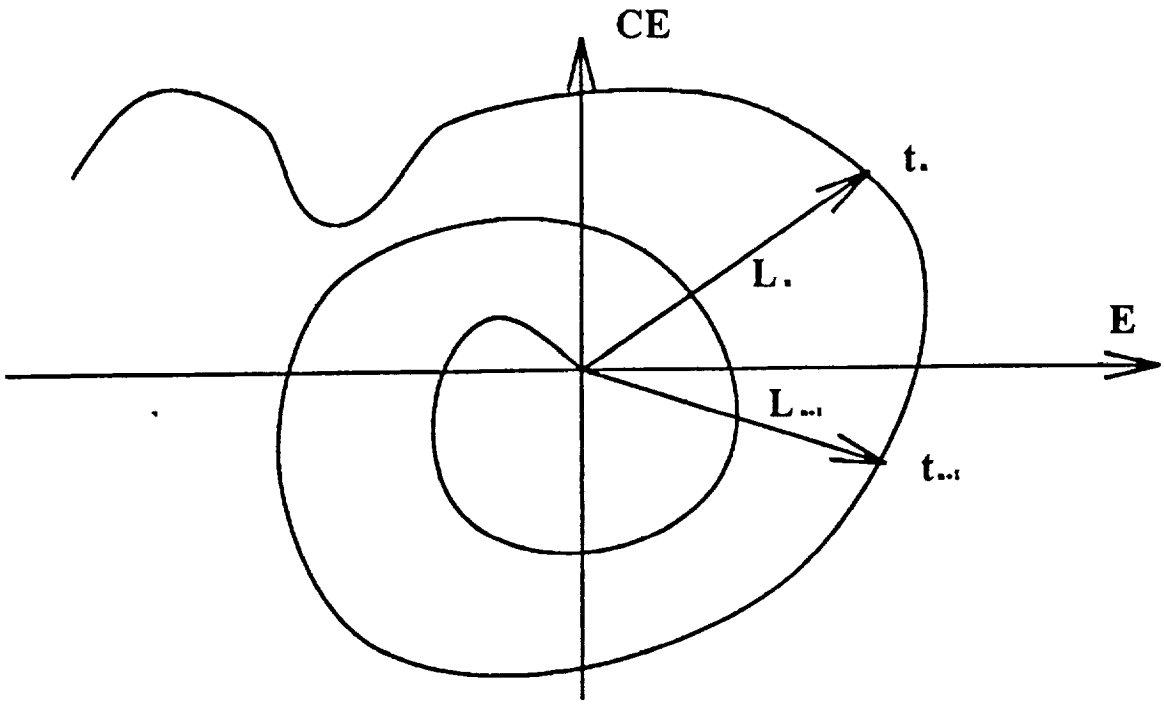

Figure 6: Phase portrait

The algorithm for the error is

If

$$L_{n+1} > \delta L_n \quad \text{and} \quad |E_{n+1}| > \delta |E_n|$$

Then

$$K^E_{n+1} = K^E_n - \text{sign}(E_n \times E_{n+1}) \times \lambda \times \min\left(1, \frac{2 \times (|E_{n+1}| - |E_n|) \times K^E_n}{LU\, D^E}\right)$$

Else

$$K^E_{n+1} = K^E_n$$

The algorithm for the rate of error change is

If

$$L_{n+1} > \delta L_n \quad \text{and} \quad |CE_{n+1}| > \delta |CE_n|$$

Then

$$K^{CE}_{n+1} = K^{CE}_n - \text{sign}\,(CE_n \times CE_{n+1}) \times \lambda \times \min\left(1, \frac{2 \times (|CE_{n+1}| - |CE_n|) \times K^{CE}_n}{LU\, D^{CE}}\right)$$

Else

$$K_{n+1}^{CE} = K_n^{CE}$$

where

K_i^E is the scaling factor for the error at time t_i.

K_i^{CE} is the scaling factor for the rate of error change at time t_i.

LU D^E is the length of universe of discourse for the error.

LU D^{CE} is the length of universe of discourse for the rate of error change.

δ and λ are convergence coefficients and $1 > \delta, \lambda > 0$.

The sign(x) function is defined as

$$\text{sign}(x) = \begin{cases} 1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x = 0 \\ -1 & \text{if} \quad x < 0 \end{cases}$$

Simulation results for a low-order linear but unknown system are compared with self-tuning and without self-tuning in Figure 7 and Figure 8.

3. Self-Learning Scheme

The control rules of a classical fuzzy logic controller are developed based on the experience and knowledge of an operator. These rules are unchanged during the process. But in some cases, full knowledge of the process is not available. Even with enough knowledge of the process, one still hopes that the controller can adapt itself to new operating conditions; so a controller with learning ability would be desirable.

A self-learning fuzzy logic controller consists of two parts, (Figure 9): one is the identification part which develops the control rules based on the control performance; the other part is control part which computes the control input for the process.

A control rule can be written as

If X^i, *And* Y^i, *Then* Z^i

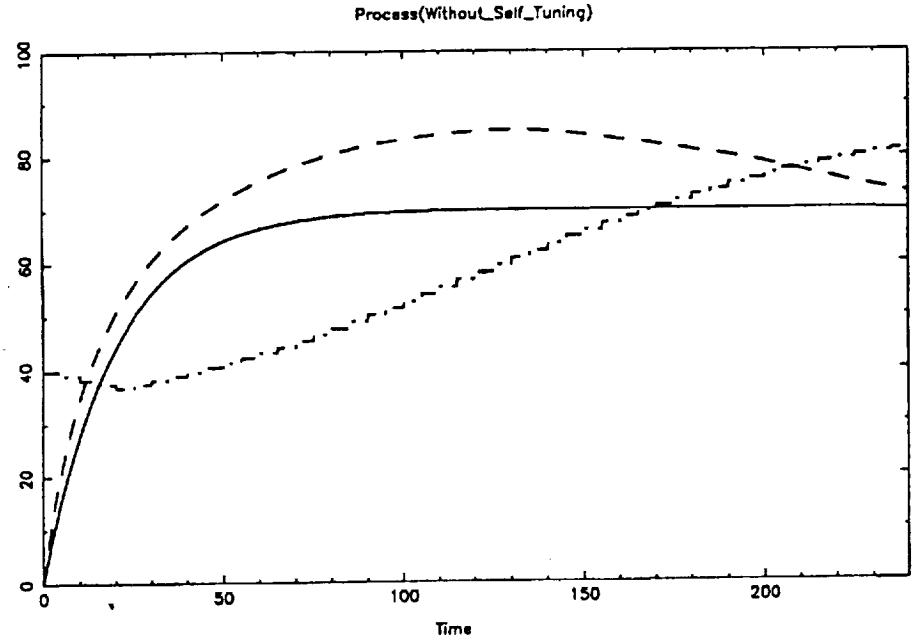

Figure 7: Process response without self-tuning (solid line : desired response) (dashed line : actual response) (dashed-point line : control signal)

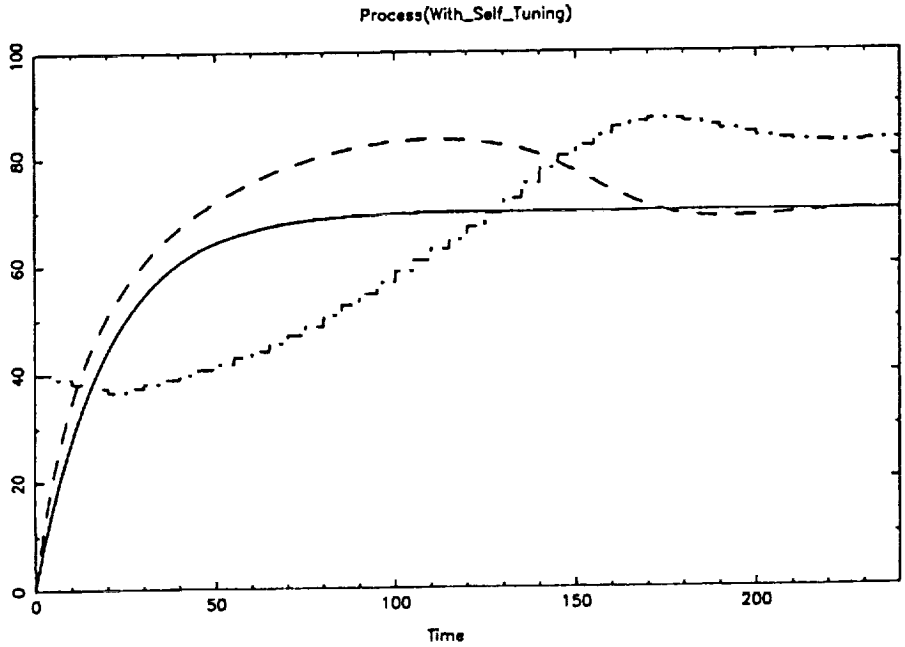

Figure 8: Process response with self-tuning (solid line : desired response) (dashed line : actual response) (dashed-point line : control signal)

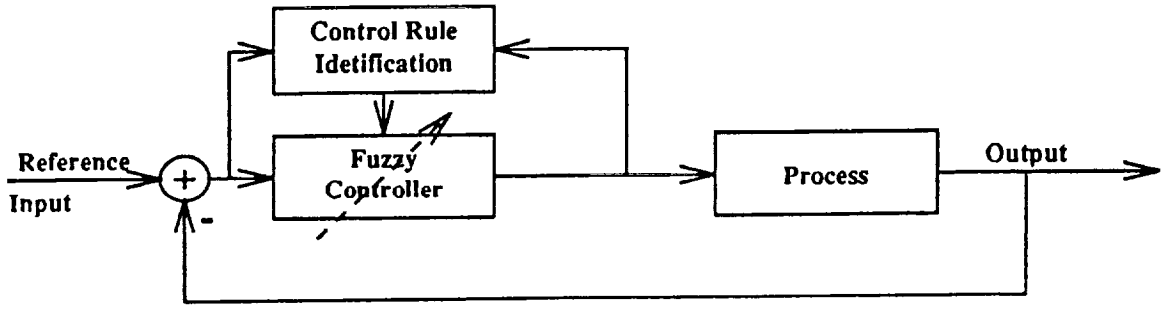

Figure 9: Structure of a self-learning fuzzy logic controller

or

$$X^i \cap Y^i \Rightarrow Z^i$$

X^i and Y^i are facts, Z^i is the conclusion which needs to be identified. The overall control rules used at certain sampling time is

$$\bigcup_i (X^i \cap Y^i) \Rightarrow Z$$

The Z here is time varying.

Define control performance index Θ as

$$\Theta = f(\rho, E, CE)$$

where ρ is learning convergence coefficient.

The learning scheme is

$$Z_{n+1} = (Z_n \times D) \oplus \{\Theta \otimes \bigcup_i (X^i \cap Y^i)\}$$

where D is a forgetting factor.

So the control rules become

$$\bigcup_i (X^i \cap Y^i) \Rightarrow Z_n \rightarrow \bigcup_i (X^i \cap Y^i) \Rightarrow Z_{n+1}$$

In order to illustrate the scalar adaptive FLC method, a simple low-order system with unknown parameters is to be controlled. As is typical with robotic systems, a desired terminal point is to be achieved. Figures 10-21 illustrate the iteration (learning) process. In particular, Figure 10 shows the initial control surface (essentially flat with no information) while Figure 11

shows what the projected results would be with the desired response also shown. As the iteration progresses with the applied control at each sample, the control surface improves resulting in the tenth run (Figures 18 and 19) where the desired response is achieved. The fifteenth run validates that this is the preferred control strategy and that the fuzzy controller adapts and learns about the system as it controls. Hence the controller works quite well in this example. The approach has also been applied to other processes with similar promising results.

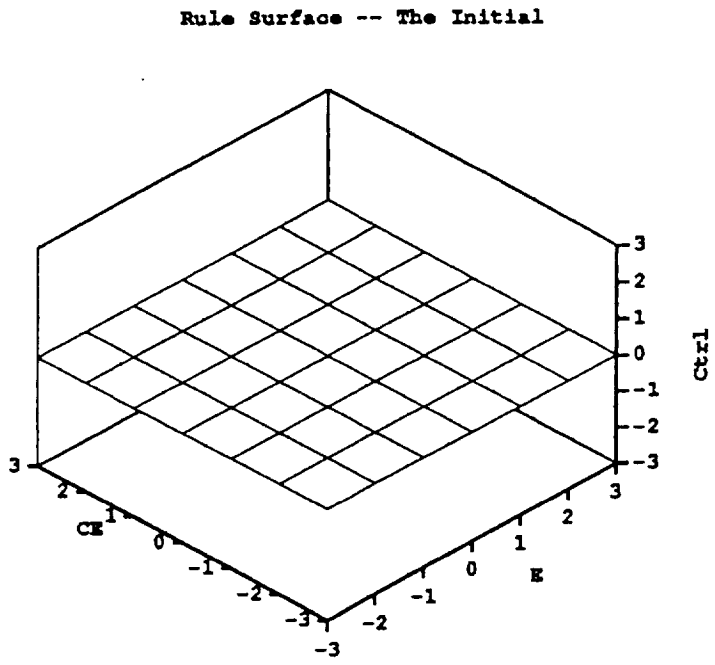

Figure 10: The initial control rule surface with (-3:LN) (-2:MN) (-1:SN) (0:ZE) (1:SP) (2:MP) (3:LP)

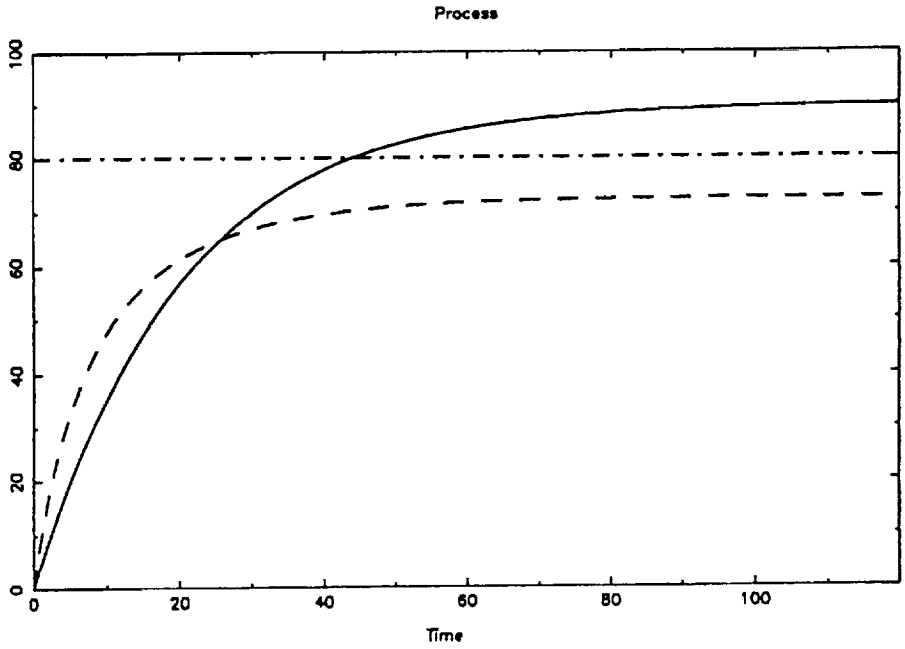

Figure 11: Process response of the initial run (solid line : desired response) (dashed line: actual response) (dashed-point line : control)

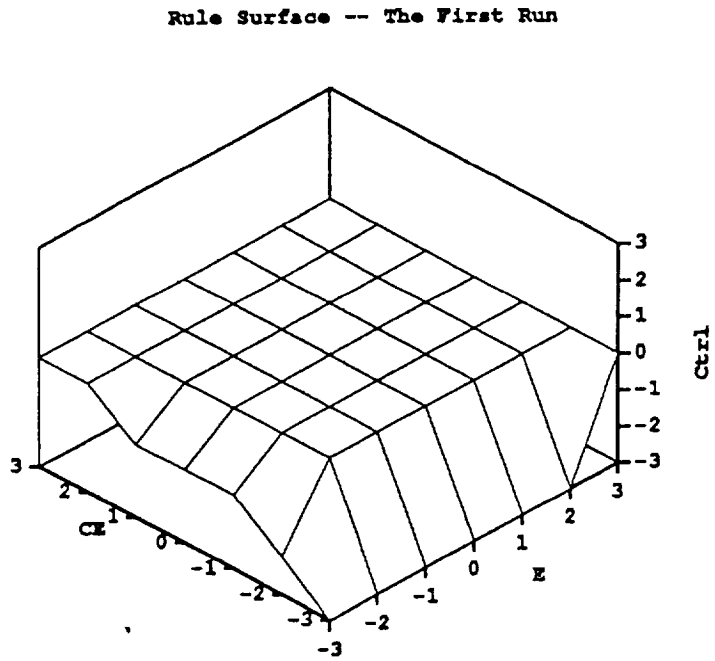

Figure 12: The first control rule surface with (-3:LN) (-2:MN) (-1:SN) (0:ZE) (1:SP) (2:MP) (3:LP)

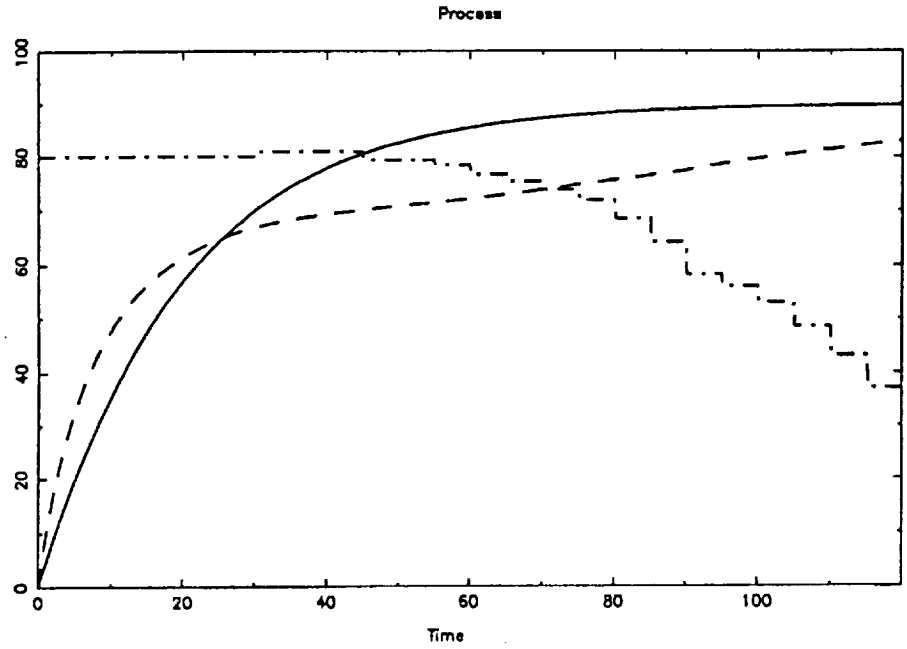

Figure 13: Process response of the first run (solid line : desired response) (dashed line : actual response) (dashed-point line : control)

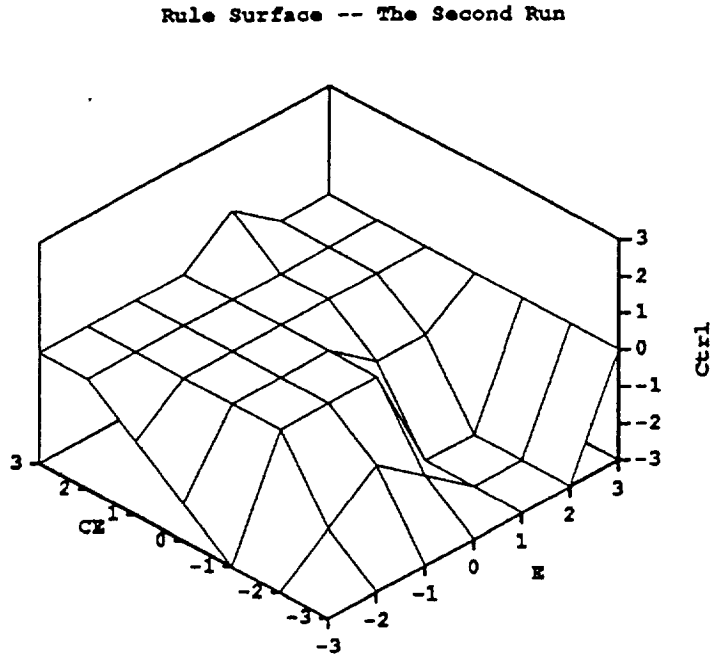

Figure 14: The second control rule surface with (-3:LN) (-2:MN) (-1:SN) (0:ZE) (1:SP) (2:MP) (3:LP)

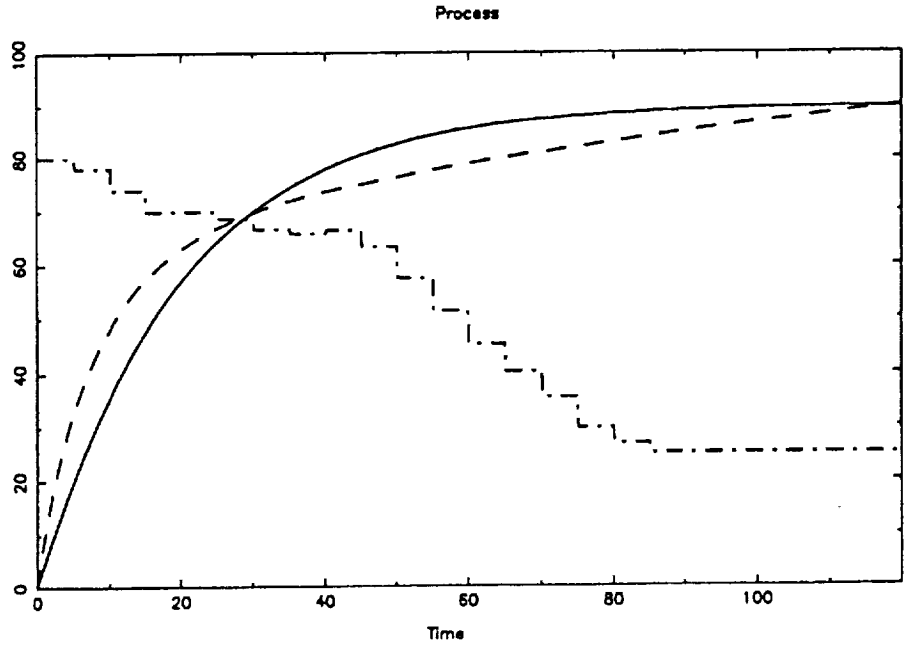

Figure 15: Process response of the second run (solid line : desired response) (dashed line : actual response) (dashed-point line : control)

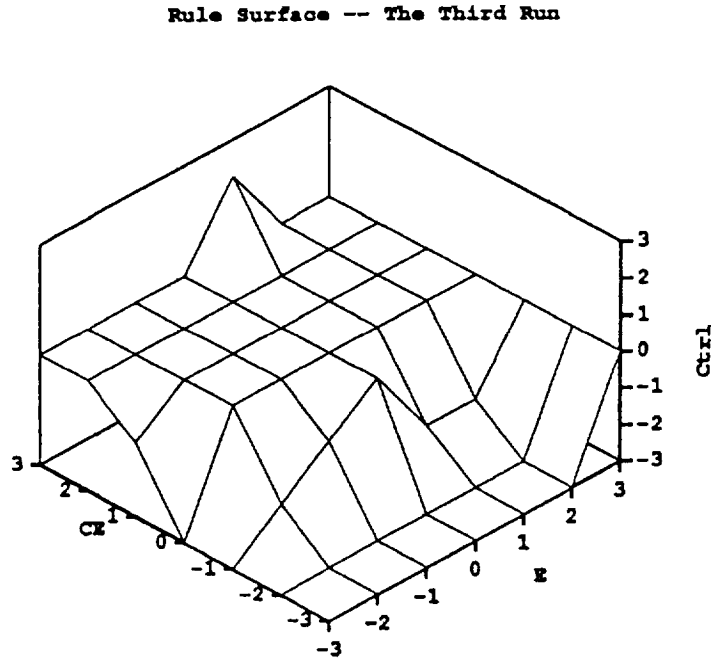

Figure 16: The third control rule surface with (-3:LN) (-2:MN) (-1:SN) (0:ZE) (1:SP) (2:MP) (3:LP)

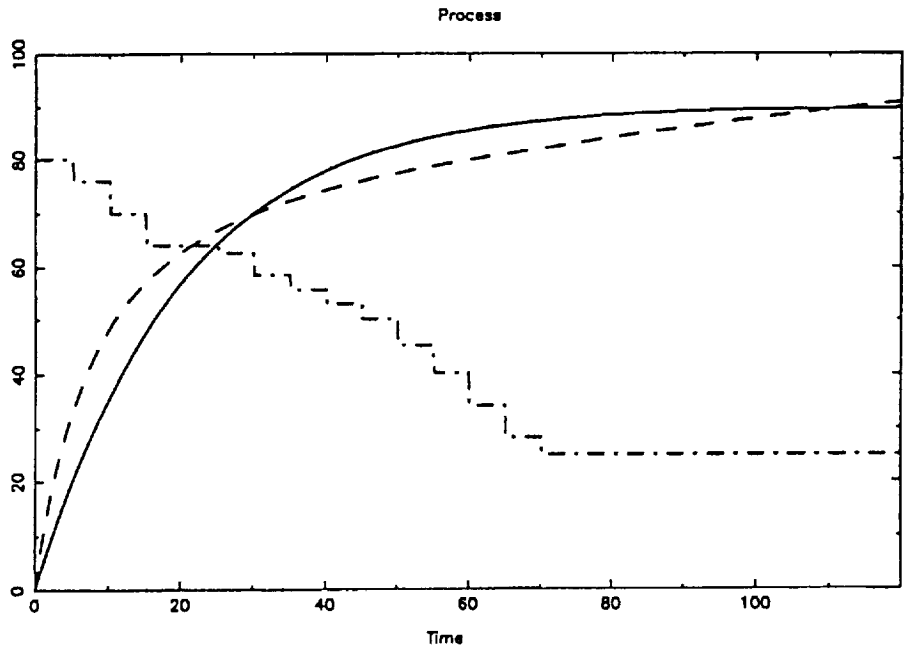

Figure 17: Process response of the third run (solid line : desired response) (dashed line : actual response) (dashed-point line : control)

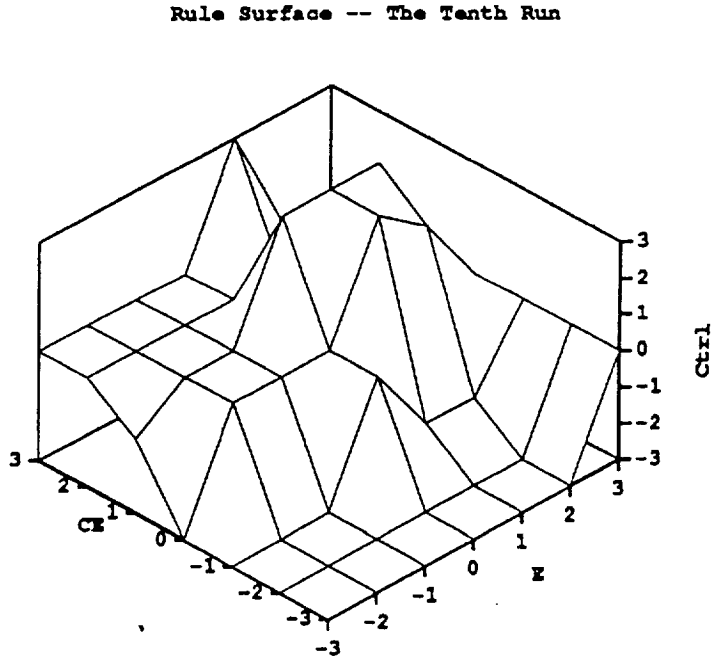

Figure 18: The tenth control rule surface with (-3:LN) (-2:MN) (-1:SN) (0:ZE) (1:SP) (2:MP) (3:LP)

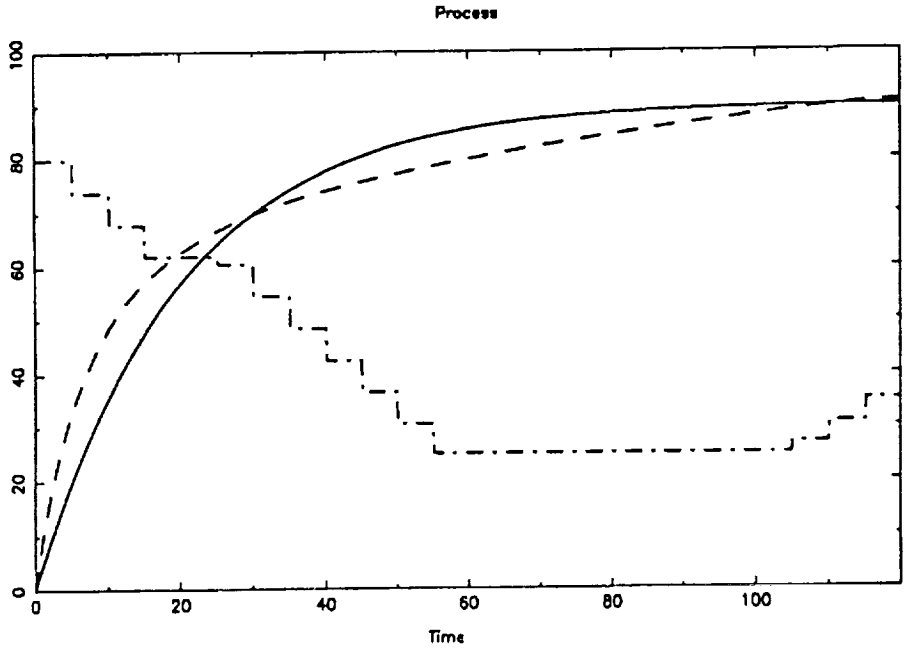

Figure 19: Process response of the tenth run (solid line : desired response) (dashed line : actual response) (dashed-point line : control)

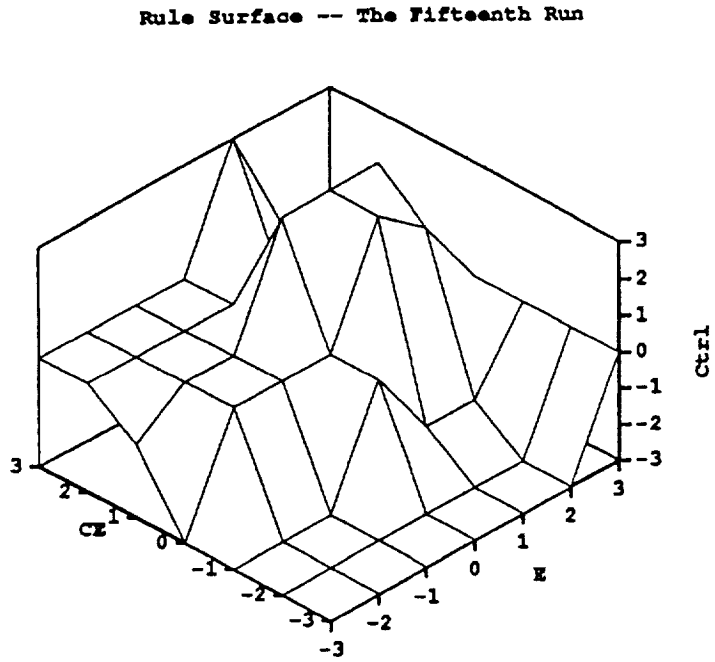

Figure 20: The fifteenth control rule surface with (-3:LN) (-2:MN) (-1:SN) (0:ZE) (1:SP) (2:MP) (3:LP)

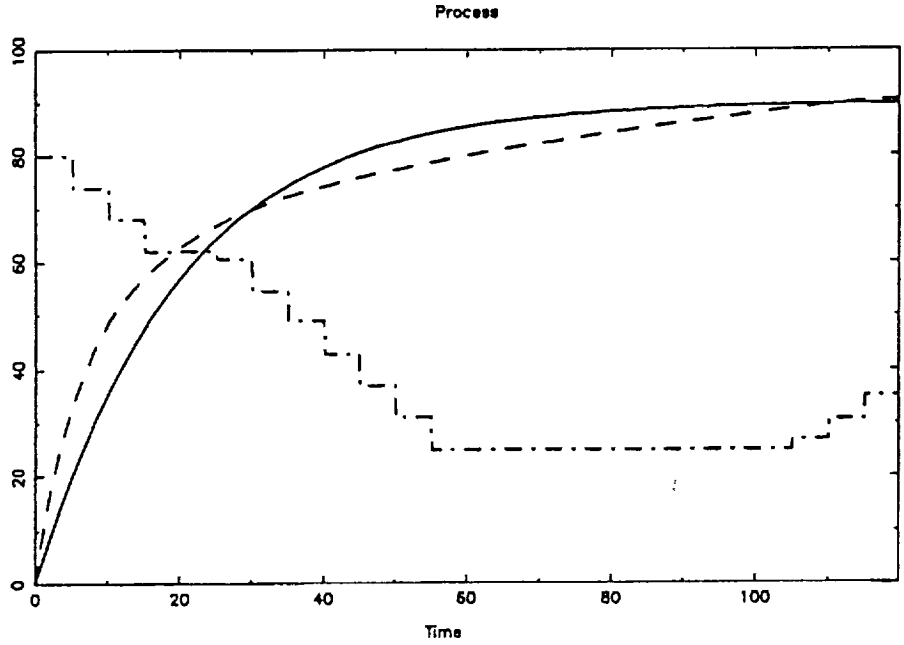

Figure 21: Process response of the fifteenth run (solid line : desired response) (dashed line : actual response) (dashed-point line : control)

4. The Multivariable Fuzzy Logic Control Algorithm

In order to extend the self-learning method of Section 3 to the multi-input, multi-output case, we consider the unknown system to be such that there are n inputs and m outputs, designated by I_i and O_j, respectively. A mapping between each input and output pair is established using the fuzzy logic rules discussed in Section 3. Hence $n \times m$ rulebases are built. Denote the control rulebase between the ith input and jth output by R_{ij}. Further denote the difference between the ith actual output and the ith desired output by e_i and the change of the difference between the ith actual output and the ith desired output by Δe_i. The control rulebase matrix and error/error rate matrix form a pair which when operated on by the fuzzy control process produces a matrix of the control effort as designated by:

$$\begin{bmatrix} R_{11} & \cdots & R_{1m} \\ \vdots & & \\ R_{n1} & \cdots & R_{nm} \end{bmatrix} \otimes \begin{bmatrix} e_1 & \Delta e_1 \\ \vdots & \\ e_m & \Delta e_m \end{bmatrix} \Rightarrow \begin{bmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & & \\ u_{n1} & \cdots & u_{nm} \end{bmatrix}$$

where u_{ij} is the control effort for the (i,j) pair.

To formulate the optimal fuzzy logic controller for the multivariable system, one minimizes the performance measure:

$$J = \underline{E}^T P \underline{E} + \underline{I}^T Q \underline{I}$$

subject to $L < I < H$, where L and H are constraints on the control inputs, P and Q are weighting matrices, and where

$$\underline{E} = \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix} \qquad f_j = \left(\sqrt{e_j^2 + \Delta e_j^2}\right)\left(1 - \sum_{i=1}^{n} I_i / u_{ij}\right) \qquad j = 1,\ldots,m$$

$$\underline{I} = \begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix}$$

The performance measure can be minimized numerically using classical gradient methods, resulting in the control inputs for the next iteration. One notes from the performance measure constructed that, when there is no control rulebase (i.e., no rule is applicable for a particular input-output pair), $u_{ij} \rightarrow \infty$. Hence output j has no effect on input i in the function f_j. This completes the multivariable fuzzy logic control algorithm.

In order to test the fuzzy logic control, a simple low order linear but unknown system with two inputs and three outputs is investigated.

Applying the multivariate FLC yields the results as shown in Figure 22. Here each output achieves its desired terminal value within two units. Figure 23 shows the individual outputs along with the desired terminal values.

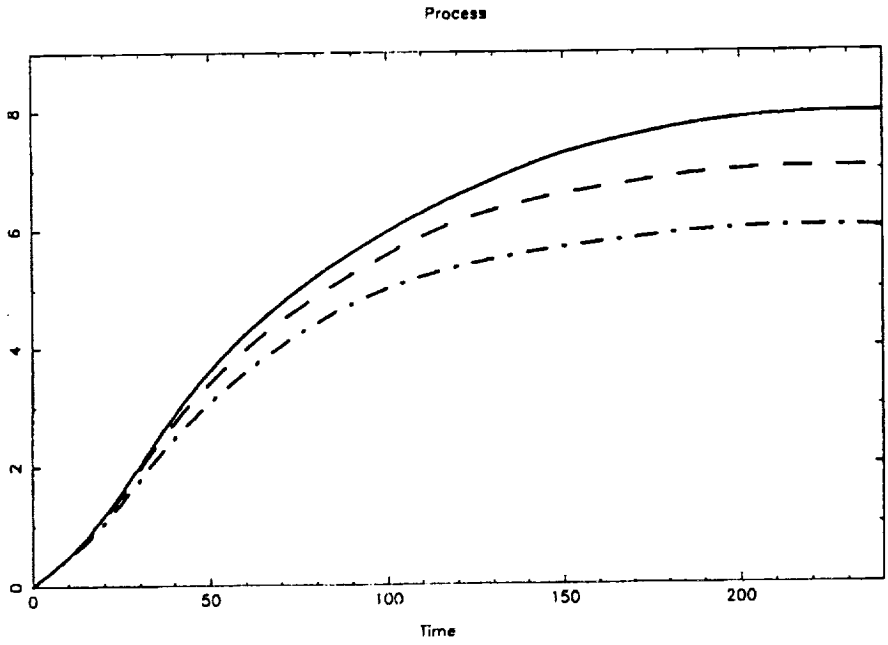

Figure 22: Process response with (solid line : output one) (dashed line : output two) (dashed-point line : output three)

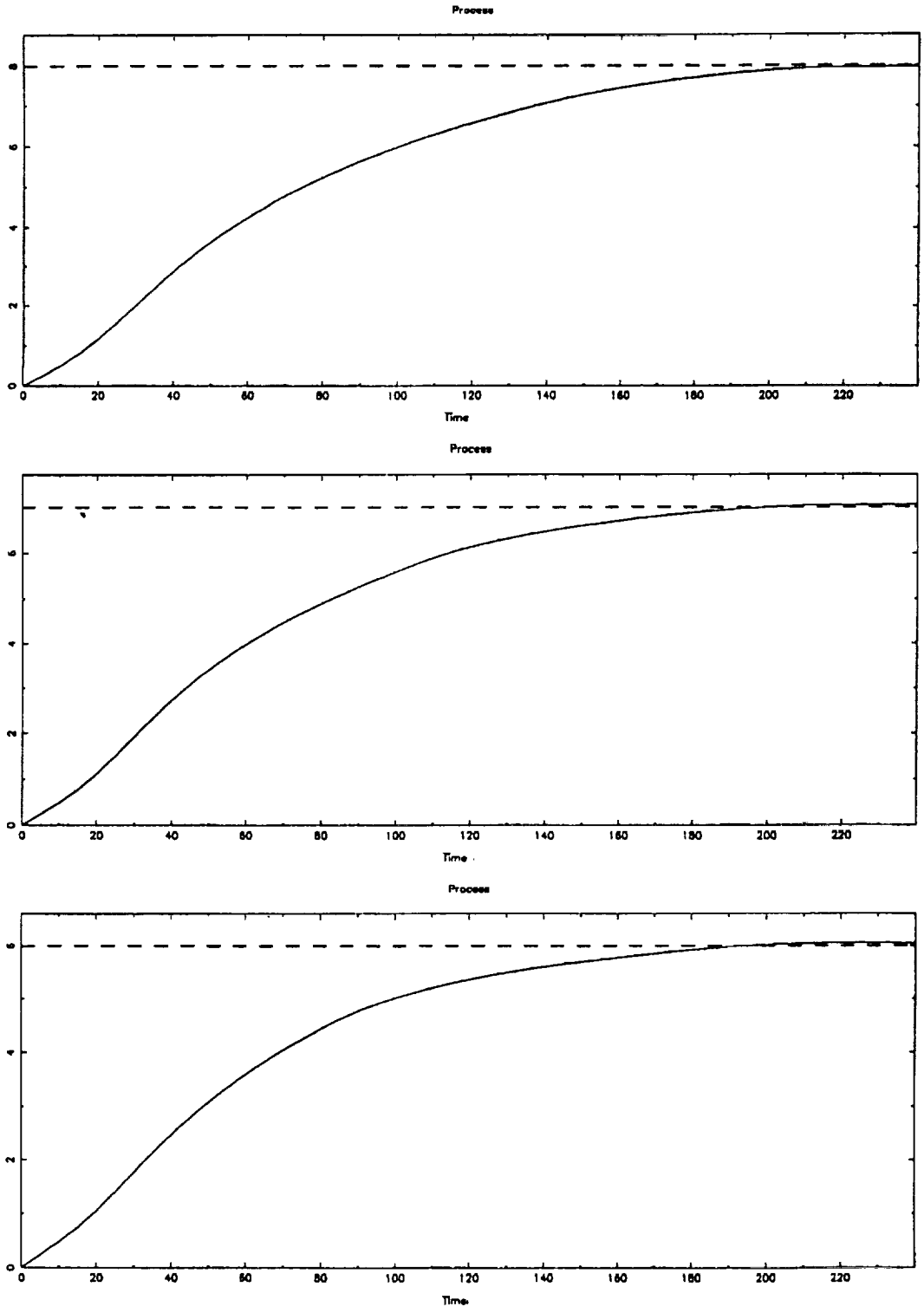

Figure 23: Process response with (solid line : actual output) (dashed line : desired output)

The FLC for the two control inputs is shown in Figures 24 and 25. Physical constraints were placed on both controllers.

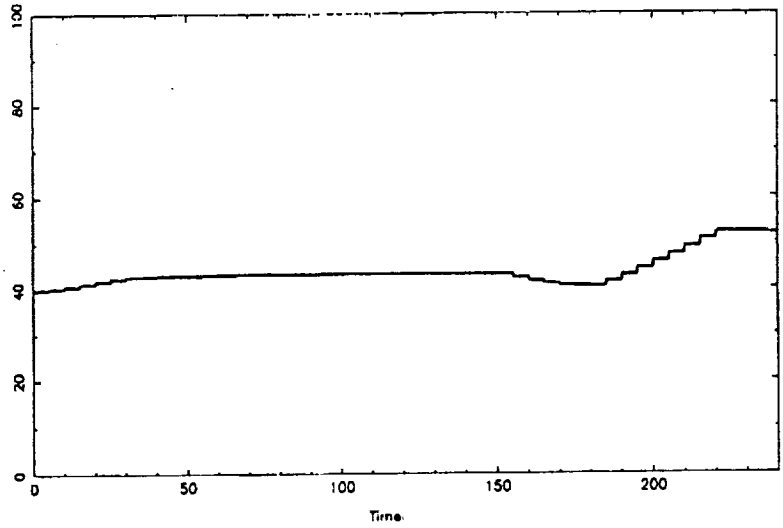

Figure 24: Control one

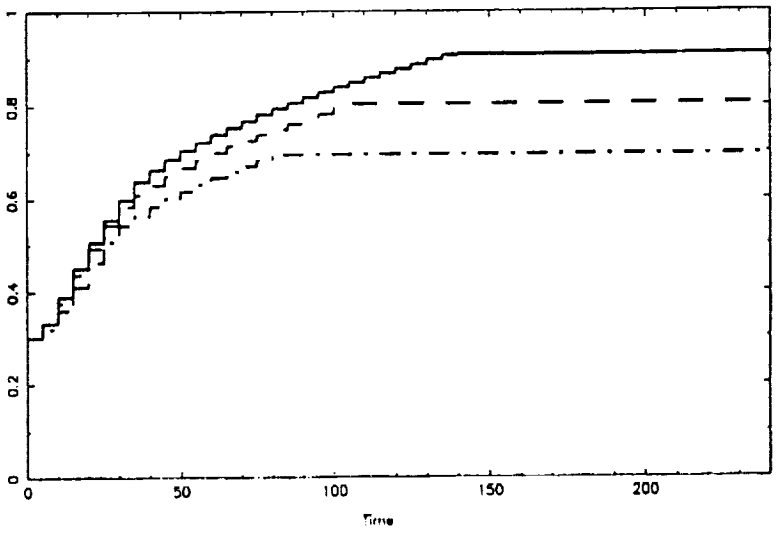

Figure 25: Control two with (solid line : output one) (dashed line : output two) (dashed-point line : output three)

To see the effects of weighing on the importance of control effort, the matrices P and Q were changed. Results are shown in Figures 26-29. Again the FLC produced the desired terminal values.

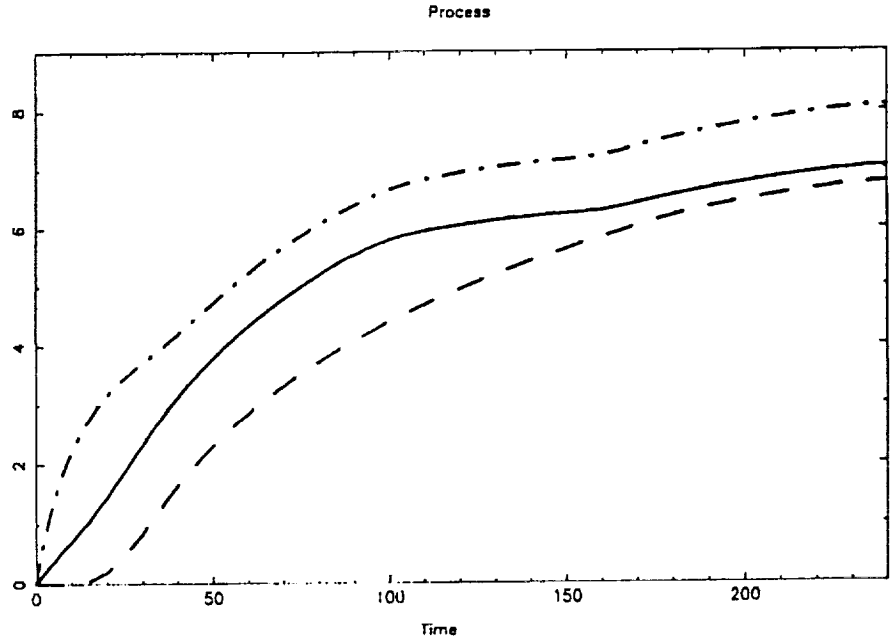

Figure 26: Process response with (solid line : output one) (dashed line : output two) (dashed-point line : output three)

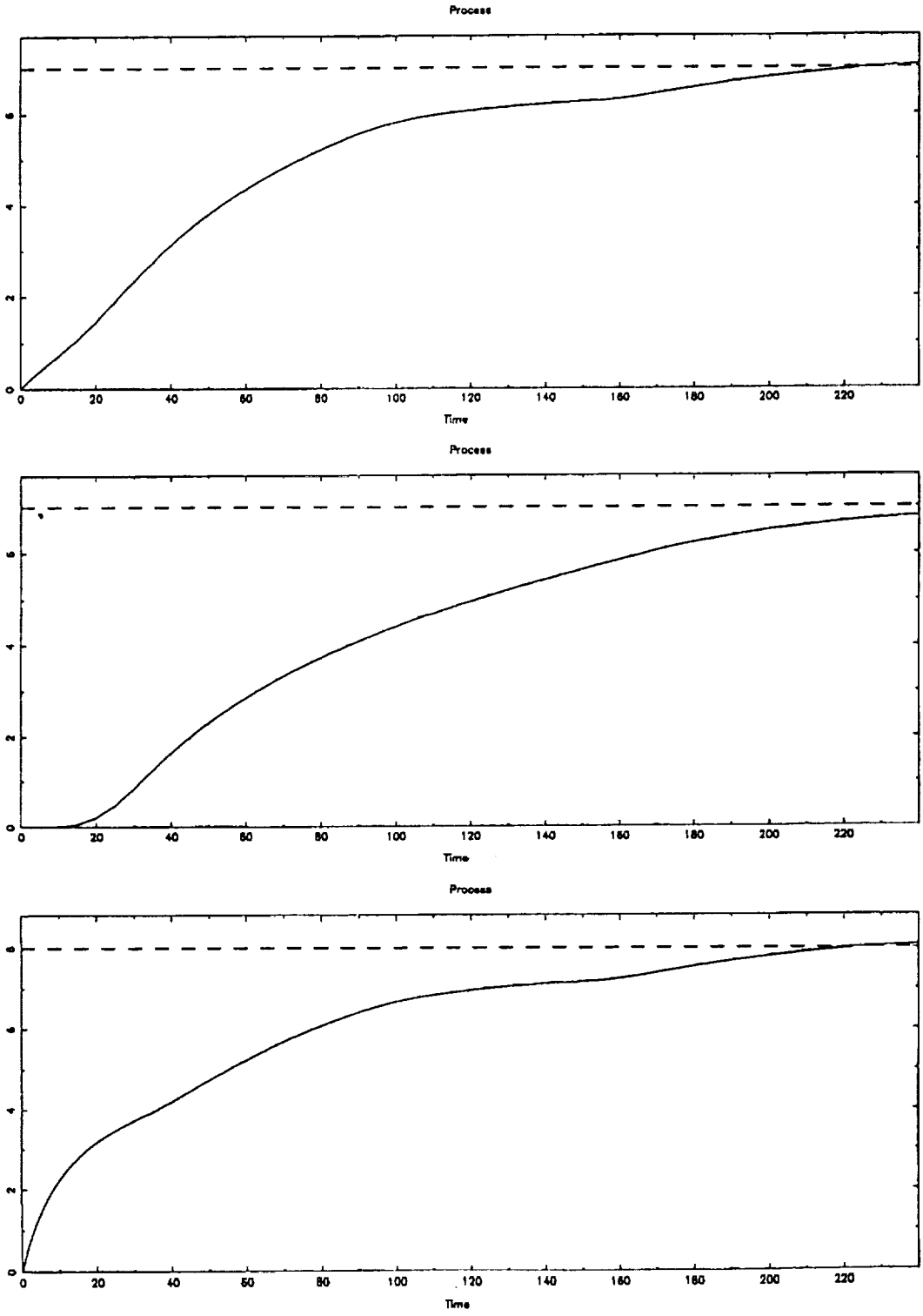

Figure 27: Process response with (solid line : actual output) (dashed line : desired output)

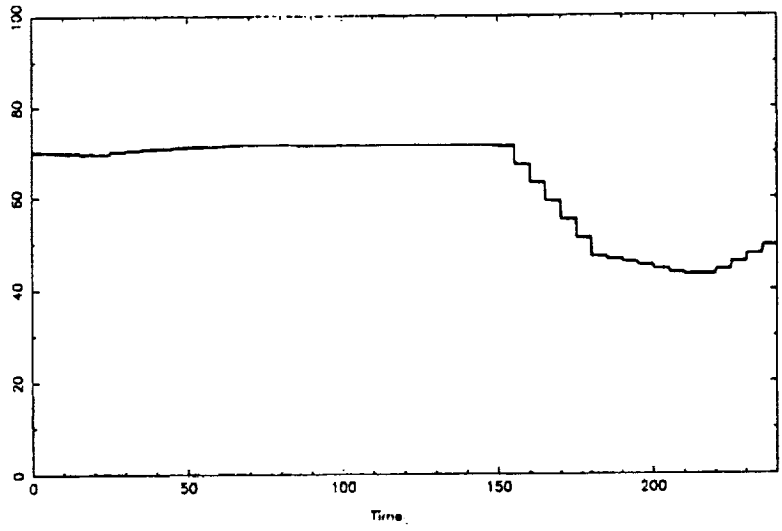

Figure 28: Control one

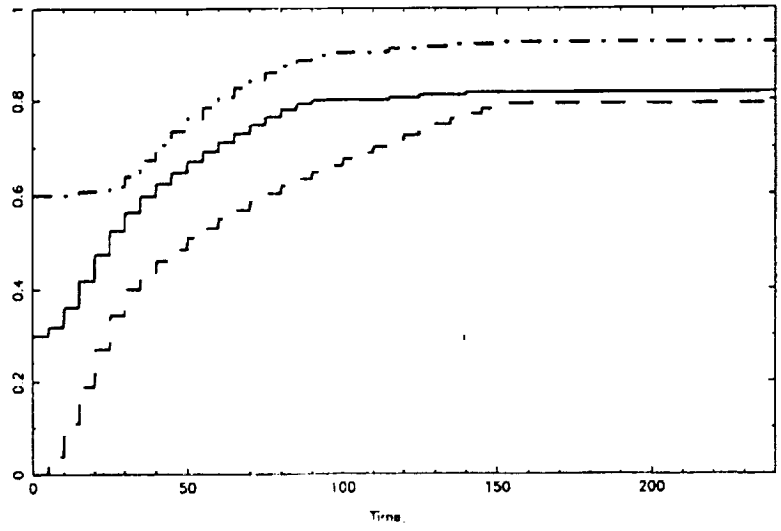

Figure 29: Control two with (solid line : output one) (dashed line : output two) (dashed-point line : output three)

5. Summary

Firstly, two schemes are developed here – the self-tuning method for choosing the scaling factors and the self-learning method for rule base developement which have enhanced the flexibility of fuzzy logic controllers. Then the method is extended to the MIMO case. These schemes can be further modified. For example, if we assume the control surface is smooth, which is similar to assume the human operator's decisions are rational, we can average the control surface after each simulation or experiment. It is expected that this approach would give quicker overall control rule convergence. This is an area of further research.

6. References

[1] Buckley, J. J., Universal Fuzzy Controllers, *Automatica*, Vol. 28, No. 6, pp. 1245-1248, 1992.

[2] Lee, C. C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I and Part II, *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 24, No., 2, pp. 404-435, 1990.

[3] Mamdani, E. H., Applications of Fuzzy Algorithms for Control of Simple Dynamic Plant. *Proc. IEEE.*, Vol. 121, No. 12, 1974.

[4] Mamdani, E. H., The Application of Fuzzy Control Systems to Industrial Processes. *Automatica*, Vol. 13, pp. 235-242, 1977.

[5] Zadeh, L. A., Fuzzy Sets. *J. Information and Control*, Vol. 8, pp. 338-353, 1965.

[6] Zadeh, L. A., Fuzzy Algorithms. *J. Information and Control*, Vol. 12, pp. 94-102, 1968.