'92 韓國自動制御學術會議論文集 1992. 10. 19~21

학습을 이용한 퍼지 제어기의 구성

안 상철* 권 욱현
서울대학교 공과대학 제어계측공학과

A Construction of Fuzzy Controller using Learning

Sang-Chul Ahn* Wook-Hyun Kwon
Dept. of Control & Instrumentation Eng. Seoul National University

ABSTRACT

The inference of fuzzy controller can be considered a mapping from the controller input to membership value. The membership value, a kind of weight, has a role to decide if the input is appropriate to the rule. The membership function is described by several values, which are decided by a learning method. The learning method is adopted from adaptive filtering theory. The simulation shows the proposed fuzzy controller can learn linear and nonlinear functions. The structure of the proposed fuzzy controller becomes a kind of neural network.

1. 서론

Mamdani에 의해 처음으로 사용된 퍼지 논리 제어기는 그 이후 여러분야에서 응용되어 사용되고 있다[1,2,3]. 퍼지 논리를 기본으로하는 퍼지 논리 제어기는 정성적인 언어를 정량적으로 표현할 수 있다는 점과 인간의 언어를 사용하여 인간의 사고를 모방할 수 있다는 점을 그 특징으로 하고있다[1,2,3]. 따라서 플랜트의 모델이 잘 알려져 있지 않거나 기존의 방식으로 잘 정의가 않되는 플랜트에 적용하는 것이 유리하게 된다. 근래에 들어 퍼지 시스템에 대한 연구가 활발히 진행되고 여러분야에 응용되고 있는 가운데 학습을 이용한 시스템의 성능 향상 방법도 여러가지로 대두되고 있다[4,5]. 하지만 퍼지 시스템에서 사용하고 있는 퍼지화, 비퍼지화, 추론 방법등은 학습을 하는데 있어서 많은 어려움을 준다.

본 연구에서는 퍼지 제어기의 멤버쉽 함수를 학습시키므써 퍼지제어기를 구성하도록 하였다. 이를 위하여 퍼지 제어기의 추론 방식을 개념적인 면에서 간략화시켜서 이를 입력으로부터 규칙을 적용시키기에 적합한가를 나타내는 가중치로의 사상으로 보았다. 이 때 이 가중치를 나타내는 멤버쉽 함수를 몇개의 대표값으로 표현하고 이들 대표값을 학습을 통하여 결정할 수 있도록 하였다. 또한 학습을 통해서 규칙의 결론부 값도 바뀌게 하였다. 학습은 adaptive filter를 학습시키는 방법으로 kalman filter를 사용하였다. 본 연구에서 제안한 퍼지 제어기의 구조는 비선형 함수를 사용하지 않은 신경회로망과 같게 되고 각 규칙의 멤버쉽 함수는 신경회로망의 가중치와 같게 된다. 제안한 퍼지 제어기의 성능을 모의 실험으로 보인다.

2. 퍼지 제어기

일반적으로 퍼지 제어기는 물리적인 값을 퍼지값으로 바꾸는 퍼지화부 (fuzzifier)와 이 퍼지 입력에 제어 규칙을 적용하여 결론을 이끌어 내는 추론부 (inference engine), 그리고 결론으로 나온 퍼지값을 물리적인 값으로 바꾸는 비퍼지화부 (defuzzifier)로 구성되어 있다. 여기서 퍼지 제어기의 추론방식을 살펴보기로 하자.

간단히 다음과 같은 두개의 퍼지 제어 규칙이 있다고 가정하자.

R_1 : x_1가 A_1 이고 x_2가 B_1이면 f는 C_1
R_2 : x_1가 A_2 이고 x_2가 B_2이면 f는 C_2

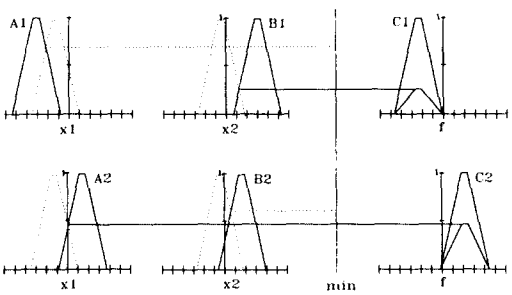


그림 1. 퍼지 제어기의 추론법

이 때 가장 많이 사용되는 min-max 추론법을 사용하면 그림 1 과 같은 결과를 얻을 수 있다. 각 규칙에서 생성되는 결과들을 이용해 궁극적인 출력을 도출하는 비 퍼지화 방법도 여러가지 있으나 많이 사용되는 면적 중심법을 사용

하면 다음과 같다.

$$z = \frac{h_1 f_1 + h_2 f_2}{h_1 + h_2} \qquad (1)$$

여기서 z는 퍼지 제어기의 출력, h_1과 h_2는 추론의 결과로 나온 각 규칙의 멤버쉽 값들이고, f_1, f_2는 각 규칙의 결론 값이다.

위의 과정을 잘 살펴보면 퍼지 제어기의 추론에서 얻어지는 것은 h_1, h_2 같은 멤버쉽 값으로, 이는 각 규칙들이 출력에 미치는 영향의 정도를 나타내고 있는 것을 알 수 있다. 결국 추론 과정에서 얻어지는 것은 각 규칙의 결론부가 퍼지 제어기의 출력을 결정할 때 사용되는 가중치라고 말할 수 있는 것이다.

이와 같은 사실을 생각하면 퍼지 제어기의 추론 과정은 입력을 각 규칙마다 하나의 가중치 (멤버쉽 값)에 사상시키는 함수가 되고, 여기에 규칙의 결론부를 곱하여 출력을 만들어 내게 된다. 따라서 규칙을 구성하는 멤버쉽 함수를 바꾸어 주게되면 규칙도 바뀌고, 그에 해당하는 출력도 바뀐다. 여기서는 이러한 사실에 입각하여 규칙을 구성하고 있는 멤버쉽 함수를 학습하는 퍼지 제어기를 구성한다.

3. 제안된 퍼지 제어기

3.1 구성

퍼지 제어기에서는 일반적으로 삼각형이나 종형의 멤버쉽 함수를 사용하지만, 이제는 일정한 형태를 가지지 않은 멤버쉽 함수를 사용한다. 이는 일정한 형태의 멤버쉽 함수를 사용하는 데에서 오는 에러를 줄이고 적은 수의 규칙으로 원하는 출력을 만들기 위해서이다. 추론 과정을, 입력을 하나의 가중치로 사상시키는 함수로 생각하면, 이 함수는 각 규칙당 입력이 하나일 때 평면위의 곡선이 되고, 입력이 두개일 때는 삼차원상의 곡면이 된다. 이를 그림으로 나타내면 그림 2 와 같다.

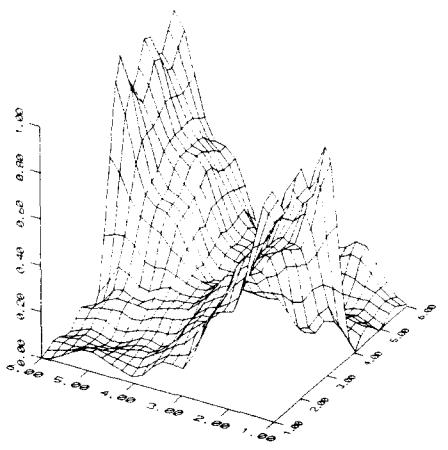


그림 2. 퍼지 제어기 추론함수

그러나 이러한 곡선이나 곡면을 메모리에 다 기억한다면 이는 굉장한 낭비 이므로 이를 줄이기 위한 노력이 필요하게 된다. 이를 위해 대표값들을 이용한 멤버쉽 함수의 표현 방법을 사용한다. 이는 입력의 변화 범위를 몇개의 구역으로 나누고, 각 구역의 양 끝점에 이 점을 대표하는 멤버쉽 값을 두어 그 구역에 해당되는 입력의 멤버쉽 값은 양 끝점 값의 내분점값을 취하는 방법이다. 즉 대표값들을 취해서 이들을 직선이나 평면으로 이은 것을 멤버쉽 함수로 하는 것이다. 이 방법에서는 입력의 변화 범위를 n으로 나누었다면 입력이 하나일 때는 각 규칙당 (n+1)개의 값만을 기억하면 되고 입력이 두개일 때는 $(n+1)^2$개의 값을 기억하면 되는 것이다.

이 때 이 멤버쉽 함수를 이용하여 멤버쉽 값, 즉 규칙의 가중치를 구하는 것을 보이기 위해 다음과 같이 입력이 하나이고 입력의 변화 구간을 두 부분으로 나누었을 때를 살펴보자. 그러면 대표값은 3개 (=2+1)가 생기고, 그림 3 과 같이 입력이 m_1, m_2를 양 끝점으로 하는 영역에 들어왔을 때 규칙의 가중치를 구하면 다음과 같다.

$$h = \frac{b*m_1 + a*m_2}{b + a} = \frac{(x_2-x)*m_1 + (x-x_1)*m_2}{x_2 - x_1} = \frac{x_2-x}{x_2-x_1}*m_1 + \frac{x-x_1}{x_2-x_1}*m_2 + 0*m_3 \qquad (2)$$

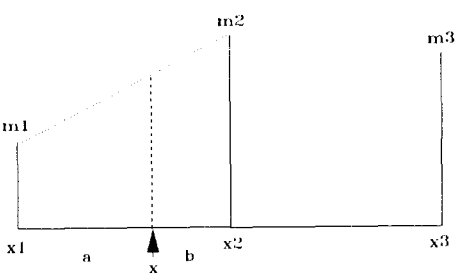


그림 3. 멤버쉽 값 계산 (입력이 하나)

여기서 m_1, m_2는 대표가 되는 멤버쉽 값, x_1, x_2는 구역의 양 끝점에서 입력값, 그리고 x는 실제 입력을 나타내고 있다. 여기서와 같이 멤버쉽 값을 구할 때 주의할 점은 입력 x가 퍼지 값이 아니라는 점이다. 대신 식 (2)를 보면 입력 x가 들어가면 이것이 각 대표값에 곱해지는 벡터로 변하는 것을 볼 수 있다. 즉, 식 (2)에서는 입력이 다음과 같이 변한다.

$$x \dashrightarrow \left[\frac{x_2-x}{x_2-x_1} \quad \frac{x-x_1}{x_2-x_1} \quad 0 \right]$$

또한 입력이 둘이고 각 입력의 변화 구간을 두 부분으로 나누었을 때는 대표값은 9개 ($=(2+1)^2$)가 생긴다. 그림 4와 같이 입력이 m_1, m_2, m_3, m_4 를 모서리 점으로 하는 영역에 들어왔다면 이 때 규칙의 가중치는 다음과 같다.

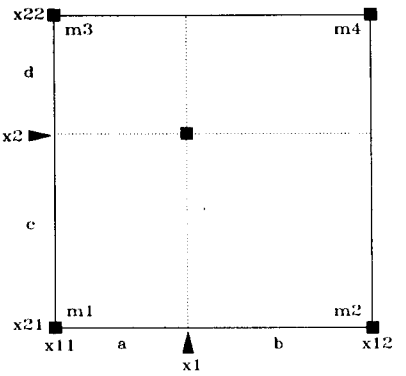


그림 4. 멤버쉽 값 계산 (입력이 둘)

$$h = \frac{b*d*m_1 + a*d*m_2 + b*c*m_3 + a*c*m_4}{(a+b)(c+d)} \qquad (3)$$

$$a = x_1 - x_{11}$$
$$b = x_{12} - x_1$$
$$c = x_2 - x_{21}$$
$$d = x_{22} - x_2$$

$$x \;\text{---}> \frac{1}{(a+b)(c+d)} [\; b*d \;\; a*d \;\; b*c \;\; a*c \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;]$$

여기서 x_1, x_2는 실제 입력이다.

결국 각 규칙들의 멤버쉽 값들은 다음과 같이 구할 수 있게 된다.

$$h_i = \sum_{j=1}^{L} m_{ij} x_j \qquad (4)$$

여기서 i는 규칙의 번호를 나타내고, j는 대표값 번호를, 그리고 L은 입력 구역을 나눈 값 + 1 이다.

이렇게 해서 멤버쉽 값, 가중치를 구하면 이를 이용해 퍼지 제어기의 출력을 구한다. 출력을 구할 때는 일반 퍼지 제어기가 (1)식으로 출력을 구하는 것과는 달리 식 (3)과 같이 분모 부분을 생략한 간단한 식으로 구한다.

$$z = \sum_{i=1}^{N} h_i f_i \qquad (5)$$

여기서 i는 규칙의 번호를 나타내고, f_i는 규칙의 결론 값, 그리고 N은 규칙의 갯수이다.

3.2 학습 방법

여기서는 앞에서 제안한 퍼지 제어기의 학습 방법에 대해서 논의 한다. 여기서 학습을 하게 되는 것은 앞에서 설명한 멤버쉽 함수를 나타내는 대표값들이다. 즉, 입력의 멤버쉽 함수를 바꾸어서 원하는 출력을 얻도록하는 것이다. 이 방법은 일종의 Adaptive filter를 학습시키는 방법으로 신경회로망을 학습시키는 데 사용되기도 하였다 [6,7].

다음과 같이 제어기 출력의 오차를 정의한다.

$$E = \sum_{k=1}^{t} (d(k) - z(k))^2 \qquad (6)$$

여기서 d(k)는 원하는 출력 값이고 z(k)는 퍼지제어기의 출력, k는 시간을 나타낸다. 한편 (5)식에 나타난 퍼지제어기의 출력을 다시 써 보면 다음과 같다.

$$z(k) = \sum_{i=1}^{N} h_i f_i \qquad (7)$$

여기서 N은 규칙의 갯수, h_i는 규칙의 가중치, f_i는 각 규칙의 결론부 값을 나타낸다. 이 퍼지제어기의 출력을 (4)식을 넣어 풀어 써 보면 다음과 같다.

$$\begin{aligned} z(k) &= \sum_{i=1}^{N} \sum_{j=1}^{L} m_{ij} x_j(k) f_i \\ &= \sum_{j=1}^{L} x_j(k) \sum_{i=1}^{N} m_{ij} f_i \\ &= \sum_{j=1}^{L} x_j(k) w_j \end{aligned} \qquad (8)$$

$$w_j = \sum_{i=1}^{N} m_{ij} f_i \qquad (9)$$

이제 제어기 출력의 오차를 위에서 정의된 w_n로 편미분하면 다음과 같은 식을 L개 얻는다.

$$\frac{E}{w_n} = -2 \sum_{k=1}^{t} (d(k) - \sum_{j=1}^{L} w_j x_j(k)) x_n(k) = 0 \qquad (10)$$

또는

$$\sum_{k=1}^{t} d(k) x_n(k) = \sum_{k=1}^{t} \sum_{j=1}^{L} w_j x_j(k) x_n(k) \qquad (11)$$

이 된다. 여기서 n은 1에서 L까지이다. 이제 우측식을 다시 써 보면 다음과 같다.

$$\sum_{k=1}^{t} \sum_{j=1}^{L} w_j x_j(k) x_n(k) = \sum_{k=1}^{t} x_n(k) \sum_{j=1}^{L} w_j x_j(k) \qquad (12)$$

오른쪽 합을 벡터곱으로 표시하면

$$\sum_{k=1}^{t} x_n(k) \underline{w}^T \underline{x}(k) = \sum_{k=1}^{t} x_n(k) \underline{x}(k)^T \underline{w} \qquad (13)$$

이다. 여기서 $\underline{w}$, $\underline{x}(k)$는 벡터이다. 이제 다음을 정의하자.

$$\underline{R} = \sum_{k=1}^{t} \underline{x}(k)\,\underline{x}(k)^T \qquad (14)$$

$$\underline{p} = \sum_{k=1}^{t} d(k)\,\underline{x}(k) \qquad (15)$$

그러면 (11)식은 다음과 같은 행렬식으로 나타내어 진다.

$$\underline{p} = \underline{R}\,\underline{w} \qquad (16)$$

여기서 행렬 $\underline{R}$은 입력 $\underline{x}$의 correlation 행렬이고 벡터 $\underline{p}$는 입력과 원하는 출력과의 cross correlation이다.

(16) 식에서 $\underline{w}$ 벡터는 (17)식과 같이 구할 수 있지만 이러한 방법은 $\underline{R}$의 크기가 커질 경우 계산량이 많아지고, 정해진 길이의 데이타가 모두 입력이 된 후에야 $\underline{w}$를 새로이 변화시킬 수 있다는 단점이 있다.

$$\underline{w} = \underline{R}^{-1}\,\underline{p} \qquad (17)$$

따라서 입력이 계속들어오는 경우 순환적으로 $\underline{w}$를 변화시켜 나가는 방법이 필요하다. 이를 위해서 (14) (15) 식을 순환형으로 바꾸면 다음과 같다.

$$\underline{R}(t) = b\,\underline{R}(t-1) + \underline{x}(t)\,\underline{x}^T(t) \qquad (18)$$

$$\underline{p}(t) = b\,\underline{p}(t-1) + d(t)\,\underline{x}(t) \qquad (19)$$

여기서 b는 forgetting factor이다. 하지만 (17)식의 $\underline{w}$ 을 구하기 위해서는 $\underline{R}$의 순환형식 보다는 $\underline{R}^{-1}$의 순환식이 필요하다. 따라서 여기에 Kalman filter 방법을 적용한다.

결국 $\underline{R}^{-1}$의 순환식은

$$\underline{R}^{-1}(t) = [\underline{R}^{-1}(t-1) - \underline{k}(t)\underline{x}^T(t)\underline{R}^{-1}(t-1)]\,b^{-1} \qquad (20)$$

이 되고, 이 때 Kalman gain vector $\underline{k}$는 다음과 같다.

$$\underline{k}(t) = \frac{\underline{R}^{-1}(t-1)\underline{x}(t)}{b + \underline{x}^T(t)\underline{R}^{-1}(t-1)\underline{x}(t)} \qquad (21)$$

$\underline{w}$ 벡터의 학습 방법은

$$\underline{w}(t) = \underline{w}(t-1) + \underline{k}(t)\,(d(t) - z(t)) \qquad (22)$$

이 된다. 그런데 우리가 학습을 시키려고 하는 것은 $\underline{w}$ 벡터가 아니고 $\underline{w}$를 구하는데 사용하는 $\underline{m}$벡터이므로 $\underline{m}$ 벡터 변환식을 만들어야 한다. 식 (9)의 $\underline{w}$ 를 보면 다음을 알 수 있다.

$$dw_j = dm_{ij}\,f_i \qquad (23)$$

따라서

$$dm_{ij} = dw_j \,/\, f_i \qquad (24)$$

이 된다. 이를 이용하면 멤버쉽 함수의 대표값 벡터 $\underline{m}_i$를 학습시키는 식은 다음과 같다.

$$\underline{m}_i(t) = \underline{m}_i(t-1) + k(t)\,(d(t)-z(t))\,/\,f_i \qquad (25)$$

이렇게 멤버쉽 값을 바꾸어 주게 되면 멤버쉽 값은 0에서 1사이의 값을 넘어서게 될 수도 있는데 0 이하 일 때는 0으로 만들어 주고 1이상이 될 때에는 전체적으로 정규화를 시켜준다. 이 때 정규화된 값만큼 규칙의 결론부 값을 보상해 주어 규칙의 결론부도 학습하는 효과를 보게 한다.

플랜트와 결합된 퍼지 제어기에서는 적절한 제어기의 출력, 즉 플랜트의 입력을 알지못하기 때문에 앞에서 정의한 제어기 출력의 오차를 구하지 못하고 따라서 학습을 하지 못한다. 하지만 플랜트를 신경회로망에서의 하나의 계층으로 보고 오차 역전달 방법을 사용하면 제어기의 출력측에서의 오차를 구할 수있게 된다. 이렇게 구한 오차를 이용하여 제어기를 학습시키게 된다.

4. 모의 실험

퍼지 제어기의 학습능력을 보기위하여 다음과 같은 비선형 함수를 학습시켜 보았다.

$$z = x(x-0.2)(x+0.2)$$

퍼지 제어기는 규칙을 5개로 하였고, 입력의 변화범위를 10개의 구역으로 나누어 학습을 시켰다. 그림 5 는 학습이 끝난 후 같은 랜덤 입력에 대한 위 함수의 출력과 제어기의 출력을 비교해 본 것이다. 그림을 보면 두개의 그래프를 구별할 수 없을 정도로 학습이 잘된 것을 알 수 있다. 그림 6은 학습이 끝난 퍼지 제어기의 멤버쉽 함수를 나타낸 것으로 각 규칙당 멤버쉽 함수를 모아놓은 것이다. 0에서 10까지 표시되어 있는 축이 대표값을 나타내는 것이고, 1에서 5까지 표시되어 있는 축이 규칙을 나타내는 것이다. 표 1 은 그림 6에서 나타낸 각 규칙 멤버쉽 함수의 대표값들과 그 규칙들의 결론부 값이다.

퍼지 제어기가 플랜트를 제어하는 것을 보이기 위해 간단한 플랜트를 잡아서 제어해 보았다. 플랜트는 다음과 같다.

$$\dot{x} = -2x + u$$

여기서도 퍼지 제어기는 규칙을 5개로 하였고, 입력의 변화범위를 10개의 구역으로 나누어 학습을 시켰다. 플랜트는 0.5초 마다 제어 입력을 가하였다. 그림 7은 학습이 끝난 후 플랜트를 제어하는 제어기의 성능을 보인 것이다. 그린 8는 학습이 끝난 퍼지 제어기의 멤버쉽 함수를 나타낸 것이다. 표 2 는 역시 퍼지제어기 입력에 대한 멤버쉽 함수의 대표값과 규칙의 결론부 값이다.

5. 제어기의 해석

제안된 퍼지 제어기는 멤버쉽 함수를 변화시켜서 원하는 출력을 얻고자 하는 것이다. 즉, 멤버쉽 값들은 입력이 각각의 규칙을 적용하기에 적합한지 적합하지 않은지를 나타내는 것이므로, 멤버쉽 값들을 변화시켜서 들어오는 입력에 따라 규칙을 적용하게 한다. 하지만 멤버쉽은 '어떤 집합의 소속도'라는 개념을 빼고 제어기 출력을 얻는 식 (4),(5)을 보면 입력 벡터 x와 멤버쉽 대표값 벡터 $\underline{m}_i$의 내적으로 h_i를 구하고, 다시 이들로 이루어진 $\underline{h}$벡터와 규칙의 결론부를 나타내는 $\underline{f}$벡터의 내적으로 출력을 이끌어 내는 것을 볼 수 있다. 이는 시그모이드(sigmoid) 함수같은 비선형 함수대신 선형함수를 사용한 3계층 신경회로망과 같은 식이 된다. 다만 여기서는 스칼라 입력 x를 식 (2)나 (3)처럼 벡터 $\underline{x}$ 꼴로 변환하여 사용한다. 따라서 다계층 신경회로망을 사용하면 다단계 추론을 거치는 퍼지 제어기를 구성할 수도 있을 것이다.

6. 결론

퍼지 시스템에 대한 연구가 활발히 진행되는 가운데 학습을 이용한 시스템의 성능 향상 방법도 여러가지로 대두되고 있다. 본 연구에서는 퍼지 제어기의 추론 방식을 개념적인 면에서 간략화시켜서 이를 입력으로부터 규칙을 적용시키기에 적합한가를 나타내는 가중치로의 사상으로 보았다. 이 때 이 가중치를 나타내는 멤버쉽 함수를 몇개의 대표값으로 표현하고 이들 대표값을 학습을 통하여 결정할 수 있도록 하였다. 이러한 학습의 결과로 규칙의 결론부 값도 학습이 되게 되었다. 학습은 adaptive filter를 학습시키는 방법을 사용하였다. 제안된 퍼지제어기는 모의 실험을 통해서 학습 성능을 확인했고, 간단한 플랜트에 적용하여 제어 성능을 보았다. 본 연구에서 제안한 퍼지 제어기의 구조는 비선형 함수를 사용하지 않은 신경회로망과 같게 되므로 이와 같은 사실을 이용하면 앞으로 퍼지제어기와 신경회로망 제어기의 장점을 모두 갖춘 제어기를 설계할 수 있을 것이라 생각된다.

7. 참고문헌

[1] Chuen Chien Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part I and part II," *IEEE Tr. on SMC, Vol.20, No. 2*, March/April, 1990.

[2] Michio Sugeno, "An Introductory Survey of Fuzzy Control," *Information Sciences 36*, pp. 59-83, 1985.

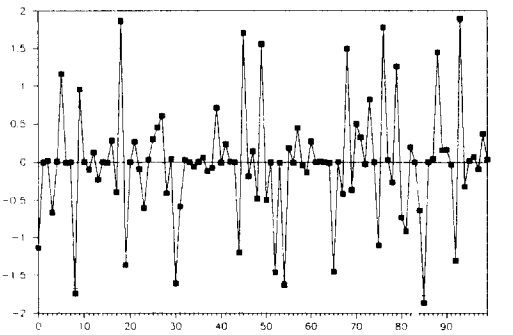
[3] **菅野道夫 著**, 박민용·최항식 **譯**, 퍼지 제어 시스템, 대영사, 1990 년.

[4] Shin-ichi Horikawa, et al, "A Fuzzy Controller Using a Neural Network and Its Capability to Learn Experts Control Rules," *Proc. of Intr. Conf. on Fuzzy Logic & Neural Networks*, pp. 103-106, 1990.

[5] 박세희, 김용호, "유전 알고리듬을 이용한 퍼지 규칙 베이스의 자동생성," 전자공학회논문지 29권, pp. 172-179, 1992.

[6] Athanasios Papoulis, *Probability, Random Variables, and Stochastic Process*, pp. 467-473, 1984.

[7] Robert S. Scalero, Nazif Tepedelenlioglu, "A Fast New Algorithm for Training Feedforward Neural Networks," *IEEE Tr. on Signal Processing*, Vol.40, No.1, Jan., pp. 202-210, 1992.



■ : 함수의 출력 + : 퍼지제어기

그림 5. 퍼지 제어기의 출력과 함수의 출력

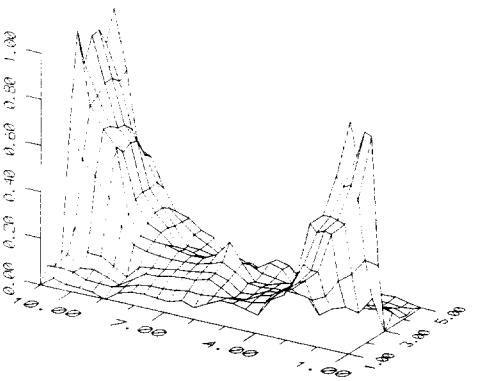


그림 6. 멤버쉽 함수 (비선형 함수 학습)

규칙	1	2	3	4	5
	1.000	1.000	0.000	0.000	0.000
	0.571	0.517	0.024	0.026	0.024
대	0.187	0.204	0.000	0.000	0.000
	0.076	0.125	0.119	0.004	0.037
표	0.109	0.060	0.077	0.154	0.029
	0.051	0.143	0.233	0.052	0.000
값	0.046	0.120	0.142	0.075	0.016
	0.055	0.103	0.161	0.073	0.123
	0.000	0.000	0.171	0.234	0.178
	0.018	0.016	0.624	0.538	0.382
	0.000	0.000	1.000	1.000	1.000
결론값	-4.310	-4.719	3.200	2.930	3.212

표 1. 멤버쉽 함수의 대표값및 결론값 (비선형 함수)

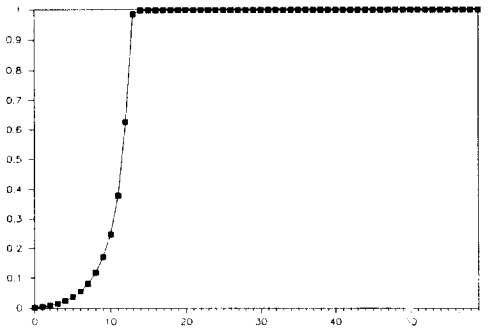


그림 7. 플랜트의 출력

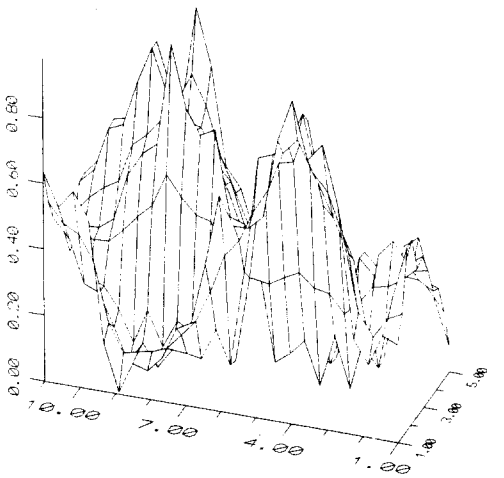


그림 8. 멤버쉽 함수 (제어기)

규칙	1	2	3	4	5
	0.301	0.603	0.461	0.302	0.085
	0.197	0.047	0.520	0.347	0.422
대	0.879	0.608	0.008	0.443	0.042
	1.00	1.00	0.000	0.000	0.000
표	0.570	0.896	0.000	0.326	0.000
	0.658	0.000	0.464	0.261	0.142
값	0.250	0.000	0.806	0.520	0.453
	0.088	0.000	1.000	0.646	1.000
	0.003	0.003	0.630	1.00	0.340
	0.630	0.212	0.365	0.713	0.441
	0.639	0.459	0.292	0.459	0.424
결론값	-1.659	-1.645	0.799	1.400	2.261

표 2. 멤버쉽 함수의 대표값및 결론값 (제어기)