대한기계학회논문집(A) 제 20 권 제 2 호, pp. 399~407, 1996

〈논 문〉

최적의 퍼지제어규칙을 얻기위한 퍼지학습법

정 병 묵*

(1995년 1월 9일 접수)

A Learning Algorithm for Optimal Fuzzy Control Rules

Byeong-Mook Chung

Key Words : Fuzzy Control(퍼지제어), Learning Control(학습제어), Optimal Control(최적제어), Model Reference Fuzzy Control(모델규범 퍼지제어), Regulation Control(조정제어)

Abstract

A fuzzy learning algorithm to get the optimal fuzzy rules is presented in this paper. The algorithm introduces a reference model to generate a desired output and a performance index function instead of the performance index table. The performance index function is a cost function based on the error and error-rate between the reference and plant output. The cost function is minimized by a gradient method and the control input is also updated. In this case, the control rules which generate the desired response can be obtained by changing the portion of the error-rate in the cost function. In this learning algorithm, the meaning of the learning delay is also very important. In SISO(Single-Input Single-Output) plant, only by the learning delay, it is possible to express the plant model and to get the desired control rules. In the long run, this algorithm gives us the good control rules with a minimal amount of prior information about the environment.

1. 서 론

최근 몇 년 동안 퍼지제어는 퍼지집합 이론의 응용분야, 특히 생산현장의 문제와 같이 종래의 방법으로는 제어가 거의 불가능한 분야에서 연구가 가장 활발하였고, 또한 실속있는 결과를 얻을 수 있었다. 이러한 퍼지제어는 종래의 논리 시스템보다는 인간의 사고나 언어구조와 흡사한 퍼지논리 시스템에 기초를 두고 있으며 퍼지논리 제어기는 전문가의 지식을 기초로 한 언어적 제어규칙을 일반적인 자동제어의 원리로 변환할 수 있는 알고리즘을 제공한다. 그러나 지금까지의 퍼지제어는 인간이 축적한 경험지식을 단순히 제어규칙으로 활용하는 데에만 중점을 두어왔기 때문에 복잡한 시스템의 경우에 대해서는 인간의 직관과 경험만으로 퍼지제어의 규칙을 얻기가 거의 불가능하였고 비록 제어규칙을 구했다고 하더라도 그 규칙의 성능이 최상인지 혹은 개선의 여지가 있는지를 알 길이 없었다. 아울러 시스템이 조금만 달라지거나 또는 전혀 새로운 시스템의 제어가 필요한 경우, 기존의 규칙을 어떻게 수정해야 하며 새롭게 만들어야 할 것인지가 문제가 되었다. 따라서 이제는 인간의 경험적 제어규칙에 대한 성능을 평가함으로써 획득된 경험이 최적인지의 여부를 알아 볼 필요가 있고 또한 위에서 언급한 여러가지 문제점들에 대한 해결책이

*회원, 영남대학교 기계공학과

필요하겠기에 최적의 퍼지제어규칙을 얻을 수 있는 학습법을 제안하고자 한다.

이러한 문제를 해결하기 위해 제안한 퍼지학습법에 관한 연구사례가 몇 가지 있긴 하지만,[1~4] 이들 대부분의 특징은 두 개의 규칙베이스를 갖는 계층적 구조로 되어 있다는 것이다. 즉, 첫번째의 것은 일반적인 퍼지제어기의 규칙베이스인데 반해 두번째의 것은 인간과 같은 학습능력을 나타내기 위해 원하는 성능을 기초로 성능지수표를 만들어 둔 것으로 이 지수표에 따라 첫번째의 제어규칙 다시 말하자면 플랜트의 퍼지제어규칙을 생성하거나 수정하도록 하는 성능에 관한 규칙베이스이다. 따라서 두번째 규칙베이스인 성능지수표는 플랜트의 출력을 계속 감시하면서 제어의 성능이 나쁠 경우에만 제어규칙을 바꾸도록 하고 있다. 지금까지의 퍼지학습법은 이러한 성능지수표를 만들어 놓고 이것을 기준으로 규칙을 개선하도록 했기 때문에 먼저 자신이 원하는 제어결과를 반영할 수 있는 성능지수표를 만들고 나서 이것을 기준으로 학습함으로써 제어규칙을 습득하게 하였다. 그러나 이렇게 습득된 규칙의 결과가 원하는 결과와 다를 경우에는 먼저 성능지수표를 수정한 다음 이 성능지수표에 따라 제어규칙을 다시 학습해야 한다. 따라서 얻어진 결과가 다소 불만족스러울 경우 성능지수표를 어떻게 수정해야 할 지가 상당히 어려운 문제일 뿐만 아니라 이렇게 해서 얻어진 규칙이 정말로 최상의 규칙인지도 알 수가 없다. 그래서 Chung & Oh[5]는 출력의 사양을 간접적으로 표현하는 성능지수표 대신에 규범모델(reference model)을 통하여 원하는 출력을 바로 제안하게 하였고, 이렇게 제안된 출력과 실제 출력을 서로 비교하여 그 차이가 주어진 오차의 한계보다 작을때 비로소 학습이 완료되게 함으로써 원하는 출력을 가장 잘 표현하는 최적의 제어규칙을 얻을 수 있다는 것을 비최소위상(non-minimum phase) 플랜트의 대표적인 사례인 유연외팔보에 대한 사례를 제시했었다. 그러나 유연외팔보 문제와는 달리 초기상태가 불안정한 플랜트의 경우는 처음에 규칙이 하나도 없다면 원하는 제어규칙을 얻기 위한 학습이 이루어 지기도 전에 제어가 불가능한 상태로 될 수 있으므로 더 이상의 학습을 기대하기 어려운 문제가 나타나게 된다. 따라서 본 논문에서는 불안정한 플랜트인 막대차문제를 통하여 원하는 규칙을 얻기위한 학습의 초기조건을 알아보고 아울러 이렇게 획득된 제어규칙이 어떻게 추론되어 사용될 수 있는지를 자세히 알아보고자 한다.

2. 퍼지제어법

제어란 플랜트의 실제출력이 원하는 출력과 일치하도록 플랜트의 입력을 적절히 조정하는 것이다. 따라서 제어를 위한 규칙은 원하는 출력이 제시될 때, 이를 가장 잘 만족하도록 입력을 결정해 주는 것이다. 일반적으로 동적 시스템의 출력은 다음과 같은 비선형 차분방정식으로 표현할 수 있다.

$$y(k)=g[y(k-1),\ y(k-2),\ \cdots,\ y(k-p),\ u(k-n),\ \cdots,\ u(k-m)] \tag{1}$$

여기서, n은 시스템의 시간지연, m은 입력의 차수이고, p는 시스템의 차수$(0<n<m<p)$이다. 이때 플랜트의 제어입력 $u(k)$는 다음과 같이 구해진다.

$$u(k)=f[y_d(k+n),\ \cdots,\ y_d(k+1),\ y(k),\ \cdots,\ y(k-p+n),\ u(k-1),\ \cdots,\ u(k-m+n)] \tag{2}$$

여기서, $y_d(k+n)$: $(k+n)$번째의 원하는 출력이다.

이 식에서 볼때, 현재의 입력 $u(k)$는 과거의 입력과 출력뿐만 아니라 미래에 원하는 출력치의 함수임을 알 수가 있다. 즉, 미래의 출력치에 대한 정확한 예측이 가능해야만이 현재의 제어 입력값을 제대로 추정할 수 있다. 따라서 위와 같은 시스템 방정식의 경우는 모델링하기도 어렵지만 현재의 입력에 대해서 역모델 식을 푸는 것도 쉽지 않다. 따라서 퍼지제어에서는 위와 같은 시스템 방정식을 풀지 않고 제어입력에 대한 역모델을 에러에 대한 함수의 표현으로 바로 구하게 되는데 다음과 같은 퍼지관계식으로 나타낼 수 있다.

$$u(k)=F[e(k),\ ie(k),\ de(k)]\ \text{at}\ y=y_0+\Delta y \tag{3}$$

여기서, F는 퍼지관계식, y_0은 기준 출력, $e(k)$은 출력 에러, $ie(k)$는 출력에러의 합, $de(k)$는 출력에러의 변화이고 $u(k)$는 제어 입력이다.

이 식은 퍼지제어에서 기존의 PID 제어기와 같은 형태로 표현한 것으로써 여기서, 말하는 인간의 경험적 규칙이란 "만일 출력의 에러가 positive big

이고 에러의 변화율이 negative small이면 제어입력을 positive midium으로 하라"라는 식으로 나타낼 수 있다. 그러나 인간의 경험적 지식으로 규칙을 구성하는 것이 아니라 학습에 의해 제어규칙을 만들어 낼 때에는 규칙의 후건부(consequent part)에 대해서만은 언어적 표현을 쓰지 않고 비퍼지화한 수치(crisp value)로 표현하는 것이 학습과정에서 컴퓨터에 의한 계산을 용이하게 하고 규칙의 추론 과정도 간단하게 해 준다. 따라서 퍼지학습법에서 쓰여질 규칙은 "만일 출력의 에러가 positive big이고 에러의 변화율이 negative small이면 제어입력을 6.0으로 하라"라는 식으로 후건부의 표현을 조금 다르게 나타낸 규칙을 얻을 수 있다.[5,6] 퍼지집합 A의 멤버쉽 함수를 $A(x)$라고 나타낼때, 전제 "*If e is E and s is S and c is C*"의 진리값은 $E(e)\ \Lambda\ S(s)\ \Lambda\ C(c)$로 표현할 수 있고 이러한 진리값을 사용한 퍼지함의(fuzzy implication) R은 다음과 같이 나타낸다.

$$R : E(e)\ \Lambda\ S(s)\ \Lambda\ C(c) \rightarrow v \tag{4}$$

여기서, 사용된 후건부의 값 v는 비퍼지화된 값(defuzzified value)으로서 퍼지싱글톤(fuzzy singleton)이라고도 하는데 우리가 흔히 사용하는 일반변수와 같다. 학습에 의해 퍼지규칙을 생성할 때에는 후건부의 제어입력이 계속적으로 수정되어야 하므로 퍼지화된 언어적 변수대신에 우리가 흔히 사용하는 일반변수를 사용하였다. 이러한 규칙의 구조는 상태변수 e, s, c로 구성된 3차원 규칙 메트릭스를 나타내며 k번째 제어규칙 R_k는 다음과 같이 표현할 수 있다.

$$R_k : E_k(e)\ \Lambda\ S_k(s)\ \Lambda\ C_k(c) \rightarrow v_k$$
$$k=1,\ 2,\ \cdots,\ M \tag{5}$$

따라서, 이 규칙의 의미를 살펴보면 플랜트의 제어입력이 v_k라는 값을 가질 가능성은 각각의 퍼지집합 $E_k(e)$, $S_k(s)$, $C_k(c)$에서 멤버쉽함수가 갖는 확률의 최소값과 같다는 것을 나타낸다. 여기서, M은 전체 규칙의 개수를 나타내며 각각의 규칙은 퍼지 함수에 의해 계산되어야 하므로 다음과 같이 정의된다.

$$\mu_{R_k} \approx \mu(E_k \text{ and } S_k \text{ and } C_k \rightarrow v_k)$$
$$= [\mu_{E_k}(e)\ \Lambda\ \mu_{S_k}(s)\ \Lambda\ \mu_{C_k}(c)] \rightarrow v_k \tag{6}$$

규칙의 전제부(premise)인 $\mu_{E_k}(e)\ \Lambda\ \mu_{S_k}(s)\ \Lambda\ \mu_{C_k}(c)$는 k번째 규칙이 실제 제어환경과 비교해서 얼마나 적합한 지를 나타내고 있기 때문에 플랜트의 제어입력을 결정할 때 k번째 규칙의 영향을 전건부의 적합도로 나타낼 수 있고 플랜트의 제어입력을 결정할 때에 후건부의 값 v_k를 얼마나 반영해야 할 것인지를 알게 된다. 따라서 k번째 규칙의 적합도(fitness)를 ϕ_k라고 하면, 이 값은 다음과 같이 계산할 수 있다.

$$\phi_k = \frac{\mu_{E_k}(e)\ \Lambda\ \mu_{S_k}(s)\ \Lambda\ \mu_{C_k}(c)}{\sum_{k=1}^{M}\{\mu_{E_k}(e)\ \Lambda\ \mu_{S_k}(s)\ \Lambda\ \mu_{C_k}(c)\}} \tag{7}$$

여기서, $[\mu_{E_k}(e)\ \Lambda\ \mu_{S_k}(s)\ \Lambda\ \mu_{C_k}(c)] \approx \min[\mu_{E_k}(e), \mu_{S_k}(s), \mu_{C_k}(c)]$이다. 이렇게 구해진 적합도에 k번째 규칙의 후건부 값을 곱하면 k번째 규칙에 의한 제어입력량이 계산되고 마찬가지로 각각의 규칙에 대한 제어입력의 합을 계산하면 플랜트에 적용하게 될 최종적인 제어입력을 다음과 같이 구할 수 있다.

$$u = \sum_{k=1}^{M} \phi_k \times v_k \tag{8}$$

3. 학습 알고리즘

플랜트를 제어하기 위한 제어규칙의 학습은 지도학습(supervised learning)이므로 학습을 위한 목표가 필요하다. 따라서 제어목표를 제시하기 위해 Fig. 1과 같은 모델규범형 퍼지학습 제어기(model reference fuzzy control)를 사용하여 원하는 출력을 제시하고 이에 따라 제어규칙을 학습하고자 한다. 이 학습 제어기는 현재 적응제어에서 널리 사용하고 있는 모델규범형 제어기(model reference adaptive control)와 유사하여 먼저, 규범모델을 통하여 원하는 출력을 제시하고 이 출력과 실제 플랜트의 출력과의 차이를 퍼지제어규칙에 피드백하

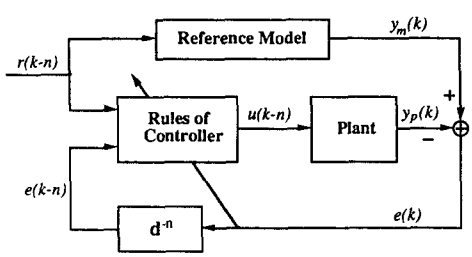

Fig. 1 Block diagram of the model based fuzzy control

여 과거에 사용했던 규칙에 대해 수정을 하도록 하는 것이다. 이때 시스템의 성능은 플랜트와 규범모델의 출력차이, 즉 에러의 함수로 표현될 수 있으므로 성능지수표를 사용하지 않고도 학습이 가능하게 된다. 따라서 이렇게 구성한 퍼지제어기의 학습원리는 다음과 같다. 즉, Fig. 1에서 처럼 n 스텝의 시간지연을 갖고 입출력 쌍으로 $\{u(k-n), y_p(k)\}$를 갖는 플랜트가 주어졌을 때 원하는 입출력 쌍 $\{r(k-n), y_m(k)\}$를 갖는 규범모델을 제시할 수 있다. 이때, 플랜트의 출력 $y_p(k)$가 모델의 출력 $y_m(k)$과 일치하도록 플랜트의 입력 $u(k-n)$를 결정하는 것이 제어기의 역할이다. 만일 $G(z)=1$인 규범모델을 사용한다면 플랜트에서의 출력지연 n 스텝을 보상하기 위해 제어기는 n 스텝을 앞서서 예측해야만 하고 이러한 경우에는 플랜트의 출력을 제어기의 입력으로 바로 사용할 수 있다. 플랜트의 제어규칙을 학습하는 경우, 초기의 제어규칙이 전혀 없다면 퍼지추론의 결과로써 제어입력은 영(0)이 되고 이에 따른 플랜트의 출력도 영(0)이 된다. 그러나 원하는 출력이 영(0)이 아닐 때에는 이를 근거로 제어규칙을 수정하게 되는데 여기서는 이러한 예상되는 에러를 줄이기 위해 에러 최소화법(error minimization method)을 사용하게 되는데 이 가운데서도 일반적으로 널리 사용되고 있는 기울기법(gradient approach)을 사용한다. 이 방법은 에러 및 기타 여러가지 제어목표를 포함하는 비용함수 J를 정의한 다음, J가 최소화되도록 플랜트의 입력값을 고치는 것이다. 이를 위해서는 J의 기울기를 구하여 음(negative)의 방향으로 규칙의 후건부를 수정해 나간다.

실제출력이 원하는 출력과 일치하기를 바라는 제어목표는 문제에 따라 다양한 비용함수로 표현할 수 있어서 에너지 최소화 문제인 경우에는 출력에러에 추가적으로 플랜트의 입력량을 고려하므로써 플랜트의 제어에 드는 에너지를 최소화할 수도 있고, 조정(regulation)문제의 경우에는 에러만을 고려할 때에 원치 않는 과잉응답(overshoot)이나 진동이 발생할 수 있으므로 이를 줄이기 위해서는 에러의 변화율을 추가로 고려하는 편이 좋다. 따라서 에러와 에러의 변화율을 동시에 최소화한다면 비용함수는 다음과 같이 나타낼 수 있다.

$$J(h)=\frac{\omega_e}{2}e(h)^2+\frac{\omega_c}{2}c(h)^2 \tag{9}$$

여기서, $e(h)=y_d(h)-y(h)$, $c(h)=e(h)-e(h-1)$이고, ω_e, ω_c : weighting constant이다.

이때 최적의 규칙을 구하기 위해서는 제어구간 전체에 대한 비용함수를 계산한 다음 이 값을 최소화하도록 퍼지규칙을 수정해 나가야 한다. 그러나 동적 시스템에서 현재의 입력은 과거의 출력에는 어떤 영향도 주지 않고 미래의 출력에만 영향을 주므로(properity of causality) 이 점을 고려한다면, k 스텝에서 투입한 입력은 그 이후의 모든 출력에 포함되어 있다고 볼 수 있으므로, 따라서 비용함수는 다음과 같은 학습법칙으로 수정되어야 한다.

$$\Delta v_i \propto -\frac{\partial J}{\partial v_i} \tag{10}$$

여기서, $J=\sum_{h=0}^{\infty}J(k+h)$

그러나 이렇게 계산할 경우 전체의 제어공정이 끝나기 전에는 규칙의 학습이 이루어 질 수 없으므로 온라인으로의 학습이 곤란해진다. 따라서 온라인 학습이 가능하도록 다음과 같이 제어입력 $u(k)$에 대해 가장 큰 영향을 받는 출력 $y(k+n)$에 대해서만 고려하도록 하자. 즉, 현재의 입력이 n 스텝이 지난 후의 출력에 가장 큰 영향을 준다고 가정한 다음, 이 출력에 대한 비용함수만 최소화하도록 퍼지제어의 규칙을 수정한다면 온라인 학습이 가능하게 된다. 이렇게 함으로써 식 (10)은 다음과 같은 간단한 식으로 나타내어질 수 있다.

$$\begin{aligned}\Delta v_i \propto & -\frac{\partial J(k+n)}{\partial v(k)}\frac{\partial u(k)}{\partial v_i}\\ =&\Big\{\omega_e\frac{\partial y(k+n)}{\partial u(k)}e(k+n)\\ &+\omega_c\frac{\partial\{y(k+n)-y(k+n-1)\}}{\partial u(k)}\\ &c(k+n)\Big\}\frac{\partial u(k)}{\partial v_i}\end{aligned} \tag{11}$$

또한 $\dot{y}(k+n)=\dfrac{y(k+n)-y(k+n-1)}{\Delta t}$이고, $\phi_i=\dfrac{\partial u(k)}{\partial v_i}$이므로

$$\begin{aligned}\Delta v_i \propto =&\Big\{\omega_e\frac{\partial y(k+n)}{\partial u(k)}e(k+n)+\omega_c\\ &\frac{\partial\{\dot{y}(k+n)}{\partial u(k)}\Delta t\ c(k+n)\Big\}\phi_i\end{aligned} \tag{12}$$

다중 입출력 시스템의 경우, $\dfrac{\partial y(k+n)}{\partial u(k)}$는 플랜트의 자코비안 매트릭스를 나타내고 $e(k+n)$은 플랜트의 입력과 같은 차수의 벡터를 나타내기 때문에 플랜트의 자코비안 메트릭스를 추정할 수 없으

면 학습이 곤란해 진다. 그러나 플랜트가 단일 입출력 시스템이라면 이 값은 목적함수를 최소화하는 방향의 기울기를 나타내므로 규칙의 학습은 다음의 식과 같이 간단히 이루어질 수 있다.

$$v_{i(new)}=v_{i(old)}+\Delta v_i \tag{13}$$

$$\Delta v_i=\{\eta_e e(k+n)+\eta_c C(k+n)\}\phi_i \tag{14}$$

여기서, 만일 $\frac{\partial y(k+n)}{\partial u(k)}$과 $\frac{\partial \dot{y}(k+n)}{\partial u(k)}$이 둘 다 양의 값을 갖는다면 η_e와 η_c는 양의 값으로 선택되는 학습률 상수이다. 따라서 $\frac{\partial y(k+n)}{\partial u(k)}$과 $\frac{\partial \dot{y}(k+n)}{\partial u(k)}$ 둘 모두가 양의 값을 가질수 있도록 학습지연 값 n을 적절히 선택할 수 있다면 식 (14)를 사용해서 퍼지규칙을 간단히 학습할 수 있게 된다.

〈정리 1〉 학습지연량 : n

단위 계단입력 $\Delta u(k)$에 대한 플랜트의 출력을 $\Delta y(k+n)$, $n=1, 2, \cdots$ 이라고 가정하자. 그리고 $\frac{\Delta y(k+n)}{\Delta u(k)}>0$을 만족하는 n의 최소값을 n_1이라고 하고 $\frac{\Delta y(k+n)}{\Delta u(k)}$를 최대화하는 n의 값을 n_2고 할 때, 만일 $n_1<n<n_2$에 대하여 $\frac{\Delta y(k+n)}{\Delta u(k)}$이 단조증가함수라면 이것을 만족하는 임의의 n 값에 대하여 $\frac{\partial y(k+n)}{\partial u(k)}>0$과 $\frac{\partial \dot{y}(k+n)}{\partial u(k)}>0$이 항상 동시에 만족된다.

이 〈정리 1〉의 증명은 부록에 유첨하였다. 이 정리에서 알 수 있듯이 수렴을 위한 학습지연 값의 선택범위는 상당히 넓다. 또한 학습지연 스텝 n은 플랜트의 출력지연시간과 단위샘플링 시간에 대해 밀접한 관련이 있다. 특히 유연 외팔보와 같은 비최소위상(non-minimum phase) 플랜트에 있어서는 이러한 미분치를 양의 값으로 만들기 위해 충분히 큰 학습지연을 고려할 필요가 있다.[5]

4. 시뮬레이션

본 논문에서 제시한 학습 알고리즘의 효과를 살펴보기 위하여 1 radian(57°)정도 쓰러져 있는 막대기를 거꾸로 세우는 막대차 문제에 대해 적용해 보았다. 제어의 목표는 적절한 수평력을 막대차에 가함으로써 막대기를 수직으로 세워 균형을 유지하는 것이다. 따라서 이 문제에 대한 퍼지제어의 규칙을 학습에 의해 구성할 때, 본 논문에서 제시한 목적함수와 학습지연량 n의 의미를 알아 보고자 한다. 문제의 막대차는 마찰이 없는 트랙을 따라 한 방향으로만 움직이고 막대기는 마찰이 없는 힌지에 의해 막대차에 연결되어 트랙에 대해 수직으로 세워지는 것이다. 막대차의 각도 동역학을 묘사하는 미분방정식은 다음과 같이 주어진다.[7]

$$\ddot{\theta}(t)=\frac{g\sin\theta(t)-\frac{\cos\theta(t)}{m_p+m_c}\{m_p l\dot{\theta}(t)^2\sin\theta(t)+F(t)\}}{\frac{4}{3}l-\frac{m_p l\cos^2\theta(t)}{m_p+m_c}} \tag{15}$$

여기서, 각각의 변수는 다음과 같이 정의된다.

$\theta(t)$: 수직축에 대한 막대기의 각도
$F(t)$: 막대차에 작용하는 힘
$2l$: 막대기의 길이(1.0 m)
m_p : 막대기의 질량
m_c : 차의 질량
g : 중력 가속도(9.8 m/sec^2)

이때 각속도와 각도는 위의 식 (15)를 수치적분하므로써 구한다. 그리고 이러한 계산은 단지 시뮬레이션을 위해 임의의 제어입력에 대한 플랜트의 출력을 알아보기 위해 사용하는 식일뿐 본 퍼지제어에서는 전혀 이용하지 않는 식이다.

4.1 시뮬레이션의 조건

퍼지 규칙베이스를 학습하기에 앞서 먼저 설계변수를 결정해야 하는데 설계변수로는 각각의 입력변수에 대한 퍼지집합의 개수와 멤버쉽함수의 형상, 학습지연시간 그리고 비용함수의 가중치 상수 등이 있다. 이 가운데에서 입력변수의 양자화(quantization)는 규칙의 개수를 결정짓게 되고 또한 학습시간과 출력의 정확도에 큰 영향을 주기 때문에 중요하지만 양자화를 효과적으로 하기란 상당히 어려운 일이다.[2] 또, 이 문제에서 플랜트는 초기 각도가 수직축에 대해 1 radian이고 초기 각속도가 제로이므로 학습이 이루어지기도 전에 쓰러져 버릴 수 있으므로(unstable system) 초기규칙을 주지 않으면 안된다. 따라서 상식적인 측면에서의 간단한 규칙을 제공한다고 생각해 보면 쓰러지는 방향으로 막대차에 힘을 가하면 쓰러지지 않게 된다는 것을 알 수 있다. 따라서 우리가 구해야 하는 규칙도 단순하게 에러(error)와 에러의 변화율(change in error)에 대해 음의 값(negative), 제로(zero), 양의 값(positive)으로 각각 3가지씩의 상황만을 고려하므로써 2차원 배열로 9개의 규칙만으로 구성하

고 초기규칙으로는 4개에 대해서만 다음과 같은 출력 값을 제공하도록 미리 설정했다.

규칙 1, 2 : 만일 에러가 양이고 에러변화율이 양이거나 영이면, 막대차에 $-F_{max}/2$라는 힘을 가하라.

규칙 3, 4 : 만일 에러가 음이고 에러변화율이 음이거나 영이면, 막대차에 $+F_{max}/2$라는 힘을 가하라.

규칙 5~9 : 그 외의 경우에는, 어떠한 힘도 막대차에 가하지 않는다.

어떤 멤버쉽함수를 사용할 것인가 하는 문제에 있어서 Chung & Oh[5]는 특히, 추종문제(tracking problem)의 경우에서는 가우스함수를 사용한 경우가 삼각형(triangular) 함수나 종형(bell-shape)의 함수를 사용한 것보다 조금 더 우수한 특성을 가진다는 것을 보인 바 있지만 막대차와 같은 조정문제(regulation problem)에서는 멤버쉽함수로 어떤 것을 사용해도 큰 차이가 나지 않는다. 다만 임의의 영역을 퍼지집합으로 할당하는 문제에서 삼각형이나 종형의 경우에는 설계를 잘못하게 되면 함수의 특성상 어느 집합에도 속하지 않는 영역이 존재할 수 있는 반면, 가우스함수의 경우는 항상 전 영역을 포함하는 강건성(robustness)이 있다. 따라서 본 논문에서 사용한 멤버쉽함수는 에러와 에러의 변화율에 대해서 Fig. 2에서 보여주는 바와 같이 각각 positive, zero 그리고 negative라는 세가지 집합으로 구성하고 각각의 퍼지집합은 50% 정도의 확률값에서 서로 겹치도록 했다. 그리고 에러에 대해서는 radian 단위를 그대로 사용해서 선형양자화했고 에러의 변화율에 대해서는 에러의 10분의 1로 역시 선형양자화했다. 이렇게 만들어진 멤버쉽함수와 초기의 규칙만으로 시스템을 제어했을 때 막대기의 각도궤적을 Fig. 3에서 보여주고 있다. 그림의 결과에서 보는 바와 같이 퍼지제어에서는 단지 이런 몇 개의 제어규칙만으로도 4초 정도만 허락한다면 막대차가 막대기를 세우는 데에는 전혀 문제가 없음을 알 수 있다. 그러나 보다 더 빨리(만일 1초 이내에) 정상상태에 이르기를 원한다던지 혹은 과잉응답이 없어야 한다던지 하는 문제가 제시된다면 주어진 규칙을 수정해야 하는 문제가 발생된다. 따라서 이러한 문제를 해결하기 위하여 본 논문에서는 규범모델을 통한 규칙의 학습을 제안하고자 하며 학습에 대한 조정변수로 학습지연시간과 목적함수를 추가적으로 고려하므로써 보다 더 정확한

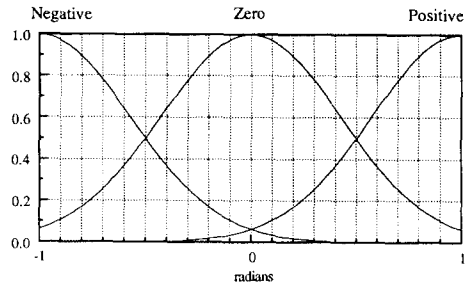

Fig. 2 Membership function of error and change in error

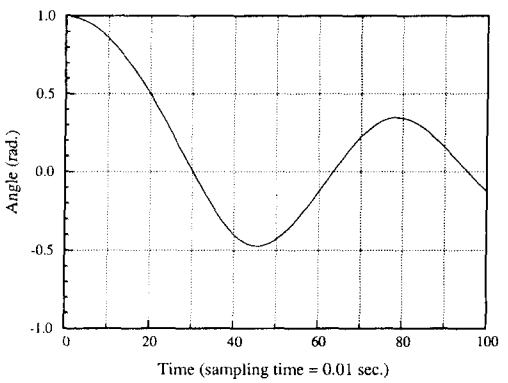

Fig. 3 Cart pole control by initial rules

제어규칙을 얻고자 한다. 막대차 문제의 경우는 단위입력에 대해 양의 출력값이 2 스텝 이후에 나타나므로 학습지연은 적어도 2보다 커야 하고 이때 학습지연 값의 범위는 〈정리 1〉에 따르면 $n_1=2$이고 $n_2=\infty$임을 알 수 있다. 마지막으로 플랜트의 관성력으로 인해 원치 않는 과잉응답이 예상되므로 비용함수에서는 에러와 에러의 변화량을 동시에 고려한다면 과잉응답을 피하는 보다 좋은 결과를 얻을 수 있을 것이라고 예상할 수 있다.

4.2 시뮬레이션의 내용

막대차의 운동방정식에서 볼때, 어떤 힘이 작용한 뒤 2 스텝 후에야 비로소 막대기는 어떤 각도의 변위를 갖게 된다. 따라서 학습지연은 〈정리 1〉에 의하면 최소한 2 스텝을 필요로 하므로 본 논문에서 제시한 학습 알고리즘의 효용성을 알아 보기 위하여 먼저, 학습지연을 2 스텝으로 고정시킨 상황에서 목적함수의 에러부분과 에러의 변화량 부분을 변화시켜 가면서 학습되는 규칙의 특징을 살펴 보았다. 그리고 학습을 지도하는 규범모델로는 $G(z)$

=1을 사용했고 목적함수에서 에러의 변화량을 전혀 고려하지 않고 학습한 결과가 Fig. 4의 점선이다. 예상한 바와 같이 동적 시스템이 규범모델에서 제시하는 스텝출력을 곧 바로 따라가기란 거의 불가능하다. 따라서 이러한 경우, 지연시간, 상승시간, 과잉출력(overshoot)량과 정착시간(settling time)등을 잘 고려하여 원하는 성능에 가장 가까운 제어규칙을 획득할 필요가 있다. 이와 같은 영점조정(zero regulation) 문제에서 에러만을 고려한 목적함수를 사용한 경우 상당히 큰 과잉응답이 나타나므로 이 과잉응답을 줄이기 위해서는 목적함수에 에러의 변화율을 동시에 고려하는 것이 필요하므로 에러의 변화율을 포함한 목적함수에서 이에 대한 비중을 점점 증가시켜 보았다. 에러에 비하여 50%, 즉 전체 목적함수의 35% 가량을 차지하도록 에러의 변화율에 대한 비중을 증가시킨 경우가 Fig. 4의 실선과 같은 출력을 나타내는 퍼지제어규칙이었고 계속해서 이값의 비중을 높여 에러와

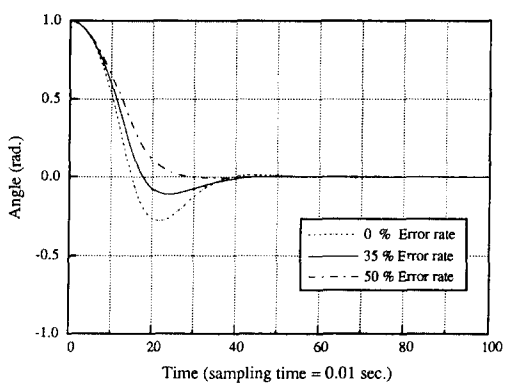

Fig. 4 Cart pole control (according error rate)

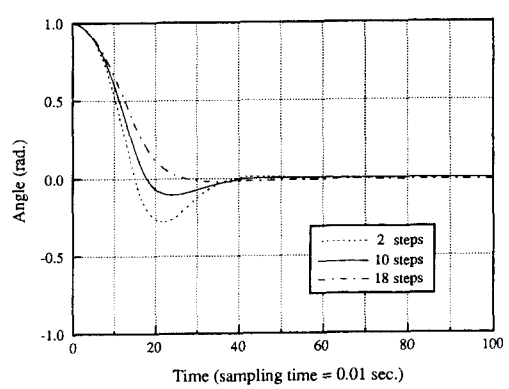

Fig. 5 Cart pole control (according learning delay)

에러의 변화율에 대한 비율이 거의 같게 되었을 때에는 Fig. 4의 일점쇄선과 같은 과잉응답이 전혀 없는 제어규칙을 얻을 수 있었고 학습이후에는 정상상태에 이르기까지 걸리는 시간은 대부분이 0.3~0.5초 이내였다.

다음에는 목적함수에서 에러의 변화율은 제외시킨채 에러량만을 고려하면서 학습지연의 효과를 관찰해 보았다. 학습지연 n이 취할 수 있는 범위는 $n_1=2$이고 $n_2=\infty$이므로 최소 2 스텝에서 무한대까지 이다. 그러나 학습지연을 무한정 늘리게 되면 빠른 응답을 기대하기가 어렵기 때문에 플랜트의 수렴시간을 고려하여 적절히 그 범위를 제한해서 생각하는 것이 바람직하다. 학습지연으로 2 스텝만을 지연시킨 경우는 Fig. 5의 점선인데 Fig. 4의 점선의 결과와 동일한 경우이므로 그대로 사용했다. 그리고 나서 학습지연을 계속 늘렸는데 10 스텝 만큼을 지연시킨 결과는 Fig. 5의 실선과 같은 제어응답을 나타났는데 이것은 Fig. 4의 실선인 목적함수에서 에러의 변화량을 35% 가량 고려한 결과와 유사한 응답이라는 것을 알 수 있다. 그리고 학습지연이 18 스텝인 경우에는 Fig. 5의 일점쇄선에서 보는 바와 같이 목적함수에서 에러의 변화량을 50% 가량 고려한 결과와 유사하게 나타남을 알 수 있었다.

4.3 시뮬레이션의 결과

원하는 제어규칙을 얻기 위하여 규범모델을 사용하였고 실제 플랜트와 원하는 출력값의 차이를 목적함수로 두고 이것을 최소화하는 퍼지규칙을 구했다. 이때 목적함수에는 출력에러의 변화량을 추가적으로 고려할 수 있었고 이것을 얼마나 고려했는가에 따라 학습된 제어규칙과 플랜트의 응답결과에 큰 영향을 주었고 이들 값은 플랜트에 대해 댐핑(damping)효과를 나타낸다는 것을 알게 되었다. 단일 입출력 시스템의 경우에는 플랜트의 모델링을 구하지 않고도 출력의 에러를 입력의 에러로 변환하여 학습하는 것이 가능한데 이 경우에 어떤 입력이 출력으로 변환되어 나타나기까지 걸리는 출력지연스텝이 학습에서는 학습지연으로써 고려되어야 하는 아주 중요한 요인이라는 것을 알았다. 그리고 지연스텝의 값에 대해서도, 이 값을 크게 고려하면 할수록 출력의 과잉응답이 줄어들고 댐핑의 효과가 많이 나타난다는 것을 알 수 있었다. 본 시뮬레이션의 결과에서 학습지연량의 증가는 학습시에 고려

Table 1 Rule table before learning in cart pole change in error (c)

		Negative	Zero	Positive
Error (e)	Negative	50	50	0
	Zero	0	0	0
	Positive	0	-50	-50

Table 2 Rule table after learning in cart pole change in error (c)

		Negative	Zero	Positive
Error (e)	Negative	100	100	14
	Zero	78	0	-78
	Positive	-14	-100	-100

한 에러의 변화율과 거의 같은 물리적인 의미를 갖고 있다는 것을 알 수 있었다. 따라서, 에러의 변화율과 학습지연을 동시에 고려한 결과는 본 논문에서 언급하지는 않았지만 어느 한 쪽을 더 많이 고려한 것과 같은 결과로 예상할 수 있다. Table 1과 2에서는 각각 학습 전과 학습 후의 퍼지규칙을 보여 주고 있다(규칙의 대칭성을 주기 위해서 초기 위치를 ±1 radian에서 교대로 학습하였다). 본 논문에서는 규칙의 학습에서 멤버쉽 함수의 조정과 같은 여러가지의 다른 퍼지학습법[8]은 전혀 사용하지 않았기 때문에 학습전에 비해 달라진 것은 Table 2에서 보는 바와 같이 규칙의 후건부 값 뿐이다. 따라서 학습 전에는 Table 1에서 보는 것과 같은 직관에 의한 초기규칙으로 Fig. 3에서 보여주는 출력밖에 얻지 못했지만 학습 후에는 Fig. 5의 실선(목적함수로 에러만을 고려하고 10 스텝의 학습지연)과 같은 좋은 결과를 얻을 수 있었다. 어떻게 해서 불과 9개의 규칙만으로 Fig. 4나 5의 실선과 같은 정교한 제어가 가능한 것일까? 본 논문의 알고리즘에 대한 이해를 돕기 위해 학습 후의 규칙을 Fig. 6에서 보여주는 바와 같이 3차원 형상으로 나타내었고 제어변수로서는 에러와 에러의 변화율만을 고려하면서 임의의 에러와 에러의 변화율에대해 해당되는 플랜트의 제어입력을 다양하게 추론할 수 있음을 보여주고 있다. 즉, 이 그림의 추론곡면은 비록 9개의 규칙만을 사용했으면서도 비슷한 상황에 대해 결코 같지 않은 조금씩은 다른 결과를

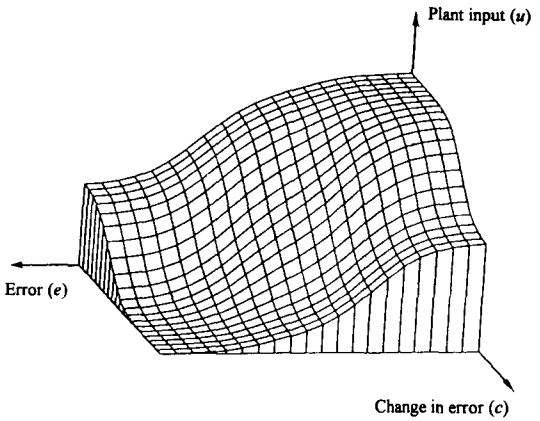

Fig. 6 Shape of rules after the learning cart pole

제공하고 있음을 알 수 있다. 따라서 이 그림을 통해서 퍼지제어에서는 단지 몇 개의 규칙만으로도 굉장히 다양한 결과를 추론할 수 있음을 보여주고 있는 것이다. 여기에서는 추론결과를 3차원 입체형상으로 나타내기 위하여 퍼지 PD 제어기만을 사용하였으므로 Fig. 5의 18 스텝 지연에 의한 학습제어의 경우에서 보는 바와 같이 정상상태(steady state)에서의 에러가 나타나는 경우에는 퍼지 I(에러의 적분) 제어기를 동시에 고려하므로써 정상상태에서의 에러를 제거할 수가 있다.

5. 결 론

제안한 학습 알고리즘은 원하는 제어규칙을 얻기 위하여 기존의 방법과는 달리 성능지수표를 사용하지 않고 규범모델을 사용하므로써 원하는 출력과 획득된 제어규칙을 직접 비교하도록 했다. 그리고 목적함수에는 실제 플랜트와 원하는 출력값과의 차이인 에러와 에러의 변화율을 동시에 고려하였고 이것을 최소화하도록 하는 퍼지학습 제어법을 제안했다. 이때, 단일 입출력 플랜트의 경우는 단순히 n 스텝 지연의 단조(monotonic)함수로 모델링될 수 있어서 지연스텝의 결정만으로 학습에서의 수렴성까지 보장된다는 사실을 증명하였다. 따라서 플랜트의 동특성 방정식을 잘 모르는 경우에도 목적함수나 지연스텝을 적절히 잘 선택함으로써 과잉응답을 줄이고 빠른 시간내에 수렴하는 제어규칙을 학습에 의해 얻을 수 있었다. 특히 불안정 시스템에 대한 학습제어의 경우는 초기의 규칙이 전혀 없다면 학습이 전개될 수가 없으므로 먼저, 극히 상

식적인 선에서의 초기 규칙을 제공하므로써 안정한 시스템으로 만든 후에 학습을 행했을 때에 원하는 제어 규칙을 잘 얻을 수 있었다. 결론적으로, 본 논문에서는 퍼지제어의 규칙을 얻기가 어렵거나 이미 획득된 규칙의 개선을 원할 때에 원하는 시스템 출력에 대해 최적의 규칙을 얻을 수 있는 퍼지학습법임을 보였다.

참고문헌

(1) Procyk, T. J. and Mamdani, E. H., 1979, "A Linguistic Self-Organizing Process Controller," *Automatica*, Vol. 15, No. 1, pp. 15~30.

(2) Takagi, T. and Sugeno, M., 1985, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 15, , No. 1, pp. 116~132.

(3) Shao, S., 1988, "Fuzzy Self-Organizing Controller and Its Application for Dynamic Processes," *Fuzzy Sets and Systems*, Vol. 26, pp. 151~164.

(4) Zang, B. S. and Edmunds, J. M., 1992, "Self-Organizing Fuzzy Logic Controller," *IEE Proc.-D*, Vol. 139, No. 5, pp. 460~464.

(5) Chung, B. M. and Oh, J. H., 1993, "Control of Dynamic Systems Using Fuzzy Learning Algorithm," *Fuzzy Sets and Systems*, Vol. 59, No. 1, pp. 1~14.

(6) Lee, C. C., 1990, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I, II," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 20, No. 2, pp. 404~435.

(7) Barto, A., Sutton, R. and Anderson, C., 1983, "Neuron-Like Adaptive Elements that Can Solve Difficult Learning Control Problems," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 13, No. 5, pp. 834~846.

(8) Chung, B. M. and Oh, J. H., 1993, "Autotuning Method of Membership Function in a Fuzzy Learning Control," *Journal of Intelligent & Fuzzy Systems*, Vol. 1, No. 4, pp. 335~349

부 록

〈정리 1의 증명〉

먼저, n_1이 $\frac{\Delta y(k+n)}{\Delta u(k)}>0$을 만족하는 최소값이므로

$\frac{\Delta y(k+n_1-1)}{\Delta u(k)}\leq 0$이고,

$\frac{\Delta y(k+n_1)}{\Delta u(k)}-\frac{\Delta y(k+n_1-1)}{\Delta u(k)}>0$이다.

따라서, $\frac{\Delta \dot{y}(k+n_1)}{\Delta u(k)}>0 \rightarrow \frac{\partial \dot{y}(k+n_1)}{\partial u(k)}>0$

다음으로,

$\frac{\Delta y(k+n_2)}{\Delta u(k)}=\max\left\{\frac{\Delta y(k+n)}{\Delta u(k)}\right\}$이기 때문에

$\frac{\Delta y(k+n_2)}{\Delta u(k)}-\frac{\Delta y(k+n_2-1)}{\Delta u(k)}>0$

마찬가지로, $\frac{\Delta \dot{y}(k+n_2)}{\Delta u(k)}>0 \rightarrow \frac{\partial \dot{y}(k+n_2)}{\partial u(k)}>0$

결국,

$n=n_1, \cdots, n_2$에 대하여 $\frac{\Delta y(k+n)}{\Delta u(k)}$는 단조증가함수이므로

$0<\frac{\Delta \dot{y}(k+n_1)}{\Delta u(k)}< \cdots <\frac{\Delta \dot{y}(k+n_2)}{\Delta u(k)}$이다.

따라서

$n=n_1, \cdots, n_2$를 만족하는 모든 n에 대하여 $\frac{\partial y(k+n)}{\partial u(k)}>0$과 $\frac{\partial \dot{y}(k+n)}{\partial u(k)}>0$이다.