유전 프로그래밍을 이용한 규칙 기반 제어기의 설계와 퍼지로직 제어기로의 응용

Design of a Rule Based Controller using Genetic Programming and Its Application to Fuzzy Logic Controller

정 일 권, 이 주 장
(Il-Kwon Jeong and Ju-Jang Lee)

Abstract : Evolutionary computation techniques can solve search problems using simulated evolution based on the 'survival of the fittest'. Recently, the genetic programming (GP) which evolves computer programs using the genetic algorithm was introduced. In this paper, the genetic programming technique is used in order to design a rule based controller consisting of condition-action rules for an unknown system. No a priori knowledge about the structure of the controller is needed. Representation of a solution, functions and terminals in GP are analyzed, and a method of constructing a fuzzy logic controller using the obtained rule based controller is described. A simulation example using a nonlinear system shows the validity and efficiency of the proposed method.

Keywords : genetic programming, rule based, fuzzy, control

I. 서론

1970년대에 개발되어 최적화 문제를 푸는 방법의 하나로 요즘 각광받고 있는 진화연산 (Evolutionary Computation; EC) 이란 컴퓨터를 이용하여 적자생존을 토대로 생물학적 진화를 모의 실험하여 탐색문제의 해를 찾는데 이용하는 기법이다. D. B. Fogel에 의하면 진화연산은 크게 유전 알고리즘 (Genetic Algorithm; GA) 과 진화 알고리즘 (Evolutionary Algorithm; EA) 으로 나눌 수 있다[4]. 진화 알고리즘은 다시 진화 기법 (Evolution Strategy; ES) 과 진화 프로그래밍 (Evolutionary Programming; EP) 으로 나뉘어 진다. 이들은 해를 표현하는 방법, 구체적인 해의 변화 방법, 해의 선택방법에 따라 약간씩 차이가 있다. 하지만 세가지 방법 모두 해의 집단을 다루는 탐색방법이고 무작위 변화를 적용하며 해의 선택 과정이 존재한다는 공통점이 있다. 이들 방법은 공통적으로 다음과 같은 동작을 하게 된다. 후보 해에 해당하는 각 개체들의 집단이 임의로 초기화된 후 선택, 돌연변이, 재결합과 같은 확률적 유전 연산자를 적절히 거쳐 탐색 공간상의 더 나은 공간으로 진화한다. 유전 알고리즘은 조합 최적화 문제에, 진화 프로그래밍 기법은 연속 최적화 문제에 잘 동작하는 특성이 있다. 제어 분야에선 특히 유전 알고리즘이 주로 적용되어 왔다. 대표적인 예로 시스템 동정화[5], 신경회로망 구조 그리고(또는) 가중치의 최적화[6][11] 그리고, 퍼지로직 제어기의 규칙표 (rule base) 그리고(또는) 소속함수의 최적화[7][8]를 들 수 있다 한편, 유전 프로그래밍 (Genetic Programming;GP)이란 방법이 최근 J. R. Koza 에 의해 소개되었다[1]. 이는 컴퓨터 프로그램을 유전 알고리즘을 이용하여 진화시키는 방법으로서 유전 알고리즘의 확장된 형태라고 볼 수 있다. 아직 많이 적용되고 있지는 않지만 유전 프로그래밍 기법의 잠재적인 활용 범위는 매우 넓다고 할 수 있다. 해를 찾기 위해 각 개체의 부호화 과정을 거쳐야 하는 유전 알고리즘은 풀고자 하는 문제에 따라 부호화 과정을 새로 정의해주어야 하는 반면에 유전 프로그래밍을 사용하면 부호화 과정을 간단하게 할 수 있게되고 문제에 따라선 필요 없을 수도 있다. 해의 길이에 제한이 없기 때문이다.

제어기의 설계에 시스템에 대한 정보를 필요로 하지 않으며, 또 설계된 제어기가 규칙 기반 제어기 더 나아가 퍼지로직 제어기인 방법이 있다면 제어기 설계뿐 아니라 설계된 제어기로부터 시스템의 특성 및 제어방법 등을 추출할 수 있게되어 매우 유용할 것이다. 본 논문에서는 유전 프로그래밍을 사용하여 미지의 시스템에 대하여 제어기에 대한 구체적인 정보 없이 if-then과 같은 조건 연산자로 이루어진 규칙 기반 제어기를 설계한다. 또, 규칙 기반 제어기가 퍼지로직 제어기와 유사한 점에 착안하여, 설계된 규칙 기반 제어기를 이용하여 퍼지로직 제어기를 보다 용이하게 설계할 수 있음을 보인다. 하지만, 유전 프로그래밍과 같은 집단 기반 탐색 기법의 특성상 미지의 실제 시스템에 적용하기 위해서는 무수한 실험, 초기치 설정의 어려움, 시스템 손상을 가져올 수 있는 제어 입력의 가능성 때문에 어려움이 따른다. 따라서, 본 논문에서 제안되는 방법은 시스템의 동적 방정식은 알고 있으나 제어기 설계가 어려운 복잡한 비선형 시스템에 대하여 모의 실험을 통하여 퍼지로직 제어기를 구성하고 이를 실제 시스템에 적용하고 조정하여 사용할

접수일자 : 1998. 5. 19., 수정완료 : 1998. 9. 21.
정일권, 이주장 : 한국과학기술원 전기및전자공학과
※ 이 논문은 1996년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었습니다.

때 실용적인 방법이 될 수 있을 것이다. 기존에 진화연산을 이용한 퍼지로직 제어기의 설계는 퍼지로직 제어기의 복잡성과 진화연산의 많은 계산으로 인하여 많은 계산량과 시간을 요하였으나 제안된 방법은 간단한 규칙기반 제어기를 먼저 설계한 후 그로부터 퍼지로직 제어기를 유도하기 때문에 계산량이 상대적으로 작다. II장에서는 유전프로그래밍 기법에 대해 간략히 설명하고 III장에서는 유전프로그래밍을 이용한 규칙기반 제어기 및 퍼지로직 제어기의 설계 방법을 보인다. IV장에서는 모의실험을 통해 제안된 방법의 타당성을 확인한다.

II. 유전 프로그래밍

진화연산 방법에는 크게 유전 알고리즘, 진화 기법, 진화 프로그래밍의 세가지 부류가 있다. 이중 유전 알고리즘은 자연계의 적자생존과 유전학에 근거를 둔 탐색 알고리즘으로서 다음과 같은 진화 과정을 가진다[2][3]. 먼저 문제를 해결하기 위한 초기해들로 구성된 첫 세대를 임의로 선정한다. 여기서 각 해들은 실제 해가 아닌 부호화된 해이며 이는 개체로 간주되고 개체들의 집합은 집단이라 부른다. 이 개체들은 성능 평가 함수(목적함수)를 만족시키는 정도에 따라 적합도를 부여받게 되고, 이는 다음 세대에 부모가 될 자격이 있는지의 심사기준이 된다. 적합도가 높은 개체일수록 부모가 될 확률이 높으며, 선택된 부모들은 교차변이와 돌연변이와 같은 유전 연산자에 의해 자손들을 형성하게 된다. 이들은 다시 성능 평가 함수를 통해 적합도를 부여받게 되고, 새로운 부모를 형성하게 된다. 이와 같은 과정을 수 세대 반복하여 원하는 개체가 나올 수 있도록 한다. 유전 알고리즘은 해를 부호화 하는 과정이 필요하게 되고 보통 이진수를 이용한다. 따라서, 이진수로 해의 표현이 용이한 조합 최적화 문제에 유용하다.

J. R. Koza에 의해서 소개된 유전 프로그래밍 기법은 유전 알고리즘에 뿌리를 두고 있다. 유전 알고리즘과는 달리 부호화된 해 대신 함수들로 이루어진 구조를 사용하고 해의 길이에 제한이 없다는 점에 차이가 있고, 나머지 과정은 유전 알고리즘과 매우 유사하다. 컴퓨터 프로그램을 진화시키는 알고리즘인 유전 프로그래밍은 다음의 세 과정으로 이루어진다.

1) 함수와 터미널들의 무작위 조합으로 이루어진 초기해(컴퓨터 프로그램)를 생성한다.
2) 종료 조건이 만족될 때까지 다음 과정을 반복한다.
   a. 집단내의 각 개체를 실행시켜 얼마나 문제를 잘 풀었나에 따라 적합도를 부여한다.
   b. 재생산, 교차변이 그리고 돌연변이와 같은 유전학적 연산자를 유전 알고리즘과 같은 방법으로 적용하여 새로운 컴퓨터 프로그램들의 집단을 구성한다.
3) 알고리즘이 종료될 때까지 생성되었던 프로그램 중에서 가장 성능이 뛰어난 개체가 유전 프로그래밍의 결과가 된다.

위와 같은 유전 프로그래밍을 어떤 문제에 적용하기 위해선 다음의 5가지를 먼저 행해야 한다. 터미널의 선택, 함수의 선택, 적합도 함수 결정, 유전 프로그래밍의 변수 설정, 그리고 종료 조건 결정이 그것이다.

III. 유전프로그래밍을 이용한 규칙 기반 제어기의 설계 및 퍼지로직 제어기의설계

1. 함수와 터미널 집합

유전 프로그래밍은 유전 알고리즘과는 달리 해의 표현에 함수와 터미널들로 이루어진 구조를 사용한다. 유전 프로그래밍에 사용될 수 있는 함수들은 다음과 같이 분류할 수 있다. 산술 연산자 (+, -, ×, ÷ 등), 수학 함수 (sin, cos, exp, log 등), 논리 연산자 (and, or, not 등), 조건 연산자 (if-then-else 등), 반복 유도 함수(do-until 등), 재귀 함수 등이다. 터미널 집합에는 주로 대상 시스템의 상태 변수 등이 포함된다.

위에 열거한 모든 원소들로 이루어진 구조를 사용하여 해를 표현할 수 있겠지만 필요 이상으로 알고리즘이 복잡해질 우려가 있다. 따라서, 주어진 문제에 따라 어떤 함수를 선택할 것인가 하는 문제와 터미널 집합을 어떻게 구할 것인가 하는 문제가 생긴다. 예를 들어 규칙 기반 제어기의 구성에 조건 연산자는 필수적일 것이다. 터미널 집합은 주로 상태변수를 그 원소로 하는데, 상태변수를 모를 때는 측정 가능한 파라미터들을 활용하여 유전 프로그래밍을 구성해도 무방할 것이다. 측정한 파라미터가 상태변수의 함수일 경우 유전 프로그래밍이 상태변수를 사용했을 때와 같은 제어기를 구성할 가능성이 있기 때문이다.

2. 유전 프로그래밍을 이용한 규칙 기반 제어기의 설계 구현

앞 단계에서 해의 표현에 쓸 함수와 터미널들의 집합을 구했다 하더라도 해의 구조를 어떻게 할 것인가 하는 문제가 생기고 이는 또한 프로그램 코딩 문제를 발생시킨다. 자세한 방법은 IV장에서 모의 실험과 함께 자세하게 다루도록 한다. 일반적으로 유전 프로그래밍에서의 해는 그림 1과 같은 트리 (tree) 구조로 표현할 수 있다. 이 트리 정보를 나타내기 위해서는 리스트 (list) 가 흔히 사용된다. 경우에 따라서는 배열로 표현할 수도 있을 것이다. 리스트를 가장 잘 다루기 위해서는 유전 프로그래밍의 코딩을 인공지능 언어인 LISP 언어로 하는 게 이상적일 것이다. 본 논문에서도 LISP 으로 구현하였다.

유전 프로그래밍은 유전 알고리즘과는 달리 각 개체마다 크기가 다르므로, 교차변이와 돌연변이 연산자를 문제에 맞게 잘 정의해 주어야 하지만, 트리구조를 이용하여 해를 표현하면 노드를 단위로 이루어지는 교차변이는 그림 2의 예처럼 항상 유효한 결과를 낳게된다.

3. 퍼지로직 제어기로의 변환

유전 프로그래밍에 의해서 얻어진 규칙 기반 제어기는 퍼지 제어기와 유사한 형태를 지닌다. 이에 착안하여, 설계된 규칙 기반 제어기를 퍼지로직 제어기로 변환할 수 있다. 규칙 기반 제어기를 분석하면 조건문들에 의하여 분할된 상태 공간의 형태와 각 공간에서의 제어 값을 알 수 있고, 이로부터 퍼지로직 제어기의 규칙표 (rule base) 와 입력 및 출력 변수의 언어 변수를 유도해 낼 수 있기 때문이다.

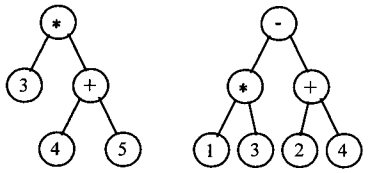

그림 1. (3 * (4 + 5)) 와 ((1 * 3) - (2 + 4)) 의 트리 구조 표시.
Fig. 1. (3 * (4 + 5)) and ((1 * 3) - (2 + 4)) represented in the tree structure.

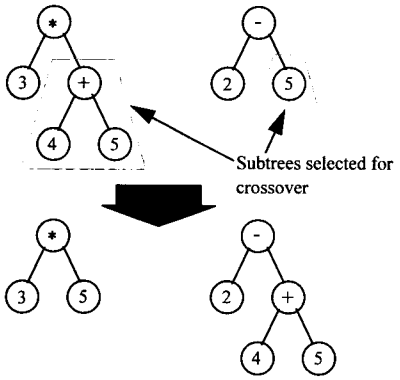

그림 2. 트리 구조로 표현된 해의 교차 변이.
Fig. 2. Crossover in the tree structure.

IV. 모의 실험

1. 역진자 시스템

가장 많이 사용되는 비선형 시스템 중의 하나인 역진자 시스템을 제안된 방법의 적용 대상으로 사용한다 (그림 3). 신경망 또는 퍼지로직 제어기 등을 사용하여 성공적으로 제어할 수 있음이 이미 알려져 있지만, 본 논문에서 제안된 방법의 적용 및 성능평가용으로 사용하며 역진자 시스템 방정식과 같은 시스템에 대한 사전 지식이 없는 것으로 가정한다.

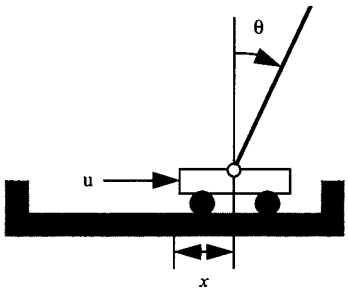

그림 3. 역진자 시스템.
Fig. 3. Cart-pole system.

본 모의 실험에서는 나중에 설명할 상태공간에서의 규칙 기반 제어기의 분석에 대한 설명의 용이성을 위하여 역진자의 기울어진 각도만을 제어 대상으로 삼는다. 제어 목적은 (7)의 초기조건에 대하여 진자를 쓰러지지 않도록 제어하는 것이다. 수레의 움직임을 제외한 진자의 각만에 관계된 시스템 방정식은 다음과 같다.

$$\dot{\theta} = \frac{d\theta}{dt} \tag{1}$$

$$\ddot{\theta} = \frac{(M+m)g\sin\theta - \cos\theta(u + ml\,\dot{\theta}^2\sin\theta)}{\frac{4}{3}(M+m)l - ml\cos^2\theta} \tag{2}$$

여기에서, 수레의 질량 M=0.4kg, 진자의 질량 m=0.1kg, 중력가속도 g=9.8m/s2, 그리고, 진자의 길이의 절반인 l=0.5m 이다.

2. 규칙기반 제어기의 설계

역진자 시스템의 터미널 집합에는 상태변수인 진자의 각변위와 이의 시간에 대한 미분 값이 포함될 수 있다. 이들을 이용하면 상태 궤환 제어가 가능하게 된다. 하지만, 본 논문에서는 규칙 기반 제어기의 설계후 퍼지로직 제어기로의 확장을 염두에 두고 있으므로 약간 다른 방식을 적용해야 한다. 그리고, 일반적인 조건-행동 규칙들로 이루어진 해를 다루기 위해서는 LISP로 유전 프로그래밍 기법을 구현함에 있어 약간의 제약이 따르게 된다. 조건문이라는 특수한 함수로 인하여, 노드에서 이루어진 교차변이라 할지라도 유효한 자손을 생성하지 못할 수도 있기 때문에 교차변이를 좀더 복잡하게 구성해야 하기 때문이다.

따라서, 위와 같은 어려움을 해결하고 퍼지로직 제어기로의 확장의 용이성을 위해 본 논문에서는 표 1과 같은 함수 및 터미널 집합을 사용한다.

표 1. 함수 및 터미널 집합.
Table 1. Function and terminal sets.

함수	{ifTN6, ifTN5, ..., ifTN1, ifTZ, ifTP1, ..., ifTP6, ifDN2, ifDN1, ifDZ, ifDP1, ifDP2}
터미널	{random number in [-5, 5]}

여기에서 T는 θ를 나타내고 D는 $\dot{\theta}$를 나타낸다. 그리고, N, Z, P 는 각각 음수, 0, 양수를 의미한다. 각 함수들은 인수를 2개 가질 수 있으며, 다음과 같은 일을 한다. 예를 들어 (ifTN1 arg1 arg2) 라는 명령은,

$$\begin{aligned} &\text{if } \theta \geq c_{TM} \text{ then}(\text{if}(\text{arg}2 = \text{number}) \text{ then } u \leftarrow \text{arg}2 \\ &\qquad \text{else evaluate arg}2) \\ &\text{else}(\text{if}(\text{arg}1 = \text{number}) \text{ then } u \leftarrow \text{arg}1 \\ &\qquad \text{else evaluate arg}1) \end{aligned} \tag{3}$$

여기에서 argi 는 i번째 인수를 나타내고, c_{TN1}은 함수 ifTN1의 중심 값이다. 함수이름의 숫자가 커질수록 중심값의 절대값도 커지게 된다. 위와 같은 함수를 사용하면 노드를 중심으로 이뤄지는 어떠한 교차변이도 유효한 자손을 생성하게 된다. 중심값의 사용 대신 중심값도 인자로 가지는 함수를 생각할 수 있으며, 이 경우 함수의 개수가 많이 줄어들게 되나 모의 실험 결과 제안된 방법에 비해 상대적으로 계산량이 많고 수렴 특성도 더 안 좋은 것으로 드러났다. 또한, 이 경우엔 교차변이 후의 유효한 자손 생성에 대한 고려를 따로 해주어야 한다. 모의 실험에 사용된 중심 값들은 다음과 같다.

$$\{0, \pm0.01, \pm0.02, \pm0.04, \pm0.08, \pm0.16, \pm0.32\} \text{ for } \theta \tag{4}$$

$$\{0, \pm0.3, \pm0.9\} \text{ for } \dot{\theta} \tag{5}$$

사용된 중심 값들은 제어 목표점 근처에서 보다 정밀한 제어를 위해 0 근처에 많이 분포하도록 사전정보 없이

적당히 정해준 것들이지만, 실험 결과가 안 좋을 때는 시행 착오를 거쳐 조정할 수도 있을 것이다. LISP 언어를 비롯한 대부분의 프로그래밍 언어가 함수의 인수를 함수의 실행 전에 미리 평가하도록 되어있으므로 위의 함수들은 모두 pseudo 매크로로 씌어졌다. 터미널 집합은 단순히 -5에서 5사이의 실수 값을 가지는 난수(random number)로만 이루어져 있고, 제어 입력 u 에 해당된다. 집단의 초기화 및 돌연변이 과정에서 난수 함수가 호출되어 새로운 난수가 터미널 값으로 쓰이기 위해 생성되고, 재생산 및 교차변이 과정에서는 터미널 값의 변화가 필요하지 않으며 각 터미널 값들이 그대로 유지된다.

유전프로그래밍을 적용하기 위한 적합도 함수는 다음과 같이 주어진다.

$$Fitness = 1 / \sum_{trials} \int_0^t t^2\theta^2 dt \qquad (6)$$

진자의 변위가 작을수록, 빨리 안정화 될 수록 큰 값이 나오도록 하는 함수이다. 모의 실험은 각 개체마다 9가지의 초기조건에 대해 각각 3초씩 수행되며 계산량 단축을 위해 Euler 방법을 이용하여 적분하였다. 샘플링 시간은 0.02초이다. 9가지의 초기조건은 다음과 같다.

$$\{(\theta, \dot{\theta}) | \theta \in \{-0.1, 0, 0.1\},\ \dot{\theta} \in \{-0.3, 0, 0.3\}\} \qquad (7)$$

이상과 같은 함수 및 터미널 집합, 그리고 적합도 함수를 사용하여 교차변이 확률 0.9, 돌연변이 확률 0.1, 집단의 개체 수는 50으로 하여 300세대 동안 유전프로그래밍을 적용하였다. 적용결과 가장 나은 결과를 보인 개체는 200세대 째에서 나왔으며 적합도 0.429를 가졌다. 원래 개체는 270줄이었으나, 유효한 부분만을 정리하면 다음과 같았다.

```
(ifDZ (ifTN4 (ifDN1 3.0194750558675629
                    -1.744521617304776)
(ifTZ -4.0784779559255018 1.0078543499148704))      (8)
(ifTN4 (ifDP2 -1.520462891329295
                    -1.744521617304776)
    (ifTZ -1.520462891329295 3.8547063939528101)) )
```

정리된 해는 사용된 함수의 개수에 비해 비교적 간단한 형태임을 볼 수 있다. (8)의 제어기는 상태변수로이루어진 $\theta - \dot{\theta}$ 평면을 8곳으로 나누고 각 공간에 제어 입력이 하나씩 해당되는 꼴임을 알 수 있다. 그림 4에 분할된 상태공간과 각 제어입력을 나타내었다.

그림 5-7에 (8)을 사용한 결과를 보였다. 그림 5와 6은 초기조건 $(\theta, \dot{\theta}) = (0.1, 0.3)$ 에 대한 시간에 대한 θ 와 $\dot{\theta}$ 의 그래프를 보이고 있다. 그림 7은 초기조건 $(\theta, \dot{\theta}) = (0.4, 0.6)$ 에 대한 결과를 보이고 있다. 유전 프로그래밍 적용시에 포함되지 않았던 초기조건임에도 제어가 제대로 이뤄지고 있음을 보이고 있다. (7)의 초기조건보다 절대값이 작은 θ와 $\dot{\theta}$ 에 대해서도 제어가 잘 이루어짐을 확인할 수 있었다.

3. 퍼지로직 제어기의 설계

그림 4의 제어 출력은 다음과 같은 분석을 통해, $\theta - \dot{\theta}$ 평면의 각 사분면에 제어입력이 대략 하나씩 대응되는

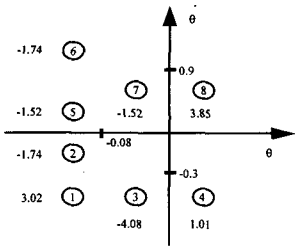

그림 4. 규칙 기반 제어기의 출력.
Fig. 4. Control output of the rule based controller.

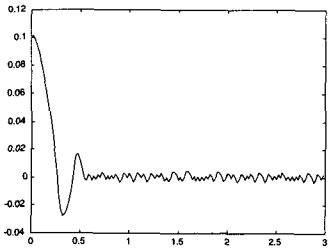

그림 5. $(\theta_0, \dot{\theta}_0) = (0.1, 0.3)$ 에 대한 θ.
Fig. 5. θ for $(\theta_0, \dot{\theta}_0) = (0.1, 0.3)$.

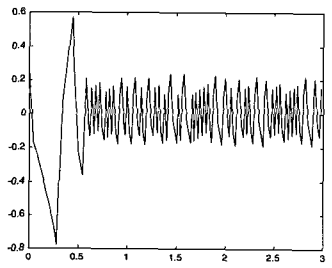

그림 6. $(\theta_0, \dot{\theta}_0) = (0.1, 0.3)$에 대한 $\dot{\theta}$.
Fig. 6. $\dot{\theta}$ for $(\theta_0, \dot{\theta}_0) = (0.1, 0.3)$.

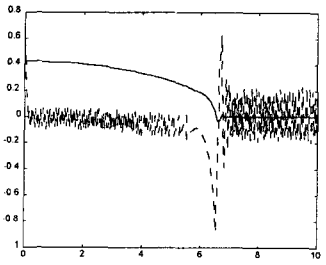

그림 7. $(\theta_0, \dot{\theta}_0) = (0.4, 0.6)$에 대한 θ 와 $\dot{\theta}$.
Fig. 7. θ and $\dot{\theta}$ for $(\theta_0, \dot{\theta}_0) = (0.4, 0.6)$.

꼴이라는 것을 알 수 있다. 상태 평면 ① 에서의 제어 출력은 ②, ③ 과 크게 다르지만 9개의 초기조건에 대해 행해졌던 모의실험의 결과를 분석한 결과, ⑥ 에서의 제어 입력과 함께 실제 제어에는 사용되지 않았고, ⑤, ⑦ 의 입력은 동일하며 ②, ③ 의 입력은 부호가 같아 상대적으로 유사하다. 또, 진자가 안정화될수록 ③, ④, ⑦, ⑧ 에서의 제어 입력이 중요한 역할을 함을 알 수 있다. 이를 바탕으로 $\theta - \dot{\theta}$ 평면의 각 사분면에 해당하는 4개의 규칙을 가지는 퍼지로직 제어

기로 변환이 가능하다. 설계된 규칙 기반 제어기의 8개의 상태공간 분할과 제어 입력을 그대로 사용해도 되지만, 위와 같은 해석을 통해 퍼지로직 제어기의 규칙 수를 줄일 수 있게 된다. 1-4 사분면의 제어 출력은 각각 4, -1.5, -4, 1.5 로 대표할 수 있을 것이다. 여기서 규칙 기반 제어기의 제어 입력을 그대로 사용해도 무방하나, 역진자 시스템의 대칭성을 알고 있다고 가정하여 원점에 대칭인 사분면의 제어 출력의 절대값을 같게 정하였다.

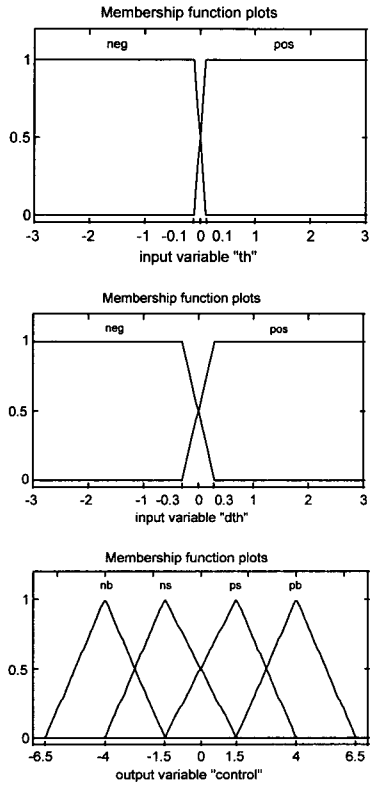

그림 8. 퍼지로직 제어기의 입력 및 출력의 언어 변수와 소속함수.

Fig. 8. Linguistic variables and membership functions of the input and output of the fuzzy logic controller.

그림 8에 각 퍼지로직 제어기의 입력인 상태변수 (th, dth) 와 출력인 제어 입력 (control) 에 대한 언어 라벨들과 그 소속함수들을 나타내었다. 소속함수의 모양은 그림 4의 분석 결과 및 초기조건을 고려하여 결정된다. 예를 들어 입력 변수 dth는 사분면의 정의를 위해서 0을 중심으로 NEG, POS라는 포화함수 형태의 소속함수를 가진다. 포화함수의 꺾인 점을 0으로 하면, 구성될 퍼지로직 제어기와 규칙 기반 제어기가 같아지게 된다. 일반적인 경우, 이 꺾인 점은 소속함수가 해당되는 상태공간의 중심값으로 정한다. 여기서는 사분면이 무한하므로, 초기값을 이용하여 꺾인 점을 ±0.3으로 정하였다. 출력 변수인 control은 각 제어입력을 나타내도록 삼각함수 형태의 소속함수를 가지도록 하였다.

그림 4와 그림 8로부터 다음과 같은 퍼지 규칙을 얻을 수 있다.

If TH is POS and DTH is POS
then CONTROL is PB
If TH is NEG and DTH is POS
then CONTROL is NS (9)
If TH is NEG and DTH is NEG
then CONTROL is NB
If TH is POS and DTH is NEG
then CONTROL is PS

이상과 같이 구성된 퍼지 제어기의 출력 평면을 그림 9에 나타내었다. max-min 추론과정과 무게중심점 비퍼지화 방법을 사용하였다[13].

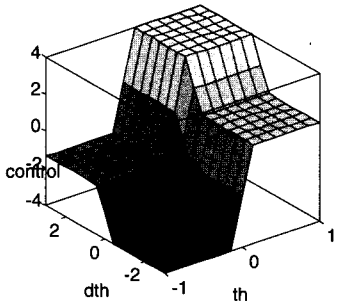

그림 9. 퍼지로직 제어기의 출력 평면.

Fig. 9. Output surface of the fuzzy logic controller.

원래의 규칙 기반 제어기와 달리, 구성된 퍼지로직 제어기는 원점 근처에서 미분가능한 출력을 보이고 있다. 그림 10-12에는 설계된 퍼지로직 제어기를 사용했을 때 그림 5-7에 대응하는 결과를 보였다.

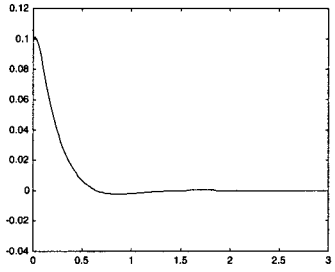

그림 10. $(\theta_0, \dot{\theta}_0)=(0.1, 0.3)$에 대한 θ.

Fig. 10. θ for $(\theta_0, \dot{\theta}_0)=(0.1, 0.3)$.

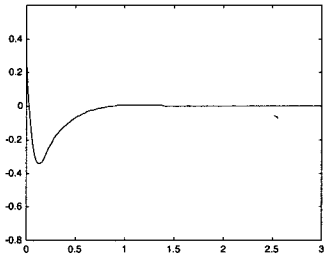

그림 11. $(\theta_0, \dot{\theta}_0)=(0.1, 0.3)$에 대한 $\dot{\theta}$.

Fig. 11. $\dot{\theta}$ for $(\theta_0, \dot{\theta}_0)=(0.1, 0.3)$.

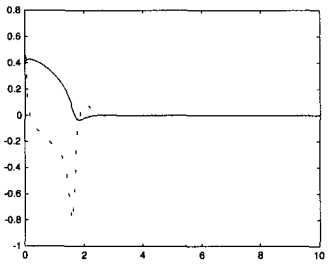

그림 12. $(\theta_0, \dot{\theta}_0)=(0.4, 0.6)$에 대한 θ 와 $\dot{\theta}$.

Fig. 12. θ and $\dot{\theta}$ for $(\theta_0, \dot{\theta}_0)=(0.4, 0.6)$.

규칙 기반 제어기를 이용했을 때에 비해 훨씬 매끄럽고 빠른 응답이 얻어졌음을 볼 수 있다. 규칙 기반 제어기를 사용했을 때 정상상태에서 발생했던 채터링도 퍼지로직 제어기로의 변환을 통해 완전히 없어졌음을 알 수 있다. 규칙 기반 제어기의 경우 출력 평면이 분할된 상태 공간의 경계에서 불연속이고 나머지 부분에선 상수함수이기 때문에 평형점인 $(\theta^*, \dot{\theta}^*)=(0, 0)$에 수렴하지 못하고 채터링이 생기며 초기에 오버슈트가 크게 생기는 반면에, 퍼지로직 제어기는 그림 9에서 보는바와 같이 연속인 출력 평면을 가지고 있으며 평형점에서의 제어출이 0이 되므로 부드럽고 빠른 응답과 평형점으로의 좋은 수렴성을 보이게 된다.

V. 결론

본 논문에서는 유전 프로그래밍 기법을 이용하여 제어기의 설계가 어려운 시스템에 대해서도 제어기 구조에 대한 정보 없이 if-then 으로 이루어진 규칙 기반 제어기를 설계할 수 있도록 유전 프로그래밍에 사용되는 해의 구조, 함수 및 터미널 집합들을 분석하였다. 제안된 함수들을 사용한 해는 노드에서 이뤄지는 교차변이 후에는 항상 유효한 자손이 생성된다는 장점이 있다. 기존의 진화 연산 기법을 사용한 퍼지로직 제어기의 설계는 진화 연산 기법 적용 과정에 퍼지로직 제어기가 포함되기 때문에 탐색 공간이 복잡해지고 시간이 많이 걸리는 단점이 있다. 하지만, 본 논문에서는 규칙 기반 제어기를 먼저 설계하고, 설계된 규칙 기반 제어기를 이용하여 더 나은 특성을 갖는 퍼지로직 제어기를 보다 용이하게 설계할 수 있음을 보였다. 미지의 비선형 시스템의 예로 든 역진자 시스템에 대한 모의실험을 통하여 제안된 방법의 타당성을 보였다.

참고문헌

[1] J. R. Koza, *Genetic Programming,* MIT Press, 1993

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and MachineLearning,* Addison-Wesley, 1989

[3] L. Davis, *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, 1991

[4] D. B. Fogel, Tutorial S-5 : evolutionary computation, *IEEE Int. Conf. on Robotics and Automation,* 1994

[5] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Trans. Syst., Man, Cybern.,* vol. 22, no. 5, pp. 1033-1046, 1992.

[6] Y. Ichikawa and T. Sawa, "Neural network application for direct feedback controllers," *IEEE Trans. Neural Networks,* vol. 3, no. 2, pp. 224-231, 1992.

[7] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.,* vol. 1, no. 1, pp. 46-53, 1993

[8] W. R. Hwang and W. E. Thompson, "Design of intelligent fuzzy logic controllers using genetic algorithms," *Proc. of the Third IEEE Int. Conf. on Fuzzy Systems,* pp. 1383-1388, 1994.

[9] M. L. Wong and K. S. Leung, "Combining genetic programming and inductive logic programming using logic grammars," *IEEE Int. Conf. on Evolutionary Computation,* pp. 733-736, Dec., 1995

[10] I. K. Jeong and J. J. Lee, "Genetic algorithms and neural networks for identification and control," *proc. of the First Asian Control Conference,* pp. 697-700, 1994.

[11] I. K. Jeong and J. J. Lee, "A modified genetic algorithm for neurocontrollers," *IEEE Int. Conf. on Evolutionary Computation,* pp. 306-311, 1995.

[12] I. K. Jeong and J. J. Lee, "Self organizing genetic algorithm," *Proc. of Int. Symposium on Artificial Life and Robotics,* pp. 18-21, Feb., 1996.

[13] W. Pedrycz, *Fuzzy Control and Fuzzy Systems,* John Wiley & Sons, 1993.

정 일 권

제어 · 자동화 · 시스템공학 논문지 제3권, 제3호, 참조.

이 주 장

제어 · 자동화 · 시스템공학 논문지 제1권, 제2호, 참조.