論文 한국정밀공학회지 제 12권 제6호 1995. 6
Journal of the Korean Society of Precision Engineering Vol. 12. No 6. June. 1995.

CMAC 제어기를 이용한 점착 미끄럼 마찰의 제어

박 종 현*

Control of Stick-Slip Friction with a CMAC

Jong Hyeon Park*

ABSTRACT

This paper proposes a CMAC-based controller for servo systems with stick-slip friction. Performance of the controller was evaluated from computer simulations and compared with that of a conventional PID controller. Firction model used in the simulations is based upon the one proposed by Tustin. It was shown that the CMAC-based controller settles more quickly, and overshoots less than the PID. It was also shown that the CMAC is less sensitive to the changes of the plant parameters.

Key Words : Stick-Slip(점착 미끄럼), Friction Model(마찰 모델), PID Controller(비례 적분 미분 제어기), Computer Simulation(컴퓨터 모의 실험), Plant Parameters(플랜트 파라미터)

1. 서 론

정밀한 위치를 유지해야 할 목적으로 사용되고 있는 서보들이 점착 미끄럼 마찰(stick-slip friction)에 의해 그 성능이 제한되거나 마찰 파라미터가 시간에 따라 변하여 성능이 시간에 따라 변하는 경우가 많이 있다. 공작기계나 반도체 장비들이 그 대표적인 예라고 하겠다. 이렇게 마찰의 영향을 크게 받는 플랜트를 제어하기 위해서 전통적으로 정상상태오차를 줄일 수 있는 적분항이 포함된 PID 제어기가 사용되어 왔다. 그러나, PID 제어기는 쿨롱(Coulomb) 마찰에 대해서는 정상상태오차를 0으로 만들 수 있지만, 점착 미끄럼 마찰이 있는 경우에서는 음(-)의 댐핑으로 인하여 리미트사이클(limit cycle) 현상을 발생시킬 수 있다. 또한, 일반적으로 PID 제어기의 적분항으로 인하여 시스템의 반응속도는 느려지게 된다.

지금까지 마찰에 대해서는 많은 연구가 되어왔다. 대표적인 마찰의 모델로는 Tustin[1]이 쿨롱 마찰, exponential 함수로 표현되는 점착 마찰 및 점성 마찰의 합으로 표현한 것이 있다. 마찰의 영향을 받고 있는 서보의 제어에 대해서는 Canudas de Wit과 Seront가 비선형 마찰모델의 파라미터(파라미터에 대해서는 선형)의 동정(identification)을 통한 적응제어 등 여러 연구가 된 바가 있다.[2-4]

본 논문에서는 이러한 점착 미끄럼 마찰을 갖고 있는 서보를 실시간 제어가 가능한 CMAC(Cerebellar

* 한양대학교 정밀기계공학과

Model Articulation Controller)을 사용하여 제어하는 시스템을 제안하고 있다. CMAC은 그 동안 Miller, Miyamoto 등에 의해 주로 로봇 시스템에 적용되어 왔다.[7-9]

2. CMAC을 이용한 점착 미끄럼 마찰의 제어

2.1 점착 미끄럼 마찰

본 논문에서는 점착 미끄럼 마찰의 모형으로 Tustin이 제한한 것이 사용되었다. 이 모델에서는 마찰력이 회전속도의 함수로 다음과 같이 표현된다.

$$\tau_f = \left[\tau_s - \Delta\tau\left(1 - e^{-|\dot{\theta}|/\dot{\theta}_0}\right) + b\dot{\theta}\right]\cdot \mathrm{sgn}(\dot{\theta}) \qquad (1)$$

여기서, τ_s는 쿨롱마찰을 나타내며 이로 인하여 힘 공간에서의 불감대(dead-band)가 형성된다. $\Delta\tau$와 $\dot{\theta}_0$에 의해 결정되는 지수함수(exponential function)는 각속도의 크기가 작은 영역에서 토크 대 각속도의 기울기를 음으로 만들어 음의 댐핑을 형성하여, 시스템을 불안정하게 만드는 역할을 한다. b는 점성마찰계수로 각속도에 비례하는 마찰토크를 나타내는 파라미터이다. 가공의 부정확성 등의 원인으로 이들 파라미터들은 일반적으로 각도에 대한 함수가 될 수도 있으나, 본 논문에서의 모의실험에서는 고정된 값이 사용되었다.

2.2 CMAC 제어

CMAC은 신경망의 한 형태로, 인간의 뇌에 대한 기능 분석을 통해 Albus에 의해 처음으로 개발되었다.[5] CMAC은 기본적으로는 참조표(look-up table)를 이용한 방법으로, 입력 대 출력의 관계가 이 참조표에 의해 결정된다. CMAC은 그 원리가 간단하며 실시간 제어도 가능하기 때문에 쉽게 사용될 수 있으나, 신경망을 이용한 여러 다른 제어기와 마찬가지로 제어기의 안정도는 증명되지 않고 있다. 따라서, 본 논문에 사용된 CMAC을 제어기로 사용하는 방법에서는 CMAC을 이용하여 제어 대상 플랜트의 모델을 근사적으로 구하고, 이 모델을 이용하여 피드퍼워드 신호를 구한 다음, 이것을 전통적인 기존의 제어기로부터 나오는 출력에 첨가하였다.

CMAC에 대한 기본적인 설명은 (6)을 참조하여 설명하고자 한다. 신경망을 이용한 다른 많은 제어기와 같이 CMAC은 입력 벡터 공간 S로부터 출력 벡터 공간 G로의 비선형 매핑(mapping)을 표현하는 데 사용된다. 즉,

$$g = f(s) \qquad (2)$$

여기서 s는 S에 속해 있는 다차원 이산 입력 벡터이고, g는 G에 속해 있는 다차원 이산 출력 벡터이며, $f(\cdot)$는 s와 g를 연결짓는 임의의 비선형 함수이다. 각 이산 입력 벡터 s는 메모리 내의 여러 장소(C개의 장소)에 매핑되며 출력 벡터 g는 s에 의해 매핑된 이들 장소에 저장된 값들을 모두 더하여 계산된다. 이 매핑 알고리즘은 공간 S에서 서로 인접된 위치에 있는 벡터들이 메모리(A)상에서 일부 동일한 메모리에 매핑되며(일반화 과정), 서로 멀리 떨어져 있는 벡터들은 메모리 상에서 서로 독립된 장소에 매핑되도록 된다. 이러한 과정을 통해 서로 비슷한 입력들에 대해서는 비슷한 출력을 내보낼 수 있기 때문에, 입력 공간 S상에서의 일반화 과정이 이루어지게 된다. 따라서, 한 입력 벡터에 대해 매핑되는 메모리 장소의 수, C는 일반화의 수준을 결정하는 요소가 된다. 즉, C가 클 경우에는 많은 메모리에 의해 평균적인 값이 출력으로 결정되어 일반화 수준이 높게 되나, C가 작을 경우에는 국부적인 메모리에 의해서만 출력이 결정되어 일반화 수준이 낮게 된다.

이러한 방법을 실행하기 위해서는 많은 메모리가 필요한데, 이렇게 많은 메모리를 갖는 것은 대부분의 경우에 있어 불가능하다. n차원의 입력 벡터의 각 요소들이 N 비트 AD 변환에 의해 양성화될 경우, 모두 $2^{n\times N}$개의 메모리가 필요하다. 예를 들어, 10 비트 AD 변환을 통한 3차원 입력 벡터의 경우, 모두 1.074×10^9개의 메모리 장소가 필요한데, 각 메모리 장소의 데이타를 4 바이트 부동 소수점 형으로 사용한다면 4.296 기가 바이트가 필요하게 된다. 이러한 이유로 인해, 메모리 A는 실제 제어에서는 메모리가 아닌 가상 메모리(hypothetical memory)로 여겨져, 이 가상 메모리의 각 장소는 상당히 작은 크기의 실제 메모리 공간 M의 한 장소로 다시 매핑된다. 실제 제어에서는 메모리 A의 전영역이 사용되지 않기 때문에 메모리 A로부터 그 보다 작은 크기의 메모리 M으로 랜덤(random) 매핑을 시켜 사용할 수 있다. 그런 다음, 출력 f(s)는 이산 입력 벡터 s에 매핑된 메모리 M의 장소들에 저장된 값들을 더함으로써

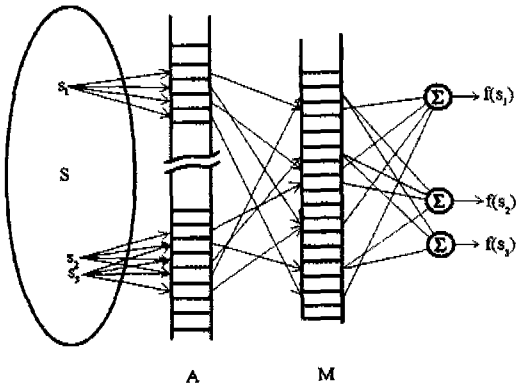

Fig. 1 A block diagram of CMAC structure

구해진다. Fig. 1은 s_1, s_2 및 s_3에 대해 출력 f_1, f_2 및 f_3이 계산되는 과정을 보여준다.

입력 공간 S내에 있는 인접한 두 입력은 A에서 일부 메모리를 공유하게 되고, 이에 따라 M에서도 공유하게 된다. 입력 공간 S내에서 멀리 떨어져 있는 두 입력은 메모리 공간 A에서는 공유 메모리가 없지만, 서로 다른 A의 메모리 장소들이 랜덤 매핑을 통해 M공간에서 동일한 메모리로 매핑될 수 있기 때문에 메모리 충돌이 일어날 수도 있다. 즉, 공간 S에서는 멀리 떨어져 있는 두 입력이 메모리 공간 M에서 동일한 메모리를 공유할 수도 있다. 이것을 방지하기 위해서는 메모리 공간 M의 크기를 너무 작게 설정해서는 안되며, 이러한 메모리 충돌이 일어날 수 있는 경우의 수가 매우 작도록 메모리 공간 M의 크기를 충분히 크게 설정해야 한다.

입력과 출력 간의 임의의 함수를 배우기 위해서는 CMAC이 모델링하려는 함수의 출력을 얻어지는대로 메모리 공간 M이 갱신되고 수정되어야 한다. 이러한 함수의 출력, 즉 모델링하고자 하는 플랜트의 출력이 얻어지면 그 출력에 상대되는 입력에 대해 매핑되는 메모리 장소들(C개)에 있는 데이타 값들을 다음과 같이 수정한다.

$$\Delta m = \beta \cdot \frac{g_0 - f(s_0)}{C} \qquad (3)$$

여기서 C는 입력 벡터 s_0에 매핑된 메모리 장소의 갯수이고, g_0은 CMAC이 모델링하려는 함수의 s_0에 대한 출력이며, $f(s_0)$는 s_0에 대한 CMAC의 출력이다. 또한 β는 0과 1 사이의 수로 학습률을 나타내는데 얼마나 빨리 시스템이 플랜트를 학습하는가를 결정한다. 학습률이 너무 큰 값을 가질 경우에는 시스템의 학습이 안정적이지 못할 수가 있다.

2.3 CMAC을 이용한 점착 미끄럼 마찰의 제어

CMAC은 입력과 출력 사이의 비선형 동적 관계를 모델링하여 PID 제어기의 피드퍼워드 신호를 만드는데 사용된다. 점착 미끄럼 마찰을 가진 일반적인 서보 시스템의 동력학은 다음의 식으로 표현된다.

$$\tau = J\ddot{\theta} + B\dot{\theta} + \tau_f \qquad (4)$$

여기서 J는 회전 질량, B는 댐핑에 관한 상수이고, τ는 회전토크이며, τ_f는 마찰토크이다. 식(1)과 (4)로부터 회전토크는 다음과 같이 표현될 수 있다.

$$\tau = \tau\left(\ddot{\theta}, \dot{\theta}, \theta\right) \qquad (5)$$

즉, 회전토크는 각도, 각속도, 각가속도의 함수가 된다. CMAC은 각도, 각속도, 각가속도로부터 토크의 크기를 예측하며, 이 예측된 마찰토크는 Fig. 2에서와 같이 PID 제어기에 피드퍼워드 된다. CMAC의 입력은 플랜트의 출력 θ, $\dot{\theta}$, $\ddot{\theta}$가 사용될 수 있었으나, 측정할 필요가 없는 기준 입력(reference input) θ_d, 이를 미분한 $\dot{\theta}_d$와 측정된 각속도 $\dot{\theta}$를 이용하여 추측한 $\ddot{\theta}$가 사용되었다. 플랜트로부터 직접 측정한 각가속도를 CMAC에 사용할 수도 있으나, 별도의 센서가 필요하게 되고 가속도 측정에는 잡음이 많게 되어 어려움이 있게 된다.

Fig. 2에서 윗부분은 CMAC이고 아랫부분은 일반적으로 사용되는 PID 제어기이다. 제어기가 처음 사용될 때에는 플랜트에 대한 지식이 전혀 없기 때문에 CMAC의 메모리 장소에는 0의 데이타 값들이 들어있게 되며, 따라서 CMAC의 출력은 0이 된다. 시간이 가면서 CMAC은 플랜트, 즉 식(5)에 대해 배우게 된다. CMAC이 완벽히 플랜트에 대해 배웠을 경우에는 CMAC에서 정확한 토크를 피드퍼워드하기 때문에 추적오차는 0이 된다. 이 경우에는 피드백에 의한 PID

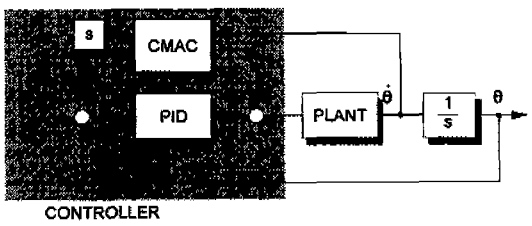

Fig. 2 CMAC controller with a PID

제어기는 그 입력이, 즉 추적오차가 0이 되므로 제어 기능을 발휘하지 못하게 된다. 시스템 사용 초기에는 CMAC이 플랜트를 완전히 학습하지 못하여 그 출력이 정확하지 못하므로 시스템이 불안정해질 수 있는데, 이를 막기 위해 PID 제어기가 사용되는 것이다. 또한, 플랜트의 파라미터가 갑자기 변하는 경우에도 CMAC이 변화된 플랜트를 학습하는 동안에 PID 제어기는 시스템의 안정도를 공급해 준다.

3. 모의 실험

점착 미끄럼 마찰이 존재하는 플랜트에 대한 CMAC 제어의 성능을 평가하고자 컴퓨터 시뮬레이션을 하였다. 대상 플랜트 모델로서 점착 미끄럼 마찰이 존재하는 부하 질량을 드라이브하는 서보 모터 시스템으로 하였다. 이 플랜트 모델에 관련된 파라미터는 Table 1에 나타나 있다. 이 연속적인 플랜트의 시뮬레이션은 4차 Runge-Kutta 방법을 사용하였으며 스텝 크기를 1ms로 하였다. 제어기는 이산 제어기를 사용하였으며, 샘플링 주기는 10 ms로 하였다. 또한, CMAC 제어기의 입력은 10 비트 AD 변환기를 통해 양자화하였다. 제어기의 기준 입력 θ_d는 최대의 가감속을 이용하는 사다리꼴 속도 파형을 적분한 위치 데이타를 사용하였다.

여기에 사용된 PID 제어기의 이득은 마찰을 무시한 플랜트 모델을 이용하여 극점을 $s=-5, -3\pm j3$에 배치하여, k_p=27.389, k_i=51.354, K_d=12.504로 정하였다. 이 이산 PID 제어기의 출력 u_n은 다음의 식으로 구하였다.

$$u_n = k_p e_n + k_i T \sum_{i=0}^{n} e_i + k_d \frac{(e_n - e_{n-1})}{T} \qquad (6)$$

여기서, T는 고정된 샘플링 주기(10ms)이고, e_n은 기준 위치 입력과 플랜트로부터 측정된 위치의 차이인 추종오차이다.

Table 1 Key plant parameters

Parameter	Size	Unit
Kt (Torque Constant)	0.5	kgf-cm/A
Kv (Voltage Constant)	1.9415	V-s/rad
J (Inertia)	0.3170	kgf-cm-s-s
R (Armature Resistance)	1.8	Ω
τ_c	0.6	kgf-cm
$\Delta\tau$	0.4	kgf-cm
$\dot{\theta}_0$	0.1	s/rad
b	0.0	kgf-cm

CMAC의 입력으로는 기준 위치입력, θ_d, 기준 속도 $\dot{\theta}_d$ 및 가속도 $\ddot{\theta}$의 3가지 입력이 사용되었다. 가속도는 직접 플랜트로부터 측정할 경우 잡음이 많게 되어 실용적이지 못하기 때문에 속도를 측정하여 다음의 식으로부터 구해진 값으로 가속도의 값을 근사화하였다.

$$\ddot{\theta}_n = \frac{\dot{\theta}_n - \dot{\theta}_{n-1}}{T} \qquad (7)$$

여기서, T는 고정된 샘플링 주기(10ms)이고 $\dot{\theta}_n$과 $\dot{\theta}_{n-1}$은 타코미터를 이용하여 측정된, 양자화된 속도이다. CMAC 입력 3개가 모두 10 비트로 양자화되었는데 이를 직접 사용하기 위해서는 필요한 메모리가 너무 커지므로, 여기서는 해싱 함수(hashing function)를 이용하여 더 작은 메모리 공간으로 매핑시켰다. 모의 실험에서는 500개, 1000개, 10,000개의 메모리 공간을 사용하여 실험하였다. 일반화 수준 C는 64를 사용하였고, 학습률 β는 0.1, 0.3 및 0.6을 사용하여 실험하였다.

4. 결과 및 고찰

메모리 공간의 크기(M)를 1000, 일반화 수준(C)을, 64, 학습률(β)을 0.6으로 하였을 때에는 메모리 충돌이 일어났다. Fig. 3에서 30회의 학습을 한 후에도 시스템의 반응이 일정하지 못하고 계속 변화하고 있는 것은 메모리 충돌이 발생하여 입력과 출력의 안정적인 매핑을 발견하지 못했기 때문이다. Fig. 4는 M=1000, C=64, b=0.6인 CMAC을 1000회 학습시키면서 측정한 1% 정착시간을 나타낸 것인데, 정착시간이 대부분의 경우 작았으나 가끔 큰 값으로 변하는 경우가 계속적으로 발생하였다. 이를 피하기 위해 1000개의 메모리

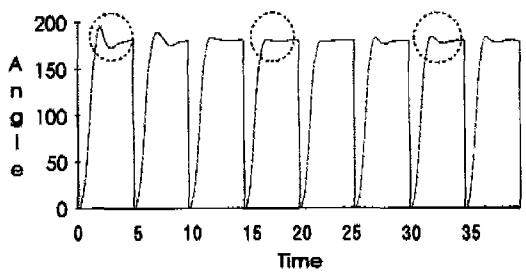

Fig. 3 Memory conflicts occur after 30 learning cycles when 1,000 memory cells are used with C=64. (The shape of circled regions are different)

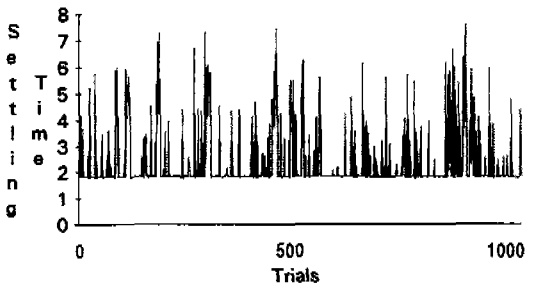

Fig. 4 Settling times are often very long and change very frequently for M=1000, C=64, and β=0.6.

공간을 사용하고 일반화 수준 C를 8로, 그리고 β를 0.1로 하였을 때 만족스런 결과가 얻어졌는데, 이 결과는 Fig. 5에 잘 나타나 있다. 이것도 30회의 학습을 통한 다음에 얻은 시스템 반응인데, Fig. 3과는 달리 반응의 변화는 거의 없으며, 입력에 대해 빠르고 일정하게 수렴함을 보여주고 있다. Fig. 6은 이러한 CMAC을 1000번 학습시키는 과정에서 측정한 1% 정착시간을 나타낸 것인데, 처음에는 정착시간이 컸으나

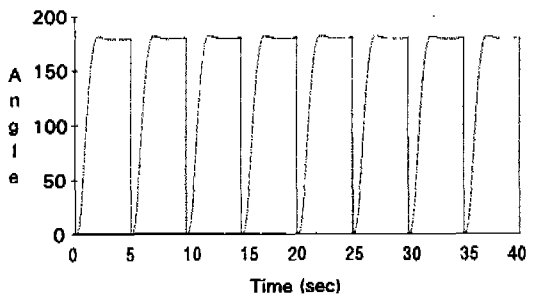

Fig. 5 CMAC controller shows good steady responses even after 30 learning trials for M=1000, C=8, and β=0.1.

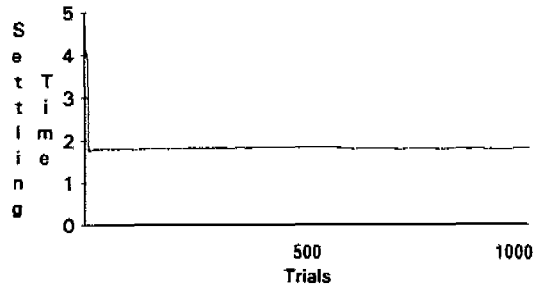

Fig. 6 Settling times are steady low after few initial trials for M=1000, C=8, and β=0.1.

곧 1.8초 정도로 줄어들었으며, 계속적으로 그 시간대를 유지했다. Fig. 7은 이와 같은 CMAC에 학습률 β만을 0.3으로 높인 것인데, 가끔 정착시간이 커지는 경우가 있는 것을 볼 수 있다. Fig. 8은 Fig. 6과 같은 파라미터를 사용한 것인데 단지 메모리 공간 크기를 10,000으로 증가시킨 것이다. 이것은 정착시간의 미소변동이 더욱 줄어들었지만, Fig. 6과 거의 비슷한 양상

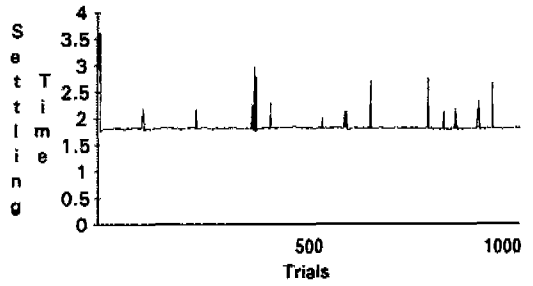

Fig. 7 Settling times are steady low with few exceptions for M=1000, C=8, and β=0.3: too large β.

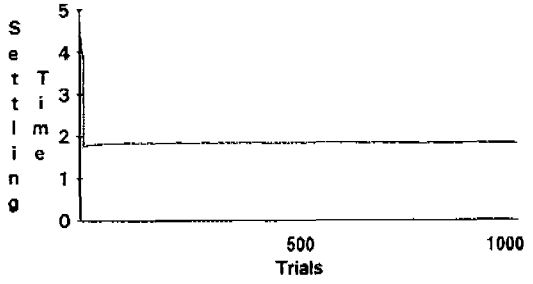

Fig. 8 Settling times are very much steady low after a few initial trials for M=10,000, C=8, and β=0.1.

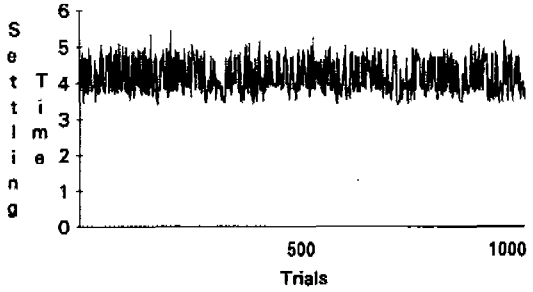

Fig. 9 Settling times are long and change very often for M=500, C=8, and β=0.1: too small memory areas.

을 보였다. 따라서, M=1000인 경우가 메모리를 적게 사용하고 성능도 좋은 것으로 나타났다. 이와는 반대로 메모리 공간 크기가 1000이하인 경우를 고려해 보기 위해 500개의 메모리를 사용하여 실험을 하였는데, Fig. 9에 그 결과가 나타나 있다. 이것은 Fig. 6에 비해 메모리 충돌이 매우 자주 발생하였으며, 정착시간도 길어졌음을 보여주고 있다.

30회의 학습을 통한 CMAC 제어 시스템과 PID 제어만을 사용한 제어 시스템을 비교해 보면, 똑같은 크기의 PID 제어 이득을 사용했어도 CMAC 제어기에 의한 시스템이 성능면에서 월등함을 보여 주었다. Fig. 10(a)는 PID와 CMAC와 반응을 비교한 것인데, 1% 정착시간은 PID 제어기의 경우 4.71초(기준 입력이 최종치에 도달한 후로부터 2.71초)이고 CMAC 제어기의 경우는 평균 1.83초(기준 입력이 최종치에 도달하기 0.17초 전)로써 오히려 기준 입력이 최종치(180도)에 도달하기 전에 1%내에 정착하였다. 이는 제어기가 계속 반복되는 입력에 대해 학습을 하였기 때문에 그 반복 입력에 알맞는 출력을 내보낼 수 있었음을 의미한다. Fig. 10(b)는 Fig. 10(a)에 나타난 반응을 2-5초 사이의 것만을 확대

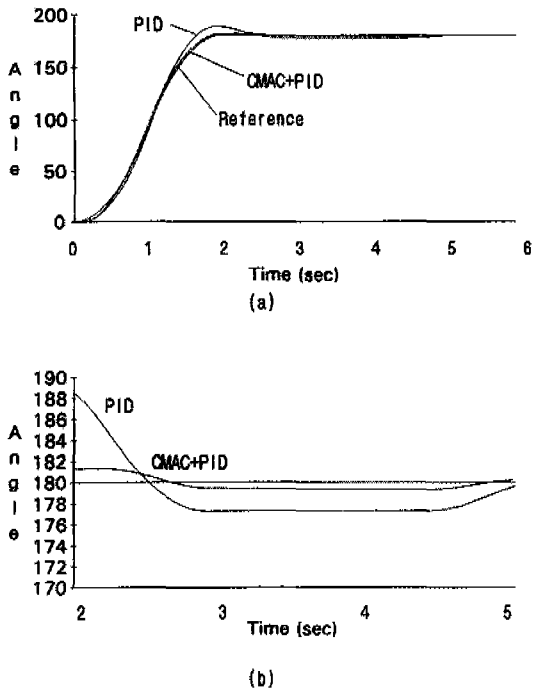

Fig. 10 Responses of the CMAC and the PID: the CMAC settles much more quickly.

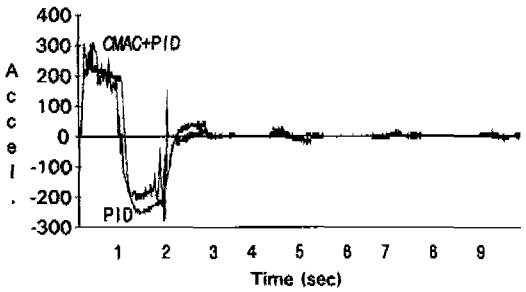

Fig. 11 The CMAC exhibits a slightly higher acceleration than the PID.

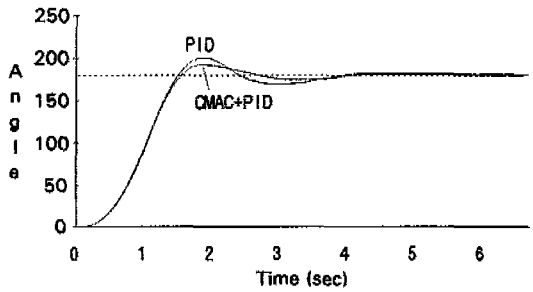

Fig. 12 Responses of the CMAC and PID when the inertia, J, is increased by 50%: the CMAC still performs better than the PID.

한 것인데, CMAC이 보다 빨리 목표점에 수렴함을 볼 수 있다. 오버슈트(overshoot)에서도 PID의 경우에는 4.76%이었으나 CMAC의 경우에는 0.12%로 CMAC이 더 우수함을 보여주었다. 플랜트 출력의 가속도는 Fig. 11에 나타난 바와 같이 CMAC의 경우 PID에 비해 더 큰 최대가속도를 나타냈으며, 가속도 값의 변화의 폭도 좀 더 컸다. 이는 CMAC 제어가 더 많은 제어 활동을 하였음을 나타낸다.

플랜트 파라미터의 변화에 대한 적응능력을 평가하기 위해 서보의 회전관성을 50% 갑자기 증가시킨 경우를 모의 실험하였는데 그 결과가 Fig. 12에 나타나있다. PID와 CMAC의 성능이 모두 Fig. 7에 나타난 것에 비해 나빠졌다. 그러나 CMAC 제어는 그 성능이 약간 나빠졌을 뿐, 여전히 PID 보다 더 좋은 반응을 보여주었다. 이것은 비록 CMAC이 학습 초기에는 PID 제어에 의존하여 CMAC 제어의 성능이 PID의 성능에 많이 의존하지만, 학습의 속도가 빨라서 작은 양의 학습을 통해서도 PID 제어 성능을 능가함을 보여준다. 물론 CMAC 제어 시스템은 이 변화된 관성에 대해 계속

학습함으로써 그 성능을 더욱 개선하게 된다.

5. 결 론

CMAC을 이용한 제어기를 비선형성의 점착 미끄럼 마찰이 있는 모터 시스템에 적용하여 그 유용성을 컴퓨터 시뮬레이션을 통해 검증하였다. CMAC에서는 메모리 공간의 크기, 일반화 수준 및 학습률이 중요한 파라미터가 되었다. 시뮬레이션 결과 CMAC은 PID 단독으로 사용되었을 경우보다 오버슈트(overshoot)및 정착시간을 줄였다. 또한 CMAC 제어기는 모터의 회전관성의 변화에 대해 PID보다 더 좋은 성능을 보였다.

CMAC 제어기는 고가의 장비 없이도 쉽게 사용할 수 있다. 본 논문에 사용된 CMAC은 486/50 PC를 이용해 실시간으로 실행할 수 있다. 메모리 공간도 1000개의 4 바이트 부동소수점 수를 사용하여 4 킬로바이트의 메모리만 필요하다. 이러한 CMAC의 좋은 여러특성에도 불구하고, 신경망에 근거한 많은 제어기가 그렇듯이 CMAC 제어기의 안정성은 증명되지 않고 있다.

후 기

본 연구는 과학재단 핵심전문 연구과제(931-1000-039-12) 지원에 의해 수행되었으며, 이에 감사를 드립니다.

참고 문헌

1. Tustin, A., "The Effects of Backlash and of Speed-Dependent Friction on the Stability of Closed-Cycle Control Systems", J. of IEE, Vol. 94, No. 2A, pp. 143-151, 1947.
2. Southward, S. C., Radcliffe, C. J., and MacCluer, C. R., "Robust Nonlinear Stick-Slip Friction Compensation", J. of Dynamic Systems, Measurement, and Control, Vol. 113, pp. 639-645, 1991.
3. Haessing, Jr. D. A. and Friedland, B., "On the Modeling and Simulation of Friction", J. of Dynamic Systems, Measurement, and Control, Vol. 113, pp. 354-362, 1991.
4. Canudas de Wit, C. and Seront, V., "Robust Adaptive Friction Compensation", IEEE Int. Conf. on Robotics and Automation, pp. 1383-1388, 1990.
5. Albus, J. S., "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)", J. of Dynamic Systems, Measurement, and Control, pp. 220-227, 1975.
6. Miller, W. T., "Sensor-Based Control of Robotic Manipulators Using a General Learing Algorithm", IEEE J. of Robotics and Automation, Vol. 3, No. 2, April 1987.
7. Miller, W. T., Glanz, F. H., and Kraft, L. G., "Application of a General Learning Algorithm to the Control of Robotic Manipulators", Int. J. of Robotics Research, Vol. 6, No. 2, pp. 84-98, 1987.
8. Kraft, L. G. and Campagna, D. P., "A Summary Comparison of CMAC Neural Network and Traditional Adaptive Control Systems", In Neural Networks for Control, Ed. W. T. Miller, Ⅲ, R. S. Sutton, and P. J. Werbos, MIT Press, 1990.
9. Miyamoto, H. and *et al.*, "Feedback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator", Neural Networks, Vol. 1, pp. 251-265, 1988.