한국정밀공학회지 제20권 제2호 (2003년 2월)
Journal of the Korean Society of Precision Engineering Vol. 20, No. 2, February 2003.

자코비안을 이용한 최적의 신경망 제어기 설계

임윤규*, 정병묵**, 조지승*

Optimal Neural Network Controller Design using Jacobian

Yoon Kyu Lim*, Byeong Mook Chung** and Che Seung Cho*

ABSTRACT

Generally, it is very difficult to get a modeling equation because multi-variable system has coupling relations between its inputs and outputs. To design an optimal controller without the modeling equation, this paper proposes a neural-network (NN) controller being learned by Jacobian matrix. Another major characteristic is that the controller consists of two separated NN controllers, namely, proportional control part and derivative control part. Simulation results for a catamaran system show that the proposed NN controller is superior to LQR in the regulation and tracking problems.

Key Words : Multi-variable system(다변수 시스템), Neural Network Controller(신경망 제어기), Jacobian(자코비안), Proportional Control(비례 제어), Derivative Control(미분 제어)

1. 서론

자동제어는 로봇, 자동차, 조선 및 우주 항공 분야 등 첨단산업 분야의 발달을 촉진하였으며 이들 분야에 대한 많은 연구가 진행되고 있다[1-3]. 그리고 이들 분야의 제어 대상 플랜트는 대부분 입출력이 많은 다변수 시스템이다. 다변수 시스템은 입력과 출력이 서로 상관관계(correlation)를 가지고 있기 때문에 플랜트를 해석하는데 어려움이 많다[4-6]. 다변수 시스템의 제어기를 설계하기 위하여 현대 제어에서는 다양한 형태의 제어기법에 대해 연구하고 있다. 그러나 이러한 제어기를 설계할 때, 시스템의 모델을 모르는 경우에는 시행착오를 거듭하여 제어기를 설계하기 때문에 많은 시간과 노력이 필요하다. 이러한 문제를 해결하기 위하여 주어진 환경에 적응할 수 있는 자기구성 제어기법(SOC: Self-Organized Controller)에 대한 연구가 활발히 이루어지고 있다[7-11].

신경망(NN: Neural Network) 이론은 인간 두뇌의 정보처리 방식을 모방하여 만든 알고리즘으로 1980년대 이후부터 적응 신경망을 이용하여 자기구성 제어기를 설계하는 방법이 많이 연구되고 있다. 일반적으로 다층 신경망 학습에 많이 사용되는 방법이 역전파 알고리즘인데[9-11] 이것은 목표치와 시스템의 출력 사이의 에러를 신경망의 연결강도로 역전파하여 연결강도를 수정하는 방식이다. 자동 제어에서는 목표치와 플랜트 출력 사이의 에러를 신경망의 연결강도로 역전파하기 위해서는 먼저 에러를 플랜트 출력에서 입력으로 역전파해야 한다. 이를 위해서는 플랜트 출력과 입력

2002년 3월 13일 접수
* 영남대학교 기계공학부
** 영남대학교 기계공학부 대학원

사이의 관계를 알아야 하는데 에러를 플랜트 출력에서 입력으로 역전파할 수 있는 방법의 하나로 플랜트를 신경망으로 모델링하는 방법이 있다[9-11]. 신경망 모델은 제어 대상인 플랜트에 적합한 모델의 신경망 구조를 개발해야하며 이 모델 신경망을 이용하여 입출력사이의 관계를 알아낼 수가 있다. 그러나 이러한 경우에 플랜트의 특성이 조금만 바뀌면 새로운 신경망 모델을 구해야 하는 어려움이 있고 실제 플랜트와 모델 신경망 사이에는 에러를 줄이는 데에는 한계가 있다[12]. 그러나 만일 플랜트의 입력과 출력사이의 관계를 나타내는 자코비안[7-8,13-15]을 이용한다면 플랜트를 모델화하는 작업 없이 플랜트 출력의 에러를 입력의 에러로 바꿀 수 있을 뿐만 아니라 플랜트의 특성이 조금 바뀌는 경우에 대해서도 큰 영향을 받지 않는다. 따라서 본 논문에서는 자코비안을 이용하여 출력 에러를 입력 에러로 바꾸고 이를 신경망으로 역전파하여 궁극적으로 출력 에러를 최소화하는 신경망 제어기를 만드는 방법을 제안하고자 한다. 이 제어기의 특징은 기존의 일반적인 신경망 제어기 대신에 비례 제어 신경망(PCNN: Proportional Control NN)과 미분 제어 신경망(DCNN: Derivative Control NN)으로 분리한 신경망 구조를 사용하였다. 그리고 연결강도의 학습에서 입력층과 은닉층 사이의 연결강도 수정은 연결강도와 연결된 변수의 에러만을 고려하였고 제안한 신경망 제어기를 쌍동선의 레귤레이션 문제와 추적 문제에 적용하여 이 방법의 우수성을 확인하였다.

2. 최적의 신경망 제어기

일반적인 제어 시스템은 Fig. 1과 같이 플랜트 출력 $\boldsymbol{y}(t_n)$을 피드백하여 목표치 $\boldsymbol{d}(t_n)$와 차이 $\boldsymbol{e}(t_n)$을 이용하여 제어기에서 입력 $\boldsymbol{u}(t_n)$을 계산하여 플랜트를 구동하여 새로운 출력 $\boldsymbol{y}(t_{n+1})$을 발생한다. 이 과정에서 출력과 목표치의 차이를 줄이는 것이 제어기의 역할이다.

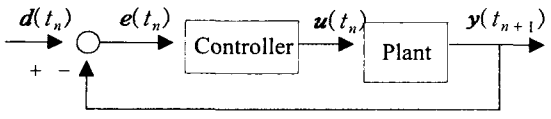

Fig. 1 Block diagram of control system

2.1 신경망 제어기의 구조

신경망 제어기는 일반적인 제어기 대신에 신경망 제어기를 사용한다. 그러나 본 논문에서 제안한 신경망 제어기는 Fig. 2에서처럼 목표 궤도와 플랜트 출력의 차인 에러를 입력으로 하는 비례 제어 신경망과 목표 궤도의 변화율과 플랜트 출력 변화율의 차인 에러 변화율을 입력으로 하는 미분 제어 신경망으로 구성된다.

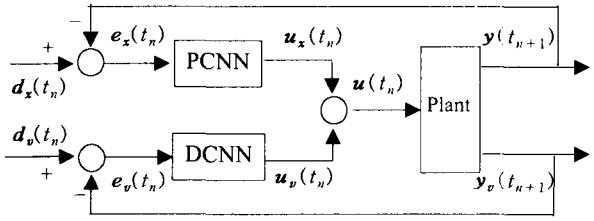

Fig. 2 Block diagram of proposed Neural Network Controller

비례 제어 신경망에서 목표 벡터는 다음과 같다.

$$\boldsymbol{d}_x(t_n) = [\, d_{x_1}(t_n) \;\cdots\; d_{x_q}(t_n)\,] \tag{1}$$

여기서 q는 출력 변수의 개수이다. Fig. 3은 3계층의 비례 제어 신경망과 미분 제어 신경망을 나타내었다. 비례 제어 신경망은 Fig. 3의 a)에서 보는 것처럼 입력층은 에러이고 출력층은 $\boldsymbol{u}_x(t_n)$으로 구성되어 있다. 에러는 목표치와 출력으로 나타낼 수 있다.

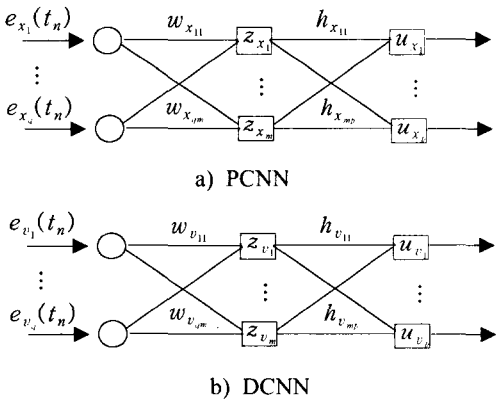

Fig. 3 Constructions of PCNN and DCNN

$$\begin{aligned} \boldsymbol{e_x}(t_n) &= \boldsymbol{d_x}(t_n) - \boldsymbol{y}(t_n) \\ &= [e_{x_1}(t_n) \cdots e_{x_q}(t_n)] \end{aligned} \tag{2}$$

m개의 은닉층 뉴런이 사용되었을 때, 입력층과 은닉층 사이의 연결 강도 행렬을 $\boldsymbol{W_x}(t_n)$로 나타내면 은닉층의 출력은 에러와 입력층 연결강도의 곱에 의해 구할 수 있다.

$$\begin{aligned} \boldsymbol{z_x}(t_n) &= F(\ \boldsymbol{NET}_{z_x}(t_n)) \\ &= F(\ \boldsymbol{W_x}^T(t_n)\,\boldsymbol{e_x}(t_n)) \\ &= [z_{x_1}(t_n),\ \cdots,\ z_{x_m}(t_n)]^T \end{aligned} \tag{3}$$

여기서 F는 활성화 함수(activation function)이고, $\boldsymbol{NET}_{z_x}(t_n) = \boldsymbol{W_x}^T(t_n)\,\boldsymbol{e_x}(t_n)$ 이다. 신경망 제어기의 출력층은 제어 입력이므로, 제어 입력이 p개 일 때 출력층 뉴런의 개수 역시 같다. 따라서 은닉층과 출력층 연결 강도 행렬 $\boldsymbol{H_x}(t_n)$로 나타내면 출력층은 은닉층의 출력과 출력층 연결강도에 의해 구할 수 있다.

$$\begin{aligned} \boldsymbol{u_x}(t_n) &= F(\ \boldsymbol{NET}_{u_x}(t_n)) \\ &= F(\ \boldsymbol{H_x}^T(t_n)\,\boldsymbol{z_x}(t_n)) \\ &= [u_{x_1}(t_n),\ \cdots,\ u_{x_p}(t_n)]^T \end{aligned} \tag{4}$$

미분 제어 신경망은 Fig. 3의 b)에서 보는 것처럼 입력층은 에러 변화율 $\boldsymbol{e_v}(t_n)$이고 출력층은 $\boldsymbol{u_v}(t_n)$으로 구성되어 있다. 연결강도와 은닉층의 뉴런은 비례 제어 신경망과 같으며, 첨자 x를 v로 바꾸어 나타내었다.

2.2 학습 알고리즘

일반적인 신경망에서 학습은 출력층의 출력과 목표치의 차를 역전파하여 학습을 한다. 그러나 신경망 제어기에서는 플랜트의 출력과 목표치의 차를 역전파해야 한다. 그러나 역전파 과정에서 정보를 알지 못하는 플랜트가 있기 때문에 역전파가 불가능하다. 그래서 플랜트의 입력과 출력에 대한 자코비안을 이용하여 역전파가 가능하도록 한다. Fig. 4는 자코비안을 이용한 신경망 제어기 학습의 블럭선도를 나타낸 것이다. 비례 제어 신경망은 입력과 출력에 대한 자코비안 $\boldsymbol{J}$을 이용하여 학습을 하고, 미분 제어 신경망은 입력과 출력 변화율에 대한 자코비안 $\boldsymbol{J_v}$을 이용하여 학습을 한다.

신경망 제어기의 목적은 에러 $\boldsymbol{e}(t_n)$와 에러 변화율 $\boldsymbol{e_v}(t_n)$을 최소화시키는 것이다. 따라서 신경망 제어기의 목적 함수를 다음과 같이 정의하였다.

$$\begin{aligned} C &= \frac{1}{2}\sum_{k=1}^{q} c_k \\ &= \frac{1}{2}\sum_{k=1}^{q}(\phi_{x_k} e^2_{x_k}(t_n) + \phi_{v_k} e^2_{v_k}(t_n)) \end{aligned} \tag{5}$$

여기서, c_k는 k번째 변수의 목적함수이고, ϕ_{x_k}와 ϕ_{v_k}는 가중치이다. 먼저 비례 제어 신경망의 학습은 다음과 같다.

- 입력층과 은닉층의 연결 강도 조정

일반적으로 신경망은 출력층의 모든 에러를 역

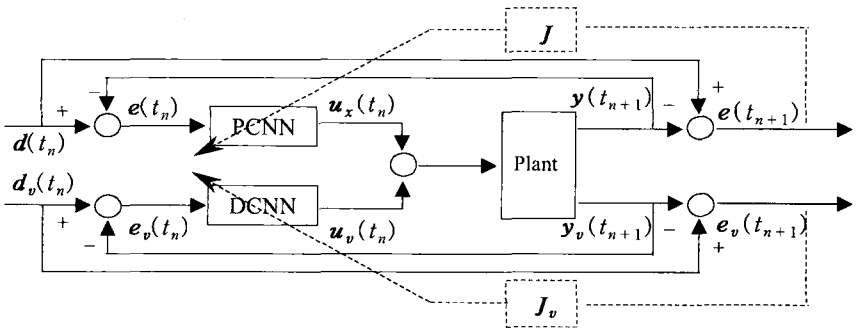

Fig. 4 Block diagram of learning of Neural network controller

전파하여 연결강도를 수정한다. 그러나 제안한 신경망 제어기에서 입력층과 은닉층의 연결강도 조정은 Fig. 5에 나타낸 것처럼 연결강도와 연결된 입력층 변수의 에러만을 역전파한다. 다시 말해서, 입력층 뉴런 k와 은닉층 뉴런 j의 연결강도 $w_{x_{jk}}$는 $e_{x_k}(t_n)$와 곱해져서 출력층으로 전달된다. 이 출력층의 결과가 플랜트의 입력으로 들어가서 플랜트의 새로운 출력을 발생시킨다. 이 과정에서 $e_{x_k}(t_n)$와 연결된 $w_{x_{jk}}$는 다른 에러에도 영향을 주지만, k번째 에러 $e_{x_k}(t_{n+1})$에 가장 많은 영향을 준다. 따라서 본 논문에서는 $w_{x_{jk}}$를 수정하기 위하여 k번째 에러 $e_{x_k}(t_{n+1})$만을 역전파한다. 그러므로 입력층 뉴런 k와 은닉층 뉴런 j의 연결강도 $w_{x_{jk}}$를 조정하기 위하여 k번째 변수의 목적함수를 최소화하는 방향으로 연결강도를 조정하여야 한다.

$$\Delta w_{x_{jk}}(t_n) = -\eta \frac{\partial c_k}{\partial w_{x_{jk}}(t_n)} \tag{6}$$

여기서 η는 학습율이다. 연결강도는 은닉층으로 전달되므로, 위 식은 다음과 같이 나타낼 수 있다.

$$\Delta w_{x_{jk}}(t_n) = -\eta \frac{\partial c_k\,(t_{n+1})}{\partial NET_{z_{x_j}}(t_n)} \frac{\partial\, NET_{z_{x_j}}(t_n)}{\partial w_{x_{jk}}(t_n)} \tag{7}$$

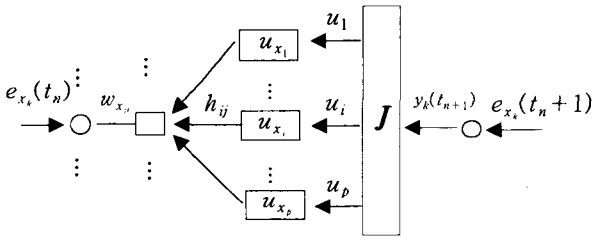

Fig. 5 Backpropagation of $w_{x_{jk}}$

은닉층 뉴런의 오차 신호 $\delta_{z_{x_j}}(t_n)$을 다음과 같이 정의한다.

$$\delta_{z_{x_j}}(t_n) \equiv -\frac{\partial c_k\,(t_{n+1})}{\partial\, NET_{z_{x_j}}(t_n)} \tag{8}$$

$\frac{\partial\, NET_{z_{x_j}}(t_n)}{\partial w_{x_{jk}}(t_n)}$은 $e_{x_k}(t_n)$이 되므로, 식(9)는 은닉층 뉴런의 오차신호 정의에 의해

$$\Delta w_{x_{jk}}(t_n) = \eta\, \delta_{z_{x_j}}(t_n) e_{x_k}(t_n) \tag{9}$$

이 되고, $NET_{z_{x_j}}(t_n)$는 z_{x_j}의 함수이고, z_{x_j}는 연결강도 $w_{x_{ij}}(t_n)$와 곱하여 출력층 $u_x(t_n)$로 전달되므로

$$\delta_{z_{x_j}}(t_n) = -\sum_{i=1}^{N}\left\{\frac{\partial c_k}{\partial u_{x_i}(t_n)} \frac{\partial u_{x_i}(t_n)}{\partial NET_{u_{x_i}}(t_n)} \frac{\partial NET_{u_{x_i}}(t_n)}{\partial z_{x_j}(t_n)}\right\} \frac{\partial z_{x_j}(t_n)}{\partial NET_{z_{x_j}}(t_n)} \tag{10}$$

이 된다. 식(10)에서 $\frac{\partial z_{x_j}(t_n)}{\partial NET_{z_{x_j}}(t_n)}$와 $\frac{\partial u_{x_i}(t_n)}{\partial NET_{u_{x_i}}(t_n)}$는 활성화 함수의 미분으로 표현할 수 있다. 그리고 $\frac{\partial NET_{u_{x_i}}(t_n)}{\partial z_{x_j}(t_n)}$는 $w_{x_{ij}}(t_n)$이므로

$$\delta_{z_{x_j}}(t_n) = -\sum_{i=1}^{N}\left\{\frac{\partial (j_c)_k\,(t_{n+1})}{\partial u_{x_i}(t_n)} w_{x_{ij}}(t_n) F'[NET_{u_{x_i}}(t_n)]\right\} F'[NET_{z_{x_j}}(t_n)] \tag{11}$$

이 된다. 출력층의 출력 $u_{x_i}(t_n)$는 플랜트의 입력 $u_i(t_n)$으로 들어가서 새로운 플랜트 출력에 가장 많은 영향을 주므로 다음과 같이 나타낼 수 있다.

$$\delta_{z_{x_j}}(t_n) = \phi_{x_k} e_{x_k}(t_{n+1}) F'[NET_{z_{x_j}}(t_n)] \sum_{i=1}^{N}\left\{w_{x_{ij}}(t_n) \frac{\partial y_k(t_{n+1})}{\partial u_i(t_n)} F'[NET_{u_{x_i}}(t_n)]\right\} \tag{12}$$

식(13)에서 시스템의 입력과 출력의 관계를 나타내는 자코비안을 정의한다.

$$J \equiv \frac{\partial \boldsymbol{y}(t_{n+1})}{\partial \boldsymbol{u}(t_n)} = \begin{bmatrix} \frac{\partial y_1(t_{n+1})}{\partial u_1(t_n)} & \cdots & \frac{\partial y_1(t_{n+1})}{\partial u_p(t_n)} \\ \vdots & & \\ \frac{\partial y_q(t_{n+1})}{\partial u_1(t_n)} & \cdots & \frac{\partial y_q(t_{n+1})}{\partial u_p(t_n)} \end{bmatrix} \quad (13)$$

따라서 식(12)의 $\frac{\partial y_k(t_{n+1})}{\partial u_i(t_n)}$ 는 i번째 입력에 대한 k번째 출력의 자코비안이다. 식(12)를 식(9)에 대입하면 연결강도의 조정 식을 구할 수 있다.

$$\Delta v_{x_{jk}}(t_n) = \eta \phi_{x_k} e_{x_k}(t_n) e_{x_k}(t_{n+1}) F'[NET_{z_j}(t_n)] \sum_{i=1}^{N} \left\{ w_{x_{ij}}(t_n) \frac{\partial y_k(t_{n+1})}{\partial u_i(t_n)} F'[NET_{u_i}(t_n)] \right\} \quad (14)$$

- 은닉층과 출력층의 연결강도 조정

은닉층과 출력층의 연결강도 조정은 일반적인 신경망에서 처럼 모든 에러를 역전파한다. 따라서 연결강도 조정 식을 구할 수 있다.

$$\Delta w_{x_{ij}}(t_n) = \eta z_{x_j}(t_n) F'[NET_{u_i}(t_n)] \sum_{k=1}^{M} \left\{ \phi_{x_k} e_k(t_{n+1}) \frac{\partial y_k(t_{n+1})}{\partial u_i(t_n)} \right\} \quad (15)$$

미분제어 신경망도 비례 제어 신경망과 같은 방법으로 입력층과 은닉층의 연결강도와 은닉층과 출력층의 연결강도의 조정 식을 구할 수 있다. 출력의 속도에 대한 자코비안 J_v를 사용한다.

3. 자코비안

플랜트의 이산시간에서 상태 방정식은 다음 식과 같다.

$$\boldsymbol{x}(t_{n+1}) = \boldsymbol{\Phi x}(t_n) + \boldsymbol{\Gamma u}(t_n)$$
$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{H x}(t_n) \quad (16)$$

여기서, $\boldsymbol{\Phi}$, $\boldsymbol{\Gamma}$, $\boldsymbol{H}$는 시스템 행렬이다. 아래 정리는 선형 시스템의 자코비안을 증명하였다.

정리 1

이산 시간에서 선형 시스템의 자코비안은

$$\boldsymbol{J} = \boldsymbol{\Gamma H} \quad (17)$$

으로 구해진다.

증명

식(13)의 자코비안 정의를 이용하여 이산 시간의 선형 시스템에서 자코비안을 다음과 같이 구할 수 있다.

$$\boldsymbol{J} = \frac{\partial \boldsymbol{y}(t_{n+1})}{\partial \boldsymbol{u}(t_n)} = \frac{\partial [\boldsymbol{H\Phi x}(t_n) + \boldsymbol{H\Gamma u}(t_n)]}{\partial \boldsymbol{u}(t_n)} \quad (18)$$

위 식을 풀면 다음과 같다.

$$\boldsymbol{J} = \boldsymbol{\Gamma H} \quad \blacksquare$$

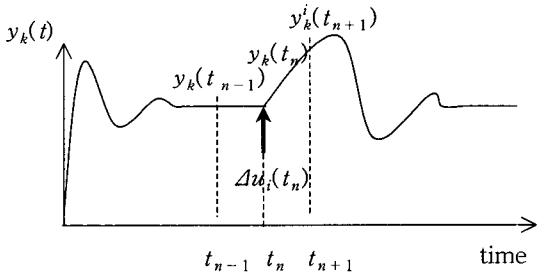

Fig. 6 Estimation of Jacobian

시스템의 운동 방정식을 아는 경우에는 운동 방정식을 이용하여 자코비안을 구하지만, 시스템의 운동 방정식을 모르는 경우에는 실험을 통하여 추정 자코비안을 구해야 한다. 이장에서 운동 방정식을 모르는 안정한 시스템의 자코비안을 실험적인 방법을 통해 근사적으로 구하는 방법을 제시한다. Fig. 6은 안정한 시스템에 입력을 가하였을 때 k번째 출력 $y_k(t_n)$을 나타낸 것이다. Fig. 6에서 보는 것처럼 입력을 가한 후 일정한 시간이 흐르면 출력

은 정상 상태에 도달한다. 이 정상 상태에서 i번째 입력 $\Delta u_i(t_n)$을 임펄스(impulse)로 가한다. 그리고 한 스텝 후에 출력, 즉 i번째 입력의 영향을 받은 k 번째 출력은 $y_k^i(t_{n+1})$이 된다. 이 값들과 자코비안의 관계를 유도하여 자코비안을 근사적으로 구하는 방법을 제시하려고 한다.

$\boldsymbol{x}(t_n)$와 $\boldsymbol{y}(t_n)$은 상태 변수 $\boldsymbol{x}(t_{n-1})$으로 나타낼 수 있다.

$$\boldsymbol{x}(t_n) = \boldsymbol{\Phi}\,\boldsymbol{x}(t_{n-1}) + \boldsymbol{\Gamma}\boldsymbol{u}_0 \tag{19}$$

$$\begin{aligned}\boldsymbol{y}(t_n) &= \boldsymbol{H}\,\boldsymbol{x}(t_n)\\ &= \boldsymbol{H}\,\boldsymbol{\Phi}\,\boldsymbol{x}(t_{n-1}) + \boldsymbol{H}\,\boldsymbol{\Gamma}\,\boldsymbol{u}_0\end{aligned} \tag{20}$$

여기서 $\boldsymbol{u}_0$는 초기 입력 상수 값이다. 시간 t_n에서 i번째 입력 변화량 $\Delta u_i(t_n)$을 가하였다. 이 입력 변화량에 영향으로 시간 t_{n+1}에서 상태 변수 $\boldsymbol{x}^i(t_{n+1})$와 출력 $\boldsymbol{y}^i(t_{n+1})$을 구할 수 있다.

$$\begin{aligned}\boldsymbol{x}(t_{n+1}) &= \boldsymbol{\Phi}\,\boldsymbol{x}(t_n) + \boldsymbol{\Gamma}\boldsymbol{u}_0\\ &\quad + \boldsymbol{\Gamma}[\,0\ \cdots\ \Delta u_i(t_n)\ \cdots\ 0\,]\end{aligned} \tag{21}$$

$$\begin{aligned}\boldsymbol{y}^i(t_{n+1}) &= \boldsymbol{H}\,\boldsymbol{x}(t_{n+1})\\ &\quad + \boldsymbol{\Gamma}[\,0\ \cdots\ \Delta u_i(t_n)\ \cdots\ 0\,]\end{aligned} \tag{22}$$

시간 t_{n-1}과 t_n에서 정상 상태이므로 출력은 $\boldsymbol{y}(t_n) = \boldsymbol{y}(t_{n-1})$이고, 상태 변수 역시 $\boldsymbol{x}(t_n) = \boldsymbol{x}(t_{n-1})$이다. 따라서 식(22)는 식(20)과 정리1에 의해 다음과 같이 나타낼 수 있다.

$$\begin{aligned}\boldsymbol{y}^i(t_{n+1}) &= \boldsymbol{H}\,\boldsymbol{\Phi}\,\boldsymbol{x}(t_{n-1}) + \boldsymbol{H}\boldsymbol{\Gamma}\boldsymbol{u}_0\\ &\quad + \boldsymbol{H}\boldsymbol{\Gamma}[\,0\ \cdots\ \Delta u_i(t_n)\ \cdots\ 0\,]^T\\ &= \boldsymbol{y}(t_n) + \boldsymbol{J}\,[\,0\ \cdots\ \Delta u_i(t_n)\ \cdots\ 0\,]^T\end{aligned} \tag{23}$$

위 식은 $\boldsymbol{y}^i(t_{n+1})$와 $\boldsymbol{y}(t_n)$의 차로 나타내어 정리하면

$$\begin{aligned}\boldsymbol{y}^i(t_{n+1}) - \boldsymbol{y}(t_n) &= \boldsymbol{J}\,[\,0\ \cdots\ \Delta u_i(t_n)\ \cdots\ 0\,]^T\\ &= \Delta u_i(t_n)\frac{\partial\,\boldsymbol{y}(t_{n+1})}{\partial u_i(t_n)}\end{aligned} \tag{24}$$

이 되므로 자코비안을 구할 수 있다.

$$\begin{aligned}\boldsymbol{J} &= \begin{bmatrix} \dfrac{\boldsymbol{y}^1(t_{n+1}) - \boldsymbol{y}(t_n)}{\Delta u_1(t_n)} & \cdots & \dfrac{\boldsymbol{y}^p(t_{n+1}) - \boldsymbol{y}(t_n)}{\Delta u_p(t_n)} \end{bmatrix}\\ &= \begin{bmatrix} \dfrac{y_1^1(t_{n+1}) - y_1(t_n)}{\Delta u_1(t_n)} & \cdots & \dfrac{y_1^p(t_{n+1}) - y_1(t_n)}{\Delta u_p(t_n)} \\ & \vdots & \\ \dfrac{y_q^1(t_{n+1}) - y_q(t_n)}{\Delta u_1(t_n)} & \cdots & \dfrac{y_q^p(t_{n+1}) - y_q(t_n)}{\Delta u_p(t_n)} \end{bmatrix}\end{aligned} \tag{25}$$

입력에 대한 출력 변화율에 대한 자코비안도 위와 같은 방법으로 구할 수 있다.

4. 시뮬레이션

본 논문에서 제안한 신경망 제어기를 다변수 시스템인 수중익 쌍동선에 적용하였다. 수중익 쌍동선은 Fig. 7에서 보는 것처럼 두개의 날개가 수중에서 상하로 움직여서 수중익 쌍동선의 상하 진동과 종방향 진동을 제어한다. 식(26)은 수중익 쌍동선의 상태 방정식을 나타내었다[3].

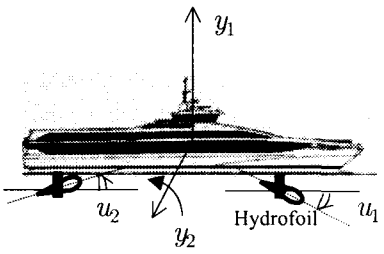

Fig. 7 Schematic diagram of Hydrofoil catamaran

$$\begin{bmatrix}\dot{x}_1\\ \ddot{x}_1\\ \dot{x}_2\\ \ddot{x}_2\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ -109.209 & -4.190 & -0.237 & -5.691\\ 0 & 0 & 1 & 0\\ -26.452 & -3.539 & -6.432 & -94.102\end{bmatrix}\begin{bmatrix}x_1\\ \dot{x}_1\\ x_2\\ \dot{x}_2\end{bmatrix}$$

$$+\begin{bmatrix} 0 & 0 \\ 1.680 & 1.302 \\ 0 & 0 \\ -5.914 & 7.613 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}[x_1,\ \dot{x_1},\ x_2,\ \dot{x_2}]^T \tag{26}$$

여기서 $y_1(t)$은 상하 진동 변위이고, $y_2(t)$는 피치 각도이다. 제안한 신경망 제어기의 성능을 LQR 제어기의 성능과 비교하려고 한다.

LQR 제어기의 제어 입력은

$$\boldsymbol{u} = -\boldsymbol{Kx} \tag{27}$$

이며, $\boldsymbol{K}$는 제어 게인 행렬이다. LQR 제어기를 설계하기 위한 상태 가중행렬을 식(28)과 같이 선정하였다[3].

$$\boldsymbol{Q} = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix},\quad \boldsymbol{R} = \begin{bmatrix} 1.0 & 0 \\ 0 & 0.8 \end{bmatrix} \tag{28}$$

위 상태 가중치를 이용하여 피드백 게인을 Matlab의 'LQR' 함수를 이용하여 구하였다.

$$\boldsymbol{K} = \begin{bmatrix} 3.0649 & 4.7635 & -3.1602 & -1.6825 \\ 4.3914 & 3.5900 & 3.6632 & 2.4160 \end{bmatrix} \tag{29}$$

위 식에서 구한 LQR 제어기와 모델 식을 모른다고 가정하고 실험적인 방법으로 자코비안 $\boldsymbol{J}$와 속도 자코비안 $\boldsymbol{J_v}$을 구하였다.

$$\boldsymbol{J} = 10^{-2}\begin{bmatrix} 0.0083 & 0.0063 \\ -0.0290 & 0.0369 \end{bmatrix} \tag{30}$$

$$\boldsymbol{J_v} = 10^{-2}\begin{bmatrix} 1.6491 & 1.2636 \\ -5.7481 & 7.3397 \end{bmatrix} \tag{31}$$

신경망은 3계층 신경망을 사용하였으며, 비례 제어 신경망은 입력층의 뉴런의 개수는 2개(e_{x_1}, e_{x_2})이고, 은닉층의 뉴런의 개수는 20개이고, 출력층 뉴런의 개수는 2개(u_{x_1}, u_{x_2})를 사용하였으며, 미분 제어 신경망도 같은 개수로 하였다. 신경망 제어기를 학습하기 위하여 학습 변수인 가중치를 결정하여야 한다. 가중치에는 에러에 대한 가중치 ϕ_{x_1}, ϕ_{x_2}와 에러 변화율에 대한 가중치 ϕ_{v_1}, ϕ_{v_2}가 있다. 시스템의 모델을 모르기 때문에 가중치 결정은 플랜트의 응답을 이용하여 구한다. Fig. 8은 제어 전의 응답을 나타내었다. Fig. 8에서 보듯이 $y_1(t)$, $y_2(t)$의 상승 시간은 비슷하기 때문에 에러에 대한 가중치를

$$[\phi_{x_1}\ \phi_{x_2}] = [100\ 100] \tag{32}$$

로 정하였다. 그리고 Fig. 8에서 과도응답의 크기를 비교하면, $y_1(t)$의 과도응답이 $y_2(t)$의 과도 응답보다 10배 이상 크다는 것을 알 수 있다. 따라서 ϕ_{v_1}=15, ϕ_{v_2}=1로 결정하였다. 에러 변화율의 가중치는 에러에 비해 상대적으로 작게 해야 한다. 이 값이 크게 되면 상승 시간이 많이 걸린다. 에러 변화율의 가중치는

$$[\phi_{v_1}\ \phi_{v_2}] = 10^{-2}[15.0\ 1.0] \tag{33}$$

으로 선정하였으며, 학습율은 η=1으로 하였다. 초기 값은 $[y_1(t_0), \dot{y}_1(t_0), y_2(t_0), \dot{y}_2(t_0)]$=[0.5, 0.0, 0.5, 0.0]으로 하여 반복 학습을 하였다. Fig. 9는 비

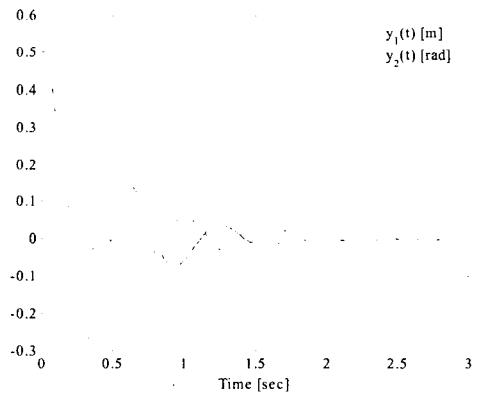

Fig. 8 Response without controller

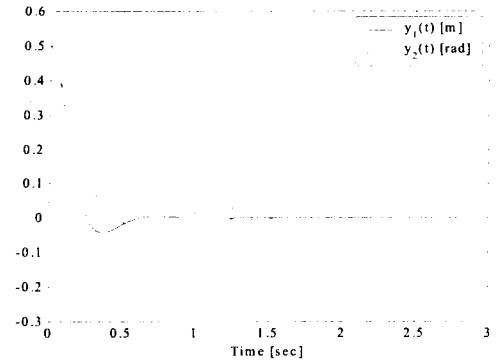

Fig. 9 Response of generalized neural network controller

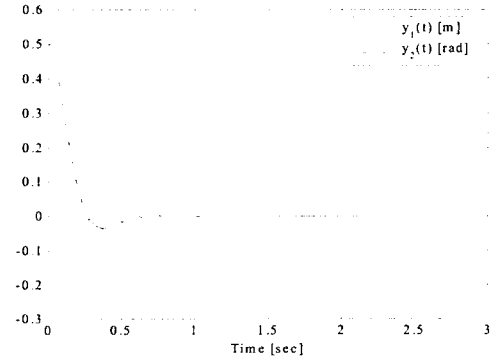

Fig. 10 Response of proposed NNC

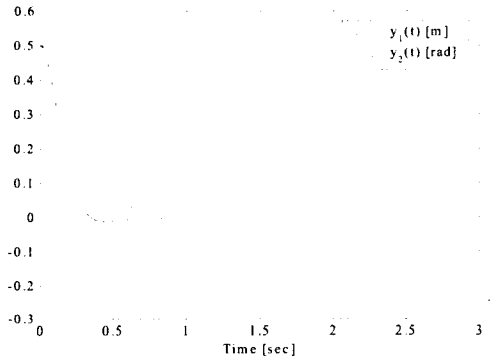

Fig. 11 Response of LQR controller

레 신경망과 미분 신경망으로 분리하지 않은 일반적인 신경망의 학습 결과를 나타내었다. Fig. 10은 제안한 신경망을 이용하여 학습한 결과를 나타내었다. 두 결과에서 제안한 신경망이 더 좋음을 알 수 있다. Fig. 11은 LQR 제어기의 응답을 나타내었다. 두 응답을 비교해보면 제안한 신경망 제어기의 결과가 LQR 제어기의 결과 보다 나쁘지 않음을 알 수 있다. 제안한 신경망 제어기를 궤도 추적 문제에 적용하였다. 가중치를

$$[\phi_{x_1}\ \phi_{x_2}]=[100\ 100]$$
$$[\phi_{v_1}\ \phi_{v_2}]=[0.01\ 0.01] \quad (34)$$

으로 하여 학습했을 때, $y_1(t)$이 정상 상태 추적에러를 가지고 있었다. 그래서 가중치를 다음과 같이 수정하여 학습하였다.

$$[\phi_{x_1}\ \phi_{x_2}]=[200\ 100]$$
$$[\phi_{v_1}\ \phi_{v_2}]=[2\times10^{-2}\ 10^{-2}] \quad (35)$$

그 결과를 Fig. 12에 나타내었다. Fig. 12에서 보듯이 목표 궤도를 잘 추적하는 것을 확인할 수 있었다.

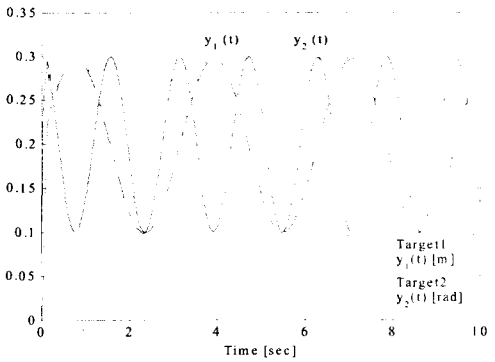

Fig. 12 Tracking response of proposed NNC

5. 결론

지금까지는 다중 입출력 시스템의 신경망제어기를 학습하기 위해 출력 에러를 입력 에러로 변환하는 신경망 모델의 학습이 선행되어야 했다. 그러나 본 논문에서는 플랜트의 자코비안을 이용해 신경망 제어기를 학습하는 방법을 제안함으로써 신경망 모델을 사용하지 않고도 제어기를 학습하는 방법을 제안하였다. 또 PID 제어에서와 같이 최적의

신경망 제어기를 설계하기 위하여 비례 제어 신경망과 미분 제어 신경망으로 분리된 신경망 구조를 제안하였다. 그리고 제어기의 학습에서 은닉층과 출력층의 연결강도는 기존의 역전파 학습법을 사용한 반면 입력층과 은닉층의 연결강도 조정에서는 입력층에 연결된 에러만을 역전파하였다. 이렇게 함으로써 제안한 신경망 구조는 PD제어를 하는데 있어서 기존의 신경망에 비해 구조적으로 많은 메모리를 절약하고 학습시간을 단축하였을 뿐만 아니라 성능면에서도 아주 우수함을 입증하였다. 즉, 제안한 신경망 제어기의 성능을 검증하기 위하여 LQR 제어기와 비교한 결과 제안한 신경망 제어기가 정확한 모델식을 이용한 LQR 제어기에 비해 전혀 뒤지지 않는다는 것을 확인하였고 추적 문제에서도 제안한 신경망 제어기의 학습이 무난하였고 학습된 제어기는 목표 궤도를 잘 따라가는 것을 확인하였다.

후 기

이 논문은 2002년도 두뇌한국 21 사업에 의하여 지원되었음.

참고문헌

1. W. Gharieb and G. Nagib, "Fuzzy control to mu-ltivariable systems case study:Helicopter Model," Fuzzy-IEE, Vol. 1, pp. 400-405, September 1996.
2. S. Arimoto, S. Kawamura, and F. Miyazaki, "Bet-tering operation of robots by learning," Journal of Robotic Systems, Vol. 1, pp. 123-140, 1984.
3. 이심용, "수중익 쌍동선의 자세제어 시스템에 관한 이론적 및 실험적 연구," 서울대학교, 박사학위논문, 1999.
4. 임윤규, 정병묵, 소범식, "불안정한 다변수 시스템에 대한 퍼지 학습 제어," 제어 · 자동화 · 시스템공학 논문지, 제5권, 제7호, pp. 808-813, 1999.
5. 임윤규, 정병묵, "퍼지학습법을 이용한 크레인 시스템의 다변수 제어," 한국정밀공학회지, 제16권, 제7호, pp. 144-150, 1999.
6. B. Chung, J. Lee, H. Joo and Y. Lim, "Hybrid Fuzzy Learning Controller for an Unstable Nonli-near System," J. of the KSPE, Vol. 1, pp. 79-84, 2000.
7. T. J. Procyk, "A Self-Organizing Controller For Dynamic Processes," Ph. D. Thesis, Queen Mary College, Univ. of London, 1977.
8. T. J. Procyk and E. H. Mandani, "A lingustic self-organizing process controller," Automatica, Vol. 15, No. 1, pp. 15-30, 1979.
9. R. M. Sanner and J. -J. E. Slotine, "Gaussian networks for direct adaptive control," IEEE Trans. Neural Networks, Vol. 3, pp. 837-863, 1992.
10. F. C. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," IEEE Control Sys. Mag.(Special issue on Neural Networks for Control Systems), Vol. 10, pp. 44-48, 1990.
11. F. C. Chen and H. K. Khalil, "Adaptive control of nonlinear systems using neural networks," Int. J. Control, Vol. 55, pp. 1299-1317, 1992.
12. S. -H. Yu and A. M. Annaswamy, "Adaptive control of nonlinear dynamic systems using θ -adaptive Neural Networks," Automatica, Vol. 33, No. 11, pp. 1975-1995, 1997.
13. B. Chung and J. Oh, "Control of dynamic systems using fuzzy learning algorithm," Fuzzy sets and Systems, Vol. 59, pp. 1-14, 1993.
14. B. Chung and J. Oh, "Autotuning method of membership function in a fuzzy learning control," Journal of Intelligent & Fuzzy Systems, Vol. 1, pp. 335-349, 1993.
15. 임윤규, 정병묵, "기울기법을 이용한 최적의 PID 제어 학습법," 한국정밀공학회지, 제18권, 제1호, 2001.