'93 한국자동제어학술회의논문집 (1993.10.20~22)

고급 분산 제어시스템을 위한 신경 회로망 제어 알로리즘의 개발

° 이승준, 박세화, 박동조, 김병국, 변증남
한국과학기술원 전기 및 전자공학과

Development of Neural Network Algorithm for an Advanced Distributed Control System

° S.-J. Lee, S.-H. Park, D.-J. Park, B.-K. Kim, Z. Bien
Department of Electrical Engineering, KAIST

Abstract

We develop a neural network control algorithm for the ACS (Advanced Control System). The ACS is an extended version of the DCS (Distributed Control System) to which functions of fault detection and diagnosis and advanced control algorithms are added such as neural networks, fuzzy logics, and so on. In spite of its usefulness proven by computer simulations, the neural network control algorithm, as far as we know, has no tool which makes it applicable to process control. It is necessary that the neural network controller should be turned into the function code for its application to the ACS. So we develop a general method to implement the neural network control systems for the ACS. By simulations using the simulator for the boiler of 'Seoul fire power plant unit 4', the methodology proposed in this paper is validated to have the applicability to process control.

1 서론

1970년대 초반 컴퓨터 기술의 발달로 인해 등장한 마이크로 프로세서 기술은 산업 공정 제어 분야에 큰 영향을 미쳐 오늘날의 디지탈 분산 제어 시스템(Digital Control System, DCS)을 출현케 했다. 국내(Korea Advanced Institute of Science and Technology, KAIST)에서도 개발한 예를 참조 문헌 [1, 2]에서 찾아볼 수 있다. 분산 제어 시스템은 발전소와 같은 대규모이면서 복잡한 구조를 갖는 산업 공정 제어를 위해 널리 보급되어 사용되고 있다. 일반적으로 분산 제어 시스템의 구성은 직접적으로 플랜트의 제어를 담당하는 PCS(Process Control System), PCS를 위한 엔지니어링 작업을 담당하는 EWS(Engineering Work-Station), 그리고 PCS가 보낸 플랜트의 공정 제어 정보를 그래픽 처리하여 운영자에게 보여주는 OIS(Operator Interface Station)로 이루어진다. 또한 고급 분산 제어 시스템(Advanced Control System, ACS)이라 함은 위와 같은 기존의 분산 제어 시스템에 예측 제어, 신경 회로망 제어 등의 최근 기법인 고급 제어 알고리즘과, 플랜트 및 제어 시스템의 고장 정보와 그 대책을 제공하는 기능이 부과된 확장된 개념의 분산 제어 시스템을 지칭한다.

한편 고전적 제어 기법의 한계점을 극복하기 위해 신경 회로망을 이용한 지능 제어 기법의 연구가 활발하게 진행되고 있으며 그 유용성도 실험적으로 입증되고 있다[4, 5, 6, 7, 8]. 신경 회로망을 이용하면 고전적 제어 기법에서 필수적으로 요구하는 대상 시스템의 수학적 모델링을 피할 수 있다. 또한 신경 회로망의 입출력 함수 관계가 비선형이므로 비선형 제어 문제에서 큰 기대를 걸고 있다. 그런데 신경 회로망 제어 알고리즘이 ACS에 도입되어 실제 공정 제어에 사용되기 위해서는 모듈화된 형태의 구현이 필요하다. 하지만 아직은 신경 회로망 제어 기법을 대상 플랜트에 적용하는데 있어 주로 컴퓨터 시뮬레이션으로만 그치고, 실제 프로세서에 올려서 활용할 수 있는 일반적인 구현 방법의 연구는 미흡한 형편이다. 따라서 본 논문에서는, 신경 회로망 제어기가 범용으로 사용할 수 있는 기능 코드 형태로 구현됨으로서, 고급 분산 제어 시스템에 사용되기 위한 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서 신경 회로망 제어 시스템의 장점을 간단히 언급한 뒤, 신경 회로망 제어 시스템을 구성하는 몇 가지 방법과 본문에서 채택한 구성 방법을 소개한다. 3장에서는 대규모 공정 제어에서 제어기를 구현하기 위한 일반화된 형태인 기능 코드에 대해 기술한다. 4장에서는 신경 회로망 제어 알고리즘을 구현하기 위한 구체적인 방법을 제안한다. 5장에서는 서울 화력 4호기 보일러 시뮬레이터를 대상으로 모의 실험을 통해 4장에서 제안한 방법의 적용 가능성을 확인한다. 6장에서는 앞의 내용들을 정리하고 추후 연구되어야할 내용에 대해 요약한다.

2 신경 회로망을 이용한 제어 시스템

제어하고자 하는 시스템이 거대하고 복잡해짐에 따라 고전적 제어 기법만으로는 감당하기 어려우므로 보다 향상된 고급의 제어 기법이 요구되고 있다. 고전적 제어 기법은 대상 플랜트의 모델에 근거해서 단계적 계산을 거쳐 수학적인 제어 알고리즘을 도출하므로

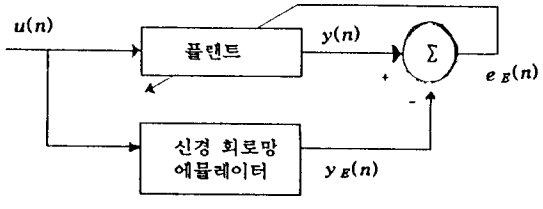
제어 대상에 대한 수학적 모델링이 필요하다. 그러나 실제는 측정 불가능한 파라메타들과 외란이나 잡음등의 불확정성으로 인해 대상 플랜트를 정확하게 모델링하는 것이 매우 어려운 경우가 많다. 최근에는 신경 회로망을 이용한 지능 제어 기법을 통해 고전적 제어 기법의 한계성을 극복하려는 노력과 성과가 활발하다. 신경 회로망을 이용한 제어 시스템의 주요 장점들을 열거하면 다음과 같다.

- 신경 회로망은 비선형 제어 문제에서 큰 기대를 끌고 있다. 이것은 신경 회로망이 임의의 비선형 함수를 근사화할 수 있다는 이론적 배경에 기인한다.
- 신경 회로망은 병렬 분산 처리 구조를 갖고 있다. 한 개의 뉴런의 기능은 단순하므로 신경 회로망이 병렬 구조로 구현되었을 경우에는 처리 속도가 아주 빠르다. 또한 병렬 분산 구조는 고장 허용(fault tolerance)을 좋게 한다.
- 신경 회로망은 대상 시스템의 관련 데이타로 부터 그 시스템의 특성을 학습할 수 있다. 적절하게 학습된 신경 회로망은 학습에 사용되지 않은 입력에 대해서도 올바른 출력값을 유추할 수 있는 일반화(generalization) 능력을 갖고 있다.

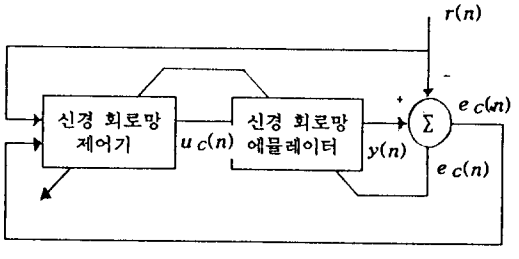
인공 신경 회로망은 뉴런이라 불리는 기본 처리기들이 목적에 따라 특정 모양으로 연결되어 이루어 진다. 각각의 뉴런들은 다른 뉴런들의 출력값에 가중치를 두어 더해주는 덧셈기와 덧셈기의 결과로 부터 자신의 출력값을 계산하는 activation 함수로 구성된다. 인공 신경 회로망은 그 종류가 50 가지를 넘는데 뉴런들의 연결 구조에 따라 크게 recurrent 네트워크와 nonrecurrent 네트워크의 두 부류로 나뉜다. recurrent 네트워크의 대표적 예로는 Hopfield 네트워크와 Cellular 네트워크를 들 수 있으며, nonrecurrent 네트워크의 대표적 예로는 순방향 신경 회로망(feedforward neural 네트워크)을 들 수 있겠다.

신경 회로망을 제어에 적용하는 방법도 다양하게 소개되어 왔다. Psaltis[4]는 일반 학습(general learning) 방법과 특별화된 학습(specialized learning) 방법이라는 두가지 방법을 제시했다. 일반 학습 방법에서는 신경 회로망이 대상 플랜트의 역 동특성(inverse dynamics)를 학습하게 된다. 특별화된 학습 방법에서는 신경 회로망 제어기를 학습시키기 위해서 플랜트를 통해 오차가 역전파된다. 플랜트에서의 오차 역전파를 위해서는 플랜트의 Jacobian이 사용되도록 하고 있다. Nguyen[5]과 Wu[6]는 두 개의 신경 회로망을 이용하는 방법을 제시했다. 첫번째 신경 회로망인 신경 회로망 에뮬레이터(emulator)는 대상 플랜트의 순방향 동특성을 학습하고 두번째 신경 회로망인 신경 회로망 제어기는 신경 회로망 에뮬레이터를 제어하도록 학습된다. 이것은 실제 대상 플랜트를 통해 오차가 역전파되는 것을 피하기 위함이다. 신경 회로망 에뮬레이터를 적절하게 제어하도록 학습된 신경 회로망 제어기는 실제 대상 플랜트를 제어하는데 바로 사용될 수 있다. Narendra[7]는 미지의 비선형 다이나믹(dynamic) 시스템을 동정(identification)하고 제어하기 위한 일반화된 신경 회로망 모델을 제안했다. 일반화된 신경 회로망 모델의 학습을 위해서는 동적(dynamic) 역전파 방법이 사용된다. 플랜트의 제어를 위해서는 모델 기준(model reference)적응 제어에 기초를 둔 직접 방법과 간접 방법을 제시했다. 하지만 이 방법에서는 대상 플랜트의 선형 동특성(linear dynamics) 부분은 알고 정적 비선형 함수(nonlinear static mapping) 부분만 모른다고 가정하고 있다. Chen[8]은 자기 동조(self-tuning) 제어에 신경 회로망을 응용하여서 되먹임 선형 가능한(feedback linearizable) 시스템을 제어할 수 있음을 보여주었다.

위에서 소개한 방법들 중에서 실제 적용에 제약성을 가장 적게 받으면서 폭넓게 사용될 수 있는 것은 Nguyen이 사용했던 신경 회로망 에뮬레이터와 신경 회로망 제어기로 이루어진 제어 시스템이다. 이 방법은 대상 플랜트에 대한 입출력 데이타만 주어진다면 플랜트에 대한 특별한 사전 정보나 가정 없이 적용할 수 있다. 따라서, 본 논문에서 다루는 신경 회로망 제어 시스템도 이 구조에 기초하고 있다. <그림 1>은 두 개의 신경 회로망의 학습 구조를 보여준다. <그림 1>의 (가)에서는 플랜트의 입출력 데이타를 이용해 플랜트의 동특성을 신경 회로망 에뮬레이터에 학습시키는 구조가 나와있다. <그림 1>의 (나)에서는 신경 회로망 에뮬레이터의 학습이 끝난 후 학습된 신경 회로망 에뮬레이터를 이용해 신경 회로망 제어기를 학습시키는 것을 나타낸다.



(가) 신경 회로망 에뮬레이터의 학습



(나) 신경 회로망 제어기의 학습

그림 1: 신경 회로망의 학습 구조

3 기능 코드를 이용한 제어 언어

대부분의 공정 제어 형태는 다중 루프(loop)로 이루어진다. 다중 루프 제어기는 여러개의 PID 제어 기능 외에 적응 필터링(filtering), 선행/지연(lead/lag) 보상, 신호 처리 등의 다양한 기능을 내장해야 한다. 대규모의 공정 제어를 위해서는 제어기 구성에 있어서 모듈(module)화에 의해 확장성을 갖으면서 복잡한 시스템 구성을 용이하게 해주는 장치가 필요하다. 이러한 목적에 맞

개 개발된 것이 바로 기능 코드를 이용하는 configurable 제어기이다. configurable 제어기는 기능 블록 개념의 언어로 구성된다. 이 기능 블록들은 제어 다이어그램(diagram)에서 각각 어떤 특정 기능을 갖는 블록들에 해당하는데, 자주 사용되는 기능들은 개발자가 기능 서브루틴(subroutine)으로 미리 구현해 놓게 된다. 사용자는 이들을 유기적으로 결합하여 제어기를 구성하는데, 이 때 사용자는 기능 블록내의 구조를 알 필요가 없으며 단지 그 블록의 입력과 출력이 무엇인지만 이해하면 된다. 이런 견지에서 보면 위에서 사용하고자 하는 제어 언어는 문제 지향(problem oriented) 언어라 하겠다.

기능 블록은 기능 코드(function code)로 표현된다. 기능 코드는 기능 블록이 갖는 입출력 관계와 내재된 파라메타들을 사용자가 알기 쉽도록 아스키 문자로 표현된 것이다. 제어기의 구성은 이러한 기능 코드의 나열로서 이루어진다. 기능 코드의 나열로서 이루어진 아스키 파일(file)을 configuration 파일이라 한다. 사용자가 이해하기에 편하게 만들어진 configuration 파일은 목적 프로세서(target processor)가 알아볼 수 없으므로 목적 프로세서에 맞는 코드(code)로 바뀌어가는 작업이 필요하다. 실제 플랜트를 제어하기 위해서 PCS에서는 통신 테스크(task), 제어 테스크, 명령어 처리 테스크가 다중 처리되어야 한다. 특히 통신 테스크는 OIS로 부터 받은 명령들을 분석하여 PCS의 동작을 조절하게 된다. 이러한 테스크들을 위한 코드들이 생성되어야 한다.

목적 프로세서에 맞는 코드로 변환시켜주는 방법은 여러가지가 있을 수 있지만, Baily 형식[3]을 참조로 configuration 파일이 일단 널리 사용되는 컴퓨터 공용어인 C 언어로 변환되도록 하였다[2]. C 언어를 목적 프로세서에 맞는 코드로 바꿔주는 cross 컴파일러(compiler)는 일반적으로 쉽게 구할 수 있다. 기능 코드로 이루어진 configuration 파일을 C 언어로 변환시키기 위해서는 UNIX에서 제공하는 어휘 분석기(lexical analyzer) lex와 구문 및 어휘 해석기 (syntax/semantic analyzer) yacc이 사용된다. 제어 테스크 생성의 전체 흐름도를 <그림 2>에 도시화했다.

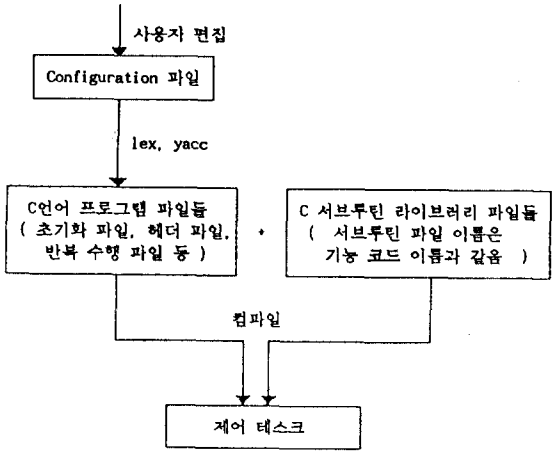


그림 2: 제어 테스크 생성의 전체 흐름도

4 신경 회로망 제어기의 개발

4.1 신경 회로망 제어기 구현을 위한 전체 과정

앞서 언급했다시피 본 논문에서 개발한 신경 회로망 제어 시스템은 두 개의 신경 회로망-신경 회로망 에뮬레이터와 신경 회로망 제어기-으로 이루어져 있으며, 두 신경 회로망의 학습은 EWS에서 오프 라인으로 행해진다. 대상 플랜트에 대한 입출력 학습 데이타가 주어지면, 신경 회로망 에뮬레이터는 이를 통해 대상 플랜트의 동특성을 학습하게 된다. 신경 회로망 에뮬레이터가 어느 정도 잘 학습되면, 다음에는 신경 회로망 에뮬레이터의 학습은 멈추고 신경 회로망 제어기의 학습이 진행된다. 신경 회로망 에뮬레이터와 신경 회로망 제어기의 학습을 위한 프로그램에서는 신경 회로망의 은닉층의 갯수(최대 3개)와 은닉 뉴런의 갯수, 입력 변수, 학습 횟수 등을 사용자가 선택할 수 있게 하였다. 또한 선행 처리 부분과 후행 처리 부분을 따로 두고 그 구조를 사용자가 정하도록 하였다. 신경 회로망 제어기의 학습까지 완료되면 학습 결과 얻어진 신경 회로망 제어기의 가중치 값들은 제어기 구성을 위해서 파일에 저장된다. 파일 형태로 저장된 신경 회로망 제어기의 가중치 값들은 기능 코드를 이용한 제어기 구성시에 초기화 과정에서 사용되거나 PCS에서 동작중인 신경 회로망 제어기의 가중치 값 갱신용으로 사용될 수 있다. 신경 회로망 제어기 구성 과정의 전체 흐름도가 <그림 3>에 나와있다.

4.2 신경 회로망 제어기를 위한 기능 코드

4.2.1 신경회로망기능코드

신경 회로망 기능 코드의 형태는 다음과 같다.

$y = \mathrm{nn}(N_i, N_o, x, y_{tr}, sw_{tr}, N_{h1}, N_{h2}, N_{h3}; \text{“}NNwght.dat\text{”})$

- y : 신경 회로망 출력 변수(벡터)
- N_i : 신경 회로망 입력 갯수(벡터 x의 크기 결정)
- N_o : 신경 회로망 출력 갯수(벡터 y의 크기 결정)
- x : 신경 회로망 입력 변수(벡터)
- y_{tr} : 트랙(track) 신호(벡터이며 그 크기는 벡터 y의 크기와 같다)
- sw_{tr} : 트랙 스위치
- N_{h1}, N_{h2}, N_{h3} : 첫번째 은닉층의 뉴런 갯수, 두번째 은닉층의 뉴런 갯수, 세번째의 은닉층 갯수
- NNwght.dat : 신경 회로망 제어기의 가중치 값이 저장되어 있는 파일 이름

트랙 스위치 sw_{tr}가 1이면 입력 x와 가중치를 토대로 신경회로망 출력값을 계산하여 y로 보낸지만, 트랙 스위치가 0이면 y_{tr}

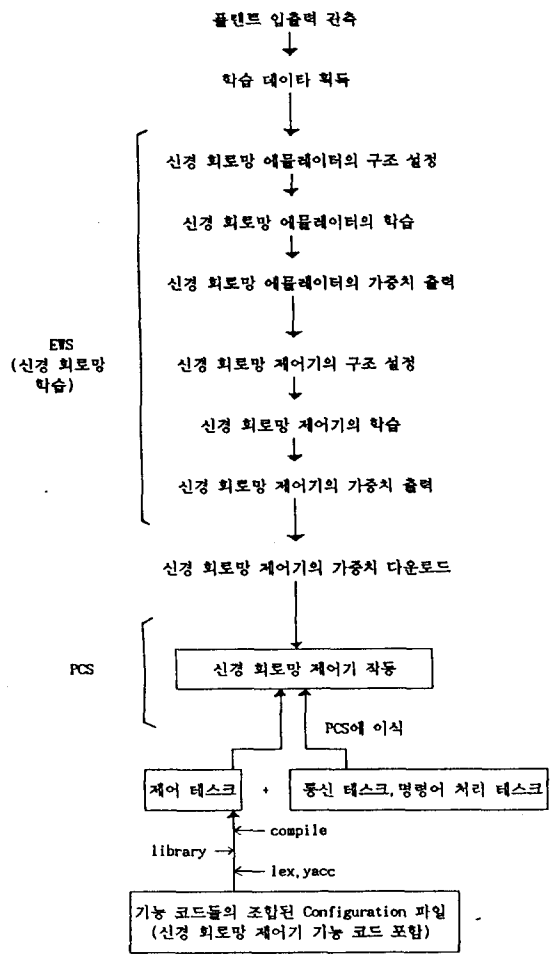


그림 3: 신경 회로망 제어 알고리즘 구성의 흐름도

값을 그대로 y로 보낸다. 은닉층 갯수의 최대값은 3으로 고정되었으며, 어떤 은닉층의 뉴런 갯수가 0으로 지정되면 해당 은닉층은 없는 것으로 간주된다.

신경 회로망 제어기를 구성하기 위해서 신경 회로망 함수를 나타내는 기능 코드외에 선행 처리기(pre-processor)와 후행 처리기(post-processor)를 위한 기능 코드가 설계되도록 하였다. 이것은 신경 회로망 함수의 기능 코드를 일반화시켜주기 위함이다.

4.2.2 C 파일의 생성

Configuration 파일로부터 lex와 yacc에 의해 C 파일들이 생성된다. 이렇게 해서 만들어진 파일들에서는 초기화 루틴 initnn()과 주기적으로 사용하는 루틴 nn() 이 사용된다. 이 두 함수는 include되는 파일에 미리 만들어져 있으며 그 형태는 다음과 같다.

$$\text{initnn}(N_i, N_o, N_{h1}, N_{h2}, N_{h3}, \text{ptr})$$

$$\text{nn}(y, N_i, N_o, x, y_{tr}, sw_{tr}, N_{h1}, N_{h2}, N_{h3}, \text{ptr})$$

초기화 함수 initnn() 은 신경 회로망 제어기의 가중치 값을 저장하고 은닉 뉴런의 출력값을 임시로 저장할 수 있도록 메모리를 할당한다. 생성되는 파일중에서 파라메타 값을 초기화하는 파일에는 initnn()에 의해 할당된 메모리에 가중치 값을 저장하는 명령어들이 놓이게 된다.

5 실험

5.1 실험 환경

실험 환경을 위해서는 OIS/EWS 기능을 갖는 호스트 컴퓨터로서 SUN 워크스테이션을 이용하였고, 공정 제어에 핵심이 되는 프로세서의 모듈로는 모듈 자체에 네트워크를 위한 ethernet 포트(port)가 내장되어 있으며 고속의 연산 기능을 가진 CPU 30을 이용하였다. 또한 서울 화력 4호기 발전소 보일러 시뮬레이터가 소프트 웨어로 구현되어 또다른 CPU 30 에 장착된다. CPU 30 은 VME 버스에 연결되었으며, SUN 워크스테이션은 네트워크를 통해 전체적으로 통신 정보를 주고 받게된다. 실시간 운영체제인 VxWorks를 OIS/EWS 기능을 하는 SUN 워크스테이션에 구축해서, 타겟이 되는 프로세서 모듈인 CPU 30 이 부팅할때 VxWorks 관련 파일이 LAN을 통해 RAM에 올려져 프로세서 모듈에서 실시간 작업을 위해 사용되어진다. 실험 시스템 구성도는 < 그림 4>와 같다.

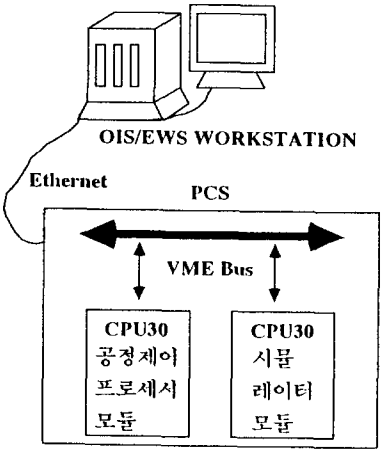


그림 4: 실험 환경

5.2 실험 내용

신경 회로망 제어기의 구성을 위해서는, 우선 제어하고자 하는 플랜트의 동특성을 동정하는 신경 회로망 에뮬레이터의 학습이 필요하다. 그리고, 신경 회로망 에뮬레이터의 학습을 위해서는 플랜트의 입출력 데이타가 필요하다. 플랜트의 입출력 데이타를 얻기 위해서는 우선 드럼 수위가 PID 제어기에 의해 제어되도록 한다. 이렇게 해서 얻은 플랜트의 입출력 데이타를 통해 신경 회로망 에뮬레이터를 학습시키고 이어서 신경 회로망 제어기를 학습시킨다. 신경 회로망 제어기의 학습이 완료되면, 신경 회로망 제어기의 가중치 값을 파일에 저장한 다음, 실제 PCS에서 동작되는 신경 회로망 제어기의 초기화나 온라인 파라메타 변환에 이 파일을 이용한다. 학습에 의해 얻어진 신경 회로망 제어기의 가중치가 PCS에 전달된 후부터 신경 회로망 제어기는 드럼 수위를 제어하기 위한 제어 신호를 보낼 수 있게 된다.

5.3 실험 결과

신경 회로망 에뮬레이터와 신경 회로망 제어기의 학습을 위한 프로그램의 초기 화면이 < 그림 5 >에 나와있다. 신경 회로망의 학습이 진행되는 전체 절차를 볼 수 있다. 이와 같은 학습에 의해 신경 회로망 제어기의 가중치가 구해지면 Configuration 파일에 신경 회로망 제어기를 구현한다. < 그림 6> 는 드럼 수위를 제어하기 위한 신경 회로망 제어기가 기능 코드로 구현된 것을 보여준다. 신경 회로망 제어기에 의한 드럼 수위 제어 결과가 < 그림7>에 나와있다. 드럼 수위(L), 밸브 조작량(BFPa), 전력 요구량(MW)을 퍼센트로 환산한 량(최대이면 100, 최소이면 0)을 시간(단위는 초)의 변화에 따라 그렸다. 부하에 해당하는 전력 요구량이 75 퍼센트에서 100 퍼센트로 변동함에도 드럼 수위가 거의 일정하게 제어되고 있는 모습을 볼 수 있다.

```
Neural Emulator Learning

<<  Summary of the total procedure  >>

[1] Determine the training data file name.
[2] Determine the input variables for the neural emulator.
[3] Determine the number of the hidden neurons
    for the neural emulator.
[4] Learn the neural emulator.
[5] Save the weights of the neural emulator learned completely.

Neural Controller Learning

<<  Summary of the total procedure  >>

[1] Determine the pre-processor type.
[2] Determine the post-processor type.
[3] Determine the number of the hidden neurons
    for the neural controller.
[4] Learn the neural controller.
[5] Save the weights of the neural controller learned completely.
```

그림 5: 신경 회로망 에뮬레이터와 제어기 학습 프로그램의 초기 화면

```
! Drum Level Loop
! NN COntroller
 56 L= ain( ; 0, 2, 4) !--------------------------L
 57 offset1= msc( ; 50.0)
 58 y= sumII(L, offset1; 0.3, -0.3)
 59 yd = delayone(y ;) !--------------------------------Pre-processing
 60 setval = msc( ; 0.0)
 61 suby1 = sumII(setval, y; 1.0, -1.0) !--------------Pre-processing
 62 suby2 = sumII(yd, y; 1.0, -1.0) !------------------Pre-processing
 63 dummy = msc( ; 0.0)
 64 nnx = ftov( 2, suby1, suby2, dummy, dummy; ) !-----Pre-processing
 65 nnytr = ftov( 1, dummy, dummy, dummy, dummy; )
 66 nntrsw= mss( ; 1)
 67 nny = nn(2, 1, nnx, nnytr, nntrsw, 4, 0, 0; 1) !---NNcontroller
 68 snny = sumII( nny[0], dummy; 8.3333, 0.0)
 69 bfpprev1 = sumII( bfpprev2, snny; 1.0, 1.0) !------Post-processing
 70 bfpprev2 = hilo(bfpprev1; 100.0, 0.0)
 71 bfpa= aout(bfpprev2; 0, 1, 4) !----------------BFPa
 72 bfpb= aout(bfpprev2; 0, 1, 5) !----------------BFPb
 73 bfpc= aout(dummy; 0, 2, 0) !------------------BFPc
```

그림 6: 신경 회로망 제어의 기능 코드로 구현

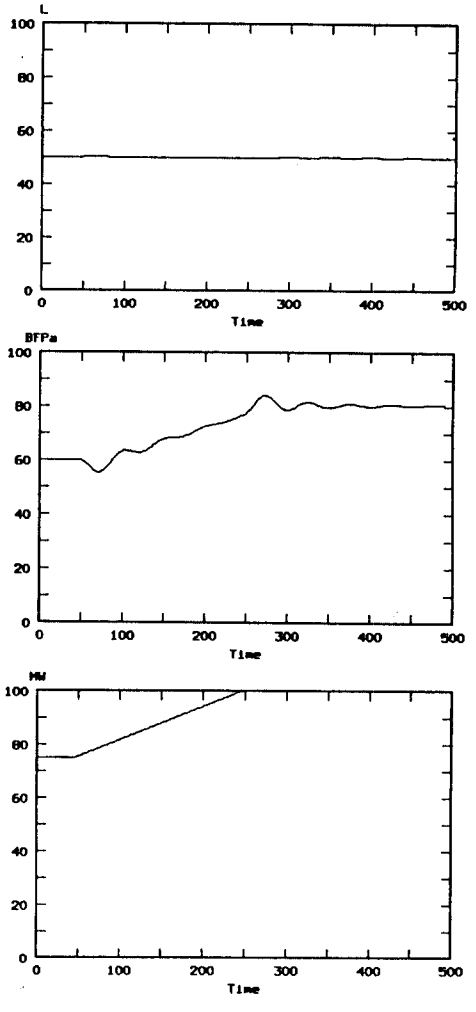


그림 7: 신경 회로망 제어기에 의한 드럼 수위 제어 결과

6 결론 및 추후 연구 과제

지금까지 공정 제어를 위한 고급 분산 제어 시스템에 신경 회로망 알고리즘이 구현되는 일반적인 방법을 제안하였다. 서울 화력 4호기의 보일러 시뮬레이터를 대상으로 한 실험을 통해 제안된 방법의 효용성을 확인하였다. 제안된 방법에서는 신경 회로망의 학습은 EWS에서 오프라인으로 수행되며, 학습에 의해 얻어진 가중치 값들은 신경 회로망 제어기의 기능 코드 구현시에 파라메타값 초기화나 온라인 파라메타 변환에 사용되어진다.

한편 신경 회로망 에뮬레이터와 신경 회로망 제어기 학습을 위해서는 많은 시간과 시행 착오를 필요로 한다. 또한 신경 회로망 에뮬레이터의 학습을 위해서 플랜트의 입출력 데이타를 얻는 작업도 쉽지만은 않다. 제어하고자 하는 플랜트를 잘 학습하기 위해서는 플랜트에 대한 보다 많은 정보를 담고있는 학습 데이타가 필요하기 때문이다. 이런 문제를 보다 효과적으로 처리할 수 있는 방법에 대한 연구가 추후 과제로 남아 있다.

참고 문헌

[1] 한전 기술 연구원, "발전소 제어용 계장 제어 시스템 개발(최종 보고서)," 1990년 8월.

[2] 한전 기술 연구원, "분산 제어 시스템의 고장 대처 기능및 제어 언어의 구현(최종 보고서)," 1993년 3월.

[3] Baily Controls, "Baily Network 90 - Function Code Reference Manual".

[4] D. Psaltis, A. Sideris, and A. A. Yamamura, "A Multilayer neural network controller," *IEEE Contr. Sys. Mag.*, vol. 8, pp. 17-21, Apr. 1988.

[5] D. H. Nguyen and B. Widrow, " Neural networks for self-learning control systems," *IEEE Contr. Sys. Mag.*, vol. 8, pp. 18-23, Apr. 1990.

[6] Q. H. Wu, B. W. Hogg, and G. W. Irwin, "A neural network regulator for turbogenerators," *IEEE Trans, Neural Networks*, vol. 3, pp. 95-100, Jan. 1992.

[7] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, Mar. 1990.

[8] F. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Contr. Sys. Mag.*, vol.10, pp. 44-48, Apr. 1990.