

2002 추계학술발표회 논문집  
한국원자력학회

## 공기구동밸브 결함 결정 알고리즘 개발

### Development of Algorithm for Symptom Definition of Air-Operated Valve

채장범, 김윤철, 최현우

아주대학교

경기도 수원시 팔달구 원천동 산 5번지

김우식

세종대학교

서울시 광진구 군자동 98번지

#### 요 약

공기구동밸브(Air-Operated Valve)의 현재 상태가 건강한 상태인지 아닌지, 건강한 상태가 아니면 어떤 결함을 갖는지를 결정하기 위한 알고리즘을 개발하였다. 결함 결정을 위해 사용된 진단 알고리즘은 Neural Net과 Non-neural Net(Simple Pattern Matching) 알고리즘이다. 진단 알고리즘의 입력 값으로는 이전에 구축된 결함 관련 데이터베이스를 사용하였으며 이를 진단 알고리즘에 적용하기 위해 수치적으로 변환하였다. 두 알고리즘에 적용한 결과를 놓고 비교하여 서로 공통되는 결함을 최종적으로 선택하였으며 결함을 정확히 판별해 냄을 확인하였다.

#### Abstract

We developed a diagnostic algorithm to find out whether an Air-Operated Valve system is healthy or not and, if it is not, to identify what kind of symptoms the system has. The algorithm is composed of a Neural Net part and a Non-neural Net (Simple Pattern Matching) part. The algorithm takes a series of numerical vectors as inputs, which were converted from the arrow patterns of symptoms, which we had already developed before. We pick a common part of the results of two algorithms as a final decision and we confirmed that the algorithm worked well.

#### 1. 서 론

원자력 발전소를 오랫동안 운전해 온 미국에서의 경험에 의하면 원자력 발전소의 불시 정지나 성능 저감은 부품의 진동, 오염, 부식, 그리고 오작동이 전체 시스템에 영향을 줌으로써 발생하고 있음을 보여주고 있다. 그러므로 원자력 발전소 전체의 안전성 향상과 신뢰성 증대를 위해서는 전체 시스템에 영향을 줄 수 있는 부품의 감시와 진단이 필수적으로 요구되고 있다. 본 연구에서 진단하고자 하는 공기구동밸브는 원자력 발전소의 급수 시스템 등에서 유량을 제어하기 위해 필수적으로 사용되는 밸브로 미국의 경우 NRC에서 주기적인 성능 점검과 감시를 수행하도록 권장하고 있다. 이에 본 연구에서는 공기구동밸브의 결함(symptom)에 대해 구축된 데이터베이스를 이용하여 결함을 결정할 수 있는 알고리즘을 개발하였다. 본 논문에서는 먼저 전체 알고리즘의 구성을 살펴본 후, 구축된 데이터베이스를 진단 알고리즘에 적용하기 위한 방법에 대해 설명하고, 결함 결정을 위해 적용한 알고리즘과 그 결과에 대해 논의하도록 한다.

## 2. 결함 결정 알고리즘의 구성

전체 결함 결정 알고리즘의 구성은 결함 패턴을 입력하는 부분, 알고리즘 부분, 그리고 마지막으로 결과를 표시해 주는 부분으로 구성된다. 그림 1은 전체 결함 결정 알고리즘의 구성을 블록선도로 표시한 것이다.

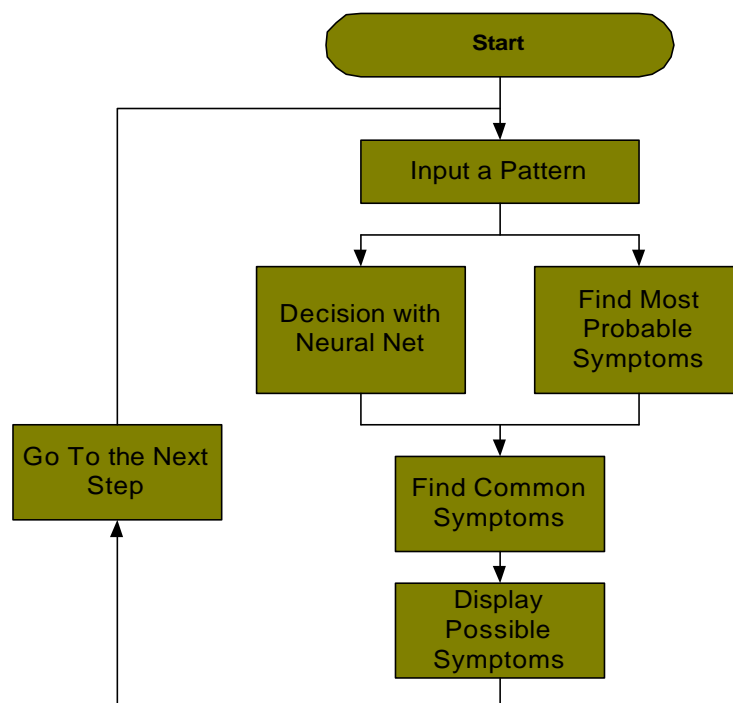


그림 1 결함 결정 알고리즘의 블록선도

### 3. 입력 패턴 처리







여러 가지 결함 중 18가지 결함을 선택하여 입력 패턴으로 사용하였으며 이를 표 1에 정리하였다.

표 1 결함의 종류

| 결함 번호 |      | 결함 항목                 |
|-------|------|-----------------------|
| 1     |      | 제한된 공급 공기             |
| 2     | 2-1  | 전압-공압 변환기 영점 높게 변함    |
|       | 2-2  | 전압-공압 변환기 영점 낮게 변함    |
| 3     | 3-1  | 전압-공압 변환기 작동 범위 크게 변함 |
|       | 3-2  | 전압-공압 변환기 작동 범위 작게 변함 |
| 4     |      | A 위치 누설               |
| 5     |      | A 위치 막힘               |
| 6     | 6-1  | 위치 제어기 시작점 빠르게 변함     |
|       | 6-2  | 위치 제어기 시작점 늦게 변함      |
| 7     |      | 피드백 링크지 고착            |
| 8     |      | B 위치 누설               |
| 9     |      | B 위치 막힘               |
| 10    | 10-1 | 구동기 스프링 프리로드 크게 변함    |
|       | 10-2 | 구동기 스프링 프리로드 작게 변함    |
| 11    | 11-1 | 패킹 부하 크게 변함           |
|       | 11-2 | 패킹 부하 작게 변함           |
| 12    | 12-1 | 피드백 링크지 스프링 강성 크게 변함  |
|       | 12-2 | 피드백 링크지 스프링 강성 작게 변함  |

각각의 결함 항목에 대하여 순간순간 값을 측정하고, 기본 값 (baseline value)과의 차이를 계산하여 변화를 arrow로 나타낸다. 다음 표 2는 구축된 데이터베이스인 경향 테이블의 arrow pattern을 만들기 위한 arrow 값에 대하여 나타내었다. 표의 오른쪽은 arrow 값과 알고리즘에 입력하기 위해 대응시킨 숫자이며, 이런 arrow pattern은 실제로 알고리즘을 개발하는데 그 자체로는 구현하기가 어려우므로, 수치적인 변환을 한 결과이다. 여기서 b는 bias로 충분히 큰 수 또는 작은 수를 나타낸다. 또 한편으로 입력 패턴에서 결함에 따라 중요한 파라미터와 중요하지 않은 파라미터들이 있게 된다(데이터베이스에서
















































































표 2 arrow pattern과 대응되는 숫자

| arrow pattern   | 대응되는 숫자 |
|---|---------|
|  | b       |
|  | +1      |
|  | 0       |
|  | -1      |
|  | b       |
|  | b       |

는 이를 \*으로 표시하였다). 이들 파라미터들의 중요도는 시뮬레이션에서는 mask를 이용하여 구현하였다.

입력 패턴을 예를 들면 A 위치 누설의 경우 arrow pattern은 다음 표 3과 같다.

표 3 A 위치 누설의 arrow pattern

|    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 항목 | 1-1   | 1-2   | 1-3   | 1-4   | 1-5   | 1-6   | 1-7   | 1-8   | 1-9   | 1-10  | 1-11  | 1-12  |   |   |
|    |    |  * |  * |    |    |    |    |    |    |    |    |    |   |   |
| 항목 | 2-1   | 2-2   | 2-3   | 2-4   | 2-5   | 2-6   | 2-7   | 2-8   | 2-9   | 2-10  | 2-11  | 2-12  | 2-13  | 2-14  |
|    |    |  * |  * |    |    |    |    |    |  * |  * |  * |    |  * |  * |
| 항목 | 3-1   | 3-2   | 3-3   | 3-4   | 3-5   | 3-6   | 3-7   | 3-8   | 3-9   | 3-10  | 3-11  | 3-12  | 3-13  | 3-14  |
|    |    |    |    |  * |  * |    |    |    |    |    |    |    |    |  * |
| 항목 | 3-15  | 3-16  | 3-17  | 3-18  | 3-19  | 3-20  | 3-21  | 3-22  | 3-23  | 3-24  |   |   |   |   |
|    |  * |  * |  * |  * |  * |  * |  * |  * |  * |    |    |   |   |   |
| 항목 | 4-1   | 4-2   | 4-3   | 4-4   | 4-5   | 4-6   | 4-7   | 4-8   | 5-1   | 5-2   | 5-3   | 5-4   | 5-5   | 5-6   |
|    |    |    |    |    |  * |  * |  * |  * |  * |  * |    |  * |  * |    |
| 항목 | 6-1   | 6-2   | 6-3   | 6-4   | 6-5   | 6-6   | 6-7   | 6-8   | 6-9   | 6-10  | 7-1   | 7-2   | 7-3   | 7-4   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

이 예의 경우 이 arrow pattern으로부터 arrow와 \* 표시를 각각 숫자로 변환을 하면 다음과 같다.

$P\{1\}=[b;-1;-1; b; b; b; b; b; b; b; b; b; b;-1;-1; b; b; b; b; 1;-1; 1;$   
 $1;-1; 1; 1; b; b; b; 1; 1; b; b; b; b; b; b; b; b; 1; 1; 1; 1;-1;$   
 $-1;-1;-1;-1; b; b; b; b; b; b; b; 1; 1;-1;-1;-1;-1; 1;-1;-1; 1; b; b;$

b; b; b; b; b; b; b; b; b; b; b; b;];

mask 함수는 다음과 같이

$M\{1\}=[0; 1; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0; 0; 0; 0; 0; 1;$   
 $1; 1; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1;$   
 $1; 1; 1; 1; 1; 1; 0; 0; 0; 0; 0; 0; 1; 1; 1; 1; 1; 1; 0; 1; 1;$   
 $0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;];$

로 주어진다. 모든 결함에 관하여 arrow pattern와 mask pattern을 숫자로 변환을 시킨 다음에 알고리즘 개발을 위한 입력으로 사용하며, 실제의 데이터도 이와 유사한 패턴으로 나올 것으로 생각된다. 다음은 본 연구에서 사용한 알고리즘에 대해 알아보도록 한다.

#### 4. Neural Net 알고리즘

그림 2는 이 연구에서 사용된 Neural Net의 구조<sup>1)</sup>를 나타내는 그림이다. 그림에서 입력은  $x_1 - x_R$ 까지로 R개의 입력으로 구성되어 있고, 단층 Layer Neural Net을 통하여 처리된 다음에 Hard Limiter를 통하여 결과를 판정하게 된다. 출력의 수는 S개이고, 따라서

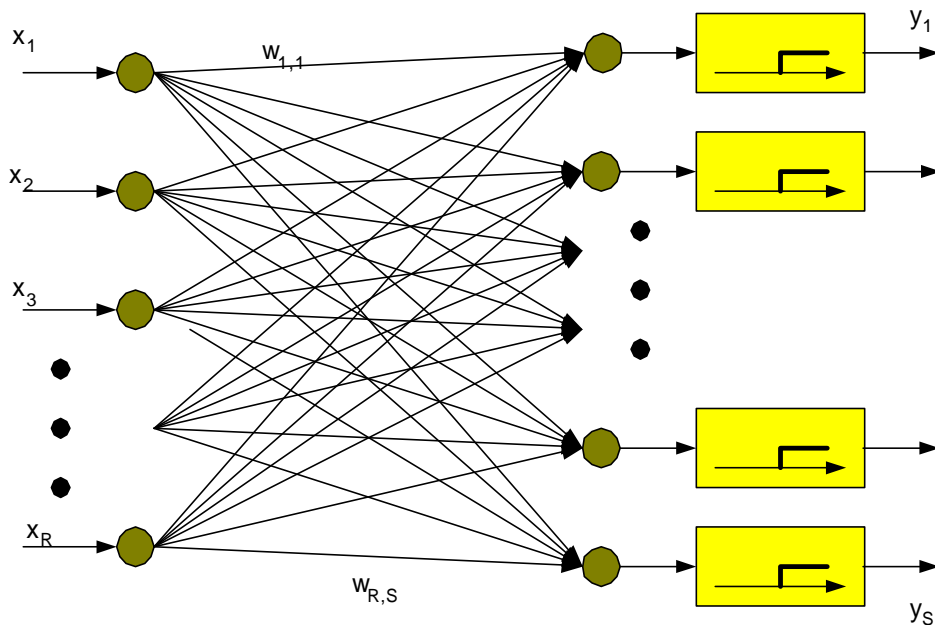


그림 2 간단한 Neural Net 모델

Neural Net 안에서는 RxS 개의 coefficient  $\omega_{i,j}$ 를 갖게 된다. 그림의 Neural Net의 입출력 관계를 식으로 나타내면

$$\bar{y} = f(W\bar{x} + \bar{b}) \quad (1)$$

로 주어진다. 여기에서 W는 RxS weight matrix, f는 전달 함수,  $\bar{b}$ 는 bias,  $\bar{y}$ 는 1xS의 출력 벡터가 된다. 위 식 (1)에서 입력과 출력의 식을 풀어 써보면, 행렬의 식이 된다. 즉,

$$y_j = \sum_{i=1}^R w_{i,j} x_i + b_j, j = 1, \dots, S \quad (2)$$

와 같이 된다. 여기서  $w_{i,j}$ 는 coefficient matrix의 (i,j) 성분,  $x_i$ 는 입력,  $b_j$ 는 bias이다. 여기서 coefficient 행렬의 성분은 일련의 supervised training, unsupervised training 또는 adaptation 과정을 통하여 결정할 수 있으며 이 과정에서 계수들은 적합한 값으로 갱신(update)된다. 본 연구에서 사용한 전달 함수는 일반적으로 Neural Net에서 가장 많이 사용하는 함수인 Hardlimiter로 그 특징은 다음과 같이 주어진다.<sup>1),2)</sup>

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (3)$$

그림으로 나타내면, 그림 3과 같다.

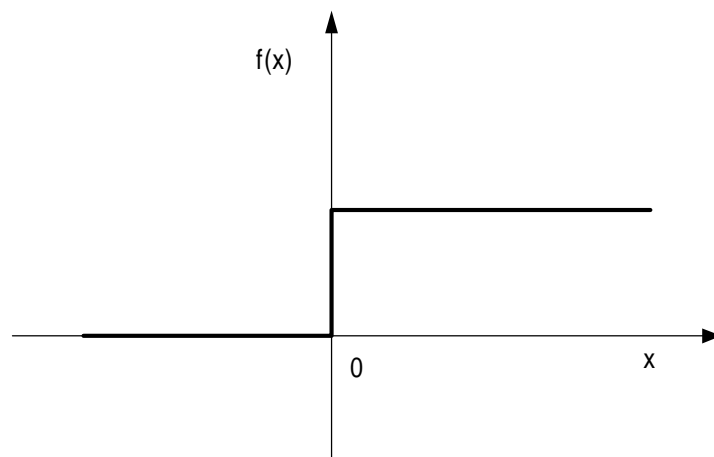


그림 3 Hardlimiter의 전달 특성

Neural Net의 시뮬레이션은 Matlab의 Neural Net toolbox에서 제공하는 함수와 알고리즘을 이용하였다.<sup>3),4)</sup> 학습(training)을 위해 가장 많이 사용하는 기능은 adapt와 train 함수이다. 이 중에서 adapt 함수를 이용하여 coefficient와 bias를 구하였다. adapt 함수는 incremental training 방법의 하나로 batch training을 하는 train 함수와 대조가 된다. 이 방법은 주로 adaptive filter와 같은 동적(dynamic) 시스템에 주로 사용이 되지만 본 연구의 경우와 같이 정적(static) 시스템에서도 사용할 수 있다. 일반적인 adapt 함수의 형태는

$$[\neq t, a, e, pf] = \text{adapt}(\neq t, P, T) \quad (4)$$

와 같다. 여기에서 사용되는 각종 파라미터들은 다음과 같다. 입력에서

- R : 입력의 개수로 sensor의 개수에 해당한다.
- S : 출력의 개수로 결함의 개수와 일치한다. 이 연구에서는 18개의 결함을 사용하였다.
- net : Neural Net의 각종 파라미터를 갖고 있는 변수로, 각종 파라미터를 저장할 수 있는 공간을 확보하고, 저장하는 역할을 한다. 또한 학습한 결과를 저장하여 출력하는 작용도 아울러 한다.
- P : 학습을 하기 위한 입력 데이터를 말한다. S개의 Rx1 Neural Net의 학습 벡터
- T : Neural Net의 target 값으로 이 경우 SxS의 identity matrix로 가정할 수 있다.

출력의 경우에는

- net : 갱신된 Neural Net 값
- Y : Neural net 출력을 말한다.
- E : Error를 말한다.

이다.

net 변수는 일종의 구조체 데이터 변수로 많은 파라미터를 포함하고 있다. 중요한 파라미터는 다음과 같다.

|  |   |
|--|---|
| <pre>net = Neural Network object: ■ architecture:     numInputs: 1     numLayers: 1     biasConnect: [1]     inputConnect: [1]     layerConnect: [0]     outputConnect: [1]     targetConnect: [1]     numOutputs: 1 (read-only)     numTargets: 1 (read-only)     numInputDelays: 0 (read-only)     numLayerDelays: 0 (read-only)  ■ subobject structures:     inputs: {1x1 cell} of inputs     layers: {1x1 cell} of layers     outputs: {1x1 cell} containing         1 output     targets: {1x1 cell} containing         1 target     biases: {1x1 cell} containing 1 bias     inputWeights: {1x1 cell} containing 1         input weight     layerWeights: {1x1 cell} containing         no layer weights</pre> | <pre> ■ functions:     adaptFcn: 'adaptwb'     initFcn: 'initlay'     performFcn: 'mae'     trainFcn: 'trainwb'  ■ parameters:     adaptParam: .passes     initParam: (none)     performParam: (none)     trainParam: .epochs, .goal, .max_fail,         .show, .time  ■ weight and bias values:     IW: {1x1 cell} containing 1 input         weight matrix     LW: {1x1 cell} containing no layer         weight matrices     b: {1x1 cell} containing 1 bias vector  ■ other:     userdata: (user stuff)</pre> |
|--|---|

특히 net.IW에는 Neural Net의 weight의 값이 저장되게 되며, net.B에는 bias 값이 저장된다. 나중에 Neural Net를 구현하려면 이 두 가지의 파라미터를 이용하면 된다. 일반적으로 주어진 데이터를 이용하여 한번의 학습을 하게 되면 충분히 수렴(converge)하지 않게 된다. 따라서 weight matrix의 계수들이 충분히 수렴하여, 보다 정확한 결과를 내기 위해서는 많은 수의 학습을 하여야 한다. 이 실험에서는 60번의 학습을 하였고, 이 숫자는 net.adaptParam.passes에 할당해 준다.

## 5. Non-neural Net(Simple Pattern Matching) 알고리즘

Neural Net 방법의 입력되는 패턴이 학습 패턴과 일치하는 경우에는 100% 가까이 인식하는데 반하여 입력 패턴이 학습 패턴과 조금 상이한 경우에는 오류가 일어나는 현상이



목격되었다. 이로 인한 오동작을 방지하기 위하여 다른 형태의 패턴 인식 방법을 병행하여 보완하는 것이 바람직하다는 결론에 이르게 되었다. Non-neural Net 알고리즘은 기본적으로 패턴 일치(pattern matching) 알고리즘이라고 할 수 있다. 입력으로 들어오는 패턴에 대하여 성분(component)별로 일치하는 가를 검사하여 일치(matching) 정도에 따라 점수를 주고, 이를 합하여 각 결함이 가질 수 있는 값과 비교하여 일치 정도를 나타내는 방법이다. 예를 들어 \*표가 표시된 (다시 말하여 결함의 결정에 중요한 역할을 하는 항목의) arrow에 대하여 arrow가 정확히 일치되면 5점, 약간 불일치하면 3점, 전혀 일치하지 않으면 1점을 주었으며, \*표가 표시되지 않은 항목에 대하여는 일치하면 2점을 그 밖의 경우는 0점을 주었다. 총점은

$$Score_j = \sum_{i=1}^R g(x_i, p_{i,j}) \quad (5)$$

으로 나타낼 수 있다. 여기서  $g()$ 는 일치에 따른 점수를 산정하는 함수,  $x_i$ 는 입력된 패턴의  $i$  번째 성분을,  $p_{i,j}$ 는  $j$ 번째 결함의  $i$ 번째 성분을 나타낸다. 일치 백분율(Matching percentage)은 위에서 계산된  $Score_j$ 와 입력되는 패턴과 정확히 일치되었을 때 가질 수 있는 최대값( $Max_j$ )을 이용하여 다음의 식

$$Matching\ percentage(\%) = \frac{Score_j}{Max_j} \times 100(\%) \quad (6)$$

로서 정의하였다.

## 6. 시뮬레이션 및 결과

시뮬레이션(Simulation)은 주어진 학습 데이터를 이용하여 해보았다. 구축된 데이터베이스에서 주어진 데이터를 이용하여 Neural Net 파라미터(weighting matrix의 계수와 bias)를 구하고, 이 결과와 함께 Non-neural Net 방법으로 각각 결과를 낸 다음에 두 가지의 결과에서 공통된 부분을 추출하여 일치 백분율과 함께 출력하도록 하였다. 그림 4는 Matlab 함수를 실행했을 경우의 화면을 나타내는 그림이다. 이 시뮬레이션에서는 개발된 프로그램의 입력으로 A 위치 누설에 해당되는 패턴을 입력한 경우에 결과 화면을 나타낸 것이다. 예상한대로 일치 백분율은 A 위치 누설의 경우 100%가 되었고, 이와 비슷한 전압-공압 변환기 작동 범위 작게가 93%의 일치 정도를 나타내었다. 다른 결함의 경우는 모두 90% 미만이 되기 때문에 나타내지 않았다. 또한 Neural Net 프로그램 결과도 예상한 바와 같이 정확히 인식하였다. 따라서 전체적인 결과도 정확히 인식하는 것으로 나왔음을 알 수 있다. 표 4에는 위에서 수행한 방식과 같은 방법으로 모든 결함에 대하여

수행한 결과를 정리하였다. 이 표에서 볼 수 있는 것과 같이 Neural Net의 결과는 100% 정확히 일치하였고, Non-neural Net(Simple Pattern Matching) 알고리즘의 경우에 대해서도 사용한 학습 데이터와 대부분 맞게 나온 것을 알 수 있었다. 그 중에서 일부 결함의 패턴은 다른 것과 매우 일치 (99%, 97% 등)하는 경우도 있었다. 이런 경우는 결함의 특성이 매우 유사한 것으로 가끔 오동작을 유발하기도 하는 것으로 나타났으며, 이런 경우 결함을 통합하여 정리하는 것이 바람직 할 것으로 생각된다.

```

b; b; b; b; b; b; b; b; b; b;
b; b; b; b;
]
out_score =
Columns 1 through 7
83.6957 57.4713 84.9398 48.3766 92.5325 100.0000 72.0000
Columns 8 through 14
84.2593 52.1605 81.3433 85.1852 71.7532 70.5882 70.0000
Columns 15 through 18
74.0566 69.3396 81.1538 59.6154
=====
No. 1: Result with Non neural Net
=====
ans =
Degree of matching with Symptom No. 3-2 : 전압-공압 변환기 작동 범위 낮게 is 93 %
ans =
Degree of matching with Symptom No. 4 : A 위치 누설 is 100 %
=====
No.2: Results with Neural Net
=====
ans =
Matched Pattern(s) from Neural Net is No. 4 : A 위치 누설.
=====
From these results, the recognized symptom is
=====
ans =
Matched Pattern(s) from Both Approach is No. 4 : A 위치 누설 with Matching Percentage 100 %.
□

```

그림 4 프로그램 실행 예

표 4 시뮬레이션 결과 정리

| 결함<br>번호 | 결함 항목                     | Non-Neural Net 결과<br>(일치 백분율)                | Neural Net<br>결과 | 최종결과 |
|----------|---------------------------|--|------------------|------|
| 1        | 제한된 공급 공기                 | 1(100)                                       | 1                | 1    |
| 2        | 2-1 전압-공압 변환기<br>영점 높게    | 2-1(100), 3-1(91)                            | 2-1              | 2-1  |
|          | 2-2 전압-공압 변환기<br>영점 낮게    | 2-2(100), 3-2(92)                            | 2-2              | 2-2  |
| 3        | 3-1 전압-공압 변환기<br>작동 범위 크게 | 3-1(100)                                     | 3-1              | 3-1  |
|          | 3-2 전압-공압 변환기<br>작동 범위 작게 | 3-2(100)                                     | 3-2              | 3-2  |
| 4        | A 위치 누설                   | 3-2(93), 4(100)                              | 4                | 4    |
| 5        | A 위치 막힘                   | 5(100)                                       | 5                | 5    |
| 6        | 6-1 위치 제어기<br>시작점 빠르게     | 3-2(90), 6-1(100), 7(93),<br>8(92), 12-1(93) | 6-1              | 6-1  |
|          | 6-2 위치 제어기<br>시작점 늦게      | 3-1(90), 6-2(100),<br>12-2(93)               | 6-2              | 6-2  |
| 7        | 피드백 링크지 고착                | 6-1(94), 7(98)                               | 7                | 7    |
| 8        | B 위치 누설                   | 6-1(97), 8(100),<br>12-1(91)                 | 8                | 8    |
| 9        | B 위치 막힘                   | 9(100)                                       | 9                | 9    |
| 10       | 10-1 구동기 스프링<br>프리로드 크게   | 10-1(100)                                    | 10-1             | 10-1 |
|          | 10-2 구동기 스프링<br>프리로드 작게   | 10-2(100)                                    | 10-2             | 10-2 |
| 11       | 11-1 패킹 부하 크게             | 11-1(100)                                    | 11-1             | 11-1 |
|          | 11-2 패킹 부하 작게             | 11-2(100)                                    | 11-2             | 11-2 |
| 12       | 12-1 피드백 링크지<br>스프링 강성 크게 | 12-1(100)                                    | 12-1             | 12-1 |
|          | 12-2 피드백 링크지<br>스프링 강성 작게 | 12-2(100)                                    | 12-2             | 12-2 |

## 7. 결 론

이 논문에서는 원자력 발전소에서 사용하는 공기구동밸브의 현재 상태가 건강한 상태인지 아닌지, 건강한 상태가 아니면 어떤 결함을 갖는지를 결정하기 위한 알고리즘을 개발하였다. 결함 결정을 위해 사용된 진단 알고리즘은 기본적으로 Neural Net을 사용하였으나, Neural net의 결과가 입력이 training pattern과 정확히 일치하면 100%에 가까운 판

단 능력을 보여주지만, 입력 패턴이 유사하지만 정확히 일치하지 않으면 판정에 오류가 생기는 현상이 일부 목격되어 Simple pattern matching에 근거한 Non-neural Net 알고리즘과 병행하였으며, 두 알고리즘의 결과에서 공통된 부분을 최종 결정하도록 하였다. 진단 알고리즘의 입력 값으로는 이전에 구축된 결함 관련 데이터베이스를 사용하였으며 이를 진단 알고리즘에 적용하기 위해 수치적으로 변환하였다. 시뮬레이션 결과 입력으로 training pattern과 일치하는 패턴이 입력될 경우 결함을 정확히 판별해 냄을 확인하였다.

## 8. 후기

본 연구는 과학기술부가 지원하는 원자력연구개발 선진기술확보사업으로 아주대학교가 주관하여 수행되었습니다. 관계자 여러분의 협조에 감사드립니다.

## 9. 참고 문헌

- [1] 김대수, "신경망 이론과 응용", 하이테크 정보, 2001, 8.
- [2] 이관형 외 5, "인공지능의 원리와 실무사례", 동영출판사, 1998, 8.
- [3] 이현엽, 문경일, "Matlab 을 이용한 퍼지-뉴로", 아진 , 1999,12.
- [4] Howard Demuth, Mark Beal, "Neural Network Toolbox For Use with Matlab" User's Guide Version 3.0, MathWorks, Inc. January 1998.