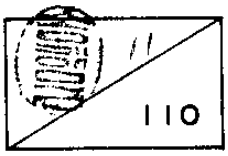
'89국책연구개발
"공정 인공지능시스템(PAI)개발"
사업의 세부과제



공정조업 진단 및 제어용 전문가 시스템의 개발연구

Development of an Expert System for Process Diagnosis and Control

연구기관
한국과학기술연구원

과 학 기 술 처

제 출 문

과학기술처장관 귀하

본 보고서를 "공정 인공지능 시스템(PAI)개발" 사업의 세부과제인 "공정조업 진단 및 제어용 전문가 시스템의 개발연구" 과제(1차년도)의 년차 보고서로 제출합니다.

1990년 7월 11일

주관연구기관명 : 한국과학기술연구원

총괄연구책임자 : 송형근 (한국과학기술연구원 책임연구원)

여영구 (〃 선임연구원)

연 구 원 : 김경미 (〃 연 구 원)

노 균 (〃 위촉연구원)

김홍석 (〃 〃)

요 약 문

Ⅰ. 제 목

공정조업 진단 및 제어용 전문가 시스템의 개발연구

Ⅱ. 연구개발의 목적 및 중요성

가. 목 적

본 연구의 목적은 다양한 공정제어기술 및 제반 경우의 제어기 조정방법과 제어 감지기술, 그리고 발생 가능한 공정의 결함 및 사고를 망라한 진단용 지식 베이스를 확립하고 이들 지식 베이스의 구조화와 공정지식 rule 시스템의 구축 및 외부 제어 시스템과의 정보 교환을 가능하도록 하여주는 일반적인 shell을 구성하여 공정조업 진단 및 제어용 전문가 시스템을 구축하며 선정된 공정의 pilot 플랜트 조업자료를 이용하여 보정 보완시켜 상용 패키지화를 도모하는데 있다.

제1차년도에서의 연구 목적은 기본적인 지식 rule 운용기능의 확립과 공정지식 데이타 베이스 구조의 개발, 그리고 시험용 pilot장치의 구성에 있다.

나. 중요성

국내 S/W개발기술의 수준은 단위공정의 최적화, 모사 및 제

어시스템을 설계하는 중급 단계에 와 있으나 이를 선진국 수준으로 끌어올리기 위한 노력이 미흡하였다. 화학공장의 최적조업과 장치의 결함이나 돌발적인 사고의 사전방지를 위해서는 화학공정의 정확하고 신속한 진단과 진보된 공정제어구조의 도입이 절실하지만 이를 기업체에서 자체개발하는것은 우리 실정에서 불가능하다.

공장조업 진단 및 제어용 전문가 시스템의 개발에는 많은 시간과 고급전문인력이 소요되는데 아직까지 공정용 전문가 시스템 개발을 위한 정부차원에서의 체계적이고 지속적인 지원은 전무하였다. 공정용 전문가 시스템을 외국에서 도입하는 경우에는 국내 공정기술의 대외노출로 인한 경쟁력 약화 및 S/W 기술의 해외 예속화 등 심각한 문제들이 야기된다. 따라서 공정산업의 국제 경쟁력을 향상시키고 공정의 효율성과 제품의 품질을 높여나가기 위해서는 공정용 Expert system개발기술의 확보와 실제적인 활용이 필수적인 과제이다.

Ⅲ. 연구개발의 내용 및 범위

가. 프로그램 언어의 확립

나. Rule구조의 결정

다. FORTRAN과의 호환체계 확립

라. Pilot플랜트의 구성

마. 전문가 진단 및 제어구조설치

Ⅳ. 연구개발의 결과 및 건의

1차년도 연구수행결과 공정용 전문가 시스템 구조의 확립에 있어서는 일차적으로 사용 언어와 기본적인 골격의 선정을 토대로 공정지식의 효율적 처리를 위한 데이타 운용 시스템의 구축작업이 진행되었다. 아울러 공정지식 표현의 매카니즘과 지식 운용기능의 구성이 이루어 졌으며 공정의 진단 및 제어기법에 대한 연구가 진행되었다.

가. 공정 데이타의 표현 및 운용은 tree 구조 및 linked list 구조와 이에 바탕을 둔 frame식 구조에 의해 효율적으로 이루어질 수 있음이 확인되었다.

나. 공정 조업용 prototype 전문가 시스템의 개발에 있어서는 프로그래밍 언어로서 C를 사용하여 기본적인 지식 베이스의 구축 및 운용과정과 초보적인 공정진단과정을 구성하였다. 비교적 간단한 장치나 공정의 조업지식의 표현 및 운용이나 진단은 현재 구축된 시스템으로 가능함이 확인되었다.

다. 공정 진단기법에 있어서는 대상이 되는 공정의 모델링과 identification을 통해서 공정의 이상유무를 판단하는 방법이 연구되었다.

라. 공정의 제어를 위한 전문가 시스템 연구에서는 일차적으로 현재 해외에서도 그에 대한 연구가 활발히 진행되고 있는 fuzzy

logic제어기법이 주 연구대상이었다. 조업 현장에서의 경험적 지식이 중요한 요소가 되는 화학공정의 제어에 있어서 fuzzy logic제어기법은 전문가 시스템에서 특히 유용하게 활용될 수 있는 제어방법이다. 이외에 예측제어 기법이나 적응제어 방법들로서 본 연구팀에 의해 이미 개발된 시스템들이 보완되어서 전문가 시스템에 적용될 것이다.

SUMMARY

Many of the tasks encountered in the chemical engineering operation cannot be completely articulated into a well-informed algorithmic procedure. They are a combination of a variety of special-purpose heuristic problem-solving methods and traditional algorithmic parts. Advances in Artificial Intelligence provide the means of automating the heuristic tasks. A number of successful expert systems or knowledge-based systems have been developed for areas such as chemistry, medicine, computer systems and mathematics. Unfortunately, these programs lacked facilities to reflect the rich variety of knowledge necessary to solve chemical engineering problems.

As the expert system area has matured, it has been possible to start proposing architectures to solve control and fault diagnosis problems in chemical engineering processes. The goal of the present study is to develop an expert system for process control and diagnosis. A prototype expert system for process operation was developed and techniques for model based fault diagnosis and fuzzy logic control were investigated.

The present expert system illustrates the integration of different paradigms along some of the several dimensions of

expert systems applications: knowledge representation, problem-solving methods, and levels of knowledge abstraction. All these properties are achieved through the use of C languages. In particular we have presented the methods of fuzzy logic control and model based process fault diagnosis, described how they consisted of a variety of subtasks, and presented results of computer simulations to show their usefulness.

Several other issues will be investigated in the future in order to make the application of the present expert system to chemical processes more practical. Pilot plant test is planned and the database structure will be revised to incorporate different types of chemical engineering data. For the purpose of comparison and modification, a commercial expert system shell will be utilized. Many other techniques for control and diagnosis will be coded into the system.

CONTENTS

목 차

제 1 장 장 서 론

제 1 절 개요

인공지능의 한 부류인 전문가 시스템은 지난 몇년동안 상당한 관심의 대상이 되어왔다. 1970년대 중반부터 컴퓨터 기술의 비약적인 발전에 힘입어 전문가 시스템에 대한 연구가 다방면에 걸쳐 활발하게 이루어져 오고 있는데 특히 의료진단등의 분야에서는 실제로 전문가 시스템을 이용하여 대단한 성공을 거두어 왔다. 현재 선진국에서는 광물탐사, 경영진단, Computer System의 설계, Language해석, 예측 및 계획 시스템, 감시와 제어장치 외에 공정의 오류탐지와 진단등 광범위한 영역으로 그 응용이 확대되어 수많은 종류의 전문가 시스템이 개발되어 활용되어오고 있다. 예를들면 MYCIN, XCON/XSEL, HEARSAY, FALCON, PICON등 전문지식을 갖춘 Knowledge-based 전문가 시스템 (KBES)들이 개발되었고 또 전문가 시스템을 만드는 tool들도 많이 개발 보급되어 전문지식만을 채워 넣음으로써 용이하게 사용할 수 있는 package들이 많다.

화학공장의 조업에 있어서 고장이나 사고의 신속하고 정확한 진단과 공정의 효율적이고 경제적인 최적제어는 가장 중요한 당면과제이다. 1960년대 이래 공정의 진단 및 제어방법에 대한 연구가 다양하게 이루어져오고 있으나 개발된 이론이 실제 현장에서

이용되는 경우는 매우 드물었다. 이론의 개발에 있어서는 매우 단순화된 공정모델과 비실제적인 조업조건이 이용되고 있으나 현장 조업에서는 이론에서 무시되었던 여러 사항들이 복합적으로 영향을 미치게 되며 또한 이론이나 수식으로는 다룰 수 없는 현장 조업 기술자의 경험적 지식이 조업에 반영되게 된다. 결국 개발된 진단이론이나 제어이론은 제한된 양의 공정조업지식에 의존할 수 밖에 없으며 따라서 이들 이론의 실제적인 활용에는 뚜렷한 한계가 존재하는 것이다.

현대의 컴퓨터는 아주 빠른 속도로 소형 경량화 및 고성능화 되어가고 있으며 조업 현장에서의 컴퓨터의 설치 활용은 이미 보편화 되어가고 있다. 공정의 조업, 특히 공정의 진단과 제어에 있어서 전문가 시스템은 앞서 언급한 여러 이론적인 지식들은 물론 현장 기술자들이 지니고 있는 경험적인 지식까지를 컴퓨터를 통하여 활용할 수 있게 하여주는 가장 적절한 도구가 되며 이의 적용은 공장 전산화를 위한 투자의 가치를 극대화시켜주는 첩경이 된다.

전문가 시스템 개발의 예로서 1983년 미국의 Du Pont 회사와 Foxboro, 그리고 Delaware대학의 공동 연구팀에 의한 화학공장 조업용 전문가 시스템의 개발을 들수 있다. FALCON(Fault Analysis Consultant)으로 명명된 이 프로젝트에서는 대상공정의 선정과 모사, 선정된 공정의 전문가에 의한 지식 베이스의 구축, 그리고 개발된 시스템의 실제 적용시험 단계로 개발이 진행되었다. 그 결과 전문가 시스템 기술은 화학공장의 조업에 지대한

영향을 주는 것으로 나타났다. 자동제어기술이 인간노동의 양을 줄여준 것과 같이 전문가 시스템은 조업지원기술의 소요노력을 줄여준다. 이는 곧 안정조업의 margin이 증진되고 생산량이 증가하며 제품의 질과 공정효율이 향상되는 결과로 귀착됨을 의미하여 주는 것이다.

우리나라에서도 공정 조업용 전문가 시스템에 대한 관심은 매우 높은 편이나 일부 대학에서 산업체와의 협력으로 그에 대한 기초연구를 진행하고 있는것 이외에는 국내에서 개발된 공정 조업용 전문가 시스템은 전무한 상태이다.

제2절 연구의 목적과 범위

공정용 전문가 시스템의 중추 기술은 inference engine을 효율적으로 설계하는 일이다. 방대한 공정지식 데이타 베이스를 보정 보완하고 문제의 해결에 필요한 rule들만을 처리 운용하는 기능이 빠른 시간내에 이루어져야만 실제의 real-time process에 유용하게 활용할 수 있다. 전문가 시스템 구축에 있어서 또하나 중요한 문제가 되는 것은 전문가가 지니는 지식을 어떻게 표현함으로써 컴퓨가 쉽게 식별할 수 있도록 할 것인가 하는 것이다. 현재까지는 공정용으로 production rule의 형태를 주로 사용해 오고 있지만 rule이 늘어남에 따라 시스템이 복잡해지고 경우에 따라서는 중복 또는 상반되는 rule들의 존재 가능성도 배재할 수 없다. 따라서 rule base외에 frame식 표현방법을 도입한다든가

하는등의 새롭고 보다 효율적인 지식표현방법이 요구된다.

본 연구에서 개발하고자 하는 공정진단 및 제어용 전문가 시스템은 공정의 제반 데이타를 기억 장소에 저장하여 두고 공정의 현 상태를 dynamic simulation 및 경험적 지식들을 바탕으로 분석 진단함으로써 공정의 비정상적인 이상상태를 파악하고 이를 사전에 조업자에게 알려 주어서 조업자로 하여금 필요한 조치를 취하게 하거나 전문가 시스템 자체가 적절한 작용을 취하게 된다. 그러므로 전문가 시스템의 개발에 관련되는 주요 핵심기술로서는 우선 dynamic simulation기술과 최신 제어기법을 들 수 있다.

전문가 시스템의 개발에 있어서는 전문가 시스템 Shell의 개발과 지식 베이스의 개발이 작업의 주종을 이룬다. 빠른 시일내에 적은 경비로 전문가 시스템을 개발하기 위한 일반적인 방법은 적절한 시스템 Shell을 구입하고 이를 이용하여 지식 베이스를 구축해 나가는 방법이다. 이 방법이 경제적이고 신속한 방법이기는 하지만 화학공정의 종류가 지극히 다양하고 많으며 다른 분야와 비교할 때 상호간에 유사성이 적으므로 개발되는 전문가 시스템의 용도와 활용범위에 한계가 있다. 어떠한 종류의 화학공정에도 두루 이용될 수 있는 시스템 Shell의 개발은 사실상 불가능한 일이지만 기본적인 시스템 Shell을 개발하는 일은 특정 공정을 위한 전문가 시스템 개발기술의 축적을 위해서뿐만 아니라 전문가 시스템 software의 해외 예속화 탈피를 위해서도 필수적이다.

본 연구의 1차년도 목표 가운데 하나는 기본적인 시스템 Shell의 개발연구이지만 이는 어디까지나 가장 기초적인 proto-

type의 Shell이 될 것이다. 전문가 시스템 프로그램 언어의 선택은 매우 중요한 문제인데 LISP, PROLOG, C, FORTRAN들 가운데에서 화학공정의 특성을 가장 잘 표현할 수 있는 언어로서 C와 LISP가 최종적으로 고려되었다. 화학공정 조업의 전산화에 있어서는 공정의 모사 및 설계기능이 또한 중요한 요소가 되기 때문에 보다 처리속도가 빠르고 공정지식 표현이 자유로운 C가 주 언어로 이용될 것이며 C와 LISP의 호환체계가 아울러 연구될 것이다.

제 2 장 Expert 진단 및 제어기술현황

제 1 절 공정진단용 전문가 시스템

1. 개 요

공정진단은 접근방법에 따라 인과법칙이나 고장 트리(fault tree)등을 이용하는 정성적 접근법(qualitative approaches) 과 수학적 모델을 이용하는 정량적 접근법(quantitative approaches)으로 나눌 수 있다.[1] 정성적 고장 진단방법 중에서 널리 이용되는 몇 가지를 추론방법에 의하여 분류하면 다음과 같다.[2] 먼저 고장 트리를 이용하는 방법으로, 이는 이상조건(failure condition)을 야기하는 고장들의 조합과 시퀀스를 미리 트리로 구성하는 방법이다. 그러나 이 접근법은 복잡한 공정의 고장 트리를 구성하는 데에 어려움이 많으며, 시퀀스를 정할 때 작성자의 주관성이 개입되는 등의 문제가 있다. 또한 다중 고장에는 대처하지 못하며, 한 공정에 대하여 구성한 트리를 다른 공정에 이용하지 못하는 단점이 있다. 다음으로는 공정변수간의 인과관계를 노드와 아크들로 나타낸 부호유향도표(signed directed diagram)를 이용하는 인과모델(cause-effect model)방법이 있다. 이 방법의 목적은 관측된 고장을 설명할 수 있는 가능한 가정집단을 찾는 것으로, 상대적으로 적은 양의 정보가 필요하다. 그러나 이 방법은 모든 시스템의 증상들이 하나의 원인에서 비롯되는 것이라고 가정하기

때문에 고장 트리를 이용하는 방법과 마찬가지로 다중고장을 다룰 수 없다. 또한 진단 알고리즘이 복잡하고, 추론과정을 현장조업자가 이해하기 어려우며, 실제 공정에 적용했을 때 해석도(resolution)가 낮아 너무 많은 결함후보를 제시한다.

현재까지 연구발표된 고장진단 전문가시스템은 위와 같은 접근법을 기본으로 한 것들과 정량적 혹은 정성적 모델을 첨부하여 효용성을 높힌 것들이다. 먼저 고장 트리를 이용한 전문가시스템 (Expert System, ES)의 예로는 선박내 엔진 냉각시스템의 고장을 진단하는 것[3]과 시멘트 소성공정을 진단, 제어하는 것[4] 등이 있다. 이들은 결함트리를 규칙베이스 형태로 구현하기 용이하도록 논리합 / 곱 (and/or)트리를 이용하여 ES를 구성하였으나, 공정배치가 약간만 변해도 많은 규칙을 수정해야 하며, 입력정보의 중요성 정도가 구분되지 않는 등의 단점을 갖는다. 부호유향도표를 이용한 ES는 이를 규칙기반(rule base)형태로 바꾸어 계산시간의 단축과 해석도를 향상시켜 주기는 하였지만, 단일고장 가정과 생성규칙 (production rule) 형태의 지식표현에 따른 한계는 그대로 지니고 있다. 정성적 모델을 기반으로 하는 대표적인 ES로는 MODEX1[5], MODEX 2[6]라는 오프라인 대화식 고장진단 전문가시스템이 있다. 이것은 경험적지식과 간단한 정성적 단위모델에 의한 지식등 2단의 혼성지식 (hybride knowledge)표현방법을 갖는다. MODEX는 ES의 공정에 대한 일반성을 향상시키기 위하여 프레임으로 모듈화된 대상물의 지식표현을 행하였다. 위에서 살펴본 정성적 접근방법에 의한 전문가시스템들이 공통적으로 지닌 문제점은 경보 발

단점(alarm threshold)에서 민감도(sensitivity)에 관한 것이다. 즉 공정으로부터 입력되는 모든 자료를 부울리안(Boolean) 방식으로 높음, 정상 및 낮음의 형태로 나누는 경우, 정상에서 벗어나는 경보발단점을 높게하면 지나친 경보발생과 경보의 혼동은 방지할 수 있지만 고장진단의 민감도가 떨어져 위험상태를 발견할 수 없게되며, 반대로 경보발단점을 낮게하면 증상에 혼동이 생겨 정확한 진단이 어렵게 된다. 이러한 문제는 측정자료의 확률분포 등을 도입하여 해결할 수 있으나 매우 복잡한 양상을 띠게 된다. 다른 방법으로는 경험화된 정량적 관계식을 기반으로 전문가시스템을 구성하는 것으로 동적모사를 이용하게 되어 공정특수성(process - specific)을 지닌다. 이의 대표적인 ES가 실제 화학공장에 도입하여 운용하고 있는 FALCON[7] 이다. FALCON은 실용화된 경험적 지식(experimental knowledge) 접근법을 이용한 대표적인 ES로 실제공정과 동적모사 결과에서 발생한 증상을 온라인으로 받아 최소고장집합에 정의된 모든 고장을 진단할 수 있다.

이상에서 살펴본 바와 같이 지식베이스를 이용하는 이상진단 전문가 시스템개발의 추세는 종래의 경험적 법칙만을 이용하는데서 생기는 한계를 극복하기 위하여 어떤 형태로든지 모델을 이용하는 방향으로 나아가고 있다. 다음에서는 이와 같은 모델을 이용하는 대표적인 두개의 전문가시스템 MODE와 FALCON에 대하여 간단히 살펴보고자 한다.

정량적 수지식만을 이용하여 공정의 이상을 진단하는 방법은 공정의 수학적 모델을 구성하고 이 모델이 가지는 공정 매개변수

의 추정에 의한 방법, 공정과 모델의 출력을 비교하는 방법 등이 있다.[8] 혹은 공정모델을 이용하지 않고 공정출력의 변화경향을 이용하거나 공정의 효율이나 연료소모비 등을 이용하는 방법도 정성적 접근법에 대비하여 이 범주에 포함시킬 수 있다. 모델을 이용하는 경우에는 정확한 수학적 모델식이 요구되지만 잘 정의되어 있는 공정에 대해서는 적당한 가정을 도입하면 실제 공정과 유사한 특성을 갖으며, 이상진단에 적합한 모양의 모델을 구할 수 있다. 공정모델의 상태변수 추정에 의한 이상진단은 공정이상을 계단함수나 펄스함수로 가정하여 모델식에 포함하고, 모델예측오차가 정상 모드와 이상모드에서 다른 시퀀스를 갖는 성질을 이용하여 이 두 시퀀스를 분별함으로써 수행된다. 공정매개변수의 추정에 의한 방법에는 변수추정기법이 필요하며 널리 쓰이는 방법들로는 최소자승법과 휠터링방법 등이 있으며, 이들을 변형하여 이용하기도 한다. 이에 관한 자세한 내용은 5장에서 살펴 보기로 한다.

2. 정성적 모델을 이용한 전문가시스템

(1) MODEX1(Model Oriented Diagnostic EXpert)[5]

전문가시스템을 구성하는 핵심적인 세가지 요소인 지식표현(knowledge representation), 추론기관(inference engine)및 설명 능력 (explanation facility)을 중점으로 살펴보기로 한다.

지식표현 방법에는 동적지식 (behavioral knowledge)과 휴리스틱 (heuristic) 정보의 저장에 알맞는 그림 2-1과 같은 생성규

칙(production rule)과 대상물들의 구조적 특성이나 연결성을 표현하는데 적당하며, 여러 관련 지식요소를 하나의 엔트리로 묶을 때 편리한 그림 2-2 와 같은 프레임(frame)구조가 있다. 이 프레임구조는 한 대상(object)에서 그의 곁가지로의 정보전달이 계승(inheritance)이라는 기구를 통해 자연스럽게 수행된다.

```
IF 〈valve〉 〈state〉 is normal
 and 〈inlet〉 〈stream〉 〈flow-rate〉 is 〈fin〉 that is
 not nil
THEN 〈outlet stream〉 〈flow-rate〉 is 〈fin〉
```

그림 2-1. 생성규칙의 예

```
inlet connected to:                                    valve1
outlet connected to:                                   tank1
flow-rate:                                             low
pressure:                                              high
temperature:                                           normal
state of pipe(through which stream flows):blocked
```

그림 2-2. 공정흐름에 대한 프레임의 예

MODEX에서는 공정의 정보는 프레임구조로 표현하고, 수지식이나 제약조건 등은 생성규칙으로 저장하는 혼성지식 표현방법을 도입하였다. 즉 모든 단위공정과 공정흐름(stream)은 이들의 기본

적인 정보를 갖는 프레임으로 표현하고, 상태변수들(온도, 압력, 유량, 수위 등)은 낮음, 정상, 높음의 부울리안 형태로 상태를 나타내며 낮음과 높음은 이상으로 처리한다. 밸브와 펌프등의 장치들은 닫힘, 열림, 이상, 막힘, 유출 등으로 표현한다. 그림 2-2 에서 입출력(connect) 흐름은 공정의 배열을 표현하기에 용이하다. 프레임의 값은 ES의 질문에 사용자가 응답하거나, ES 스스로 공정입력자료로부터 추론에 의해 채워 넣는다. 이와 같은 공정정보를 갖는 프레임을 그림 2-2 에 대하여 LISP언어로 나타내면 다음과 같다.

```
(stream ^name stream 3 ^inlet valve1 ^outlet tank1
^flow-rate low ^pressure high ^temperature normal
^pipe pipe 3 ^state blocked)
```

생성규칙은 그림 2-1 과 같은 형태로 다음과 같은 여러가지 규칙을 저장한다. 먼저 물질수지식이나 에너지수지식 및 보존법칙들로부터 유도되는 제약조건들을 표현한다. 다음으로는 한 상태변수가 다른변수에 미치는 영향을 나타내는 합류방정식(confluence equation)을 규칙으로 저장한다. 예를 들어 밸브가 정상적으로 동작하면,

$$\frac{\partial(\text{flow out})}{\partial(\text{inlet stream pressure})} = +$$

이것을 생성규칙으로 표현하면 다음과 같다.

IF 〈vlave〉 〈state〉 is normal

and 〈inlet stream〉 pressure is 〈low〉

THEN 〈outlet stream〉 flow-rate is 〈low〉

이 외에 상태변수 값의 낮음과 높음의 지엽적인 원인(local cause)들을 설명할 수 있는 고장모델의 저장고로 생성규칙을 이용한다. 예를 들어 흐름의 유속이 낮은 경우는 그림 2-3 의 가능한 규칙들 중에서 찾을 수 있다.

주어진 공정에서 나타나는 여러가지 증상의 원인이 확실하게 지식베이스내에 저장되어 있지 않으므로, MODEX에서는 프레임과 생성규칙으로 표현되는 인과모델을 이용하여 추론을 행한다. 위에서 살펴본 바와 같이 지식베이스는 단위공정의 인과모델, 공정배열에 관한 정보, 공정의 기본 이론, 단위공정의 고장모델 등으로 구성되어 있으므로 지식베이스에 포함되어 있는 단위공정들의 배열이 다른 경우에도 지식베이스를 다시 작성하는 불편함이 없다. 추론은 사용자에 의해 낮음 혹은 높음으로 이상입력이 주어지면 작업메모리내의 단위공정과 흐름(stream)의 상태(status)가 active 로 변한다. 이 후 지식베이스내의 규칙을 검사하여, 초기변수가 영향을 준 다른 상태변수를 쌍방향으로 추출한다. 문제축소(problem reduction)는 위의 공정의 비정상성의 원인을 지식베이스를 체계적으로 검사함으로써 수행되며 검색형태는 논리곱/합(and/or) 트리로 표현될 수 있다. 근본원인이 찾아지면 이 근본원인에서 초기

1 . pipe break or blockage or

2 . if upstream is connected to a valve and either

a . valve stuck closed or

b . valve opening properly set and inlet stream to valve has low pressure or

c . valve opening properly set and inlet stream to valve has low flow-rate or

3 . if upstream connected to a pump

a . pump off

b . pump failure

c . pump is functioning properly and pressure of inlet stream to pump is low

4 . if downstream connected to a closed valve or

5 . if downstream connected to a failed pump.

그림 2-3. 간단한 고장모델

관측된 이상상태로 거슬러 올라가며 상태변수와 관련된 다른 고장을 조사하여 다중원인에 대한 검색을 수행한다. MODEX의 설명능력은 4단계로 구성되어 있고 사용자가 이유를 물으면 단계를 따라 자세한 대답을 얻을 수 있다.

이상에서 살펴본 바와 같이 MODEX 1은 특정공정에 적합한 실험적지식 (shallow, compiled knowledge)대신 심층적 (deep-level) 지식이라 불리는 단위공정의 기능과 구조에 근거한 인과모델에 의

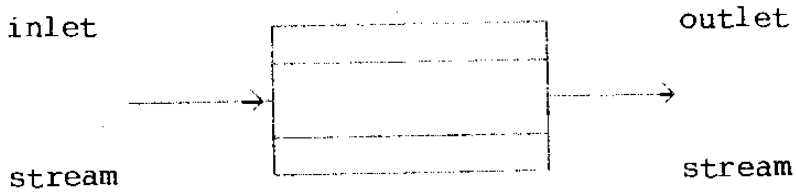
한 진단을 행함으로써 공정일반성 (process-general) 을 갖추었고, 쌍방추론에 의한 다중 고장원인에 대처할 수 있다. 그러나, 단위공정모델이 국부적인 인과관계 등을 독립적으로 표현하여 사용함으로 인하여 실현성이 없는 이상 전달양식이 생성되고, 따라서 선명하게 고장의 원인을 찾아내지 못하는 등의 진단해석도가 저하된다. 이러한 문제는 정성적모델을 사용하는 경우에 발생하는 근본적인 문제점으로 보다 많은 제약식이나, 잉여 제약식과 함께 이용하는 방법이 제시되고 있다. 또한 화학공정 전문가시스템으로 갖추어야 할 실시간대 진단 (on-line diagnosis) 이 않되는 것도 큰 약점으로 꼽힌다.

(2) MODEX 2 [6]

MODEX 1 이 인과모델만을 이용하여 진단함으로써 진단능력이 떨어지는 것을 보완하기 위하여 MODEX 2 는 실험적 지식 (compiled knowledge) 과 심층적 지식 (deep-level knowledge) 을 함께 이용하는 객체지향 (object-oriented) 2 단구조 (two-tier architecture) 를 갖는 전문가 진단시스템이다. 즉 윗 단에는 MODEX1 에서와 같이 일반적이고 공정에 종속되지 않는 단위공정의 개념, 제약조건 및 양태인 심층적 지식을, 아랫 단에는 과거에 습득한 지식들인 실험적 지식을 각각 저장한다.

화학공정에서의 단위공정들은 고유의 기능 외에도 문제를 보는 시각에 따라 한 개의 구조물로 생각할 수 있다. 따라서 단위공정에 관한 지식을 적용하는 데에도 구조적이거나 혹은 기능적으

로 지식을 분해 (decomposition of knowledge) 할 수 있다. 구조적 분해는 플랜트를 공간적인 배열 상태에 따라 혹은 쉽게 인식할 수 있는 입-출력흐름의 보조계 (subsystem) 로 나누는 것이고 기능적 분할은 고유 기능에 따라 보조계의 집합들로 나누어 생각하는 것이다. 예를 들어 그림 2-4 와 같은 열교환기는 열전달 기능 외에도 하나의 도관 (conduit) 의 역할을 담당하는 것이다.



(conduit or heat transfer function)

그림 2-4 . 열교환기의 구조적 및 기능적 분해

MODEX 2 에서는 공간배열과 장치간의 연결은 공정의 비정상성을 진단, 검색하는데 가이드를 하며, 기능은 고장모델에 포함시켜 위의 두가지 분해전략을 모두 이용하고 있다. 예를 들어 공정흐름이 낮은 경우, 그 원인을 물질흐름의 정방향과 역방향을 검색한다. 공정흐름이 그림 2-4 와 같이 열교환기의 출력에 연결되어 있다면, 구조적 분해에 의하여 열교환기의 입력흐름을 진단한다. 즉 열교환기를 하나의 도관으로 해석하여 막힘이나 유출이 있는지를 검색하는 것이다. 만약에 열교환기의 출력흐름의 온도까지 이상하다면 열

교환기의 입력흐름의 유량과 온도를 고려하며, 기능적 분해에 의하여 열교환기의 내부에 서리나 녹이 슬었는가를 진단한다.

다음은 단위 공정을 이루고 있는 보조계에 대한 구조적 분해를 이용한 제어루프에 대한 MODEX 2에서의 고려를 살펴보기로 한다. 제어루프의 기능은 조작변수를 이용하여 제어변수가 원하는 설정치를 유지하도록 하는 것이고, 이를 위하여 보조계로서 다음과 같은 것들을 갖는다고 생각한다. 즉, 루프에 의해 제어되고 있는 변수의 측정을 위한 검출단(sensor)과 조작변수 값을 바꿀 수 있는 장치—제어밸브—에 신호를 보내는 제어기와 제어계가 부착된 공정 및 공정흐름과 정보흐름을 갖는 라인등의 보조계들이다. 제어루프의 이상은 제어변수가 정상적인 조업 범위를 벗어나는 것으로 정하고 이러한 제어루프 이상의 원인은 다음의 결과로 생각한다.

* Internal loop disturbances which can either:(a) cause the control loop to malfunction(e.g. setpoint set to high or sensor biased); or (b) cause a change in the gain of the loop(e.g. reversed control valve action) or (c) valve stuck(causes gain to change to zero)
* External loop disturbances which can either: (a) are so large that the control loop cannot cancel the disturbance completely, i.e. the control loop action is saturated; or (b) cause a disturbance in the manipulated variable.

제어변수와 연관된 공정의 비정상성에 대한 진단은 인접한 단위장치내의 하나의 고장원인을 찾는 것 보다 더 많이 요구된다. 귀환제어루프(feedback control loop)에 대한 연산자는 그림 2-5와 같이 표현할 수 있다.

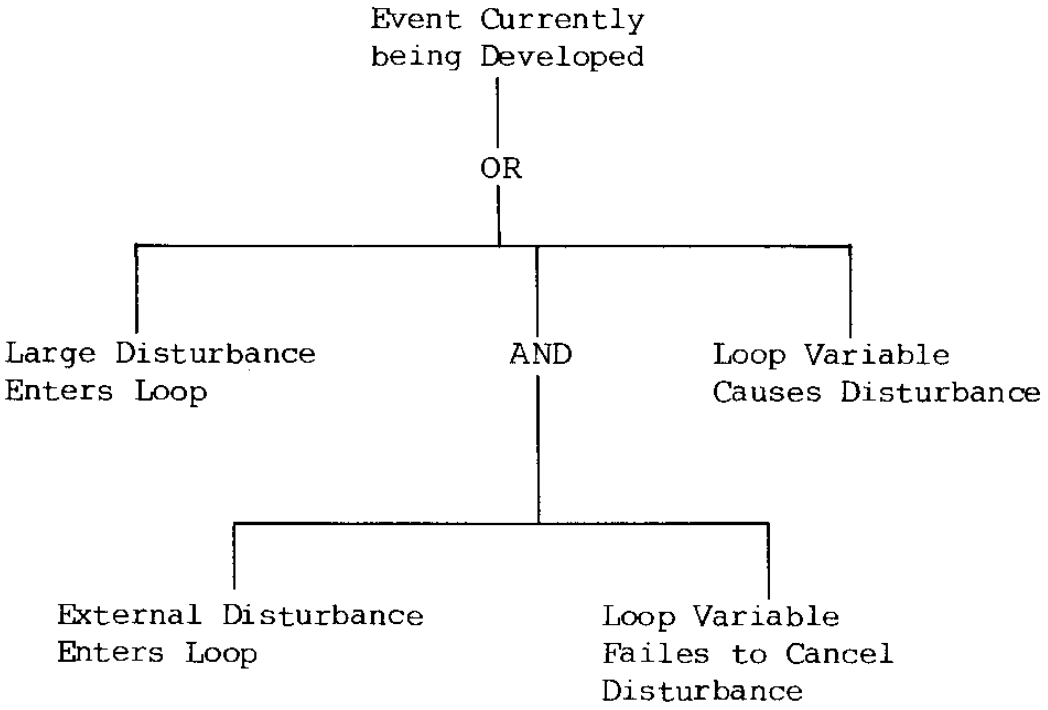


그림 2-5 . 제어루프에 대한 연산자

객체지향(object-oriented) 구조에서는 제어루프도 하나의 프레임을 이용하여 나타낼 수 있다. 즉 제어루프의 성격을 이름, 양태, 제어변수, 조작변수, 제어대상계 등으로 규정한다. 귀환 제어루프의 제어변수가 비정상인 경우, ES는 먼저 그 원인을 내부루프 간섭에서 찾는다. 다음은 외부루프 및 이득의 역행 여부 등을 순차적으로 검색하고 최후에 조작변수에 외부변화가 가해졌는지를 조

사한다.

MODEX 2는 실험적 지식과 심층적 지식을 혼성으로 이용하기 위하여 2단의 객체지향 접근을 시도하였다. 객체지향 프로그래밍에서 수행(action)은 객체(object)들간의 메시지(message)에 의해 이루어진다. 객체는 methods라 불리는 과정을 수행하여 메시지에 응답한다. 객체에 붙어 있는 methods는 각종 제약조건들 합류 방정식(confluence equation) 및 인과모델들이다. Confluence methods는 특정장치나 단위공정의 confluence와 관련된 실험적 지식(behavioral knowledge)을 모은다. 한 객체가 그의 confluence method를 수행하도록 메시지가 전달되면, 그 method내에 있는 모든 가능성들이 어떠한 추론이라도 유추가 될 것인지를 결정하기 위하여 검색된다. 고장모델에 해당하는 methods는 고장의 모든 가능한 지엽적인 원인을 시험하는 집합들의 리스트이다. Methods내의 규칙의 순서, 즉 고장의 지엽적 원인의 시험순서는 고장들의 빈도에 의해 결정된다. 각 단위장치 계열내의 상태변수는 이러한 method를 하나씩 갖는다.(예 : stream:diagnosis-temperature, tank:diagnosis-level 등) 이와같은 객체지향 고장모델의 2단 지식베이스를 탱크와 반응기 시스템에 대하여 살펴보면, 첫 단에는 공정에 종속적이며 경험적인 지식을 다음과 같이 저장한다.

```
IF    reactorl level is low
THEN  check tank 1 level first
```

이러한 지식은 비록 같은 탱크와 반응기의 공정배치를 갖는

경우에도 공정특성에 따라 항상 만족하는 규칙은 아니지만, 검색공간을 줄임으로 진단효율을 높히는 기능을 준다. 다음 단은 심층적 지식으로 탱크에 연결된 다른 장치를 고려하지 않고 입출력 만을 고려하므로 공정 일반성을 갖는다.

```
IF      the reactorl level is low
THEN    the possible causes are:
            (a) the inlet flowrate is low    or
            (b) the outlet flowrate is high  or
            (c) there is a reactor leak
```

추론방법은 먼저 실험적 지식베이스 안에서 short-cut 방법을 이용하여 해를 찾고, 실패하면 제 일법칙모델에 기초한 추론으로 고장원인을 찾는다. 고장의 근본적 원인이 지엽적 원인(local cause)에 있으면 다시 첫단의 실험적 지식베이스에서 나머지 문제의 해를 short-cut 방법으로 찾는다. 탐색은 깊이우선(depth-first) 제어를 이용하여 초기 관측된 모든 공정의 비정상성을 주 비망록(master agenda list)에 둔다. 2단의 경우에 비망록을 기초로 한 추론 제어 알고리즘의 예는 그림 2-6 과 같다. 진단에서는 가설과 검증의 추론 과정이 이용된다. 즉 비정상 가설이 공정구조와 단위공정의 특성으로부터 추론에 의해 자동적으로 생성된다. 검증은 제시되는 가설을 평가하는 사용자에 의해 수행된다.

이상에서 살펴 본 바와 같이 MODEX 2는 공정 일반성과 다중 고장원인의 진단 등 이제까지 발표된 화학공정 진단용 전문가

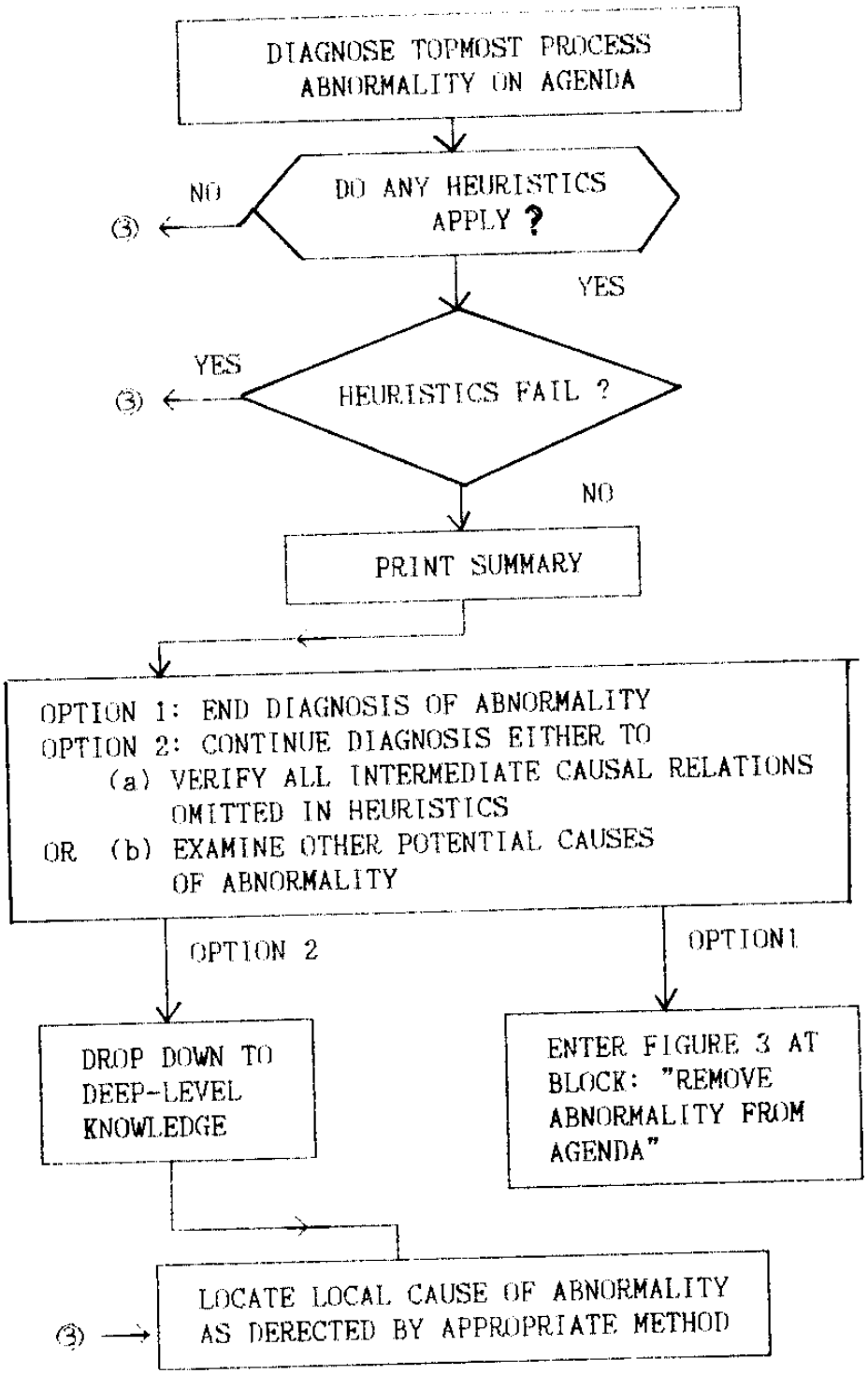


그림 2-6. 비망록에 기초한 추론제어 알고리즘

시스템(ES)들 중에서는 ES가 지향하는 좋은 점들을 많이 지니고 있지만 가장 큰 단점으로 오프라인으로 이용한다는 것이다.

3. 정량적 모델을 이용한 전문가시스템

FALCON[7] 은 Delaware대학, E.I. duPont 및 Foxoboro사가 공동으로 연구한 화학공정의 온라인 실시간대 이상진단 전문가시스템으로 실용화된 경험적 지식 외에 정량적 관계식을 사용한 대표적인 시스템이다. 대상공정은 adipic acid 제조공정으로 온라인 자료수집이 가능하며 컴퓨터 감시기능이 갖추어져 있다. FALCON은 제어루프에 관련된 34개의 고장과 펌프와 열전달면에서의 fouling 등의 5개의 고장들로 구성되는 최소 고장집합에 정의된 모든 고장을 진단할 수 있도록 구성되어 있다. 또한 이 전문가시스템은 실험적 지식의 한계를 보완하기 위하여 대략 250개의 미분방정식과 1,000개 이상의 변수를 갖는 연속 교반반응기 등의 수학적모델을 구성하여 DELSIM으로 비정상상태의 해를 구하였다. 이러한 공정모사는 기본적으로 실제 공정고장의 상황에서 전문가시스템을 시험하기 위한 것이며, 또한 공정정보의 분류와 불규칙한 고장에 대하여 자료를 제공하는 점에서 보다 큰 의미를 갖는다. 지식베이스는 혼성형으로서 정성적 지식에 의해 보완되는 정량적 지식의 형태로 되어 있으며, 방정식들은 다음과 같은 일반적인 형태를 포함한다.

(1) material balance equations(written in terms of mea-

sured variables) around defined control volumes

(2) energy balance equations on various control volumes

(3) empirical equations relating measured variables

(4) equations for PI controllers

(5) valve curve correlations relating flow measurement to controller outputs

(6) equations for calculation of heat transfer coefficients

실험적 지식베이스를 구성하는 간단한 휴리스틱은 다음과 같다.

IF(off-gas flow is erratic)

THEN (separator level is high)

IF (process circulation flow is erratic)

THEN (separator level is very low)

IF (process flow signal is not noisy)

THEN (flow signal is unreliable)

추론엔진은 VAXLISP으로 구성하였고 Fortran언어와 통신이 가능하며, 추론논리는 실시간대 이용을 위하여 명제논리(propositional logic)를 응용하였다.

이상에서 간략하게 살펴본 결과 FALCON에서는 대부분의 화학공정이 운전조건에 따라 동특성과 보존법칙 및 물리화학 이론에

관한 적용이 다르기 때문에 화학공정 진단은 공정 종속적일 수 밖에 없다는 사실을 전제로 하였다. 따라서 비록 공정 종속적이기는 하지만 정량적 방정식과 실험적 지식을 함께 사용하여 효용성을 높히고, 온라인 실시간대에서 사용이 가능한 점이 큰 장점이라 하겠다.

제 2 절 공정제어용 전문가시스템

1. 개요

전산기를 이용한 정보처리방법은 최근 20년동안 빠른 속도로 발전해왔다. 단순한 반복계산은 물론이고 이제는 인간의 의사결정에도 이용되는 많은 소프트웨어들이 개발되고 있다. 기존의 소프트웨어들은 순차적인 프로그래밍언어로 작성되며, 프로그래밍의 진행순서가 확정될 수 있는 대상에 대해서만 수행가능한 구조를 갖는다. 그러나 인간의 의사결정능력을 모방하여 한정된 분야에 대한 전문가의 의사결정행위를 대행할 수 있고 더 나아가서 스스로 지식을 습득하여 새로운 의사결정을 창출할 수 있는 전문가시스템의 필요성이 나날이 증가되고 있다. 전문가시스템은 정량화된 수치정보뿐만아니라 정성적이고 개념적인 정보 및 인간의 주관적인 판단등도 프로그램화할 수 있는 구조와 지식표현방법을 갖추어야 한다. 인간활동의 대부분은 일을 계획하여 설계하고 결과를 분석하여 이로부터 계획을 새로이 수정하는 과정의 반복으로 요약될 수 있다. 그러나 특정분야에 전문적인 지식을 갖고있는 전문가가 자신에게

주어진 일을 처리하는 과정을 관찰해보면 문제해결을 위해 체계적인 접근방법을 채택함과 더불어 축적된 경험과 세련된 직관을 유효적절하게 사용함을 알 수 있다. 전문가시스템은 전문가의 시간적, 공간적 제약을 완화시키고, 전 산업분야에 전문가의 경험과 직관을 응용할 수 있다는 장점을 지니고 있다. 따라서 전문가시스템의 개발을 위한 제반기술에 대한 연구가 활발히 진행되고 있다.

본절에서는 공정제어분야에 국한하여, 제어용 전문가시스템 구성을 위한 연구사례를 조사하고 공정제어분야에 관련된 경험법칙을 수집하였다.

2. 연구사례

공정제어분야에 종사하는 엔지니어들이 일상적으로 접하는 문제는 다음과 같다.

* 제어대상 공정의 모델링
* 모델링의 불확실성에 대한 해결방법
* 제어목적의 설정
* 측정변수, 제어변수 및 조작변수의 선정
* 제어기의 선택
* 제어기의 성능분석 및 성능향상

이러한 문제들을 전체적으로 혹은 부분적으로 해결할 수 있도록하는 전문가시스템에 대한 연구사례가 보고되고 있다. 본절에서는 이중 대표적인 몇가지 사례를 조사하였다.

(1) ROBEX

1988년 Lewin과 Morari[9]는 공정에 대한 근사모델만을 가지고 주파수영역에서의 제어기 설계에 대한 지식이 없는 초보자라도 제어계를 설계할 수 있도록 도움을 주는 전문가시스템 ROBEX를 발표하였다. 그들은 공정의 모델링의 문제는 선형모델추정(Linear Model Identification)방법을 통해 해결될 수 있다고 전제하고 실제 공정의 비선형성으로부터 발생되는 모델링의 불확실성의 문제는 복소평면에 불확실성 영역을 설정하여 보완하도록하였다. 또한 제어계의 견실성(robustness)과 안정성(stability)에 대한 정량적인 성능지표(Performance Index)를 선정하여 제어계 설계의 기준이 되도록했다. 제어기는 Morari등이 1986년 발표한 IMC(Internal Model Control)[10]를 사용하였다.

ROBEX는 KBES(Knowledge-Based Expert System) 구조를 채택하여 고정된 구조의 정보네트워크(Information Network)로 RB(Rule-Base)를 표현하고 KB(Knowledge-Base)는 수행중 습득한 사용자의 주관적인 의사를 축적시킬 수 있도록 하였다. 또한 정보네트워크의 각 노우드에서 필요로하는 각종 수치계산에 이용되는 루틴들의 프로그래밍언어에 대한 제약도 최소화하도록했다. 사용되는 루틴들로는 Nyquist도표를 그리는 것, 시간영역에서의 공정모사를 위한 루틴, 안정성 및 성능을 계산하기 위한 루틴 그리고 불확실성 영역을 계산하기 위한 루틴등 그 종류와 기능이 매우 다양하다. 그리고 트리구조의 정보네트워크의 각 노우드를 제어계 설계과정에 따라 계층화하고 계층별로 수치계산의 복잡도에 따라 우

선순위를 부가하여 추론엔진(Inference Engine)으로 하여금 반복적으로 노우드를 평가하게 하여 사용자의 목적에 부합되는 제어계를 설계하도록 하였다. 그림 2-7 은 트리구조의 노우드를 나타낸다.

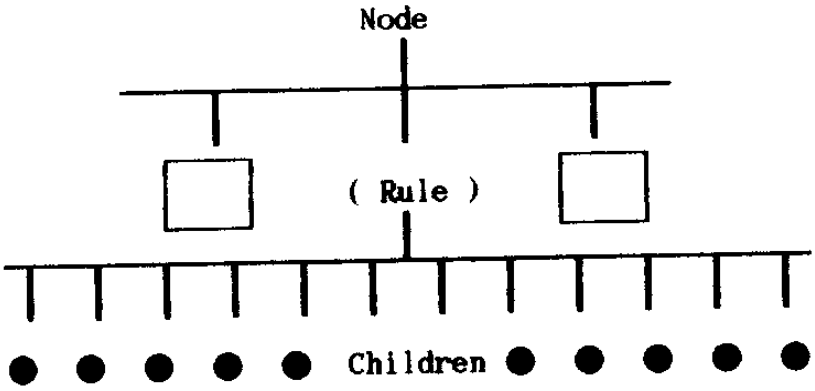


그림 2-7 . 일반적인 노우드 구조

시간영역에서의 공정모사에는 스탭, 램프등의 외란을 가할 수 있으며 시간영역 응답으로부터 ' settling time ', ' overshoot' 그리고 ' decay ratio ' 등의 만족도에 따라 순위를 정하여 목적함수를 만족시키는 제어계를 설계하도록 한다. 각 노우드의 반복 계산법칙은 ' downhill simplex ' 방법을 이용하였다. 발표된 바에 따르면 현재는 IMC SISO(Single-Input Single-Output) 제어계의 설계만이 가능하다. 사용언어는 FORTRAN과 C이며 상용 그래픽 패키지를 이용하여 IBM PC-AT상에서 그래픽 출력과 동시에 수행 가능하도록 되어있다.

(2) EMC (Expert Multivarible Control)

1988년 Tzouanas 등 [11]은 최근 개발된 많은 다변수 제어방법들이 예상했던 바와 달리 실제로 적용되는 사례가 적은 이유를

이들 제어방법이 공정의 정확한 선형모델을 필요로한다는 것과 공정에 이상이 발생한 경우 이에 대한 적절한 대응방법이 제시되어 있지 않다는 것으로 들고 이러한 단점을 해소할 수 있는 방법을 제시하였다. EMC는 공정의 이상상태를 진단하고 이에 따라 제어구조, 제어 알고리듬 및 제어기 파라메터를 적절히 변화시키는 기능을 수행한다. 그들은 측정장치의 고장, 조작부의 포화상태 그리고 공정 제약조건의 위배등의 공정이상이 기존의 이상진단 방법을 이용하여 완벽하게 진단될 수 있다고 가정하고 이에 따른 제어구조의 변경과 새로운 측정변수, 제어변수 및 조작변수의 선택등을 체계적으로 수행하는 방법을 제시하고 이를 증류탑의 탑상 및 탑저 농도제어의 경우를 예로 설명하였다.

그림 2-8 은 EMC의 구조를 나타낸다.

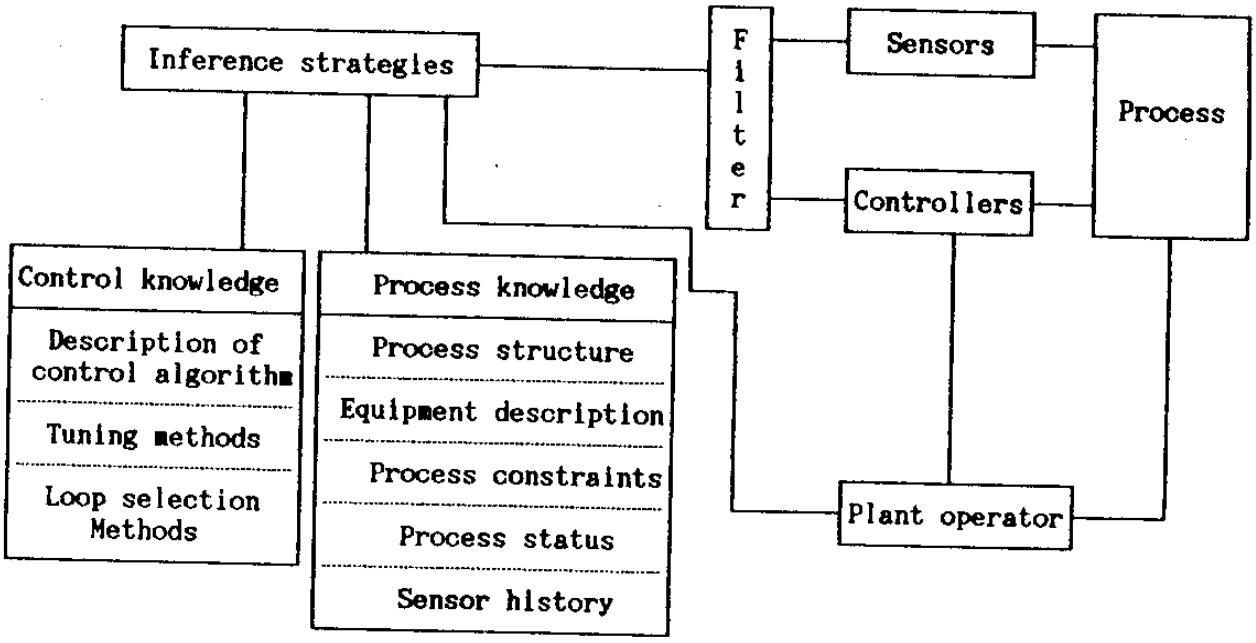


그림 2-8. EMC의 구조

EMC는 7개의 모듈로 확장과 보수가 용이한 구조를 갖고 있다. 지식표현방법은 각각의 지식의 특성을 분석하여 적합한 형태로 KB와 RB에 저장되도록 하였다. 예를들면 단위공정과 연결상태 등의 단순히 반복되는 형태의 지식등은 프레임(frame)으로 표현하고 제어법칙등은 생산법칙(Production Rule)의 형태로 표현하였다. 그리고 알고리듬화된 지식은 함수(function)형태로 표현하였다.

EMC의 RB에 저장된 증류탑제어에 관한 경험법칙들은 다음과 같다.

〈제어기 종류에 관한 경험법칙〉

* 농도제어의 경우에는 PI제어를 이용한다.
* 탑상 혹은 탑저 생산물의 유량으로 액위를 제어하는 경우에는 P제어를 이용한다.
* 환류액 혹은 스팀의 유량으로 액위를 제어하는 경우에는 PI제어를 이용한다.

〈제어구조에 관한 경험법칙〉

* 환류비가 5보다 큰 경우에는 R-V 제어구조를 사용하지 않는다.
* 증류혼합물의 상대휘발도가 1.2보다 작은 경우에는 온도측정을 이용한 농도제어구조를 사용하지 않는다.
* 환류액과 탑상 생산물의 유량 조절밸브가 포화상태일 경우 스팀의 유량으로 환류액 드럼의 액위를 제어한다.

〈탑저농도 센서가 고장인 경우의 경험법칙〉

* 탑저농도 제어를 제어하려면 탑저 생산물의 유량은 제어하지 않는다.
* 스팀의 유량과 원료 주입량의 비율을 일정하게 유지한다.

〈스팀 유량조절밸브가 포화 상태인 경우의 경험법칙〉

* 이 경우 탑저 혹은 탑상중 한 곳의 농도제어는 불가능하다.
* 공정의 이상발생시 액위제어가 농도제어보다 더 중요하다.

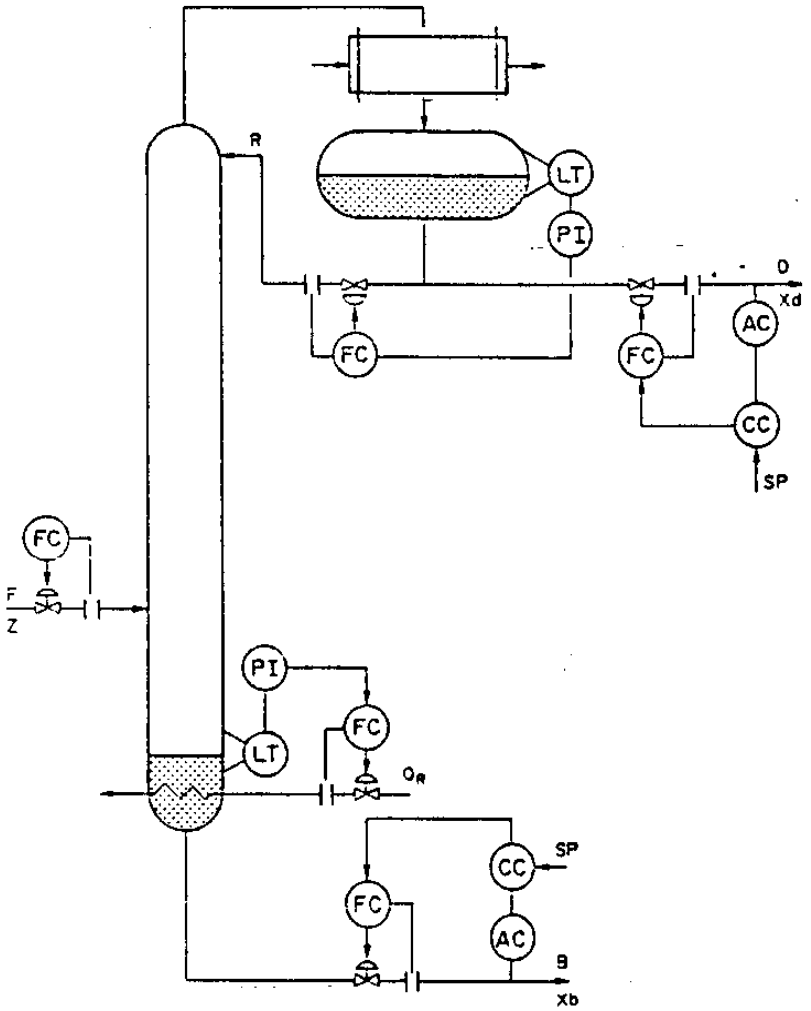


그림 2-9. (D, B)제어구조

EMC는 그림 2-9와 같은 (D, B) 구조의 증류탑제어계에 대해 탑저농도 센서가 고장인 경우 그림 2-10과 같이 제어구조의 변경시키고 제어 파라메터를 변화시킨 제어계를 제안한다. 한편, 스팀 유량조절밸브가 포화된 경우 그림 2-11과 같은 제어계의 변화를 제안한다.

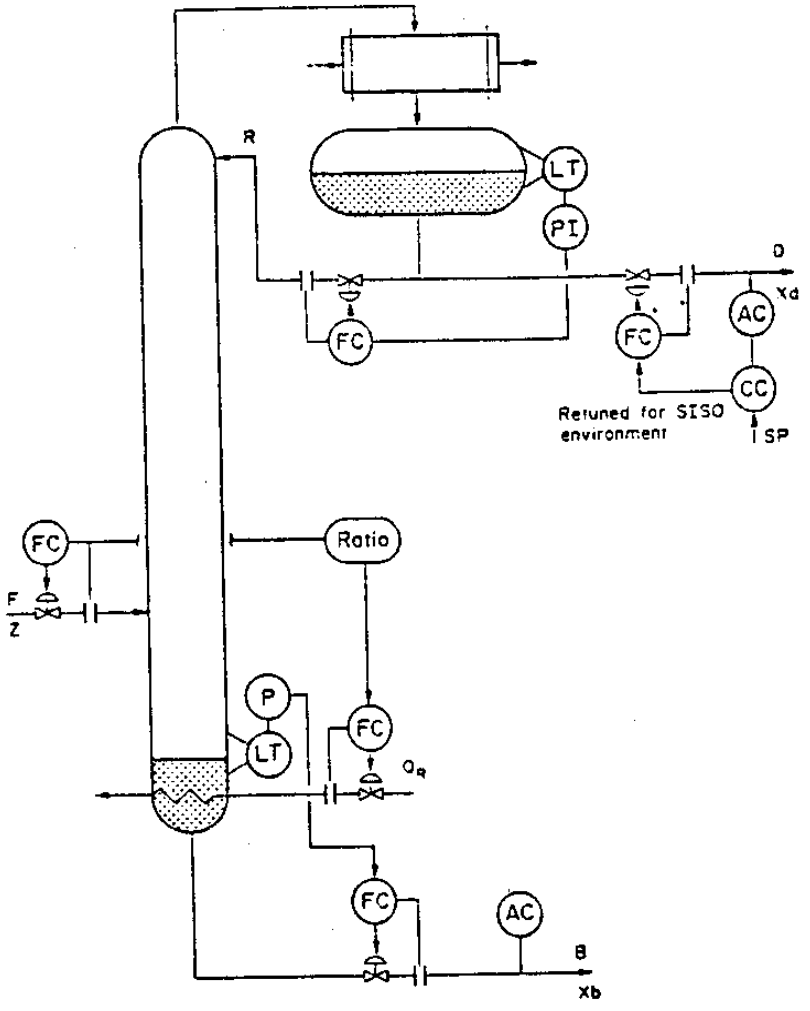


그림 2-10. 탑저농도 센서 고장시 제어계의 변화

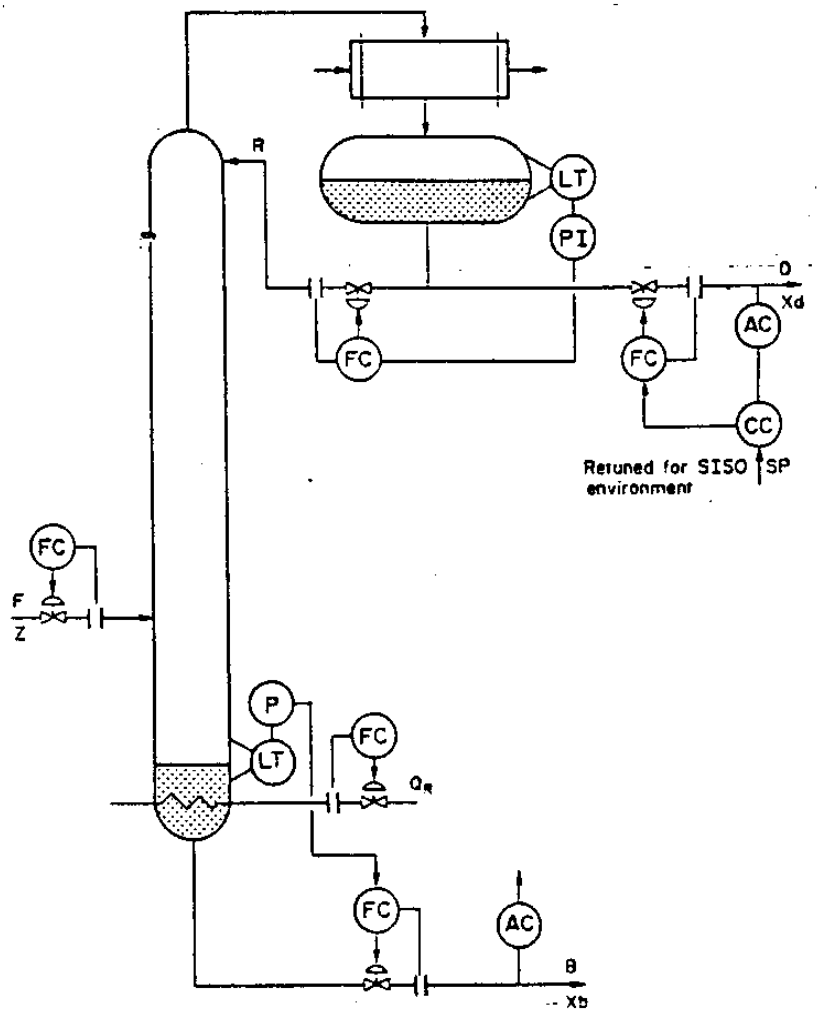


그림 2-11. 스팀 유량조절밸브 포화시 제어계의 변화

(3) Adaptive Regulator

1986년 Åström등[12]은 제어법칙의 구현과정에 실제적으로 포함된 많은 경험법칙을 전문가시스템으로 대체함으로써 기존의 방법론을 단순화시키고 더 나은 능력을 지닌 제어계를 구성하는 연구를 수행하였다. 기존의 제어계의 피이드백 루프의 일부로 전문가시스템을 도입하여 제어기능의 향상을 도모하였다. 제어성능에 영향을 미치는 중요한 요소인 경험법칙에 대한 이론적인 분석이 미약한 이유로 분석이 어려울뿐만 아니라 이론가들조차도 모르고 있는 경우가 있음을 들고 있다. 이러한 경험법칙, 특히 공정제어와 관련된 경험법칙을 전문가시스템에 표현하는 방법으로 RBES(Rule-Base Expert System)를 가장 적절한 방법으로 선택하고 각 구성요소가 갖추어야할 요건을 정리하였다.

정상상태 조업제어를 위한 적응 레귤레이터 설계를 예로 제어용 전문가시스템의 초기구조를 확립하고 RB에 70개의 법칙을 저장시켰다. 사용기종은 VAX 11/780이고 사용언어는 LISP와 OPS4이다. 그러나 진정한 의미의 실시간 제어용 전문가시스템의 구성을 위해서는 기존의 전문가시스템 개발용 도구가 적합하지 않으므로 이러한 도구의 개발이 필수적임을 강조하였다.

(4) CACE(Computer-Aided Control Engineering)-Ⅲ

1987년 James등[13]은 선형 SISO공정에 대한 'lead-lag' 보상기의 설계용 전문가시스템인 CACE-Ⅲ에 대한 연구결과를 발표하였다. 그들은 주파수응답 중 고주파 영역을 이득과 'lead' 보

상기로 조정하고 뒤이어 ' lag ' 보상기를 도입함으로써 주파수응답 중 저주파 영역을 조정했다.

수치계산용 루틴으로 CLADP(Cambridge Linear Analysis and Design Program)을 이용하였고 300개의 경험법칙을 생산법칙형태로 표현하였다. 사용된 전문가시스템 개발용 도구는 DELPHI이고 사용기종은 VAX 11/785이다.

(5) PI Controller Tuner

1987년 Porter등[14]은 PI제어기의 파라메터 튜닝에 관한 경험법칙을 근거로 튜닝용 전문가시스템을 개발하였다. 그들은 Ziegler와 Nichols 방법에 의한 제어기 파라메터의 초기 튜닝이후 패턴인식(Pattern-Recognition)방법에 의해 대부분의 튜닝이 이루어진다는 사실에 착안하여 수학적인 모델에 의존하지 않는 패턴인식 방법을 전문가시스템의 도구로 이용하였다. 공정의 설정치를 스텝으로 변화시키고 공정의 출력과 제어입력의 변화를 관찰함으로써 제어기 파라메터를 튜닝하였다. 그들은 폐회로 과도응답의 패턴을 다음과 같이 분류하였다.

* too low monotone
* too low oscillatory
* overshoot undershoot
* no overshoot undershoot
* no overshoot no undershoot

* overshoot no undershoot
* overshoot monotone
* overshoot oscillatory
* over safety limit

또한 개회로 응답특성은 다음과 같이 분류하였다.

* no delay monotone
* no delay oscillatory
* short delay monotone
* short delay oscillatory
* medium delay monotone
* medium delay oscillatory
* long delay monotone
* long delay oscillatory

위와 같이 폐회로와 개회로의 응답특성을 분류한 후 각각의 경우에 해당되는 튜닝방법을 RB로부터 찾아 파라메터를 튜닝한다. 튜닝법칙의 예는 다음과 같다.

폐회로 응답곡선이 그림 2-12와 같고 개회로응답이 ' medium delay monotone ' 일때, 즉,

IF (the control variable continues to increase after the transient first crosses the set-point)

THEN(reduce the integral gain by setting

$$K_I = \frac{A_1}{A_1 + A_2 + A_3} K_I \ )$$

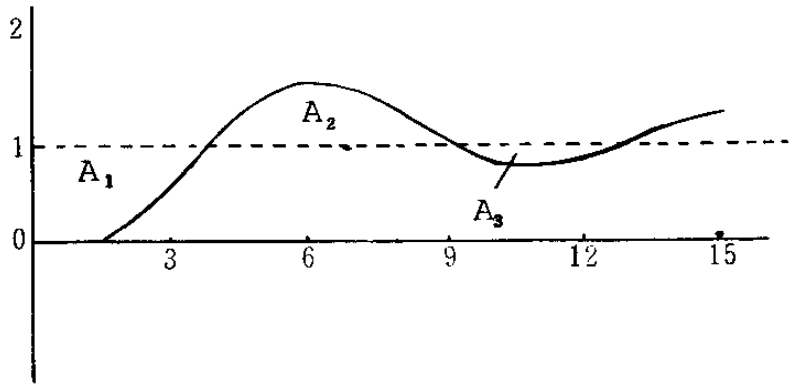


그림 2-12. 폐회로 과도응답 (overshoot undershoot)

(6) Distillation Control Design

1988년 Birky등[15]은 증류공정에 대한 SISO DCS(Distributed Control Systems)를 구성하는 전문가시스템을 발표하였다. 지식표현 방법으로 관용구적인 제어 (Idiomatic Control)와 GTST (Goal Tree-Success Tree)의 관계에 중점을 두고 연구했다. 제어계의 구성을 가능한 여러 후보구조로부터 적합하지 않은 것들을 차례로 제거해 나가는 과정으로 간주하고 후보구조의 선택의 폭이 좁아질 수록 좀 더 엄밀한 평가 기법을 도입하여 최종적인 제어구조를 선택하는 방법론을 제시했다. 그들은 Bristol에 의해 제안된 ' idiomatic ' 제어구조의 분석 및 합성법과 원자력 발전소의 고장진단에 응용되는 GTST모델에 의한 지식표현을 이용하였다.

증류공정의 제어시 고려되어야 할 사항을 표 2-1과 표 2-2와

같이 1차 및 2차 목표로 분류하고 제어전반에 관련된 구성요소를 관용구화하여 표 2-3 과 같이 표시하였다. 또한 증류공정 제어에 수반된 문제를 그림 2-13 , 그림 2-14 , 그림 2-15 , 그림 2-16 과 그림 2-17 로 GTST 모델로 표현하여 전문가시스템을 중추 역할을 담당하도록 하였다. 그리고 idiomatic제어 설계방법을 근거로 제어계 설계에 관련된 지식을 GTST모델로 표현하는 과정 전체가 프레임 (frame)으로 좀 더 쉽게 표현될 수 있음을 밝혔다.

3 . 기술현황 분석

이상에서 조사한 바와 같이 공정제어 분야는 문제의 영역이 잘 정의되어 있고 문제의 구성요소 각각에 대한 이론적인 연구가 활발하게 진행되고 있기 때문에 이를 이용한 다양한 전문가시스템의 개발이 추진되고 있는 실정이다. 작게는 제어기의 파라메터의 튜닝의 문제에서부터 크게는 대단위 공정 전체에 대한 제어계구성에까지, 또한 SISO제어계에서 MIMO제어계까지 그리고 고전적인 제어 알고리듬에서부터 최신 제어 알고리듬까지를 총 망라하여 각 분야의 전문가들이 축적한 각종 지식과 경험을 전문가시스템으로 구현하고 있다.

제어용 전문가시스템은 특성상 기존의 전문가시스템의 지식표현 방법중 RB를 이용하는 구조를 가장 많이 이용하고 있으나, 지식의 특성에 따라 그 표현방법을 다양화함으로써 전체적인 시스템의 효율을 향상시킬 수 있다. 또한 제어와 관련된 기존의 풍부한 알고리듬화된 지식을 전문가시스템이 이용할 수 있도록 하는 일

표 2-1. 증류공정 제어의 1차 목표의 관용구화

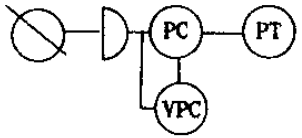
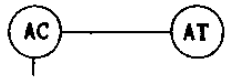
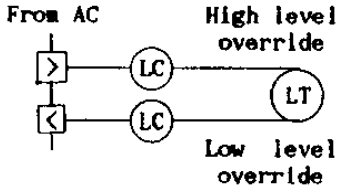
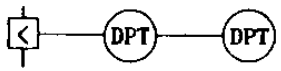
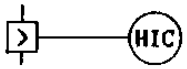
Issue	Type	Solution	Idiom
1. Minimize tower pressure to minimize energy consumption	Regulation	Use of valve position controller (VPC) on signal to pressure control valve	
2. Control composition at both ends of tower to minimize energy consumption	Regulation	Use two standard analyzer controllers	
3. Level control must never be lost	Constraint	A level controller must always be operational. Override composition controllers	
4. The tower must not flood	Constraint	Use a differential pressure controller (DPC) to measure the approach to flooding. The DPC takes over heat input if tower starts to flood	
5. The tower must not weep	Constraint	A minimum heat input must be supplied to the reboiler	

표 2-2. 증류공정 제어의 2차 목표의 관용구화

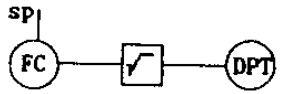
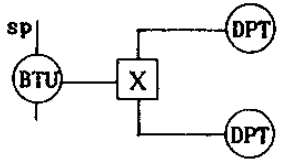
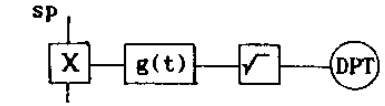
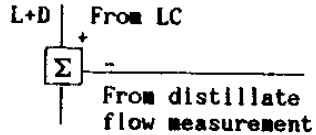
Issue	Type	Solution	Idiom
1. Correct for flow upsets quickly	Cascade	Cascade other slow controllers to flow loop	
2. Correct for heat input disturbances quickly	Cascade	Cascade other controllers to BTU loops	
3. Adjust tower operation for feed flow upsets	Feedforward	Ratio composition controllers to feed flow	
4. Avoid lag in reflux accumulator when manipulating distillate flow for composition control	Feedforward	Use sum of reflux and distillate flows for accumulator level control	

표 2-3. 일반적인 공정제어의 연구

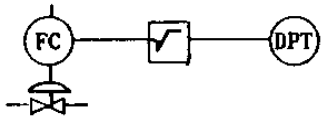
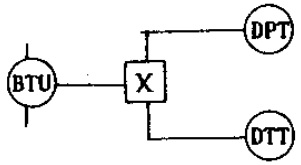
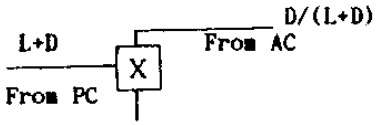
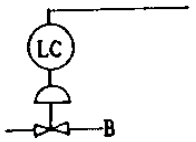
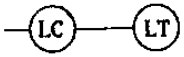
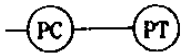
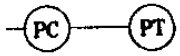
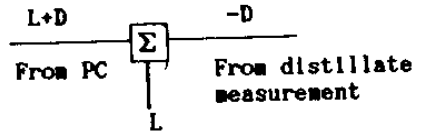
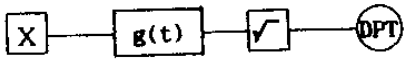
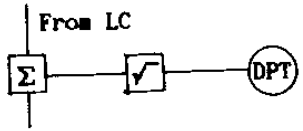
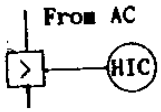
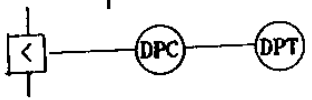
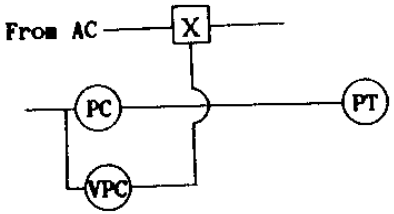
Idiom	Purpose	Variables	Components	Structure
1.	Stabilize flow	S(F,V)	I(DPT,√,FC)	
2.	Stabilize heat input	S(FDT,V)	I(DPT,DTT,X,BTU)	
3.	Eliminate stability problems because of the hysterisis of a valve		Valve position controller	
4.	Regulate composition	R(C,F)	I(AT,AC)	
5.	Regulate top composition using (L/D)	R(TC,DF)	I(X)	
6.	Regulate bottom level	R(BL, BV)	I(LC)	
7.	Regulate level of reflux drums	R(L,V)	I(LT,LC)	
8.	Regulate pressure at point in tower	R(P,OV)	I(PT,PC)	

표 2-3. (계속)

Idiom	Purpose	Variables	Components	Structure
9.	Regulate top pressure	R(P,RF or RF and DF)	I(PT,PC)	
10.	Regulate reflux flow	R(RF,RV)	I(Σ)	
11.	Regulate temperature near the feed	R(T,FV)	I(TT,TC)	
12.	Anticipate effect of feed flow on composition	A(FF,C)	I(DPT,√,g(t),X)	
13.	Anticipate effect of distillate flow on distillate composition	A(DF,DC)	I(DPT,√,Σ)	
14.	Constraint to achieve a minimum heat input to the tower	C^{LO}(FDT)	I(HIC,>)	
15.	Constrain differential pressure to avoid flooding	C^{HI}(DP,V)	I(DPT,DPC,<)	
16.	Constrain and stabilize pressure withe a bypass valve to achieve minimum energy consumption	CS(P,BP)	I(PT,PC,VPC,X)	

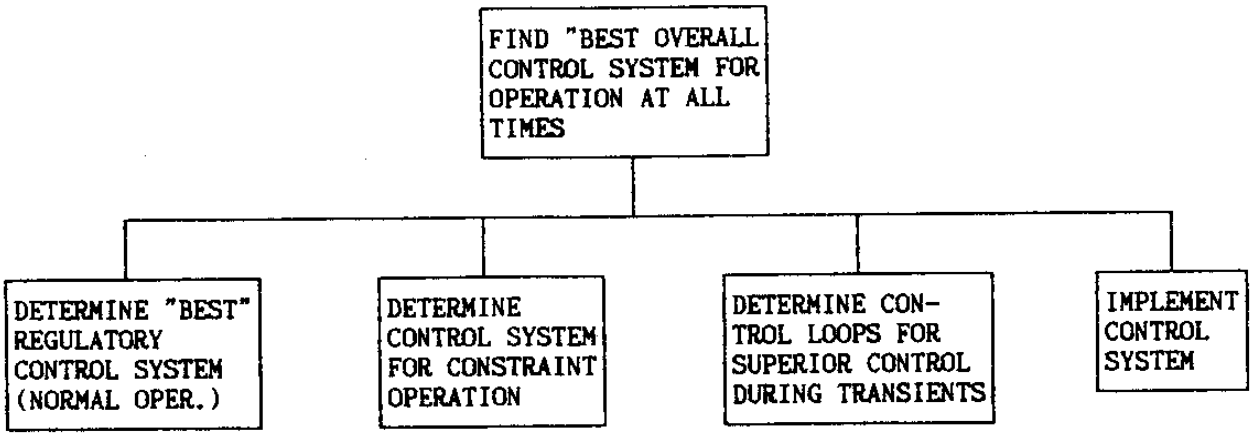


그림 2-13. Root goal and level 1 subgoals

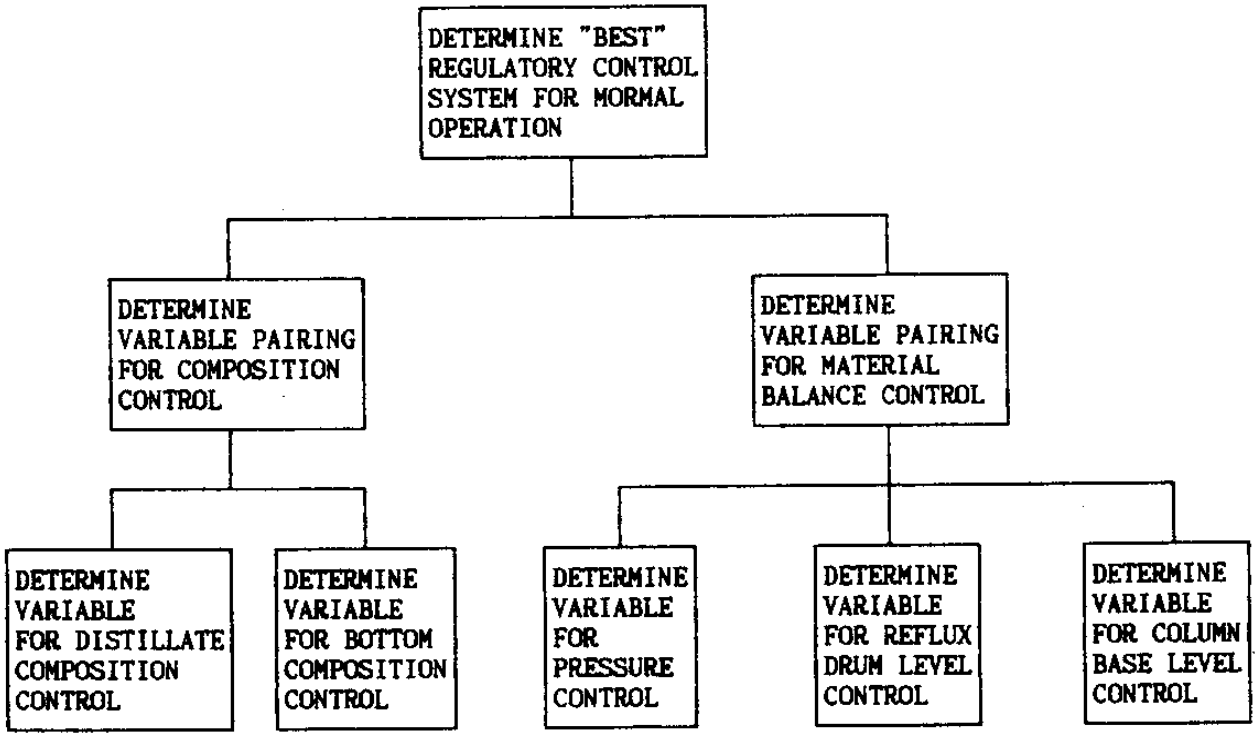


그림 2-14. Goal-tree for subgoal 1

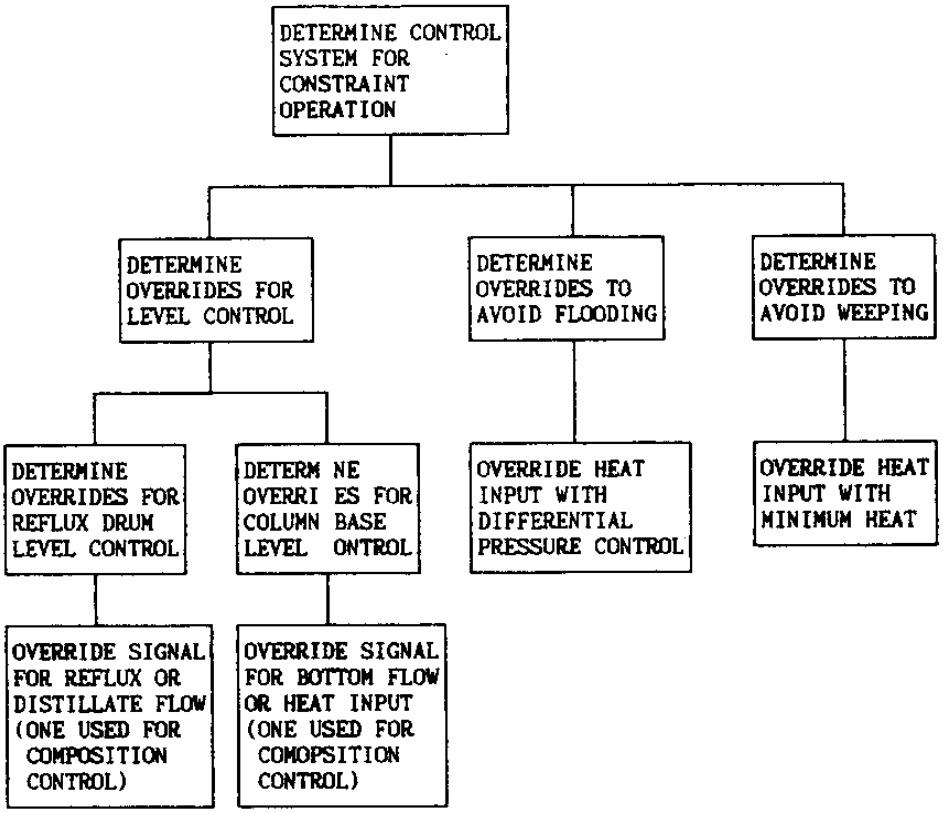


그림 2-15. Goal-tree for subgoal 2

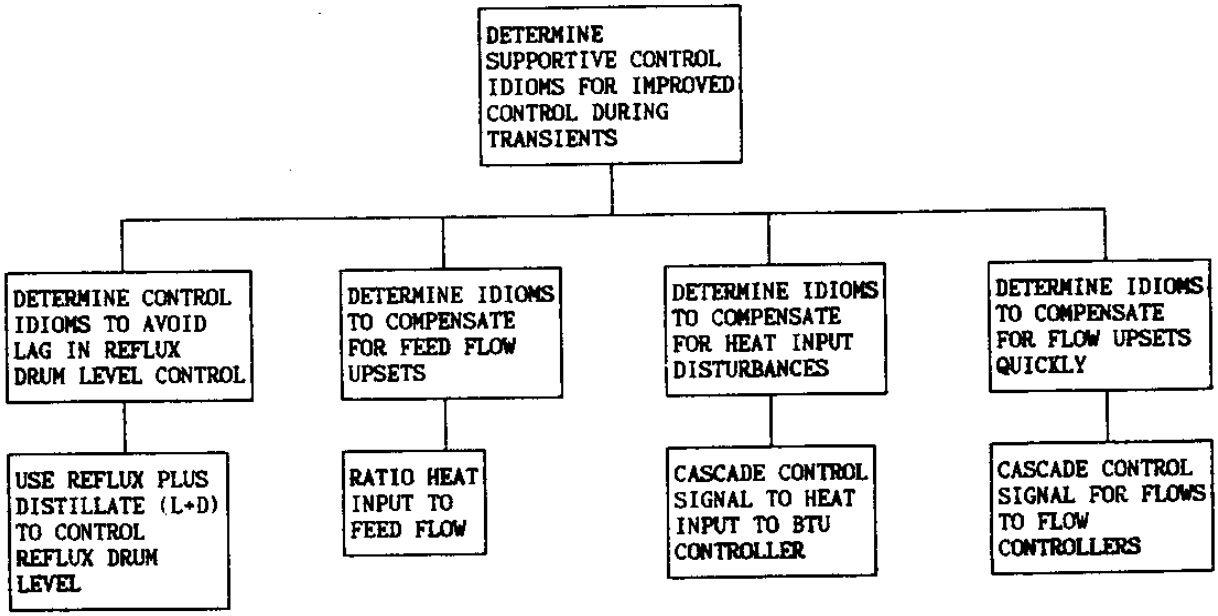


그림 2-16. Goal-tree for subgoal 3

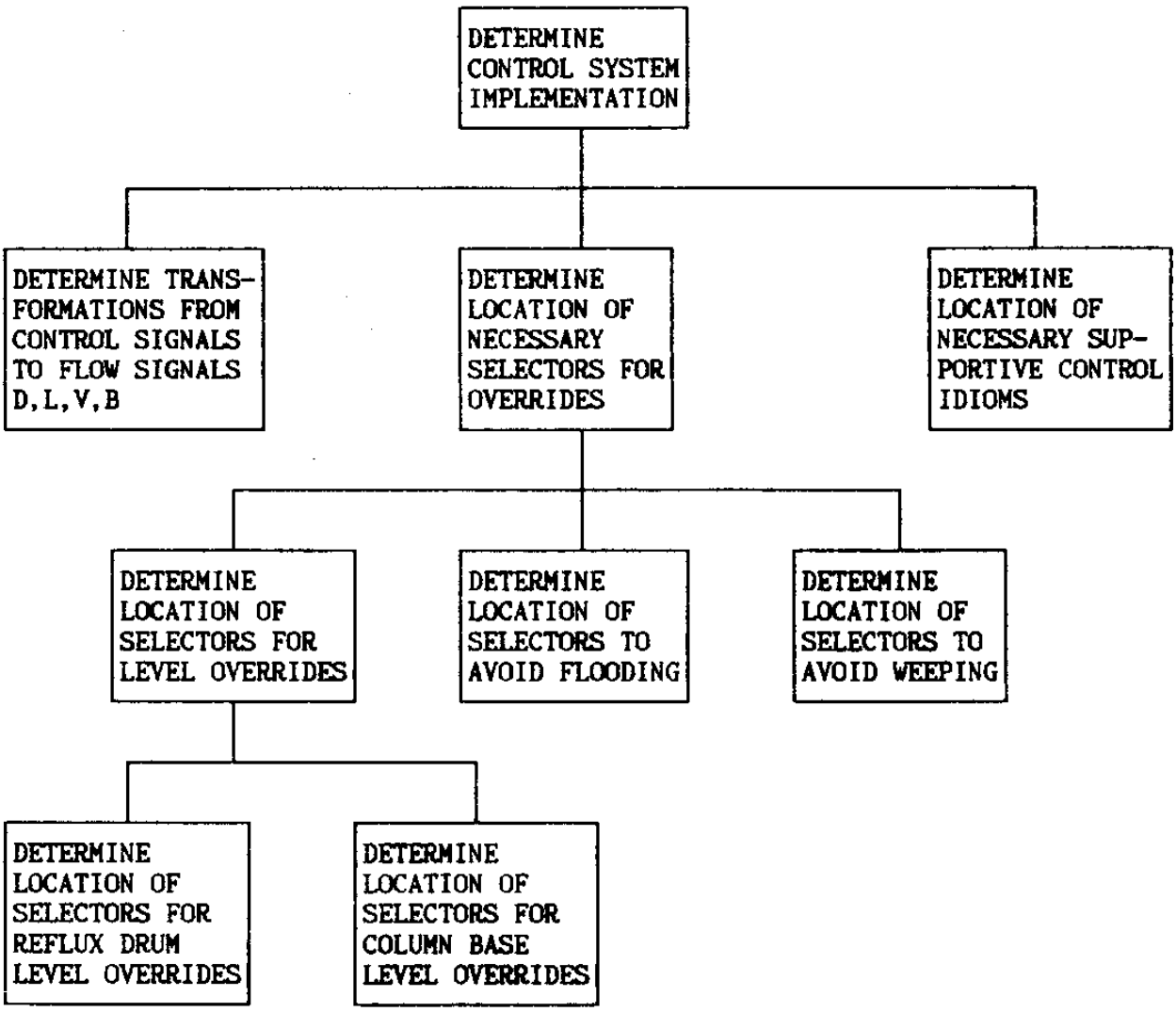


그림 2-17. Goal-tree for subgoal 4

도 매우 중요하다.

기존의 전문가시스템에서는 사용자가 좀더 쉽게 시스템을 사용할 수 있도록하는 것이 매우 중요한 과제였다. 그러나 제어용 전문가 시스템은 사용자의 편의뿐만 아니라 시스템과 제어대상이 되는 공정과의 정보교환에도 큰 비중을 두어야만 한다. 온라인 실시간 조업용 전문가시스템의 개발을 위해서는 일반적인 전문가시스템 개발용 도구의 기능은 부족한 점이 많다. 따라서 전문가시스템 개발용 도구에 의존하지 않고 사용 목적에 부합되는 형태로 전문가시스템을 개발하는 것이 바람직하다. 또한 제어관련 문제의 부분적인 해결방법을 제시하는 전문가시스템을 모듈화하여 개발하고 최종적으로 각각의 모듈을 종합하는 방식으로 제어분야 전반에 관련된 문제를 해결하는 전문가시스템 개발방법을 채택하여야 한다. 그리고 기술개발에 따라 쉽게 확장될 수 있는 구조를 지녀야한다.

참 고 문 헌

(1) 오전근, 윤인섭, "화학공정 결함진단을 위한 전문시스템 적용에 관한 고찰", '87 한국자동제어학술회의논문집, 674,1,1987.

(2) 오전근, "증상트리와 이상전파유향그래프를 이용한 공정 이상 전문가 시스템 연구", 박사학위논문, 서울대학교, 대한민국, 1990.

(3) Kumamoto, H., K. Ikenchi, K. Inoue and E.J. Henley, "Application of expert system techniques to fault diagnosis", The Chem. Eng. J., 29, 1, 1984.

(4) 윤인섭, 한종훈, "화학공장 조업중의 진단 및 제어", 화학공업과 기술, 15,4,1986.

(5) S.H. Rich and V. Venkatasubramanian, "Model-based reasoning in diagnostic expert systems for chemical process plants", Comput. Chem. Engng, 11, 111, 1987.

(6) V. Venkatasubramanian, "An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis", 12, 903, 1988.

(7) P.S. Dhurjati, D.E. Lamb and D.L. Chester, "Experience in the development of an expert system for fault diagnosis in a commercial scale chemical process", FOCAPO-87, Park City, Utah, 1987.

(8) R. Isermann, "Process fault detection based on modeling and estimation methods-a survey", Automatica, 20, 387, 1984.

(9) Lewin D.R. and M. Morari, ROBEX: An expert system for robust control sysnthesis, Comput. Chem. Engng., 12, 1187-1198, (1988).

(10) Laughlin D.L., K.G. Jordan and M. Morari, Internal model control and process uncertainty: mapping uncertainty regions for SISO controller design, Int. J. Control, 44, 1675-1698 (1986).

(11) Tzouanas V.K., C. Georgakis, W.L. Luyben and L.H. Ungar, Expert multivariable control, Comput. Chem. Engng., 9/10, 1065-1074, (1988).

(12) Åström K.J., J.J. Anton and K.E. Årzén, Expert control, Automatica, 3, 277-286 (1986).

(13) James J.R., D.K. Frederick and J.H. Taylor, Use of expert-systems programming techniques for the design of lead-lag compensators, IEE Proc.-D, 3, (1987).

(14) Porter B., A.G. Jones and C.B. McKeown, Real-time expert tuners for PI controllers, IEE Proc.-D, 4, (1987).

(15) Birky G.J., T.J. McAvoy and M. Modarres, An expert system for distillation control design, Comput. Chem. Engng., 9/10, 1045-1063, (1988).

제 3 장 화학공정지식의 표현

제 1 절 서 론

공정전문가 시스템에 있어서 공정지식의 정확하고 능률적인 표현은 전문가 시스템의 운용과 성능을 좌우하는 요소이다. 화학공정에 관련되는 지식은 매우 다양하며 그 규모도 상당히 크다. 간단한 공정장치인 경우 데이타의 수는 수백개 정도이지만 하나의 단위 플랜트인 경우 그 수는 수십만 개에 이른다. 이와같이 다양하고 광범위한 화학공정 데이타의 효율적인 처리를 위한 Software 시스템의 필요성은 오래전부터 대두되었으나 상용 시스템이 출현하여 실제로 이용되기 시작된 것은 1970 년대 중반 이후부터이다.

공정전문가 시스템에서의 공정지식의 표현은 IF-THEN 형식의 production rule 형태와 frame 방식이 주로 이용된다. Frame 방식은 경험적 지식 이외에도 공정의 단위 장치나 단위 process에 관련된 자료 및 수치 데이타의 표현에 적합하다. 이하에서는 제 4 장에서 소개될 prototype 전문가 시스템에서 이용되고 있는 데이타 베이스의 구성 및 운용방법을 다루고 있다. 제 2 절에서는 주로 tree 구조를 응용한 공정 데이타 베이스의 구성을 다루고 있으며 제 3 절에서는 tree 구조 이외에 linked list 구조를 조합한 공정 데이타 표현방법을 다루고 있다. 여기에서 다루고 있는 공정

지식의 표현방법이 전부는 물론 아니며 graph식 표현이나 도면을 활용한 시각적 표현방법도 생각할 수 있을 것이다. 보다 구체적이고 다양한 공정지식의 표현방법은 본 연구의 2차년도에 가서 보다 심도있게 연구될 것이다.

제2절 공정 데이타 베이스의 구성

1. 데이타 구조의 종류

인공 지능(AI)의 기초가 되는 대부분의 알고리듬들은 필수적으로 한 개 또는 그 이상의 데이타 베이스를 필요로 한다. 전문가 시스템에 들어온 모든 정보나 지식을 저장하는 데이타 베이스는 그 시스템의 목적에 맞는 구조를 가져야 하는데 널리 쓰이는 데이타 베이스의 구조로는 리스트(list)구조, stack, Queue, 트리(tree)구조 등을 들 수 있다. 이 구조들을 "C" 언어에 기초하여 살펴보기로 한다.

가. 리스트 구조

리스트 구조는 가장 중요한 데이타 구조중의 하나로 "node"라고 불리우는 원소 (element)들의 배열로 이루어진다. 리스트구조는 시작부분인 "head"와 끝부분인 "tail"을 가지게 된다. 리스트의 각 node는 모두 같은 구조를 갖는다. 리스트구조의 가장 주된 특징은 동적인 특성에 있다. 즉, 각 node가 언제라도 리스트에 추가되거나, 제거므로 배열(array)보다는 포인트(pointer) 를

사용하여야 효과적이다. 리스트의 각 node는 두 부분으로 구성된다. 그중 앞부분은 데이타를 저장한다. 이 데이타는 변수이거나 구조체 또는 구조체로의 포인터가 될 수 있다. node의 나머지 한 부분은 리스트의 다음 node의 위치를 나타내는 포인터가 된다. 즉 리스트는 반복적인 구조를 갖는 구조체로 구성된다. 간단한 예로써 리스트의 각 node에 정수형태 변수를 저장하는 경우를 살펴보자. 이 경우 node는 다음과 같이 정의될 수 있다.

```
Struct node
 {
   int          item
   struet  node  * next ;
 } ;
typedef struct node  node _ t ;
```

이 경우 정수 변수 item은 각 node에서 실제 데이타 값을 저장하게 되고 포인터 변수 next는 다음 node의 주소를 저장하게 된다. 위의 구조를 갖는 리스트를 그림으로 나타내면 다음과 같다.

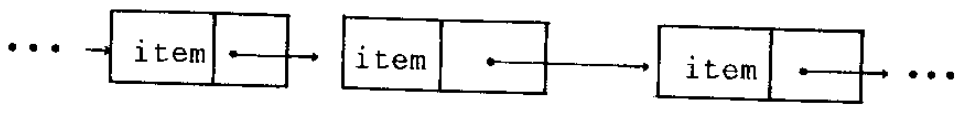


그림 3.1 간단한 list 구조 24)

그림 3.1 에서 보는 바와 같이 변수 " item "은 각 node 의

첫번째 구성원이 되며 포인터인 " next "는 각 node의 두번째 구성원이 된다. 대부분의 경우에 리스트의 각 node는 한개 이상의 데이타를 가질 수 있으므로 각 node를 구조체 형태로 선언하는 것이 보다 유용하다.

즉

```
Struct node
    { info_t  item ;
      struct node  * next;
    }
  typedef stuct node node_t
```

와 같고 여기서 info_t는 각 node에서 저장할 데이타 형태에 맞도록 미리 정의된 형태이다. 이렇게 만들어진 리스트 그림 3.1과 같은 node형태처럼 보이나 각 node가 단일 변수가 아니고 복잡한 구조체가 된다. 이렇게 구성된 리스트는 두가지 문제점을 갖는데 그 중 하나는 node에서 node로의 정보교환시 리스트 내부 구성원 각각의 assignment를 지정해 주어야 하므로 리스트 조작 함수들을 리스트를 구성할때마다 맞는 형태로 다시 써주어야 한다는 것이다. 이를 해결하기 위해 assign tunction을 도입하여 실제로는 불가능한 구조체간의 할당

$$S1 = S2$$

의 수행이 가능해진다. 즉 assign function은 S1 , S2가 각각 구조체일때

```
assign (S1, S2) ;
info_t  * S1, * S2 ;
{
    / * assign each component of struct2 to struct1 */
}
```

와 같은 형태가 되어 구조체 S1 에서 S2 로 세부 내용을 모두 옮겨주게 된다. 위와 같은 구조로 다양한 리스트 형태에 적합한 assign function 들을 미리 준비해 놓고 " # incude " 문을 사용하여 필요한 assign function 을 사용하면 된다. 두번째는 구조가 조금만 복잡하면 서로 다른기억 장소간에 많은 양의 데이타를 교환해야 된다는 것이다. 이런 문제들을 해결하기 위해 한쌍의 포인터로 구성된, 개선된 형태의 리스트 node 를 만들 수 있다. 우선 데이타가 저장되어 있는 주소만을 교환하면 되도록 하기위해 하나의 포인터에 item 의 데이타가 저장되어 있는 곳의 주소를 저장한다. 이 방법은 그림 3.2 에 나와있다.

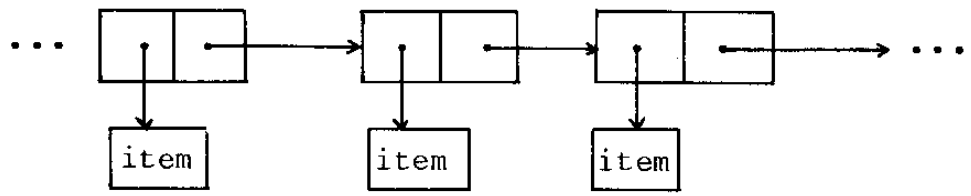


그림 3.2 개선된 리스트 구조 [24]

Linked LIST 구조를 사용하기 위해서는 우선 MALLOC 또 CALLOC 와 같은 function 을 이용하여 각 node 에 기억장소를

할당한다. 그리고 그 각 기억장소에 포인터나 데이타를 저장한다. 그러나 우선 그 리스트의 경계가 확정되어야 하는데 일반적으로 리스트의 끝을 결정하는 것은 NULL 포인터를 지정함으로써 쉽게 해결되나 (그림 3.3 참조), 리스트의 시작부분을 표시하는 몇가지 방법을 살펴보면 다음과 같다. 첫째로 리스트 시작부분의 번지수를 직접 포인터 변수에 저장하는 방법이 있다. 그러나 이 방법보다는 header node를 정의하여 사용하는 것이 더 편리하다. 이때 정의된 header node는 리스트와 그 리스트에 작용되는 function 간의 interface 역할을 한다. 가장 간단하게는 다른 node 들의 구조와 동일하게 변수 item과 포인터인 next로 구성할 수 있다.(그림 3.4 참조) 즉, header node의 next가 리스트의 시작 번지를 기억하게 되고 item부분은 사용하지 않는다. item 부분을 사용하지 않음으로써 기억장소를 허비하기는 하지만 각 node가 동일구조로 구성되어 있기 때문에 많은 function들이 유사한 형태로 구성할 수 있어서 편리하다.

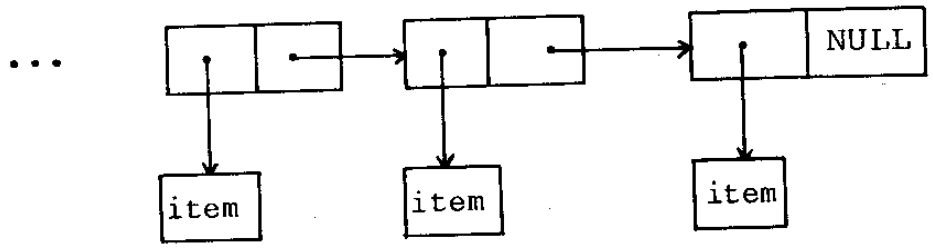


그림 3.3 리스트의 끝부분에 NULL 포인터 지정

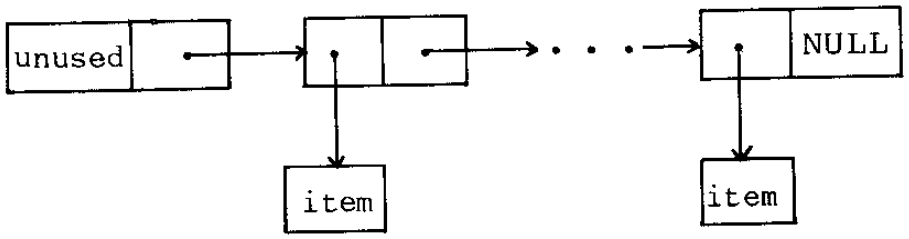


그림 3.4 header와 각 node의 구조가 같은 리스트 [24]

또, header에 필요한 정보를 저장할 수 있도록

```
Struct head
{  int  length ;                    /* current length of list */
   node_t   *first, *last ;         /* pointers to first and  */
                                    /*       last elments     */
 };
```

와 같이 따로 구조체를 구성할 수도 있다. 이경우 기억장소를 많이 소모하지만 리스트내의 node수나 마지막 node에 대한 주소 등과 같은 과외의 정보를 저장할 수 있어 가장 많이 쓰인다. 이를 custom header라고 한다. 이 방법에 대한 구조가 그림 3.5 에 나와 있다.

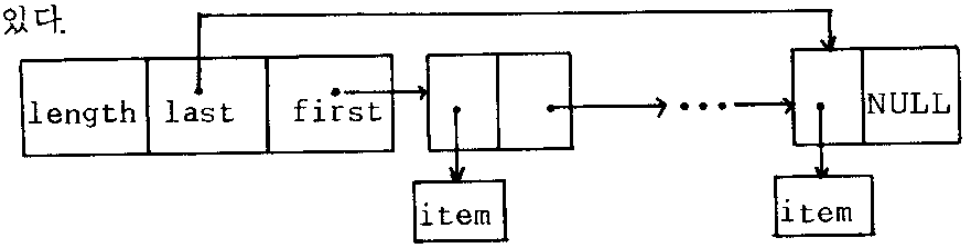


그림 3.5 Custom header를 가진 구조 [24]

기존 리스트에 새로운 리스트를 덧붙이는 경우에는 header node에 대해 기억장소를 할당해주고 그 모든 node에 적절한 값을 지정해 주면 된다. 앞에서 말한 custom header를 사용하고 모든 node를 구조체로 구성하면 새로운 리스트를 덧붙이도록 하는 function을 만들 수 있다. 한편 리스트의 시작이나 끝부분에 새로운 리스트를 덧붙이는 경우에는

(1) 새로운 리스트 node를 저장할 기억장소를 할당

(2) 새로운 node에 적절한 포인터를 지정

(3) header (또는 tail) node의 component에 새로운 값을 지정해 주면 된다.

또한 리스트 구조에서 각 node를 이중으로 결합시켜 양방향 모두 진행시키는 일이 가능하다. 즉 각 node에서 다음 node의 주소를 갖고 있는 포인터 next와 그 전 node의 주소를 담는 포인터를 지정해 줄 수 있다. 그림 3.5과 같은 구조에 새로운 리스트를 덧붙이고 이중 결합이 있는 리스트 구조를 바꾸어 주면 그 구조가 그림 3.6과 같다.

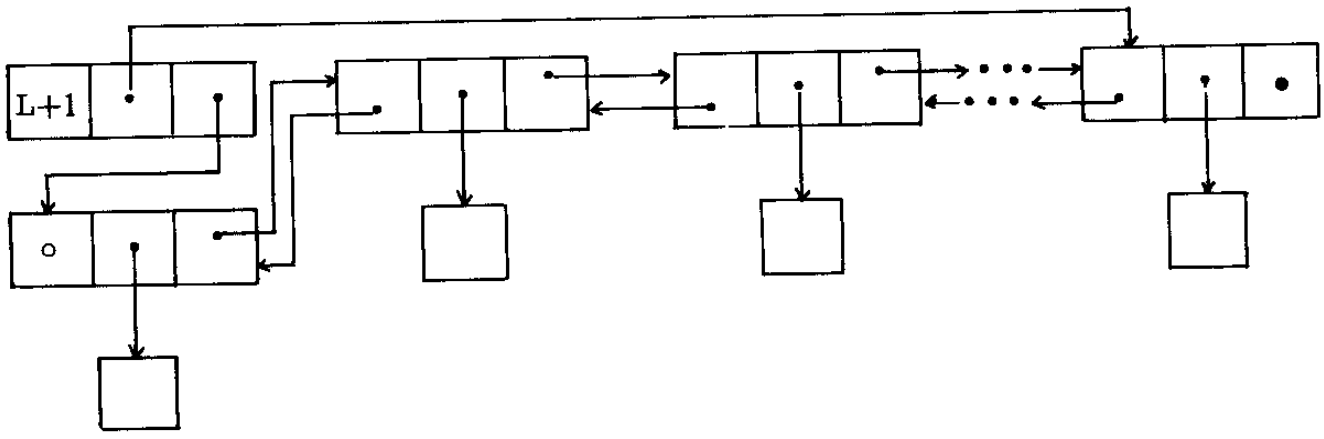


그림 3.6 node 첨가와 이중결합이 있는 리스트 구조[24]

나. Stack

Stack은 리스트의 특별한 형태이다. 일반적인 리스트 구조는 다양한 조작이 가능하지만, Stack은 item의 첨가나 삭제가 한 위치 즉 Stack의 상단에서만 가능하다. 따라서 Stack에 item이 첨가되는 것은 " pushed "되었다고 하고 삭제되는 것은 " popped " 되었다고 한다.

이러한 특성때문에 Stack은 LI FO(Last In First Out) 데이타 구조라고 불리운다. Stack은 널리 사용되는 데이타 구조이다. 일련의 operation이 nest되어 있을때 그 operation들을 역순으로 재추적하는 일이 필요하게 되는데 이때 변수에 값을 할당하고 그 때의 값들을 저장해야 한다. 이를 stack에 item을 push하고 pop함으로써 수행해 낼수 있다.

다. Queues

Queue도 또다른 리스트의 한 형태이다. Queue의 특징은 item의 첨가는 끝에서 하고 제거는 시작 부분에서 한다는 것이다. 이 구조는 각 node를 한쪽 방향으로만 수행해 나가야 할 경우에 사용된다. 각 node의 프로세싱은 FIFO (First in Fiirst Out)순으로 행해진다. Queue는 프로세싱하는 속도보다 빨리 정보를 생성해내고 있어 각 item이 수행 순서를 대기하고 있는 경우 사용된다. 예를 들어 키보드와 mouse를 동시에 사용하여 프로그램이 다룰수 있는 속도를 초과한 경우 입력된 정보를 잃어버리지

않기 위해서 차례대로 데이타를 저장해주는 일이 필요한 때와 같은 경우에 사용된다.

이제까지 다룬 리스트 구조는 주어진 형태의 변수들이 메모리에 직결로 연결되어 있어서 한 node에서 다른 node로 포인터를 따라 수행되어 나가는 것으로 각 node가 구조체로 되어 있다. 그러나 리스트 구조의 각 node는 더욱 복잡한 형태도 가능하여 리스트 각 구성요소가 또 다시 리스트인 구조도 사용된다. 이를 " list of lists "라고 부르는데 그래픽 시스템 등의 응용에 중요한 역할을 한다.

라. 트리 구조

트리는 계층적인 데이타 구조로 널리 사용되고 있는 구조 중의 하나로 반복적인 기능을 잘 수행한다. 트리는 리스트 구조처럼 각 node에 저장된 데이타로 이루어지나 리스트 구조와는 달리 각각의 node가 서로 계층적인 연계를 갖는다. 즉 그 구조가 그림 3.7과 같다.

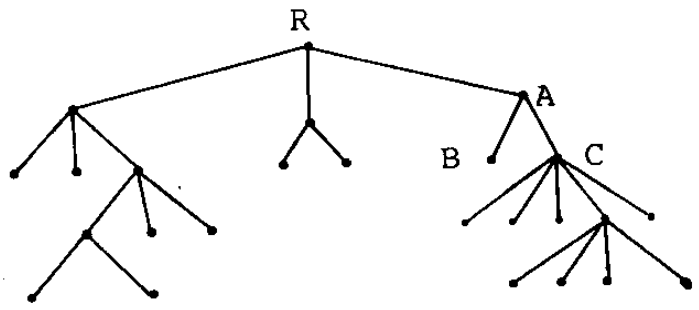


그림 3.7 트리 구조의 예

트리구조는 " root "라고 불리우는 상단의 node와, 더이상 다른 node로 뻗어나가지 않는 " leaf node ", 한개 이상의 새로운 node로 확장되는 일반적인 node등으로 구성된다. 이 트리구조는 가계의 구성에 비교되어 parent, child,ancestor node 등으로 불리우기도 한다. 즉 최상단의 node인 " root "는 모든 node의 ancestor가되고 leaf node는 descendants를 하나도 가지지 않는다. 그림 3.7 에서 node A는 leaf node B와 C의 parent이고 B와 C는 형제가 된다.

리스트 구조에서 node의 순서는 간단하여 리스트의 header 포인터가 첫번째 node가 되고 계속해서 다음 node가 이어지게 되지만 트리구조의 경우에는 각 node가 한개 이상의 형제와 child node를 가질 수 있으며 한 방향으로만 뻗어나가는 것이 아니므로 복잡한 형태를 띠게 된다. node의 순서를 이용하여 경로를 잘 정의하여야 각 node를 한번씩만 경유하면서 트리의 모든 node를 조사할 수 있게된다.

트리구조 node의 순서를 정하는 방법으로는 pre-order, in-order,post-order가 있다.

* Pre-Order

이 알고리듬은 하나의 root와 좌→우로 이어지는 부속 tree 순으로 순서를 정한다. (그림 3.8 참조)

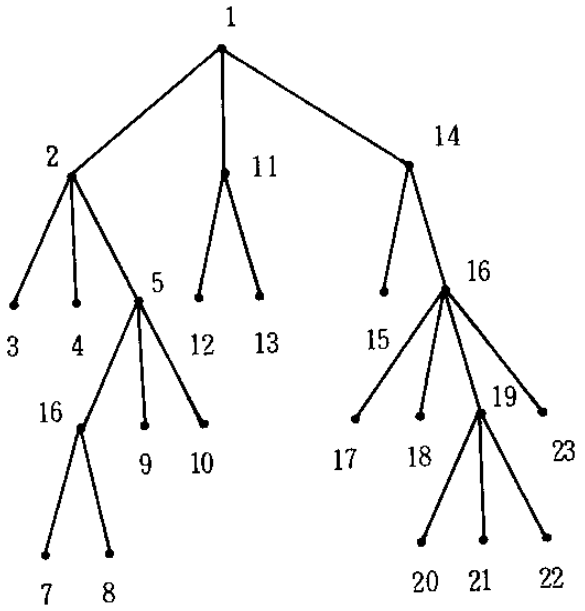


그림 3.8 pre order listing[24)]

* In-Order

Pre-order 시스템에서는 root로부터 leaf node까지 아래로 순서를 정해 나가는 반면 In-order 시스템에서는 가장 왼쪽에 있는 child부속 tree를 정한뒤 그의 root를 listing 하고 좌→우의 순으로 나머지 부속 tree의 순서를 정해 나간다.

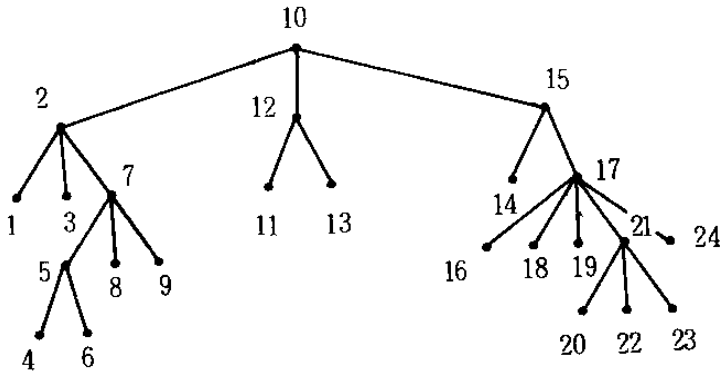


그림 3.9. In-orden listing[24)]

* Post-Order

Post order 리스팅은 모든 부속 트리들을 정한 뒤 root의 node를 정한다는 점에서 앞의 두 알고리듬과 다르다. 즉 각 부속 트리를 좌→우 순으로 정하고 root의 순서를 정해준다. (그림 3.10 참조)

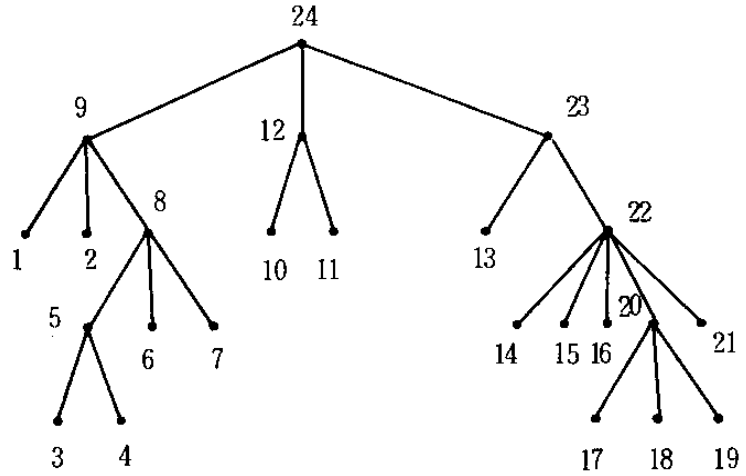


그림 3.10 post order listng[24]

한편 그 수행에 있어서는 리스트 구조에서처럼 트리구조도 반복적인 구조를 갖도록 짜여져야 하며 리스트구조와 달리 부속트리로 확장되어 나가므로 각 node에서 가질수 있는 chnild node 수를 한정해 주거나 변수로 정해 주는 두가지 방법이 있을 수 있다.

우선 child node수가 고정되어 있는 경우 중 가장 간단한 구조인 binary 트리구조를 보자. (그림 3.11) 여기서 각 node는 0, 1, 또는 2개의 child node를 가질 수 있다.

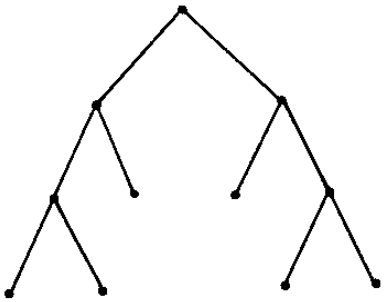


그림 3.11 Binary 트리구조

각 Child node는 좌(left), 우(right) node로도 불리우며 이 binary트리 구조에 대한 data 구조는 다음과 같다.

```
Struct bin_node
  {
    hold_t   * element ;
    Struct bin_node  * left ;
    Struct bin_node  * right ;
    Struct bin_node  * parent
  }
typedef struct bin_node bin_t
```

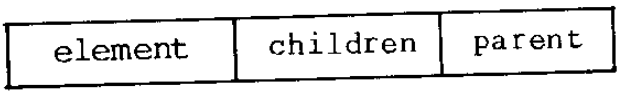
여기서 데이타 형태 hold_t는 node에 저장하고자 하는 정보를 수용할, 미리 정의된 구조체이다. 즉 element는 그 node의 데이타를 저장하고 left와 right에는 각각 binary 좌, 우 node의 포인터가, parent에는 " backing up "시에 쓸 parent node에 대한 포인터가 들어있다. 이번에는 컴파일 당시에는 Child

node를 예측할 수 없는 일반적 목적의 tree를 다루는 구조에 대해서 살펴보자. child node를 배열(array)을 사용하는 대신 리스트 구조를 사용함으로써 child node수를 어떤 크기로도 증가시킬 수 있으며 동시에 불필요한 컴퓨터 메모리를 할당하지 않아도 되는 장점을 갖는다. 이때 child의 각 node는 또 다른 일반적 트리 구조로의 포인터 값을 갖게된다.

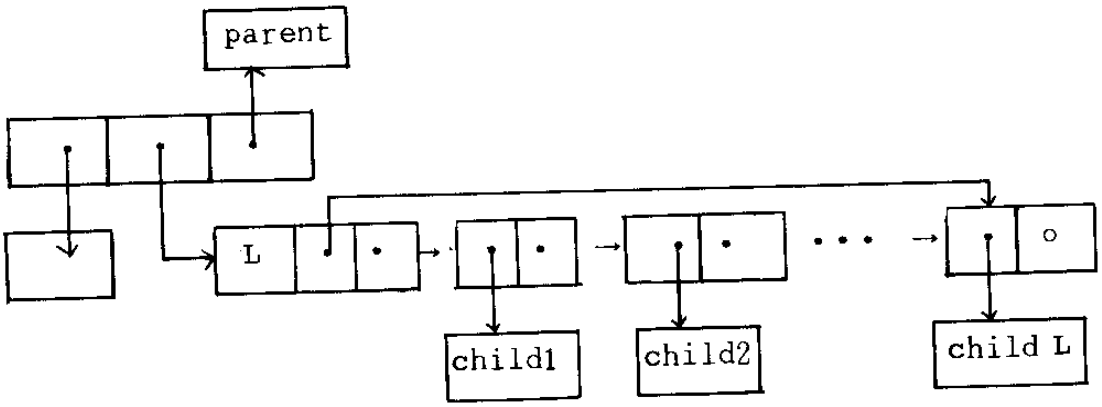
임의의 수의 child node를 갖는 트리구조는 다음과 같이 정의될 수 있다.

```
Struct info
  {
    hold_t          * element  ;
    struct head     * children ;
    struct info     * parent ;
  } ;
typedef struct into info_t ;
```

이 경우 트리의 각 node는 (info_t에 해당) 세개의 포인터로 이루어진다. (그림 3.12 참조) 첫번째 포인터 element는 그 node의 데이타 정보를 가리킨다. 두번째 포인터인 children은 그 node에 연이어져 있는 child node의 리스트를 가리키는데 그 리스트는 head에 각자 또 다른 트리 node로의 포인터를 갖고 있는 node의 리스트가 이어지게 된다. 마지막 포인터는 그 parent node에 대한 포인터로 parent 또한 트리 node가 된다.



(a) 트리의 node 구조



(b) 트리 node와 포인터

그림 3.12 일반적 트리의 구조

마지막 포인터는 그 parent node에 대한 포인터로 parent 또한 트리 node가 된다. 또한 이 트리들을 조합하여 좀 더 복잡한 형태의 데이타 구조를 만들 수 있는데 이는 " forest "라고 불리우기도 한다. 이 forest를 형성하는 가장 간단한 방법은 트리를 배열(array)화하는 것으로 예를 들면

```
# define MAX 5 ← forest 내의 트리의 상한갯수
info_t * node [MAX] ← 트리로의 포인터 array
```

와 같이 정의 할 수 있다.

2. 데이타 베이스시스템의 구성

본 연구에서는 화학 공정에 필요한 데이타 베이스의 구성을 위해 트리구조에 기초하여 데이타 베이스 구조를 확립하고 그 메니지먼트 시스템으로 간단한 inference engine을 만들어 수행시켜 보았으며 이로부터 만들어진 Rule들을 모아 Rulebase를 구성하여 보았다.

가. 데이타 구조

본 연구에서는 앞에서 언급한 트리구조에 기초하여 데이타 베이스의 구조를 완성했다. 트리의 각 node에는 그 node의 데이타에 해당하는 변수 attribute가 구조체로 짜여져 있다. 구조체 attribute는 다음과 같다.

```
Struct attribute {
  char attrib [40] ;
  char parent [40] ;
  Struct attribute  * next[MAX];
} at ;
```

즉 길이 40의 문자 변수 attrib와 그 node의 parent node의 attrib를 저장하는 변수 panert가 같은 길이로 들어있고, 똑같은 구조를 갖는 다음 node의 포인터가 배열로 잡혀 있어서 여러개의 branch가 가능하다. 여기서 branch node의 maxi-

num 숫자는 프로그램문 처음에 5로 정의되어 있으나 언제나 수정가능하고, 각 node는 0～MAX사이 갯수의 branch node를 가질 수 있다.

본 연구에서 개발한 데이타 베이스 구조의 확립과 운용을 그림으로 보면 그림 3.13과 같다.

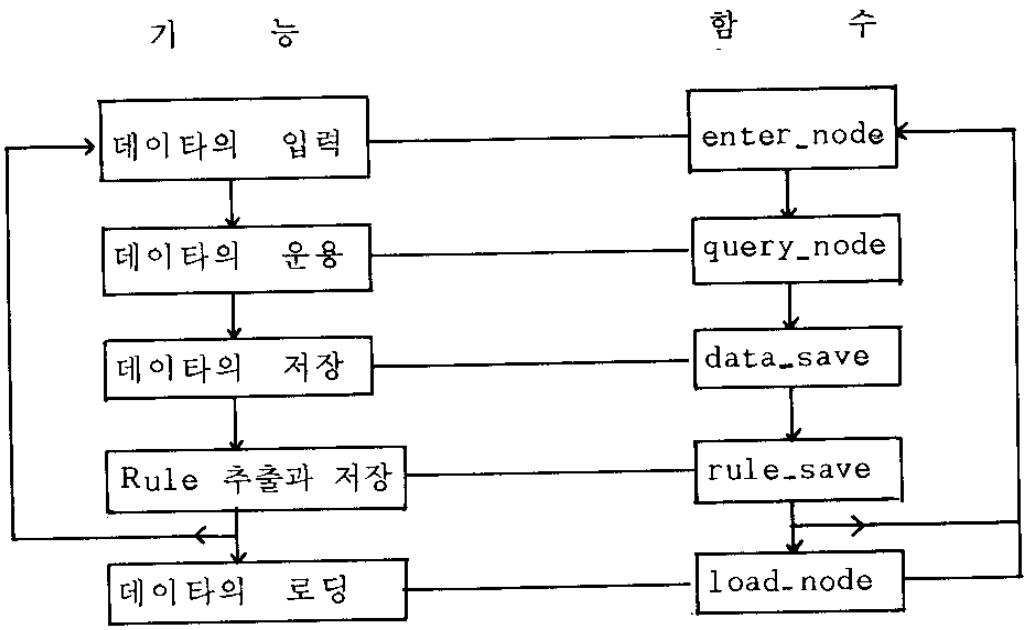


그림 3.13 데이타 베이스 시스템의 구성과 기능

각 루틴에 대해서 살펴본다.

(1) Enter_node( )

데이타 베이스에 저장하려는 데이타들을 읽어 들여 변수에 저장하는 기능을 한다. 우선 메모리에 function "malloc"를 사용하여 필요한 기억 장소를 할당한다. node의 순서는 한 level에서 왼쪽 node→ 오른쪽 node로 진행하면서 정해주도록 되어 있

다.

각 node 안의 문자 변수 attrib 에는 공정의 state 를 넣어줄 수도 있고 각 unit 를 저장해 줄 수도 있다.

Top node 의 attrib 를 읽어 들인 뒤 tree( ) 루틴을 사용하여 level 1 에 해당하는 구조체 attribute 의 내용

1. 해당 node 의 attrib

2. parent node 의 attriib

3. next node 의 구조체 배열 을 읽어들인다.

이를 level 2,3 으로 증가시켜 가면서 반복적으로 사용하여 트리구조의 데이타 구조를 확립한다. Enter_node 의 구성을 도식화 하면 그림 3.14 과 같다.

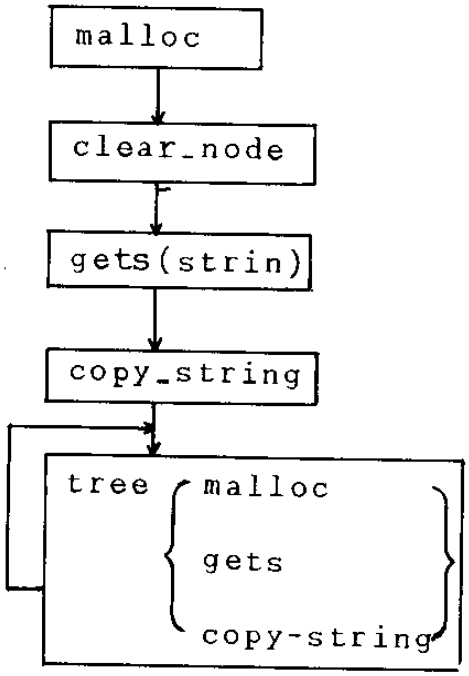


그림 3.14 enter_node ( ) 의 구성

(2) query_node( )

읽어들인 데이타가 각 변수에 할당되고 난 뒤 그 정보를 운용하는 inference engine에 해당하는 부분이다. breadth first search에 의해서 해를 찾아 나가도록 구성되어 있다. 우선 level 1의 각 node를 왼쪽에서 오른쪽으로 가면서 의문문으로 표현하여 hey board를 통해 " yes " " no "를 입력하도록 되어있다. " yes "로 선택된 node를 parent로 하는 구조체 배열에 대해 위와 같은 작업을 반복 leaf node가 나오면 그를 solution으로 선택한다. query_node( )의 구조는 그림 3.15과 같다.

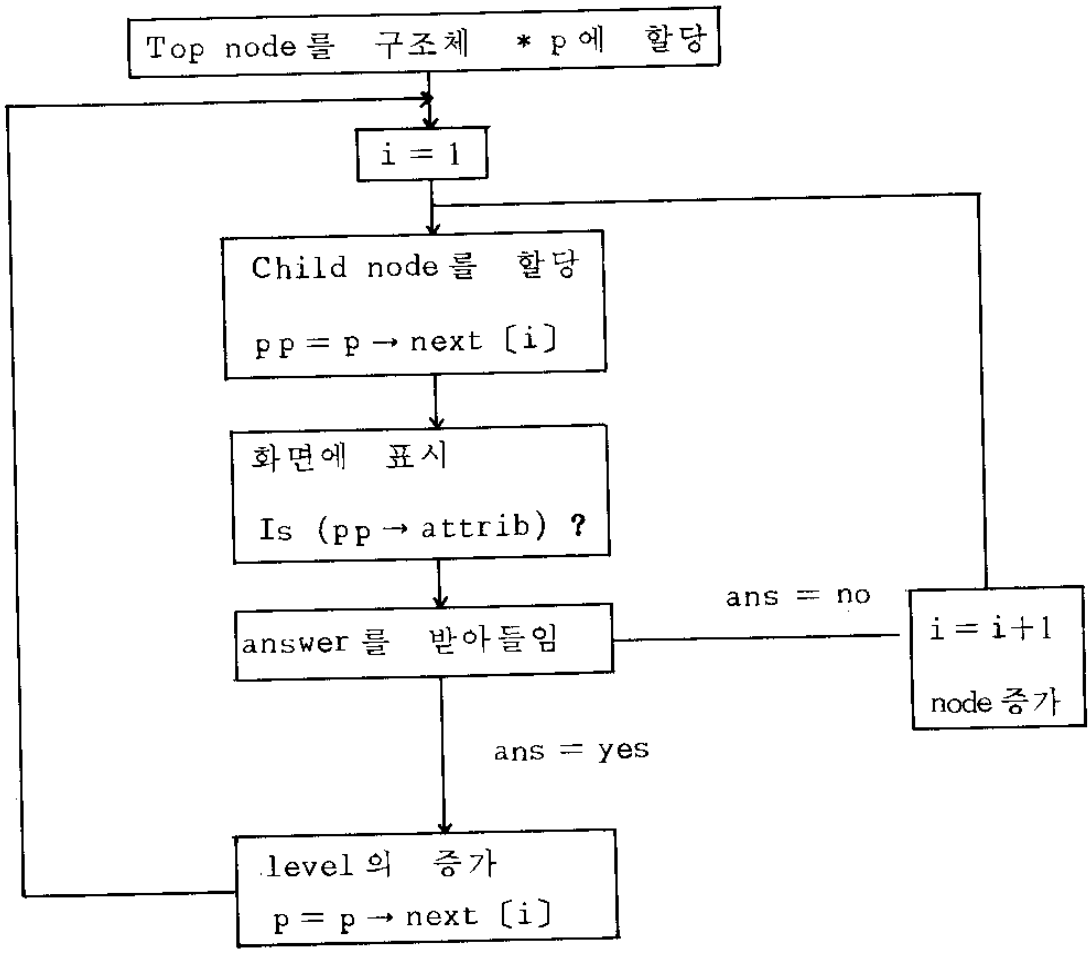


그림 3.15 query_node의 구조

(3) data _ save( )

Enter_node( )에 의해 입력되어 각 변수에 지정되어 있는 데이타 들을 화일에 저장하는 루틴이다. 다시 loading하여 사용할 때 정확히 변수에 재 할당해 주기 위해서 enter_node 에서 입력한 순서 그대로 저장해 준다. 조정하는 루틴 load_node 에서도 마찬가지이다. 문자 ' n '을 넣어 한 attrib가 끝났음을 명시했고 빈칸 한줄은 parent node가 바뀜을 나타낸다. data_save 수행 결과 " expert. dat "라는 화일이 생긴다.

(4) rule_save( )

query _ node( )의 수행 결과 생기는 rule를 데이타 화일에 저장한다. 즉 IF, AND, THEN을 사용하여 query_node를 수행하면서 각각 parent로 선택되었던 attrib와 마지막으로 얻은 해인 attrib를 나타낸다. 즉 level 1에서는 attrib 1을 택하고 attrib 1의 child중에서 attrib 2를 택하고 그 attriib 2의 child로 진행해서 선택한 attrib 3이 leaf node가 되어 query_node( )가 끝났다면 여기서 얻는 rule은

IF (attribl) AND (attrib2)

THEN (attrib 3)

과 같다.

이렇게 얻어진 rule은 " kbase. dat "라는 화일에 저장되어 다음에 같은 search가 행해질 경우 직접 읽어서 해를 내보

낸다.

(5) Load_node( )

데이타 화일에 저장된 정보를 읽어 들여 해당 변수에 저장하는 루틴이다. enter_node와 유사한 구조를 갖는다.

3. 사례 연구

앞에서 설명한 데이타 베이스와 운용 시스템의 실행을 간단한 공정진단에 적용하여 보았다.

대상계는 그림 3.16과 같은 Flash 증류 공정이다.

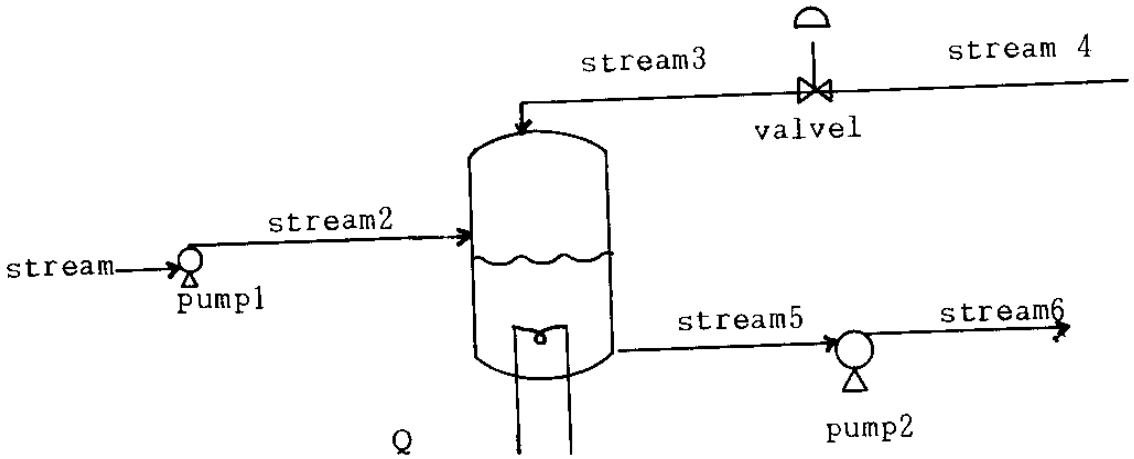


그림 3.16 Flash 증류 공정

이 경우 Tank level에 대한 진단은 다음과 같다.

level이 낮은 경우

(1) stream 2가 low이고 stream 5가 normal이다.

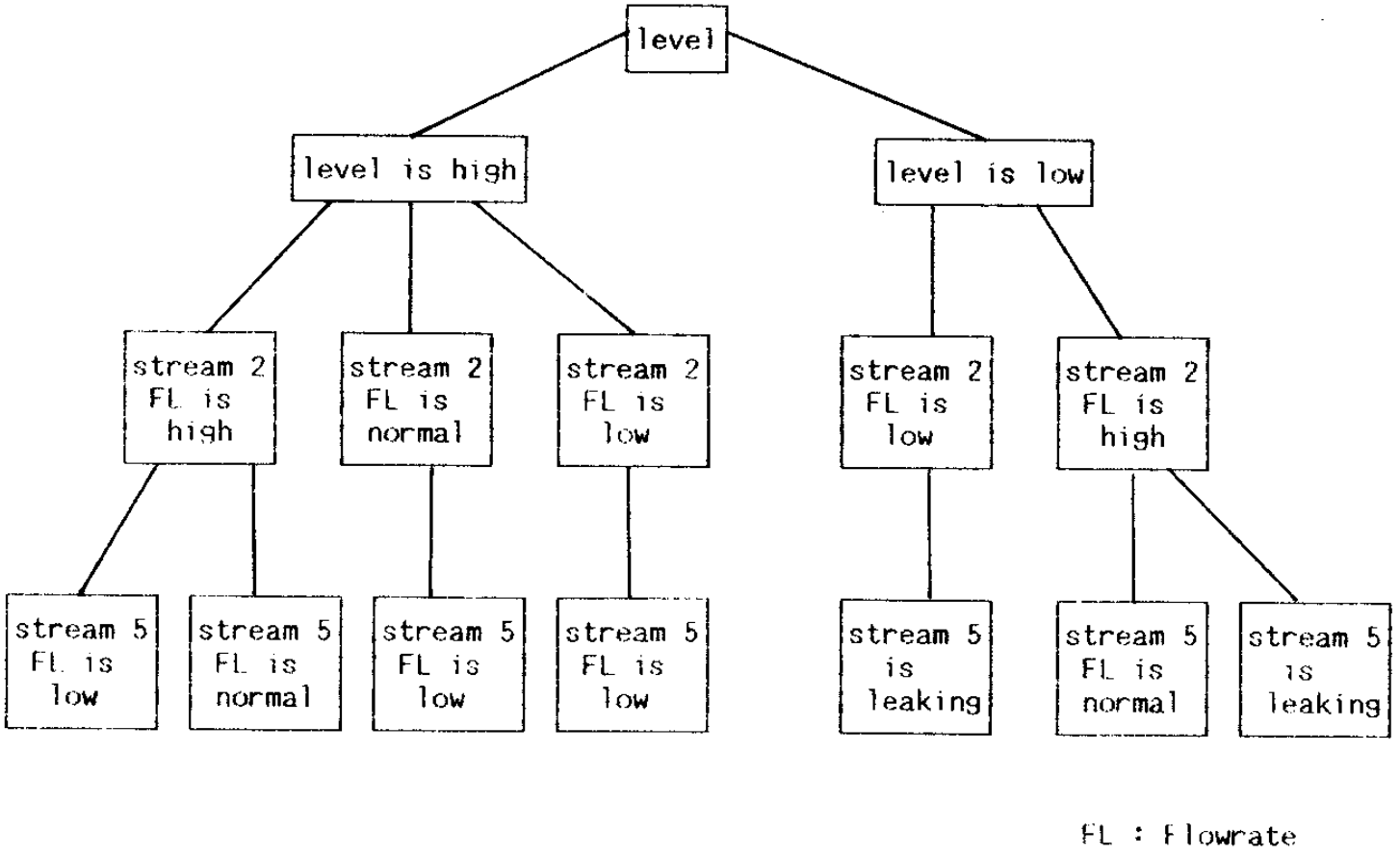


그림 3.17. tank level 진관에 대한 레이타 베이스

(2) stream 2가 low이고 tank의 stream 5의 연결 부위에 leaking이 있다.

(3) stream 2가 mormal이고 tank의 stream 5의 연결 부위에서 leaking이 있다.

level의 높은 경우;

(1) stream 2가 high이고, stream 5가 normal이다.

(2) stream 2가 high이고, stream 5가 low이다.

(3) stream 2가 normal이고, stream 5가 low이다.

(4) stream 2가 low이고, stream 5가 low이다.

이를 데이타 베이스 구조로 나타내면 그림 3.17과 같다.

본 연구에서 개발한 데이타 베이스 시스템을 수행하면 우선 그림 3.18과 같은 메뉴가 화면에 나타난다.

```
E : enter new data
S : save database
Q : query
L : load database
X : exit

Choose your option:
```

그림 3.18 메뉴 선택 화면

여기서 E는 enter_node 루틴을 불러 데이타 베이스를 받아들여 변수에 저장하며 S는 save_data를 사용 데이타 화일에

저장, Q는 query 루틴을 사용 원하는 solution을 찾아 주며 L은 load_node를 사용 데이타 화일의 내용을 메모리에 로딩시켜준다. X를 선택함으로써 프로그램 수행을 마치게 된다.

우선 첫단계로 데이타 입력을 위해 " E "를 선택하여 앞에서 얻는 데이타 구조를 입력하는 예가 그림 3.19에 나와 있다. 입력하려는 데이타의 parent node가 화면에 표시되므로 그 attrib를 갖는 node의 child node를 모두 넣어주면 된다. 즉 그림 3.17에서 " level is high "의 child node가 3개 있으므로 화면에 " enter level 2 elements " 라는 명령에 이 3개의 child node를 입력해 주고 입력이 끝났음을 나타내기 위해 enter key를 눌러준다. Save_node의 수행으로 입력된 데이타를 저장하기 위해 " S "를 선택한다. 다음으로 query_node의 실행 두가지 예가 그림 3.20의 (1),(2)에 나와 있다. 그림 3.17에 나와 있는 구조를 입력 순서대로 화면에 표시하면서 묻게 된다. 그림 3.21 (a)의 경우 level은 low이고 stream 2의 Flowrate가 low라면 solution은 stream 5에 leaking이 있다는 solution에 도달하게 된다. query 과정의 answer와 결과를 I F, THEN문으로 이루어 화면에 표시하고 이를 rule base "kbase.dat "에 저장한다.

```
    Choose your option:e

enter top element  (return to quit)
 : level

enter level 1 elements  (return to quit)
level
 : level is high
 : level is low
 :
enter level 2 elements  (return to quit)
level is high
 : stream 2 flowrate is high
 : stream 2 flowrate is normal
 : stream 2 flowrate is low
 :
level is low
 : stream 2 flowrate is low
 : stream 2 flowrate is high
 :
enter level 3 elements  (return to quit)
stream 2 flowrate is high
 : stream 5 flowrate is low
 : stream 5 flowrate is normal
 :
stream 2 flowrate is normal
 : stream 5 flowrate is low
 :
stream 2 flowrate is low
 : stream 5 flowrate is low
 :
```

그림 3.19. enter-node의 실행 - 데이타베이스 입력 예

```
***  level  ***

  level is high  ?  n
  level is low  ?  y
  stream 2 flowrate is low  ?  y
  ----------> stream 5 is leaking

  IF (level is low AND stream 2 flowrate is low )
        THEN (stream 5 is leaking )


***  level  ***

  level is high  ?  y
  stream 2 flowrate is high  ?  n
  stream 2 flowrate is normal  ?  y
  ----------> stream 5 flowrate is low

  IF (level is high AND stream 2 flowrate is normal)
        THEN (stream 5 flowrate is low )
```

그림 3.20. query_node의 수행 예

제 3 절 공정지식의 표현 및 운용

1. 공정지식 표현의 list구조

공정 지식의 표현에 있어서 linked list구조는 특히 경험적 지식의 표현에 매우 능률적이다. Linked list 구조는 대부분의 공정에 있어서 컴퓨터의 기억 용량이 허용하는 범위 까지는 계속 공정지식을 입력시키고 저장시킬 수 있으므로 단위 공정의 표현에 매우 적합하다. 그 size가 큰 지식은 다음에 보인 list 구조에서 사용자가 최초 입력시 지정해 준 size로 나누어서 처

리 할 수 있다.

```
struct record-list {char *data;
                    struct record-list *prev;
                    struct record-list *next;};
```

위의 list 구조에서 data는 character에 대한 pointer로서 실제의 공정 지식들을 나타내 준다. data의 지식은 하나 또는 다수의 항목들로 구성되는 format으로 표현된다. 하나의 data내에 최대로 포함될 수 있는 항목의 수를 MAX-ITEMS개라고 하면 data의 format은 다음과 같은 형식으로 이루어 진다.

```
struct item-descriptor {int item-position;
                        char item_title[MAX_NAME];
                        int item_size;
                        int x,y;} items[MAX-ITEMS];
```

item-position은 data내에서 해당되는 item이 차지하게 될 위치를 나타내어 주는 수로서 사용자가 정하여 주는 item의 size(byte수)에 따라 자동적으로 결정된다. item_title은 스크린에 제시되는 item의 명칭으로서 대상 공정이나 장치에 따라 사용자가 결정해 주어야 한다. x와 y는 해당되는 item이 스크린 상에서 차지하는 가로 및 세로좌표를 의미하는데 역시 사용자가 임의로 정해줄 수 있다. 공정지식의 linked list구조에 의한 표현 및 처리과정은 다음 3-21에 보인 바와 같다.

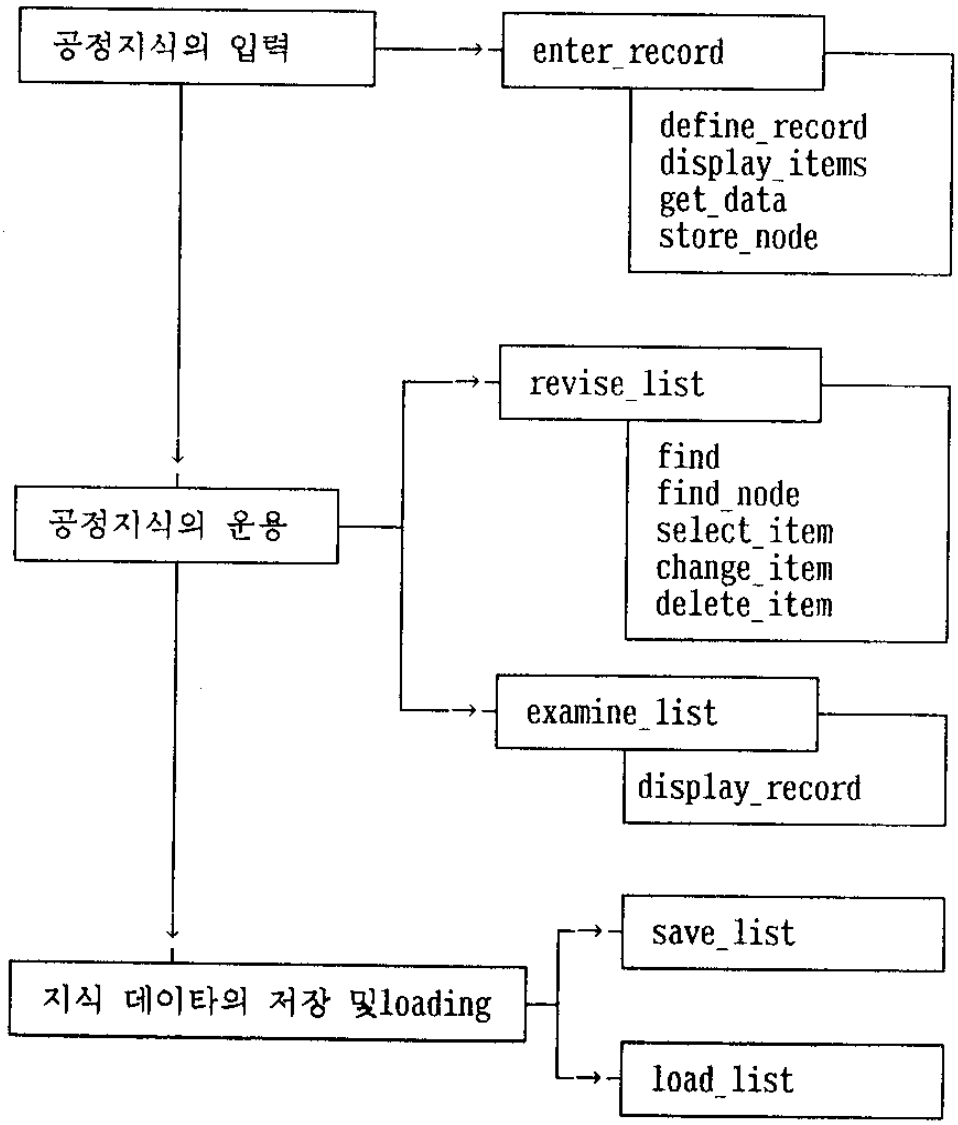


그림 3-21. 공정지식의 list식 표현 및 처리단계

가. 지식표현구조의 설정

define-record function은 record-list구조의 data에 저장되는 공정지식의 항목을 결정하여 준다. 즉 대상 공정이나 장치의 특성 및 표현되는 지식의 성질에 따라 사용자가 임의로 data item의 title과 size등을 정해줄 수 있다. define-

record의 데이타 item 설정과정은 다음의 그림 3-22 와 같이 요약해 볼 수 있다.

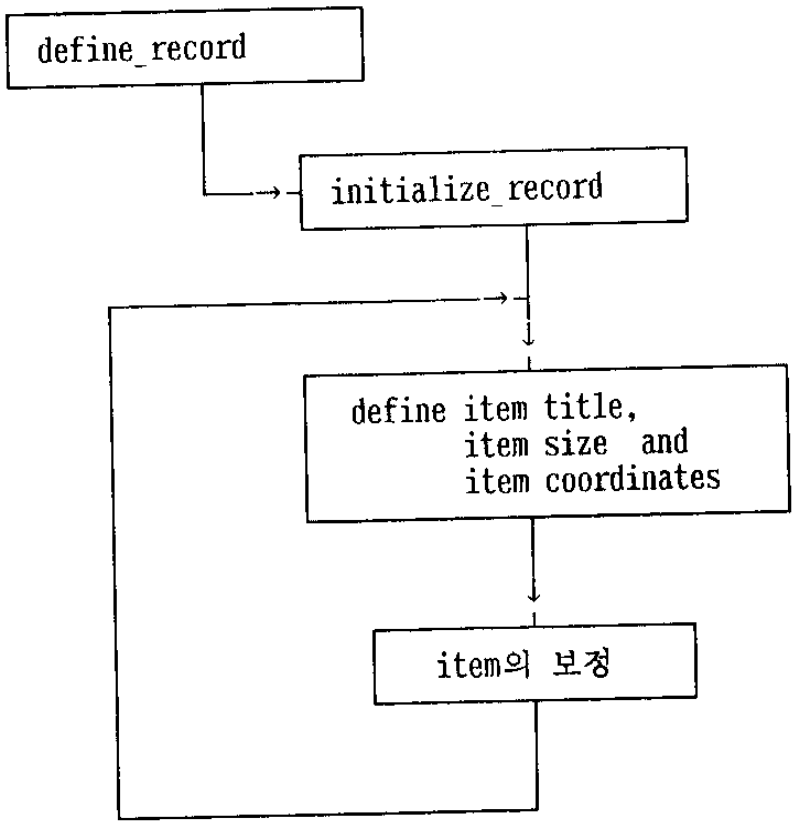


그림 3-22. 공정 지식 데이타의 format

initialize-record routine은 공정 지식 데이타 베이스에 관련된 각 변수의 초기값들을 설정해 준다. 이 routine에서는 공정지식 item의 수 외에도 제 4 장의 전문가 시스템에서 이용되고 있는 각종 변수와 list 구조에 대한 pointer 변수들의 초기화도 이루어진다. 다음의 표 3-1 은 define-record 함수에서의 각 보조 routine들의 기능을 요약한 것이다.

표 3-1. define-record 보조 routine의 기능

function	기 능
initialize_record get_data display_item select_item	변수들의 초기화 title의 입력 설정된 item들의 제시 보정할 item의 선정

나. 공정지식의 입력

공정지식의 입력은 enter_record function에 의하여 이루어진다. 앞의 define_record routine에 의하여 설정된 데이타 item format에 따라 사용자는 차례대로 데이타를 입력시키게

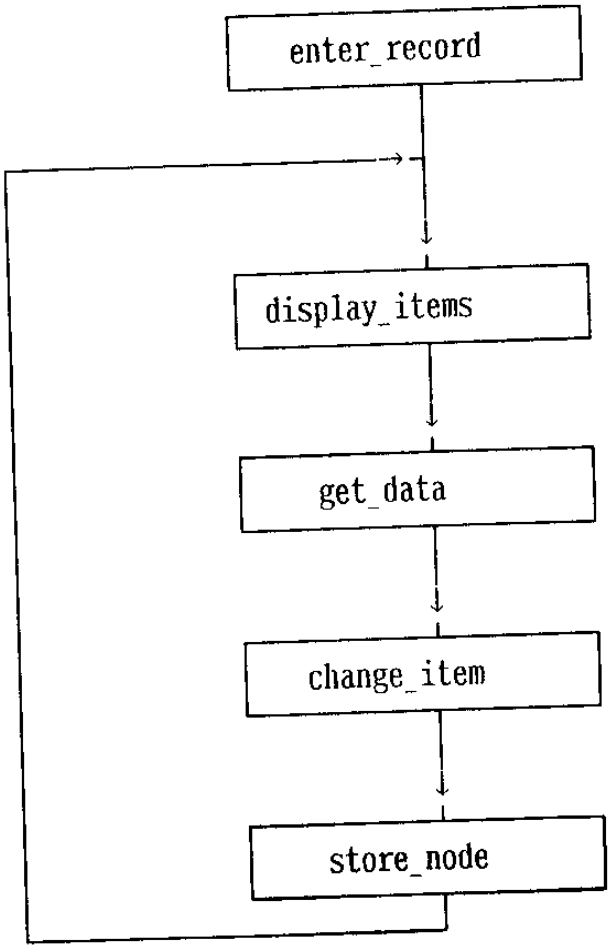


그림 2-23. enter-record routine의 기능

된다. 입력되는 데이타는 store_node function에 의해 linked list구조로 변환된다. 그림 3-23은 enter_record function의 기능을 도시한 것이며 표3-2는 각 보조함수들의 기능을 요약한 것이다.

표3-2. enter_record 보조 routine의 기능

function	기 능
display_item	설정된 item들의 제시
get_data	공정지식의 입력
select_item	보정할 공정지식 item의 선정
store_node	linked list 구조로의 변환

다. 공정지식의 보정

입력되는 공정지식은 필요에 따라 자유로이 보정시켜줄 수 있어야 한다. 잘못 입력된 데이타의 보정은 물론이고 저장되어 있는 공정 데이타의 revision도 자유로와야 할 것이다. 이러한 기능은 revise_list function에 의하여 이루어진다. 그림 3-24는 revise_list function의 기능을 도시한 것이며 표3-3은 각 보조함수들의 기능을 요약한 것이다.

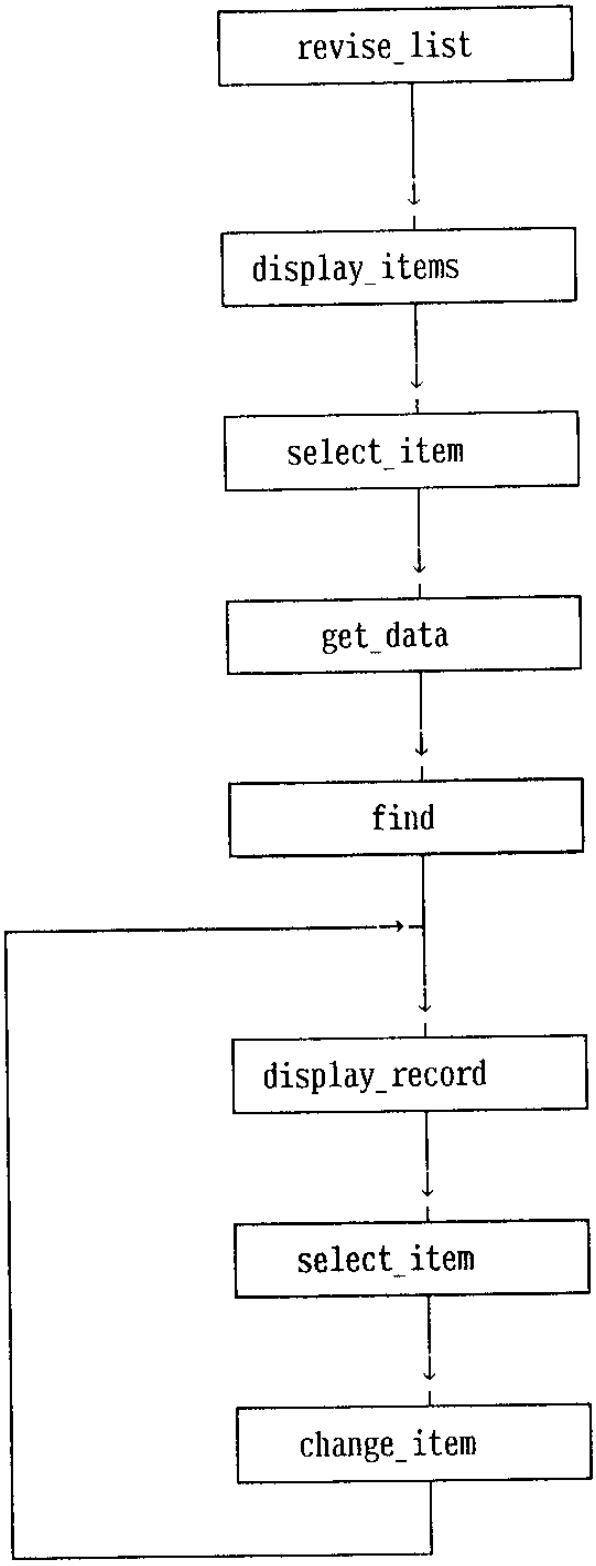


그림 3-24. revise_list routine의 기능

표 3-3. revise-list보조 routine의 기능

function	기 능
display_item	설정된 item들의 제시
select_item	보정할 공정지식 item의 선정
get_data	찾고자 하는 지식 item title의 입력
find	보정될 list의 제시
display_record	보정할 지식 데이타의 제시
change_item	보정된 공정지식의 입력

change_item routine은 다음의 그림 3-25 에서와 같은 과정에 의해 공정지식 데이타를 보정시켜 준다.

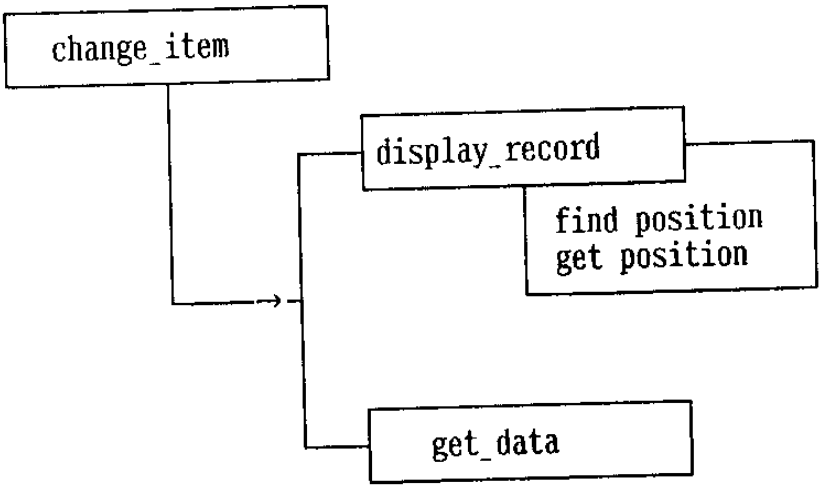


그림 3-25. change_item routine의 기능

라. 공정지식 운용 보조함수

위에서 언급한 함수들 이외에 공정지식 데이타의 운용에 중요한 역할을 하는 함수들을 보조함수라고 칭하였는데 이들은 공정지식 데이타를 컴퓨터 스크린 상에 제시하여 주고 컴퓨터의 하드

디스크나 floppy 디스크 상에 공정지식 데이타를 저장해 주며 아울러 저장된 지식의 loading작업도 맡아서 행하는 매우 중요한 역할을 담당하고 있다. Cursor의 유동이나 스크린 또는 line의 조작기능들은 이미 잘 알려져 있는 function들이며 시중의 컴퓨터 language manual들로부터 쉬게 찾아볼 수 있다. 이들 함수들을 화학공정지식의 표현 및 운용에 편리한 함수로 보정시켜서 시스템에 응용할 수 있다. 다음의 표 3-4는 현재 이용되고 있는 보조함수들의 기능을 요약한 것이다.

표 3-4. 보조함수의 기능

function	기 능
examine_list	list 데이타구조의 검증
delete_item	주어진 item의 삭제
comment1, comment2	command의 제시
goto_xy	지정된 위치로의 cursor의 이동
clear_line	지정된 line의 삭제
clear_screen	스크린의 삭제

2. 공정 데이타의 표현

화학공정에 있어서는 공정과 관련된 물성치, 조업조건, 또는 공정 유통과 관련된 자료들의 효과적인 표현과 운용기능이 매우 중요하다. 대표적인 보기로서 석유화학공정을 들 수 있는데 석유화학공정은 비교적 소수의 원료 물질에서 출발하여 점차적으로 화학물질의 복잡한 network로 팽창되어 나가며 종국적으로는 소비자가 사용하는 제품으로 귀착된다. 이러한 과정에서 각 공정 및 그와 관

련된 지식의 효율적인 표현은 전체적인 플랜트의 경제적이고 능률적인 조업에 필수적이다. 공정자료 표현의 가장 간단한 보기로서 다음과 같은 몇가지 화학물질의 제조에 관련되는 자료들을 공정지식 데이타베이스에 입력시켜 보기로 한다.

Synthesis Gas의 제조:

$CH_4 + H_2O \rightarrow 3H_2 + CO$

조업온도범위 : 830 — 850 ℃

압력범위 : 400 - 500 psig

사용되는 촉매 : Promoted nickel based

메탄올의 제조:

$2H_2 + CO \rightarrow CH_3OH$

온도범위 : 250 — 260 ℃

압력범위 : 725 — 1176 psig

Formaldehyde의 제조:

$CH_3OH \rightarrow HCOH + H_2$

온도범위 : 400 — 425 ℃

압력범위 : 대기압

사용되는 촉매 : $Fe(Mo\ O_4)_3$

수율 : 98 %

초산의 제조:

$CH_3OH + CO \rightarrow CH_3COOH$

온도범위 : 200 ℃

압력범위 : 215 psig

사용촉매 : Rhodium promoted by iodine

수율 : 99 %

Methyl Chloride의 제조 :

온도범위 : 340 - 350 ℃

압력범위 : 대기압

사용촉매 : Ignited alumina gel, chloride on pumice, etc.

수율 : 95 %

Methylamines의 제조 :

$CH_3OH + NH_3 \rightarrow CH_3NH_2 + H_2O$

$CH_3OH + CH_3NH_2 \rightarrow (CH_3)_2NH + H_2O$

$CH_3OH + (CH_3)_2NH \rightarrow (CH_3)_3N + H_2O$

온도범위 : 380 - 450 ℃

압력범위 : 200 psi

사용촉매 : Alumina - gel

수율 : 95 % (on methane)

위와 같은 공정 데이타의 입력은 다음에 보인 바와 같다. 반응식의 입력은 생략하였는데 반응식 item을 데이타 베이스 구조에 삽입시킴으로써 반응식의 입력도 용이하게 할 수 있다.

```
CHEMICALS : Synthesis Gas

Temperature : 830 - 850 C
Pressure : 400 - 500 psig
Catalyst : Promoted nickel based
Yield :



Is entry OK? (Y/N):
```

```
CHEMICALS : Methanol

Temperature : 250 - 260 C
Pressure : 725 - 1176 psig
Catalyst : Cu + Small amounts of Zn + a component
Yield :



Is entry OK? (Y/N):
```

```
CHEMICALS : Formaldehyde

Temperature : 400 - 425 C
Pressure : Atmospheric
Catalyst : Fe(MoO4)3
Yield : 98%



Is entry OK? (Y/N):
```

온도범위 : 200 ℃

압력범위 : 215 psig

사용촉매 : Rhodium promoted by iodine

수율 : 99 %

Methyl Chloride의 제조 :

온도범위 : 340 - 350 ℃

압력범위 : 대기압

사용촉매 : Ignited alumina gel, chloride on pumice, etc.

수율 : 95 %

Methylamines의 제조 :

$CH_3OH + NH_3 \rightarrow CH_3NH_2 + H_2O$

$CH_3OH + CH_3NH_2 \rightarrow (CH_3)_2NH + H_2O$

$CH_3OH + (CH_3)_2NH \rightarrow (CH_3)_3N + H_2O$

온도범위 : 380 - 450 ℃

압력범위 : 200 psi

사용촉매 : Alumina - gel

수율 : 95 % (on methane)

위와 같은 공정 데이타의 입력은 다음에 보인 바와 같다. 반응식의 입력은 생략하였는데 반응식 item을 데이타 베이스 구조에 삽입시킴으로써 반응식의 입력도 용이하게 할 수 있다.

```
CHEMICALS : Synthesis Gas

Temperature : 830 - 850 C
Pressure : 400 - 500 psig
Catalyst : Promoted nickel based
Yield :



Is entry OK? (Y/N):
```

```
CHEMICALS : Methanol

Temperature : 250 - 260 C
Pressure : 725 - 1176 psig
Catalyst : Cu + Small amounts of Zn + a component
Yield :



Is entry OK? (Y/N):
```

```
CHEMICALS : Formaldehyde

Temperature : 400 - 425 C
Pressure : Atmospheric
Catalyst : Fe(MoO4)3
Yield : 98%



Is entry OK? (Y/N):
```

```
CHEMICALS : Acetic Acid

Temperature : 200 C
Pressure : 215 psig
Catalyst : Rhodium promoted by iodine
Yield : 99%



Is entry OK? (Y/N):
```

```
CHEMICALS : Methyl Chloride

Temperature : 340 - 350 C
Pressure : Substantially atmospheric
Catalyst : Ignited alumina gel, zinc chloride on pumice
Yield : 95%



Is entry OK? (Y/N):
```

```
CHEMICALS : Methylamines

Temperature : 380 - 450 C
Pressure : 200 psi
Catalyst : Alumina-gel
Yield : 95% (on methane)



Is entry OK? (Y/N):
```

이상과 같이 입력된 공정관련 데이타의 운용에 있어서는 데이타의 자유로운 검증, 잘못된 데이타의 삭제, 새로운 데이타의 삽입, 기존 데이타의 보정 등이 용이하게 이루어질 수 있어야 한다. 이러한 기능들은 데이타 베이스 구축 및 운용 routine에 이미 잘 알려져 있는 컴퓨터 key 운용기능들을 추가하고 필요에 따라 원하는 기능들을 수행할 수 있는 함수들을 프로그램하여 둠으로써 가능해 진다. 보기로서 위에서 입력된 데이타의 검증기능 수행과정을 보이면 다음과 같다.

```
CHEMICALS : Acetic Acid

Temperature : 200 C
Pressure : 215 psig
Catalyst : Rhodium promoted by iodine
Yield : 99%




Down arrow: forward      Up arrow: backward      Home: beginning
End: end                 Return: to main menu
```

CHEMICALS : Formaldehyde

Temperature : 400 - 425 C
Pressure : Atmospheric
Catalyst : Fe(MoO4)3
Yield : 98%

Down arrow: forward Up arrow: backward Home: beginning
End: end Return: to main menu

CHEMICALS : Methanol

Temperature : 250 - 260 C
Pressure : 725 - 1176 psig
Catalyst : Cu + Small amounts of Zn + a component
Yield :

Down arrow: forward Up arrow: backward Home: beginning
End: end Return: to main menu

3. 공정지식의 frame식 표현

화학공정에 관련되는 지식들은 앞에서 언급한 바와 같이 production rule과 frame식 구조를 조합함으로써 매우 효과적으로 표현될 수 있다. 여러 공정들이 복잡한 network를 이루고 있는 플랜트에서는 대상이 되는 공정이나 화학물질을 하나의 object로 하고 이 object들을 각각 하나의 frame으로 나타내어 줄 수 있는데 이 경우 표현되는 object와 다른 object들과의 상호 연관관계를 frame내에 나타내어 줌으로써 전체적인 network의 효과적 표현이 가능해 진다. 이러한 데이타 베이스의 운용에 있어서는 해당되는 frame내의 각 요소들을 용이하게 찾아갈 수 있도록 하여 줌으로써 데이타의 처리속도와 능률을 높일 수 있는데 이러한 기능들은 매우 간단히 프로그램화 시킬 수 있다.

이와같은 데이타 표현구조는 특히 화학공정 엔지니어링에 효과적으로 활용될 수 있는데 대략 1970년대 후반부터 널리 활용되고 있다. 대표적인 예로 1977년 일본 Chiyoda회사의 CHEIS와 DPLS를 들 수 있다. 이 시스템들은 그러나 몇가지 이유로 해서 그 사용이 바로 중단되었다. 같은 계통의 데이타 베이스로서 ICI회사의 PEDB와 Chem Share회사의 Design MASTER, 그리고 Prosys Technology사의 Prodabas를 들 수 있다. Prodabas는 특히 화학공정의 설계를 위하여 개발된 데이타 베이스이다.

Frame식 데이타 표현구조에서는 여러가지 형태의 데이타와 정보, 그리고 각종 지식들을 하나의 frame구조안에 효과적으로

함축시킬 수 있으므로 필요한 부분에 대한 데이타를 신속하고 정확하게 대할 수 있으며 frame상호간의 유기적인 연결에 의해 복합적인 공정 전체에 있어서도 필요한 데이타의 운용이 매우 효율적으로 이루어질 수 있다. 하나의 object는 class로서 표현될 수 있으며 class의 모든 property들은 frame구조 내에 나타내어 진다. 이들 각각의 구조는 다음과 같다.

```
struct class {char class-name[MAX-STR];
              struct record *super-class;
              struct record *sub-class;
              struct frame *class-record; } ;
```

위에서 class_name은 표현되는 공정이나 화학물질의 title을 의미하며 super_class는 현재의 object가 속하고 있는 상위의 object를 의미하고 sub_class는 하위의 object를 나타낸다. 즉 이들 상호간의 관계는 다음의 그림 3-26에 보인 바와 같다.

Record 구조는 일반적인 linked list구조로서 일정한 size의 데이타를 나타내어 준다.

```
struct record {char property[MAX-STR];
               struct record *next; } ;
```

일반적으로 super_class나 sub_class들은 다수의 object들로 이루어 진다. 그러므로 super_class와 sub_class title

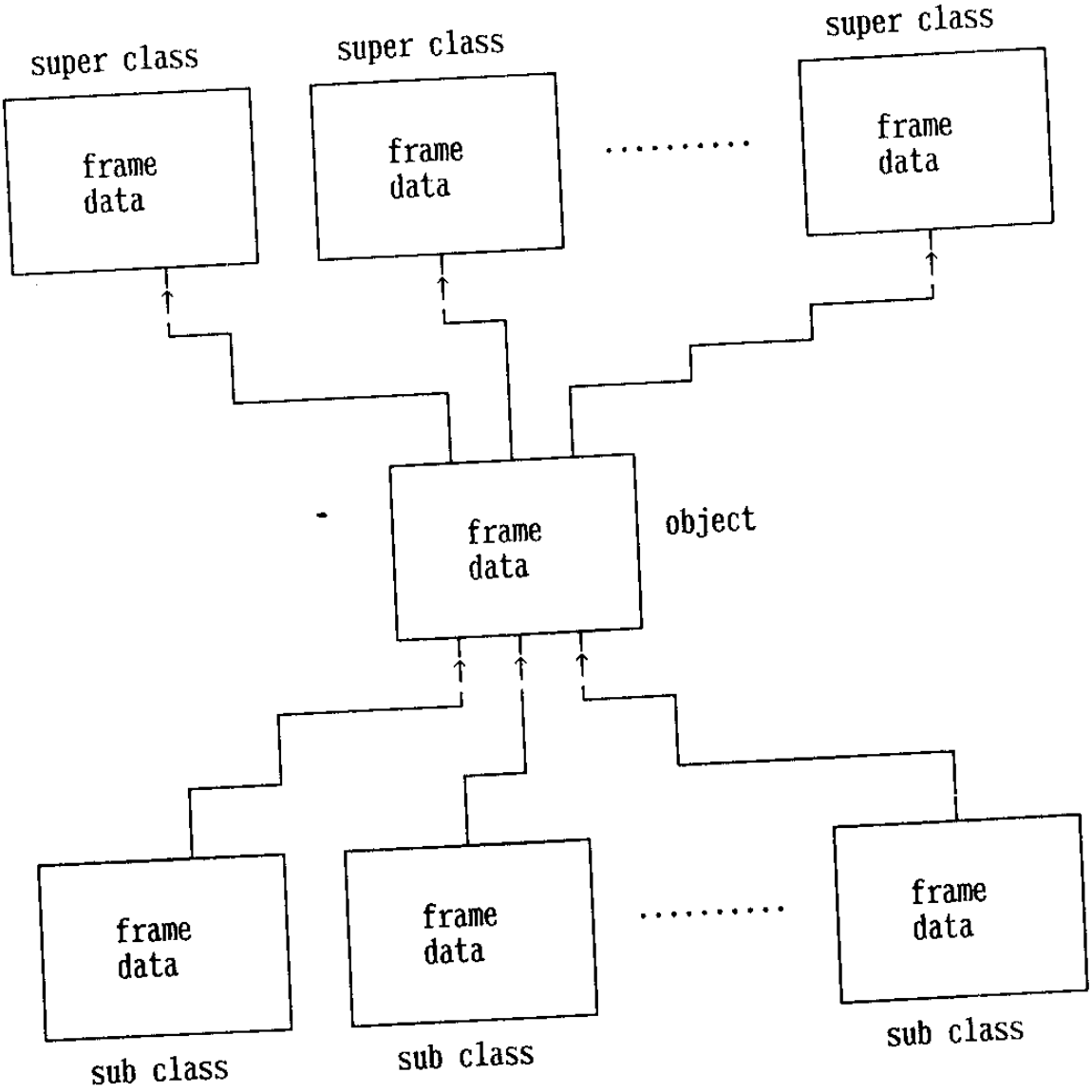


그림 3-26. Frame식 데이타 표현구조

항목을 위의 record와 같은 linked list구조로 표현함으로써 데이타의 표현에 있어서는 실질적으로 super나 sub class의 수에 아무런 제약을 받지 않는다.

대상이 되는 object의 property를 나타내어 주는 frame은 다음과 같은 구조를 갖는다.

```
struct frame {char name[MAX-STR];
              char formula[MAX-STR];
              char status[MAX-STR];
              struct frame *next; } ;
```

위의 구조는 하나의 단순한 보기이며 공정이나 화학물질의 특성과 사용자의 요구에 따라 임의로 편리한 구조로 바꿀 수 있다. 위의 보기는 대상 공정에 관련되는 반응물질들의 특성을 나타내어 주는 frame이다. 예를 들어 공정의 유통도에서 각 단위공정 장치에서 일어나는 반응과 관련되는 조업조건들을 나타내고자 한다면 frame을 다음과 같이 설정할 수 있을 것이다.

```
struct frame {char process-unit[MAX-STR];
              char reaction[MAX-STR];
              struct record *operation_condition;
              struct frame *next; } ;
```

Frame구조는 linked list형식을 취하고 있으므로 하나의 object에 관련되는 모든 데이타를 체계적으로 명료하게 표현할 수 있다. 위의 보기에서 operation condition을 나타내어 주는

pointer는 record구조를 취하는데 record구조 자체가 linked list구조이므로 주어진 process unit내에서의 모든 조업조건 및 반응조건들을 나타낼 수 있다.

4. 복합 데이타 구조

앞에서 언급한 바와 같이 화학공정의 데이타 베이스는 공정 자체가 다른 공정 unit들과 복합적인 network를 형성하고 있기 때문에 이들 상호관계를 명료하게 나타내 주어야 한다. 이러한 관계들은 frame식 고조에 의해 효과적으로 표현될 수 있음은 이미 앞에서 언급한 바와 같다. Frame구조를 활용한 복합적인 데이타 베이스 구성의 보기로서 다음의 표 3-5에 보인 공정관련 데이타를 다루어 보기로 한다.

표 3-5. 공정설계관련 데이타 [18]

조 성	명 칭	PFS ID	process
HOCH2CH2OH	에틸렌글리콜	PFS0001	Ethylene oxide hydration
HOCH2CH2OH	에탄디올	PFS0001	Ethylene oxide hydration
HOCH2CH2OH	에틸렌글리콜	PFS0002	Ethylene chloro-hydrin
HOCH2CH2OH	에탄디올	PFS0002	Ethylene chloro-hydrin
CH3OH	메탄올	PFS0003	CO-H2 synthesis
CH3OH	메틸알콜	PFS0003	CO-H2 synthesis
CH3OH	메탄올	PFS0004	Wood distillation
CH3OH	메틸알콜	PFS0004	Wood distillation

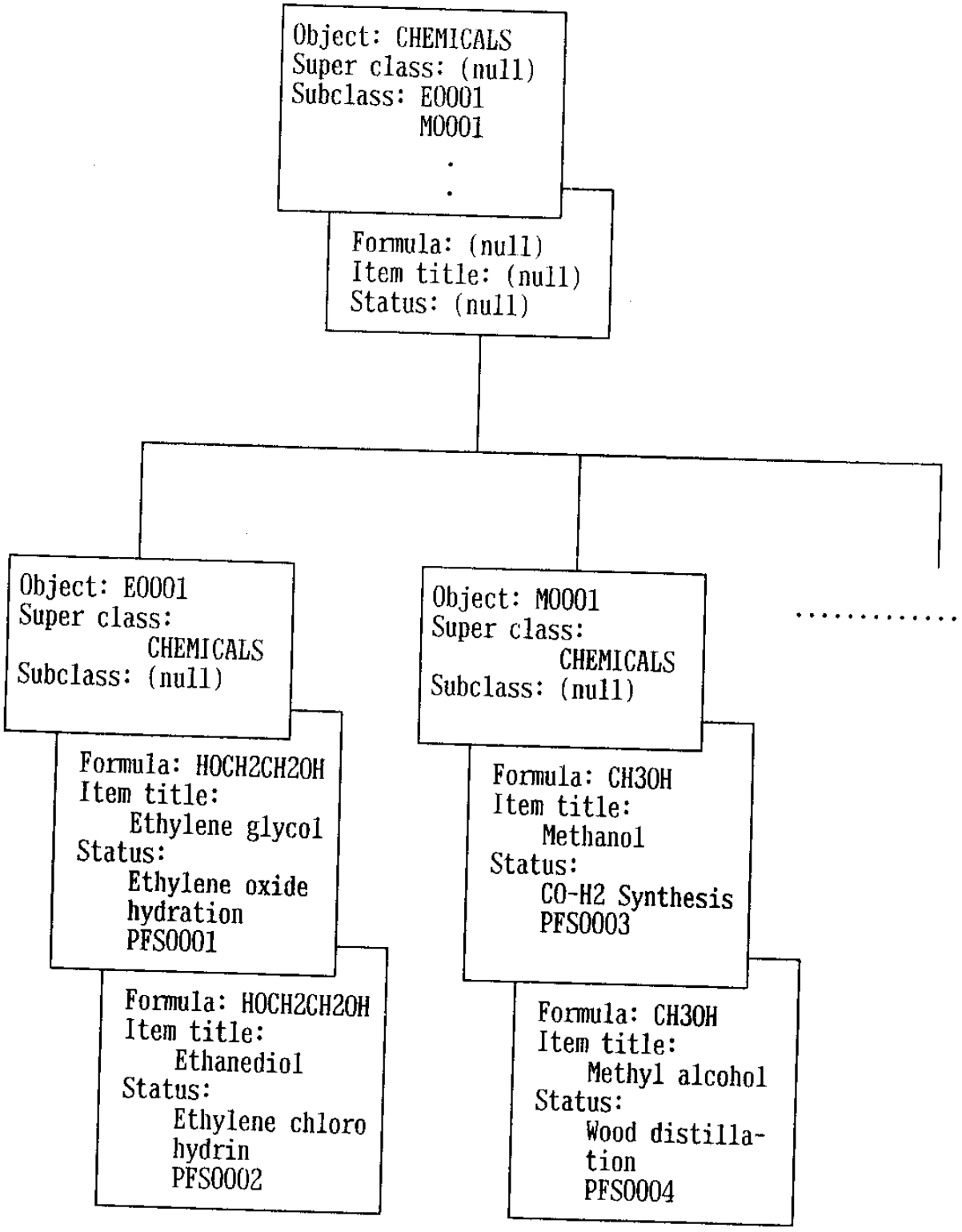


그림 3-27. Object데이타의 상호관계

표 3-5의 데이타는 앞에서 보인 frame구조에 따라 명칭은 item title slot, 조성은 formula slot에, 그리고 PES(process flowsheet) ID와 process는 status slot에 의해 나타낼 수 있다. 각 object 간의 관계는 그림 3-27에 보인바와 같다.

다음은 이들 데이타의 실제 입력과정이다. 각각의 object에 있어서 item title은 별개의 frame에 나타내어 진다. 하나의 object에는 최대 100개의 frame이 붙을 수 있는데 이정도의 size는 실제 플랜트의 표현에도 결코 부족하지 않을 것이다.

Object E0001의 입력 :

```
Object : E0001

Super class : CHEMICALS
Sub class :
```

Formula HOCH2CH2OH의 입력 :

```
Formula : HOCH2CH2OH

Item title : Ethylene glycol

Status : Ethylene oxide hydration  PFS0001
```

```
Formula : HOCH2CH2OH

Item title : Ethanediol

Status : Ethylene chlorohydrin  PFS0002
```

Object M0001의 입력 :

```
Object : M0001

Super class : CHEMICALS
Sub class :
```

Formula CH3OH의 입력 :

```
Formula : CH3OH

Item title : Methanol

Status : CO-H2 Synthesis  PFS0003
```

```
Formula : CH3OH

Item title : Methyl alcohol

Status : Wood distillation  PFS0004
```

Frame을 이용한 복합 데이타구조의 두드러진 특징은 object상호간의 정보교환 및 연결기능이 뛰어나다는 점이다. C 프로그램에서 linked list구조상의 변수의 검색은 매우 빠르게 이루어 지므로 그 size가 큰 데이타 베이스라도 신속하게 처리할 수 있다. Super class나 sub class의 member들은 당연히 독자적인 object로서 frame구조에 의해 표현되어지며 frame 내의 item title이나 formula, 또는 status들까지도 하나의 object로서 표현될 수 있다.

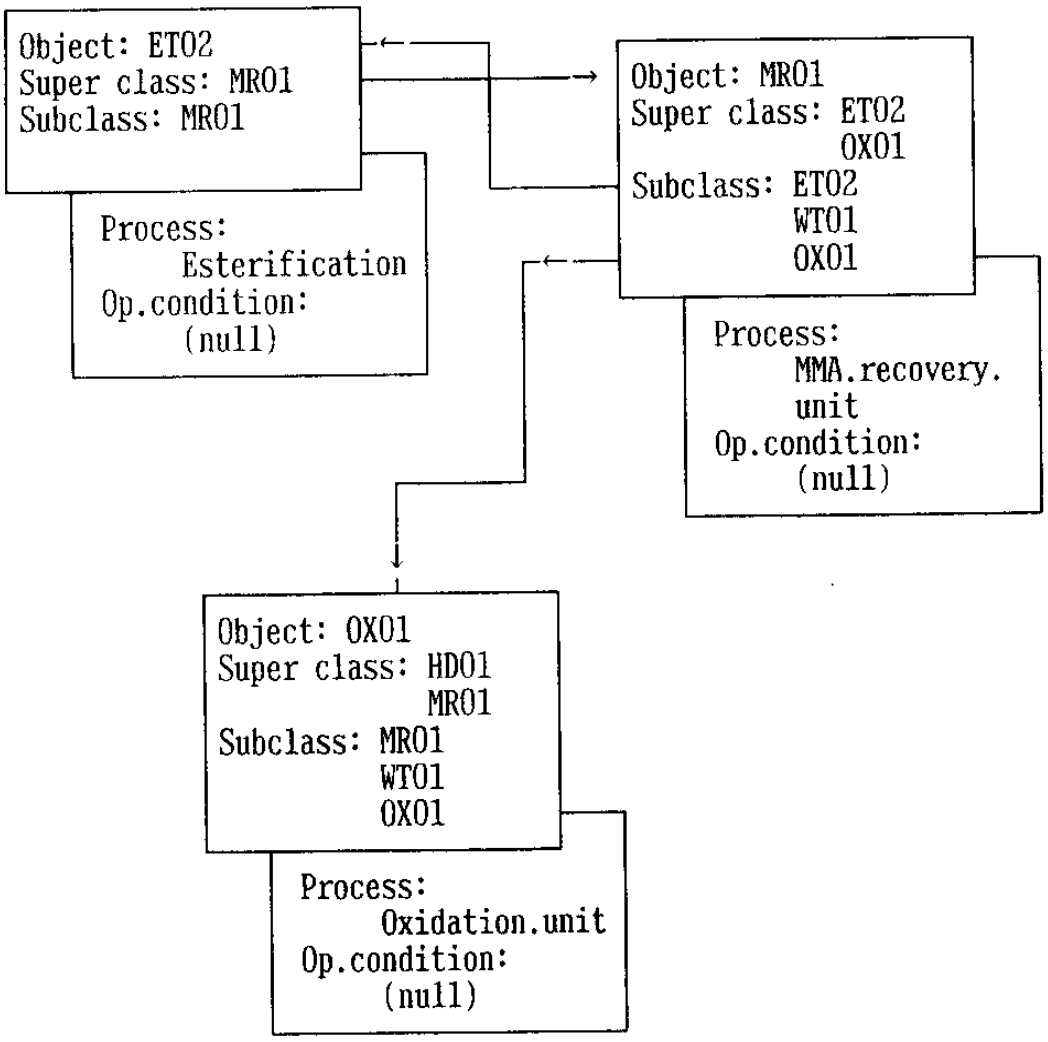


그림 3-28. Object상호간의 관계

그림 3-28에 보인 것은 메틸 메타크릴레이트 모노머(MMA)공정의 유통도의 일부를 frame구조로 표현하여본 것이다. 단위공정 상호간의 관계는 super 및 sub-class를 통하여 알 수 있으며 해당 공정에 대한 데이타는 그 object의 frame에 나타나 있다.

제 4 절 결 론

공정 데이타의 표현 및 운용은 tree 구조 및 linked list구조와 이에 바탕을 둔 frame식 구조에 의해 효율적으로 이루어질 수 있음이 확인되었다. 실질적인 데이타의 표현은 IF-THEN 형식의 production rule과 frame방법에 의해 이루어지고 데이타 베이스에 입력된다. 공정 데이타 베이스의 검색 및 보정 보완을 위해서는 depth-first 및 breadth-first search 방법과 이들의 혼합 및 iteration 기능을 가미한 방법이 이용되는데 데이타 베이스의 구조에 따라 적절한 방법이 이용되도록 하였다. 이러한 데이타 베이스의 search기법은 이후의 제 4 장에서 다루게 될 전문가 시스템에서 실제적으로 이용되고 있다.

화학공정에서 다룰수 있는 모든 종류의 데이타를 포괄적으로 처리 운용할 수 있는 데이타 베이스의 구축에는 많은 시간과 노력이 필요하다. 그러나 기본적인 골격을 제대로 갖추어 놓으면 사용자의 요구에 따라 적절한 format을 갖춘 데이타 처리구조를 용이하게 만들어낼 수 있으며 특수한 기능이 요구되는 공정의 표현을 위해서 필요한 구조를 갖는 데이타 베이스를 손쉽게 구성할

수 있다. 그러나 물성치나 일반적인 자료등의 데이타는 이미 개발되어 있는 데이타 베이스를 활용하는 것이 보다 능률적일 것이다.

참 고 문 헌

(1) M.V. O'Neill, "Implementing a Chemical Process Plant Expert System", ISA Trans., **Vol.26**, No.1, 19-25 (1987).

(2) R.A. Herrod and J. Rickel, "An Expert System for Simulating a Glass Annealing Process: AI Applications in the Glass Industry", ISA Trans., **Vol.25**, No.4, 23-29 (1986).

(3) K. Lien, G. Suzuki, and A.W. Westerberg, "The Role of Expert Systems Technology in Design", Chem. Eng. Sci., **Vol.42**, No.5, 1049-1071 (1987).

(4) R. Banares-Alcantara, A.W. Westerberg, E.I. Ko, and M.D. Rychener, "DECADE-A Hybrid Expert System for Catalyst Selection-I. Expert System Consideration", Comput. Chem. Eng., **Vol.11**, No.3, 265-277 (1987).

(5) R.R. Leitch, "Modelling of Complex Dynamic Systems", IEE Proc., **Vol.134**, Pt. D, No.4, 245-250 (1987).

(6) R.J. Paul and G.I. Doukidis, "Artificial Intelligence Aids in Discrete-Event Digital Simulation Modelling", IEE Proc., **Vol.134**, Pt. D, No.4, 278-286 (1987).

(7) G.A. Bekey, "Knowledge Based Systems in Modeling, Simulation, and Identification", IFAC, 69-74, 1988.

(8) F. Gomide, W.C. Amaral, L.V.R. Arruda, G. Favier, A.S. Barbara, and W. Fontanini, "Expert System Identification: The Supervisory Approach", IFAC, 1691-1696, 1988.

(9) T.X. Lin, L.B. Lan, and Z.Y. Jong, "The Automating Identification System with Intelligence", IFAC, 1708-1712, 1988.

(10) M. Haest, G. Bastin, M. Gevers, and V. Wertz, "An Expert Workstation for System Identification", IFAC, 1990-1995, 1988.

(11) R.L. Kirkwood, M.H. Locke, and J.M. Douglas, "A Prototype Expert System for Synthesizing Chemical process Flowsheets", Comput. Chem. Eng., **Vol.12**, No.4, 329-343 (1988).

(12) J.C. Hoskins and D.M. Himmelblau, "Artificial Neural Netork Models of Knowledge Representation in Chemical Engineering", Comput. Chem. Eng., **Vol.12**, No.9/10, 881-890 (1988).

(13) R. Banares-Alcantara, E.I. Ko, A.W. Westerberg, and M.D. Rychener, "DECADE-A Hybrid Expert System for Catalyst Selection-II. Final Architecture and Results", Comput. Chem. Eng., **Vol.12**, No.9/10, 923-938 (1988).

(14) L. Beltramini and R.L. Motard, "KNOD-A Knowledge Gased Approach for Process Design", Comput. Chem. Eng., **Vol.12**, No.9/10, 939-958 (1988).

(15) D.R. Myers, J.F. Davis, and D.J. Herman, "A Task-Oriented Approach to Knowledge-Based Systems for Process Engineering Desing", Comput. Chem. Eng., **Vol.12**, No.9/10, 959-971(1988).

(16) I.P. Popchev and N.P. Zlatareva, "Intelligent Problem-Independent Tool for Plausible Inference", IFAC, 372-377, 1987.

(17) H.P. Wang and R.A. Wysk, "Intelligent Reasoning for Process Planning", Computers in Industry, **Vol.8**, 293-309 (1987).

(18) Y.W. Huang and L.T. Fan, "Designing an Object-Relation Hybrid Database for Chemical process Engineering", Comput. Chem. Eng., **Vol.**12, No.9/10, 973-983 (1988).

(19) H.D. Engelmann, H.H. Erdmann, R. Funder, and K.H. Simmrock, "The Solving of Complex Process Synthesis Problems Using Distributed Expert System", Comput. Chem. Eng., **Vol.13**, No.4/5. 459-465 (1989).

(20) E. Lahdenpera, E. Korhonen, and L. Nystrom, "An Expert System for Selection of Solid-Liquid Separation Equipment", Comput. Chem. Eng., **Vol.13**, No.4/5, 467-474 (1989).

(21) J. Lapointe, B. Marcos, M. Veillette, and G. Laflamme, "BIOEXPERT An Expert System for Wastewater Treatment Process Diagnosis", Comput. Chem. Eng., **Vol.13**, No.6, 619-630 (1989).

(22) B. Chen, J. Shen, Q. Sun and S. Hu, "Development of an Expert System for Synthesis of Heat Exchanger Newtorks", Comput. Chem. Eng., **Vol.13**, No.11/12, 1221-1227 (1989).

(23) M. Hofmeister, L. Halasz and D.W.T. Rippin, "Knowledge-Based Tools for Batch Processing Systems", Comput. Chem. Eng., **Vol.13**, No.11/12, 1255-1261 (1989).

(24) Herbert Schildt, "ADVANCED C", McGraw-Hill, California, U.S.A. (1988).

(25) C. Townsend and D . Feucht, "Designing and Programming personal expert systems", TAB Books Inc, U.S.A. (1986).

(26) J. Martin and S. Oxman, "Building expert systems", Prentice-Hall, U.S.A. (1988).

제 4 장 진단 및 제어용 전문가 시스템의 개발

제 1 절 서 론

공장의 조업에 있어서 고장이나 사고의 신속하고 정확한 진단과 공정의 효율적이고 경제적인 최적제어는 가장 중요한 당면과제이다. 1960년대 이래 공정의 진단 및 제어방법에 대한 연구가 다양하게 이루어져오고 있으나 개발된 이론이 실제 현장에서 이용되는 경우는 매우 드물었다. 이론의 개발에 있어서는 매우 단순화된 공정모델과 비실제적인 조업조건이 이용되고 있으나 현장 조업에서는 이론에서 무시되었던 여러 사항들이 복합적으로 영향을 미치게 되며 또한 이론이나 수식으로는 다룰 수 없는 현장 조업기술자의 경험적 지식이 조업에 반영되게 된다. 결국 개발된 진단이론이나 제어이론은 제한된 양의 공정조업지식에 의존할 수 밖에 없으며 따라서 이들 이론의 실제적인 활용에는 뚜렷한 한계가 존재하는 것이다.

현대의 컴퓨터는 아주 빠른 속도로 소형 경량화 및 고성능화 되어가고 있으며 조업 현장에서의 컴퓨터의 설치 활용은 이미 보편화 되어가고 있다. 공정의 조업, 특히 공정의 진단과 제어에 있어서 전문가 시스템은 앞서 언급한 여러 이론적인 지식들은 물론 현장 기술자들이 지니고 있는 경험적인 지식까지를 컴퓨터를 통하여 활용할 수 있게 하여주는 가장 적절한 도구가 되며 이의

적용은 공장 전산화를 위한 투자의 가치를 극대화 시켜주는 가장 빠른 길이 될 것이다.

전문가 시스템의 개발에 있어서는 전문가 시스템 shell의개발과 지식 베이스의 개발이 작업의 주종을 이룬다. 빠른 시일내에 적은 경비로 전문가 시스템을 개발하기 위한 일반적인 방법은 적절한 시스템 shell을 구입하고 이를 이용하여 지식 베이스를 구축해 나가는 방법이다. 1989년을 기준하여 전세계적으로 약 2,000여개의 전문가 시스템이 개발되어 있는 것으로 추산되는데 이들 가운데 화학공정 조업에 관련되는 전문가 시스템은 약 50여개 정도이며 또한 공정조업용 전문가 시스템의 개발에 이용될 수 있는 shell은 약 20여 종류가 있다.

기존의 시스템 shell을 구입하여 전문가 시스템의 개발에 활용하는 것이 경제적이고 신속한 방법이기는 하지만 화학공정의 종류가 지극히 다양하고 많으며 다른 분야와 비교할 때 상호간에 유사성이 적으므로 개발되는 전문가 시스템의 용도와 활용범위에 한계가 있다. 예를 들어 Naphtha cracker를 위한 전문가 시스템 개발용 shell을 이용하여 촉매선택용 전문가 시스템을 개발하는 데에는 상당한 무리가 따르게 될 것이다. 어떠한 종류의 화학공정에도 두루 이용될 수 있는 시스템 shell의 개발은 사실상 불가능한 일이지만 기본적인 시스템 shell을 개발하는 일은 특정 공정을 위한 전문가 시스템 개발기술의 축적을 위해서 뿐만 아니라 전문가 시스템 software의 해외 예속화 탈피를 위해서도 필수적이다.

본 연구의 1차년도 목표 가운데 하나는 기본적인 시스템 shell의 개발연구이지만 이는 어디까지나 가장 기초적인 prototype의 shell이 될 것이다. 전문가 시스템 프로그램 언어의 선택은 매우 중요한 문제인데 LISP, PROLOG, C, FORTRAN들 가운데에서 화학공정의 특성을 가장 잘 표현할 수 있는 언어로서 C와 LISP가 최종적으로 고려되었다. 화학공정 조업의 전산화에 있어서는 공정의 모사 및 설계기능이 또한 중요한 요소가 되기 때문에 보다 처리속도가 빠르고 공정지식 표현이 자유로운 C가 주 언어로 이용될 것이며 C와 LISP의 호환체계가 아울러 연구될 것이다.

제2절 전문가 시스템 개발요소

1. 전문가 시스템의 기본구성

전문가 시스템은 공장에서 조업 전문 기술자의 역할을 대신할 수 있는 컴퓨터 software를 말한다. 전문가 시스템에 저장되고 활용되는 지식은 대상이 되는 공정의 조업관련 전문 기술자가 지니고 있는 제반 경험적 지식과 공정관련 자료들로 이루어 진다. 전문가 시스템은 조업 기술자와는 달리 스스로 지식을 만들어 낼 수 있는 능력이 없기 때문에 조업 기술자가 지니고 있는 제반 조업지식과 자료들은 적절한 프로그램 언어를 이용하여 프로그램되어야 한다. 전문가 시스템에 저장된 지식은 손쉽게 보정시킬 수 있으며 또한 새로운 지식의 삽입이 용이하다. 아울러 시스템 사용중

에도 각 문제해결 단계에서의 진행상황이나 원인의 도출이 매우 손쉽게 이루어진다. 전문가 시스템이 적절하게 사용될 수 있는 공정조업 분야로서는 공정의 감시 및 제어, 설계, 공정진단, 공정의 최적화, 조업자료의 관리 및 분석, 조업상황의 예측등을 들 수 있다.

그림 4-1은 전문가 시스템의 기본적인 구성을 보인 것이다. 전문가 시스템의 개발에 있어서 가장 많은 작업이 요구되는 부분은 user interface와 지식 베이스 부분이다. 특히 시스템의 규모로 볼 때 user interface부분은 전체 전문가 시스템의 40-50 %를 차지한다. 그림 4-1에서 Explanation subsystem은 시스템으로 하여금 사용자에게 결론이 도출되는 이유를 설명해 주는 부분이다. 즉 전문가 시스템의 집행결과에 대한 원인을 제시하고 문제의 해결에 어느 특정된 데이타가 필요하게 된 이유를 설명해 준다. 지식 베이스 editor는 새로운 지식의 추가 및 기존 지식의 보정, 그리고 지식 베이스에 대한 전반적인 관리기능을 수행한다. Inference engine은 문제의 해결을 위해서 지식 베이스의 데이타를 실제적으로 적용하는 기능을 수행한다. 전문가 시스템 shell은 그림 4－1에 보인 부분들 가운데에서 지식 베이스와 데이타 베이스 부분을 제외한 다른 모든 부분들로 구성된다.

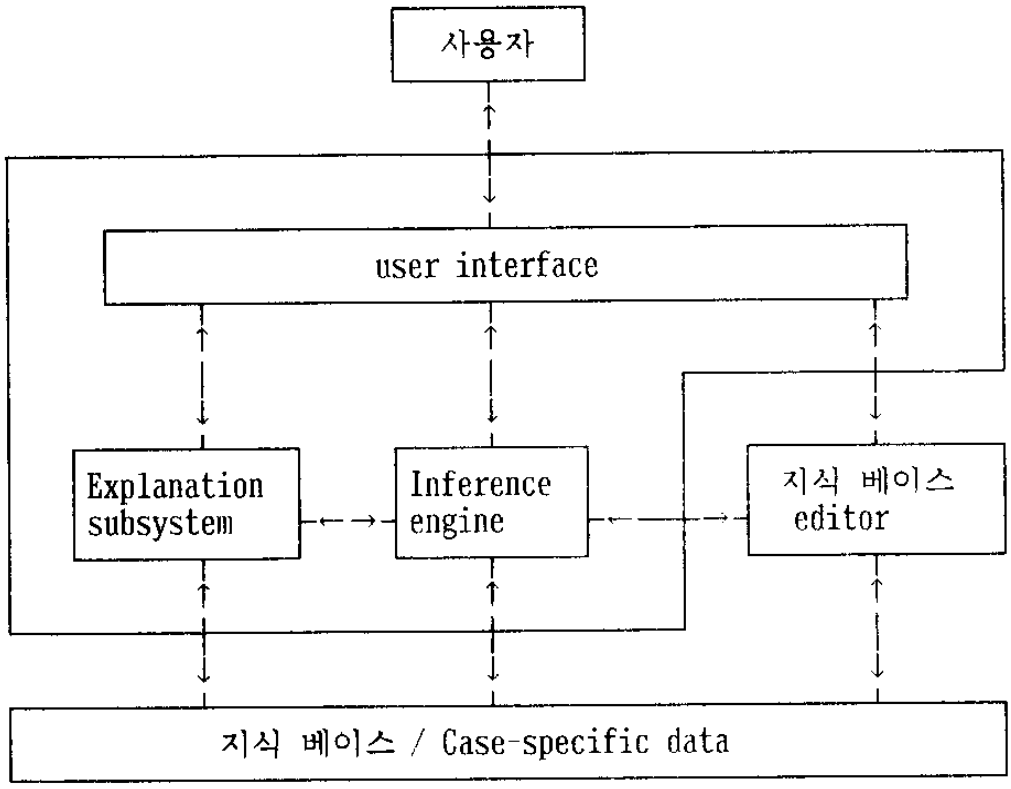


그림 4 — 1. 전문가 시스템의 기본구성

2. 전문가 시스템의 개발단계

전문가 시스템의 개발에 있어서는 LISP이나 C, 또는 PROLOG등의 언어에 바탕을 둔 표준화된 프로그램 환경의 개발이나 OPS, EMYCIN, 또는 Hearsay — Ⅱ와 같은 전문가 시스템 구축용 범용성 언어 tool의 개발이 그 주류를 이루어 오고 있다. 화학공정용 전문가 시스템의 개발에 있어서는 단위 블럭 구축결합 방법이 매우 유용하게 이용될 수 있는데 이 방법에서는 효율적인 스크린 editor라든가 데이타 베이스 운용 시스템, 또는 menu-driven interface와 같은 단위 블럭들을 조합시켜서 하나의 시스템을

이루게 한다. 여기에 대상공정의 모사, 최적화 및 설계 패키지가 단위 블럭으로서 추가될 수 있으며 각종 제어구조 패키지와 진단 방법 패키지들이 역시 단위 블럭들을 이루게 된다. 따라서 이미 개발되어 있는 기존의 여러 공정관련 패키지들이 약간의 수정만을 거쳐서 활용될 수 있다.

전문가 시스템이 시판되어 실제로 활용되기 시작한 것은 대략 1980년부터이지만 정성적인 지식의 표현과 rule의 구축 및 운용, 데이타 베이스의 구성 및 활용, 그리고 사용자와의 inter-face등 전문가 시스템에 대한 연구가 활발해지기 시작한 것은 대략 1960년대 중반부터이다. 그림 4－2에 보인 것은 이미 잘 알려져 있는 대표적인 몇가지 전문가 시스템의 발전과정이며 그림 4－3은 그 개발소요 시간을 도시한 것이다.

이들 초기의 주요 전문가 시스템들은 주로 미국의 스탠포드 대학과 MIT, 그리고 카네기 멜론 대학을 중심으로 하여 개발되어 왔다. MIT의 MACSYMA는 수학적인 문제들의 효율적인 해결을 위한 시스템이고 스탠포드 대학의 DENDRAL은 mass spectrogram 데이타로부터 복잡한 유기 화합물의 구조를 설명하여 주는 시스템이다. 이른바 범용성 전문가 시스템으로 개발된 시스템들은 매우 간단하고 단순한 문제의 해결에만 유효한 경우가 일반적이다. 따라서 화학공정에 있어서는 범용성 시스템의 적용범위와 용도가 극히 제한될 수 밖에 없다.

전문가 시스템 shell의 구축에 있어서는 프로그램 언어의 적절한 선택이 매우 중요한 문제이다. 지금까지 전문가 시스템 프

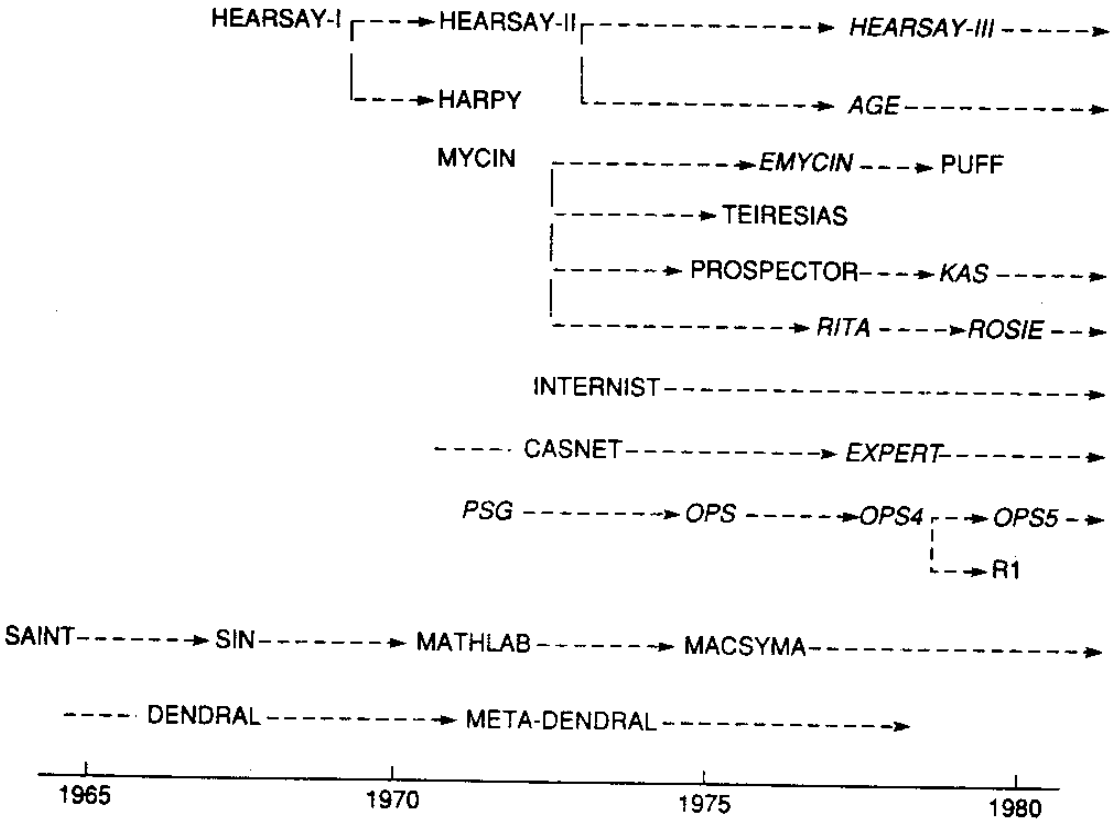


그림 4-2. 대표적인 전문가 시스템 발전과정 [52]

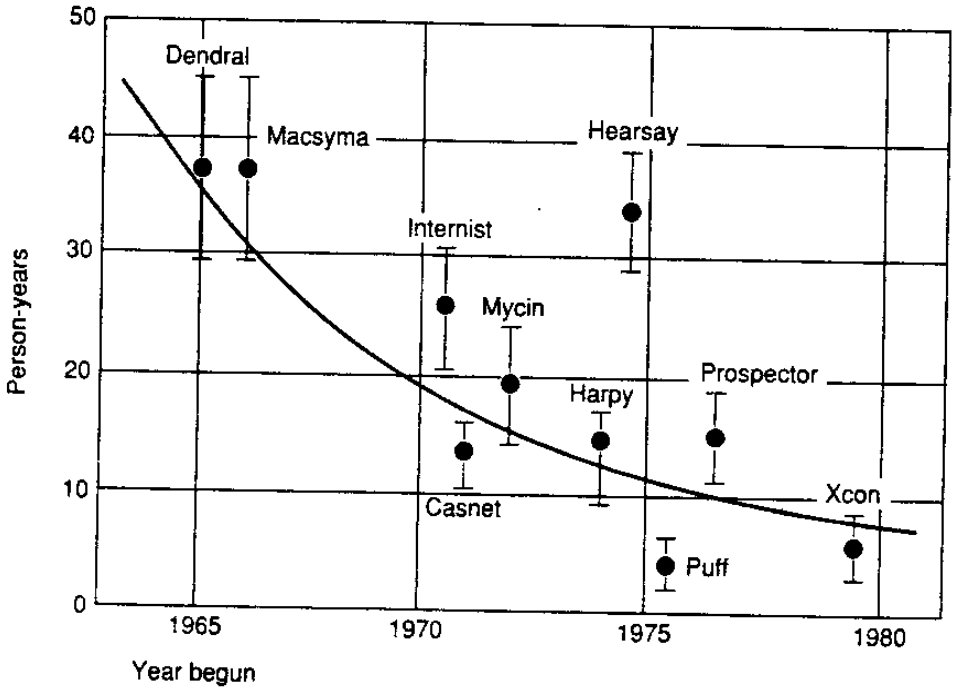


그림 4-3. 대표적인 전문가 시스템 개발소요시간 [52]

표 4-1. 시스템 shell 및 언어와 컴퓨터의 관계

	Programming Languages					Expert System Shells				
	LISP	PROLOG	OPS5	C	Pascal	ART	KEE	Insight 2+	EXSYS	PC Plus
LISP Machines										
LMI (GigaMos)	●	●		●	●	●	●			
Symbolics	●	●			●	●	●			
Texas Instruments	●	●		●		●	●			
Xerox	●	●	●				●			
Mainframes										
DEC	●	●	●	●	●	●	●	●	●	
IBM	●	●	●	●	●					
Minicomputers										
DEC	●	●	●	●	●	●	●	●	●	
Hewlett-Packard	●	●		●	●					
Workstations										
Apollo	●	●	●	●	●					
IBM	●	●		●	●		●			
Sun	●	●	●	●	●	●	●			
Personal Computers										
Apple	●		●	●	●			●		
Compaq	●	●	●	●	●			●	●	●
IBM	●	●	●	●	●		●	●	●	●
Texas Instruments	●	●		●	●					●

로그램 언어의 대종을 이루어 온 것은 LISP와 PROLOG이며 이들 외에도 C, PASCAL, FORTRAN등이 이용되어 왔다. 프로그램 언어의 선택에 있어서는 개발 대상분야 외에도 타언어와의 호환성 및 사용될 컴퓨터의 기종을 아울러 고려하여야 한다. 표 4-1은 대표적인 프로그램 언어 및 shell과 컴퓨터 기종간의 관계를 보인 것이다.

전문가 시스템 구축을 위한 프로그램 언어와 컴퓨터 기종의 대략적인 선택이 이루어 지면 shell을 직접 개발하거나 또는 시중에서 구입하여 지식 베이스를 구축하는 작업에 들어가게 된다. 지식 베이스의 구축을 위한 tool이 필수적으로 갖추어야 할 기능들은 그림 4-4에 요약되어 있다. 지식 베이스 구축용 tool에서 요구되는 대부분의 기능들은 이미 단위 시스템이나 블럭으로서 개발되어 있으므로 적절한 것을 선택하여 사용할 수 있다. 그러나 공정에 따라서는 특수한 기능이 필요한 경우가 있고 또한 사용자가 독특한 기능을 요구하는 경우가 있으므로 그 개발기술을 확립해 두어야 할 것이다.

전문가 시스템에서 지식 베이스의 구축은 knowledge engineer가 수행하여야 할 주된 역할이다. Knowledge engineer는 대상 공정의 전문 기술자로부터 지식을 얻어내어 프로그램화 시켜서 사용이 용이하고 정확한 지식 베이스를 구성하게 된다. 이를 위해 knowledge engineer는 적절한 지식 표현방법을 결정하여야 하고 기본적인 search기법을 선택하여야 하며 아울러 사용자와의 interface를 적절히 선정하여야 한다. 대략적으로 만들어진 초보적인

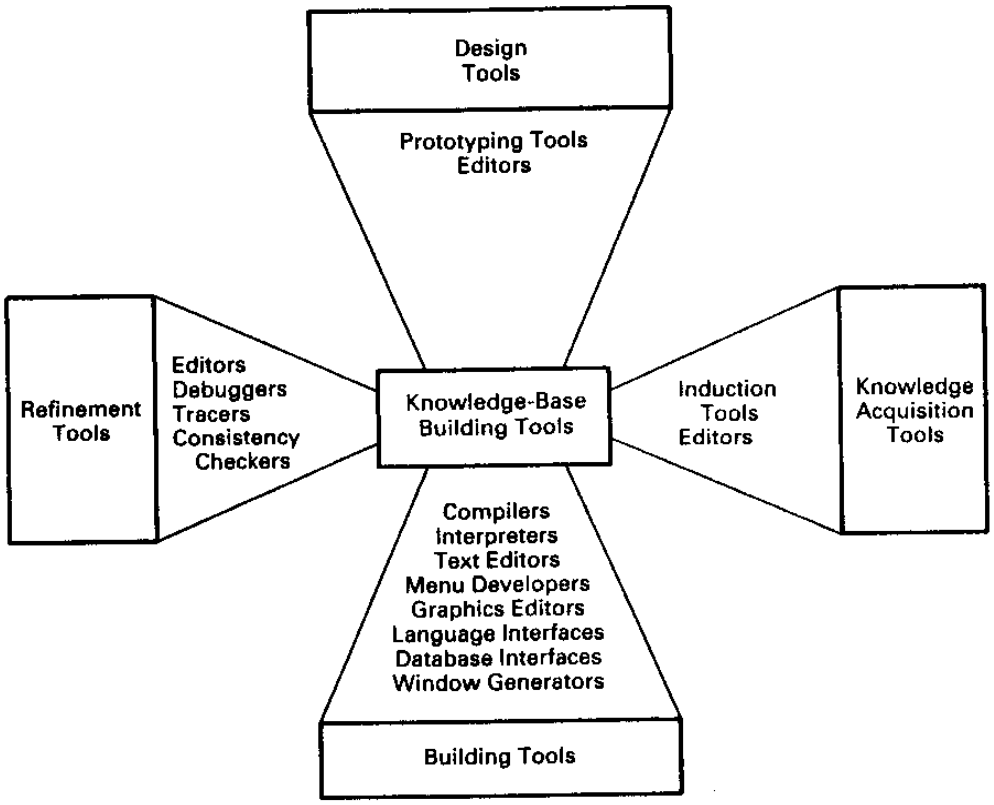


그림 4-4. 지식 베이스 구축용 tool의 소요기능

prototype의 전문가 시스템은 이후 계속적인 보정과 보완을 거쳐서 실제로 현장에서 활용할 수 있는 시스템으로 확장시켜야 하는데 만일 개발된 prototype 시스템이 적절치 않다고 판단되는 경우에는 다시 새로운 prototype시스템의 개발을 추진하여야 한다. 미국의 DEC에서 컴퓨터의 configuration용으로 개발한 XCON은 1981년 처음 개발되었을 당시에는 약 500여개의 rule을 가진 시스템이었으며 VAX 780의 configuration에 이용되었는데 이후 점차적으로 보정 보완되어서는 1984년에는 약 4,000개의 rule을 가진 대형 시스템으로 확장되었다. 실제로 XCON은 DEC에서 생산 판매되는 거의 모든 컴퓨터 기종의 configuration을 담당하고 있다. 그림 4-5는 전문가 시스템의 개발 cycle을 나타낸 것이다.

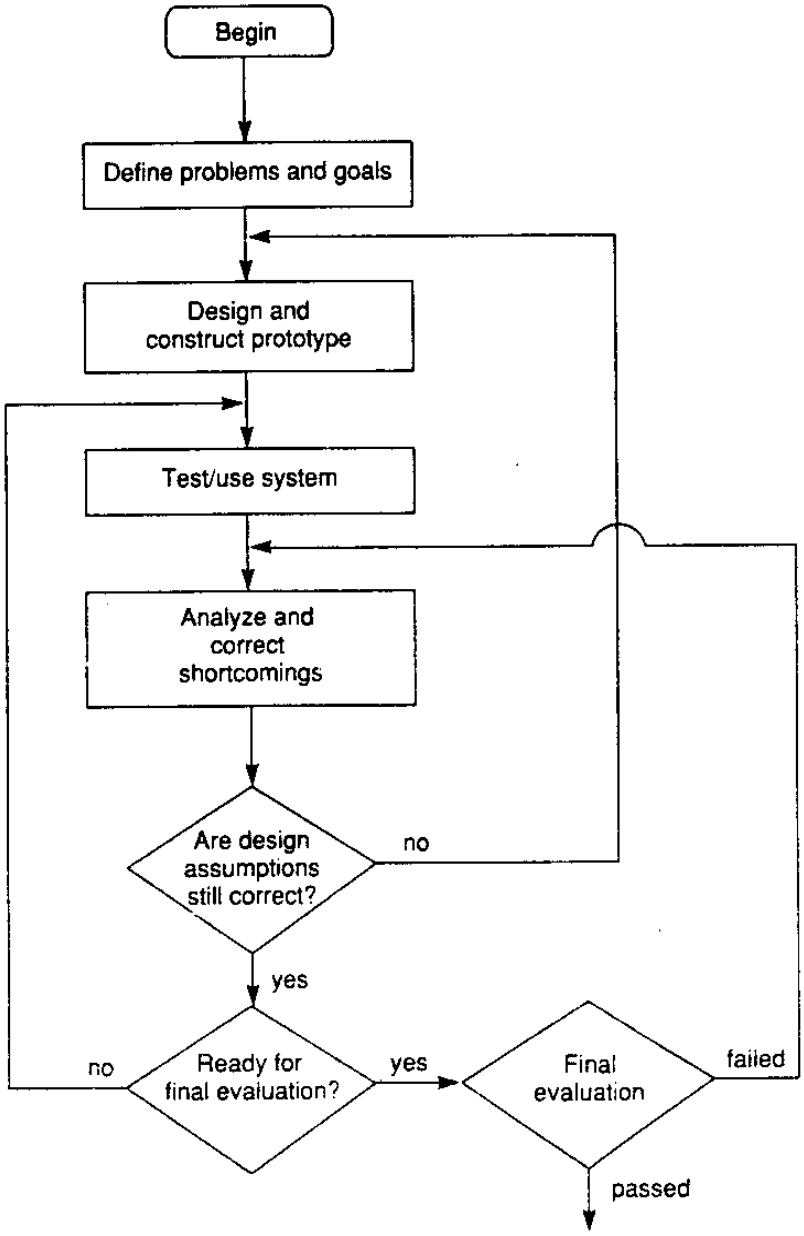


그림 4-5. 전문가 시스템의 개발cycle

제 3절 전문가 시스템의 개발

1. Rule의 표현

전문가 시스템에서의 rule은 IF-THEN 형식의 production rule이 주류를 이룬다. THEN 부분이 나타내어 주는 결론을 도출시키기 위하여 충족되어야 할 조건들로 구성되는 IF 부분은 하나 이상의 다수의 상황들로 이루어 진다. 어떤 하나의 대상공정이나 unit, 또는 상황은 다수의 IF-THEN rule들로서 표현되는데 이는 그림 4-6에 도시한 바와 같다.

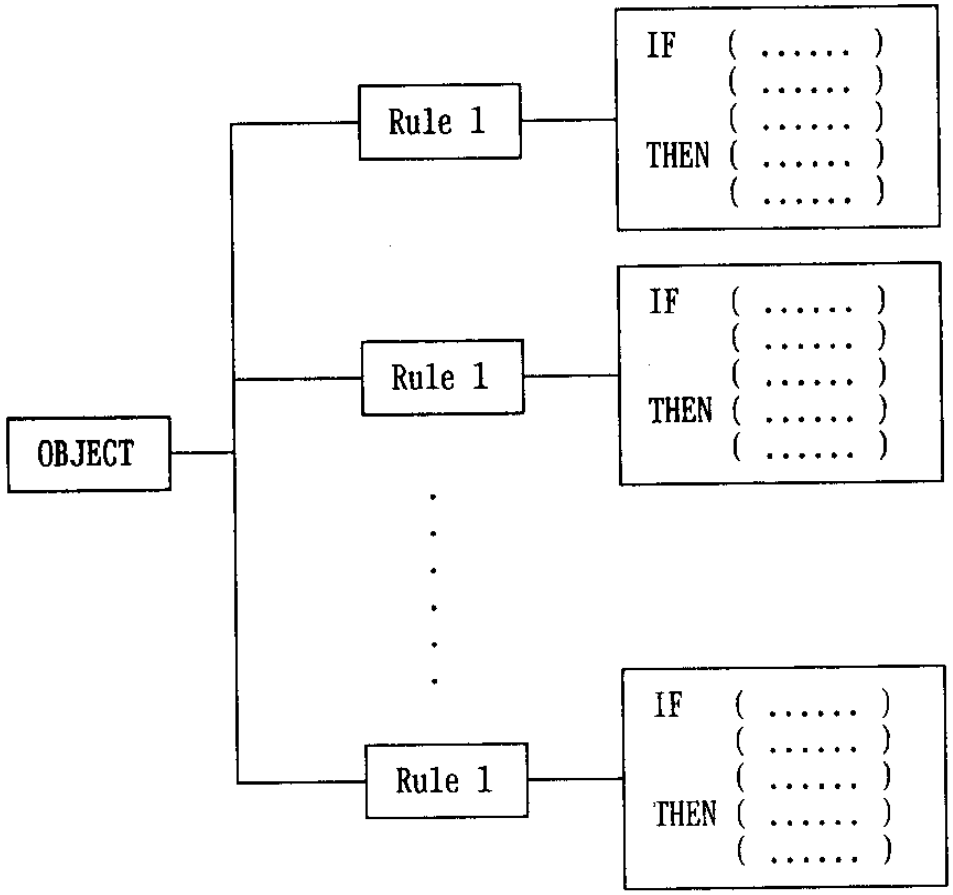


그림 4-6. Rule set의 구성

그림 4-6 에서 OBJECT는 대상공정이나 unit, 또는 상황을 의미한다. 위의 그림 4-6 에 도시한 rule 구조는 다음과 같이 표현할 수 있다.

```
struct object r_base[MAX_OBJ];
struct object { char object_name[MAX_NAME];
                struct rule rule_list[MAX_ITEM];
                int num_of_rules; };
```

r_base는 OBJECT들의 집합으로 구성되는 rule 베이스로서 단위공정이나 간단한 단위 plant를 표현하여 주는 지식 베이스가 된다. 하나의 object를 표현하여 주는 구조는 object의 title과 그 object에 소속되는 rule들의 집합, 그리고 rule 들의 수효를 나타내어 주는 변수로 구성된다. Rule을 표현하여 주는 구조는 다음과 같다.

```
struct rule { char rule_name[MAX_NAME];
              struct record *if_list;
              struct record *then_list; };
```

하나의 rule은 IF - 부분을 표현하여 주는 if-list와 THEN-부분을 표현하여 주는 then-list로 구성된다. 상황이나 공정, 또는 대상 물질에 따라 IF - 부분과 THEN - 부분의 size가 각각 따라 정해지거나 제한되어서는 안되기 때문에 if-list와 then-list는 연속된 형태의 linked list 구조로 되어 있으며 이

구조는 record로서 표현된다.

```
struct record { char property[MAX_STR];
                struct record *next; };
```

Record 구조에서 property는 실제 현상을 표현하는 부분인데 그 size는 MAX_STR개의 character로 제한되어 있다. 현상의 표현이 그 이상의 size로 되는 경우에는 얼마든지 record 구조의 next pointer를 이용해서 늘려 나갈 수 있으므로 아무리 긴 지식일지라도 그 표현이 가능하다.

Rule 표현의 보기로서 증류공정 제어 rule을 나타내 보기로 한다. Object name을 distillation으로 하고 간단한 5개의 rule들을 앞에서 언급한 구조에 따라 나타낸 결과는 다음과 같다.

1) If distillate or bottoms flow rate are used to control a level, use a P controller.

2) If the reflux ratio is greater than five, use of the R-V control configuration is not allowed.

3) If the relative volatility of the distilled mixture is less than 1.2, do not use temperature measurements for inferential control.

4) If the reflux and distillate valves are saturated, use the steam valve to control the reflux-drum level.

5) If the RGA is unknown and the process gains are known, calculate the RGA by using the RGA-function procedure.

OBJECT: Distillation

RULE NAME: Rule 1

IF: distillate or bottoms flow rate are used to control a valve

THEN: use a P controller

OBJECT: Distillation

RULE NAME: Rule 2

IF: the reflux ratio is greater than five

THEN: use of the R-V control configuration is not allowed

OBJECT: Distillation

RULE NAME: Rule 3

IF: the relative volatility of the distilled mixture is less than 1.2

THEN: do not use temperature measurements for inferential control

OBJECT: Distillation

RULE NAME: Rule 4

IF: the reflux and distillate valves are saturated

THEN: use the steam valve to control the reflux-drum level

OBJECT: Distillation

RULE NAME: Rule 5

IF: the RGA is unknown and the process gains are known

THEN: calculate the RGA by using the RGA-function procedure

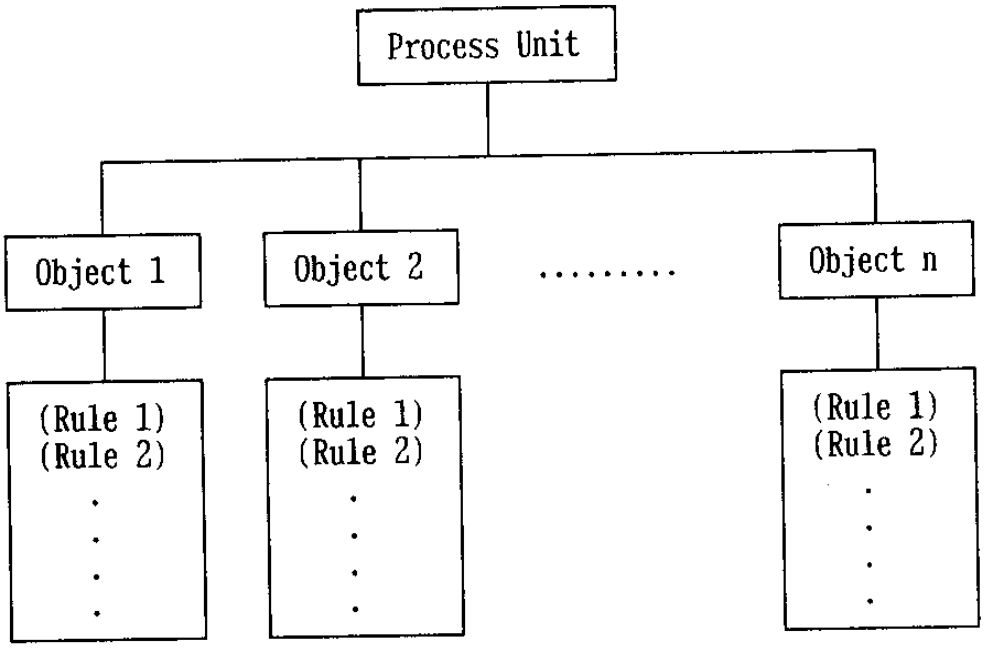


그림 4-7 . Rule베이스의 계층적 구조

하나의 단위 공정이나 플랜트를 나타내어 주는 rule 베이스는 다수의 object들로 구성된다. 즉 그림 4-7에 도시한 바와 같이 전체적인 rule 베이스는 계층적 구조로 이루어지게 된다.

2. Rule의 운용

전문가 시스템의 효율성은 rule베이스에 저장된 제반 rule들을 얼마나 효과적으로 신속 정확하게 문제의 해결에 적용하느냐에 따라 좌우된다. 지식 베이스의 규모가 커짐에 따라 사용되는 컴퓨터의 저장능력과 처리속도가 전문가시스템의 능률을 크게 좌우하게 되었다. 단위공정에 대한 rule베이스의 운용은 주어지는 menu에 따라 사용자가 필요한 기능을 선택하여 쓸 수 있다. 주요 rule운용 menu와 function들은 다음의 표 4-2에 보인바와 같다.

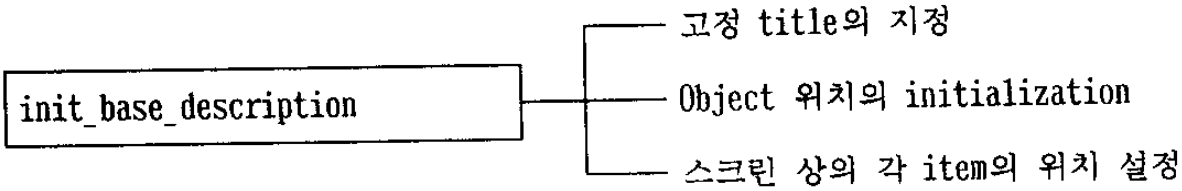
표 4-2. Rule 베이스 운용 menu

Rule menu	functions
Rule 데이타의 입력 Rule을 이용한 결과의 유추 Rule 베이스의 저장 기존 rule 베이스의 loading	enter_rule_data infer_rule save_r_base load_r_base

가. Rule 데이타의 입력

공정 지식의 입력에 있어서는 우선 입력되는 지식이 컴퓨터 스크린에 나타내어지고 기억 장소에 데이타 파일로서 저장될 수

있는 format이 결정되어야 한다. Rule 데이타 format의 결정 기능은 init_base_description function에 의하여 수행되는데 그 주요기능은 다음과 같다.



Rule 데이타의 입력은 주로 get_data function에 의해 이루어 진다. get_data function은 사용자가 정하여 주는 변수와 그 size에 해당되는 지식 데이타를 받아 들인다. IF-THEN rule의 경우 rule title이 입력된 후에 IF - 및 THEN - 데이타가 입력되는데 앞서 언급한 바와 같이 IF-및 THEN - 데이타는 linked list 구조로 되어 있으므로 IF-condition 및 THEN-부분이 각각 100 item이내인 범위에서 자유로이 데이타를 입력시킬 수 있다.

지식 데이타 입력함수인 enter_rule_data의 집행과정은 다음의 그림 4-8에 보인 바와 같으며 각 함수들의 기능은 표 4-3에 요약되어 있다.

표 4-3 . 데이타 입력함수의 기능

function	기 능
display_object_title	object title의 제시
display_rule_title	IF와 THEN 부분의 제시
get_object_position	object stack 위치의 설정
get_rule_position	rule stack 위치의 설정
get_data	지식 데이타의 입력

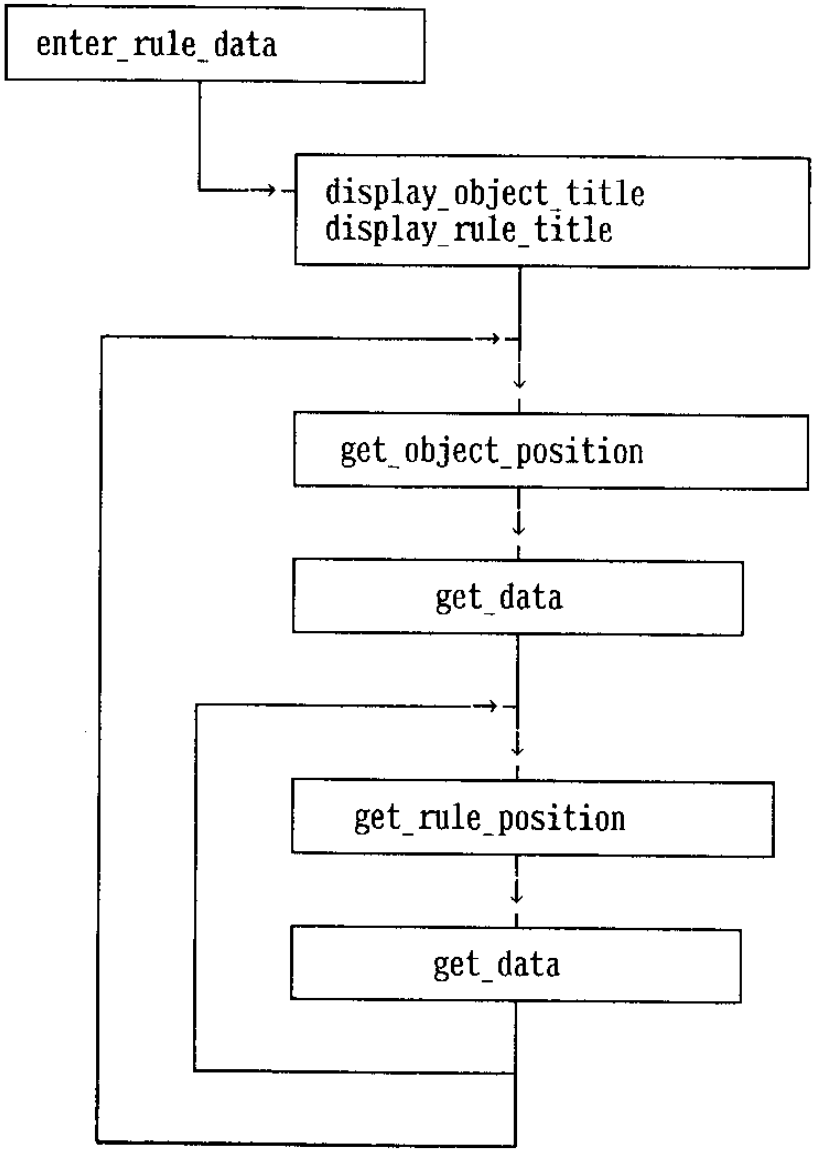


그림 4-8. 지식 데이타 입력 유통도

지식 데이타 입력의 간단한 보기로서 rule베이스에 다음과 같은 5번째의 rule을 입력시킨다고 하자.

Rule 5 : If it eats meat, it is a carnivore

이 rule에서 it는 동물을 가리킨다. 컴퓨터 스크린 상에 나타나는 입력상태 및 지시문들은 다음 그림 4-9에 보인 바와 같다.

OBJECT: Animal

RULE NAME: Rule 5

IF: it eats meat

THEN:

Enter if-list property (RETURN to quit):

그림 4-9. 데이타 입력 지시문

나. 결과의 유추

결과의 유추는 실제 현상과 rule 베이스를 대조하여 봄으로써 이루어 지는데 전문가 시스템이 수행하고자 하는 기능과 대상 공정 및 unit에 따라 달라질 수 있다. 현재의 prototype 전문가 시스템에는 사용자와 시스템 간의 간단한 응답에 의하여 결과를 유추하는 기능과 공정의 상태를 입력시킴으로써 야기되는 상황의 진단 또는 예측기능이 구축되어 있다. 이러한 기능 자체로도 간단한 공정이나 unit에 충분히 활용될 수 있는데 2차년도에서는 상기 기능들의 보완 및 확장은 물론 몇가지의 다른 inference기능들과 데이타 처리기능들이 추가될 것이다.

Rule 베이스를 이용한 inference 기능은 infer_rule( ) function에 의하여 이루어 진다. infer_rule( ) 에서 주된 유추기능을 수행하는 function은 infer인데 infer에서는 우선 true - 및 false_stack들을 검증한다. infer function에서 true - 및 false_stack의 검증을 수행하는 function은 각각 examine_ture_stack과 examine_false_stack이다. 결과의 유추에 있어서는 rule베이스의 rule들을 추적해 나가면서 IF-부분이 현 상황과 일치하면 그 부분을 true_stack에 저장하고 그렇지 않은 경우에는 false_stack에 저장하게 된다. 이렇게 함으로써 어느 rule의 IF-부분가운데 일부라도 examine_false_stack에 의해 false_stack내에 존재함이 밝혀지면 그 rule은 더이상의 검증이 필요없게 되므로 프로그램 집행시간이 매우 단축되어 진다. 시스템이 rule을 검증해 나가는 동안 사용자는 제시되는 사실이 true인지, false인지를 응답하게 된다. 시스템 스스로가 따로 저장된 데이타 베이스를 이용하여 제시되는 사실이 true인지, false인지를 판단할 수도 있다. 왜 그러한 사실에 제시되었는지 궁금할 경우에는 사용자는 why menu를 선택하여 그 이유를 알아볼 수 있다.

Rule 유추기능을 수행하는 infer_rule의 집행과정은 다음의 그림 4-10에 보인 바와 같으며 각 함수들의 기능은 표 4-4에 요약되어 있다.

store_prop_stack function에 의해 true - 및 false-부분이 저장되는 stack은 prop_stack이며 이 stack은 다음

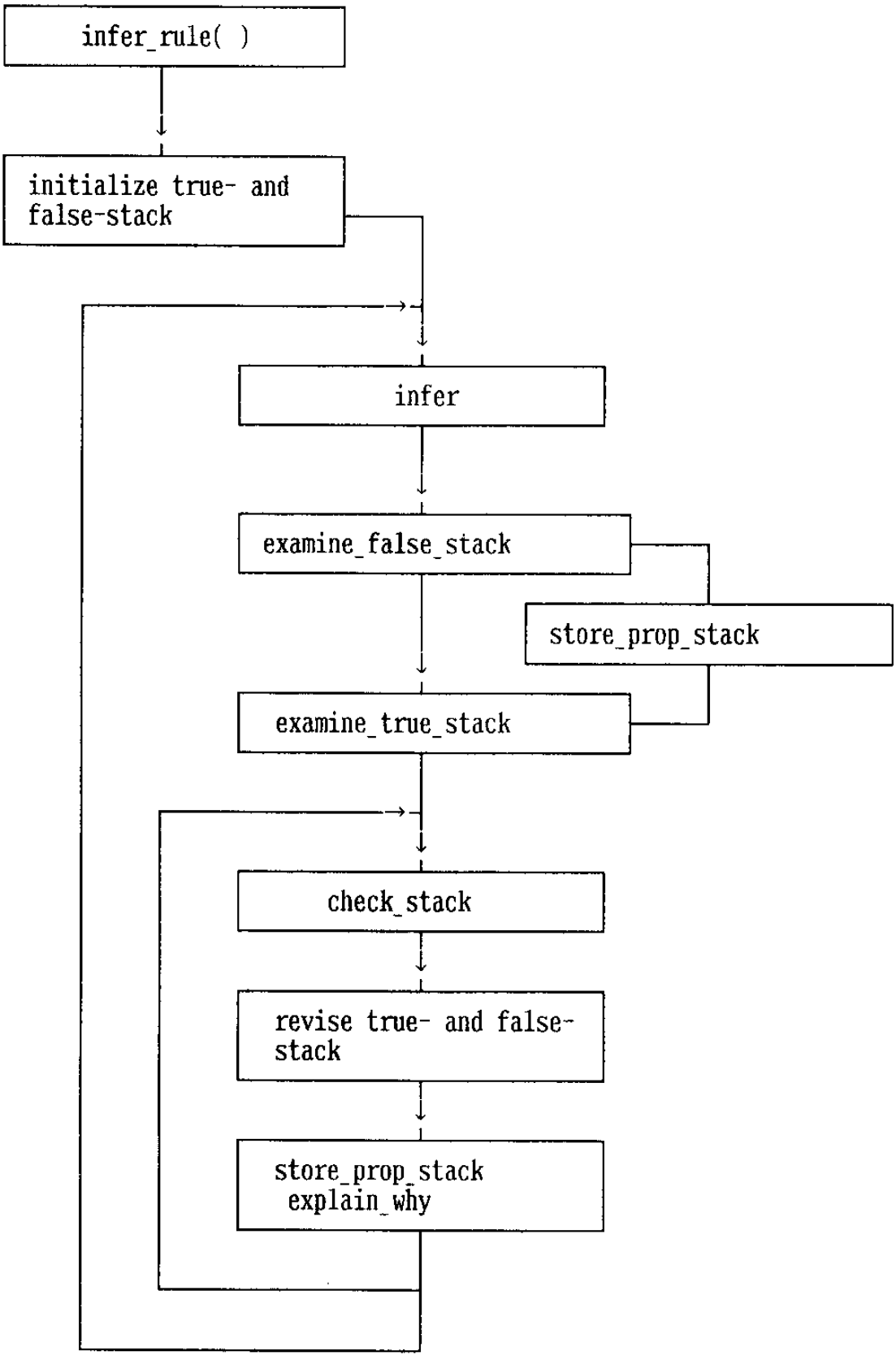


그림 4-10 . Rule 유추기능의 유통도

표 4-4 . Rule 유추함수의 기능

function	기 능
infer	보조함수들의 집행 및 true-와 false-stack의 revise
examine_false_stack	false-stack의 검증
examine_true_stack	true-stack의 검증
check_stack	rule 사실의 검증필요성 여부결정
store_prop_stack	유추이유의 설명을 위한 false- 및 true-부분의 저장
explain_why	rule 유추이유의 제시

과 같은 deposit 구조를 갖는다.

```
struct deposit { char item[MAX_NAME];
                 char property[MAX_STR];
                 char true_or_false; };
struct deposit prop_stack[MAX_ITEM];
```

deposit 구조에서 item은 저장되는 rule의 title이며 property는 해당되는 IF-부분이고 true_or_false는 't' 또는 'f'이다.

Rule유추의 간단한 보기로서 다음과 같은 rule이 rule 베이스에 저장되어 있다고 하자.

Rule 1: If it has pointed teeth, has claws, and has forward eyes, then it is a carnivore.

Rule 베이스에서 이 rule의 IF-부분은

IF it has pointed teeth
it has claws
it has forward eyes

와 같은 format으로 저장되고 THEN-부분은

THEN it is a carnivore

와 같은 format으로 저장된다. 위의 rule에서 it는 물론 animal을 나타낸다. Animal에 관한 따로 마련된 데이타 베이스가 없을 경우에는 시스템은 컴퓨터스크린을 통하여 사용자가 염두에 두고 있는 것에 대한 사실과 rule베이스의 rule의 IF-부분과의 합치 여부를 조사하게 된다. 시스템은 별다른 지시나 제약이 없을때는 다음과 같이 첫번째 rule인 Rule 1의 IF-부분부터 조사한다.

Question [1]: it has pointed teeth.
Is it true? ((T)rue, (F)alse, (W)hy):

만일 사용자가 (W)hy menu를 선택함으로써 위의 Question〔1〕이 제시된 이유를 알고자 한다면 시스템은 다음과 같이 응답할 것이다.

```
Trying rule Rule 1.
It is true that:
                    it has pointed teeth

Question [2]: it has claws.
              Is it true? ((T)rue, (F)alse, (W)hy):
```

Rule 1의 IF부분의 세번째 사항까지 true로서 응답이 끝나면 시스템은 THEN부분의 결론을 제시하여 준다.

```
Question [3]: it has forward eyes.
              Is it true? ((T)rue, (F)alse, (W)hy): t

According to the rule Rule 1:
     it is a carnivore .

Continue? (Y/N):
```

3. 공정진단구조

공정의 진단에 있어서는 공정의 현재 상황을 인식함으로써 현재 일어나고 있는 사태의 정확한 원인과 결과를 도출해 내거나 또는 앞으로 일어날 상황을 예측하는 기능이 주류를 이룬다. 공정의 상황에 영향을 미치는 변수는 일반적으로 여러개인데 이들 가운데 몇개의 변수에 대한 측정이 일률적으로 이루어 진다면 공정진단 문

제는 더욱 수월해 지고 정확해 진다.

단위공정에 대한 진단 지식 베이스의 운용은 주어지는 menu에 따라 사용자가 필요한 기능을 선택하여 쓸 수 있다. 여기에 이용되는 function들은 뒤에서 다루게 될 시스템 기본 function들에 바탕을 두고 있다. 주요 운용 menu와 function 들은 다음의 표 4-5에 보인바와 같다.

표 4-5. 진단지식 베이스 운용 menu

menu	functions
진단지식 rule 의 format	define_record
진단지식의 입력	enter_record
진단지식 베이스의 check	examine_list
필요한 진단지식의 search	find_node
진단지식의 revision	revise_list
불필요한 진단지식의 삭제	delete_item
공정의 진단	use_f_rule
진단지식 및 상황의 저장	save_list&event
기존 진단지식 베이스의 loading	load_list&event

가. 진단지식의 입력

진단rule의 구성은 공정조업 전문 기술자가 지니고 있는 지식을 프로그램화 시킴으로써 이루어 지는데 이의 입력은 뒤에서 언급하게 될 일반 데이타 입력기능을 활용함으로써 가능하다. 진단rule의 format은 다음과 같은 구조로 되어있다.

```
struct record_list { char *data;
                     struct record_list *prev;
                     struct record_list *next; };
```

진단 지식은 character pointer인 data에 저장되는데 사용자는 data에 해당 공정이나 unit의 주요 변수 및 장치에 관한 필요한 항목들을 필요한 만큼 모두 규정지워 둘 수 있다.

한편 변수의 상태에 따라 야기될 수 있는 상황들은 다음과 같은 event-list 구조로 표현된다.

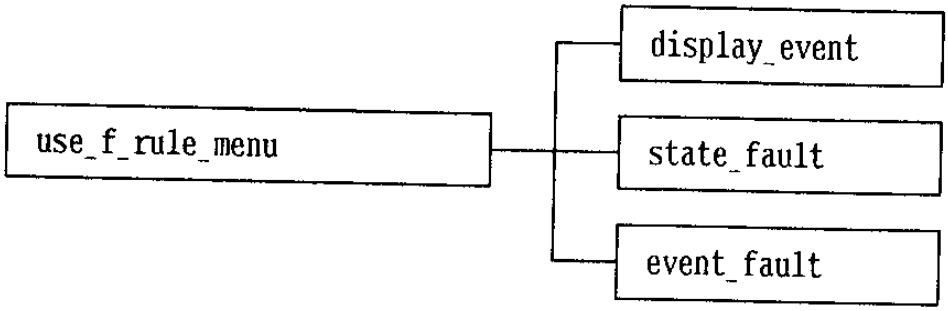
```
struct event_list { char event_name[FILE_NAME_SIZE];
                    char fact[MAX_STR];
                    struct event_list *next; };
```

편의상 발생가능한 상황들에 각각 적당한 title을 붙여서 프로그램은 이들 title을 찾아 가도록 하였는데 이렇게 함으로써 event들을 별도로 다루는 작업도 매우 용이하게 이루어질 수 있다. Event의 입력은 enter_event_data함수에 의하여 이루어진다.

나. 진단기능의 구조

공정이나 대상unit의 진단에 있어서는 공정이나 unit의 현 상태를 알고서 앞으로 야기될 상황이나 사고를 도출해 내는 경우와 현재 발생한 상황에 대하여 사용자와의 문답에 따라 가능

한 원인을 추적해 내는 경우의 두가지로 나누어 생각해 볼 수 있다. 상황에 따라 사용자는 어느 한가지 방법을 택할 수 있는데 공정이나 unit의 조업 데이타가 on-line으로 컴퓨터에 입력됨으로써 결과를 유추해 내는 기능은 앞으로 그 작업이 진행되어 현재의 시스템에 추가될 것이다. 진단기능의 선택은 use_f_rule_menu에 의하여 이루어 진다.



display-event routine은 title 그대로 저장된 event 데이타 내용을 스크린에 나타내어 주는 기능을 수행한다. 스크린 상에는 해당 공정이나 unit에 대하여 발생가능한 상황이나 사고 각각에 부여된 event title과 그 내용이 나타나는데 현재는 각각의 event의 size가 최대 80 character이내로 제한되어 있으며 그 이상의 상황은 다수의 별개 event들로 나누어 표현하는 방식을 채택하고 있다. 상황표현의 size를 제한하지 않는다던가 상황을 graphic이나 다른 적절한 기호로써 표현하는 시도는 2차년도부터 진행될 것이다.

다. 상황의 도출

예상되는 상황의 도출은 state_fault routine에 의하여 수행된다. state_fault routine은 일차적으로 공정이나 unit내의 각 장치나 부품들의 가능한 상태를 check한 다음에 사용자로 하여금 장치의 현재 상태를 입력시키도록 한다. 장치 상태의 check 기능은 check_state가 수행하는데 이로써 사용자가 장치의 상태를 잘못 입력시키는 경우를 바로잡을 수 있다. 예를 들어 valve 3의 상태를 알아보는데 사용자가 "too high"라고 응답했다면 이는 진단에 적용될 수 없는 잘못된 응답이 될 것이다. check_state에 의하여 valve 3에 적용될 수 있는 응답은 "open"과 "closed"의 두가지로 규정될 것이며 시스템은 올바른 응답을 계속 다시 요구할 것이다. 동일한 공정이나 unit에 대하여 진단작업을 계속 수행할 경우 매번 check_state를 집행시킬 필요는 없다. 즉 시초에 check_state를 집행시킨 결과를 기억장소에 저장해 두면 이후는 더이상 이 routine을 집행시킬 필요가 없고 저장된 결과만을 계속 이용하면 된다.

주요 상황도출 기능은 state_infer routine에 의하여 수행된다. state_infer는 사용자가 입력시킨 장치의 상태를 rule베이스의 IF-부분과 대조한 다음 결과를 도출하게 되는데 정보가 더 필요한 경우에는 추가로 사용자에게 해당되는 장치 상태의 입력을 요구하게 된다.

상태진단기능을 수행하는 state_fault의 집행과정은 다음의 그림 4-11에 보인 바와 같으며 각 함수들의 기능은 표 4-6에 요

약되어 있다.

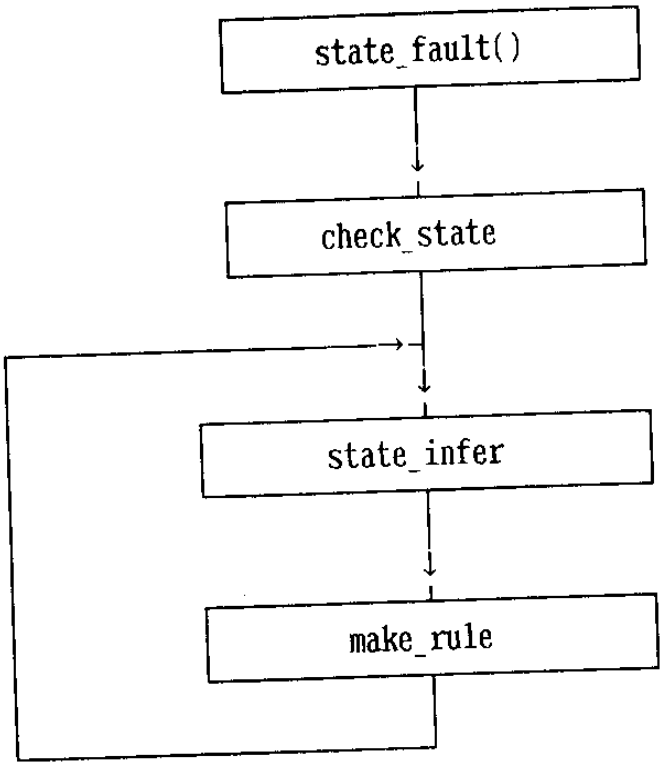


그림 4-11 . 상태진단기능의 유통도

표 4-6 . 상태진단 routine의 기능

function	기 능
state_fault check_state state_infer make_rule	보조함수들의 집행 장치상태의 검증 진단기능의 수행 rule 사실의 제시

라. 원인의 유추

어떤 상황이 돌발했을 때 그 원인의 파악은 공정진단에서 가장 중요한 기능이다. 시스템이 도출하는 결과의 정확도는 진단지식 베이스에 따라 좌우된다. 보다 체계적이고 폭넓은 결과의 제시는 조업전문 기술자의 경험적 지식 이외에도 대상 공정의 모사 및 최적화 프로그램들이 시스템에 결합되어 활용됨으로해서 가능해진다. 대상 공정이나 unit에 대한 추가의 정보가 없으면 전문가 시스템은 경우에 따라 다수의 원인들을 제시하여 주며 그들 가운데 어느 것이 정확한 원인인지는 전적으로 사용자가 결정해야 한다. 실제에 있어서는 야기된 상황의 원인은 어느 하나의 장치의 결함이나 조업착오에 있는 경우는 매우 드물고 여러장치들의 결함이 서로 복합적으로 작용하거나 조업착오와 장치의 고장이 어우러져 그 요인이 되기도 한다. 따라서 공정진단에 있어서 그 원인이나 상황의 도출은 복잡한 tree나 signal flow diagram 구조를 따라 진행된다. 제 5 장에서 다루게 되는 공정진단방법에서는 공정의 인식에 바탕을 둔 기단기법을 제시하고 있다. 공정상태의 정확한 인식은 야기되는 사태의 정확한 판단과 분석에 필수적이며 공정진단의 기본이 된다.

우리가 다루고 있는 prototype 공정진단 시스템에서는 가장 간단한 유추기능을 이용하고 있다. 사용자의 선택에 따라 스크린에 제시되는 상황들 가운데 하나를 선택하면 시스템은 진단지식 베이스를 찾아 그 상황을 야기시켜 주는 원인들을 유추해 낸다. 복합적인 공정에서는 제시되는 원인들 자체가 각각 하나의 상황이

되기도 하며 이 경우에는 시스템 스스로가 다시 그 원인을 찾아 내어야 하는데 이 부분에 대한 작업은 2차년도에 이루어질 것이다.

상황에 대한 원인도출 기능은 event_infer routine에 의하여 수행된다. event_infer는 사용자가 입력시킨 현재의 상황을 event베이스의 내용과 대조한 다음 원인을 도출하게 되는데 정보가 더 필요한 경우에는 추가로 사용자에게 해당되는 장치 상태의 정보를 요구하게 된다.

원인유추기능을 수행하는 event_fault의 집행과정은 다음의 그림 4-12에 보인 바와 같으며 각 함수들의 기능은 표4-7에 요약되어 있다.

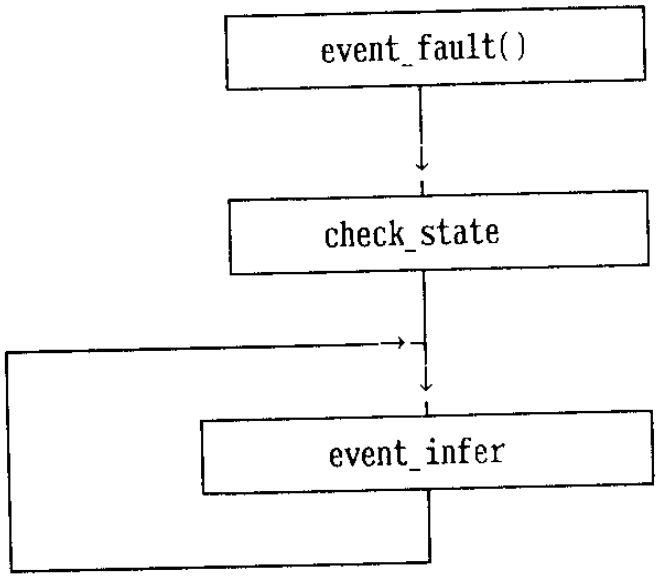


그림 4-12. 원인유추기능의 유통도

표 4-7 . 원인유추 routine의 기능

function	기 능
event_fault	보조함수들의 집행
check_state	장치상태의 검증
event_infer	진단기능의 수행

마. 진단기능 수행예

공정 진단기능 수행의 간단한 보기로서 그림 4-13에 보인 tank의 경우를 살펴보기로 한다. 이 tank시스템은 극히 간단한 장치로서 별다른 복잡한 진단과정을 필요로 하지는 않으나 이러한 간단한 시스템에 대한 진단기능을 단계적으로 구축해둠으로써 규모

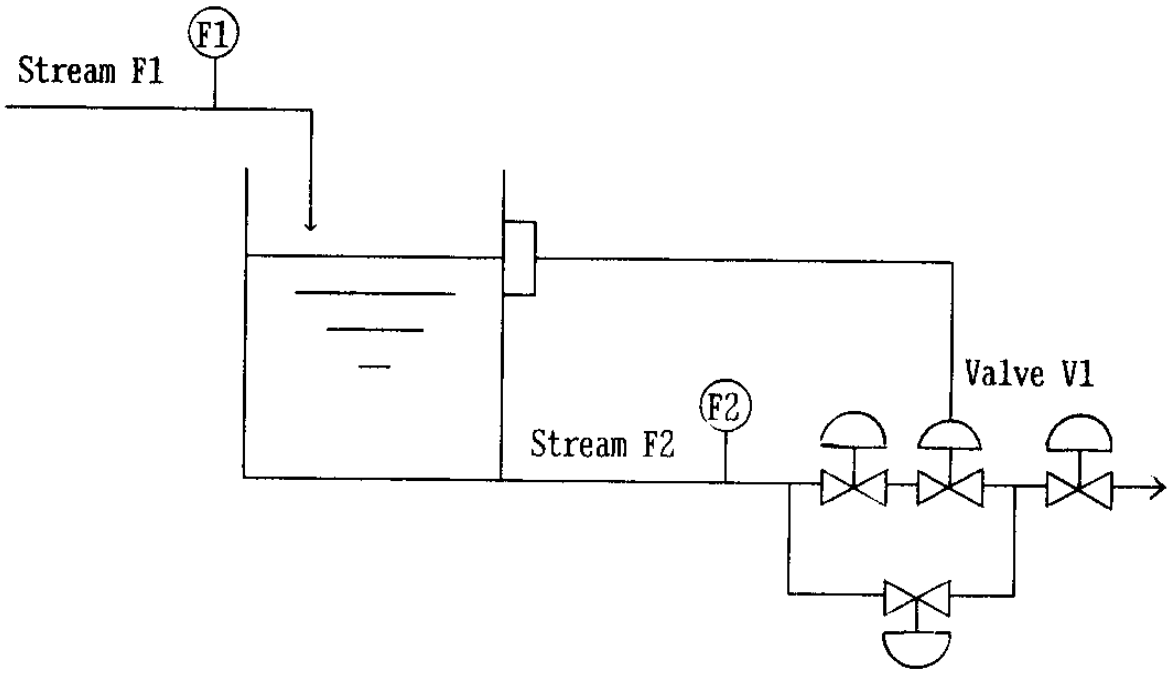


그림 4-13 . Tank unit의 구조 [51]

가 크고 복잡한 공정의 진단 시스템 구성에 하나의 블럭으로 활용될 수 있다.

그림 4-13에 보인 tank에 대하여 장치들의 가능한 상태에 따른 상황들은 다음의 표4-8과 같이 정리해 볼 수 있다. 각각의 event에 따른 상황들은 event list에 저장되며 이들은 다시 장치의 상태와 함께 진단rule 베이스로서 저장된다. Event list는 enter_event_data routine에 의해 구축되는데 사용자는 필요에 따라 event list를 검증하고 modify할수 있다. 표 4-8 의 경우 event list는 다음과 같이 스크린 상에 나타난다.

```
EVENT LIST :

EVENT e1: operation is normal
EVENT e2: there is pipe leakage between FLOW 2 and VALVE 1
EVENT e3: VALVE 1 by-pass open in error
EVENT e4: there is blockage in exit pipe line
EVENT e5: there is leak in tank
EVENT e6: throughput is abnormal
EVENT ea: it is an anomaly

(There are 7 events in the list.)

Enter event title (e1, e2, ...):
```

그림 4-13에 보인 tank에 대한 상황의 예측기능과정을 보기로 한다. 사용자는 우선 관련 장치 및 변수들의 상태를 시스템의 요구에 따라 입력시키게 된다.

표 4-8 . Tank 장치의 상태에 따른 가능한 상황 [51]

장치의 상태			events	
F1	V1	F2	title	상 황
normal	normal	normal	e1	Normal operation
normal normal normal high high high low low low	normal closed closed normal closed closed normal closed closed	high normal high high normal high high normal high	e2	Pipe leakage between F2 and V1
normal normal normal high high high low low low	normal open open normal open open normal open open	low normal low low normal low low normal low	e4	Blockage in exit line
normal high high	closed normal closed	low normal low	e5	Leak in tank
high low	open closed	high low	e6	Abnormal throughput
normal low low	open normal open	high normal high	ea	Anomaly

다음과 같이 stream 1의 상태가 low이고 valve 1의 상태는 정상이며 stream 2의 상태는 high라고 가정해 보기로 한다.

```
Enter states:
State of FLOW 1 (  normal high low ): low

State of VALVE 1 (  normal open closed ): normal

State of FLOW 2 (  normal high low ): high
```

이 경우 시스템의 응답은 스크린 상에 다음과 같은 형태로 나타난다.

```
According to RULE 20 :


If
        FLOW 1 is low and
        VALVE 1 is normal and
        FLOW 2 is high

Then
        there is pipe leakage between FLOW 2 and VALVE 1
```

다음의 예는 stream 1이 low 상태이고 valve 1은 닫힌 상태이며 stream 2가 낮게 나타났을 경우의 과정을 보인 것이다.

```
Enter states:

State of FLOW 1 (  normal high low ): low

State of VALVE 1 (  normal open closed ): closed

State of FLOW 2 (  normal high low ): low

According to RULE 27 :


If
        FLOW 1 is low and
        VALVE 1 is closed and
        FLOW 2 is low

Then
        throughput is abnormal
```

Stream 1이 normal 상태이고 valve 1은 열린 상태이며 stream 2가 정상이라면 4번째 rule에 의해 출구 line이 막힌 사실을 알아낼 수 있는데 이 과정은 다음에 보인 바와 같다.

```
Enter states:

State of FLOW 1 (  normal high low ): normal

State of VALVE 1 (  normal open closed ): open

State of FLOW 2 (  normal high low ): normal
```

According to RULE 4 :

If
FLOW 1 is normal and
VALVE 1 is open and
FLOW 2 is normal

Then
there is blockage in exit pipe line

다음은 야기된 상황의 원인들을 유추해 보기로 한다. 조업상태가 정상이면 모든 장치나 변수들의 조작도 정상이라고 할 수 있는데 이 경우 사용자가 조업이 정상인 원인을 알아보기 위해 해당되는 event의 title을 입력시키면 시스템은 그 원인들을 다음과 같이 제시하여 준다.

According to RULE 1 :

operation is normal because

FLOW 1 is normal and
VALVE 1 is normal and
FLOW 2 is normal

Press any key for more possible cases.

컴퓨터 스크린 하단에 나타나는 command는 주어진 상황의 원인이 여러가지일 때 그 각각을 제시하여 주기 위한 것이다. 예를 들어 pipe의 출구 line이 막혔을 경우 그 원인은 표4-8에 보인 바와 같이 9가지가 되는데 이는 다음과 같이 스크린 상에 제시된다.

```
According to RULE 12 :


there is blockage in exit pipe line because

        FLOW 1 is high and
        VALVE 1 is normal and
        FLOW 2 is low




Press any key for more possible cases.


According to RULE 13 :


there is blockage in exit pipe line because

        FLOW 1 is high and
        VALVE 1 is open and
        FLOW 2 is normal



Press any key for more possible cases.
```

According to RULE 15 :

there is blockage in exit pipe line because

FLOW 1 is high and
VALVE 1 is open and
FLOW 2 is low

Press any key for more possible cases.

According to RULE 21 :

there is blockage in exit pipe line because

FLOW 1 is low and
VALVE 1 is normal and
FLOW 2 is low

Press any key for more possible cases.

According to RULE 22 :

there is blockage in exit pipe line because

FLOW 1 is low and
VALVE 1 is open and
FLOW 2 is normal

Press any key for more possible cases.

According to RULE 24 :

there is blockage in exit pipe line because

FLOW 1 is low and
VALVE 1 is open and
FLOW 2 is low

Press any key for more possible cases.

According to RULE 3 :

there is blockage in exit pipe line because

FLOW 1 is normal and
VALVE 1 is normal and
FLOW 2 is low

Press any key for more possible cases.

According to RULE 4 :

there is blockage in exit pipe line because

FLOW 1 is normal and
VALVE 1 is open and
FLOW 2 is normal

Press any key for more possible cases.

According to RULE 6 :

there is blockage in exit pipe line because

FLOW 1 is normal and
VALVE 1 is open and
FLOW 2 is low

Press any key for more possible cases.

이례적인 상황(anomaly)을 야기시키는 요인들은 표 4-8 에서 알 수 있는 바와 같이 3 가지가 있다.

According to RULE 19 :

it is an anomaly because

FLOW 1 is low and
VALVE 1 is normal and
FLOW 2 is normal

Press any key for more possible cases.

According to RULE 23 :

it is an anomaly because

FLOW 1 is low and
VALVE 1 is open and
FLOW 2 is high

Press any key for more possible cases.

According to RULE 5 :

it is an anomaly because

FLOW 1 is normal and
VALVE 1 is open and
FLOW 2 is high

Press any key for more possible cases.

4. 시스템 보조함수

앞에서 언급한 rule의 운용 및 진단기능의 수행은 여러가지 보조함수들의 도움으로 가능해 진다. 이들 보조함수는 지식데이타의 입력 및 보정 보완과 저장 및 loading기능을 주로 맡아

처리한다. 이들을 요약해 보면 다음의 그림 4-14에 보인 바와 같다.

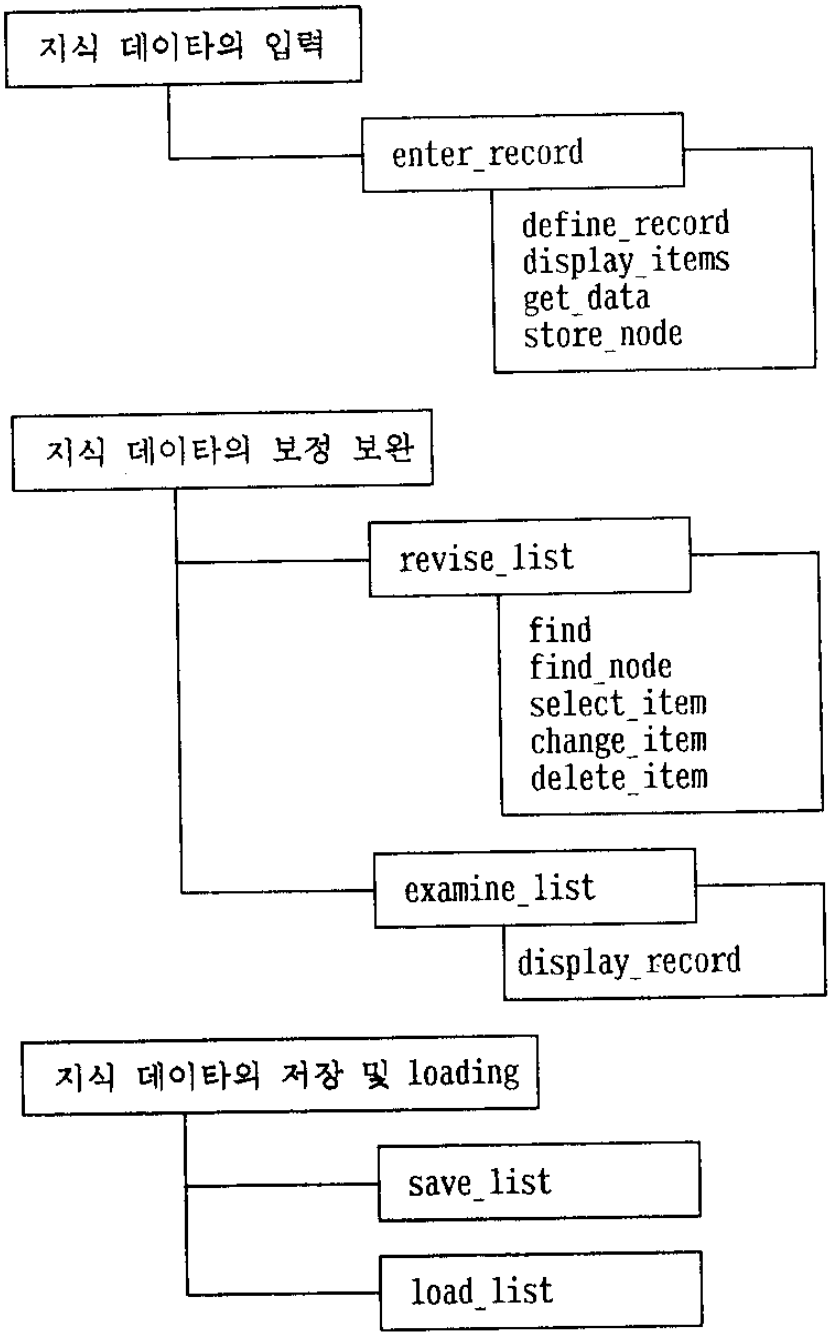


그림 4-14. 시스템 보조함수

제4절 결 론

공정 전문가 시스템의 개발에 있어서는 일차적으로 개발대상이 되는 공정을 선택한 다음 적절한 전문가 시스템 구축용 tool이나 shell을 구입하여 지식 베이스와 utility들을 정하여 주는 방식이 지금까지의 거의 일관된 개발패턴이었다. 시간과 경비문제를 고려한다면 이러한 패턴은 물론 가장 경제적인 방법이 될수 있다. 그러나 소규모 공정이나 어느 특정공정, 또는 사용자가 원하는 기능들을 요구에 맞게 충족시켜 주기 위해서는 기본적인 전문가 시스템 software 기술의 축적이 불가피하다. 최근들어 급격히 향상된 컴퓨터 hardware 기술로 인하여 조업 현장에도 소형 컴퓨터가 광범위하게 활용되기 시작하고 있으며 따라서 기존의 workstation급 컴퓨터용은 물론 소형 컴퓨터를 위한 공정 조업용 전문가 시스템의 개발 활용이 요망되고 있다. 전문가 시스템 software의 개발은 나아가 공정 인공지능 기술의 해외 예속화 탈피를 위해서도 필수적으로 수행하여야 할 과제이다.

이번의 1차년도에서는 prototype의 시스템 개발을 시도하여 보는 것이 주된 목표가운데 하나였다. 프로그래밍 언어로서 C를 사용하여 기본적인 지식 베이스의 구축 및 운용과정과 초보적인 공정진단과정을 구성하였다. 비교적 간단한 장치나 공정의 조업지식의 표현 및 운용이나 진단은 현재 구축된 시스템으로 가능함이 확인되었다. 본 연구의 2차년도에서는 제5장과 6장에서 다루고 있는 진단 및 제어 메카니즘과 몇가지의 기법들이 시스템에 조합

될 것이며 아울러 따로 구입하게 될 시스템 shell과의 비교 활용을 통해 보정과 보완작업이 계속될 것이다.

참 고 문 헌

(1) B. Khoshnevis and M.H. Chignell, "A Framework for Artificial Intelligence Applications Software Development", Computers in Industry, **Vol.6**, 363-369 (1985).

(2) P. Winter and S.M. Rosen, "Process-Engineering Database", Chem. Eng., July 7, 65-67 (1986).

(3) S. Kumara, A.L. Soyster, and R.L. Kashyap, "An Introduction to Artificial intelligence", IE, Dec., 9-20 (1986).

(4) S.J. Bailey, "Artificial intelligence in Industry: Expert Knowledge Bases in Control loops", Control Eng., Dec., 45-48 (1986).

(5) N.S. Rajaram, "Expert System Building Tools-Present Trends and Future Neds", ISA Trans., **Vol.26**, No.1, 53-55 (1987).

(6) D.V. Ward, "Issues of Liability in Expert Systems Software", ISA Trans., **Vol.26**, No.1, 57-58 (1987).

(7) K.R. O'Neal, "Roundtable Participants Discuss AI Trends and Developments", IE, 52-63, Feb. 1987.

(8) J. Reason, "Expert System Promise to Cut Critical Machine Downtime", Power, 17-24, March 1987.

(9) "Problems Below the Engineering Surface", Expert Systems

User, 18-20, March 1987.

(10) J.D. Folley, Jr. and R.J. Hritz, "Embedded AI Expert System Troubleshoots Automated Assembly", IE, 32-36, April 1987.

(11) M.A. Vonderembse and G.S. Wobser, "Steps for Implementing a Flexible Manufacturing System", IE, 38-48, April 1987.

(12) P. Falster, "Planning and Controlling Production Systems Combining Simulation and Expert Systems", Computers in Industry, **Vol.8**, 161-172 (1987).

(13) R.A. Herrod, "Artificial Intelligence: A Natural in CIM Applications", I & CS, 41-44, June 1987.

(14) G. Stephanopoulos, "The Scope of Artificial Intelligence in Plant-wide Operations", FOCAPO-87, July5-10, Utah, 1987.

(15) "Editorial. Artificial Intelligence: The Emergence of a Formal Basis for Engineering", IEE Proc., **Vol.134**, Pt. D, No.4, 217 (1987).

(16) I. Aleksander and H. Morton, "Artificial Intelligence: An Engineering Perspective", IEE Proc., **Vol.134**, Pt. D, No.4, 218-223 (1987).

(17) P. Jackson, "Review of Knowledge-Representation Tools and Techniques", IEE Proc., **Vol.134**, Pt. D, No.4, 224-230 (1987).

(18) D. Rang, J. Bigham, and E.H. Mamdani, "Reasoning with

Uncertain Information", IEE Proc., **Vol.134**, Pt. D, No.4, 231-237 (1987).

(19) M.E. Bennett, "Real-Time Continuous AI Systems", IEE proc., **Vol.134**, Pt. D, No.4, 272-277 (1987).

(20) W.B. Gevarter, "Introduction to Artificial Intelligence", Chem. Eng. Prog., 21-37, Sept. 1987.

(21) M.F. Russo and R.L. Peskin, "Knowledge-Based Systems for the Engineer", Chem. Eng. Prog., 38-43, Sept. 1987.

(22) G. Stephanopoulos, "The Future of Expert Systems in Chemical Engineering", Chem. Eng. Prog., 44-51, Sept. 1987.

(23) J.P. Bartl, "Current Research in Computer Science at MIT", Chem. Eng. Prog., 60-75, Sept. 1987.

(24) T. Gupta and B.K. Ghosh, "A Survey of Expert Systems in Manufacturing and Process Planning", Computers in Industry, **Vol.11**, 195-204 (1988).

(25) F.J. Bartos, "AI Now More Realistically At Work in Industry", Control Eng., July, 90-93 (1989).

(26) W.K. Chan and R.G.H. Prince, "Heuristic Evolutionary Synthesis with Non-Sharp Separators", Comput. Chem. Eng., **Vol.13**, No.11/12. 1207-1219 (1989)

(27) A. Betta and D.A. Linkens, "Intelligent Knowledge-Based System for Dynamic System identification", IEE Proc.,

Vol.137, Pt. D, No.1, 1-12 (1990).

(28) M.V. O'Neill, "Implementing a Chemical process Plant Expert System", ISA Trans., **Vol.26**, No.1, 19-25 (1987).

(29) R.A. Herrod and J. Rickel, "An Expert System for Simulating a Glass Annealing Process: AI Applications in the Glass Industry", ISA Trans., **Vol.25**, No.4, 23-29 (1986).

(30) K. Lien, G. Suzuki, and A.W. Westerberg, "The Role of Expert Systems Technology in Design", Chem. Eng. Sci., **Vol.42**, No.5, 1049-1071 (1987).

(31) R. Banares-Alcantara, A.W. Westerberg, E.I. Ko, and M.D. Rychener, "DECADE-A Hybrid Expert System for Catalyst Selection-I. Expert System Consideration", Comput. Chem. Eng., **Vol.11**, No.3, 265-277 (1987).

(32) R.R. Leitch, "Modelling of Complex Dynamic Systems", IEE Proc., **Vol.134**, Pt. D, No.4, 245-250 (1987)

(33) R.J. Paul and G.I. Doukidis, "Artificial Intelligence Aids in Discrete-Event Digital Simulation Modelling", IEE Proc., **Vol.134**, Pt. D, No.4, 278-286 (1987).

(34) G.A. Bekey, "Knowledge Based Systems in Modeling, Simulation, and Identification", IFAC, 69-74, 1988.

(35) F. Gomide, W.C. Amaral, L.V.R. Arruda, G. Favier, A.S. Barbara, and W. Fontanini, "Expert System Identification: The Supervisory Approach", IFAC, 1691-1696, 1988.

(36) T.X. Lin, L.B. Lan, and Z.Y. Jong, "The Automating Identification System with Intelligence", IFAC, 1708-1712, 1988.

(37) M. Haest, G. Bastin, M. Gevers, and V. Wertz, "An Expert Workstation for System Identification", IFAC, 1990-1995, 1988.

(38) R.L. Kirkwood, M.H. Locke, and J.M. Douglas, "A Prototype Expert System for Synthesizing Chemical Process Flowsheets", Comput. Chem. Eng., **Vol.12**, No.4, 329-343 (1988).

(39) J.C. Hoskins and D.M. Himmelblau, "Artificial Neural] Network Models of Knowledge Representation in Chemical Engineering", Comput. Chem. Eng., **Vol..12**, No.9/10, 881-890 (1988).

(40) R. Banares-Alcantara, E.I. Ko, A.W. Westerberg, and M.D. Rychener, "DECADE-A Hybride Expert System for Catalyst Selection-II. Final Architecture and Results", Comput. Chem. Eng., **Vol.12**, No.9/10, 923-938 (1988).

(41) L. Beltramini and R.L. Motard, "KNOD-A Knowledge Based Approach for Process Design", Comput. Chem. Eng., **Vol.12**, No.9/10, 939-959 (1988).

(42) D.R. Myers, J.F. Davis, and D.J. Herman, "A Task-Oriented Approach to Knowledge-Based Systems for Process Engineering Design", Comput. Chem. Eng., **Vol.12**, No.9/10, 959-971 (1988).

(43) I.P. Popchev and N.P. Zlatareva, "Intelligent Problem-Independent Tool for Plausible Inference", IFAC, 372-277, 1987.

(44) H.P. Wang and R.A. Wysk, "Intelligent Reasoning for Process Planning", Computers in Industry, **Vol.8**, 293-309 (1987).

(45) Y.W. Huang and L.T. Fan, "Designing and Object-Relation Hybrid Database for Chemical Process Engineering", Comput. Chem. Eng., **Vol.12**, No.9/10, 973-983 (1988).

(46) H.D. Engelmann, H.H. Erdmann, R. Funder, and K.H. Simmrock, "The Solving of Complex Process Synthesis Problems Using Distributed Expert System", Comput. Chem. Eng., **Vol.13**, No.4/5, 459-465 (1989).

(47) E. Lahdenpera, E. Korhonen, and L. Nystrom, "An Expert System for Selection of Solid-Liquid Separation Equipment", Comput. Chem. Eng., **Vol.13**, No.4/5, 467-474 (1989).

(48) J. Lapointe, B. Marcos, M. Veillette, and G. Laflamme, "BIOEXPERT-An Expert System for Wastewater Treatment Process Diagnosis", Comput. Chem. Eng., **Vol.13**, No.6, 619-630 (1989).

(49) B. Chen, J. Shen, Q. Sun and S. Hu, "Development of an Expert System for Synthesis of Heat Exchanger Networks", Comput. Chem. Eng., **Vol.13**, No.11/12, 1221-1227 (1989)

(50) M. Hofmeister, L. Halasz and D.W.T. Rippin, "Knowledge-Based Tools for Batch Processing Systems", Comput. Chem. Eng., **Vol.13**, No.11/12, 1255-1261 (1989).

(51) D.M. Himmelblau, Fault Detection and Diagnosis in Chemical and Petrochemical Processes, 1978, Elsevier Company.

(52) G.F. Luger and W.A. Stubblefield, Artificial Intelligence and the Design of Expert System, 1989, Benjamin/Cummings, Inc.

제 5 장 모델링 및 모델인식에 의한 이상진단

제 1 절 서 론

화학공정의 이상진단 방법 중에서 수학적 모델에 의한 접근법은 자동제어의 한 분야로 주요한 연구대상이 되고 있다. 이러한 접근법은 비교적 문제가 잘 정의되어 있고, 이상 (fault) 의 수가 제한되어 있으며 이상의 조기 검색이 중요한 몇가지 화학공정에서는 많은 연구가 진행되고 있다.[1] 모델링 및 추정 (estimation)에 의해 공정의 이상을 탐지하는 이 방법은 수학적 공정모델이나 신호모델 (signal model) 의 구성과 응용이 가능한 공정범위에서 신호의 예측 혹은 측정이 불가능한 공정상태변수나 매개변수 혹은 공정의 특성변수등을 추정함으로써 이루어진다. 본 연구에서는 유체역학적으로 수학적 모델의 구성이 용이하며, 이상 (fatlt) 이 scaling 혹은 paraffining으로 인한 도관내경의 감소와 유출의 두 가지로 제한되어 있고, 유출의 경우 이상 발생시 경제적인 손실은 물론 환경오염이 심각한 파이프라인 진단에 관한 문제를 다루고자 한다. 이 문제는 장거리에 걸쳐있는 송유관, 도시가스 및 상하수도 등에 관련된 것으로 중요성과 효용성이 매우 큰 문제이다.

일반적으로 공정의 관리는 공정감시 혹은 이상탐지, 이상진단, 이상평가 및 결정단계로 구성된다. 이상평가는 이상의 영향을 단계적으로 분석하거나 이상트리를 이용한다. 결정단계는 이상의 영향을

분석한 후 공정을 멈추고 이상을 제거하거나 혹은 운전조건을 바꾸어 줄 것인지를 결정하는 단계로 이 두 과정은 공정종속성을 갖는다. 먼저 공정관리의 핵심인 공정의 이상 탐지방법과 진단방법을 참고문헌 〔1〕, 〔2〕, 〔3〕등에 의거하여 살펴보기로 한다.

제 2절 공정의 이상탐지

종래의 공정 이상탐지는 공정 입출력만을 측정하여 이 값들이 정해준 한계를 초과하는 지를 판단하는 방법이었으나, 이러한 방법은 2장 2절에서 살펴 본 바와 같이 경보발단점 (alarm threshold) 에서의 민감도 문제와 이상에 의해 공정 출력이 상당히 영향을 받은 후에 감지된다. 따라서 보다 일찍 공정의 이상을 탐지하기 위하여 측정변수 외에 미측정변수를 이용한 방법들이 제시되고 있다.

1. 이상탐지방법

이상탐지를 위해 이용되는 변수에 따라 다음과 같은 4가지 방법으로 나눌 수 있다. 먼저 측정가능한 공정 신호인 입력 U(t)와 출력 Y(t)만을 이용하는 방법이다. 이 방법 중에서 널리 이용되는 것이 경계 (limit) 와 경향 (trend) 을 확인하는 것으로, 경계확인은 스팀 발생기의 수위감시 등에 이용되는 것으로 출력의 최소와 최대 값을 정하고 이 범위를 벗어나면 경보를 울린다. 경향확인은 출력의 경향 즉 변화율에 최대, 최소의 범위를 설정하는 방

법이며, 경계확인과 함께 이용되기도 한다. 이와 같은 방법은 경보 발단점에서의 민감도 문제가 있으므로 이를 해결하기 위하여 신호의 예측이나 발단점을 초과한 시간의 예측 등을 함께 이용한다. 이를 위해서는 결정론적 (deterministic) 신호의 수학적 모델이나, 결정론적 공정과 추계적 (stochastic) 신호모델 혹은 추계적 신호들이 요구된다. 이러한 신호모델의 변수추정은 다단예측 (multistep prediction) 과 같은 기법이 필요하며, 이렇게 구해지는 공정출력의 예측치에 경계 확인 방법을 이용한다. 이외의 측정 가능한 신호를 이용하는 방법으로는 신호해석법 (analysis of signals) 이 있다. 이는 공정신호가 상대적으로 큰 양의 저주파 신호와 작은 양의 고주파 신호의 합성으로 되어 있다고 보고, 고주파 신호모델을 인식 (identification) 하는 방법이다. 이와같은 비매개변수 (nonparametric) 신호모델에 이용되는 것으로는 autocorrelation function, spectral density, 진동분석법 등이 있다.

다음으로 측정 불가능한 미지수를 이용하여 공정이상을 탐지하는 방법은 이상탐지에 이용하는 미지수에 따라 상태변수 X(t)를 이용하는 방법, 공정 매개변수 θ(t)를 이용하는 방법 및 공정의 효율과 같은 특성치 η(t)를 이용하는 방법들로 나눌 수 있다.

2. 상태변수를 이용한 이상탐지

상태변수를 이용하는 방법은 공정모델과 측정신호를 이용하여 상태변수를 추정하는 방법이다. 정적 (static) 공정의 경우는 공정모델 만으로 추정이 가능하지만, 일반적인 동적 계에 대해서는 공

정모델 외에 이상결정 (fault decision) 을 위한 휠터링이나 GLR (generalized likelihood ratio) 방법이 요구된다. 공정 입출력과 상태변수와의 관계를 한 운전조건에서 선형화하면 다음과 같은 상태공간 방정식 (state space equation) 을 얻을 수 있다.

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

여기서 x, y 및 u는 X, Y 및 U의 변화량이다. 위 식에서 상태변수를 측정 입출력으로 부터 계산하기 위해서는 결정론적인 경우에는 상태변수의 관측자 (observer) 로 추계적인 경우에는 휠터가 다음과 같이 구성된다.

$$\hat{\dot{x}}(t) = A\hat{x}(t) + Bu(t) + H[\, y(t) - C\hat{x}(t) \,]$$

동적계의 상태변수나 측정변수에 돌발적인 이상이 생기는 것에 관해서는 처음 Willsky[4] 에 의해 탐지방법이 제시되었다. 공정이나 공정잡음 혹은 구동부 (actuator) 에 갑작스런 변화가 생기는 것은 상태식에서 $\nu(t)$로 표현한다.

$$\dot{x}(t) = Ax(t) + Bu(t) + Fv(t) + \nu(t)$$

검출단 (sensor) 에서의 변화는 $\mu(t)$로 나타낸다.

$$y(t) = Cx(t) + n(t) + \mu(t)$$

위 식들에서 v(t)와 n(t)는 각각 공정잡음과 측정잡음이다. $\nu(t)$와 $\mu(t)$를 계단함수나 delta impluse 함수로 가정하면, 상태변수 혹은 측정치에 이상이 생긴 것을 모델링할 수 있다. 이상의 결정은 다음과 같은 특별한 시험방법들로 행한다.

1) 'Fault sensitive filter's where the feedback matrix H is

chosen so that particular fault modes manifest themselves as residuals in a fixed direction or in a fixed plane, a deterministic approach.

2) A Whiteness and a chi-squared test of the residuals of the normal Kalman-filter.

3) A finite bank of Kalman-filters with a standard multiple hypothesis testing that the systems most likely respond to one of the assumed models with hypothesized faults included.

4) A generalized likelihood ratio (GLR) test which results in a correlation of the observed residuals with the precomputed filter responses due to certain faults.

위와 같은 상태변수를 이용한 공정관리의 계통도는 그림5-1과 같고 이 방법은 그림에서 A,B,C의 공정 매개변수를 가진 전이 행렬을 정확히 알고 있어야 적용이 가능하다.

(1) GLR방법에 의한 이상탐지

상태변수 추정에 의한 공정의 이상탐지 중에서 위의 (4)번에 해당하는 GLR 방법을 이용한 이상결정에 관하여 Tanaka[2]의 연구내용을 살펴보기로 한다. 선형계의 이산시간대 모델은

$$x(k+1) = Ax(k) + Bu(k) + \Gamma w(k)$$

$$y(k) = Hx(k) + v(k)$$

여기서, w와 v는 zero mean과 각각 covariance Q, R을 갖는 상

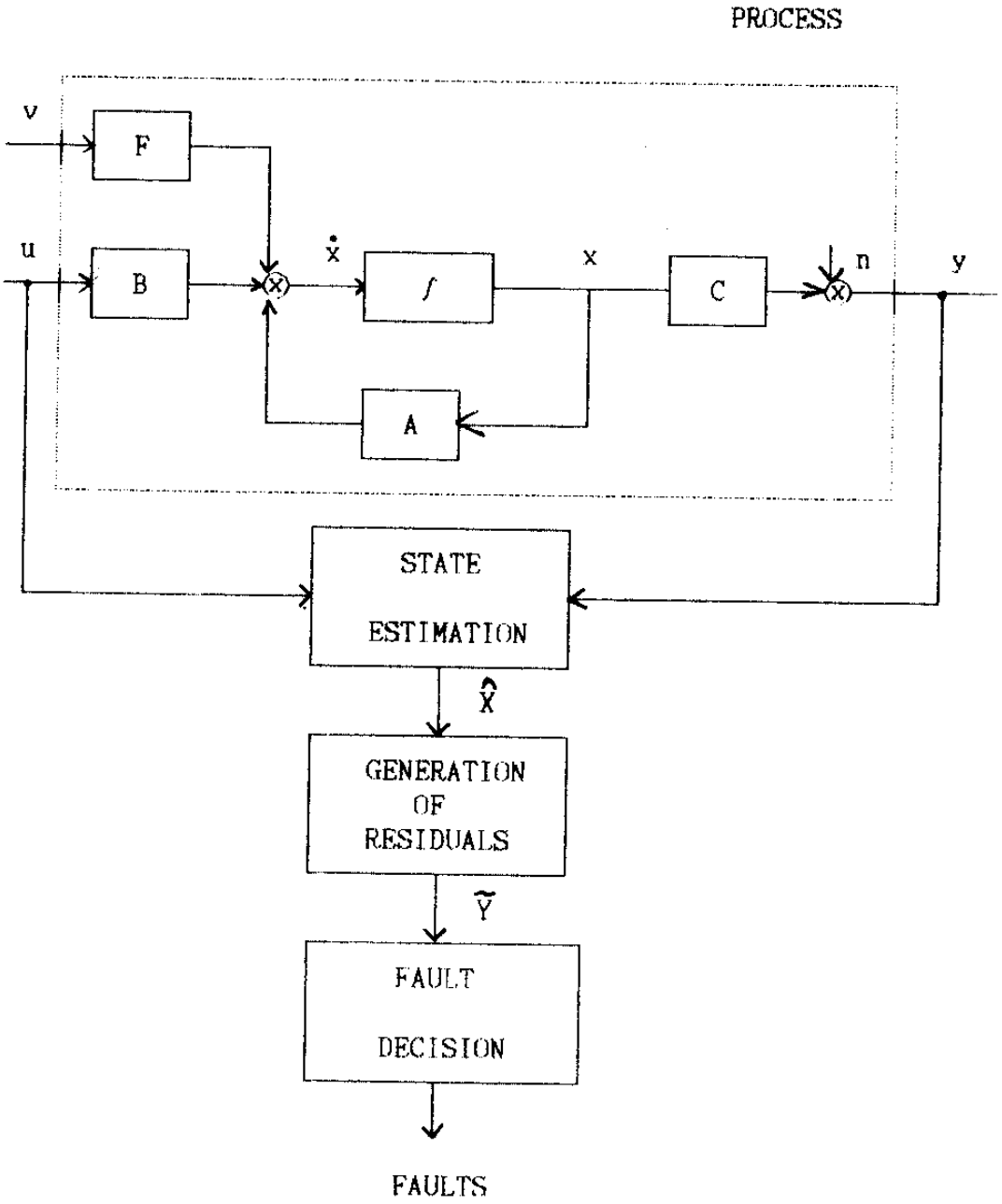


그림 5-1. 상태변수추정에 의한 이상탐지

호 독립적인 백색 가우시안 잡음이다. 공정이나 검출단에 비이상성, 혹은 고장이 생기면 일반적으로 매개변수 A, B 및 H에 변화가 생긴다. 이것을 상태식에 이상이 생긴 시간부터 고려하면 다음과 같이 표현된다.

$$x(k+1) = Ax(k) + Bu(k) + \Gamma w(k) + \sigma(k+1, \theta_d) f_d(k+1)$$

$$y(k) = Hx(k) + v(k) + \sigma(k, \theta_s) f_s(k)$$

여기서 f는 비정상성을 나타내는 미지의 벡터이고, $\sigma(k, \theta)$는 단위 계단함수이다. 즉 이 방법은 step-hypothesized GLR이 된다.

시스템 방정식의 최적 상태변수 추정은 Kalman-filter로 구해진다.

$$\hat{x}(k/k-1) = A\hat{x}(k-1/k-1) + Bu(k-1)$$

$$\hat{x}(k/k) = \hat{x}(k/k-1) + K(k)\,\gamma(k)$$

$$\gamma(k) = \gamma(k) - H\hat{x}(k/k-1)$$

$$P(k/k-1) = AP(k-1/k-1)A^T + \Gamma Q \Gamma^T$$

$$P(k/k) = P(k/k) - K(k)HP(k/k-1)$$

where, $K(k) = P(k/k-1)\, H^T V^{-1}(k)$

$$V(k) = HP(k/k-1)H^T + R$$

위 식에서 γ는 innovation항으로 이 값은 정상운전 상황에서는 zero mean과 covariance V(k)를 갖는다. 이때의 값을 $\gamma^{\circ}(k)$라 두면 시간 θ에서 이상이 생기면 이 영향은 비정상함수 $f(k)\sigma(k, \theta)$를 통하여 innovation γ에 $\triangle\gamma$만큼 나타난다. 따라서

$$\gamma(k) = \begin{cases} \gamma^{\circ}(k) & ; \text{no failure} \\ \triangle\gamma(k, \theta) + \gamma^{\circ}(k) & ; \text{failure} \end{cases}$$

where,

$$\triangle\gamma(k,\theta) = G^{\circ}(k,\theta)\, f(k)$$

$$= \sum_{j=\theta}^{k} G^{\circ}(k,j) f(j)$$

여기서 $G^{\circ}(k,\theta)$는 동적 점프에 대한 이상신호 행렬이다. 공정에 이상이 생기면 innovation 값이 변하게 되므로 이상이 없는 경우의 값에서의 변화를 인식하면 이상탐지가 가능하다. 두 innovation 값의 차이를 계산하는 방법으로 여기서는 Kullback이 제시한 분별자 (discriminating measure)인 divergence를 이용하였다. 즉 두 신호 H_0과 H_1의 divergence measure는

$$D(H_0 ; H_1) = \sum_{j=\theta}^{k} \triangle\gamma^{T}(j;\theta) V^{-1}(j) \triangle\gamma(j;\theta)$$

위 divergence는 양수이며, 이 양이 클수록 관측에 의해 분별이 더 용이하다. 이와 같은 일반적인 GLR 방법은 동적계에 대하여 빠른 이상검색 성능을 갖고 있으나, 발생할 여러가지 가설을 미리 세워야 하는 단점이 있다. 이 논문[2]에서는 이러한 방법을 이용한 이상 탐지의 검색성 (detectability)에 관하여 논하고, 분석을 통하여 weakly-diagnosable-space가 존재하는 것을 보이고 이의 해결을 위하여 두가지 개선된 GLR 방법을 제시하였다. 한 가지는 탐지시간이 길어지는 것을 감수하고, 오경보를 발생할 가능성을 줄여주는 weighted step-hypothesized GLR 방법이고, 다른 하나는 계의 상태와 제어입력을 이용하여 bias 대신에 다른 양을 추정하는 GLR 방법 (detection method Ⅱ)이다. 2차 계에 대하여 모사한 결과가 표 5－1과 같다. 관측신호를 이용한 detection method Ⅱ가 발단점

도 작고 올바른 경보수도 많음을 볼 수 있다. 이에 비하여 Chi-squared 방법은 발단점도 크고 경보수도 작아 성능이 뒤짐을 알 수 있다.

Table 1. The number of correct alarms for each method

Method	Threshold ε	Number of correct alarms
Detection Method Ⅱ (using observation signals)	8.4	91
Detection Method Ⅱ (using input sequence)	10.4	81
Step-hypothesized GLR method	10.0	61
Chi-squared method	44.0	74

(2) SPRT 방법에 의한 이상탐지

Innovation 크기의 변화를 이용한 이상탐지 방법은 단계적 확율비조사 (SPRT, sequential probability ratio test) 가 있으며, 여기서는 SPRT를 개선한 K. Uosaki[5] 의 연구를 살펴보기로 한다. Wald에 의해 1947년에 제시된 SPRT는 한 모드 (정상) 에 대하여 일정한 크기의 변화가 일어난 다른 모드 (이상) 를 구별하는 binary decision을 하는 데는 평균적으로 최소의 샘플 수를 요구하기 때문에 최적의 것으로 평가되어 왔다. 그러나 Wald의 방법은 관측공정에 대하여 모드들의 변화양태 (transient) 가 없는 것으로 가정하므로, 미지의 시간에 정상에서 이상으로 변화하는 transient가 있는 공정진단용으로는 부적합하며 따라서 이방법을 이

용한 이상진단에는 많은 시간 지연이 따르게 된다. 이 문제를 해결하기 위하여 위에서 살펴본 것과 같은 likelihood ratio function(LLR)의 논리를 이용한 방법[2]과 여기서 살펴보고자 하는 후향 SPRT방법들이 개발되었다. 먼저 이상탐지와 SPRT와의 관계를 알아보기 위하여 앞에서 예로든 선형계에 대하여 비이상상태에서 normalized innovation sequence $\{n_k\}$는 unit variance를 갖는 zero mean white Gaussian process로 다음과 같이 정의된다.

$$n_k = V(k)^{-1/2}\ \gamma(k)$$

이것을 정상모드로 취하고, 계에 이상이 생기면 공정변수가 변하고 따라서 innovation process는 정상모드와는 다르게 된다. 즉

$$E\{n_n\} \neq 0$$

$$Var\{n_n\} \neq 1$$

그러나 실제로 공정에 이상이 생기면 innovation의 평균 보다는 편차(variance)가 더 많이 변하므로 문제를 단순화하기 위하여 이상모드는 정상모드에서 편차의 증가로만 구분된다고 가정한다. 따라서 이상의 진단은 다음의 두 가설 중 어느것이 옳은 지를 판단하는 문제로 바뀐다.

정상모드 : $(H_0) : E\{n_k\} = 0$, $Var\{n_k\} = 1$,

이상모드 : $(H_1) : E\{n_k\} = 0$, $Var\{n_k\} = \triangle^2 > 1$,

이와 같은 고전적인 SPRT를 이용한 두 가설에 대한 진위를 조사하기 위해서는 joint likelihood ratio function(LLR)을 계산해야 한다.

$$\lambda_k = \log \frac{p(n_0, n_1, \cdots, n_k ; H_1)}{p(n_0, n_1, \cdots, n_k ; H_0)}$$

그리고 오경보(false alarm)와 실경보(miss alarm)에 대한 특정한 오차확율로 부터 두가지의 발단점 (threshold) 값인 A*와 B*(A* > 0 > B*)를 비교한다.

$$A^* = \log \frac{1-\beta}{\alpha}$$

$$B^* = \log \frac{\beta}{1-\alpha}$$

LLR이 발단점 A*를 초과하거나 B* 보다 작으면 각각 가설 H_1 (이상모드)과 H_0 (정상모드)을 옳은 것으로 판정하고 관측을 끝낸다. 그렇지 않은 경우에는 LLR을 다음과 같이 회귀적으로 계산하며 새로운 자료가 입력될 때마다 조사를 수행한다.

$$\lambda_k = \lambda_{k-1} + \log \frac{p(n_k ; H_1)}{p(n_k ; H_0)}$$

$$= \lambda_{k-1} + \frac{\triangle^2 - 1}{2} n_k^2 - \log \triangle$$

이상과 같은 고전적인 Wald의 SPRT를 이용한 공정의 이상탐지는 조사의 시작단계에 시스템이 이상이나 정상모드에 있으면, 특정한 오차확률에 대하여 결정하는 데 최소의 시간을 소모한다. 그러나, 여기서 고려하는 공정의 이상은 초기에는 정상모드로 있다가 관측하는 동안의 임의의 시간에 변화양태를 갖으며 이상모드로 전환되기 때문에 이 고전적인 방법을 이용하면, 공정이 정상모드인 경우는 LLR이 평균에서 음의 방향으로 표류하고, 이상모드에서는 양의 방

향으로 표류하게 된다. 이러한 경우는 탐지에 시간지연이 많으며 또한 이상진단에서는 SPRT와는 달리 이상모드 만을 탐지하면 된다. 후향 SPRT는 이러한 문제의 해결을 위해 SPRT를 개선한 것으로, 즉 시간 i에서 이상이 생기고 따라서 normalized innovation process의 편차가 변하는 경우에

정상모드 : $(H_0) : E\{n_i\} = 0,\ Var\{n_i\} = 1,$

이상모드 : $(H_1) : E\{n_i\} = 0,\ Var\{n_i\} = \triangle^2 \times 1,$

$$i = \theta,\ \theta + 1,\ \cdots\cdots$$

이것을 $n > \theta$에 대하여 다시 쓰면,

정상모드 : $(H_0) : E\{n_{n-i+1}\} = 0, Var\{n_{n-i+1}\} = 1,$

이상모드 : $(H_1) : E\{n_{n-i+1}\} = 0, Var\{n_{n-i+1}\} = \triangle^2 \times 1,$

$$i = 1, 2, \cdots, n - \theta + 1$$

후향 LLR을 현재 관측치로 부터 과거 관측치로 역방향으로 likelihood ratio function의 논리를 정의하면,

$$\lambda^B{}_{n,k} = \log \frac{p(n_n, n_{n-1}, \cdots, n_{n-k}; H_1)}{p(n_n, n_{n-1}, \cdots, n_{n-k}; H_0)}$$

$$= \frac{\triangle^2 - 1}{2\triangle^2} \sum_{i=n-k+1}^{n} \left(n_i{}^2 - \frac{\triangle^2 - 1}{2\triangle^2} \log \triangle\right)$$

$$, k = 1, 2, \cdots, n$$

후향 SPRT는 이 후향 LLR로 SPRT에서 처럼 위의 가설을 조사하는 것으로 정의할 수 있다. 이 후향 SPRT는 이상모드에 해당하는 가설 H_1에 대하여 normalized innovation의 편차가 초기에 1이 아니라 정상모드에서 누적된 음의 값을 갖으므로 시간

지연이 생기지 않는다. 후향 LLR을 고전적인 LLR로 나타내면

$$\lambda^{B}{}_{n,k} = \frac{\triangle^2 - 1}{2\triangle^2} \sum_{i=t}^{n} \left(n_i{}^2 - \frac{\triangle^2 - 1}{2\triangle^2} \log \triangle \right)$$

$$- \frac{\triangle^2 - 1}{2\triangle^2} \sum_{i=1}^{n-k} \left(n_i{}^2 - \frac{\triangle^2 - 1}{2\triangle^2} \log \triangle \right)$$

$$= \lambda_n - \lambda_{n-k}, \qquad k = 1, 2, \cdots, n$$

또한 공정의 이상 만을 탐지하는 경우에는 결정규칙 (decision rule)을 다음과 같이 정의할 수 있다.

"If backward LLR $\lambda^{B}{}_{n,k} > K$ for some $k = 1, 2, \cdots, n$ where K is a suitable constant, we terminate observation with acceptance of the hypothesis that system is in the failure mode. Other - wise, we continue observation as system is not likely in the failure mode."

또한 이 결정규칙을 다른 방법으로 표현하면, "만약 $S_0 = 0$ 이고,

$$S_n = \max \left(0, \frac{\triangle^2 - 1}{2} n_n{}^2 - \log \triangle + S_{n-1} \right) > K$$

이 만족되면, 이 시스템은 이상이 있다."

후향 SPRT를 이용하기 위해서는 적당한 reference value △와 결정의 경계치 K를 설정해야 하며, 이를 위해서는 이 후향 SPRT를 이용한 탐지방법의 평균탐지시간 (MDT, mean detection time)을 고려해야 한다. 여기서 MDT는 innovation편차가 $\triangle^2$로 변했을 때 S_n이 결정경계 K를 초과할 때 까지의 평균관측 횟수로 정의된다. 이 MDT를 K와 △로 나타내기 위하여 먼저, $(-\infty, \infty)$를 $W = K(N-2)$를 이용하여 $(-\infty, 0]$, $(0, W]$, $(W, 2W]$,

…, ((N-3)W,k] 및 (K,∞) 등의 N개의 구간으로 나누고, stationaty finite Marcov chain의 states를 $E_1 = (-\infty, 0]$, $E_i = ((i-2)W,(i-1)W]$, $(i = 2, \cdots, N-1)$ 및 $E_N = (k, \infty)$으로 정의한다. P를 $p_{ij} = Pr(S_{n-1} \in E_i$ 과 $S_n \in E_j, i, j = 1, \cdots, N)$ 인 Markovchain의 변이확률함수로 두면

$$p_{11} = Pr\{z-h \leq 0\} = F(Dh)$$

$$p_{1i} = Pr\{(i-2)W \leq z-h < (i-1)W\}$$

$$= F(D(i-1)W = h)) - F((i-2)W+h)), \quad (i = 2, \cdots, N-1)$$

$$p_{1N} = Pr\{z-h > K\} = 1 - F(D(K+h))$$

$$p_{i1} = Pr\{z-h \leq (-i+3/2)W\} = F(D((-i+3/2)W+h)),$$

$$(i=2, \cdots, N-1)$$

$$p_{ij} = Pr\{(j-i-1/2)W \leq z-h < (j-i+1/2)W\}$$

$$= F(D((j-i+1/2)W+h)) - F(D((j-i+1/2)W+h)),$$

$$(i, j=2, \cdots, N-1)$$

$$p_{iN} = Pr\{z-h > (N-i-1/2)W\} = 1-F(D(N-i-1/2)W+h))$$

$$(i=2, \cdots, N-1)$$

$$p_{Ni} = 0, \qquad (i=2, \cdots, N-1)$$

where,

$$z = \frac{\triangle^2 - 1}{2\triangle^2}, \quad h = \log \triangle, \quad D = \frac{2\triangle^2}{(\triangle^2 - 1)\triangle_n^2}$$

and $F(x)$: cumulative distribution function of chi-squared distribution with degree of freedom 1, $= 2\Phi(x^{1/2}) - 1$, with standard normal distribution Φ.

따라서 변이확률함수 P는

$$P = \begin{vmatrix} R & p \\ 0 & 1 \end{vmatrix}$$

E_i 상태에서 출발하는 평균흡수시간 벡타는

$$\mu = (I - R)^{-1} \xi ,$$

where, I : unit matrix and

$$\xi = (1 , \cdots, 1)^T$$

공정이 초기에 정상모드에서 출발하면, N-discrete state Marcov chain 근사에 의하여 μ의 첫번째 원소는 MDT가 된다. 상태의 수가 많은 경우에 MDT는 다음과 같이 가정할 수 있다.

$$\mu(N) = \mu(\infty) + \frac{A}{N}$$

여기서 $\mu(\infty)$는 여러 N값에 대하여 $\mu(N)$로 부터 최소자승법을 이용하면 구할 수 있다. 이 값들을 참조하여 reference value △와 결정 경계치 K를 얻는다. 한 예로 reference value △를 납득할 수 있을 정도로 큰 innovation 편차의 변화로 주고, 정상모드 ($\triangle_n^2$)의 MDT값을 이용하여 결정경계 K를 구한다.

3. 공정 매개변수를 이용한 이상탐지

공정모델의 매개변수는 공정 입출력 관계를 수학적으로 표현하면 상수 혹은 시간에 따라 변하는 항이다. 정적 공정 모델에서는 아래 식과 같이 다항식의 계수로 표현된다.

$$Y(U) = \beta_0 + \beta_1 U + \beta_2 U^2 + \cdots$$

Lumped 매개변수를 갖는 동적 모델은 한 운전조건에서 선형화하면 다음과 같은 미분식으로 표현된다.

$$y(t)+ a_1 y^{(1)}(t)+ a_2 y^{(2)}(t)+\cdots+ a_n y^{(n)}(t)=$$

$$b_0 u(t)+ b_1 u^{(1)}(t)+ b_2 u^{(2)}(t)+\cdots+ b_m^{(m)}\ u(t)$$

이와 같은 공정모델의 매개변수 $\theta^T = [\ \beta_0\ ,\ \beta_1\ ,\ \beta_2\ ,\ \cdots\ ]$, 혹은 $\theta^T = [\ a_1 \cdots a_n\ :\ b_1 \cdots b_m\ ]$는 대부분 길이, 질량, 점도 및 저항 등의 물리적 공정계수와 관련되어 있다. 따라서 고장의 이상을 나타내는 물리적 공정계수가 직접 측정할 수 없는 경우에는 이 값의 변화를 공정모델의 매개변수의 변화로 바꾸어 주는 과정이 필요하게 된다. 미측정 매개변수를 이용한 공정의 이상검색 구성도는 그림 5-2 와 같고, 그 과정은 다음과 같다.

1) 이론적 모델링에 의하여 측정변수들 간의 공정 수식을 구성한다.

   $Y(t)= f\ \{\ U(t),\ \theta\ \}$

2) 모델 매개변수 θ_i 와 공정의 물리적 계수 p_j 와의 관계를 결정한다. $\theta = f(p)$

3) 측정신호 U(t)와 Y(t)를 이용하여 공정모델의 매개변수 θ 를 추정한다.

4) 공정계수를 계산하고 이의 변화량 $\triangle p_j$ 를 결정한다.

   $p = f^{-1}(\theta)$

5) 공정계수의 변화량과 공정의 이상과의 관계를 미리 설정해 둔 자료를 이용하여 이상을 탐지한다.

이 방법은 2)번 과정이 필요하므로 잘 정의된 공정에서만 이용이

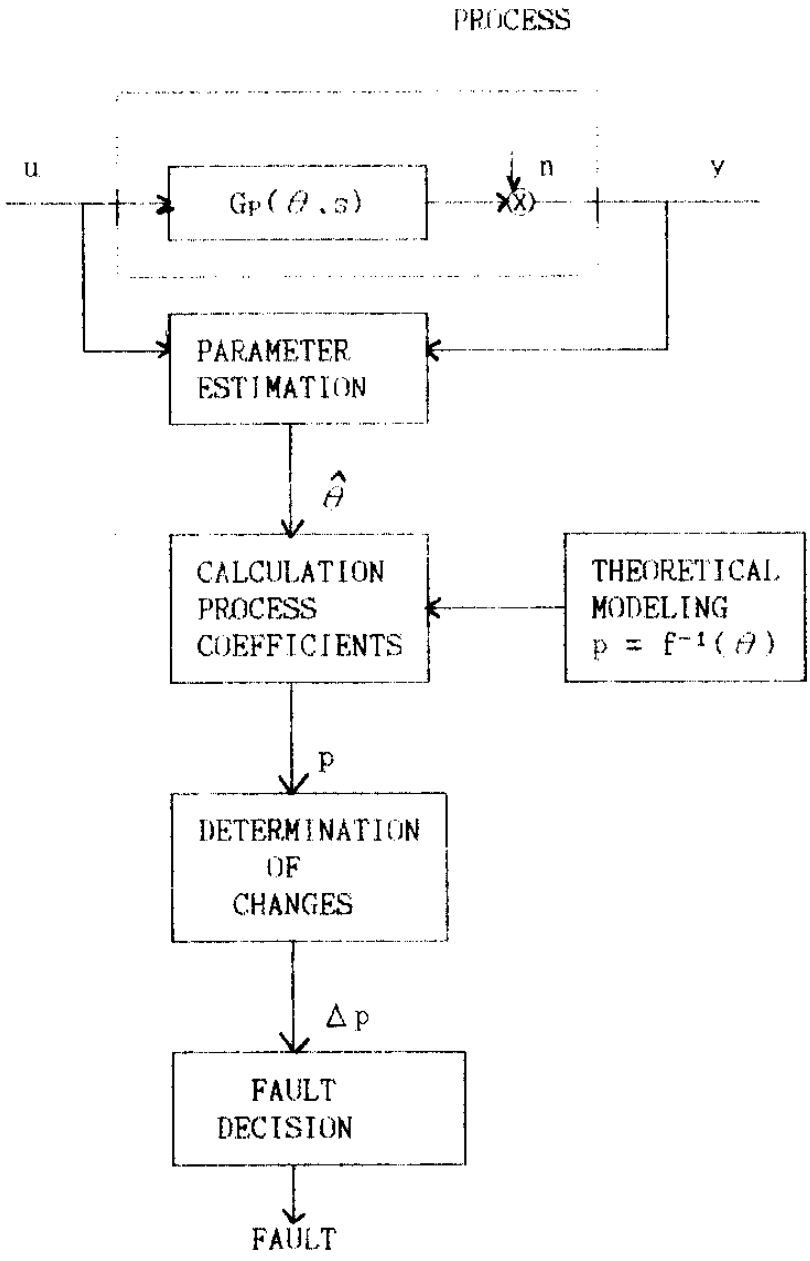


그림 5-2. 공정 매개변수 추정과 이론적 모델링에 의한 이상탐지

가능하며, 상태변수를 이용한 기법이 공정 매개변수를 알고 있는 상수로 가정하고 신호를 감시하는데 반하여, 이 방법은 물리법칙에 기초를 두고 공정을 직접 감시하는 장점을 갖는다. 이 두 방법은 상호 보완관계를 갖고 있으므로 두가지 방법을 혼합하여 사용할 수 있다.

이외의 공정의 이상탐지 방법으로는 대단위 공정 등에서 측정 불가능한 공정특성치를 검사하여 현재 공정의 내부상태에 관한 정보를 이용하는 방법이 있다. 이때 이용되는 양은 주로 다음과 같다.

1) 엔진이나 기계류 및 열교환기 등의 효율

2) 단위 생산물이나 시간당의 연료 소모량

3) 내부연소 엔진이나 압축기 등의 오일 소모율

4) 도구(tool) 이용율

5) 모타 혹은 연삭기 등의 마모율

이상의 공정특성치들은 공정의 입출력 등의 측정치들로 부터 계산되며 주로 정적 관계를 이용하며, 이 양들의 절대치의 변화나 경향을 분석하여 이상 유무를 판별한다.

이상에서 살펴 본 공정의 미측정치-상태변수, 매개변수, 특성치-를 이용한 공정 이상탐지의 구성도는 그림 5-3과 같다. 그림에서 보듯이 이와 같은 방법을 이용한 공정 이상탐지는 정상모델에서 벗어나는 관측치의 차이인 오차신호나 나머지(residuals)를 이용하여 이상을 결정(fault decision)하는 과정이 필요하다. 이러한 이상 결정에는 방향의 변화, variance의 변화 혹은 패턴의 변화

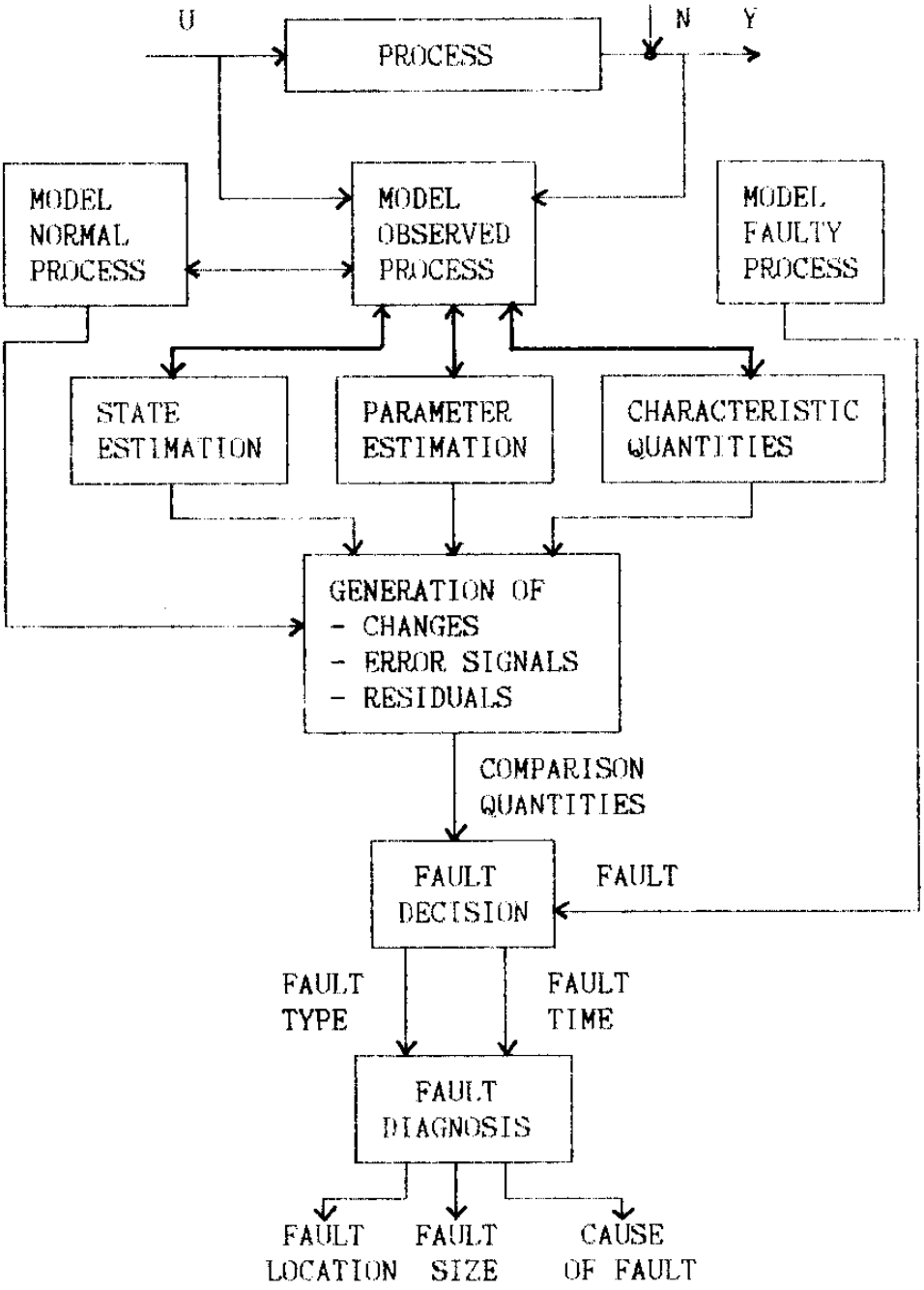


그림 5-3. 공정모델과 미측정변수를 이용한 공정 이상탐지 구성도

를 이용하는 방법들이 있으며, 이를 위해서는 통계학적 결정이론이나 패턴인식 등의 응용이론이 요구된다.

선형계에 대하여 매개변수 추정에 의한 공정 이상탐지에 관한 연구는 1976년 Willsky[4] 이래로 많은 연구결과가 발표되었으며, 여기서는 파이프라인과 모타 등에 관한 몇가지 연구결과들을 살펴보기로 한다.

(1) 직류모타-발생기의 이상탐지

직류모타-발생기에 관한 Li Zhi[5]의 연구는 공정 매개변수의 추정을 위하여 널리 쓰이는 최소자승법 (least-squares method)을 이용하였다. 즉 출력오차 모델구조 (output error model structure)를 갖는 선형계의 이산시간대 모델은

$$y = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n + e$$

여기서 y는 공정출력, x는 공정입력, e는 공정잡음항이다.

위 식은 다시 $Y = \theta X^T + e$로 쓸 수 있고, 여기서

$$X = [x_1, x_2, \cdots, x_n]^T$$

$$\theta = [\theta_1, \theta_2, \cdots, \theta_n]^T$$

이 식을 다시 k개의 샘플링에 대하여 벡타로 구성하면, 최소자승법을 적용시킬 수 있는 linear regression모델을 구할 수 있다. 최소자승법을 이용한 회귀적 변수추정식은 표준화된 알고리즘으로

$$\theta(k+1) = \theta(k) + H(k+1)[y(k+1) - x^T(k+1)\theta(k)]$$

$$H(k+1) = P(k)x(k) / [\alpha + x^T(k+1)P(k)x(k+1)]$$

$$P(k+1) = [P(k) - H(k+1)x^T(k+1)P(k)] / \alpha$$

여기서 α는 알고리즘이 시변변수에 적응하기 위한 forgetting

factor이다. 미분식으로 표현되는 다음과 같은 연속시간대의 모델에 대해서도 유사한 방법으로 최소자승법을 적용할 수 있다.

$$\frac{d^n}{dt^n} y(t) + a_{n-1} \frac{d^{n-1}}{dt^{n-1}} y(t) + \cdots + a_0 y(t) =$$

$$b_m \frac{d^m}{dt^m} u(t) + b_{m-1} \frac{d^{m-1}}{dt^{m-1}} u(t) + \cdots + b_0 u(t) + e(t)$$

시간에 대한 미분을 다음과 같이 표현하고,

$$\frac{d^i}{dt^i} y(t)\Big|_{t=t_k} = y^{(i)}(t), \quad i = 0, 1, \cdots, n$$

$$\frac{d^i}{dt^i} u(t)\Big|_{t=t_k} = u^{(j)}(t), \quad j = 0, 1, \cdots, m$$

변수벡타를 정의하여 모델을 벡타꼴로 쓰면 다음과 같다.

$$\theta = [\, a_0, a_1, \cdots, a_{n-1}, b_0, b_1, \cdots, b_m \,]^T$$

$$x(k) = [\, -y^{(0)}(k), -y(k)^{(1)}, \cdots, -y(k)^{(n-1)},$$

$$u^{(0)}(k), u^{(1)}(k), \cdots, u^{(m)}(k) \,]^T$$

$$y^{(n)}(k) = x(k)^T \theta + e(k)$$

Li Zhi는 마이크로 컴퓨터를 이용하여 온라인으로 이 연구를 수행하는 데는 계산속도와 저장용량의 문제가 생기는 것을 지적하고 이의 해결을 위하여 recursion in group이라는 계산기법을 도입하였다. 이는 컴퓨터가 한 샘플링 구간 동안에 계산을 끝내지 못하는 경우에, 그룹으로 자료를 처리하는 방법이다. 즉 한 그룹의 데이타를 메모리에 저장하고 이를 이용하여 변수 추정을 행하고 다

시 다음 그룹의 데이타를 받아들여 전작업의 수렴결과 covariance를 초기값으로 하여 변수추정을 행하는 작업을 반복적으로 수행하는 방법으로 공정신호와 잡음의 비(signal-to-noise ratio)가 작고 입력자료가 많은 경우에 유용한 방법이다. 이상결정은 innovation항을 이용하고 이 값의 경계는 실험적으로 구하였다.

$$\hat{\varepsilon}(k) = y(k) - x(k)^T \hat{\theta}(k)$$

$$0 < \hat{\varepsilon}^2(k) < \gamma^2$$

잦은 오경보를 피하기 위해 경계를 chi-squared detection을 이용하고 이 때에도 합산횟수와 경계는 실험적으로 구하였다.

$$0 < \frac{1}{M} \sum_{i=k-M+1}^{k} \hat{\varepsilon}^2(i) < \gamma^2$$

변수의 점핑현상이 위의 식으로 부터 발견되면, 저장메모리를 지우고 새로운 초기값과 데이타를 받아서 점핑 후의 수렴변수 값을 구하였다. 앞에서 설명한 공정의 매개변수와 물리적 계수와의 관계등을 알아보기 위하여 d.c. driving system의 모델식을 살펴보기로 한다. 먼저 몇 가지 가정을 도입하여 모터의 상태방정식을 수학적으로 구하면 다음과 같다.

$$\frac{di_1}{dt} = -\frac{R_1}{L_1} i - \frac{\Psi_1}{L_1} \omega + \frac{1}{L_1} u_1$$

$$\frac{d\omega}{dt} = \frac{\Psi_1}{J} i_1 - \frac{\beta}{J} \omega - \frac{1}{J} T_0$$

where, u : voltage

i : current

R :resistance

L : inductance

Ψ : flux linkage

ω : angular velocity

β : load coefficient

J : motor-generator inertia

T_0 : friction torque independent on ω

이 시스템에서 직접 측정 가능한 변수는 입력전압 u_1, armature 전류 i_1 및 축 회전속도 ω 등 3개이며, 수학적 모델식을 최소자승법에 의한 변수추정에 알맞도록 변수화 (parameterization) 하고 적당한 공정잡음항을 추가하면 다음과 같다. 즉 $\theta = f(p)$의 과정으로

$$\theta_{11} = -\frac{R_1}{L_1},\quad \theta_{12} = -\frac{\Psi_1}{L_1},\quad \theta_{13} = \frac{1}{L_1}$$

$$\theta_{21} = -\frac{\Psi_1}{J},\quad \theta_{22} = -\frac{\beta}{J},\quad \theta_{23} = -\frac{1}{J}T_0$$

$y_e : di_1/dt$

$\theta_e : [\theta_{11}, \theta_{12}, \theta_{13}]^T$

$x_e : [i_1, \omega, u_1]^T$

$y_m : d\omega / dt$

$\theta_m : [\theta_{21}, \theta_{22}, \theta_{23}]^T$

$x_m : [i_1, \omega, 1]^T$

$$y_e = x_e^T\theta_e + e_e$$

$$y_m = x_m^T\theta_m + e_m$$

여기서 하첨자 e는 electrical 방정식을 m은 mechanical 방정식을 각각 나타낸다. 실험은 계를 여기(exciting) 시켜주기 위하여 입력 전압 u_1에 PRBS(pseudo random binary sequence)를 부가하면서 저항의 점핑을 진단하였다. 이상의 연구 결과를 보면 널리 이용되고 있는 최소자승법을 이용하여 수학적 모델링이 용이한 d.c. motor driving 시스템의 신속한 이상진단을 성공적으로 수행할 수 있음을 알 수 있다.

(2) Fault zone을 이용한 파이프라인 내경진단

다음은 fault zone이라는 개념을 도입하고 이의 추정을 통한 파이프라인 내경축소의 진단에 관한 L. Tao와 C. Fang의 연구[3]를 살펴보기로 한다. 즉 분산변수계(distributed parameter system DPS)에서 이상은 system space의 한 영역에서 상태변수나 공정 매개변수의 변동으로 나타나므로, 이 영역을 fault zone(FZ)으로 두고 이 영역에 대하여 FZ의 위치, FZ의 크기 및 FZ내에서의 변수의 변화폭 등의 세가지 변수를 인식(identification)하는 방법이다. 분산변수계에서 space-dependent parameter의 추정은 매우 힘든 작업으로 이 연구에서는 다음과 같은 일반적인 사실을 근거로 문제를 단순화하여 진단을 행하였다. 즉 이상변수의 추정은 일반적인 변수추정과는 다음의 여러가지 면에서 구별된다.

1) 근사적인 변수추정 만이 필요하며, 때로는 단지 parameter variation의 존재유무나 변화 방향의 결정 만이 요구된다.
2) 이상의 양상을 알 수 있다.
3) 몇가지 추정하려는 이상에 영향을 받는 변수를 제외하고는

대부분의 공정변수를 알고있다.

4) 진단시간이 짧아야 한다.

5) 대부분의 잘 설계된 공정에 대해서는 이상의 발생확률이 매우 작다.

DPS에서 이상의 유형은 파이프라인의 유출과 같은 system space 내의 한 지점에서 변수나 구조의 변동과 icing이나 buckling 혹은 paraffine deposition등에 의한 파이프라인에서의 내경축소와 같은 system space내의 미지의 영역에서 변수에 변동이 일어나는 두가지로 나눌 수 있다. 전자에 관해서는 다음 절에서 살펴보기로 한다. 위에서 밝힌 변수추정의 차이를 기초로 L. Tao와 C.Fang은 다음의 세가지 가정을 도입하였다.

1) These parameters can be estimated from input and output measurements under certain allowable test conditions ;

2) The occurence probability of faults is so small that they happen only one at a time ;

3) The mathematical model of the normal process is known and the mode of faulty parameter variation can be reasonably predetermined.

이와 같은 가정을 바탕으로 도입한 fault zone의 개념은 "정상적인 DPS정의구역내의 비정상적인 영역으로, 공정변수는 이 영역을 제외하고는 모두 알고 있으며, 이 영역에서 공정변수의 비정상 변동 양태를 합리적으로 예견할 수 있다."는 것이다. 이 개념을 파이프라인 내경축소 진단에 이용하기 위하여 정의한 세가지 추정변

수는 그림 5-4에서 보듯이 위치(L_f), 범위(E_f) 및 축소된 내경(D_f)이다. 여기서 위치와 범위의 두가지 FZ변수는 고전적인 원리인 "하나는 둘로 나뉘어 진다."는 단계적 공간분할 방법으로 추정하고, 축소된 내경은 추계적 근사 알고리즘을 이용하였다.

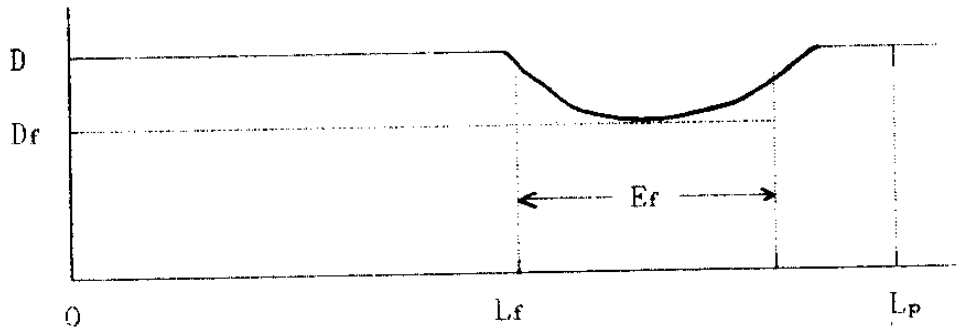


그림 5-4. 파이프라인의 내경축소

일반적인 DPS에서 FZ를 추정하는 방법은 다음 과정을 따른다.

1 단계 : 공정의 실시간대 동특성 모델의 구성.

2 단계 : 가능한 이상 및 이상에 영향을 받는 공정변수 조사.

3 단계 : 2 단계에서 찾은 공정변수 인식성 (identifiability) 조사.

4 단계 : 이상유무를 탐지하는 온라인 알고리즘 개발.

5 단계 : 이상변수 변동에 가장 민감하고 쉽게 액세스할 수 있는 system space에 필요한 측정장치의 설치.

6 단계 : FZ내의 비정상변수 변동양태를 결정.

7 단계 : 가정한 FZ를 갖는 모델과 실제 공정 사이의 차이 정도를 나타내는 적당한 기준 J_d을 정의.

8 단계 : 단계 10 과 11 에 이용할 단계적 공간분할 방식의 설계

9 단계 : 정상 운전조건 하에서 FZ에 관한 정보를 추출할 수 있는지를 조사. 만약 불가능하면, FZ인식을 위한 적당한 외부 자극신호(stimulating signal)의 설계.

10 단계 : FZ내에서 고려하는 변수에 대한 J_d 값에 대한 기울기나 감소 방향을 계산하고, 가정한 FZ의 위치와 범위를 기초하여 실시간대 모델을 이용 J_d 를 최소화하는 이상에 영향을 받은 변수 값을 추정.

11 단계 : FZ와 J_d 에 대응하는 공정변수를 찾기 위하여, 단계 8에서 설계한 단계적 공간분할 방식을 이용하여 FZ의 위치 및 범위를 변화.

12 단계 : FZ의 3가지 변수에 대하여 J_d 가 최소가 될 때까지 반복적으로 단계 11을 수행.

위와 같은 과정을 따른 파이프라인의 모델링, 모사 및 실험결과들은 다음과 같다. 먼저 수학적 모델링은 도관을 흐르는 유체에 대한 방정식을 구하고, 이 비선형 편미분방정식을 method of characteristics로 풀면 다음과 같은 비선형 이산시간-공간대의 상태방정식을 얻는다. 그림 5-5와 같이 파이프라인을 N개의 구간으로 나누면

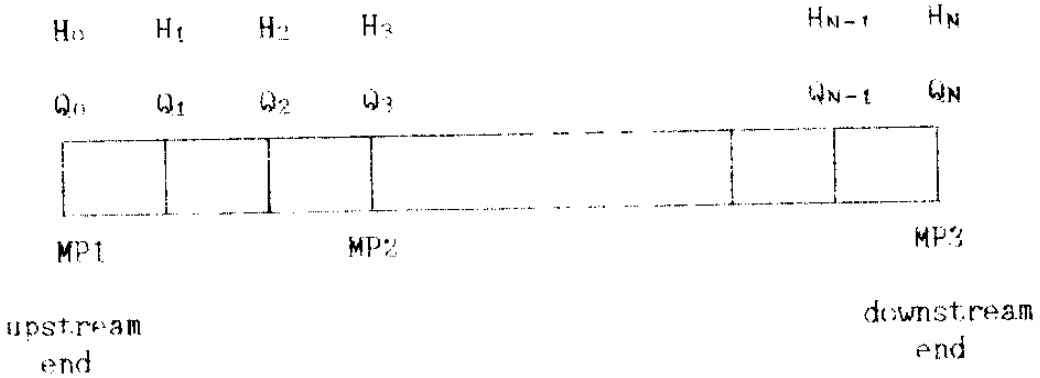


그림 5-5. 파이프라인의 분할 및 상태변수 정의

$$x(k+1) = Ax(k) + Mx^2(k) + u$$

$$x(k) = [H_1, H_2, \cdots, H_{N-1}, Q_0, Q_1, \cdots, Q_N]_k^T, \quad Q_i > 0$$

$$x^2(k) = [H_1^2, H_2^2, \cdots, H_{N-1}^2, Q_0^2, Q_1^2, \cdots, Q_N^2]_k^T$$

여기서 H는 내수압(piezometric head)이고, Q는 유속을 나타내며 MP1, MP2 및 MP3은 실제 압력측정 위치이다.

행렬 A와 M 그리고 제어벡타 u의 내용은 본 장 3절에서 설명하기로 한다. 이 모델식은 파이프라인의 입구와 출구에서의 측정치들을 공정입력으로 취하고, 샘플링을 pressure wave가 한 파이프라인 구간을 통과하는 데 걸리는 시간마다 행하면 실시간대 관측자(observer)로 사용할 수 있다.

$$T_s = L_p / aN$$

여기서, T_s : sampling interval

L_p : total length of pipeline

a : wave speed in pipe fluid

N : number of sections

다음으로 이상에 의한 측정치 변화의 예측과정은 정적 압력분포를 이용한다. 즉 내경의 축소에 따른 정적 압력분포는 FZ의 위치,범위 및 내경에 따라 직선적으로 감소하는 분포에서 입구부는 상승하고 출구부는 하강하는 분포를 갖는다. 그러나 파이프라인의 진단을 적용하는 일반적인 대상 공정들—송유관, 도시가스관, 상하수도등—은 도관 입출구에 가까운 부근에서의 측정치 만을 구할 수 있으므로 이 분포를 이용한 FZ의 인식은 불가능하다. 따라서 이 연구에서는 FZ에 따라 압력파장의 전파특성이 달라진다는 Wylie[7]

의 연구결과를 이용하여 인식을 수행하였다. 측정은 그림 5-5 와 같이 압력측정 MP1, MP2, MP3 및 입력유속 Q_1 등의 4개이며 이들 중에서 Q_1 은 FZ의 탐지와 내경추정에, H_0 과 H_N 의 측정을 위한 MP1 과 MP3 는 공정모델의 입력으로, H_3 의 측정을 위한 MP2 는 기준치 J_d 의 계산을 위하여 각각 이용하였다.

$$J_d = \sum_{k=k_0}^{k_0+N_J-1} [\hat{H}_3(k) - H_3(k)]^2 \Big/ \sum_{k=k_0}^{k_0+N_J-1} [\hat{H}_3(k) - H_3(k)]^2$$

여기서 $H_3(k)$는 k번째 샘플링 시간의 측정치이고 $\hat{H}_3(k)$는 모델에 의한 추정치이며, N_J 는 기준치 J_d 의 계산을 위한 샘플링 자료의 길이이다. 단계적 공간 분할방식은 '하나는 둘로 나뉘어진다'는 간단한 원리를 이용하여, FZ를 가정하였다. 즉 FZ를 세가지 범위-전구간, 전반부의 반과 후반부의 반-로 가정하고 실시간모델을 이용하여 각 범위에서 내경 D_f 를 변화시키면서 $Q_1 - \hat{Q}_1$ 의 값이 영이되는 최적 내경을 찾는다. 이 값을 이용하여 각 범위에서의 기준치 J_d 를 구하고, 이 값이 제일 작은 FZ범위를 다시 이등분하여 위의 과정을 반복하여 기준치가 최소될 때까지 이등분할을 실시한다. 실험과 모사는 길이가 120 m, 내경이 10 mm인 water pipeline에 대하여 수행하였고, 샘플링 시간은 파장전달시간으로 부터 구한 5ms으로 하였다. 입구부의 내수압은 대략 55m이고 계를 여기시켜 주기 위하여 입구부에 solenoid밸브를 설치하여 random sequence(PRBS)를 부가할 수 있도록 하였다. 먼저 모사는 PRBS의 크기를 입구내수압의 5%에 해당하는 2.8 m로 부가하고 부가주기는 50ms에서 250ms 사이이고, 측정잡음의 크기는 측정치의 5

%로 하였다. 모델은 전라인을 18구간으로 나누어, 36개의 이산상태 방정식으로 구성하였다. 이상을 파이프라인의 66.7 m에서 80 m 사이의 내경이 9 mm로 축소되는 것으로 가정하고 모사한 결과가 그림 5-6과 같다. 그림에서 =는 구간을 나타내며 이등분할에 의한 FZ인식을 볼 수 있다. 인식한 FZ값이 실제와 차이가 나는 것을 신호대 잡음의 크기비(signal-tonoise ratio)가 작은 이유로 분석하고 있다. 이것의 영향을 알아보기 위하여 PRBS의 크기 및 부가주기를 변화시키면서 모사한 결과들을 예로 들었다. 실험은 모사조건과 같은 이상 구간에 내경 4 mm, 외경 6mm의 나이론 튜브를 삽입하였다. 이 부분에서는 마찰계수가 달라지고 내경은 대략 7.41 mm로 축소되는 결과가 생긴다. 기준치 계산을 위한 자료 갯수를 200개로 취하고 실험한 결과는 그림 5-7과 같다.

이상의 연구결과를 살펴보면 Faulty zone이라는 개념과 실시간대 모델 및 인식을 통하여 파이프라인의 내경진단을 수행할수 있음을 볼 수 있다. 그러나 이 방법은 공정모델을 필요로 하며 공정신호대 잡음의 비에 크게 영향을 받으므로 PRBS등으로 계를 끊임없이 여기 시켜야만 하는 제약이 있다.

presumed FZ	J_d	D_f (mm)
==========	1. 90	9. 78
=======	1. 97	9. 58
=======	1. 83	9. 58
===	1. 79	9. 27
===	1. 93	9. 27
==	1. 96	8. 99
==	1. 61	8. 99
=	1. 78	8. 42
=	1. 69	8. 42
===	1. 70	9. 27
FZ found ==	1. 61	8. 99
FZ real ==		9. 00

그림 5-6. FZ를 이용한 파이프라인 내경진단 모사결과

presumed FZ	J_d	D_f (mm)
==========	.300	9. 78
=======	.469	8. 58
=======	.252	8. 58
===	.170	7. 87
===	.710	7. 87
==	.289	7. 42
==	.166	7. 42
=	.397	6. 60
=	.173	6. 60
===	.317	7. 벽7
FZ found ==	.166	7. 42
FZ real ==		7. 41

그림 5-7. FZ를 이용한 파이프라인 내경진단 실험결과

(3) 복합공정의 이상진단

앞에서 소개한 몇가지의 매개변수추정에 의한 공정의 이상진단 연구들은 비교적 간단한 공정을 대상으로 한 것들이나, 대부분의 화학공정들은 최소한 몇가지 기본 단위공정의 복합체로 구성되어 있으므로 좀 더 복잡한 계에 대한 이상진단 예를 살펴보고자 한다. 대상공정은 Geiger [8]가 연구한 속도제어가 되는 직류모터에 의해 운전되는 원심펌프가 있는 유체순환계로 직류모터, 펌프 및 순환시스템에서의 이상탐지를 목적으로 한다. 첫단계로 에너지와 모멘텀수지식 등을 이용하여 모터, 펌프 및 순환계의 수학적 모델식을 구성한다. 이 과정에서는 가능하면 적당한 가정을 도입하여 공정매개변수의 수를 줄이는 작업이 필요하다. 모델식의 유도과정은 Isermann의 조사논문 [1] 부록에 있으며, 모델인식을 위한 결과식들은 다음과 같다.

(a) Armature circuit

$$\frac{dI_1(t)}{dt} = a_{11}\triangle I_1(t) + a_{12}\triangle w(t) + b_1\triangle U_1(t)$$

(b) Mechanics motor and pump

$$\frac{dw(t)}{dt} = a_{21}\triangle I_1(t) + a_{22}\triangle w(t) + a_{23}\triangle M(t)$$

(c) Pipe system

$$\frac{dM(t)}{dt} = a_{33}'\triangle M(t) + d_3\triangle Y(t)$$

(d) pump(specific energy Y)

$$\triangle Y(t) = h_W(t) + h_M \triangle M(t)$$

여기서 공정 매개변수들과 물리적 계수사이에는 다음과 같은 관계를 가진다.

$$a_{11} = -\frac{R_1}{L_1} \qquad b_1 = \frac{1}{L_1}$$

$$a_{12} = -\frac{\psi}{L_1}$$

$$a_{21} = -\frac{\psi}{\theta} = \frac{\psi}{\theta_M + \theta_P}$$

$$a_{22} = -\frac{c_{F1} + g_W}{\theta} \qquad a_{23} = -\frac{g_M}{\theta}$$

$$a_{33}' = -\frac{a_R}{a_{ac}} \qquad d_3 = \frac{1}{a_{ac}}$$

상태방정식을 구성하기 위하여 변수와 상태벡타 및 전이행렬을 다음과 같이 정의한다.

$$a_{32} = \frac{h_W}{a_{ac}} \qquad a_{33} = a_{33}' + \frac{h_W}{a_{ac}}$$

$$x(t) = \begin{bmatrix} \triangle I_1(t) \\ \triangle W(t) \\ \triangle M(t) \end{bmatrix} \qquad y(t) = \begin{bmatrix} \triangle I_1(t) \\ \triangle W(t) \\ \triangle M(t) \\ \triangle Y(t) \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & h_W & h_M \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ 0 \\ 0 \end{bmatrix} \qquad u(t) = \triangle U_1(t)$$

$$\dot{x}(t) = Ax(t) + bu(t)$$

$$y(t) = Cx(t)$$

전이행렬 A, B, C의 원소들은 최소자승법에 의하여 구할수 있으며, 이렇게 추정된 공정 매개변수로부터 물리적계수를 구하여 공정을 감시하는 데는 공정의 운전상태와 측정변수에 따라 달라진다. 이 과정을 여러 경우에 대하여 살펴보도록 한다.

먼저 순환계의 밸브가 닫힌 경우에는 $M(t) = 0$이 되며, 측정신호가 $\triangle I_1(t)$, $\triangle w(t)$, $\triangle U_1(t)$의 세가지일때는 공정수지식을 최소자승법을 적용시키기에 알맞도록 다음과 같이 쓸 수 있다.

$$y_1(t) = dI_1(t) / dt \qquad y_2(t) = dw(t) / dt$$

$$\psi_1^T(t) = [\ \triangle I_1(t) \quad \triangle w(t) \quad \triangle U_1(t)\ ]$$

$$\hat{\theta}_1^T = [\ \hat{a}_{11}\ \hat{a}_{12}\ \hat{b}_1\ ]$$

$$\psi_2^T(t) = [\ \triangle I_1(t) \quad \triangle w(t)\ ]$$

$$\hat{\theta}_2^T = [\ \hat{a}_{21}\ \hat{a}_{22}\ ]$$

물리적 계수와 매개변수와의 관계는 추정된 다섯개의 매개변수 $\hat{\theta}_1$ $\hat{\theta}_2$와 위에서 구성한 관계식을 이용하여 구할 수 있다.

$$\hat{L}_1 = 1/\hat{b}_1$$

$$\hat{R}_1 = -\hat{a}_{11}\hat{L}_1 = \hat{a}_{11}/\hat{b}_1$$

$$\hat{\phi} = -\hat{a}_{12}\hat{L}_1 = -\hat{a}_{12}/\hat{b}_1$$

$$\hat{\theta} = \hat{\phi}/\hat{a}_{21} = -\hat{a}_{12}/\hat{b}_1\hat{a}_{21}$$

$$C_{F1} = C_{FM1} + C_{FMW} = \hat{a}_{22}\hat{\theta} = \hat{a}_{22}\hat{a}_{12}/\hat{b}_1\hat{a}_{21}$$

그러나 모델식을 간략하게 하기 위하여 lumping한 모타와 펌프의 마찰계수 C_{FM1}, C_{FMW}와 관성모멘트 θ_M, θ_P는 각각의 합으로만 구해진다. 또한 동적인식을 하지않고 단지 정상상태인식을 수행하면 수지식에서 $d/dt = 0$으로 두면 L_1과 θ는 구할 수 없다. 따라서 동적인식을 수행해야 더 많은 매개변수를 추정할 수 있고 결과적으로 더 많은 물리적 계수를 감시할 수 있게 된다. 또한 위와 같은 선형화한 동적관계식을 사용하면 접착 마찰계수 C_{FM0}, C_{FP0}를 구할 수 없다. 이의 해결을 위해서는 마찰토오크는 속도에만 선형적으로 의존한다는 가정을 세운다.

$$T_{FM}(t) = C_{FM0} + C_{FM1W}(t)$$

$$T_{FP}(t) = C_{FP0} + C_{FP1W}(t)$$

$$(\theta_M + \theta_P)\frac{dw(t)}{dt} = \phi I_1(t) - C_{F0} + C_{F1W}(t)$$

$$\theta = \theta_M + \theta_P$$

$$C_{F0} = C_{FM0} + C_{FP0}$$

$$C_{F1} = C_{FM1} + C_{FP1}$$

따라서 θ를 추정하는데 w(t)와 $I_1(t)$의 △대신 절대값을 사용하므로 C_{F0}를 추정할 수 있다. 다음으로는 운전조건은 같지만 측정신호가 $U_1(t)$와 △w(t)로 armature전류를 측정할 수 없는 경우에는 모델식이 다음과 같이 바뀐다.

$$\frac{d^2w(t)}{d^2t} + \alpha_1 \frac{dw(t)}{dt} + \alpha_0 w(t) = \beta_0 \triangle U_1(t)$$

where,

$$\alpha_1 = \frac{R_1}{L_1} + \frac{C_{F1}}{\theta},\ \alpha_0 = \frac{1}{\theta L_1}(C_{F1}R_1 + \psi^2)$$

$$\beta_0 = \psi / L_1\ ;\ C_{F1} = C_{FM1} + C_{FPW}$$

위 식에서는 물리적 계수는 5개이고 공정매개변수는 3개이므로 유일한 해를 구할 수 없다. 따라서 두개의 계수를 알고 있어야 하며, 이 문제로부터 가능하면 많은 변수를 측정할 필요가 있다는 것을 알 수 있다. 정상상태특성만 이용하면 앞의 경우와 마찬가지로 추정가능한 변수가 줄어 여기서는 1개가 된다. 따라서 두 계수의 물리적 계수를 가정해야 한 변수를 구할 수 있다. 즉,

$$\frac{\alpha_0}{\beta_0} = \frac{C_{F1}R_1}{\psi} + \psi$$

그러나 물리적 변수를 가정할 수 없는 경우에는 변수의 변화만을 추정할 수 있다. 즉 표5-1과 같이 공정계수의 변화(공정이상)에 종속되는 공정 매개변수의 변화방향을 결정해 두면 유일한 패턴을 인식할 수 있다.

표 5-1 . 공정매개변수의 변화패턴

	$\triangle\alpha_0$	$\triangle\alpha_0$	$\triangle\beta_0$
$+\triangle R_1$	+	+	0
$+\triangle \psi$	0	+	+
$+\triangle L_1$	-	−	−
$+\triangle \theta$	-	−	−
$+\triangle C_{F1}$	+	+	0

다음은 순환계에서 밸브가 열려 있고 측정신호가 $\triangle I_1(t)$, $\triangle U_1(t)$, $\triangle w(t)$, $\triangle Y(t)$, $\triangle M(t)$인 경우로서 이때에는 앞에서 구성한 4개의 수지식을 모두 최소자승법을 이용한 변수추정에 사용한다. 즉,

$$y_1(t) = dI_1(t)/dt \qquad y_2(t) = dw(t)/dt$$

$$y_3(t) = dM(t)/dt \qquad y_4(t) = \triangle Y(t)$$

$$\psi_1^T(t) = [\ \triangle I_1(t) \quad \triangle w(t) \quad \triangle U_1(t)\ ]$$

$$\theta_1^T = [\ a_{11}\ a_{12}\ b_1\ ]$$

$$\psi_2^T(t) = [\ \triangle I_1(t) \quad \triangle w(t) \quad \triangle M(t)\ ]$$

$$\theta_2^T = [\ a_{21}\ a_{22}\ a_{23}\ ]$$

$$\psi_3^T(t) = [\ \triangle M(t) \quad \triangle Y(t)\ ]$$

$$\theta_3^T = [\ a_{33}'\ \ d_3\ ]$$

$$\psi_4^T(t) = [\ \triangle w(t) \quad \triangle M(t)\ ]$$

$$\theta_4^T = [\ h_W\ \ h_M\ ]$$

공정의 물리적 계수들은 앞에서와 마찬가지로 4개의 매개변수 벡타 θ에 의해 모두 계산된다.

$$\hat{L}_1 = 1/\hat{b}_1$$

$$\hat{R}_1 = -\hat{a}_{11}\hat{L}_1 = \hat{a}_{11}/\hat{b}_1$$

$$\hat{\psi} = -\hat{a}_{12}\hat{L}_1 = -\hat{a}_{12}/\hat{b}_1$$

$$\hat{\theta} = \hat{\psi}/\hat{a}_{21} = -\hat{a}_{12}/\hat{b}_1\hat{a}_{21}$$

$$\hat{c}_{F1} + \hat{g}_W = -\hat{a}_{22}\hat{\theta} = \hat{a}_{22}\hat{a}_{12}/\hat{b}_1\hat{a}_{21}$$

$$\hat{g}_M = -\hat{a}_{23}\hat{\theta} = \hat{a}_{23}\hat{a}_{12}/\hat{b}_1\hat{a}_{21}$$

$$\hat{a}_{ac} = 1/\hat{d}_3$$

$$\hat{a}_R = -\hat{a}_{33}\hat{a}_{ac} = -\hat{a}_{33}/\hat{d}_3$$

그러나 순환계의 밸브가 닫힌 경우의 문제에서처럼 측정변수가 부족하면 몇개의 공정계수들은 구할 수가 없다. 예를들면 Y(t)가 측정되지 않으면 수지식으로부터 a_{33}, d_3, h_M, 및 h_W가 구해지지 않음을 알 수 있고 따라서 a_R, a_{ac}, h_M, h_W, 및 c_{F1}이 결정되지 않는다. 만일 M(t)가 측정되지 않으면, d_3, h_M, 및 a_{23}가 구해지지 않고 따라서 공정계수중에서 a_R, a_{ac}, h_M, g_M 및 c_{F1}을 결정할 수 없다. 따라서 이들 값들을 가정하고 문제를 풀어야 한다. 만약 센서의 동특성이 공정에 비하여 무시할 수 없는 경우에는 이 영향을 모델에 고려하여야 한다.

이상에서 살펴본 바에 의하면 공정매개변수의 모니터링에 의해 공정의 이상을 진단하는 일은 공정의 엄밀한 수학적 모델과

모델인식과정이 필요하다. 또한 정적모델에 의한것 보다는 동적모델을 이용하는 것이 더 많은 공정계수를 추정할 수 있고 따라서 이상진단의 정확도를 기대할 수 있다. 측정변수는 이것의 수에 의해 공정이 나뉘어지므로 가능하면 공정이 1차로 나뉠 수 있도록 즉 모든 상태변수가 측정될 수 있도록 선정해야 한다.

(4) 적응제어계에서의 변수추정에 의한 이상진단

적응제어를 행하고 있는 시스템에서 이상의 발생으로 인하여 물리적 계수가 변했을 때, 이상의 영향은 적응메카니즘때문에 출력으로 잘 나타나지 않는다. 또한 공정매개변수는 직접 접근에 의한 회귀적 인식으로 결정되기 때문에 레귤레이터의 변수를 감시해서는 이 값의 변화를 감지하기가 어렵다. 이와 같은 경우에 이상의 영향은 적응제어기설계에 이용한 예측모델의 변화로 생각할 수 있으며, 따라서 이상탐지는 모델분별문제 (discrimination problem) 가 된다. 적응제어계의 이상진단에 관한 연구는 그리 많지 않으며, 이에 관한 연구로는 1983년 Hägglund가 온라인 이상진단을 위한 새로운 추정기법을 제시하였다. 여기서 이상탐지는 회귀적 인식으로 구한 공정매개변수의 변화에 관한 정보를 이용하는 것으로 간접 적응제어법칙의 설계에 이용하였다. 또 다른 접근법으로는 1988년의 Kumamaru 등이 Kullback분별자를 이용한 직접법[9]이 있으며, 여기서는 이 연구에 관하여 살펴보기로 한다.

Kullback Discrimination Index(KDI)는 원래 두 개의

확률밀도함수를 비교하기 위하여 Kullback에 의하여 고안된 것으로 가우시안시스템에서 정보처리에 널리 이용되어 왔다. 이 KDI를 이용한 모델분별은 다음과 같이 이루어진다. 선형 다변수계는

$$S : y(t) = G_0(q^{-1})\, u(t) - H_0(q^{-1})\, e(t),\ t = 1,\ 2 \cdots\cdots$$

여기서 $y(t)$는 n_y차원의 출력벡타, $u(t)$는 n_u차원의 입력벡타이고 $e(t)$는 $n_e\ (= n_y)$차원의 백색가우시안 잡음으로 영평균과 공분산행렬 Λ을 갖는다.

$$e(t) = N(0,\ \Lambda)$$

또한 $G_0(q^{-1})$와 $H_0(q^{-1})$는 q^{-1}의 분수함수로 이루어진 행렬이며 아래와 같은 성질을 갖는 것으로 가정한다.

1) $G_0(q^{-1})$ and $H_0(q^{-1})$ are causal

2) $G_0(0) = 0$, $H_0(0) = 1$

3) $H_0(q^{-1})$ and $H_0^{-1}(q^{-1})$ are asymptotically stable

위와 같은 시스템에 대한 모델인식을 행하기 위하여 선정한 모델구조를 공정매개변수 θ와 함께 쓰면 다음과 같다.

$$M(\theta):\ y(t) = G(q^{-1};\theta)\, u(t) - H(q^{-1};\theta)\, e(t)$$

$$e(t) = N(0,\ \Lambda(\theta))$$

즉 공정과 모델구조를 같게 함으로써 모델집합 $M(\theta)$가 실제공정을 포함하도록 하고, 이 집합내의 어떠한 θ도 위의 G_0, H_0

에 대한 가정을 만족하며 또한 공정으로부터 그림5-9와 같은 두개의 구간 I_1, I_2에서 자료를 입력받을 수 있는 것으로 가정한다. 이제 두 샘플링구간에서 공정매개변수의 변화로 인한 이상이 발생하면 이 영향은 인식된 모델 $M(\hat{\theta}_1)$과 $M(\hat{\theta}_2)$의 차이로 나타난다. 변형측정자(distortion measure)로 KDI를 다음과 같이 likelihood함수를 이용하여 나타내면

$$I_{N1}[1, 2] = \int \rho(Y_{N1}/\hat{\theta}_1, U_{N1-1}) \ln \frac{\rho(Y_{N1}/\hat{\theta}_1, U_{N1-1})}{\rho(Y_{N1}/\hat{\theta}_1, U_{N1-1})} dY_{N1}$$

여기서 $\rho(Y_{N1}/\hat{\theta}_1, U_{N1})$은 모델 $M(\hat{\theta}_1)$에 대한 likelihood함수이고, Y_k와 U_k는 구간 I_1에서 시간지수 k까지의 데이타 모음이다.

$$Y_k = [y(1)^T, y(2)^T \cdots\cdots y(k)^T]^T$$

$$U_k = [u(1)^T, u(2)^T \cdots\cdots u(k)^T]^T$$

따라서 지표 I_{N1}은 구간 I_1내의 자료를 모델 $M(\theta_2)$가 얼마나 잘 표현하는가를 나타낸다.

회귀적으로 KDI를 계산하는 알고리즘은 다음과 같이 likelihood함수의 Bayesian analysis를 이용하여 구할 수 있다.

$$\rho(Y_{k+1}/\hat{\theta}_1, U_k) = \rho(y(k+1)/\hat{\theta}_1, Y_k, U_k)\ \rho(Y_k/\hat{\theta}_1, U_{k+1})$$

$$k = 1, 2, \cdots, N1+1$$

$$I_{N1}[1,2] = I_0[1,2] + \sum_{j=1}^{J} I_{N1}[1,2]^{(j)}$$

KDI를 이용하여 모델을 분별하는 방법은 매개변수의 성분들 간의 가중치를 고려하지 않는 direct norm criterion $\|\hat{\theta}_1 - \hat{\theta}_1\|^2$를 이용하는 것보다 훨씬 효율적이다.

공정에서 발생한 이상의 영향이 인식된 모델들간의 차이로 나타나므로 이상탐지는 모델분별접근법에 의한 batch procedure로 수행된다. 즉 발단치(threshold) η를 이용하는 결정문제가 되는 것이다. 즉,

$$I_{N1}[1,2] > \text{or} < \eta \begin{cases} M(\hat{\theta}_1) \neq M(\hat{\theta}_2); \text{ fault} \\ M(\hat{\theta}_1) = M(\hat{\theta}_2); \text{ no fault} \end{cases}$$

이때 발단치 η는 KDI의 통계적 성질에 따라 결정되므로 정상적인 상황에서 KDI의 근사적 분포가 분석되어야 한다. 분석결과는 $I_{N1}[1,2]^{(j)}$가 매개변수들의 차원과 같은 degree를 갖는 asymptotical Chi-squared distrbution을 갖는다는 것이다. 이러한 통계학적 성질은 발단치 η를 정하는데 이용된다.

단일 입출력계(SISO)에 대하여 위와 같은 직접접근법에 의한 적응제어 알고리즘을 전개하면 다음과 같다. ARMAX모델에 대하여

$$S : A(q^{-1})\, y(t) = q^{-d} B(q^{-1})\, u(t) + C(q^{-1})\, e(t),\ t = 1,2,\cdots$$

여기서 d는 순수시간 지연이고 A, B 및 C는 q^{-1}의 상수 다항식으로서

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_n q^{-n}$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_m q^{-m}, (b_0 \neq 0)$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \cdots + c_l q^{-l}$$

공정 변수벡타는 다음과 같이 정의한다.

$$a = [a_1, a_2, \cdots, a_n]^T$$

$$b = [b_0, b_1, \cdots, b_m]^T$$

$$c = [c_1, c_2, \cdots, c_l]^T$$

다음과 같은 'one-step ahead' 비용함수를 고려하면

$$J(t+d) = E\{[y(t+d) - y^*(t+d)]^2/2 + \lambda u(t)^2/2$$

여기서 $y^*(t+d)$는 설정치이고, λ는 가중치이다. 따라서 이 비용함수를 최소화하는 weighted one-step ahead 제어입력은

$$u(t) = \beta_0/(\beta_0^2 + \lambda)\{q[\beta_0 - \beta(q-1)]u(t-1) + y^*(t+d) + [C(q^{-1}) - 1]y(t+d/t) - \alpha(q-1)y(t)\}$$

여기서,

$$\alpha(q^{-1}) = a_0 + a_1 q^{-1} + \cdots + a_{n-1} q^{(n-1)} = G(q^{-1})$$

$$\beta(q^{-1}) = \beta_0 + \beta_1 q^{-1} + \cdots + \beta_{m+d-1} q^{-(m+d-1)} = F(q^{-1})B(q^{-1})$$

여기서 F와 G는 다음 관계식에서 유일하게 구해진다.

$$C(q^{-1}) = A(q^{-1})F(q^{-1}) + q^{-d}G(q^{-1})$$

$$F(q^{-1}) = 1 + f_1 q^{-1} + f_2 q^{-2} + \cdots + f_{d-1} q^{-(d-1)}$$

$$G(q^{-1}) = g_0 + g_1 q^{-1} + g_2 q^{-2} + \cdots + g_{n-1} q^{-(n-1)}$$

또한 $y^0(t+d/t)$는 $y(t+d)$의 one-step ahead 최적예측치로

$$C(q^{-1})y^0(t+d/t) = \alpha(q^{-1})y(t) + \beta(q^{-1})u(t)$$

예측오차 $\tilde{y}(t+d/t) = y(t+d) - y^0(t+d/t)$는

$$\tilde{y}(t+d/t) = F(q^{-1})e(t+d) = \varepsilon(t+d)$$

공정 매개변수벡타 θ와 regression벡타 ϕ를 도입하여 one-step ahead최적예측치를 표현하면

$$y^0(t+d/t) = \phi^T(t)\theta$$

$$\theta = [\alpha_0\ \alpha_1 \cdots \alpha_{n-1}\ \beta_0 \cdots \beta_{m+d-1}\ c_1 \cdots c_1]^T$$

$$\phi(t) = [y(t) \cdots y(t-n+1)\,u(t) \cdots u(t-m-d+1)$$

$$y^0(t+d-1/t-1) \cdots y^0(t+d-1/t-1)]^T$$

공정 변수벡타 a, b 및 c를 모르는 경우 직접접근법에 의한 적응제어입력은 다음의 순서로 계산할 수 있다. 먼저 아래의

criterion를 이용하여 θ를 추정한다.

$$\hat{\theta} = \arg \min E[y(t+d) - y^0(t+d/t)]^2$$

$$y(t+d) = \phi^T(t)\theta + \varepsilon(t+d)$$

θ를 추정하는데 회귀적 최소자승법을 이용하면

$$\hat{y}(t) = \phi^T(t-d)\hat{\theta}(t-1)$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)[y(t) - \hat{y}(t)]$$

$$L(t) = p(t-1)\phi(t-d)/[1 + \phi^T(t-d)p(t-1)\phi(t-d)]$$

$$P(t) = P(t-1) - L(t)\phi^T(t-d)P(t-1)$$

$$\phi(t-d) = [y(t-d) \cdots y(t-d-n+1)\ u(t-d) \cdots u(t-m-d+1)\ \hat{y}(t-1) \cdots \hat{y}(t-\ell)]^T$$

다음으로 적응제어법칙은 " certainty equivalence principle "에 의하여 앞에서 구한 제어입력 u(t)에 θ(t)와 $y^0(t+d/t) = \phi^T(t)\theta$를 대입하여 구한다. KDI를 이용한 이상진단의 구성도는 그림 5-8과 같고, 이상의 영향은 예측모델의 변화로 나타난다.

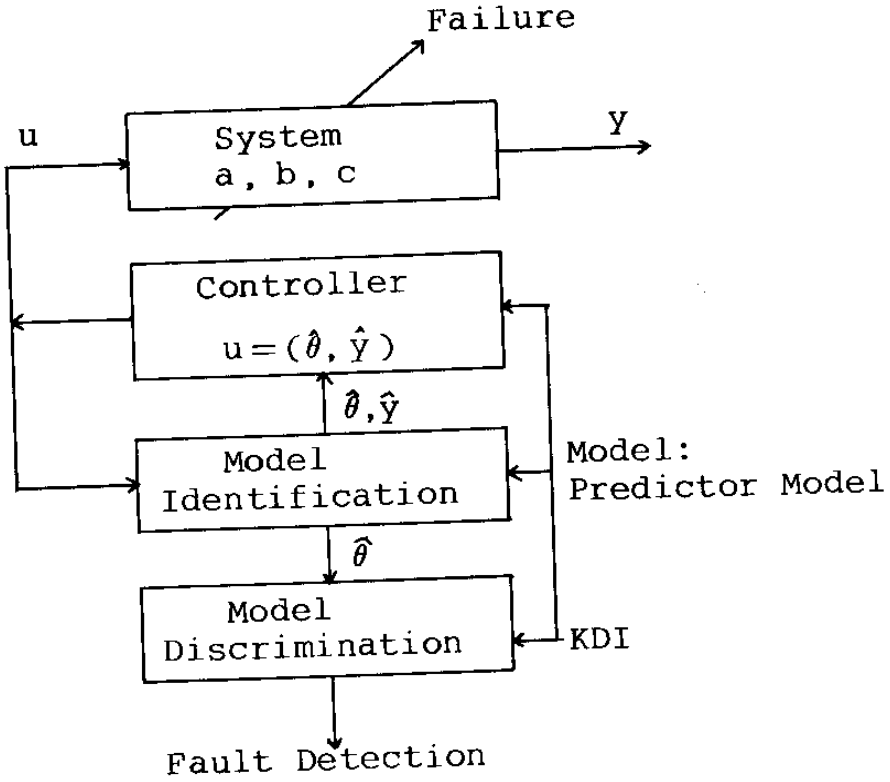


그림 5-8. 적응제어계에서 이상진단

그러나 이러한 현상은 적응메카니즘때문에 한정적으로 출력으로 관측되지 않는다. 매개변수가 느리게 변하는 공정에서 이것은 사실이나, 한편으로는 제어계에서 매개변수의 변화는 레귤레이타변수의 직접 감시로는 탐지하기가 어렵다. 이러한 상황에서 이상탐지를 위해서는 이상에 민감한 효과적인 탐지자(detection measure)가 필요하며, 이러한 탐지자로 KDI의 응용을 살펴보기 위하여 먼저 예측모델을 입출력모델로 바꾸어 쓰면 다음과 같다.

$$M(\theta): y(t) = G(q^{-1};\theta)\,v(t) - H(q^{-1};\theta)\,e(t)$$

$$e(t) = N(0, \sigma^2)$$

$$G(q^{-1}; \theta) = \left[\frac{q^{-d}\beta(q^{-1})}{1 - q^{-d}\beta(q^{-1})} \quad \frac{1 - C(q^{-1})}{1 - q^{-d}\alpha(q^{-1})} \right]$$

$$H(q^{-1}; \theta) = \frac{F(q^{-1})}{1 - q^{-d}\alpha(q^{-1})}$$

$$v(t) = \begin{bmatrix} u(t) \\ y(t/t+d) \end{bmatrix} \text{ 입력벡타}$$

여기서 적응예측 $y(t/t+d)$를 모델의 입력으로 이용하였다.

KDI를 이용한 제어계의 온라인 이상탐지방법을 살펴보기 위하여 그림 5-9 와 같은 두개의 구간에 대하여 I_1은 정상, I_2 는 이상구간으로 가정한다. 데이타의 수는 I_1에서 $k = 1, 2 \cdots, N$, I_2에서 $k = 1, 2, \cdots t$이며 현재 시간을 $k = t$로 하고, I_1 구간에서 최종추정치는 $\hat{\theta}_1$, I_2구간에서는 $\hat{\theta}_2$로 한다. 두 모델 $M(\hat{\theta}_1), M(\hat{\theta}_2)$ 사이의 차이를 KDI를 이용하여 온라인형태로 나타내면,

$$I_N^t[1,2] = \int \rho(Y_N/\hat{\theta}_1, V_{N-1}) \ell n \frac{\rho(Y_N/\hat{\theta}_1, V_{N-1})}{\rho(Y_N/\hat{\theta}_1, V_{N-1})} dY_N$$

$$= I_N^t[1,2]^{(2)} + I_N^t[1,2]^{(3)}$$

where, $I_N^t[1,2]^{(2)} = 1/2 \sum_{k=1}^{N} \| H_2^{-1}(G_1 - G^2) V(k) \|_A^{-2}$

$$I_N^t[1,2]^{(3)} = N/4\pi i \oint [H_2^{-1}(z) H_1(z) - 1] [H_2^{-1}(z^{-1}) H_1(z^{-1}) - 1] dz/z$$

$G_1 = G(q^{-1}; \hat{\theta}_1)$, $G_2 = G(q^{-1}; \hat{\theta}_2(t))$

$H_1 = H(q^{-1};\ \hat{\theta}_1)$, $H_2 = H(q^{-1};\ \hat{\theta}_2(t))$

$\Lambda = \sigma^2$ (주어진 상수로 가정)

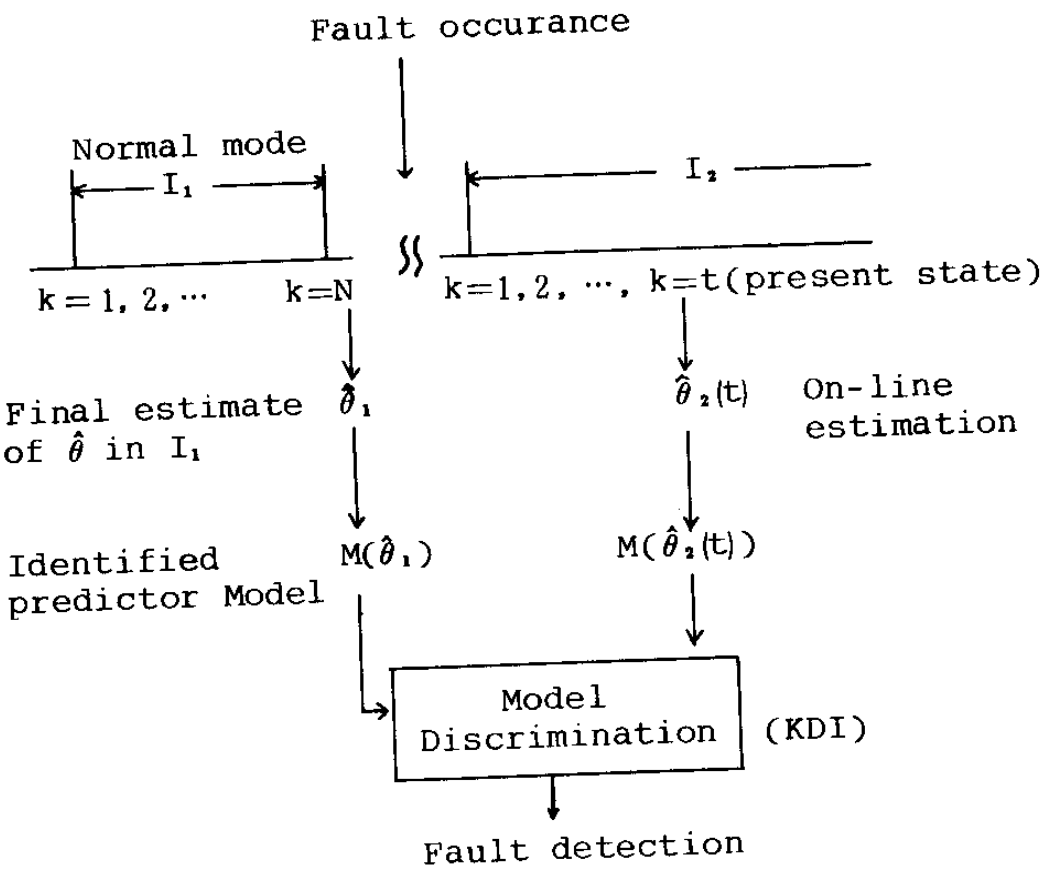


그림 5-9 . KDI를 이용한 온라인 이상탐지

이 연구에서는 2차계를 대상으로 KDI를 이용한 이상탐지 모사를 수행하여 그 결과가 우수함을 보였고, 공정변수가 느리게 변하는 시스템에 대해서는 KDI값의 증가경향으로부터 이상을 추론하였다.

제 3 절 파이프라인 시스템의 유출탐지

액체나 기체를 전송하는 파이프라인의 유출탐지는 안전성과 환경오염 그리고 경제적인 면에서 매우 중요한 문제이다. 그러나 이들 시스템을 감시할 수 있는 센서는 대개의 경우 파이프라인의 입출구 부위에서 유량과 압력을 측정하기 위한 것이 전부이고, 종래에는 이 값을 이용하여 수지식의 차이로 이상을 탐지하였으나 이와같은 방법은 측정잡음과 이것의 독특성때문에 액체의 경우는 전체흐름의 2 %, 기체는 10 %보다 작은 유출은 탐지하지 못하였다. 유출탐지방법을 알아보기 위하여 이에 따른 경우를 나누어 보면,

(i) Medium : liquid-gas-multiple phase

(ii) Operation : standstill - stationary - small change-unstationary

(iii) Size of leakage : large - medium - small(pipe burst)

(iv) Development of leakage : abrupt(cracking of welding seam); slow(hole corrosion) or already existing

(v) Leak monitoring : continuously-on request

파이프 내경축소문제는 앞에서 살펴보았고, 여기서는 참고문헌 [1]에 있는 정적운전 (stationary operation)시 연속감시와 적은 양의 유출탐지에 관한 탐지방법들을 알아보도록 한다.

1. 유출탐지방법

파이프라인에 대하여 물질 및 모멘텀수지식과 2차마찰법칙, 등온 기체상태방정식과 여러가지 모델을 단순화하기 위한 가정들을 이용하면 비선형 hyperbolic 편미분 방정식을 얻을 수 있다. 이식을 적당한 구간으로 나눈 파이프라인에 적용하면 상미분 방정식으로 변형할 수 있고, 또한 작은변화에 대해서는 테일러전개로 선형화하여 최종적으로 선형 상태공간모델을 얻을 수 있다.[1] 즉,

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

where, $x^T(t) = [\ \triangle M_0\ \ \triangle M_2 \cdots \triangle M_1\ \triangle p_1\ \triangle p_3 = \triangle p_{I-1}]$

$$u^T(t) = [\ \triangle P_{in}\ \triangle P_{out}\ ]$$

$$y^T(t) = [\ \triangle M_0\ M_I)$$

적은 양의 유출의 영향은 물질수지식에만 고려하여

$$\dot{x}(t) = Ax(t) + Lv(t) + Bu(t)$$

where, $v(t)^T = [\ 0\ 0 \cdots M_L \cdots 0 \cdots 0\ ]$

$$L = \begin{vmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 \\ \cdot & & \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot & & \cdot \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & & \cdot & 0 & \cdots & 0 \\ \cdot & & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ 0 & \cdots & g_{1(m-1)} & 0 & \cdots & 0 \end{vmatrix}$$

유출감시작업은 유출이 일어난 지점 Z_L과 유출량 M_L을 탐지하는 일로 구성되며, 측정은 입출구에서의 유량과 압력치가 대부분이다.

(1) Fault sensitive filters

이 방법은 상태변수를 예측하고, 예측오차 $\varepsilon(t) = y(t) - C\hat{x}(t)$를 계산하기 위해서 다음과 같은 상태변수 filter를 이용한다.

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + H[y(t) - C\hat{x}(t)]$$

첫단계에서는 filter가 고주파의 예측오차에 덜 민감하게 하기 위하여 filter이득을 가능한한 크게 준다. 그리고 나서 유출탐지를 위하여 이득을 줄여 준다. 유출이 발생하면 예측오차는 입구부에서는 증가하고 출구부에서는 감소하는 변화를 보이므로 이상의 유무를 이 값을 이용하여 판단할 수 있다. 이 방법의 단점은 유출에 의한 정보가 시간에 따라 없어지므로 filter가 오랫동안 유출에 영향을 받은 공정을 일치시키려는 불필요한 작업을 수행한다.

(2) Fault model filter

이 방법은 그림 5-10과 같이 filter에 이상의 영향을 모델링하여 유출 $v(t)$를 인식한다. 그러나 이 방법은 공정잡음이 있는 경우에는 'bank of filters'가 필요하게 되어 계산량이 많고 저장메모리를 많이 차지한다.

(3) Cross-correlation methed

파이프라인 시스템을 정적이고 정상상태운전으로 가정하고 입출구에서의 유량만 측정하는 경우에 유출감시를 널리 쓰이는 방법으로 정적수지식을 이용하는 것이 있다. 즉,

$$M_L(k) = M_0(k) - M_I(k)$$

이 값이 어떤 한계를 초과하면 경보를 발생한다. 그러나 이 방법은 측정잡음과 drifting measurement 등으로 인하여 적은 양의 유출은 탐지할 수 없다. 이 방법을 개선한 것으로 low-pass filtering을 이용하여 측정치의 저주파 성분만을 구별하여 이상을 판단하는 방법이 있다.

$$M_j^*(k) = \kappa_M M_j^*(k-1) + (1-\kappa_M) M_j^*(k), \quad (j = 0, 1)$$

기준치로 이용되는 $M_0^*(k)$, $M_1^*(k)$은 센서의 보정오차, 온도와 점도 등으로 인하여 다른 값을 가질 수 있다. 이 값들을 이용하면,

$$\triangle M_0(k) = M_0(k) - M_0^*(k)$$

$$\triangle M_1(k) = M_1(k) - M_1^*(k)$$

$$M_L(k) = \triangle M_0(k) - \triangle M_I(k)$$

여기에 low-pass filtering을 한번 더 취하면

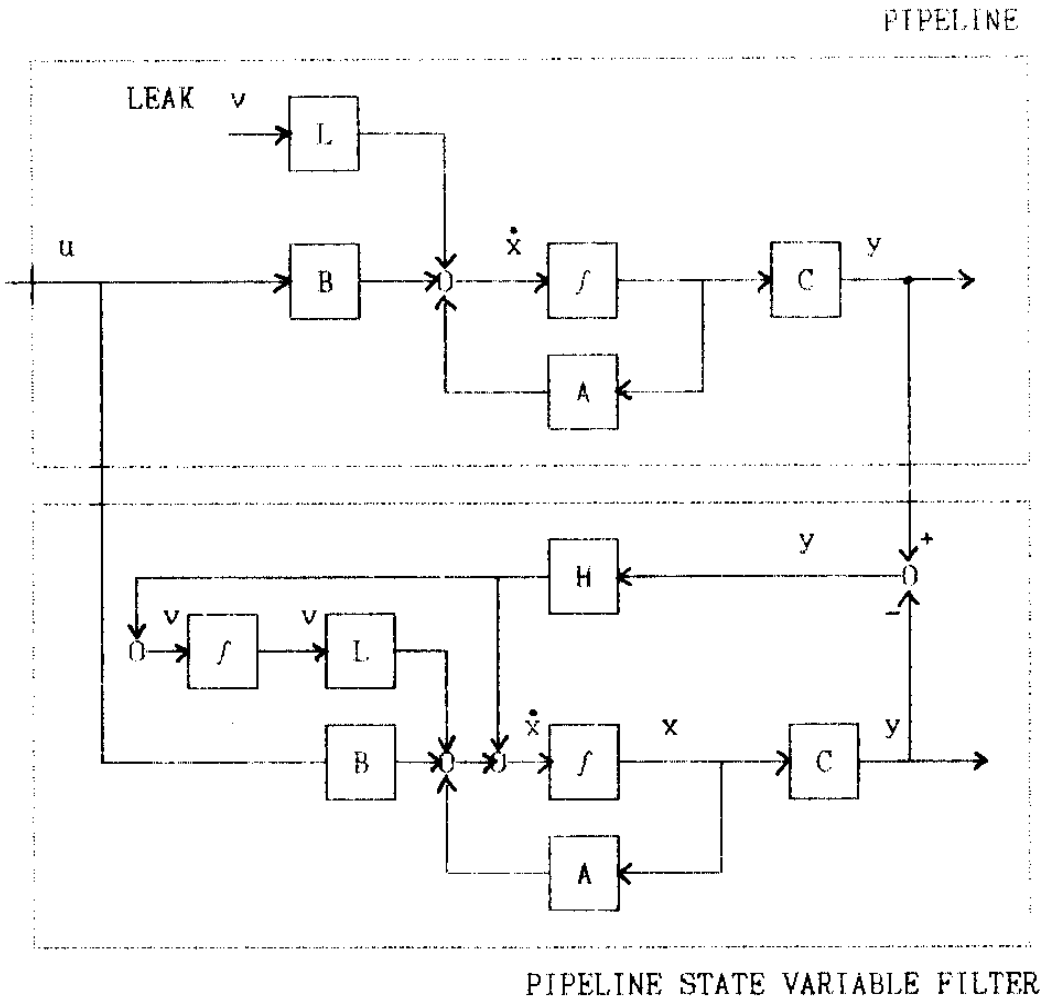


그림 5-10 . 유출영향모델을 가진 파이프라인 상태변수 filter

$$M_L''(k) = \kappa_L M_L'(k-1) + (1-\kappa_L) M_L'(k)$$

이때 급작스런 유출을 감지하기 위해서는 $\kappa_L < \kappa_M$을 만족하도록 해야 한다. 이렇게 함으로써 공정잡음과 느리게 표류하는 영향 등을 부분적으로 보완할 수는 있지만, 유체역학에 의한 유량의 변화는 빈번한 오경보를 막기위해 상대적으로 큰 경보발단치를 설정하게 만든다.

Cross-correlation함수를 이용한 방법은 이와같은 문제들을 모두 해결할 수 있으며, 모델이나 filter를 이용하지 않음으로 계산과 저장에 용이한 방법이다. Cross-correlation과 잡음영향의 축소를 위한 평균치는 각각 다음과 같이 계산된다.

$$\Phi_{MM}(\tau) = \frac{1}{N}\sum_{k=1}^{N} \triangle M_0(k-\tau)\ \triangle M_I(k)$$

$$\Phi_S = \frac{1}{2P+1}\sum_{\tau=1}^{P} \Phi_{MM}(\tau)$$

유출이 발생하면 $+\triangle M_0(k)$와 $-\triangle M_I(k)$의 변화로 이상신호가 생성되므로 이 두 양의 곱은 음이 되고, 따라서 Φ_S는 감소하며 어느 한계이하로 떨어지면 경보를 발생한다.

$$\Phi_S < \Phi_{SL}$$

Fading memory를 이용하여 cross-correlation 함수를 회귀적으로 계산하면

$$\Phi_{MM}(\tau, k) = \lambda\,\Phi_{MM}(\tau, k-1) + (1-\lambda)[\triangle M_0(k-\tau) - \triangle M_I(k)]$$

여기에 $0.9 < \lambda < 1.0$이고 그 값이 크면 잡음을 제거하는 데는 이로우나 경보가 늦다. 정적운전상태에서 유출지점은 압력곡선의 교점을 계산하여 추정할 수 있다. 즉, 난류(turbulent)의 압력기울기는

$$\frac{\partial p}{\partial z} = c_P = \frac{M^2}{p_0 - p_I}$$

이 값역시 회귀적으로 계산하면

$$c_j(k) = \kappa_c c_j(k-1) + (1-\kappa_c)\ \frac{M_j^2(k)}{p_0(k) - p_1(k)},\ (j = 0, 1)$$

기준치는 다음과 같이 계산된다.

$$M_j^{*2}(k) = c_j(k)\,[\,p_0(k) - p_1(k)\,],\ (j = 0, 1)$$

경보가 울리고 나면, c_0와 c_1를 고정시키고, 차이를 구한다.

$$\Delta_{Mj}^2(k) = M_j^2(k) - M_j^{*2}(k),\ (j = 0, 1)$$

유출지점은

$$z_L = L\left(1 - \frac{\sum \Delta_{M0}^2(k)}{\sum \Delta_{M1}^2(k)}\right)$$

이와같은 방법으로 길이 70 *km*, 내경 265 *mm*의 파이프라인시스템을 대상으로 35.8 *km*지점에서 액체 0.19 %의 유출이 발생한 경우 98초가 지나서 경보가 울리고, 이 후 90초만에 유출지점을 0.7 %의 거리오차 (500 m)로 추정하였다. 이 외에도 150 *km*의 기체운송관의 유출은 2 %의 경우에는 10시간, 5 %의 경우에는 3.5시간이 소요되며 유출지점과 유출양을 근사적으로 추정하였다고 보고 있다.

이상에서 살펴본 바와 같이 유출탐지방법들은 정밀한 정상 (normal)상태의 모델이나 기준치를 필요로 하고, 정밀한 모델을 구성할 수 없는 경우에는 cross-correlation과 같은 비변수 접근법이 효율적이며, 잡음의 영향을 효과적으로 처리해야 한다.

2. 모델링 및 모델검증

본 장에서 모델링과 모델인식 방법에 의한 공정의 이상진단에 관한 이제까지의 연구를 폭넓게 살펴본 결과 본 1차년도 연구에서는 대상공정을 파이프라인 시스템으로 정하고 이의 유출진단 방법으로 cross-correlation 함수를 이용한 접근법을 택하였다. 따라서 1차년도에서는 기체와 액체의 경우에 대하여 각각 모델링을 행하고 액체의 경우에 전산기 모사로 모델검증과 이 모델을 공정으로 두고 cross-corelation 함수를 이용한 유출진단 모사를 행하였다. 참고로 실험적 검증을 위하여 2차년도에 구성하려는 실험장치의 사양은 길이 150 m, 내경 10 ㎜이고 입출구에 유량과 압력을 측정할 수 있는 센서들과 토출펌프 및 시스템을 여기시켜 주기 위한 solenoid 밸브 등으로 구성할 예정이다.

(1) 기체가 흐르는 파이프라인의 수학적 모델링

파이프라인 시스템은 수학적으로 잘 정의된 계로서[10] 쉽게 모델식을 구할 수 있다. 먼저 물리적 계수를 정의하면,

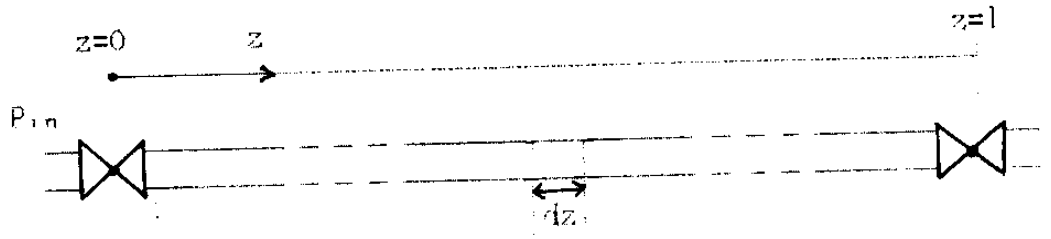


그림 5-11. 파이프라인 시스템의 개략도

L : 파이프라인의 총 길이
d_F : 파이프 길이

A_F : 파이프의 단면적
H : 파이프라인의 높이
p : 유체압력
ρ : 유체밀도
w : 유체속도
M : 유체질량
$\dot{M}$: 유체질량흐름
c_F : 소리의 속도
λ : 마찰계수

위 그림의 미소 구간 dz 에서 물질 수지식과 연속방정식은

$$\frac{\partial M}{\partial t} = A_F w\rho - A_F\left(w + \frac{\partial w}{\partial z}dz\right)\left(\rho + \frac{\partial \rho}{\partial z}dz\right)$$

$$\dot{M} = A_F w\rho$$

미소한 항을 무시하고 정리하면,

$$\frac{\partial}{\partial z}(w\rho) + \frac{\partial \rho}{\partial t} = 0$$

모멘텀 수지식도 정리하면

$$\frac{\partial}{\partial t}(w\rho) + \frac{\partial}{\partial z}\left(p + \frac{w^2\rho}{2}\right) = -F - Y$$

where,

$$F = \frac{\partial p_F}{\partial z} = \frac{\lambda \rho}{2 d_F} w|w| \quad ; \text{ friction term}$$

$$Y = g\rho \frac{dH}{dz} ; \text{ static pressure term}$$

등온 가정을 도입하면 기체상태방정식을 이용하여 문제를 간략하게 바꿀 수 있다.

$$\frac{p}{\rho} = Z(p, T_0) R T_0 = C_F^2(p)$$

여기서 $C_F(p)$ 는 소리의 속도이며, 위 관계식을 수지식에 대입하면 $p(z,t)$ 와 $M(z,t)$ 로 식을 표현할 수 있다. 또한 다음과 같은 가정을 도입하여 긴 파이프라인의 모델식을 hyperbolic 편미분 방정식으로 구할 수 있다.

가정 1 : 파이프라인 구간 j내에서 등온인 경우 소리의 속도는 일정하다. $C_F(p) = c_{Fj}$

가정 2 : 유체의 속도가 소리의 속도에 비해 아주 작다. 따라서

$$w_F^2 \ll C_F^2$$

$$\frac{M^2 C_F^2}{A_F^2 P^2} = \frac{w_F^2}{C_F^2} \simeq 0$$

가정 3 : 긴 파이프라인에서 다음 항은 무시한다.

$$\frac{MC_F^2}{A_F^2 P}\ \frac{\partial M}{\partial z} \simeq 0$$

따라서,

$$\frac{A_F}{C_{Fj}^2}\ \frac{\partial p}{\partial t} + \frac{\partial M}{\partial z} = 0$$

$$\frac{1}{A_F}\ \frac{\partial M}{\partial t} + \frac{\partial p}{\partial z} = \frac{\lambda C_{vj}^2}{2d_F A_F}\ M\ \frac{|M|}{p}$$

이와 같은 편미분방정식의 해는 파이프라인을 N개의 구간으로 나누어 상미분방정식으로 변환한 다음 다음의 밸브방정식을 경계조건으로 하여 구할 수 있다.

$$M_0 = c_{vo}(P_{in} - p_o)^{1/2}$$

$$M_1 = c_{vI}(p_I - p_{ex})^{1/2}$$

Compressible liquid 의 경우에는 M|M| 의 계수가 다음과 같이 상수가 된다.

$$\frac{\lambda C_{Fj}^2}{2d_F A_F p_j} = \frac{\lambda}{2d_p A_F \rho}$$

적은 양의 유출 M_L 은 물질 수지식에 다음과 같이 고려된다.

$$\frac{A_F}{C_{Fj}^2}\frac{\partial p}{\partial t}+\frac{\partial M}{\partial z}+\frac{\partial M_L}{\partial z}=0$$

(1) 액체가 흐르는 파이프라인의 수학적 모델링

물과 같이 incompressible fluid로 가정할 수 있는 액체가 흐르는 파이프라인의 수학적 모델은 정상(normal) 상태에서 모멘텀 수지식 만을 구성할 수 있으므로 이상진단용으로 사용하거나 본 연구에서와 같이 실제 공정의 모사용으로 이용하기 위해서는 참고문헌〔3〕에 있는 바와 같이 그림 5-5의 파이프라인 구간에 대하여 액체 유선을 따르는 특성방정식을 풀어야 한다. 그림 5-5의 각 구간에서 특성방정식은 다음과 같다.

$$Q_0^{k+1}=Q_1^k-\frac{fL}{2d_FA_FN}Q_1^{k2}+\frac{gA_F}{a}(H_0^{k+1}-H_1^k)$$

$$Q_1^{k+1}=(Q_1^k+Q_2^k)/2-\frac{fL}{4d_FA_FN}(Q_0^{k2}+Q_2^{k2})+\frac{gA_F}{2a}(H_0^k-H_2^k)$$

...

$$Q_N^{k+1}=Q_{N-1}^k-\frac{fL}{2d_FA_FN}Q_{N-1}^{k2}+\frac{gA_F}{a}(H_{N-1}^k-H_N^k)$$

$$H_1^{k+1}=\frac{a}{2gA_F}(Q_0^k-Q_2^k)-\frac{fL}{4gd_FA_F^2N}(Q_0^{k2}-Q_2^{k2})+(H_0^k+$$

$$H_2^k)/2\ \cdots$$

$$H_{N-1}^{k+1} = \frac{a}{2gA_F}(Q_{N-2}^{k} - Q_N^{k}) - \frac{fL}{4gd_F A_F^2 N}(Q_{N-2}^{k2} - Q_N^{k2}) +$$

$$(H_{N-2}^{k} + H_N^{k})/2$$

이 특성방정식들을 상태공간방정식으로 변환하기 위하여 다음과 같이 변수치환과 상태벡타 및 입력벡타를 정의한다.

$$B = a/gA_F, \quad R = fL/2gd_F A_F N$$

where, a : wave speed in fluid
g : gravity acceleration
f : friction factor

$$x(k) = [Q_0\ Q_1 \cdots Q_N\ H_1\ H_2 \cdots H_{N-1}]^T$$

$$x(k)^2 = [Q_0^2\ Q_1^2 \cdots Q_N^2\ H_1^2\ H_2^2 \cdots H_{N-1}^2]^T$$

$$u(k) = [H_{0k}/2\ 0 \cdots 0\ H_{Nk}/2$$

$$H_{0k+1}/B\ \ H_{0k}/2B\ 0 \cdots 0\ -H_{Nk}/2B - H_{Nk+1}/B]^T$$

이에 따라 특성방정식은 다음과 같은 비선형 공간상태방정식으로 변환된다.

$$x(k+1) = Ax(k) + Mx(k)^2 + u$$

여기서 행렬 A와 M은

$$A = \left[\begin{array}{c|c} \begin{matrix} 0 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1/2 & 0 & 1/2 \\ & & & 1/2 & 0 \end{matrix} & \begin{matrix} B/2 & 0 & -B/2 & & \\ 0 & B/2 & 0 & -B/2 & \\ & \ddots & \ddots & \ddots & \\ & & & B/2 & 0 & -B/2 \end{matrix} \\ \hline \begin{matrix} -1/B & & & & \\ 0 & -1/2B & & & \\ 1/2B & 0 & -1/2B & & \\ & \ddots & \ddots & \ddots & \\ & & 1/2B & 0 & -1/2B \\ & & & 1/2B & 0 \\ & & & & 1/B \end{matrix} & \begin{matrix} 0 & 1 & & \\ 1/2 & 0 & 1/2 & \\ & \ddots & \ddots & \ddots \\ & & 1/2 & 0 & 1/2 \\ & & & 1 & 0 \end{matrix} \end{array}\right] \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ (n-1) \\ \\ \\ \\ \end{matrix}$$

$$\begin{matrix} \\ \\ \\ \\ \\ \\ (n-1) \\ \\ \\ \\ \end{matrix} \qquad \begin{matrix} \\ \\ \\ \\ \\ \\ (n+1) \\ \\ \\ \\ \end{matrix} \times (n+1)$$

$$M = \left[\begin{array}{c|c} 0 & \begin{matrix} -R/2 & 0 & R/2 & & \\ & -R/2 & 0 & R/2 & \\ & & \ddots & \ddots & \ddots \\ & & & -R/2 & 0 & R/2 \end{matrix} \\ \hline \begin{matrix} 0 & -R/B & & & \\ -R/2B & 0 & -R/2B & & \\ & \ddots & \ddots & \ddots & \\ & & -R/2B & 0 & -R/2B \\ & & & -R/B & 0 \end{matrix} & \end{array}\right] \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ (n-1) \\ \\ \\ \\ \end{matrix}$$

$$\begin{matrix} \\ \\ \\ \\ \\ \\ (n-1) \\ \\ \\ \\ \end{matrix} \qquad \begin{matrix} \\ \\ \\ \\ \\ \\ (n+1) \\ \\ \\ \\ \end{matrix} \times (n+1)$$

유체의 유출은 유출지점을 나타내는 벡터 L과 양(q)을 모델에 선형적으로 첨가할 수 있다.

$$x(k+1) = Ax(k) + Mx(k)^2 + u + qL$$

이 모델을 공정으로 모사하기 위하여 모델확인은 파이프라인 입구에서의 압력에 5%의 랜덤시퀀스인 PRBS(Pseudo random binary sequence)를 부가하고 출력의 영향을 살펴보았다. 그림 5-12는 입력 시퀀스와 그 때의 출력변화를 모사한 결과이다. 이 상진단시에 측정잡음은 측정치의 5%의 분산크기를 갖는 백색잡음을 얹어 주었다. 그림 5-13은 120 m의 파이프라인의 40 m 지점에서 0.8%의 유출이 발생한 경우에 입출구에서 압력과 유속의 변화양상이다. 이것은 참고문헌〔1〕에서 실제 파이프라인의 측정결과와 일치하는 경향을 보인다. 즉 입구에서 압력은 감소하고, 유속은 증가하며 출구에서의 유속은 감소한다.

이상의 모사를 통하여 비록 정적 모델이지만 이 모델을 공정으로 모사가 가능함을 확인할 수 있다.

3. 공정신호분석을 이용한 파이프라인 유출탐지모사

앞에서 구성한 파이프라인의 정적 모델을 대상으로 cross-correlation 방법을 이용한 유출탐지모사를 수행하였다. 모사를 수행한 계는 길이 120 m, 내경 10 ㎜이며 입구압력은 수두로 55 m이고, 유출지점은 40 m지점, 유출량은 0.8%로 취하였다. Cross-correlation 방법을 수행하는데 이용한 변수 값은 다음과 같다.

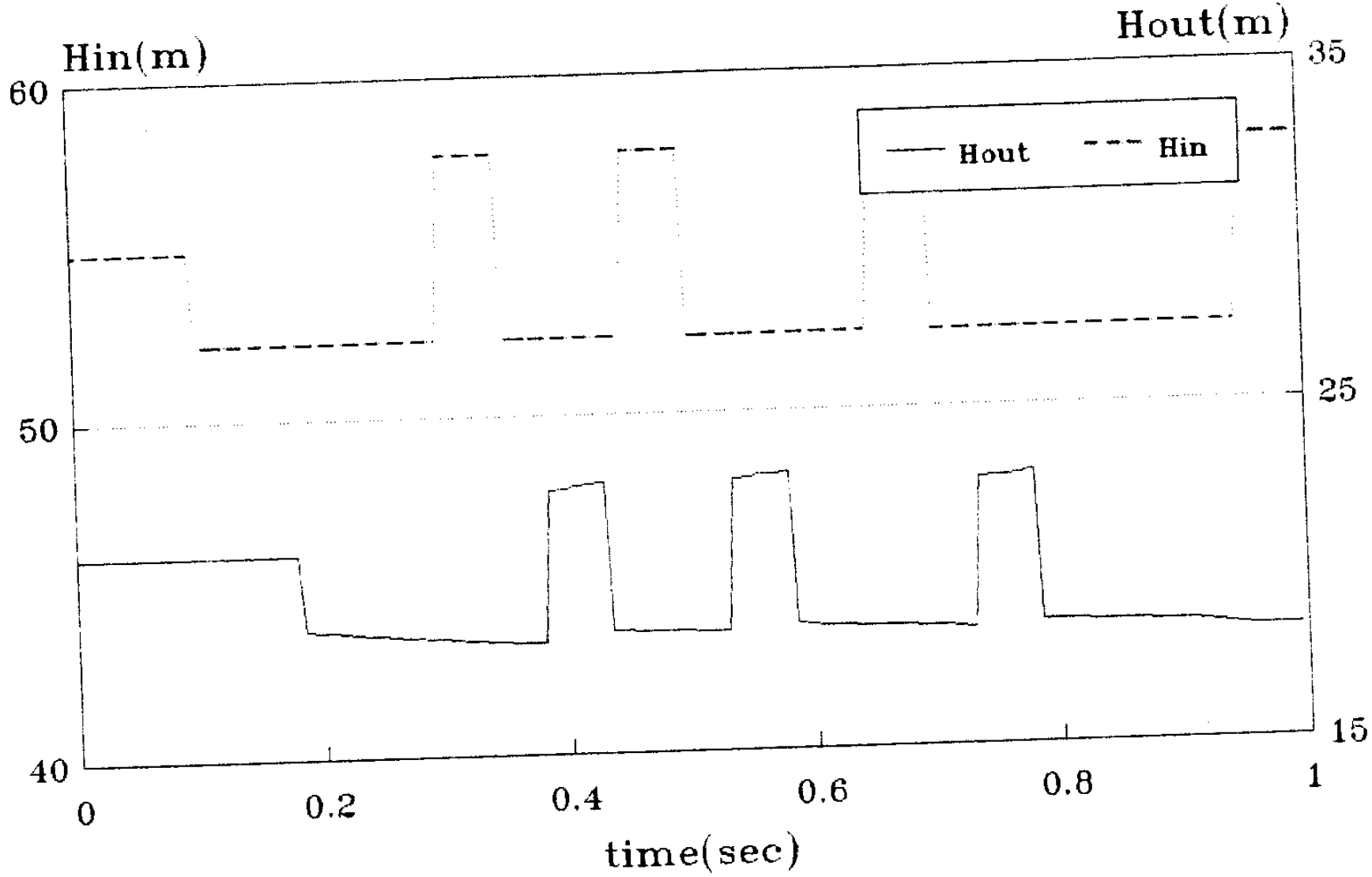


Fig.5.12. Verification of real time model.

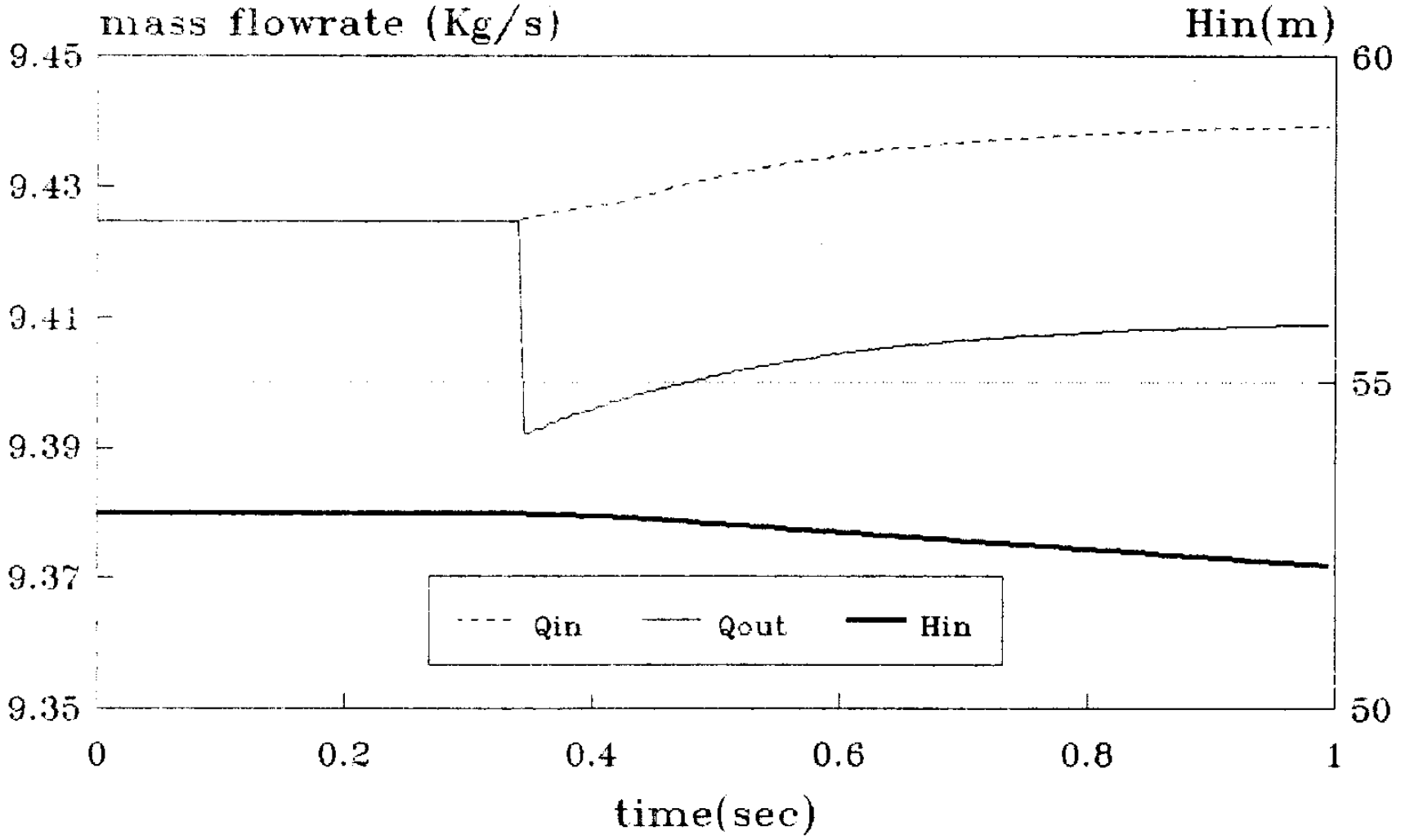


Fig.5-13. Transient of Hin, Qin and Qout after occurence of a leak at time t=0.3s at 40m with leakage ratio 0.8%.

$\kappa_L = 0.925$, $\lambda = 0.95$, $\phi_{SL} = -9.0 \times 10^{-16}$, $p = 20$

이와 같은 조건에서 수행한 유출탐지 결과는 그림 5-14 와 같다. 유출이 발생하고 0.19s 후에 경보가 발생하고, 이 후 0.2s 가 경과 하고 나서 유출지점을 탐지하는 결과를 볼 수 있다.

이상의 간단한 모사를 통하여 여러가지 면에서 중요하게 여겨지는 파이프라인의 진단문제를 다루어 보았다. 앞에서 살펴본 바와 같이 모델을 이용하지 않는 cross-correlation 함수와 같은 비변수 접근법으로도 좋은 결과를 얻을 수 있음을 알 수 있다. 그러나 극소량의 유출이나 급속한 탐지를 위해서는 동적모델과 filtering 기법을 이용하는 방법이 효과적일 것으로 생각된다.

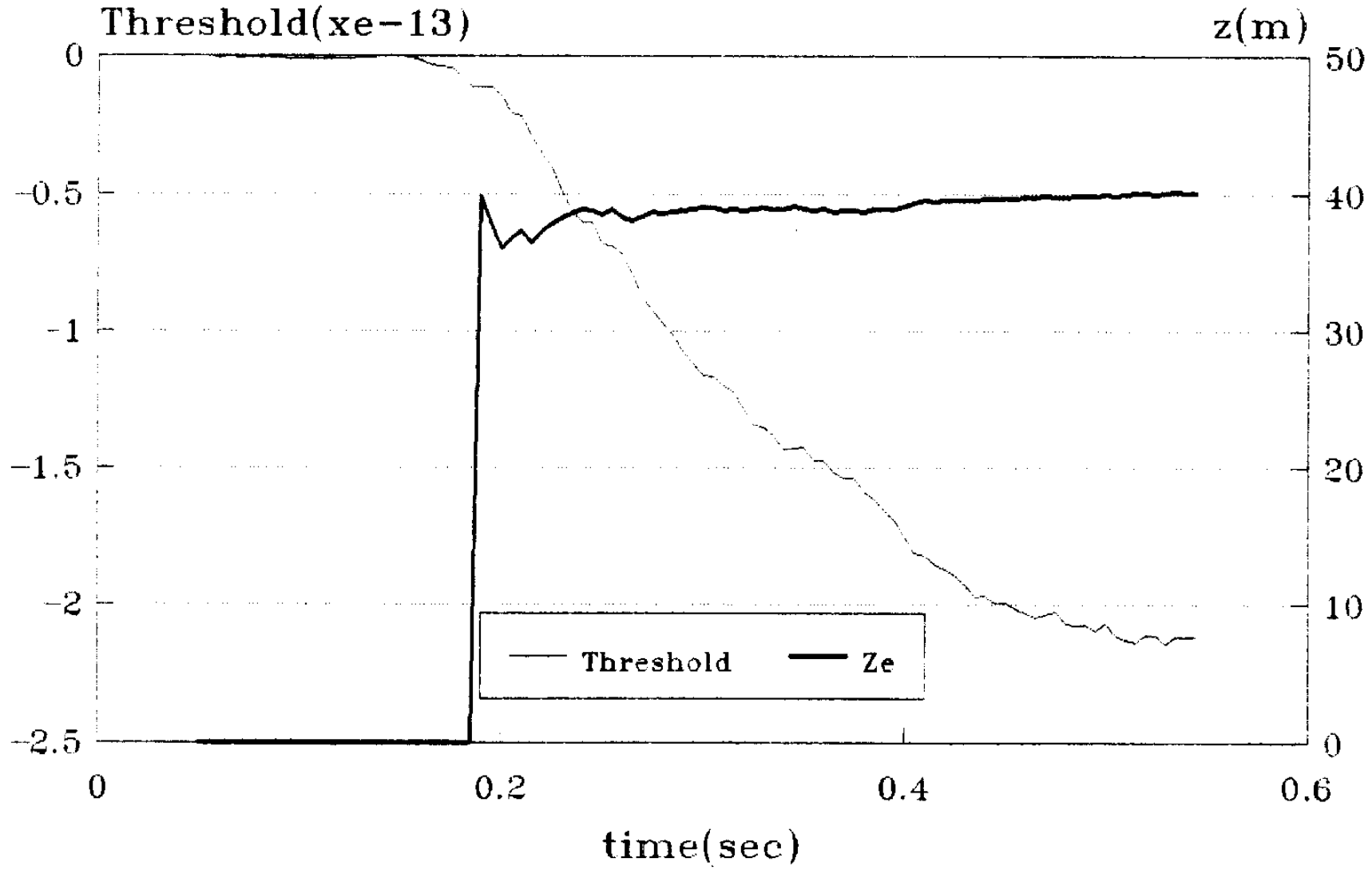


Fig.5-14. Threshold and Detected leakage locational.

참 고 문 헌

(1) R. Isermann, "Process fault detection based on modeling and estimation method-a survey", Automatica, 20, 387, 1984.

(2) S. Tanaka and T. Okita, "Failure detection in linear discrete dynamical systems and its detectability", Preprints of the 10th IFAC World Congress, 3, 74, 1987.

(3) L. Tao and C. Fang, "Diagnosis of a class of faults in distributed parameter systems', 10th IFAC, 3, 80, 1987.

(4) A.S. Wiilsky, "A survey of design methods for failure detection in dynamic systems", Automatica, 12, 601, 1976.

(5) K. Uosaki and M. Kawagoe, "Backward SPRT detection system for detection of innovation variance change", 11th IFAC. 1597, 1988.

(6) L. Zhi, "Fault diagnosis of dynamic system by means of parameter estimation", 11th IFAC, 1602, 1988.

(7) E.B. Wilie, "The microcomputer and pipeline transients", J. of Hydraulic Engineering, 109, 1723, 1983.

(8) G. Geiger, "Monitoring of an electrical driven pump using continuous-time parameter estimation methods", 6th IFAC, Pergamond Press, Oxford, 1982.

(9) K. Kumamaru, T. Soderstrom, S. Sagara, and K. Morita, "On-line fault detection in adaptive control systems by using Kullback discriminatiin index", 11th IFAC, 1578, 1988.

(10) R. Bird, W.E. Stewart and E.N. Lightfoot, Transport Phenomena, John Wiley & Sons, 1960.

제 6 장 Fuzzy Logic 제어방법

제 1 절 서 론

산업현장에서 실제 조업되고 있는 공정을 자동적으로 제어하기 위해 관련 각 분야에 대한 연구가 활발히 진행되고 있다. 대부분의 자동화된 공정들은 PLC(Programmable Logic controller)와 DDC(Direct Digital Control)의 복합적인 형태의 운전방법을 채택하는 추세로 발전되어가고 있다. 그러나 자동제어와 관련된 이론의 대부분은 정확한 공정의 모델을 근거로하기 때문에 공정의 모델이 불확실한 경우에는 실제적인 적용이 어려운 경우도 있다. 그리고 자동제어기에 의한 공정의 조업결과가 숙련된 오퍼레이터에 의존한 경우에 미치지 못하는 경우도 많다. 주로 수학적인 모델링이 어려운 공정일수록 자동제어기의 적용이 더욱 어렵다. 이러한 이유로 공정에 대한 더욱 엄밀한 수학적인 모델링에 대한 연구와 더불어 숙련된 오퍼레이터의 경험과 의사결정 행위를 분석하여 이를 체계화하고 실제 조업에 적용할 수 있도록 소프트웨어화하는 전문가시스템에 대한 필요성은 날로 증가되고 있다.

제어용 전문가시스템의 일환으로 제어관련 전문가의 경험과 의사결정행위를 체계화한 FLC(Fuzzy Logic Control)가 개발되었고 이로 인해 기존의 자동제어기의 작동 영역은 한층 더 확장되었다. FLC 는 공정에 대한 수학적인 모델에 의존하지 않고 인간의 경험과 직관을 조업에 활용할 수 있도록 한다. 회분식 반

응기, 용광로 및 시멘트 소성로등의 공정은 공정의 비선형성이 심하고 공정의 진행에 따라 공정 파라메터가 급격하게 변화되고 측정가능한 변수가 극히 제한되어 있기 때문에 기존의 자동제어방법으로는 조업의 질을 향상시키기 어렵다. 또한 공정의 물리적, 화학적 성질을 반영한 모델을 근거로 관련 파라메터를 추정하는 접근방식도 계산량과 소요시간의 증가로 실시간 조업에 적용되는 데에는 많은 문제점을 안고 있다. 제반 여건의 변화에 의해 조업조건의 변화가 불가피한 경우나 공정에 불규칙한 외란이 도입되는 경우에도 기존의 제어방법을 성공적으로 적용시키지 못하는 경우가 많다. 그러나 FLC 의 이용으로 이러한 문제점이 단계적으로 해결되어 가고 있다.

1973 년 Zadeh[1] 의 연구결과를 근거로 1974 년 Mamdani[2] 는 스팀 엔진의 제어에 최초로 FLC 를 응용하였다. 이후 다양한 분야의 공정제어에 FLC 를 응용한 연구결과가 발표되었으며 이중 대표적인 것들은 표 6-1 에 요약된 바와 같다. 또한 공정제어 이외의 분야에서도 퍼지 알고리듬을 이용하고자하는 다양한 연구가 진행되고 있다.

일반적인 전문가시스템과 FLC 는 인간의 경험과 직관을 체계적인 방법론을 도입하여 프로그래밍한다는 점에서 유사하다. 그러나 FLC 는 다음과 같은 특징을 갖는다.

* FLC 의 지식표현은 주로 RB(Rule-Base) 로 한다.
* 일반적인 전문가시스템 보다 그 작동 영역이 좁다.

표 6-1. FLC 응용 사례

제어대상공정	측정변수	조작변수	참고문헌
Steam Engine	steam pressure in the boiler, speed of the engine	heat input to the boiler, throttle opening	Mamdani[2], 1974
Activated Sludge Process	total BOD, suspended solids in the (effluent, sludge leaving the aeration tanks, recycled sludge) ammonia-N in the effluent, bulking sludge, rising sludge, DO set point, waste sludge flow rate	DO set point change, recycled sludge flow rate set point change, waste sludge flow rate change	Tong et al.[3], 1980
Cement Kiln	drive load gradient, drive load, free lime content	burning zone temperature	Holmblad et al.[4], 1982 King[5], 1982

* RB 의 구성은 FLC 설계자에 의해 이루어진다.

본장에서는 제어용 전문가시스템 개발을 위해 정성적인 경험 법칙을 프로그램화할 수 있는 방법으로 최근 많은 연구의 대상이 되고 있는 퍼지집합이론(Fuzzy Set Theory)과 이를 이용한 제어기 설계방법, 그리고 퍼지이론에 따른 모델링 방법에 대하여 설명하고 이를 이용하여 제어용 전문가시스템의 기본 모듈을 개발하였다.

제 2 절 퍼지집합이론(Fuzzy Set Theory)

1. 개 요

퍼지집합이론은 그 역사가 매우 짧다. 따라서 관련된 각종정의와 표기방법이 아직은 정착되지 않은 상태이다. 다음은 퍼지집합이론과 관련된 용어 정의를 중심으로 퍼지집합이론을 설명한다.

일반적인 집합이론에 따른 집합은 그 존재영역 엄밀하게 정의된다. 이와 달리 퍼지집합(fuzzy set)은 그 존재영역이 엄밀하게 정의되지 않는다. 예를 들면 값이 비싼 자동차의 집합, 매우 큰 수의 집합, 젊은 사람의 집합등은 기존의 집합이론으로는 정의할 수 없는 것들이었으나 퍼지집합이론은 이러한 성질의 집합에 대하여 구성원소가 얼마나 집합원소로 타당한가하는 멤버쉽 함수(membership function)를 정의하고 주관적인 판단에 의해 함수값을 각 구성원소에 할당함으로써 집합으로 표현할 수 있도록 한다. 퍼지집합이론은 사람이 일상적으로 사용하는 언어변수(lingui-

stic variable) 즉, "크다", "적다", "적당하다", "좋다" 등의 표현을 정량화하여 표현할 수 있도록 한다. 퍼지집합 A는 일반적으로 식(6-1)과 같이 정의된다.

$$A = \{(x, \mu_A(x)) | x \in X\} \quad \cdots\cdots (6\text{-}1)$$

이때 $\mu_A(x)$를 멤버쉽 함수라 한다. 예를 들어 퍼지집합 A를 10에 매우 가까운 수의 집합으로 정의하고 퍼지집합 A에 대하여 $\mu_A(x)$를 식(6-2)와 같이 정의한 경우,

$$\mu_A(x) = (1 + (x-10)^{-2})^{-1} \quad \cdots\cdots (6\text{-}2)$$

x에 대한 $\mu_A(x)$의 변화는 그림 6-1과 같다.

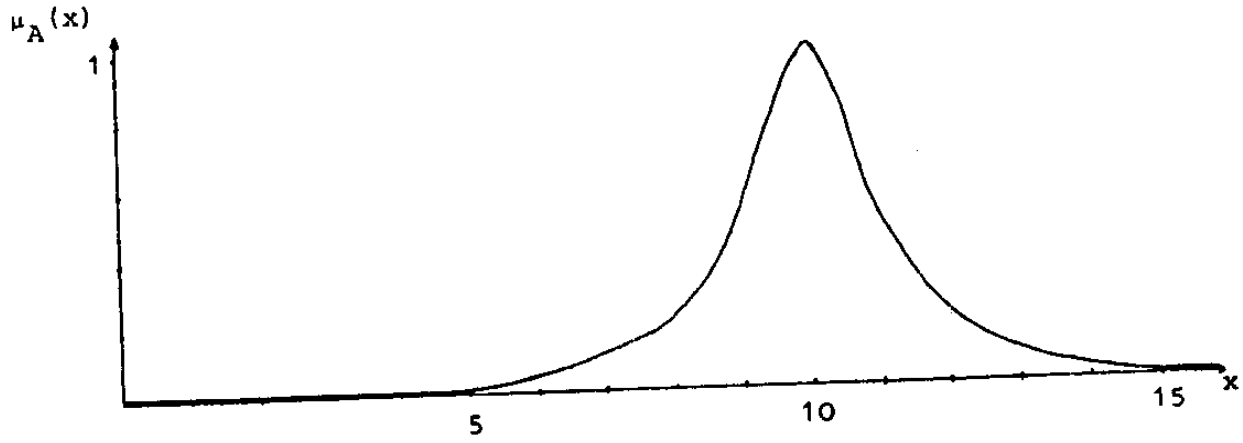


그림 6-1 Membership 함수의 예

그림 6-1에서 볼 수 있듯이 $\mu_A(x)$는 구간[0,1]의 값을 가지는 것이 일반적이다. 이런 경우를 특별히 'normal fuzzy set'이라 한다. 멤버쉽 함수의 표현방법은 그림 6-1과 같이 연속적인 함수 값을 가지는 경우도 있지만 표 6-2와 같이 'look-up table'

형태로 불연속적인 값을 갖는 경우도 있다.

표 6-2 Look-Up Table 의 예

	-4	-3	-2	-1	0	+1	+2	+3	+4
LP	0	0	0	0	0	0	0	0.6	1
SP	0	0	0	0	0	0.6	1	0.6	0
ZE	0	0	0	0.6	1	0.6	0	0	0
SN	0	0.6	1	0.6	0	0	0	0	0
LN	1	0.6	1	0	0	0	0	0	0

멤버쉽 함수의 값을 좀더 합리적으로 객관적인 근거에 따라 할당하고자 하는 연구도 많이 진행되고 있으나 자체의 성질이 매우 주관적이고 문제에 따라 다양하게 정의될 수 있기 때문에 큰 성과를 거두지는 못하고 있는 실정이다. 한편으로는 식(6-2)나 'Look-up table'형태의 예와 달리 멤버쉽 함수의 값 자체도 퍼지변수(fuzzy variable)로 표현되는 경우도 있다. 예를 들면 멤버쉽 함수의 값이 "약 0.6" 혹은 "매우 작다"등과 같은 경우이다. 이러한 경우의 퍼지집합을 'ultrafuzzy set'이라 정의한다.

기존의 집합이론에 따라 정의된 집합도 퍼지집합이론으로 표현될 수 있다. 정수 전체 집합 X에 대하여 식(6-3)과 같이 집합 B가 정의된 경우,

$$B = \{x | 0 \leq x \leq 100,\ x \in X\} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}3)$$

$\mu_C(x)$ 값이 0 혹은 1만 존재하는 식(6-4)의 퍼지집합 C와 식(6-3)의 집합 B는 동일하다.

$$C = \{(x, \mu_C(x)) \mid x \in X\} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}4)$$

$$\mu_C(x) = \begin{cases} 0, & x > 100 \\ 1, & 0 \leq x \leq 100 \end{cases}$$

일반적인 경험법칙의 표현은 'if-then' 형태로 표시된 'fuzzy implication'으로 표시한다. 제어법칙을 표현하는 "If PE is No, C PE is PS, Then HC is PS"와 같은 'fuzzy implication'의 if 절에 해당되는 부분을 전제(premise)라 하고 then 절을 결론(consequence)이라 한다.

2. 기본연산

기존의 집합 연산과 유사한 기본적인 성질들은 다음과 같다.

성질 1. 항등관계

퍼지집합 A와 B에 속하는 모든 구성원소 x에 대하여 각각의 멤버쉽 함수의 값이 같으면 집합 A와 B는 같다.

즉, if $\mu_A(x) = \mu_B(x)$, $\forall x$, $x \in A$ and $x \in B$

then $A = B$

성질 2. 여 집 합

집합 A의 'complement'를 A^C 라 하면 이런 경우 멤버쉽 함수는 다음 식(6-5)와 같이 정의된다.

$$\mu_A{}^c(x) = 1 - \mu_A(x) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}5)$$

성질 3. 공 집 합

집합 A에 모든 구성원소 x에 대하여 멤버쉽 함수의 값이 0 이면 집합 A는 공집합이다.

성질 4. 전체집합

집합 A에 모든 구성원소 x에 대하여 멤버쉽 함수의 값이 1 이면 집합 A는 전체집합이다.

성질 5. 포함 관계

퍼지집합 A와 B에 속하는 모든 구성원소 x에 대하여 집합 A의 멤버쉽 함수값이 집합 B의 멤버쉽 함수 값보다 언제나 작으면 집합 A는 집합 B에 속한다.

즉, if $\mu_A(x) \leq \mu_B(x)$, $\forall x$, $x \in A$ and $x \in B$

then $A \subset B$

성질 6. 합 집 합

퍼지집합 A와 B의 합집합 C에 대한 멤버쉽 함수는 식 (6-6)과 같이 집합 A와 집합 B에 대한 각각의 멤버쉽 함수값의 최대값이 된다.

즉, if $C = A \cup B$

then $\mu_C(x) = \max[\mu_A(x), \mu_B(x)]$ ..... (6-6)

성질 7. 교 집 합

퍼지집합 A와 B의 교집합 C에 대한 멤버쉽 함수는 식 (6-7)과 같이 집합 A와 집합 B에 대한 각각의 멤버쉽 함수값의 최소값이 된다.

즉, if $C = A \cap B$,

then $\mu_C(x) = \min [\mu_A(x), \mu_B(x)]$ ..... (6-7)

3. 추론방법

퍼지집합이론에 따라 법칙을 'fuzzy implication'으로 표현한 경우 이에따른 결론을 추론하는 방법은 다음과 같이 두가지로 구분된다.

* 추론합성법칙(compositional rules of inference)의 이용
* 퍼지 논리(fuzzy logic)의 이용

가. 추론합성법칙

퍼지집합 A와 B에 대한 멤버쉽 함수를 각각 A(x) 와 B(x) 로 정의하고 다음과 같은 두가지의 'implication' R_1과 R_2가 존재하는 경우,

R_1: if x_1 is A_{11}, x_2 is A_{12}, then y is B_1

R_2: if x_1 is A_{21}, x_2 is A_{22}, then y is B_2

주어진 x_1^0 및 x_2^0에 대하여 R_1과 R_2의 전제에 대한 참

값을 w_1 과 w_2라 하면, 이때 w_1 과 w_2 는

$$w_1 = A_{11}(x_1^0) \wedge A_{12}(x_{12}^0) \quad \cdots\cdots (6\text{-}8)$$

$$w_2 = A_{11}(x_1^0) \wedge A_{22}(x_{12}^0) \quad \cdots\cdots (6\text{-}9)$$

로 계산하거나, 혹은

$$w_1 = A_{11}(x_1^0) * A_{12}(x_{12}^0) \quad \cdots\cdots (6\text{-}10)$$

$$w_2 = A_{21}(x_1^0) * A_{22}(x_{12}^0) \quad \cdots\cdots (6\text{-}11)$$

로 계산할 수 있다.

이때 식(6-8)과 (6-9)의 $\wedge$ 는 앞에서 교집합의 성질에 대하여 설명한 바와 같이 최소값을 계산하는 연산자이다. 이와 같은 w_1 과 w_2 를 계산하는 위의 두가지 방법중 최근에는 후자의 경우가 더욱 널리 이용된다. 따라서 R_1 과 R_2 의 'implication' 은 결론으로 w_1B_1 과 w_2B_2 의 멤버쉽 함수를 추론한다. 두가지의 'implication' 사이에는 합집합의 관계 즉, 논리곱(or) 의 관계가 성립됨으로 새로운 퍼지집합 B^* 를 다음의 식(6-12) 와 같이 정의할 수 있다.

$$B^* = w_1B_1 \cup w_2B_2 \quad \cdots\cdots (6\text{-}12)$$

이러한 과정을 거쳐 x_1^0 와 x_2^0 에 대한 최종적인 결론 y^0 를 B^* 의 멤버쉽 함수 $B^*(y)$ 의 'center of mass' 를 식(6-13) 과 같이 계산하여 사용하는 방법이 추론합성법칙(compositional rule of inference) 을 이용한 추론방법이다.

$$y^0 = \frac{\int B^*(y)y\ dy}{\int B^*(y)\ dy} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}13)$$

그림 6-2는 위와 같은 추론과정을 설명한다.

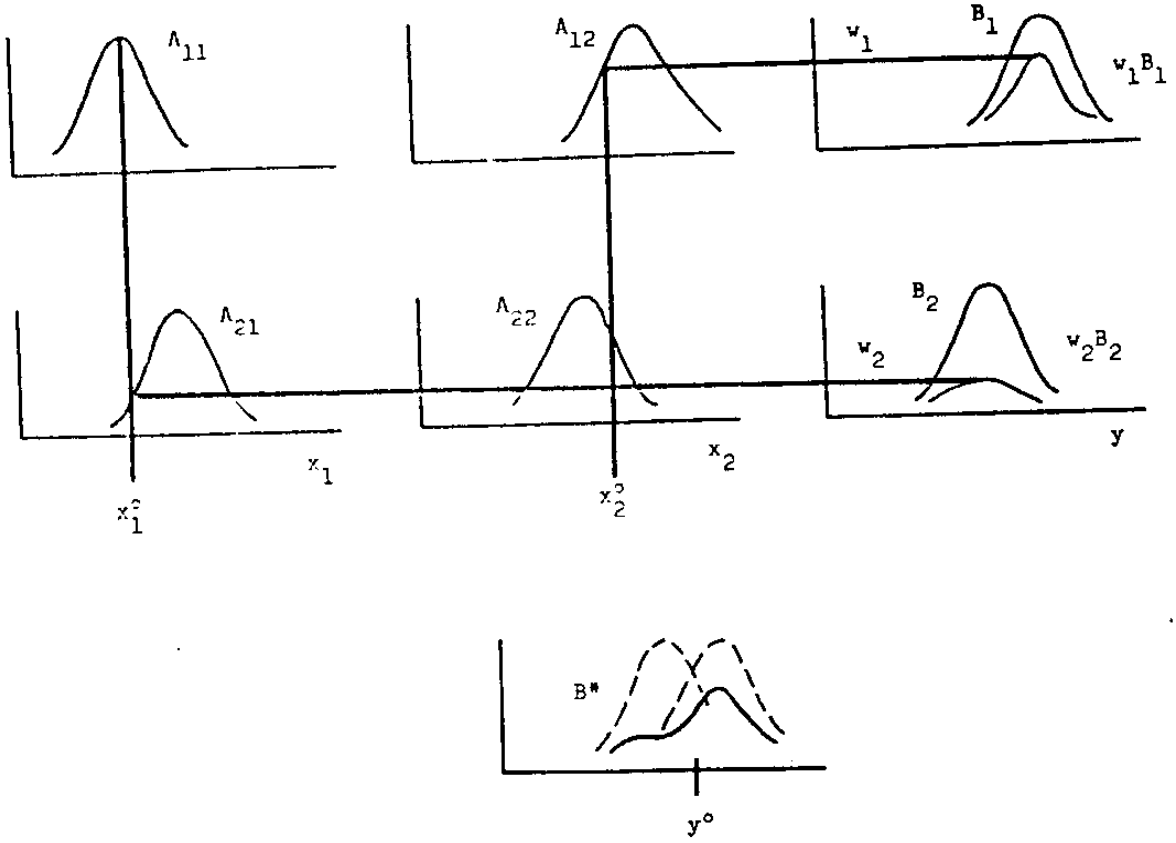


그림 6-2 Compositional rule of inference 를 이용한 추론법

나. 퍼지논리

일반적인 퍼지논리(fuzzy logic)를 이용한 추론방법은 두 가지의 'implication' R_1, R_2 에 대하여,

R_1: if x_1 is N, x_2 is P, then y is N

R_2:if x_1 is P, x_2 is N, then y is P

인 경우 주어진 x_1^0와 x_2^0 (단, $x_1^0 > x_2^0$)에 대하여 식 (6-14)과 식 (6-15)와 같이 w_1과 w_2를 정의한 경우,

$$w_1 = N(y_1) \quad \cdots\cdots (6\text{-}14)$$

$$w_2 = P(y_2) \quad \cdots\cdots (6\text{-}15)$$

결론 y^0를 식(6-16)과 같이 계산하는 방법이다.

$$y^0 = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2} \quad \cdots\cdots (6\text{-}16)$$

그림 6-3은 퍼지논리에 의한 추론방법을 설명한다.

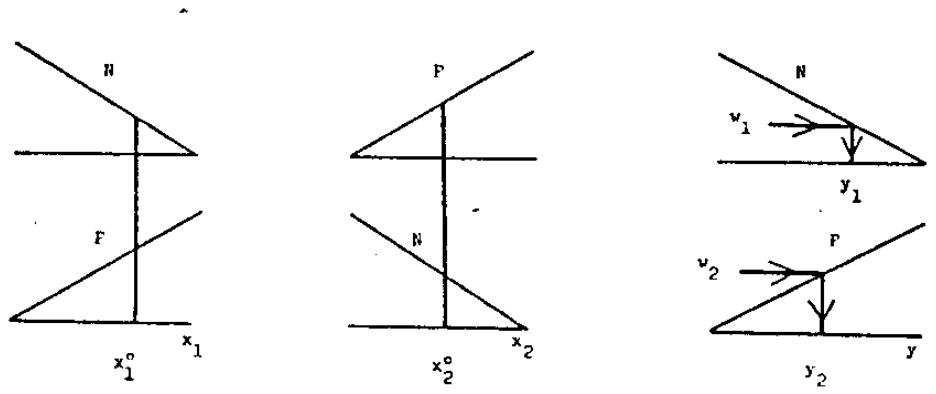


그림 6-3 Fuzzy logic을 이용한 추론법

퍼지논리에 의한 추론방법은 FLC의 설계에 가장 많이 이용된다. 따라서 이를 일반화 하여 표현하면 다음과 같다.

다음과 같은 두가지의 implication R_1, R_2가 존재하는 경우,

R_1 : if x_1 is A_{11}, x_2 is A_{12}, then $y = f_1(x_1, x_2)$

R_2 : if x_1 is A_{21}, x_2 is A_{22}, then $y = f_2(x_1, x_2)$

주어진 x_1^0 및 x_2^0 에 대하여 R_1 과 R_2 의 전제에 대한 진리값을 w_1 과 w_2 라 하면

$$w_1 = A_{11}(x_1^0) \wedge A_{12}(x_{12}^0)$$

$$w_2 = A_{22}(x_1^0) \wedge A_{22}(x_{12}^0)$$

x_1^0 와 x_2^0 에 대한 최종적인 결론 y^0 는 식(6-17)과 같이 계산된다.

$$y^0 = \frac{w_1 f_1(x_1^0, x_2^0) + w_2 f_2(x_1^0, x_2^0)}{w_1 + w_2} \quad \cdots\cdots\cdots\cdots \quad (6\text{-}17)$$

제3절 FLC(Fuzzy Logic Controller) 설계

1. 개 요

일반적으로 산업현장에서 널리 사용되고 있는 DDC 의 구조는 그림 6-4 와 같다.

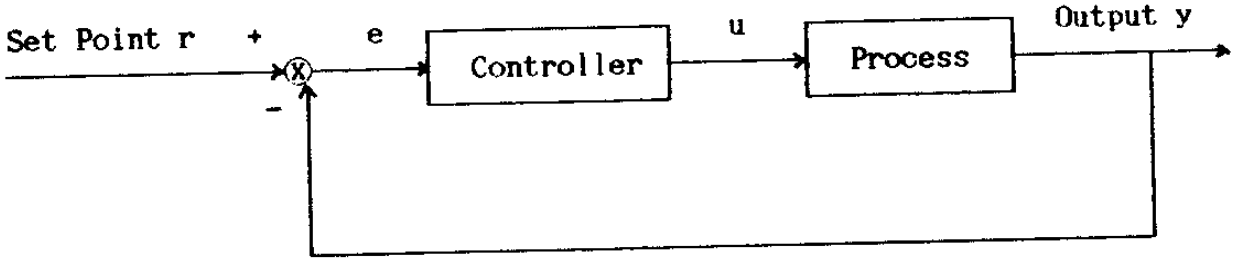


그림 6-4 DDC 의 구조

DDC 의 제어기능은 공정의 출력 $\mathbf{y} = (y_1, y_2 \cdots\cdots y_P)$와 설정치 $\mathbf{r} = (r_1, r_2 \cdots\cdots , r_P)$의 차이 $\mathbf{e} = (e_1, e_2, \cdots\cdots, e_P)$를 근거로 공정의 출력 **y**가 공정의 출력 **r**에 최대한 근접하도록 제어입력 $\mathbf{u} = (u_1, u_2, \cdots\cdots, u_r)$를 계산하는 것이다. 이러한 기존의 제어기의 구조와 유사하게 FLC 는 그림 6-5 와 같은 구조를 갖는다. 한편으로, 그림 6-5의 FLC의 기능은 그림 6-6 과 같이 요약될 수 있다.

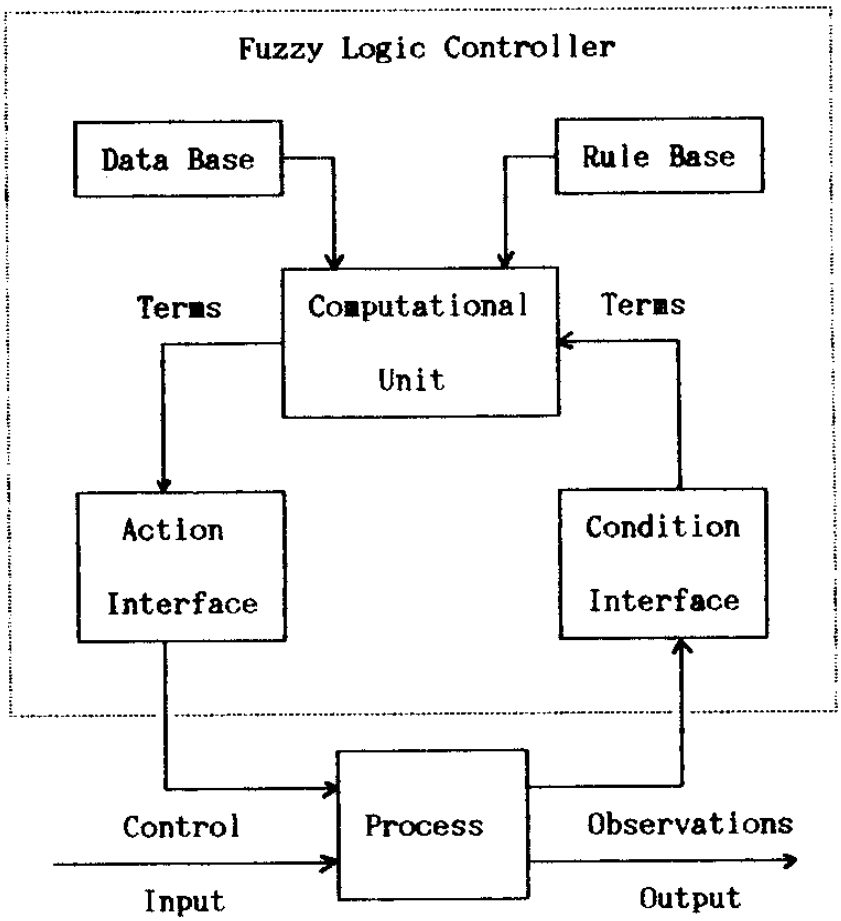


그림 6-5 FLC 의 구조

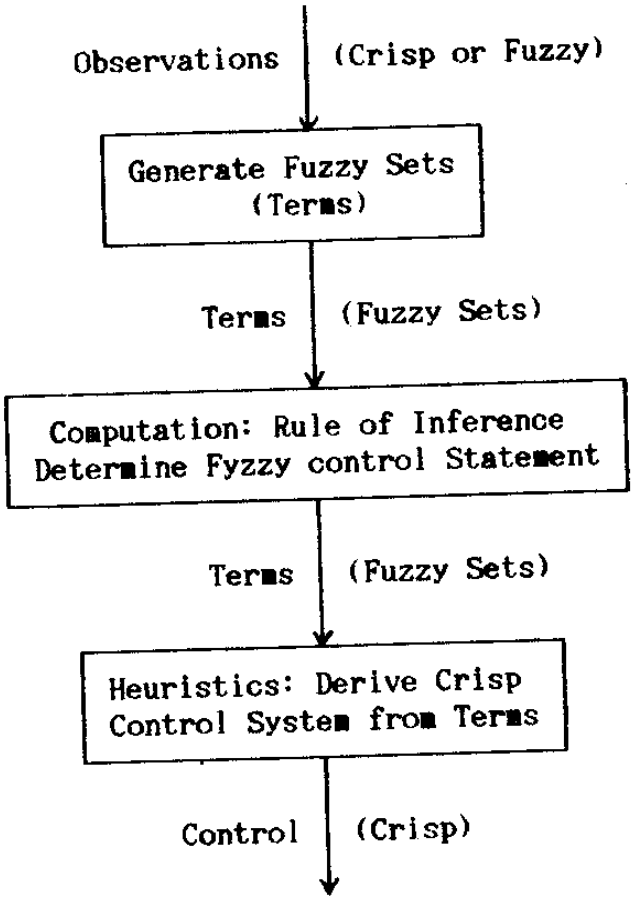


그림 6-6 FLC 의 기능

그림 6-5 와 같은 FLC 의 구조를 설계하는 방법은 다음과 같은 과정으로 요약될 수 있다.

Step 1. 공정의 관측되어야하는 측정변수와 조절가능한 조작변수를 선정한다.

Step 2. 측정변수를 퍼지집합으로 표현하는 조건처리부 (Condition Interface)를 정의한다.

Step 3. 특정조건에 적용될 법칙을 Rule Base 화 한다.

Step 4. 측정변수에 따라 RB 의 법칙을 적용하여 퍼지 제

어 입력을 계산하는 연산부를 정의한다.

Step 5. 퍼지 제어입력을 공정의 조작변수로 변환하는 조작처리부(Action Interface)를 정의한다.

FLC 의 기본적인 개념을 파악하기위해 위와 같이 정의된 설계 방법에 따라 Mamdani 등[2] 이 발표한 스팀엔진의 FLC설계를 과정별로 요약하면 다음과 같다.

① Process Interface

보일러의 스팀 압력과 엔진속도를 측정변수로 선정하였다. 그리고 보일러의 'heat input' 과 엔진 실린더 입구의 'throttle valve' 의 개폐정도를 조작변수로 선정하였다. 측정된 변수의 기준 상태와의 오차를 상태변수로 정의하고 조작변수를 HC(Heat Change) 와 TC(Throttle Change)로 정의하였다.

② Definition of Fyzzy State Descriptions

공정의 상태를 언어변수(linguistic variable)로 표현하기 위하여 공정의 오차(error) 와 오차의 변화(error change)를 정량화 하여 'universe of discourse'의 구성원소에 대응되도록 하고 각각의 구성원소에 대해 집합에 속하는 정도의 등급을 멤버쉽 함수로 정의한다. 'grades of membership'를 다음과 같은 여덟 가지의 퍼지집합으로 분류한다.

* PB is "positive big"
* PM is "positive medium"

* PS is “positive small”
* PO is “positive nil”
* NO is “negative nil”
* NS is “negative small”
* NM is “negative medium”
* NB is “negative big”

측정오차와 측정오차의 변화 각각과 ‘grade of membership’의 관계를 ‘look-up table’ 형태로 표시한다. PE(Pressure Error)와 SE(Speed Error)를 크기에 따라 구간별로 나누고 여덟개의 퍼지집합에 대하여 멤버쉽함수의 값을 표시한 ‘look-up table’은 표 6-3 과 같다.

표 6-3 Look-up Table of PE and SE

	-6	-5	-4	-3	-2	-1	-0	+0	+1	+2	+3	+4	+5	+6
PE	0	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2
PS	0	0	0	0	0	0	0	0.3	0.8	1.0	0.5	0.1	0	0
PO	0	0	0	0	0	0	0	1.0	0.6	0.1	0	0	0	0
NO	0	0	0	0	0.1	0.6	1.0	0	0	0	0	0	0	0
NS	0	0	0.1	0.5	1.0	0.8	0.3	0	0	0	0	0	0	0
NM	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0
NB	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0	0

표 6-4 Look-up Table of CPE and CSE

	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6
PE	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2
PS	0	0	0	0	0	0	0	0.9	1.0	0.7	0.2	0	0
NO	0	0	0	0	0	0.5	1.0	0.5	0	0	0	0	0
NS	0	0	0.2	0.7	1.0	0.9	0	0	0	0	0	0	0
NM	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0
NB	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0

표 6-3 과 표 6-4 에 정의된 멤버쉽 함수값 이외에도 PE,SE, CPE, CSE 의 값의 등급에 무관한 법칙의 표현을 위하여 'default' 값으로 1 을 정의하고 ANY 라는 퍼지집합에 속하도록 한다. 다음 표 6-5 와 표 6-6 은 HC (Heat Change) 와 TC(Throttle Cha-nge) 에 대한 'look-up table' 이다.

표 6-5 Look-Up Table of HC(Heat Change)

	-7	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6	+7
PE	0	0	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2	0
PS	0	0	0	0	0	0	0	0.4	1.0	0.8	0.4	0.1	0	0	0
NO	0	0	0	0	0	0	0.2	1.0	0.2	0	0	0	0	0	0
NS	0	0	0	0.1	0.4	0.8	1.0	0.4	0	0	0	0	0	0	0
NM	0	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0
NB	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0	0	0

표 6-6 Look-up Table of TC(Throttle Change)

	-2	-1	0	+1	+2
PB	0	0	0	0.5	1.0
PS	0	0	0.5	1.0	0.5
NO	0	0.5	1.0	0.5	0
NS	0.5	1.0	0.5	0	0
NB	1.0	0.5	0	0	0

표 6-3, 표 6-4, 표 6-5 및 표 6-6 의 멤버쉽 함수값은 FLC 를 설계하는 설계자의 주관적인 판단을 근거로 한다.

③ Design of Rule-Base

스팀엔진의 제어를 위한 RB 는 15 개의 'if-then' 형태로 표현된 다음과 같은 제어법칙을 가지고 있다. 또한 각각의 제어법칙은 'min-max compositional rule of inference'를 이용하여 적용범위를 확장시킬 수 있다.

```
<HEATER ALGORITHM>
   If          PE  =  NB
      and      CPE =  not(NB or NM)
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  PB
Else
   If          PE  =  NB or NM
      and      CPE =  NS
```

```
      and     SE  =  ANY
      and     CSE =  ANY
   then       HC  =  PM
Else
   If         PE  =  NS
      and     CPE =  PS or NO
      and     SE  =  ANY
      and     CSE =  ANY
   then       HC  =  PM
Else
   If         PE  =  NO
      and     CPE =  PB or PM
      and     SE  =  ANY
      and     CSE =  ANY
   then       HC  =  PM
Else
   If         PE  =  NO
      and     CPE =  NB or NM
      and     SE  =  ANY
      and     CSE =  ANY
   then       HC  =  NM
Else
   If         PE  =  PO or NO
      and     CPE =  NO
      and     SE  =  ANY
      and     CSE =  ANY
   then       HC  =  NO
Else
   If         PE  =  PO
      and     CPE =  NB or NM
      and     SE  =  ANY
      and     CSE =  ANY
   then       HC  =  PM
```

```
Else
   If          PE  =  PO
      and      CPE =  PB or PM
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  NM
Else
   If          PE  =  PS
      and      CPE =  PS or NO
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  NM
Else
   If          PE  =  PB or PM
      and      CPE =  NS
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  NM
Else
   If          PE  =  PB
      and      CPE =  not(NB or NM)
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  NB
Else
   If          PE  =  NO
      and      CPE =  PS
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  PS
Else
   If          PE  =  NO
      and      CPE =  NS
      and      SE  =  ANY
      and      CSE =  ANY
```

```
   then        HC  =  NS
Else
   If          PE  =  PO
      and      CPE =  NS
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  PS
Else
   If          PE  =  PO
      and      CPE =  PS
      and      SE  =  ANY
      and      CSE =  ANY
   then        HC  =  NS

<THROTTLE ALGORITHM>
   If          PE  =  ANY
      and      CPE =  ANY
      and      SE  =  NB
      and      CSE =  not(NB or NM)
   then        TC  =  PB
Else
   If          PE  =  ANY
      and      CPE =  ANY
      and      SE  =  NM
      and      CSE =  PB or PM or PS
   then        TC  =  PS
Else
   If          PE  =  ANY
      and      CPE =  ANY
      and      SE  =  NS
      and      CSE =  PB or PM
   then        TC  =  PS
Else
   If          PE  =  ANY
      and      CPE =  ANY
```

```
      and     SE  =  NO
      and    CSE  =  PB
   then       TC  =  PS
Else
   If         PE  =  ANY
      and    CPE  =  ANY
      and     SE  =  PO or NO
      and    CSE  =  PS or NS or NO
   then       TC  =  NO
Else
   If         PE  =  ANY
      and    CPE  =  ANY
      and     SE  =  PO
      and    CSE  =  PB
   then       TC  =  NS
Else
   If         PE  =  ANY
      and    CPE  =  ANY
      and     SE  =  PS
      and    CSE  =  PB or PM
   then       TC  =  NS
Else
   If         PE  =  ANY
      and    CPE  =  ANY
      and     SE  =  PM
      and    CSE  =  PB or PM or PS
   then       TC  =  NS
Else
   If         PE  =  ANY
      and    CPE  =  ANY
      and     SE  =  PB
      and    CSE  =  not(NB or NM)
   then       TC  =  NB
```

④ Translation of fuzzy control statements into crisp control actions

멤버쉽 함수가 'convex' 이고 'unimodal' 인 경우에는 'crisp maximizing decision' 이 제어입력을 결정하는데 이용된다. 한편, 멤버쉽 함수가 'unimodal' 이 아닌 경우에는 제어입력을 결정하는 경험법칙으로 'center of gravity' 방법을 이용한다.

2. 설계방법

다음은 FLC 설계방법론을 이를 이용한 FLC 설계사례를 중심으로 알아본다.

FLC 의 설계방법은 다음과 같이 네가지로 구분할 수 있다.

* 전문가의 제어와 관련된 경험과 지식을 이용하는 방법
* 공정 조업자의 제어행위를 모델화하는 방법
* 공정을 퍼지모델로 표현하고 제어기를 설계하는 방법
* 'Self-Organizing' 제어계를 구성하는 방법

가. 전문가의 경험과 지식의 이용

제어 대상이 되는 공정이 단순하고 일반적인 경우에는 제어계의 설계자가 가지고 있는 경험과 상식을 기본구조로 하는 FLC 를 구성할 수 있다. 공정이 시간지연이 있는 1 차계인 경우에 대한 피드백(feedback) 제어계를 구성하는 경우 설정치의 단위 변화에 따르는 출력은 일반적으로 그림 6-7 과 같다.

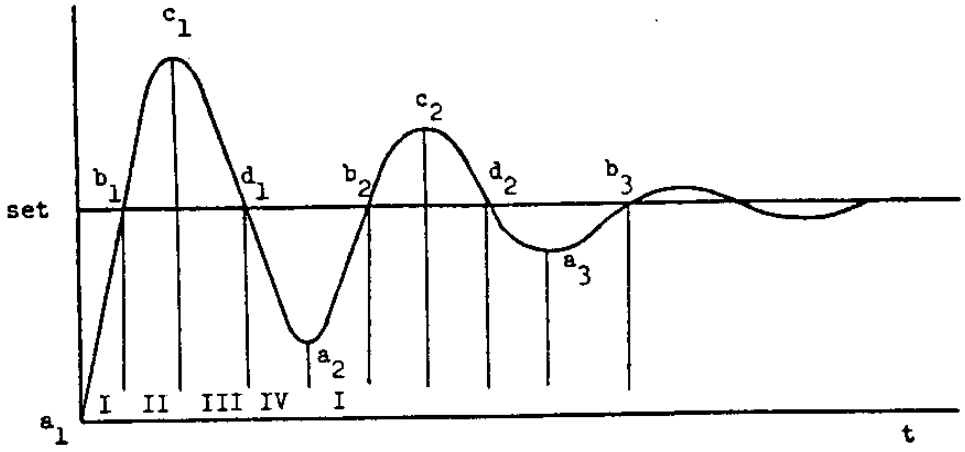


그림 6-7 설정치 단위변화에 따른 출력결과

이 경우 공정의 오차 E와 오차의 변화 CE를 이용하여 제어 입력 CO를 계산하는 경험법칙은 표 6-7과 같다.

표 6-7 일반적인 제어법칙

No.	E	CE	CO	Reference point
1	PB	ZO	PB	a_1
2	ZO	PB	NB	b_1
3	NB	ZO	NB	c_1
4	ZO	NB	PB	d_1
5	PM	ZO	PM	a_2
6	ZO	PM	NM	b_2
7	NM	ZO	NM	c_2
8	ZO	NM	PM	d_2
9	PS	ZO	PS	a_3
10	ZO	PS	NS	b_3
11	NS	ZO	NS	c_3
12	ZO	NS	PS	d_3
13	ZO	ZO	ZO	set point
14	PB	PS	PM	$a_1 - b_1$
15	PS	PB	NM	$a_1 - b_1$
16	NB	NS	NM	$c_1 - d_1$
17	NS	NS	NM	$c_1 - d_1$
18	PS	PS	ZO	$a_3 - b_3$
19	NS	NS	ZO	$c_3 - d_3$

이러한 일반적인 경험법칙중 1번 부터 13번까지의 제어법칙을 이용한 경우 그림 6-8의 A와 같은 제어결과를 얻을 수 있고 19개의 법칙 모두를 이용한 경우 제어결과는 그림 6-8의 B와 같다. 그러나 이러한 설계방법은 일반적인 방법론이 될 수는 없다.

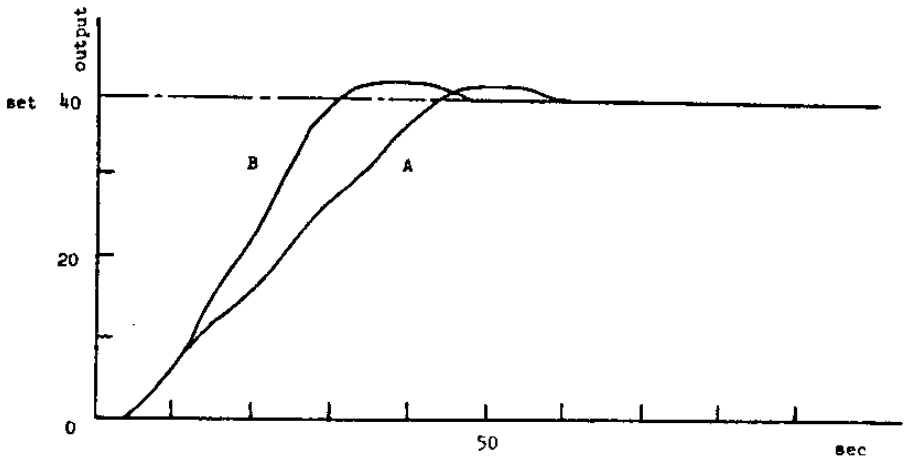


그림 6-8 FLC 의 제어결과

나. 공정 조업자의 행위 모사

공정의 조업자의 제어행위를 모델화하는 방법은 조업자의 행동의 근거가 되는 측정변수의 선정이 용이하고 조업자료를 근거로 쉽게 모델화 할 수 있는 장점을 지닌 설계방법이다. 그러나 실제적인 응용을 위해서는 앞서 설명한 방법과 병행하여 사용되는 것이 바람직하다.

다. 공정의 퍼지 모델화

공정자체에 대한 퍼지 모델을 추정방법에 의하여 인식하여 제

어계를 설계하는 방법은 FLC 의 설계방법중 가장 체계적으로 구성되어 있다. 공정에 대한 입출력 자료를 근거로 공정의 구조와 파라메터를 인식(identification) 하는 방법으로 'fuzzy implication' 의 전제와 결론에 대한 파라메터화된 기본모델을 설정하고 측정변수의 선택에 대한 구조적인 결정사항과 모델을 구성하고 있는 파라메터의 값을 주어진 목적함수에 대한 최적화문제로 간주하여 FLC 를 설계한다. 이 방법은 입출력자료가 충분한 경우 적용할 수 있는 가장 좋은 설계방법이다. 이 방법에 대해서는 다음절에서 자세히 설명한다.

라. Self-Organizing 제어계 구성

FLC의 설계방법중 일반적인 전문가시스템의 문제해결 방법과 유사하게 제어계 자체가 자신의 구조 및 파라메터를 구성해 가는 적응제어론적인 설계방법을 이용한 것이 'Self-Organizing' 제어계 설계 방법이다. FLC 에 적응제어의 기법이 병합된 구조로 아직까지 이론적인 연구외에는 별다른 성과가 없다.

제 4 절 FLC 모듈 개발

1. SPC (Set-Point Change)

1987 년 Sripada 등[6] 은 기존의 인공지능(AI)기법을 공정제어에 효율적으로 이용하는 방법으로 서어보(servo) 제어와 'regulatory' 제어의 경우에 대하여 AI 기법을 응용한 연구결

과를 발표하였다. 다음은 이들의 연구결과로 제안된 서어보 제어의 경우에 대한 알고리듬을 이용하여 개발된 설정치의 단위변화를 인지한 경우의 제어용 모듈에 대하여 설명한다.

현 시점 k에서의 설정치를 $y_{sp}(k)$, 공정출력을 $y(k)$라 하고 오차 $e(k)$를 식(6-18)과 같이 정의하자.

$$e(k) = y_{SP}(k) - y(k) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}18)$$

정의된 출력오차에 따라 공정의 설정치가 증가된 경우 제어입력을 임의의 조건을 만족하는 동안 가능한 최대치로 유지하다가 일정기간 동안 최소치의 입력을 가한 후 새로운 정상상태에 해당되는 제어입력을 가하면 출력이 설정치를 따르도록 할 수 있다.

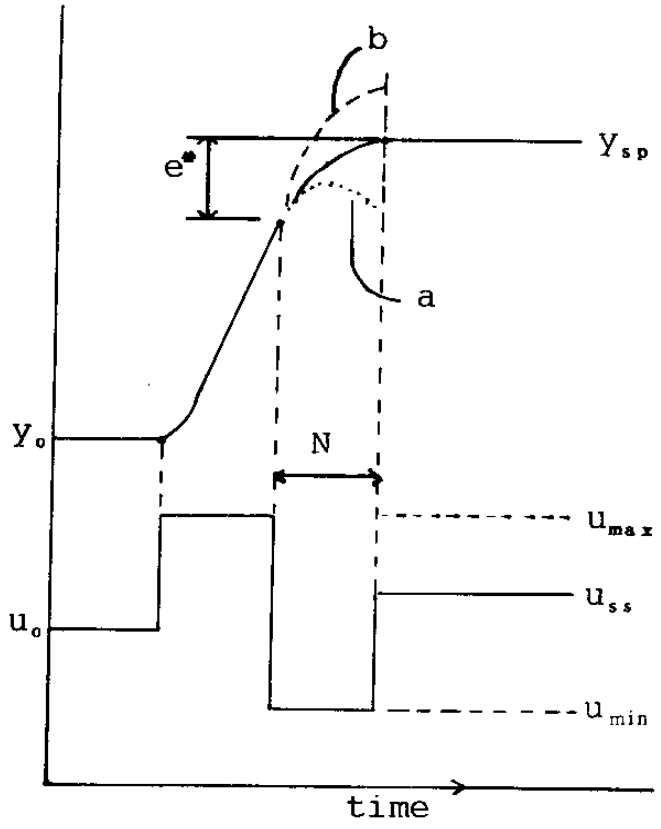


그림 6-9 SPC 작동 예

이와 같은 개념을 수학적인 모델링에 의존하지 않고 법칙화 하여 KB 에 저장하고 이를 공정의 제어에 이용할 수 있다. 그림 6-9 는 이러한 과정을 설명해 준다.

그림 6-9 에서 제어에 대한 스위칭 파라메터로 e* 와 N 이 정의되어 있는데 e* 와 N 이 정확한 경우 그림 6-9 의 실선과 같이 변화된 새로운 설정치에 공정의 출력이 정확하게 일치되지만 e* 가 큰 경우에는 그림 6-9 의 a처럼 공정의 출력이 설정치에 미치지 못하게 되고 이와 반대의 경우에는 그림 6-9 의 b처럼 공정의 출력이 설정치 보다 크게되는 결과를 나타낸다. 또한 N이 부정확한 경우에는 공정 출력의 변화율이 0 으로 되지 않아 공정에 정상상태 입력치를 입력한 이후에도 출력이 계속 변화하게 된다.

최적제어 이론에 따르자면 공정의 정확한 수학적인 모델이 존재하는 경우에 공정에 대한 제어입력의 최대치와 최소치를 가하는 기간을 정할 수 있고 그림 6-9 의 예와 같은 경우 제어입력의 스위칭 횟수를 공정의 차수 이하로 하여 최적 제어 결과를 얻을수 있다. 그러나 실제 공정의 경우 공정의 출력이 외란의 영향을 받고 공정의 정확한 수학적인 모델을 유도할 수 없는 경우가 많다. 따라서 실제적인 적용을 위해서는 스위칭 횟수를 온라인으로 조정해 줄 필요가 있다. 이러한 목적으로 다음과 같이 KB를 구성하고 이를 조업에 반영하는 제어계를 구성할 수 있다.

2 차 공정에 대하여 설정치 변화를 최적 시간내에 추적할 수 있도록 하기 위한 KB 는 제어입력의 스위칭과 스위칭 파라메터의 수정에 관한 두가지 부분으로 구성된다. 이와 같은 KB 에 의해서 FLC 의 기본 모듈인 SPC 가 구성된다.

가. Control Input Switching

< Positive Change >

* If { $(y_{sp}(k)-y_{sp}(k-1)) > 0$ },

then (i) $u(k) = u_{max}$ until $e(k) \leq e^*$,

(ii) $u(k) = u_{min}$ for N sampling intervals,

(iii) $u(k) = u_{ss}$ where, $u_{ss} = K_p^{-1} y_{sp}$

< Negative Change >

* If { $(y_{sp}(k)-y_{sp}(k-1)) < 0$ },

then (i) $u(k) = u_{min}$ until $e(k) \geq e^*$,

(ii) $u(k) = u_{max}$ for N sampling intervals,

(iii) $u(k) = u_{ss}$ where, $u_{ss} = K_p^{-1} y_{sp}$

나. Switching Parameter Adjustment

규정된 e* 와 N을 이용하여 제어한 결과에 따라 최적의 e* 와 N을 결정하기 위한 법칙으로 앞에서 설명된 법칙을 보완하도록 다음과 같이 e*를 수정하는 방법을 제시할 수 있다. 이때 △e도 수정가능한 파라메터가 된다. 또한 파라메터 N을 적절하게 조절하기 위해서는 출력결과를 분석하여 반복적으로 최적의 값을 찾아가도록 한다.

< Positive Change >

* If y overshoots y_{sp},

then e* is too large,

then $e^*(k) = e^*(k-1) - \Delta e$

* If y undershoots y_{*p},

then e* is too small,

then $e^*(k) = e^*(k-1) + \Delta e$

< Negative Change >

* If y overshoots y_{*p},

then e* is too small,

then $e^*(k) = e^*(k-1) + \Delta e$

* If y undershoots y_{*p},

then e* is too large,

then $e^*(k) = e^*(k-1) - \Delta e$

2. FMA(Fuzzy Modelling Algorithm)

다음은 앞서 설명된 퍼지집합이론과 추론방법 그리고 FLC의 설계방법을 종합하여 공정의 fuzzy 모델을 인식하고 이에 대한 제어법칙을 설정하여 공정의 제어에 이용하는 방법에 대하여 설명한다.

가. 퍼지 추론 알고리듬(Fuzzy Inference Algorithm)

경험법칙의 일반화된 표현방법으로 'fuzzy implication' R을 다음 식(6-19)와 같이 표시한다.

R: If $f(x_1$ is $A_1, \cdots, x_k$ is $A_k)$ then $y = g(x_1, \cdots, x_k)$ ..... (6-19)

이때 y는 추론되어야할 결론변수이고 x_1 부터 x_k 까지의 변수는 일반적인 제어계의 측정변수와 같이 전제변수이고 결론부의 독립변수로도 이용된다. A_1 부터 A_k 까지는 선형 멤버쉽 함수로 이루어진 퍼지집합들로 'implication' R의 추론영역을 한정한다. 또한 f는 'implication'의 전제부를 구성하는 논리함수이고, g는 전제변수를 독립변수로 하는 함수로 결론변수 y를 계산한다. 계산을 일반화시키기에 편리하도록 f는 논리곱만을 사용하고 g는 전제변수에 대하여 선형적인 관계만 존재하는 것으로 가정하여 다음 식(6-20)의 형태로 'implication'을 표시한다.

$$R : \text{If } x_1 \text{ is } A_1 \text{ and } \cdots \text{ and } x_k \text{ is } A_k$$
$$\text{then } y = p_0 + p_1 x_1 + \cdots + p_k x_k \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}20)$$

식(6-20)와 같은 형태로 표시된 n개의 'implication' R^i $(i = 1, \cdots, n)$에 대하여 전제변수의 값이 $x_1^0, \cdots, x_k^0$로 주어진 경우 결론 y를 추론하는 방법은 다음과 같은 단계를 거친다.

〈추론 알고리듬〉

Step 1.: 각각의 R^i에 대하여 주어진 g^i를 이용하여 y^i를 식(6-21)과 같이 계산한다.

$$y^i = g^i(x_1^0, \cdots, x_k^0)$$
$$= p_0^i + p_1^i x_1^0 + \cdots + p_k^i x_k^0 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}21)$$

Step 2.: 각각의 R^i에 의해 계산된 y^i에 대한 진리값을 식(6-22)와 같이 계산한다. 이때 $|*|$는 $*$에 대한 진리값을 의미한다.

$$|y = y^i| = |(x_1^0 \text{ is } A_1^i \text{ and } \cdots \text{ and } x_k^0 \text{ is } A_k^i| \wedge |R^i|$$

$$= (A_1^i(x_1^0) \quad \cdots \quad A_k^i(x_k^0)) \wedge |R^i| \quad \cdots\cdots (6\text{-}22)$$

한편, $|R^i| = 1$로 가정하면 식(6-22)는 식(6-23)과 같아진다.

$$|y = y^i| = A_1^i(x_1^0) \wedge \cdots \wedge A_k^i(x_k^0) \quad \cdots\cdots (6\text{-}23)$$

Step 3.: n개의 'implication'에 $x_1^0, \cdots, x_k^0$의 값이 입력된 경우 추론되는 출력은 다음의 식(6-24)와 같다.

$$y = \frac{\Sigma\, |y = y^i| * y^i}{\Sigma\, |y = y^i|} \quad \cdots\cdots (6\text{-}24)$$

나. 퍼지 모델링 알고리듬(Fuzzy Modelling Algorithm)

공정의 입출력 자료로 부터 식(6-20)과 같이 표현된 여러개의 'fuzzy implication'으로 구성된 퍼지모델을 인식하는 경우 추정해야할 사항은 다음과 같다.

우선 전제부의 전제변수를 어떤 변수로 구성할 것인가를 결

정해야 한다. 즉, 공정의 입출력 자료로 부터 직접 혹은 간접적으로 측정할 수 있는 변수중 어떤것을 선택하여 이용할 것인가를 결정해야 한다. 일반적인 FLC 는 앞서 살펴본 바와 같이 설정치와 출력간의 오차와 오차의 변화를 전제변수로 사용한다. 그러나 제어계에 대한 모델링이 아니고 공정 자체에 대한 퍼지모델을 구성하고자 하는 경우에는 공정의 조업자료로 부터 공정을 대표할 수 있는 전제변수를 결정하는 것은 모델에 대한 구조적인 'identification'의 문제가 된다. 다음으로 전제부를 구성하는 퍼지집합의 멤버쉽 함수의 파라메터를 결정하는 문제가 해결되어야 한다. 즉, 식(6-20)의 $A_1, \cdots, A_k$ 를 결정하여야 한다. 또한 결론부에 해당되는 파라메터를 결정해야 한다. 즉, 식(6-20)의 $p_0, \cdots, p_k$ 를 결정해야 한다.

이상과 같이 모델을 인식하기 위해서는 그림 10 과 같은 구조를 갖는 Takagi 등 [7] 이 제안한 알고리듬을 이용할 수 있다.

다음은 그림 6-10 에 제시된 모델 인식방법에 대하여 설명한다.

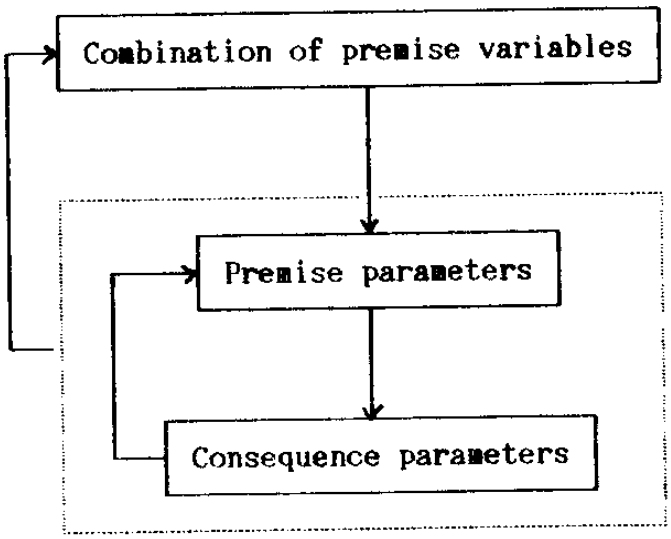


그림 6-10 퍼지 모델링 알고리듬

1) 결론부 파라메터 인식(Consequence Parameters Identification)

다음과 같이 n개의 implication이 존재하는 경우,

$$R^1 : \text{If } x_1 \text{ is } A_1^1 \text{ and } \cdots \text{ and } x_k \text{ is } A_k^1$$
$$\text{then } y = p_0^1 + p_1^1 x_1 + \cdots + p_k^1 x_k$$
$$\vdots$$
$$R^n : \text{If } x_1 \text{ is } A_1^n \text{ and } \cdots \text{ and } x_k \text{ is } A_k^n$$
$$\text{then } y = p_0^n + p_1^n x_1 + \cdots + p_k^n x_k$$

주어진 입력 $(x_1, \cdots, x_k)$에 대하여 출력 y는 식(6-25)와 같이 계산한다.

$$y = \left[\sum_{i=1}^{n} \{(A_1^i(x_1) \wedge \cdots \wedge A_k^i(x_k)) (p_0^i + p_1^i x_1 + \cdots + p_k^i x_k)\} \right] / \left[\sum_{i=1}^{n} (A_1^i(x_1) \wedge \cdots \wedge A_k^i(x_k)) \right] \cdots (6\text{-}25)$$

여기서 β_i를 식 (6-26)과 같이 정의하면,

$$\beta_i = \frac{A_1^i(x_1) \wedge \cdots \wedge A_k^i(x_k)}{\sum_{i=1}^{n} (A_1^i(x_1) \wedge \cdots \wedge A_k^i(x_k))} \cdots\cdots\cdots\cdots\cdots\cdots (6\text{-}26)$$

와 같고 식(6-25)를 식(6-27)과 같이 쓸 수 있다.

$$y = \sum_{i=1}^{n} \beta_i \ (p_0{}^i + p_1{}^i x_1 + \cdots + p_k{}^i x_k)$$

$$= \sum_{i=1}^{n} (p_0{}^i \beta_i + p_1{}^i x_1 \beta_i + \cdots + p_k{}^i \ x_k \beta_i) \quad \cdots\cdots\cdots \quad (6\text{-}27)$$

공정에 대한 입출력 자료가 $(x_{1j}, x_{2j}, \cdots, x_{kj} ; y_j)$로 $j = 1 \cdots m$의 m개가 주어진 경우 결론부 파라메터 $p_0{}^i$, $p_0{}^i$, $\cdots$, $p_k{}^i$ $(i = 1 \cdots n)$를 최소자승법(least square method)으로 구할 수 있다. 즉, 행렬 X를 식(6-28)과 같이 정의할 때,

$$X = \begin{bmatrix} \beta_{11}, \cdots, \beta_{n1}, x_{11}\beta_{11}, \cdots, x_{11}\beta_{n1}, \\ \qquad\qquad \cdots, x_{k1}\beta_{11}, \cdots, x_{k1}\beta_{n1} \\ \cdot \qquad \cdot \qquad \cdot \\ \cdot \qquad \cdot \qquad \cdot \\ \beta_{1m}, \cdots, \beta_{nm}, x_{1m}\beta_{1m}, \cdots, x_{1m}\beta_{nm}, \\ \qquad\qquad \cdots, x_{km}\beta_{1m}, \cdots, x_{km}\beta_{nm} \end{bmatrix} \quad \cdots\cdots\cdots \quad (6\text{-}28)$$

β_{ij}는 식(6-29)와 같다.

$$\beta_{ij} = \frac{A_{i1}(x_{1j}) \wedge \cdots \wedge A_{ik}(x_{kj})}{\sum_{j} (A_{i1}(x_{1j}) \wedge \cdots \wedge A_{ik}(x_{kj}))} \quad \cdots\cdots\cdots\cdots\cdots \quad (6\text{-}29)$$

한편 Y와 P를 식(6-30)과 식(6-31)로 정의한 경우 파라메터 벡터 P는 식(6-32)로 계산한다.

$$Y = [y_1, \cdots, y_m]^T \quad \cdots\cdots (6\text{-}30)$$

$$P = [p_0{}^1, \cdots, p_0{}^n, p_1{}^1, \cdots, p_1{}^n, \cdots, p_k{}^1, \cdots, p_k{}^n]^T \quad \cdots\cdots (6\text{-}31)$$

$$P = (X^TX)^{-1}X^TY \quad \cdots\cdots (6\text{-}32)$$

이때 X의 i번째 행을 x_i로 하고 Y의 i번째 원소를 y_i라 하면 식(6-32)의 P를 식(6-33)과 같이 순환법(recursive method)를 이용하여 계산할 수 있다.

$$P_{i+1} = P_i + S_{i+1}\, x^T{}_{i+1}\,(Y_{i+1} - x_{i+1}\, P_i) \quad \cdots\cdots (6\text{-}33)$$

$$S_{i+1} = S_i - \frac{S_i x^T{}_{i+1} x_{i+1} S_i}{1 + x_{i+1} S_i x^T{}_{i+1}} \quad \cdots\cdots (6\text{-}34)$$

(단, $i = 0, 1, \cdots, m-1$)

이때 초기값 P_0와 S_0는 식(6-35), 식(6-36)과 같다.

$$P_0 = 0 \quad \cdots\cdots (6\text{-}35)$$

$$S_0 = \alpha I \ (\alpha = \text{big number}) \quad \cdots\cdots (6\text{-}36)$$

2) 전제부 파라메터 인식(premise Parameters Identification)

각각의 'implication'을 구성하는 전제변수에 대한 퍼지집합을 정의하고 멤버쉽 함수를 부여하기 위하여 선형 멤버쉽 함수를 가정하고 이에 해당되는 기울기와 절편을 가정하여 결론부의 파라메터를 추정해간다. 이때 성능지표(performance index) ϕ를 추정 파라메터에 의한 추정 출력치 y^*와 실제 입출력 자료의 출력치 y간의 편차의 제곱의 합으로 식(6-37)과 같이 정의하고 정의된 성능지표가 최소화 되도록 하는 'nonlinear programming'의 문제를 해결해야 한다.

$$\phi = \sum_{j=1}^{m} (y_j^* - y_j)^2 \quad \cdots\cdots (6\text{-}37)$$

본 연구에서는 'complex method for the minimization'을 이용하여 최적의 전제부의 파라메터를 추정한다. 즉, 임의의 전제 변수 x_i에 대하여 선형 멤버쉽함수 A_i의 값이 0이 되는 x_i의 값을 X_{Zero}라 하고 x_{Zero}의 상한과 하한을 각각 U_{Zero}와 L_{Zero}라 하여 식(6-38)과 같이 구간〔L_{Zero}, U_{Zero}〕내에서 랜덤탐색법(radom search method)을 이용하여 X_{Zero}를 계산한다.

$$x_{Zero} = L_{Zero} + r\,(U_{Zero} - L_{Zero}) \quad \cdots\cdots (6\text{-}38)$$

(단, r = random number, $0 \leq r \leq 1$)

선형 멤버쉽함수 A_i 의 값이 1이 되는 x_i 의 값을 x_{one}라 하고 x_{one}의 상한과 하한을 각각 U_{one}와 L_{one}라 하여 식(6-39)과 같이 구간〔L_{one}, U_{one}〕내 에서 x_{one}를 계산한다.

$$x_{one} = L_{one} + r(U_{one} - L_{one}) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}39)$$

(단, r = random number, $0 \leq r \leq 1$)

3) 전제변수의 선택(Choice of Premise Variables)

전제변수의 선택은 제어계를 구성하는 경우에는 오차와 오차의 변화를 일반적으로 사용하게 된다. 그러나 공정에 대한 퍼지모델을 추정하고자 하는 경우 다음과 같이 'heuristic search'를 이용한다.

공정에 k개의 $x_1, \cdots, x_k$ 가 입력되고 y가 출력되는 경우,

Step 1. : x_1 을 "big" 과 "small" 의 두개의 퍼지 집합으로 나누고 $x_2, \cdots, x_k$ 는 'subspace'를 분할하지 않는 경우를 모델 1-1 이라 하고 같은 방법으로 k개의 모델 1-i ($i = 1 \cdots k$)를 구성한다.

Step 2.: Step 1.에서 구성된 k개의 모델에 대해 주어진 입출력자료를 이용하여 결론부와 전제부의 파라메터를 추정한다. 이때 각각의 모델에 대한 성능지표를 계산하고 최소의 성능지표를 갖는 모델을 안정 상태로 정의한다. 이 단계에서 안정상태로 정

의된 것을 1-i 라 하자.

Step 3.: Step 2. 에서 선정된 x_i 에 대한 두개의 퍼지집합을 고정시키고 x_i- x_j (j = 1 … k) 의 k개의 쌍을 전제변수로 선택하여 x_j 를 두개의 퍼지집합으로 나눈다. 이때 j = i 이면 x_i 를 네개의 퍼지집합으로 나눈다. 즉, 'big', 'medium big', 'medium small', 및 'small' 로 나눈다. 이 경우 Step 2. 에서 처럼 안정상태를 찾아 2 - j 라 하자.

Step 4.: Step 3. 에서 선정된 2-j 를 이용하여 Step 3. 의 과정을 반복한다.

위와 같은 Step 1., 2., 3., 4.의 과정은 그림 6-11 과 같이 요약될 수 있다.

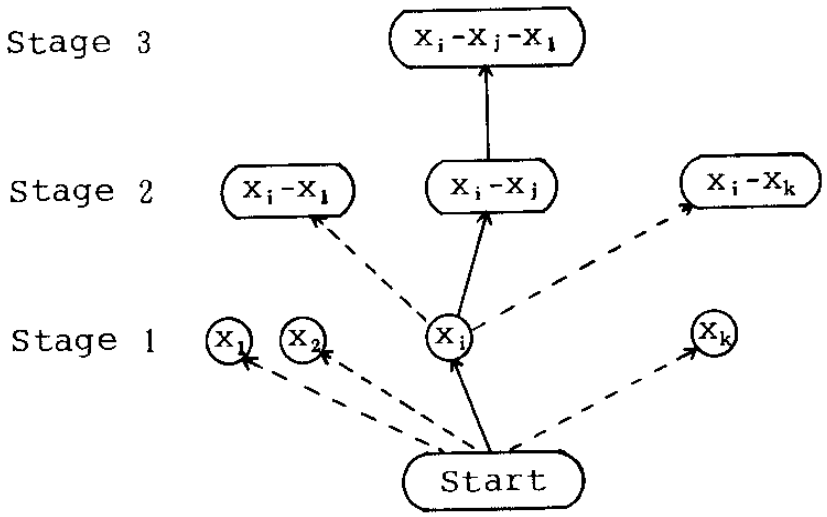


그림 6-11 전제변수의 선택

제 5 절 FLC 모사 결과

1. SPC

제 4 절에서 설명한 SPC 모듈의 성능을 실험하기 위하여 다음 식 (6-40) 과 같은 공정에 대하여 SPC 를 적용하였다.

$$y(k) = 1.7567\, y(k-1) - 0.7788\, y(k-2) + 0.1150\, u(k-1) + 0.1060\, u(k-2) \quad \cdots\cdots (6\text{-}40)$$

공정의 정상상태 이득이 1 이고 공정 입력의 상한값과 하한값은 각각 2 와 -2 로 하였다. e* 에 대한 초기 가정을 0.6 으로 하여 시간 k = 10 일때 공정의 설정치가 0 에서 1 로 단위변화하도록 했다. 이 경우 스위칭 파라메터 e* 는 주어진 초기값과 △e에 따라 감소 또는 증가하여 e* 가 최적치에 수렴하도록 하였다.

그림 6-12, 그림 6-13, 그림 6-14 는 스위칭 파라메터 N이 각각 1, 2, 3 인 경우의 수렴과정을 설명한다. 그림 6-15 는 각각의 경우의 최적 e* 에 따라 SPC 의 작동 결과를 나타낸다. 스위칭 파라메터 N에 대한 자동적인 수렴 방법에 대해서는 좀더 많은 연구를 필요로 한다. 그림 6-16 은 e* 가 초기값으로 부터 최적치에 수렴되는 과정을 설명한다. 그림 6-15 의 N이 2 와 3 인 경우의 작동 추이가 유사한 것은 그림 6-16 의 N이 2 와 3 인 경우의 결과가 일치하는 현상으로 설명될 수 있다.

설정치의 단위 변화가 자주 요구되는 공정을 실제적인 상한

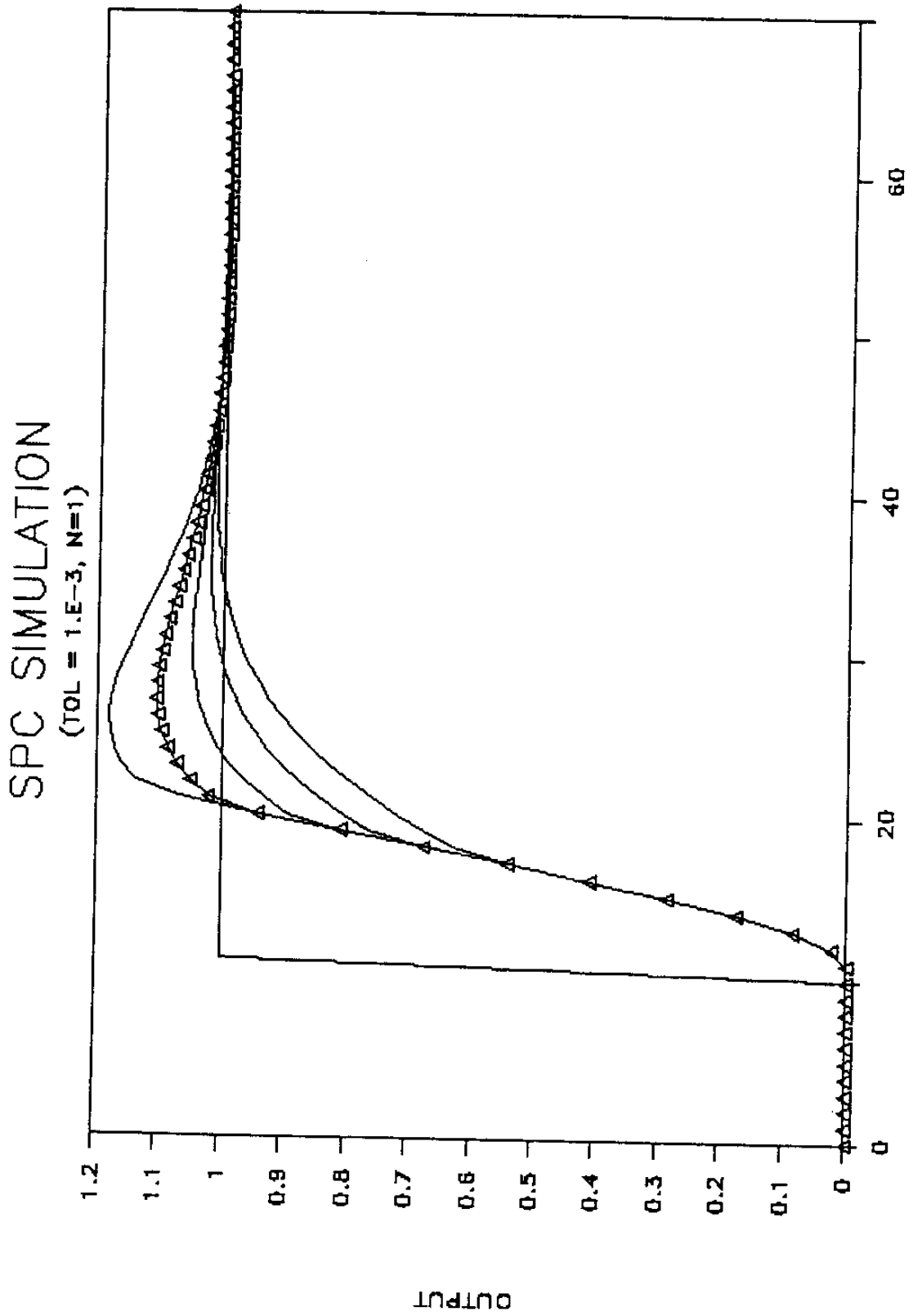


그림 6-12 SPC 모사 (N = 1)

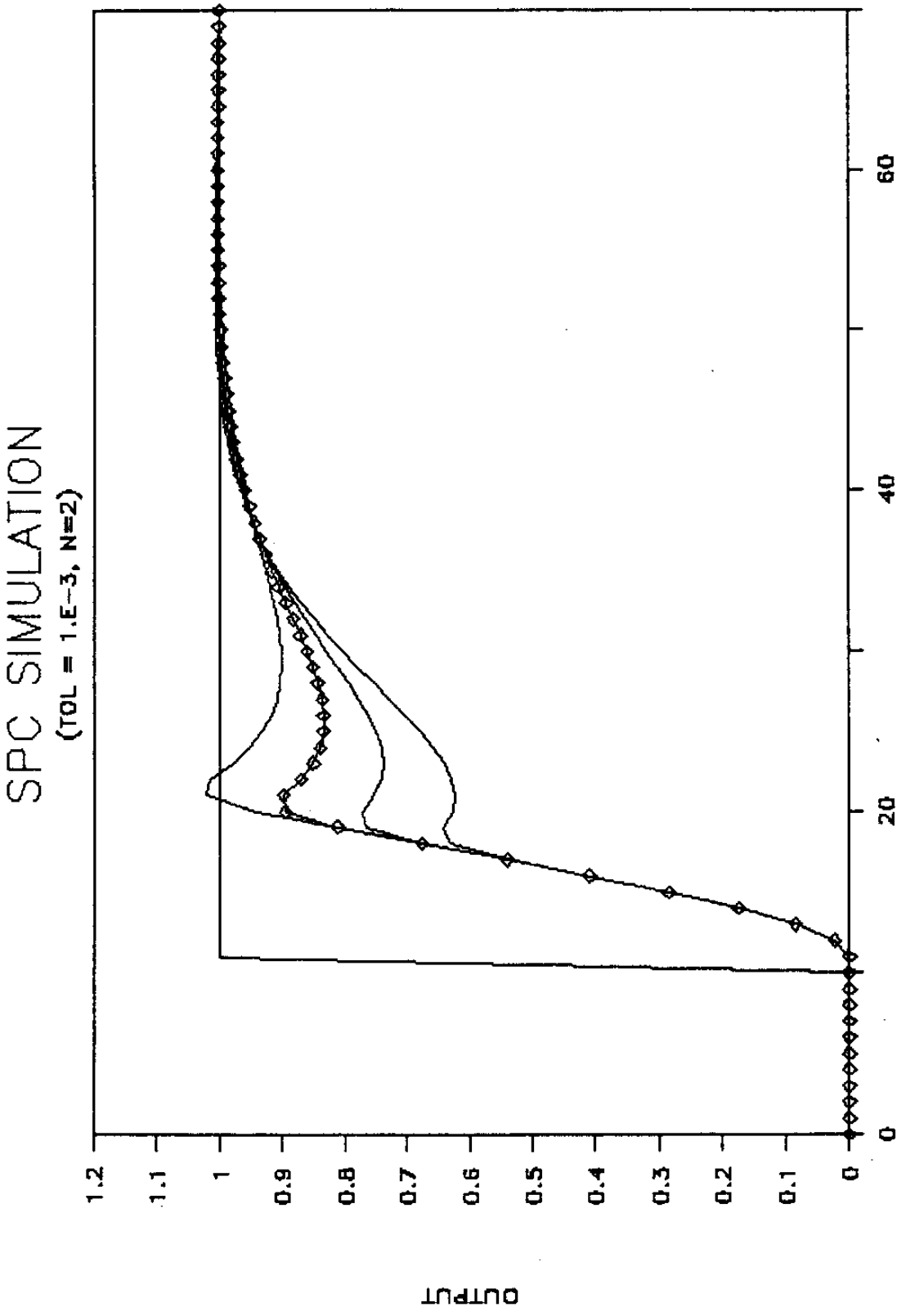


그림 6-13 SPC 모사 (N = 2)

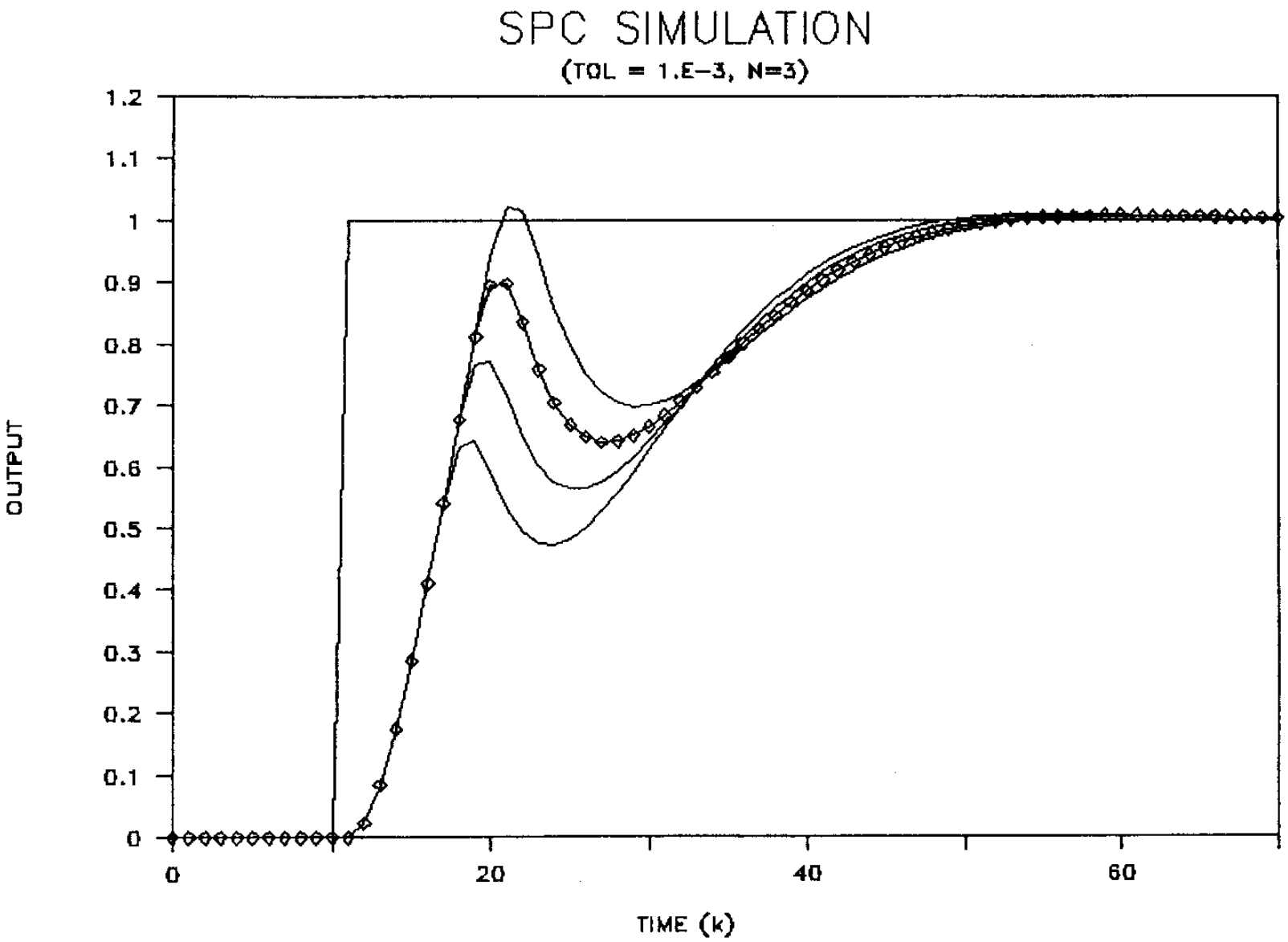


그림 6-14 SPC 모사 (N = 3)

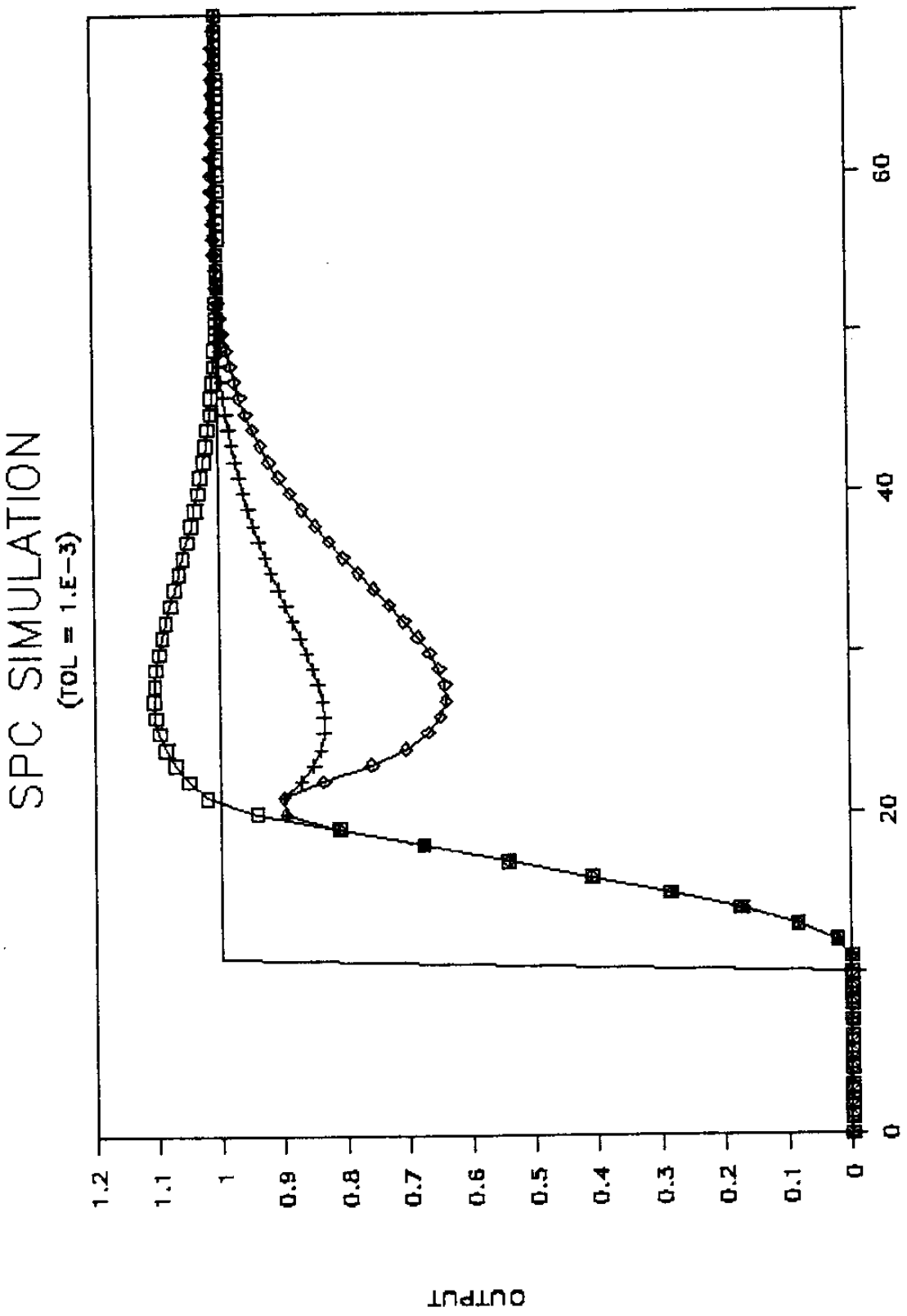


그림 6-15 SPC 모사 결과 종합 (N = 1,2,3)

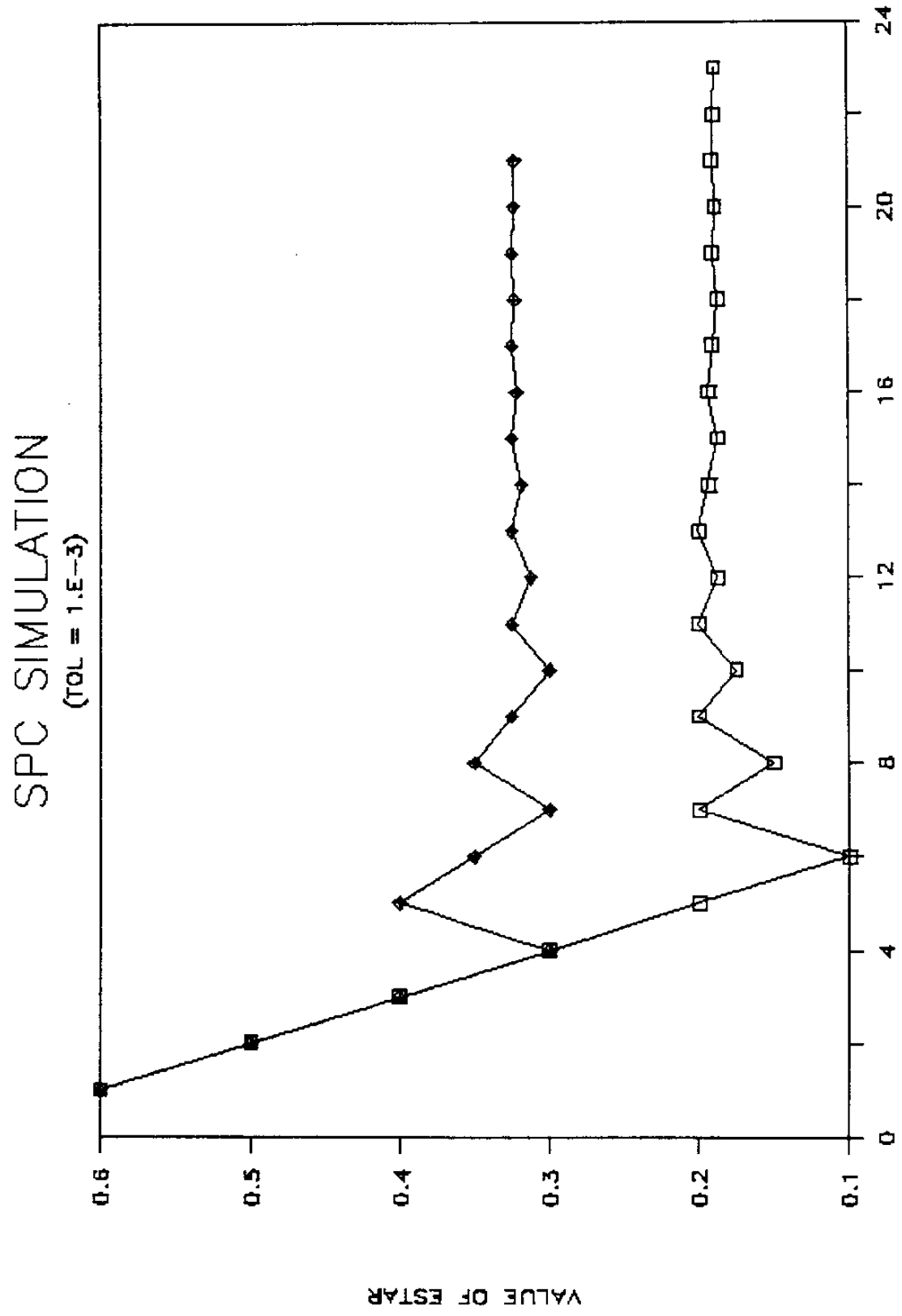

SPC SIMULATION
(TOL = 1.E-3)
VALUE OF ESTAR
0.6
0.5
0.4
0.3
0.2
0.1
0
4
8
12
16
20
24
ITERATION NUMBER
□ N = 1
+ N = 2
◇ N = 3

그림 6-16 e* 수렴과정

값과 하한값이 고정적인 조작변수로 제어하고자 하는 경우 간편하게 적용할 수 있는 제어방법이 될 수 있으나 공정에 대한 사전 정보를 필요로 하는 단점도 있으므로 실제적인 적용 이전에 모사를 통한 타당성이 입증되도록 하여야 한다.

2. FMA

제 4 절에서 설명한 FMA 모듈의 성능을 시험평가하기 위하여 그림 6-17 과 같은 제어 루프의 PID 제어기의 작동을 퍼지 모델로 인식하고 인식된 모델을 이용하여 공정을 제어하는 과정을 모사했다.

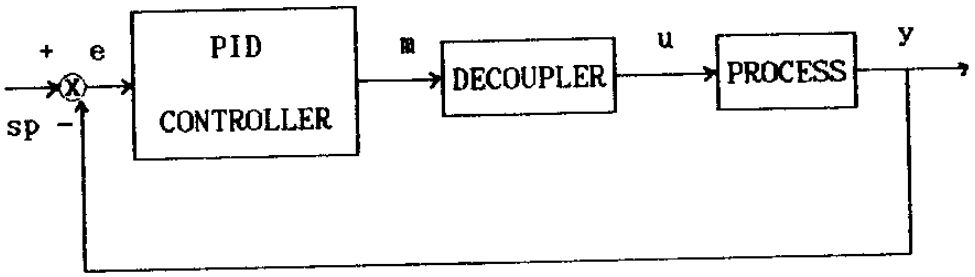


그림 6-17 FMA 대상 모델

대상 공정은 입력 u_1, u_2에 따라 y_1, y_2를 출력하는 MIMO 공정으로서 $u_1 - y_2$ 과 $u_2 - y_1$ 의 영향을 제거하기 위하여 'decoupler' 를 설치하였다. 즉 그림 6-17 의 대상 공정은 2 개의 SISO 제어 루프가 병렬로 연결된 구조가 된다. PID 제어기의 임의의 설정치 변화에 대한 임의의 입출력 자료(e,m) 를 근거로 하여 현시점 k를 기준으로 현시점의 출력 오차 e(k) 와 k-1 의 시점에서의 출력오차 e(k-1) 을 공정 모델의 전제변수로 선택하였

다. 제 4 절에서 설명된 전제변수 선택에 대한 알고리듬을 사용한 모사 결과에서도 같은 결과를 얻을 수 있다. 그러나 모사 대상 공정이 PID 제어기인 만큼 전제 변수 선택은 PID 제어기의 입력 변수로 선택한다는 것은 일종의 전문가의 경험으로 간주될 수 있고 일반적인 FLC 응용에 있어서도 전제 변수는 이와 크게 다르지 않는 범위내에서 결정된다. 또한 결론부는 식(6-41)과 같이 한 스탭 이전의 출력치와 공정의 오차로 구성되게 하여 PID 제어기의 구조와 동일하게 하여 비교의 기준을 동일하게 하였다.

$$m(k) = p_0 + p_1\ m(k-1) + p_2\ e(k) + p_3\ e(k-1) \quad (6\text{-}41)$$

PID 제어기에 대한 434 개의 입출력 자료로 부터 FMA 모듈은 각각의 제어 루프에 대하여 11 개씩의 제어 법칙을 인식하였다. 제어법칙의 일반적인 형태는 그림 6-18 과 같은 전제 변수의 표현을 다음과 식(6-42)와 같은 전제부로 인식하는 경우 각각의 11 개의 제어 법칙의 전제부는 표 6-8 과 표 6-9 와 같다. 그리고 표 6-10 과 표 6-11 은 결론부의 파라메터이다.

$$\begin{aligned} &\text{IF } \{\ ONE_1 \le e(k) \le ZERO_1\ \} \text{ AND} \\ &\quad (ZERO_1) \qquad (ONE_1) \\ &\qquad\qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (6\text{-}42) \\ &\{\ ONE_2 \le e(k-1) \le ZERO_2\ \} \\ &(ZERO_2) \qquad (ONE_2) \end{aligned}$$

$$\text{THEN} \quad m(k) = p_0 + p_1\ m(k-1) + p_2\ e(k) + p_3\ e(k-1)$$

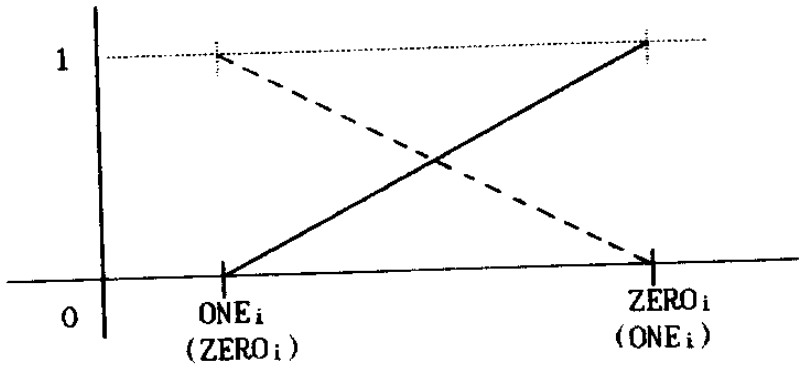


그림 6-18 전제변수 v_i 의 표시 방법

표 6-8, 표 6-9, 표 6-10, 그리고 표 6-11 의 전제부와 결론부의 파라메터를 이용하여 그림 6-17 의 공정에 대한 제어결과를 PID제어기와 비교한 결과 제어기의 출력 m_1 과 m_2 는 각각 그림 6-19 그림 6-20 과 같고 공정의 출력 y_1 과 y_2 는 그림 6-21 과 그림 6-22 와 같다. 그림에서 볼 수 있듯이 PID 제어기와 FLC 의 작동은 매우 유사하다. FLC 의 작동 가능성을 설명하기 위하여 PID 제어기의 입출력 자료를 이용하였으나 예와 같이 공정의 수학적인 모델링이 가능한 경우에 대하여서는 엄밀한 수학적인 분석을 바탕으로 하는 제어계의 구성이 신뢰도가 높다. 하지만 수학적인 분석이 어렵고 전문적인 공정 조업자의 경험과 지식에 의해 조업되는 공정에 대하여 조업자의 조업행위에 대한 입출력 자료를 근거로 앞서 설명된 바와 같은 퍼지 모델링에 의한 제어가 가능함이 입증되었다.

표 6-8. Loop-1의 전제부 파라메터

$ZERO_1$	ONE_1	$ZERO_2$	ONE_2
-.1819	0	-.18	0
-.1819	0	.8443	0
.6703	0	-.18	0
.6703	0	.8443	0
.6703	0	.2986	1
.2117	.4586	.8443	0
.2117	.4586	.2986	1
.8196	.4586	.8443	0
.8196	.4586	.2986	1
.3638	1	.8443	0
.3638	1	.2986	1

표 6-9. Loop-2의 전제부 파라메터

$ZERO_1$	ONE_1	$ZERO_2$	ONE_2
-.0856	0	-.1633	0
-.0856	0	.5855	0
.7072	0	-.1633	0
.7072	0	.5855	0
.7072	0	.2728	1
.1201	.4618	.5855	0
.1201	.4618	.2728	1
.8672	.4618	.5855	0
.8672	.4618	.2728	1
.3627	1	.5855	0
.3627	1	.2728	1

표 6-10. Loop-1의 결론부 파라메터

p_0	p_1	p_2	p_3
9.083098E-06	3.060412E-04	2.521482E-03	-2.572884E-04
9.286980E-02	2.434622E-02	1.045023E-01	1.678540E-01
1.358987E-01	1.022834E-01	1.901114E-03	2.276451E-01
2.264430E-01	2.288217E-01	2.280312E-01	4.980905E-02
2.276957E-01	2.107476E-01	2.479026E-02	1.510522E-01
1.255228E-01	2.273882E-01	-1.998743E-01	-1.981687E-01
-1.953659E-01	-2.007091E-01	6.081709E-02	-1.828377E-01
-1.010641E-01	5.089072E-02	-5.452921E-02	-3.297240E-02
-1.999194E-01	9.999325E-01	9.977873E-01	9.820902E-01
1.001848	2.496811E-01	8.461466E-01	3.178515E-01
3.529404E-02	1.819319E-01	3.133413E-01	9.929867E-01

표 6-11. Loop-2의 결론부 파라메터

p_0	p_1	p_2	p_3
-3.984950E-05	-2.112143E-04	-4.545483E-03	-4.299697E-05
-7.240065E-02	-9.787425E-02	-9.551924E-02	-1.017420E-01
-6.448774E-02	-7.116760E-02	-1.442773E-04	-5.664988E-02
-5.558926E-02	-5.889896E-02	-5.758184E-02	-2.381591E-02
-3.543465E-02	-2.038205E-02	2.479633E-03	9.635296E-02
1.427214E-02	-5.703261E-02	4.680140E-02	4.492212E-02
5.045434E-02	4.749054E-02	2.399387E-02	4.215046E-02
2.282067E-02	1.003337E-03	-8.724957E-02	-4.133355E-03
4.719295E-02	9.995689E-01	9.976844E-01	9.508845E-01
9.995335E-01	3.058110E-01	8.664101E-02	8.931703E-02
2.125727E-02	3.449558E-01	3.342425E-01	9.984765E-01

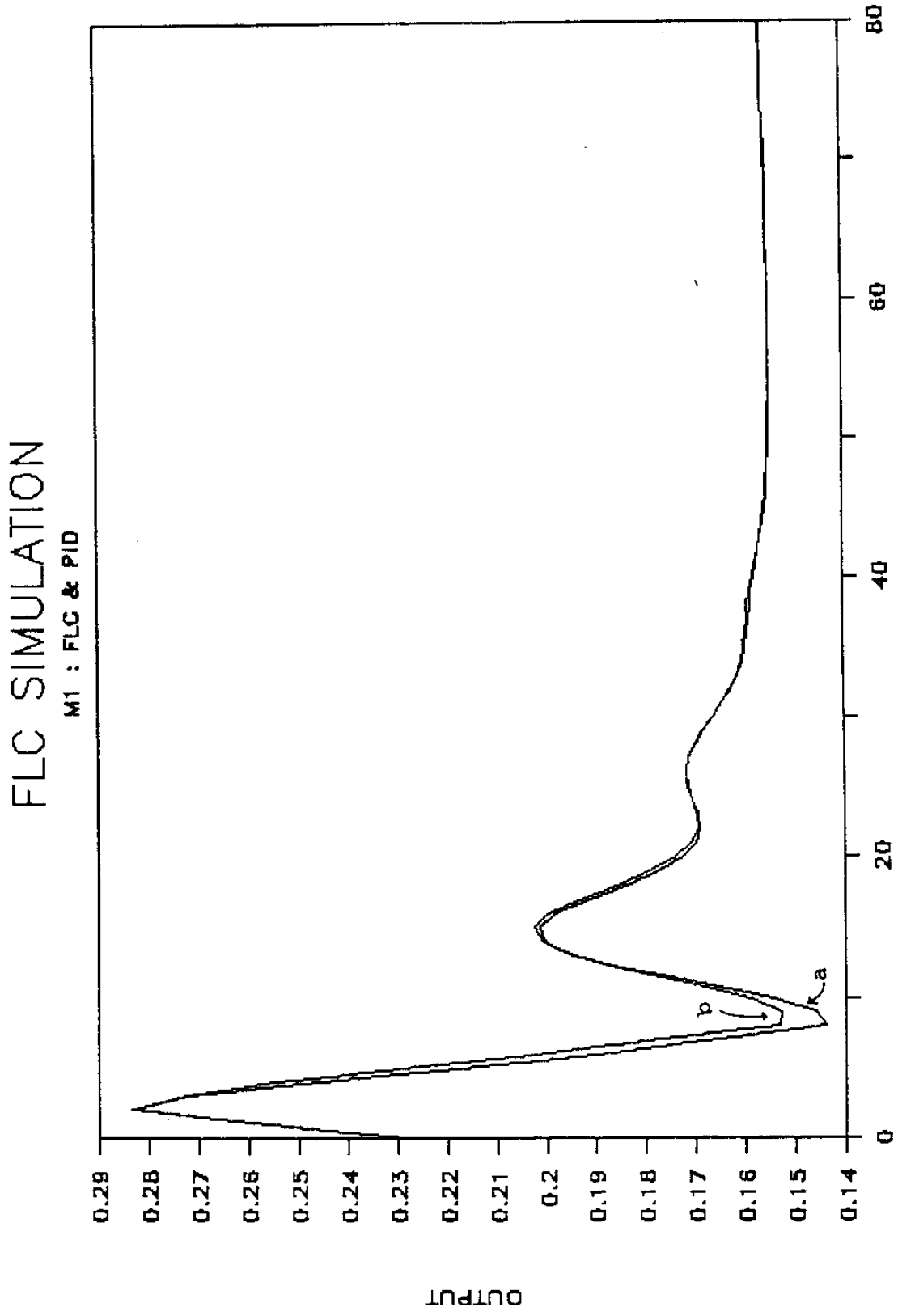


그림 6-19. FLC Simulation(M1)

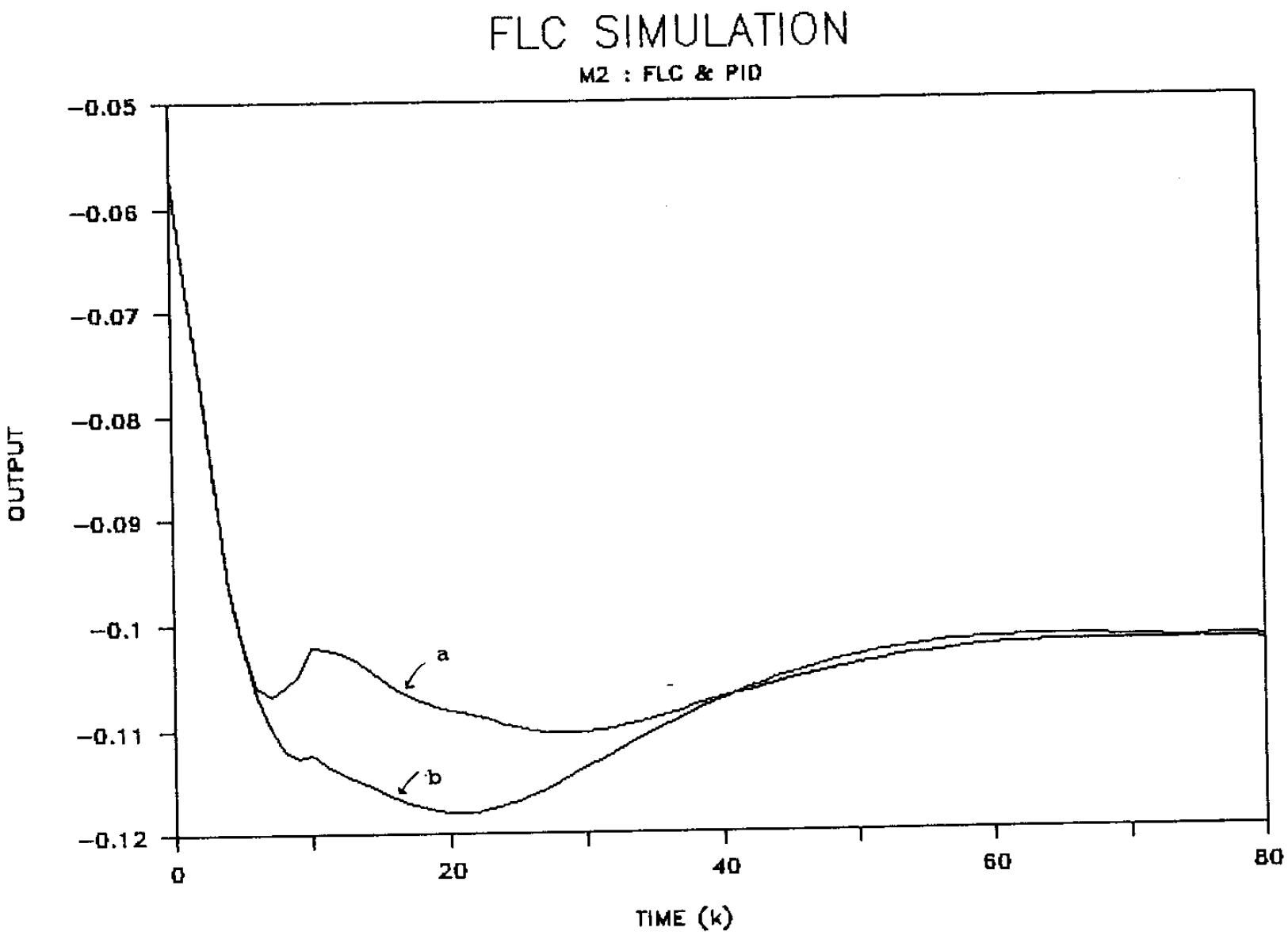


그림 6-20. FLC Simulation(M2)

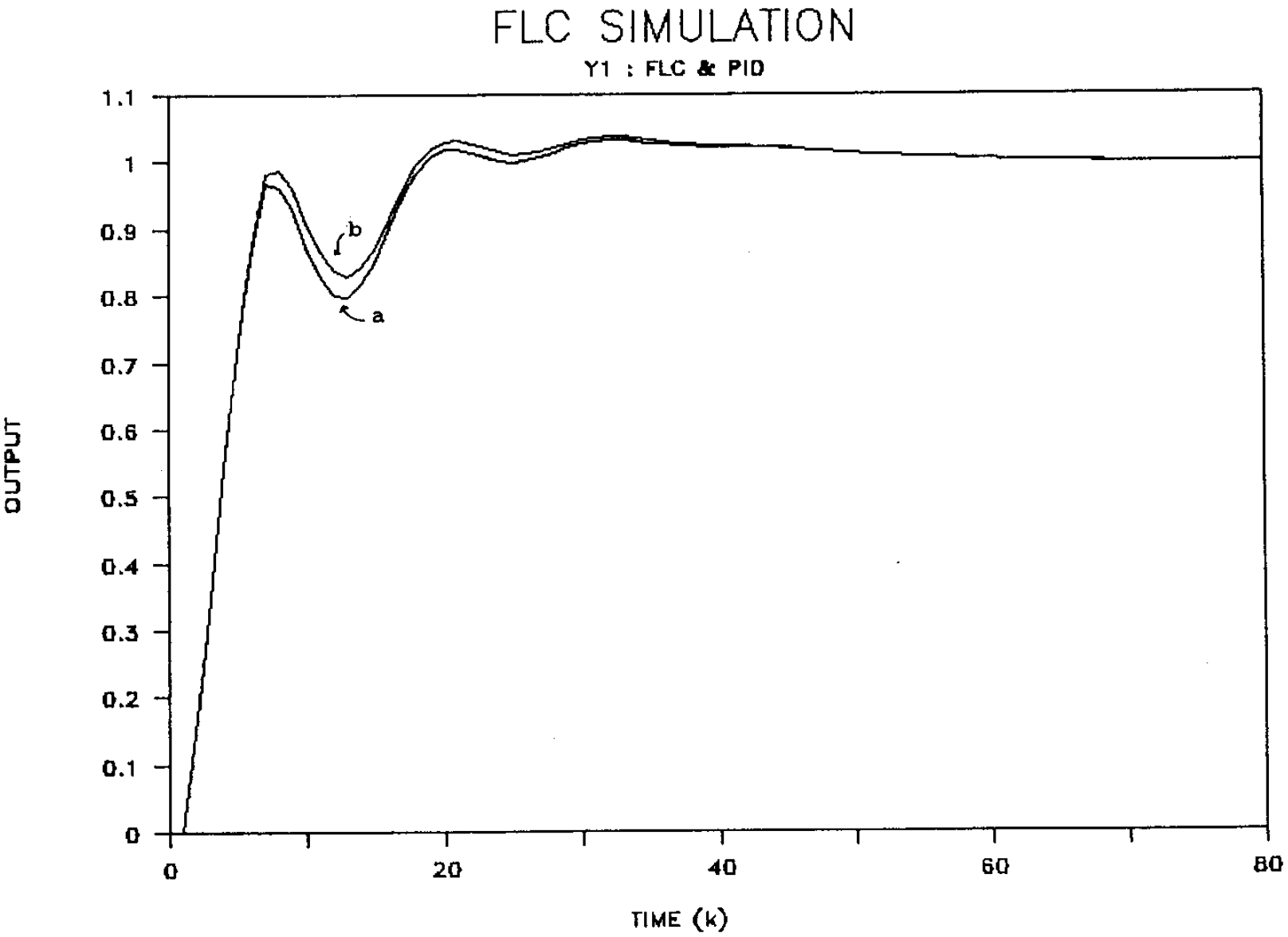


그림 6-21. FLC Simulation(Y1)

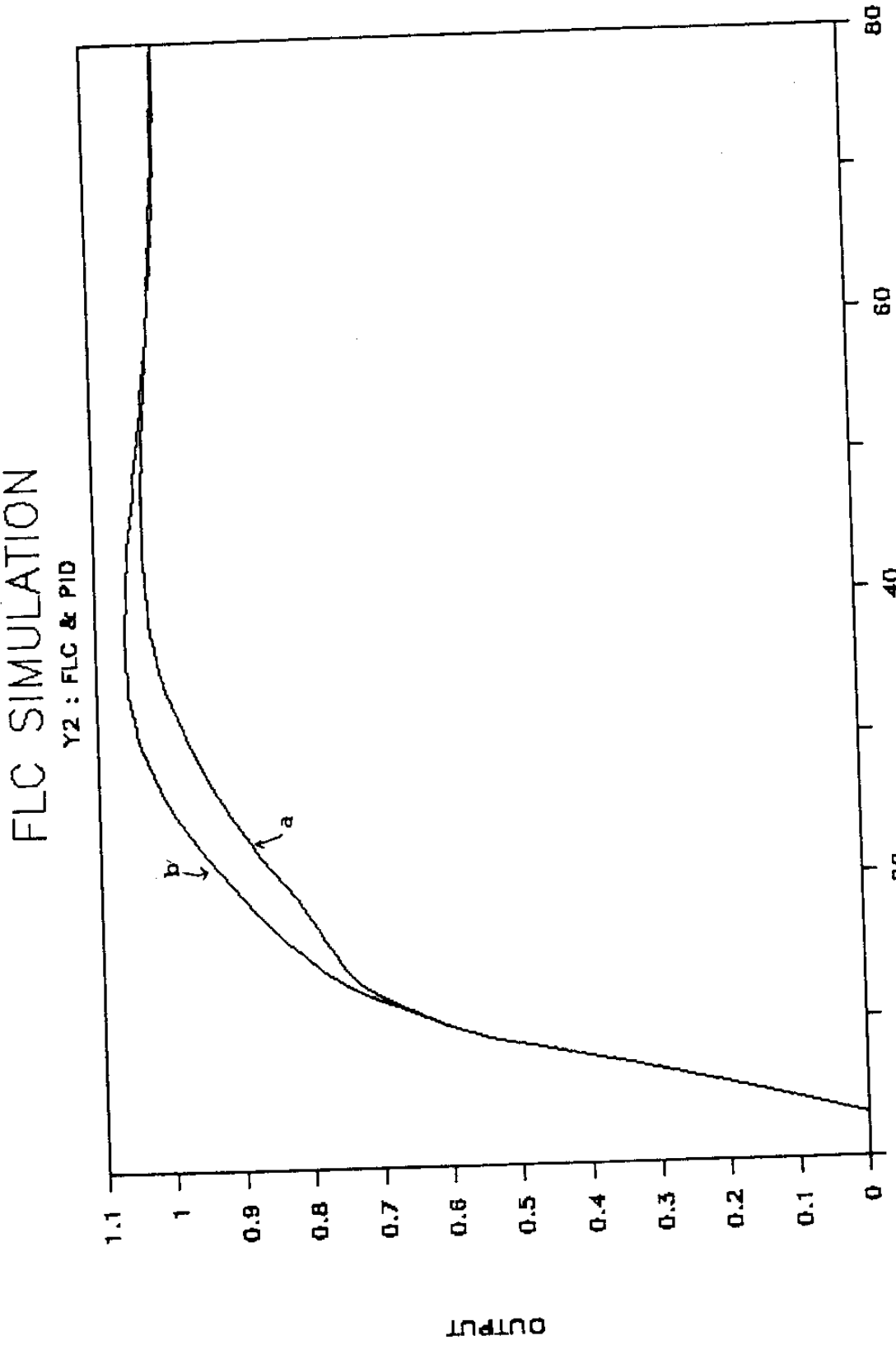


그림 6-22. FLC Simulation(Y2)

제 6 절 결론 및 제안

본장에서는 제어용 전문가시스템의 개발을 위하여 엄밀한 수학적인 모델링에 의존하지 않으며 전문가의 경험과 지식을 실제 공정의 제어에 이용할 수 있도록하는 제어방법을 연구하였다. 기존의 제어 방법과는 다르게 공정의 측정변수를 정량적으로 분석하지 않고 정성적인 판단과 축적된 경험을 공정제어에 반영하는 제어용 전문가시스템의 기본 모듈인 SPC 와 FMA 를 개발하였다. 개발된 기본 모듈을 질적으로 향상시키는 일과 더불어 기존의 제어 이론을 모듈화하는 작업이 차후 수행되어야 만한다.

개발된 기본모듈의 모사 실험으로 부터 실제 공정에의 적용 가능성이 확인되었고 앞으로 이를 이용하여 실제 공정의 조업자료를 이용한 타당성 검토와 실험이 수반되어야 한다. 기존의 연구사례를 살펴 보면 FLC 를 이용하여 작게는 기본적인 단위 공정의 제어에서 부터 크게는 대단위 화학공정의 제어에 대한 응용사례가 보고되고 있다. 이러한 기존의 연구사례를 프로그램화하여 기본 모듈을 확장하고 각각의 모듈을 종합한 제어용 전문가시스템으로 발전시키기 위해서는 부분적인 기능에 대한 연구와 더불어 시스템 전체에 대한 구조와 상관관계에 대한 많은 연구가 요구된다. 또한 실제적인 조업 경험을 지닌 전문가와의 많은 대화를 통해 지식 베이스를 확장하는 작업도 중요한 과제이다.

참 고 문 헌

(1) Zadeh L.A., Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. SMC-3, 1, 28-44 (1973).

(2) Mamdani E.H., Application of fuzzy algorithms for control of simple dynamic plant, Proc. IEEE, **Vol.121**, No.12, 1585-1588 (1974).

(3) Tong R.M., M.B. Beck, and A. Latten, Fuzzy control of the activated sludge wastewater treatment process, Automatica, **Vol.16**, 659-701 (1980).

(4) Holmblad, L.P. and J.J. Østergaard, Control of a cement by fuzzy logic, Fuzzy Information and Decision Processes, North-Holland, Amsterdam (1982)

(5) King R.E., Fuzzy logic control of a cement kiln precalciner flash furnace, Proc. of the IEEE conf. on applications of adaptive and multivariable control, Hull, U.K., (1982).

(6) Sripada N.R., D.G. Fisher and A.J. Morris, AI application for process regulation and servo control, IEE proc. **Vol.134**, Pt. D, No.4 (1987)

(7) Takagi Tomohiro and Michio Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. on SMC, **Vol.15**, No.1 (1985)

제 7 장 결론 및 향후 연구 추진계획

제 1 절 결 론

본 연구의 1차년도 목표는 기본적인 전문가 시스템 언어 및 구조의 확립과 pilot 플랜트의 구성작업이었다. Pilot 플랜트의 구성에 있어서는 실험용 증류탑의 제작이 진행중이며 시스템의 on-line화를 위한 설계 및 장치조립 작업이 계획중이며 아울러 시험용 반응기의 제작이 추진중에 있다.

공정용 전문가 시스템 구조의 확립에 있어서는 일차적으로 사용 언어와 기본적인 골격의 선정을 토대로 공정지식의 효율적처리를 위한 데이타 운용시스템의 구축작업이 진행되었다. 아울러 공정지식 표현의 매카니즘과 지식운용기능의 구성이 이루어 졌으며 공정의 진단 및 제어기법에 대한 연구가 진행되었다.

공정 데이타의 표현 및 운용은 tree 구조 및 linked list 구조와 이에 바탕을 둔 frame식 구조에 의해 효율적으로 이루어질 수 있음이 확인되었다. 실질적인 데이타의 표현은 IF-THEN형식의 production rule과 frame 방법에 의해 이루어지고 데이타 베이스에 입력된다. 공정 데이타 베이스의 검색 및 보정 보완을 위해서는 depth-first 및 breadth-first search 방법과 이들의 혼합 및 iteration기능을 가미한 방법이 이용되는데 데이타 베이스의 구조에 따라 적절한 방법이 이용되도

록 하였다. 이러한 데이타 베이스의 search 기법은 제 4 장에서 소개된 전문가 시스템에서 실제적으로 이용되고 있다.

화학공정에서 다룰수 있는 모든 종류의 데이타를 포괄적으로 처리운영할 수 있는 데이타 베이스의 구축에는 많은 시간과 노력이 필요하다. 그러나 기본적인 골격을 제대로 갖추어 놓으면 사용자의 요구에 따라 적절한 format을 갖춘 데이타 처리구조를 용이하게 만들어낼 수 있으며 특수한 기능이 요구되는 공정의 표현을 위해서 필요한 구조를 갖는 데이타 베이스를 손쉽게 구성할 수 있다. 그러나 물성치나 일반적인 자료등의 데이타는 이미 개발되어 있는 데이타 베이스를 활용하는 것이 보다 능률적일 것이다.

공정 조업용 prototype 전문가 시스템의 개발에 있어서는 프로그래밍 언어로서 C를 사용하여 기본적인 지식 베이스의 구축 및 운용과정과 초보적인 공정진단과정을 구성하였다. 비교적 간단한 장치나 공정의 조업지식의 표현 및 운용이나 진단은 현재 구축된 시스템으로 가능함이 확인되었다. 본 연구의 2 차년도에서는 제 5 장과 6 장에서 다루고 있는 진단 및 제어 매카니즘과 몇가지의 기법들이 시스템에 조합될 것이며 아울러 따로 구입하게 될 시스템 shell과 의 비교 활용을 통해 보정과 보완작업이 계속될 것이다.

공정 진단기법에 있어서는 대상이 되는 공정의 모델링과 identification을 통해서 공정의 이상유무를 판단하는 방법이 연구되었다. 공정진단기법을 이용하여 최근 경제성이 높은 이송 수단으로 많이 사용되는 파이프라인의 유출탐지 방법을 연구하였다. 이론적인 고찰과 묘사를 통해 안전성 향상과 환경오염 방지의 차

원에서 실제공정에 대한 적용가능성을 확인할 수 있었다. 단위공정의 복잡한 network로 구성되는 공장의 조업현장에서 야기되는 상황의 처리를 위한 tree구조 활용기법이나 전파 유형의 파악 및 응용기법은 현재 국내에 축적되어 있는 연구 결과들을 토대로 이후 연구될 것이다.

공정의 제어를 위한 전문가 시스템 연구에서는 일차적으로 현재 해외에서도 그에 대한 연구가 활발히 진행되고 있는 fuzzy logic 제어기법이 주 연구대상이었다. 조업 현장에서의 경험적 지식이 중요한 요소가 되는 화학공정의 제어에 있어서 fuzzy logic 제어기법은 전문가 시스템에서 특히 유용하게 활용될 수 있는 제어방법이다. 이외에 예측제어 기법이나 적응제어 방법들로서 본 연구팀에 의해 이미 개발된 시스템들이 보완되어서 전문가 시스템에 적용될 것이다.

제2절 향후 연구 추진계획

공정 전문가 시스템의 개발에 있어서는 일차적으로 개발대상이 되는 공정을 선택한 다음 적절한 전문가 시스템 구축용 tool이나 shell을 구입하여 지식 베이스와 utility들을 정하여 주는 방식이 지금까지의 거의 일관된 개발패턴이었다. 시간과 경비문제를 고려한다면 이러한 패턴은 물론 가장 경제적인 방법이 될수 있다. 그러나 소규모 공정이나 어느 특정공정 또는 사용자가 원하는 기능들을 요구에 맞게 충족시켜 주기 위해서는 기본적인 전문

가 시스템 software 기술의 축적이 불가피하다. 전문가 시스템 software의 개발은 나아가 공정 인공지능 기술의 해외 예속화 탈피를 위해서도 필수적으로 수행하여야 할 과제이다.

이후의 연구에 있어서는 우선 1차년도에 연구된 공정 조업용 데이타 베이스와 기본적인 전문가 시스템 구조의 보정 보완작업이 계속될 것이며 진단 및 제어기법의 시험 적용을 통해 공정용 전문가 시스템의 기본 구성이 이루어질 것이다. 아울러 시판되고 있는 전문가 시스템 shell을 구입하여 개발중인 시스템과의 비교검토를 통해 기존 시스템의 개선작업과 공정 및 사용자의 용도에 상응하는 최적 시스템의 확립이 이루어질 것이다. 시험용 pilot 장치의 구성은 2차년도에 완결될 것인데 기본 시스템의 실제 적용시험이 함께 진행될 예정이다.

부 록

1. 시스템 프로그램
2. Model Based Fault Diagnosis 프로그램
3. Fuzzy Logic Control 프로그램

1. 시스템 프로그램

```
/*
**
---------------------
  initialize_record
---------------------
**
** Initialize record list.
**
*/
void initialize_record(void)
{
num_items = 0;
num_records = 0;
num_events = 0;
sort_item = 0;
first_record = cur_record = last_record = NULL;
}

/*
**
-----------------
  define_record
-----------------
**
** Define linked_list data structure.
** First free any previously allocated memory.
** Up to MAX_ITEMS items can be stored in a record.
**
*/
void define_record(void)
{
char ch;
int i;
struct record_list *temp;

clear_screen();
while(first_record)
   {
   temp = first_record->next;
   free(temp->data);
   free(temp);
   first_record = temp;
   }

initialize_record();

record_size = 0;
do
   {
   goto_xy(0, 3);
   printf("Number of items in a record (1-%d) (0 to skip): ", MAX_ITEMS);
   scanf("%d%*c", &num_items);
   if(!num_items) return;
   } while(num_items<0 || num_items>MAX_ITEMS);

for(i=0; i<num_items; i++)
   {
   clear_screen();
   goto_xy(0, 3);
   items[i].item_position = record_size;
```

```
    printf("Enter title of item %d (1-%d characters): ", i+1, MAX_NAME);
    get_data(items[i].item_title, MAX_NAME);

    do
      {
      printf("\n\nEnter length of the item in bytes (0-%d): ", MAX_LLEN);
      scanf("%d%*c", &items[i].item_size);
      } while(items[i].item_size<0 || items[i].item_size>MAX_LLEN);
    items[i].item_size++;

    do
      {
      printf("\n\nEnter x and y coordinates of the item in the screen: ");
      scanf("%d%d%*c", &items[i].x, &items[i].y);
      } while(items[i].x<0 ||
              items[i].y<0 ||
              items[i].x+items[i].item_size>=MAX_LLEN ||
              items[i].y>=MAX_LIN);

    printf("\n\nIs this item defined OK? (Y/N): ");
    ch = getche();
    if(tolower(ch) != 'y') i--;
    else record_size += items[i].item_size;
    }
  printf("\n\nPress a key to see items and screen layout");
  getch();
  clear_screen();
  display_items();
  sort_item = select_item("Select item to sort on");
  }

  /*
  **
  ---------------
    enter_record
  ---------------
  **
  ** Enter data into the record_list structure.
  **
  */
  int enter_record(void)
  {
  int i, itm;
  char ch;
  struct record_list *pt;

  for(;;)
    {
    pt = malloc(sizeof(struct record_list)); /* memory for list structure */
    if(!pt)
      {
      printf("out of memory\n");
      return 0;
      }
    pt->data = malloc(record_size); /* memory for node data */
    if(!pt->data)
      {
      printf("out of memory\n");
      return 0;
      }
```

```
    clear_screen();
    display_items();

    for(i=0; i<num_items; i++) /* input data node */
       {
       goto_xy(items[i].x+strlen(items[i].item_title)+5, items[i].y);
       get_data(&(pt->data)[items[i].item_position], items[i].item_size-1);
       if(!(*pt->data+items[0].item_position)) return 1;
       }
    num_records++;

    comment2("Is entry OK? (Y/N): ");
    ch = getche();
    ch = tolower(ch);
    while(ch != 'y')
       {
       itm = select_item("Select item to modify");
       change_item(pt->data, itm);
       comment2("Is entry OK? (Y/N): ");
       ch = getche();
       ch = tolower(ch);
       }
    store_node(pt, &first_record, &last_record);
    }
}

/*
**
------------------
  display_items
------------------
**
** Display a blank template of the items.
**
*/
void display_items(void)
{
int i, j;

for(i=0; i<num_items; i++)
   {
   goto_xy(items[i].x, items[i].y);
   printf("  %s :  ", items[i].item_title);
   for(j=0; j<items[i].item_size-1; j++) putch(' ');
   }
}

/*
**
------------------
  display_record
------------------
**
** Display a record content.
**
*/
void display_record(char *thing)
{
int i;
```

```
display_items();
for(i=0; i<num_items; i++)
   {
   goto_xy(items[i].x+strlen(items[i].item_title)+5, items[i].y);
   printf("%s", thing+items[i].item_position);
   }
}

/*
**
---------------
  select_item
---------------
**
** Select an item
**
*/
int select_item(char *string)
{
union which_item {
                   int i;
                   char ch[2];
                 } key;
int i, len;

comment1(string);
for(i=0; ;)
   {
   len = strlen(items[i].item_title)+5;
   goto_xy(items[i].x+len, items[i].y);
   key.i = _bios_keybrd(0);
   if(!key.ch[0])
      switch(key.ch[1])
         {
         case 72: if(i) i--; break;              /* Up */
         case 80: if(i<num_items-1) i++; break; /*Down */
         }
   else if(key.ch[0] == '\r') return i;
   }
}

/*
**
-------------
  save_list
-------------
**
** Save record-list database file to a disk file.
** Saves both data and description.
**
*/
int save_list(char *file_name)
{
struct record_list *info;
int i;
FILE *fp;
char name[FILE_NAME_SIZE];

/* store the data in a .DAT file */
strcpy(name, file_name);
```

```
    strcat(name, ".dat");
    fp = fopen(name, "wb");
    if(!fp)
      {
      printf("cannot open database file\n");
      return 0;
      }
    comment2("Saving data...");

    info = first_record;
    while(info)
      {
      fwrite(info->data, record_size, 1, fp);
      info = info->next;
      if(ferror(fp))
        {
        printf("error writing file\n");
        fclose(fp);
        return 0;
        }
      }
    fclose(fp);

    /* store item description, number of items, and record sizes
       into a .DEF file */
    strcpy(name, file_name);
    strcat(name, ".def");
    fp = fopen(name, "wb");
    if(!fp)
      {
      printf("cannot open definition file\n");
      fclose(fp);
      return 0;
      }
    putw(record_size, fp);
    putw(num_items, fp);
    putw(sort_item, fp);
    for(i=0; i<num_items; i++)
      {
      fwrite(&items[i], sizeof(items[i]), 1, fp);
      if(ferror(fp))
        {
        printf("error writing file\n");
        fclose(fp);
        return 0;
        }
      }
    fclose(fp);
    return 1;
    }

    /*
    **
    -------------
     load_list
    -------------
    **
    ** Load record_list file.
    **
    */
```

```
int load_list(char *file_name)
{
struct record_list *info;
FILE *fp1, *fp2;
int i;
char name[FILE_NAME_SIZE];

strcpy(name, file_name);
strcat(name, ".dat");
fp1 = fopen(name, "rb");
if(!fp1)
   {
   printf("cannot open file\n");
   return 0;
   }
strcpy(name, file_name);
strcat(name, ".def");
fp2 = fopen(name, "rb");
if(!fp2)
   {
   printf("cannot open file\n");
   fclose(fp1);
   return 0;
   }
while(first_record)
   {
   info = first_record->next;
   free(info->data);
   free(info);
   first_record = info;
   }
first_record = last_record = cur_record = NULL;
num_records = 0;

comment2("Loading file...");

record_size = getw(fp2);
num_items = getw(fp2);
sort_item = getw(fp2);
for(i=0; i<num_items; i++)
   {
   if(1 != fread(&items[i], sizeof(items[i]), 1, fp2))
      {
      printf("error reading file\n");
      fclose(fp1);
      fclose(fp2);
      getch();
      return 0;
      }
   }
fclose(fp2);
while(!feof(fp1))
   {
   info = malloc(sizeof(struct record_list));
   if(!info)
      {
      printf("out of memory");
      fclose(fp1);
      return 0;
      }
```

```
  info->data = malloc(record_size);
  if(!info->data)
     {
     printf("out of memory");
     fclose(fp1);
     return 0;
     }
  if(1 != fread(info->data, record_size, 1, fp1))
     {
     if(ferror(fp1))
        {
        printf("error reading data file\n");
        fclose(fp1);
        return 0;
        }
     fclose(fp1);
     cur_record = first_record;
     return 1;
     }
   store_node(info, &first_record, &last_record);
   num_records++;
   }
fclose(fp1);
cur_record = first_record;
return 0;
}

/*
**
---------------
  revise_list
---------------
**
** Revise a data record in the record_list.
** Return a pointer to that record.
**
*/
struct record_list *revise_list(void)
{
struct record_list *info;
char ch, which_item[MAX_STR];
int itm;

clear_screen();
display_items();
itm = select_item("Select item");
comment1("Enter record to search for: ");
get_data(which_item, items[itm].item_size);
info = find(which_item, first_record, itm);
while(info)
   {
   display_record(info->data);
   comment1("Revise this record? (Y/N): ");
   ch = getche();
   ch = tolower(ch);
   if(ch != 'y') info = find(which_item, info->next, itm);
   else break;
   }
if(!info) return NULL;
```

```
itm = select_item("Select item to revise");
change_item(info->data, itm);
return info;
}

/*
**
---------------
  change_item
---------------
**
** Change the specified item.
**
*/
void change_item(char *data, int itm)
{
int i;

display_record(data);
comment2("Enter new data");
goto_xy(items[itm].x, items[itm].y);
printf("%d) %s :  ", itm+1, items[itm].item_title);

for(i=0; i<items[itm].item_size-1; i++) putch(' ');
i = strlen(items[itm].item_title)+5;
goto_xy(items[itm].x+i, items[itm].y);
get_data(data+items[itm].item_position, items[itm].item_size-1);
}

/*
**
----------------
  examine_list
----------------
**
** Walk through the record_list.
** Return pointer to the last data record examined.
**
*/
struct record_list *examine_list(void)
{
union which_item {
                   int i;
                   char ch[2];
                 } key;
struct record_list *info;

clear_screen();
clear_line(0, MAX_LIN-2);
clear_line(0, MAX_LIN-1);
clear_line(0, MAX_LIN);
goto_xy(0, MAX_LIN-2);
printf("_____________________________________________________________________________");
printf(" Down arrow: forward     Up arrow: backward     Home: beginning\n");
printf(" End: end                Return: to main menu");
if(cur_record) info = cur_record;
else info = first_record;
while(info)
   {
   display_record(info->data);
```

```
    key.i = _bios_keybrd(0);
    if(!key.ch[0])
      switch(key.ch[1])
        {
        case 71: /* Home */
          info = first_record;
          break;
        case 72: /* Up */
          if(info->prev) info = info->prev;
          break;
        case 79: /* End */
          info = last_record;
          break;
        case 80: /* Down */
          if(info->next) info = info->next;
          break;
        }
   else return info;
   }
 return NULL;
 }

 /*
 **
 --------
  find
 --------
 **
 ** Find an entry and return a pointer to it.
 **
 */
 struct record_list *find(char *which_item, struct record_list *at, int itm)
 {
 struct record_list *info;

 info = at;
 while(info)
   {
   if(!strcmp(which_item, info->data+items[itm].item_position)) return info;
   info = info->next;
   }
 return NULL;
 }

 /*
 **
 -------------
  find_node
 -------------
 **
 ** Find a data record in the record_list.
 ** Return a pointer to the last match found.
 ** It displays the record on the screen.
 **
 */
 struct record_list *find_node(void)
 {
 int itm;
 char ch, which_item[MAX_STR];
 struct record_list *info, *temp;
```

```
clear_screen();
display_items();
itm = select_item("Select item to search on");
comment2("Enter item to find: ");
get_data(which_item, items[itm].item_size);
ch = '\0';

info = find(which_item, first_record, itm);
while(info && ch != 'q')
  {
  temp = info;
  display_record(info->data);
  info = find(which_item, info->next, itm);
  if(info) printf("\nmore, ");
  else printf("\n\nNo more matches, ");
  printf("press a key ('Q' to quit): ");
  ch = getch();
  ch = tolower(ch);
  }
return temp;
}

/*
**
--------------
 store_node
--------------
**
** Create a doubly linked record-list database.
** A node can be put into the unit list at the start, in the middle,
** or at the end of the list.
**
*/
void store_node(struct record_list *new, struct record_list **first_record,
                struct record_list **last_record)
{
struct record_list *temp, *curr;

if(*last_record == NULL)
  {
  new->next = NULL;
  new->prev = NULL;
  *first_record = new;
  *last_record = new;
  return;
  }

curr = *first_record;
temp = NULL;
while(curr)
  {
  if(strcmp(curr->data+items[sort_item].item_position,
            new->data+items[sort_item].item_position)<0)
    {
    temp = curr;
    curr = curr->next;
    }
  else
    {
```

```
        if(curr->prev)
          {
          curr->prev->next = new;
          new->next = curr;
          new->prev = curr->prev;
          curr->prev = new;
          return;
          }
        new->next = curr;
        new->prev = NULL;
        curr->prev = new;
        *first_record = new;
        return;
        }
      }
    temp->next = new;
    new->next = NULL;
    new->prev = temp;
    *last_record = new;
    }

    /*
    **
    ---------------
      delete_item
    ---------------
    **
    ** Delete an element from the record_list.
    **
    */
    void delete_item(struct record_list **head, struct record_list **last_record)
    {
    struct record_list *info;
    char ch, which_item[MAX_STR];
    int itm;

    clear_screen();
    display_items();
    itm = select_item("Select item");
    comment1("Enter item: ");
    get_data(which_item, items[itm].item_size);

    info = find(which_item, first_record, itm);
    while(info)
      {
      display_record(info->data);
      comment1("Delete this record? (Y/N): ");
      ch = getche();
      ch = tolower(ch);
      if(ch != 'y') info = find(which_item, info->next, itm);
      else break;
      }
    if(!info) return;
    if(cur_record == info) cur_record = NULL;
    num_records--;

    if(*head == info)
      {
      *head = info->next;
      if(*head) (*head)->prev = NULL;
```

```
    else *last_record = NULL;
    }
  else
    {
    info->prev->next = info->next;
    if(info != *last_record) info->next->prev = info->prev;
    else *last_record = info->prev;
    }
  free(info->data);
  free(info);
  }

  /*
  **
  ------------
    comment1
  ------------
  **
  ** Display a message
  **
  */
  void comment1(char *string)
  {
  clear_line(0, MAX_LIN);
  goto_xy(0, MAX_LIN);
  printf(string);
  }

  /*
  **
  ------------
    comment2
  ------------
  **
  ** Display a message
  **
  */
  void comment2(char *string)
  {
  clear_line(0, MAX_LIN-4);
  goto_xy(0, MAX_LIN-4);
  printf(string);
  }

  /*
  **
  ------------
    get_data
  ------------
  **
  ** Read data string from the keyboard.
  ** It does not echo carriage RETURN.
  ** Use backspace to correct.
  **
  */
  void get_data(char *string, int length)
  {
  char *ps;
  int i;
```

```
ps = string;
for(i=0; ;)
   {
   *string = getch();
   if(*string == '\r') /* RETURN entered */
      {
      *string = '\0';  /* assign NULL */
      return;
      }
   if(*string == '\b') /* backspace entered */
      {
      if(string > ps)
         {
         string--;
         putch('\b');
         putch(' ');
         putch('\b');
         i--;
         }
      }
    else if (i<length)
      {
      putch(*string);
      string++;
      i++;
      }
   }
}

/*
**
-----------
  goto_xy
-----------
**
** Send cursor to specified (x, y) position
**
*/
void goto_xy(int x, int y)
{
union REGS i;

i.h.dh = y; /* row coordinate */
i.h.dl = x; /* column coordinate */
i.h.ah = 2; /* cursor addressor */
i.h.bh = 0; /* use video page 0 */
int86(16, &i, &i);
}

/*
**
--------------
  clear_line
--------------
**
** Clear to end of specified line.
**
*/
void clear_line(int x, int y)
{
```

```
goto_xy(x, y);
for(; x<MAX_LLEN; x++) printf(" ");
}

/*
**
----------------
  clear_screen
----------------
**
** Clear the screen.
**
*/
void clear_screen(void)
{
union REGS r;

r.h.ah = 6;
r.h.al = 0;         /* number of lines to scroll */
r.h.ch = 0;         /* upper left row */
r.h.cl = 0;         /* upper left column */
r.h.dh = MAX_LIN;   /* lower right row */
r.h.dl = MAX_LLEN; /* lower left column */
r.h.bh = 7;         /* use video page 7 */
int86(0x10, &r, &r);
}
```

```
/*
**
-----------------
  initialize_db
-----------------
**
** Initialize database.
**
*/
void initialize_db(void)
{
char *obj_title = "OBJECT CLASS";
char *sup_name = "SUPER CLASS";
char *sub_name = "SUB CLASS";
char *formula_name = "FORMULA";
char *item_name  = "ITEM TITLE";
char *state_name  = "STATUS";

clear_screen();
cl_pos = -1;

strcpy(field_cl.class_title, obj_title);
strcpy(field_cl.sup_title, sup_name);
strcpy(field_cl.sub_title, sub_name);

strcpy(field_fr.item_title, item_name);
strcpy(field_fr.formula_title, formula_name);
strcpy(field_fr.state_title, state_name);

field_cl.obj_x = 0;
field_cl.obj_y = 4;
field_cl.sup_x = field_cl.obj_x;
field_cl.sup_y = field_cl.obj_y + 2;
field_cl.sub_x = field_cl.sup_x;
field_cl.sub_y = field_cl.sup_y + 2;

field_fr.item_x = field_cl.sub_x;
field_fr.item_y = field_cl.sub_y + 4;
field_fr.formula_x = field_fr.item_x;
field_fr.formula_y = field_fr.item_y + 2;
field_fr.state_x = field_fr.formula_x;
field_fr.state_y = field_fr.formula_y + 2;
}

/*
**
------------
  enter_db
------------
**
** Enter database values.
**
*/
void enter_db(void)
{
struct frame *info,*temp;
struct record *sp, *sb, *cur;
int i, j, k;

clear_screen();
```

```
iisplay_class_title();
display_record_title();

for( ; ;)
   {
   i = get_db_position(); /* get next available database position */
   if(i == -1)
      {
      printf("Out of database space.\n");
      return;
      }

   goto_xy(field_cl.obj_x+strlen(field_cl.class_title)+5, field_cl.obj_y);
   for(k=0; k<MAX_LLEN; k++) printf(" ");
   goto_xy(field_cl.obj_x+strlen(field_cl.class_title)+5, field_cl.obj_y);
   get_data(classes[i].class_name, MAX_STR);
   if(!*classes[i].class_name)
      {
      cl_pos--;
      break;
      }

   /* enter superclass item */
   /* allocate memory for class record */
   sp = (struct record *) malloc(sizeof(rd));
   if(sp == NULL)
      {
      printf("Out of memory.\n");
      return;
      }
   classes[i].super_class = sp;
   goto_xy(field_cl.sup_x+strlen(field_cl.sup_title)+5, field_cl.sup_y);
   for(k=0; k<MAX_LLEN; k++) printf(" ");
   for( ; ;)
      {
      get_data(sp->property, MAX_LLEN);
      if(!sp->property[0]) break;
      cur = sp;
      sp->next = (struct record *) malloc(sizeof(rd));
      if(sp->next == NULL)
         {
         printf("Out of memory.\n");
         return;
         }
      sp = sp->next;
      sp->next = NULL;
      goto_xy(field_cl.sup_x+strlen(field_cl.sup_title)+5, field_cl.sup_y);
      for(k=0; k<MAX_LLEN; k++) printf(" ");
      }
    cur->next = NULL;

    /* enter subclass item */
    /* allocate memory for class record */
    sb = (struct record *) malloc(sizeof(rd));
    if(sb == NULL)
      {
      printf("Out of memory.\n");
      return;
      }
    classes[i].sub_class = sb;
```

```
goto_xy(field_cl.sub_x+strlen(field_cl.sub_title)+5, field_cl.sub_y);
for(k=0; k<MAX_LLEN; k++) printf(" ");
for( ; ;)
  {
  get_data(sb->property, MAX_LLEN);
  if(!sb->property[0]) break;
  cur = sb;
  sb->next = (struct record *) malloc(sizeof(rd));
  if(sb->next == NULL)
    {
    printf("Out of memory.\n");
    return;
    }
  sb = sb->next;
  sb->next = NULL;
  goto_xy(field_cl.sub_x+strlen(field_cl.sub_title)+5, field_cl.sub_y);
  for(k=0; k<MAX_LLEN; k++) printf(" ");
  }
cur->next = NULL;

/* enter frame information for the class i */
/* allocate memory for frame */
info = (struct frame *) malloc(sizeof(fr));
if(info == NULL)
  {
  printf("Out of memory.\n");
  return;
  }
classes[i].class_record = info;

/* enter frame information */
for(j=0; j<sizeof(info->name); j++) info->name[j]=' ';
for(j=0; j<sizeof(info->formula); j++) info->formula[j]=' ';
for(j=0; j<sizeof(info->status); j++) info->status[j]=' ';
comment1("Enter class values (RETURN to quit): ");
for( ; ;)
  {
  goto_xy(field_fr.item_x+strlen(field_fr.item_title)+5, field_fr.item_y);
  for(k=0; k<MAX_STR; k++) printf(" ");
  goto_xy(field_fr.formula_x+strlen(field_fr.formula_title)+5,
        field_fr.formula_y);
  for(k=0; k<MAX_STR; k++) printf(" ");
  goto_xy(field_fr.state_x+strlen(field_fr.state_title)+5,
        field_fr.state_y);
  for(k=0; k<MAX_STR; k++) printf(" ");

  goto_xy(field_fr.item_x+strlen(field_fr.item_title)+5, field_fr.item_y);
  get_data(info->name, MAX_STR);
  if(!info->name[0]) break;
  goto_xy(field_fr.formula_x+strlen(field_fr.formula_title)+5,
        field_fr.formula_y);
  get_data(info->formula, MAX_STR);
  goto_xy(field_fr.state_x+strlen(field_fr.state_title)+5,
        field_fr.state_y);
  get_data(info->status, MAX_STR);

  temp = info;
  info->next = (struct frame *) malloc(sizeof(fr));
  if(info->next == NULL)
    {
```

```
        printf("Out of memory.\n");
        return;
        }
      info = info->next;
      info->next = NULL;
      for(j=0; j<sizeof(info->name); j++) info->name[j]=' ';
      for(j=0; j<sizeof(info->formula); j++) info->formula[j]=' ';
      for(j=0; j<sizeof(info->status); j++) info->status[j]=' ';
      }
    temp->next = NULL;
    }
}

/*
**
-----------------------
  display_class_title
-----------------------
**
** Display class title.
**
*/
void display_class_title(void)
{
goto_xy(field_cl.obj_x, field_cl.obj_y);
printf(" %s: ", field_cl.class_title);
goto_xy(field_cl.sup_x, field_cl.sup_y);
printf(" %s: ", field_cl.sup_title);
goto_xy(field_cl.sub_x, field_cl.sub_y);
printf(" %s: ", field_cl.sub_title);
}

/*
**
-----------------------
  display_frame_title
-----------------------
**
** Display frame title.
**
*/
void display_frame_title(void)
{
goto_xy(field_fr.formula_x, field_fr.formula_y);
printf(" %s: ", field_fr.formula_title);
goto_xy(field_fr.item_x, field_fr.item_y);
printf(" %s: ", field_fr.item_title);
goto_xy(field_fr.state_x, field_fr.state_y);
printf(" %s: ", field_fr.state_title);
}

/*
**
--------------------
  get_db_position
--------------------
**
** Get next available d_base position.
**
*/
```

```
int get_db_position(void)
{
cl_pos++;
if(cl_pos < MAX_ITEM) return cl_pos;
else return -1;
}

/*
**
---------------
 save_d_base
---------------
**
** Save database.
**
*/
void save_d_base(char *file_name)
{
struct record *temp;
struct frame *cur;
FILE *fp;
char name[FILE_NAME_SIZE];
int i, j;

/* store the data in a .DAT file */
strcpy(name, file_name);
strcat(name, ".dat");
fp = fopen(name, "w");
if(!fp)
  {
  printf("Cannot open file\n");
  return;
  }

comment2("Saving database... \n");

for(i=0; i<=cl_pos; ++i)
  {
  for(j=0; j<sizeof(classes[i].class_name); j++)
    putc(classes[i].class_name[j], fp);

  /* save super-class list */
  temp = classes[i].super_class;
  while(temp)
    {
    for(j=0; j<sizeof(temp->property); j++) putc(temp->property[j], fp);
    temp = temp->next;
    }
  for(j=0; j<sizeof(temp->property); j++) putc('\0', fp);

  /* save sub-class list */
  temp = classes[i].sub_class;
  while(temp)
    {
    for(j=0; j<sizeof(temp->property); j++) putc(temp->property[j], fp);
    temp = temp->next;
    }
  for(j=0; j<sizeof(temp->property); j++) putc('\0', fp);

  /* save frame information */
```

```
  cur = classes[i].class_record;
  while(cur)
     {
     for(j=0; j<sizeof(cur->name); j++) putc(cur->name[j], fp);
     for(j=0; j<sizeof(cur->formula); j++) putc(cur->formula[j], fp);
     for(j=0; j<sizeof(cur->status); j++) putc(cur->status[j], fp);
     cur = cur->next;
     }
   for(j=0; j<sizeof(cur->name); j++) putc('\0', fp);
   }
putc(0, fp);
fclose(fp);
}

/*
**
---------------
  load_d_base
---------------
**
** Load database.
**
*/
void load_d_base(char *file_name)
{
struct record *info, *temp;
struct frame *cur, *pres;
FILE *fp;
char name[FILE_NAME_SIZE];
int i, j;

strcpy(name, file_name);
strcat(name, ".dat");
fp = fopen(name, "r");
if(!fp)
   {
   printf("Cannot open file\n");
   return;
   }

comment2("Loading database...\n");

/* load database file */
clear_d_base();
for(i=0; i<MAX_ITEM; ++i)
   {
   if((classes[i].class_name[0] = getc(fp)) == 0) break;
   for(j=1; j<sizeof(classes[i].class_name); j++)
      classes[i].class_name[j] = getc(fp);

   /* load super_class data */
   classes[i].super_class = (struct record *) malloc(sizeof(rd));
   info = classes[i].super_class;
   if(!info)
      {
      printf("Out of memory\n");
      return;
      }
   for( ; ;)
      {
```

```
      for(j=0; j<sizeof(info->property); j++) info->property[j] = getc(fp);
      if(!info->property[0])
        {
        temp->next = NULL;
        break;
        }
      info->next = (struct record *) malloc(sizeof(rd));
      if(!info->next) break;
      temp = info;
      info = info->next;
      }

    /* load sub-class data */
    classes[i].sub_class = (struct record *) malloc(sizeof(rd));
    info = classes[i].sub_class;
    if(!info)
      {
      printf("Out of memory\n");
      return;
      }
    for( ; ;)
      {
      for(j=0; j<sizeof(info->property); j++) info->property[j] = getc(fp);
      if(!info->property[0])
        {
        temp->next = NULL;
        break;
        }
      info->next = (struct record *) malloc(sizeof(rd));
      if(!info->next) break;
      temp = info;
      info = info->next;
      }

    /* load frame information */
    classes[i].class_record = (struct frame *) malloc(sizeof(fr));
    cur = classes[i].class_record;
    if(!cur)
      {
      printf("Out of memory\n");
      return;
      }
    for( ; ;)
      {
      for(j=0; j<sizeof(cur->name); j++) cur->name[j] = getc(fp);
      for(j=0; j<sizeof(cur->formula); j++) cur->formula[j] = getc(fp);
      for(j=0; j<sizeof(cur->status); j++) cur->status[j] = getc(fp);
      if(!cur->name[0])
        {
        cur->next = NULL;
        break;
        }
      cur->next = (struct frame *) malloc(sizeof(fr));
      if(!cur->next) break;
      pres = cur;
      cur = cur->next;
      }
  }
fclose(fp);
cl_pos = i-1;
```

```
}

/*
**
----------------
  clear_d_base
----------------
**
** Clear database before loading.
**
*/
void clear_d_base(void)
{
struct frame *info, *temp;
struct record *cur, *pres;
int i;

for(i=0; i<=cl_pos; i++)
   {
   cur = classes[i].super_class;
   while(cur)
      {
      pres = cur->next;
      free(cur);
      cur = pres;
      }
   cur = classes[i].sub_class;
   while(cur)
      {
      pres = cur->next;
      free(cur);
      cur = pres;
      }

   info = classes[i].class_record;
   while(info)
      {
      temp = info->next;
      free(info);
      info = temp;
      }
   }
}

/*
**
---------------
  init_ndata
---------------
**
** Initialize numerical data stack
**
*/
void init_ndata(void)
{
nd_pos = -1;
num_of_var = 0;
}

/*
```

```
**
---------------
  enter_ndata
---------------
**
** Enter data.
**
*/
void enter_ndata(void)
{
struct variables *pt, *temp;
char info[MAX_STR];
char name_list[MAX_ITEMS][MAX_NAME];
int i, k;

clear_screen();
init_ndata();
goto_xy(0, 3);
printf("Enter number of variables(maximum number = 15):  ");
scanf("%d%*c", &num_of_var);
for(i=0; i<num_of_var; i++)
   {
   printf("name of variable %d:  ", i+1);
   gets(name_list[i]);
   if(!name_list[i][0]) break;
   }

for(;;)
   {
   clear_screen();
   goto_xy(0, 2);
   k = get_next_position();
   if(k == -1)
      {
      printf("Out of list space.\n");
      return;
      }
   printf("\nData index (enter RETURN to quit): ");
   gets(ndata_base[k].index);
   if(!ndata_base[k].index[0])
      {
      nd_pos--;
      return;
      }
   printf("\n");
   printf("Enter data description: ");
   gets(ndata_base[k].description);
   if(!ndata_base[k].description[0])
      {
      nd_pos--;
      break;
      }
   printf("\n");
   printf("Enter variable values:\n\n");
   i = 0;
   for( ; ;)
      {
      if(i >= num_of_var) break;
      pt = (struct variables *) malloc(sizeof(va));
      if(pt == '\0')
```

```
        {
        printf("Out of memory.\n");
        return;
        }
      ndata_base[k].var_value = pt;
      strcpy(pt->var_name, name_list[i]);
      printf("    %s: ", pt->var_name);
      scanf("%f%*c", &pt->value);
      i++;
      pt->next = (struct variables *) malloc(sizeof(va));
      temp = pt;
      pt = pt->next;
      }
    temp->next = NULL;
    }
}

/*
**
---------------------
  get_next_position
---------------------
**
** Get next position.
**
*/
int get_next_position(void)
{
nd_pos++;
if(nd_pos < MAX_ITEM)  return nd_pos;
else return -1;
}

/*
**
--------------
  save_ndata
--------------
**
** Save data.
**
*/
void save_ndata(void)
{
int i, j, k;
struct variables *temp;
FILE *fp;
char name[FILE_NAME_SIZE];

clear_screen();
goto_xy(0, 10);
printf("Enter datafile name: ");
gets(name);
strcat(name, ".dat");
if((fp = fopen(name,"w")) == 0)
   {
   printf("Cannot open file.\n");
   return;
   }
comment2("Saving data set...");
```

```
for(i=0; i<=nd_pos; ++i)
   {
   for(j=0; j<sizeof(ndata_base[i].index); j++)
      putc(ndata_base[i].index[j], fp);
   for(j=0; j<sizeof(ndata_base[i].description); j++)
      putc(ndata_base[i].description[j], fp);

   temp = ndata_base[i].var_value;
   while(temp)
      {
      for(j=0; j<sizeof(temp->var_name); j++) putc(temp->var_name[j], fp);
      fprintf(fp, "%f", temp->value);
      temp = temp->next;
      }
   for(j=0; j<sizeof(temp->var_name); j++) putc('\0', fp);
   }
putc(0, fp);
fclose(fp);
}

/*
**
--------------
 load_ndata
--------------
**
** Load database.
**
*/
void load_ndata(void)
{
struct variables *info, *temp;
FILE *fp;
char name[FILE_NAME_SIZE];
int i, j;

clear_screen();
clear_ndata();
goto_xy(0, 10);
printf("Enter datafile name: ");
gets(name);
fp = fopen(name, "r");
if(!fp)
   {
   printf("Cannot open file\n");
   return;
   }

comment2("Loading database...\n");

/* load database file */
clear_ndata();
for(i=0; i<MAX_ITEM; ++i)
   {
   if((ndata_base[i].index[0] = getc(fp)) == 0) break;
   for(j=1; j<sizeof(ndata_base[i].index); j++)
      ndata_base[i].index[j] = getc(fp);
   for(j=0; j<sizeof(ndata_base[i].index); j++)
      ndata_base[i].description[j] = getc(fp);
```

```
 ndata_base[i].var_value = (struct variables *) malloc(sizeof(va));
 info = ndata_base[i].var_value;
 if(!info)
    {
    printf("Out of memory\n");
    return;
    }
 for( ; ;)
    {
    for(j=0; j<sizeof(info->var_name); j++) info->var_name[j] = getc(fp);
    if(!info->var_name[0])
       {
       temp->next = NULL;
       break;
       }
     fscanf(fp, "%f", &info->value);
     info->next = (struct variables *) malloc(sizeof(va));
     if(!info->next) break;
     temp = info;
     info = info->next;
     }
  }
fclose(fp);
nd_pos = i-1;
}

/*
**
---------------
  clear_ndata
---------------
**
** Clear data stack.
**
*/
void clear_ndata(void)
{
struct variables *cur, *temp;
int i;

for(i=0; i<=nd_pos; i++)
   {
   cur = ndata_base[i].var_value;
   while(cur)
      {
      temp = cur->next;
      free(cur);
      cur = temp;
      }
   }
 }
```

```
/*
**
--------------------
  enter_event_data
--------------------
**
** Enter event data.
**
*/
int enter_event_data(void)
{
struct event_list *ev, *temp;
int i;

clear_screen();
goto_xy(0, 5);
printf("Enter event-list data\n\n");
ev = (struct event_list *) malloc(sizeof(struct event_list));
if(!ev)
   {
   printf("Out of memory\n");
   return 0;
   }
events = ev;
for(i=0; i<sizeof(ev->event_name); i++) ev->event_name[i] = ' ';
for(i=0; i<sizeof(ev->fact); i++) ev->fact[i] = ' ';
printf("Enter event title and event (RETURN to quit) \n");
for( ; ;)
   {
   clear_line(0, 10);
   clear_line(0, 12);
   goto_xy(0, 10);
   printf("Event title: ");
   gets(ev->event_name);
   goto_xy(0, 12);
   printf("Event: ");
   gets(ev->fact);
   if(!ev->event_name[0]) break;

   num_events++;
   temp = ev;
   ev->next = (struct event_list *) malloc(sizeof(struct event_list));
   if(ev->next == NULL)
      {
      printf("out of memory\n");
      return 0;
      }
   ev = ev->next;
   ev->next = NULL;
   for(i=0; i<sizeof(ev->event_name); i++) ev->event_name[i] = ' ';
   for(i=0; i<sizeof(ev->fact); i++) ev->fact[i] = ' ';
   }
temp->next = NULL;
}

/*
**
-----------------
  display_event
-----------------
```

```
**
** Display events.
**
*/
void display_event(void)
{
struct event_list *temp;
int i, j;

clear_screen();
goto_xy(0, 2);
temp = events;
j = 0;
while(temp)
   {
   j++;
   printf("EVENT %s: %s\n\n", temp->event_name, temp->fact);
   temp = temp->next;
   }
printf("\n(There are %d events in the list.)\n", j);
printf("\n\nPress ENTER to return\n");
getchar();
return;
}

/*
**
--------------
  save_event
--------------
**
** Save event file to a disk file.
**
*/
int save_event(char *file_name)
{
struct event_list *ev, *temp;
int i;
FILE *fp;
char name[FILE_NAME_SIZE];

/* save event data */
strcpy(name, file_name);
strcat(name, ".rul");
fp = fopen(name, "wb");
if(!fp)
   {
   printf("cannot open definition file\n");
   fclose(fp);
   return 0;
   }
ev = events;
while(ev)
   {
   for(i=0; i<sizeof(ev->event_name); i++) putc(ev->event_name[i], fp);
   for(i=0; i<sizeof(ev->fact); i++) putc(ev->fact[i], fp);
   ev = ev->next;
   }
putc(0, fp);
fclose(fp);
```

```
return 1;
}

/*
**
--------------
  load_event
--------------
**
** Load event file.
**
*/
int load_event(char *file_name)
{
struct event_list *info, *temp;
char name[FILE_NAME_SIZE];
FILE *fp;
int i;

strcpy(name, file_name);
strcat(name, ".rul");
fp = fopen(name, "rb");
if(!fp)
   {
   printf("Cannot open file\n");
   fclose(fp);
   return 0;
   }

events = (struct event_list *) malloc(sizeof(struct event_list));
info = events;
if(!info)
   {
   printf("Out of memory\n");
   return 0;
   }
for( ; ;)
   {
   for(i=0; i<sizeof(info->event_name); i++) info->event_name[i] = getc(fp);
   for(i=0; i<sizeof(info->fact); i++) info->fact[i] = getc(fp);
   if(!info->event_name[0])
      {
      temp->next = NULL;
      break;
      }
   info->next = (struct event_list *) malloc(sizeof(struct event_list));
   if(info->next == NULL)
      {
      printf("Out of memory\n");
      break;
      }
   temp = info;
   info = info->next;
   }
fclose(fp);
return 0;
}

/*
**
```

```
------------
 make_rule
------------
**
** Make rule.
**
*/
void make_rule(char *object)
{
struct record_list *cur;
struct event_list *temp;
char ind[MAX_NAME], ch[MAX_NAME];
int i;

cur = first_record;
while(cur)
  {
  if(!strcmp(object, cur->data+items[0].item_position))
    {
    clear_screen();
    goto_xy(0, 2);
    temp = events;
    while(temp)
      {
      strcpy(ch, temp->event_name);
      if(!strcmp(ch, cur->data+items[num_items-1].item_position))
        {
        printf("\n\n\nAccording to RULE %s : ",
              cur->data+items[0].item_position);
        printf("\n\n\n\n\nIf \n");
        for(i=1; i<num_items-1; i++)
          {
          strcpy(ind, items[i].item_title);
          strcpy(ch, cur->data+items[i].item_position);
          printf("       %s is %s", ind, ch);
          if(i < num_items-2) printf(" and\n");
          }
        printf("\n");
        printf("\n\nThen \n");
        printf("       %s\n", temp->fact);
        getchar();
        return;
        }
      temp = temp->next;
      }
    }
  cur = cur->next;
  }
comment2("Insufficient events. Please check events database.");
return;
}

/*
**
---------------
 check_state
---------------
**
** Check states.
**
```

```
*/
void check_state(void)
{
struct record_list *temp;
struct tf_list *root, *cur, *pres;
char ch[MAX_NAME];
int i, j;

/* free state conditions */
for(i=0; i<num_items-2; i++)
  {
  strcpy(state[i].state_title, null_string);
  pres = state[i].condition;
  while(pres)
    {
    cur = pres->next;
    free(pres);
    pres = cur;
    }
  }

for(i=0; i<num_items-2; i++)
  {
  strcpy(state[i].state_title, items[i+1].item_title);

  /* allocate memory */
  cur = (struct tf_list *) malloc(sizeof(struct tf_list));
  if(cur == NULL)
    {
    printf("Out of memory\n");
    return;
    }
  state[i].condition = cur;
  strcpy(cur->thing, null_string);
  cur->next = NULL;

  temp = first_record;
  while(temp)
    {
    root = (struct tf_list *) malloc(sizeof(struct tf_list));
    if(root == NULL)
      {
      printf("Out of memory\n");
      return;
      }
    strcpy(ch, temp->data+items[i+1].item_position);
    strcpy(root->thing, ch);
    root->next = NULL;

    if(!cur) cur = append_tf_list(root, cur);
    else
      {
      pres = cur;
      j = 0;
      while(pres)
        {
        if(!strcmp(ch, pres->thing)) j = 1;
        pres = pres->next;
        }
      if(j == 0) cur = append_tf_list(root, cur);
```

```
        }
      temp = temp->next;
      }
    }
}

/*
**
---------------
  state_fault
---------------
**
**
*/
void state_fault(void)
{
char ch;

check_state();
for( ; ;)
   {
   state_infer();
   comment1("Do you want to try more situations? (Y/N) :");
   ch = getche();
   ch = tolower(ch);
   if(ch == 'n') return;
   }
}

/*
**
---------------
  event_fault
---------------
**
**
*/
void event_fault(void)
{
char ch;

check_state();
for( ; ;)
   {
   event_infer();
   comment1("Do you want to try more situations? (Y/N) :");
   ch = getche();
   ch = tolower(ch);
   if(ch == 'n') return;
   }
}

/*
**
---------------
  state_infer
---------------
**
**
*/
```

```
void state_infer(void)
{
struct record_list *cur;
struct tf_list *temp, *ind, *pres;
char r_name[MAX_NAME];
int i, j;

clear_screen();

/* free state conditions */
while(ind)
  {
  pres = ind->next;
  free(ind);
  ind = pres;
  }

goto_xy(0, 4);
printf("Enter states: \n\n");

pres = (struct tf_list *) malloc(sizeof(struct tf_list));
if(pres == NULL)
  {
  printf("Out of memory\n");
  return;
  }
strcpy(pres->thing, null_string);
pres->next = NULL;
ind = pres;

for(i=0; i<num_items-2; i++)
  {
  printf("State of %s ( ", state[i].state_title);
  temp = state[i].condition;
  while(temp)
    {
    printf("%s ", temp->thing);
    temp = temp->next;
    }
  printf("): ");
  gets(pres->thing);

  pres->next = (struct tf_list *) malloc(sizeof(struct tf_list));
  if(pres->next == NULL)
    {
    printf("Out of memory\n");
    return;
    }
  pres = pres->next;
  pres->next = NULL;
  strcpy(pres->thing, null_string);

  printf("\n\n");
  }

/* forward inference */
cur = first_record;
while(cur)
  {
  j = 1;
```

```
 pres = ind;
 for(i=0; i<num_items-2; i++)
    {
    if(strcmp(pres->thing, cur->data+items[i+1].item_position)) j = 0;
    pres = pres->next;
    }
 if(j == 1)
    {
    strcpy(r_name, cur->data+items[0].item_position);
    make_rule(r_name);
    return;
    }
  cur = cur->next;
  }
if(!cur)
  {
  clear_screen();
  goto_xy(0, 4);
  printf("No matching rule. Check event list.\n");
  getchar();
  return;
  }
}

/*
**
---------------
  event_infer
---------------
**
**
*/
void event_infer(void)
{
struct record_list *cur, *new;
struct event_list *temp;
char ch[3], ind[MAX_NAME], cond[MAX_NAME], accid[MAX_STR];
int i, j;

clear_screen();
goto_xy(0,0);
printf("EVENT LIST :\n\n");
temp = events;
j = 0;
while(temp)
   {
   j++;
   printf("EVENT %s: %s\n", temp->event_name, temp->fact);
   temp = temp->next;
   }
printf("\n\n(There are %d events in the list.)\n", j);
printf("\n\nEnter event title (e1, e2, ...): ");
do
   {
   gets(ch);
   temp = events;
   while(temp)
      {
      if(strcmp(ch, temp->event_name))
         {
```

```
        j = 0;
        temp = temp->next;
        }
      else
        {
        j = 1;
        strcpy(accid, temp->fact);
        }
      if(j == 1) break;
      }
    if(j == 0) comment1("Wrong name. Enter again.");
    } while(j != 1);

  clear_screen();
  goto_xy(0, 0);
  cur = first_record;
  while(cur)
    {
    if(!strcmp(ch, cur->data+items[num_items-1].item_position))
      {
      goto_xy(0, 3);
      printf("According to RULE %s : ",
            cur->data+items[0].item_position);
      printf("\n\n\n\n");
      printf("%s because\n\n", accid);
      for(i=1; i<num_items-1; i++)
        {
        strcpy(ind, items[i].item_title);
        strcpy(cond, cur->data+items[i].item_position);
        printf("       %s is %s", ind, cond);
        if(i < num_items-2) printf(" and\n");
        }
      printf("\n\n\n\n\n\n\n");
      printf("Press any key for more possible cases.\n\n");
      getchar();
      clear_screen();
      }
    cur = cur->next;
    }
  printf("\nNo more possible cases. Enter RETURN.\n");
  getchar();
  }

  /*
  **
  ------------------
    append_tf_list
  ------------------
  **
  ** Append list to tf_list structure.
  **
  */
  struct tf_list *append_tf_list(struct tf_list *new, struct tf_list *list)
  {
  if(list == NULL)
    {
    list = new;
    return list;
    }
  else
```

```
    {
    list->next = append_tf_list(new, list->next);
    return list;
    }
}
```

```
/*
**
------------------------
  init_base_description
------------------------
**
** Initialize rule-base description.
**
*/
void init_base_description(void)
{
char *ob_title = "OBJECT";
char *rl_title = "RULE NAME";
char *cond     = "IF";
char *then     = "THEN";

clear_screen();
ob_pos = -1;

strcpy(field_ob.object_title, ob_title);
strcpy(field_rl.rule_title, rl_title);
strcpy(field_if.if_title, cond);
strcpy(field_th.then_title, then);

field_ob.x = 0;
field_ob.y = 4;

field_rl.x = field_ob.x;
field_rl.y = field_ob.y + 5;

field_if.x = field_ob.x;
field_if.y = field_rl.y + 3;

field_th.x = field_ob.x;
field_th.y = field_if.y + 2;
}

/*
**
-------------------
  enter_rule_data
-------------------
**
** Enter object and corresponding rule data.
**
*/
void enter_rule_data(void)
{
struct record *info,*temp;
char ch;
int i, j, m;

clear_screen();
display_object_title();
display_rule_title();

for( ; ;)
   {
   i = get_object_position(); /* get next available object position */
   if(i == -1)
```

```
  {
  printf("Out of object set space.\n");
  return;
  }

goto_xy(field_ob.x+strlen(field_ob.object_title)+5, field_ob.y);
for(j=0; j<MAX_NAME; j++) printf(" ");
goto_xy(field_ob.x+strlen(field_ob.object_title)+5, field_ob.y);
get_data(r_base[i].object_name, MAX_NAME);
if(!*r_base[i].object_name)
  {
  ob_pos--;
  break;
  }

rl_pos = -1;
for( ; ;)
  {
  m = get_rule_position(); /* get next available rule position */
  if(m == -1)
    {
    printf("Out of rule-list space.\n");
    return;
    }

  goto_xy(field_rl.x+strlen(field_rl.rule_title)+5, field_rl.y);
  for(j=0; j<MAX_NAME; j++) printf(" ");
  goto_xy(field_rl.x+strlen(field_rl.rule_title)+5, field_rl.y);
  get_data(r_base[i].rule_list[m].rule_name, MAX_NAME);

  if(!*r_base[i].rule_list[m].rule_name)
    {
    rl_pos--;
    break;
    }

  /* enter rule-list information */
  /* allocate memory for if-list */
  info = (struct record *) malloc(sizeof(rd));
  if(info == NULL)
    {
    printf("Out of memory.\n");
    return;
    }
  r_base[i].rule_list[m].if_list = info;

  /* enter if- information */
  for(j=0; j<sizeof(info->property); j++) info->property[j]=' ';
  comment1("Enter if-list property (RETURN to quit): ");
  for( ; ;)
    {
    goto_xy(field_if.x+strlen(field_if.if_title)+5, field_if.y);
    for(j=0; j<MAX_LLEN; j++) printf(" ");
    goto_xy(field_if.x+strlen(field_if.if_title)+5, field_if.y);
    get_data(info->property, MAX_LLEN);
    if(!info->property[0]) break;

    temp = info;
    info->next = (struct record *) malloc(sizeof(rd));
    if(info->next == NULL)
```

```
            {
            printf("Out of memory.\n");
            return;
            }
          info = info->next;
          info->next = NULL;
          for(j=0; j<sizeof(info->property); j++) info->property[j]=' ';
          }
        temp->next = NULL;

        /* allocate memory for then-list */
        info = (struct record *) malloc(sizeof(rd));
        if(info == NULL)
          {
          printf("Out of memory.\n");
          return;
          }
        r_base[i].rule_list[m].then_list = info;

        /* enter then- information */
        for(j=0; j<sizeof(info->property); j++) info->property[j]=' ';
        comment1("Enter then-list property (RETURN to quit): ");
        for( ; ;)
          {
          goto_xy(field_th.x+strlen(field_th.then_title)+5, field_th.y);
          for(j=0; j<MAX_LLEN; j++) printf(" ");
          goto_xy(field_th.x+strlen(field_th.then_title)+5, field_th.y);
          get_data(info->property, MAX_LLEN);
          if(!info->property[0]) break;

          temp = info;
          info->next = (struct record *) malloc(sizeof(rd));
          if(info->next == NULL)
            {
            printf("Out of memory.\n");
            return;
            }
          info = info->next;
          info->next = NULL;
          for(j=0; j<sizeof(info->property); j++) info->property[j]=' ';
          }
        temp->next = NULL;
        }
      r_base[i].num_of_rules = m+1;
      comment1("More objects? (Y/N): ");
      ch = getche();
      ch = tolower(ch);
      if(ch == 'n') return;
      }
}

/*
**
------------------------
  display_object_title
------------------------
**
** Display object title.
**
*/
```

```
void display_object_title(void)
{
goto_xy(field_ob.x, field_ob.y);
printf(" %s: ", field_ob.object_title);
}

/*
**
----------------------
  display_rule_title
----------------------
**
** Display rule title.
**
*/
void display_rule_title(void)
{
goto_xy(field_rl.x, field_rl.y);
printf(" %s: ", field_rl.rule_title);
goto_xy(field_if.x, field_if.y);
printf(" %s: ", field_if.if_title);
goto_xy(field_th.x, field_th.y);
printf(" %s: ", field_th.then_title);
}

/*
**
--------------------
  display_rule_abs
--------------------
**
** Display rule abstract
**
*/
void display_rule_abs(void)
{
struct record *info;
int i, j;

clear_screen();
display_object_title();
display_rule_title();
for(i=0; i<=ob_pos; i++)
   {
   goto_xy(field_ob.x+strlen(field_ob.object_title)+5, field_ob.y);
   for(j=0; j<MAX_NAME; j++) printf(" ");
   goto_xy(field_ob.x+strlen(field_ob.object_title)+5, field_ob.y);
   printf("%s", r_base[i].object_name);
   for(j=0; j<r_base[i].num_of_rules; j++)
      {
      clear_rule();
      goto_xy(field_rl.x+strlen(field_rl.rule_title)+5, field_rl.y);
      printf("%s", r_base[i].rule_list[j].rule_name);
      info = r_base[i].rule_list[j].if_list;
      goto_xy(field_if.x+strlen(field_if.if_title)+5, field_if.y);
      printf("%s", info->property);
      info = r_base[i].rule_list[j].then_list;
      goto_xy(field_th.x+strlen(field_th.then_title)+5, field_th.y);
      printf("%s", info->property);
      getchar();
```

```
    }
  getchar();
  }
}

/*
**
--------------
 clear_rule
--------------
**
**
*/
void clear_rule(void)
{
int j;

goto_xy(field_rl.x+strlen(field_rl.rule_title)+5, field_rl.y);
for(j=0; j<MAX_NAME; j++) printf(" ");
goto_xy(field_if.x+strlen(field_if.if_title)+5, field_if.y);
for(j=0; j<MAX_LLEN; j++) printf(" ");
goto_xy(field_th.x+strlen(field_th.then_title)+5, field_th.y);
for(j=0; j<MAX_LLEN; j++) printf(" ");
}

/*
**
--------------
 free_stack
--------------
**
** Free true and false stacks.
**
*/
void free_stack(void)
{
struct record *info;

/* free true stack */
while(true_stack)
  {
  info = true_stack->next;
  free(true_stack);
  true_stack = info;
  }

/* free false stack */
while(false_stack)
  {
  info = false_stack->next;
  free(false_stack);
  false_stack = info;
  }
ps_pos = -1;
}

/*
**
-----------------------
 get_object_position
```

```
----------------------
**
** Get next available object position.
**
*/
int get_object_position(void)
{
ob_pos++;
if(ob_pos < MAX_OBJ) return ob_pos;
else return -1;
}

/*
**
---------------------
  get_rule_position
---------------------
**
** Get next available rule_list position.
**
*/
int get_rule_position(void)
{
rl_pos++;
if(rl_pos < MAX_ITEM) return rl_pos;
else return -1;
}

/*
**
--------------
  infer_rule
--------------
**
** Forward-chain inference
**
*/
void infer_rule(void)
{
struct rule *info;
struct record *temp;
int i, j;
char ch;
char rule_ind[MAX_NAME];

for(i=0; i<=ob_pos; i++)
   {
   ps_pos = -1;
   true_stack = false_stack = NULL;

   clear_screen();
   goto_xy(0, 5);
   for(j=0; j<=r_base[i].num_of_rules; j++)
      {
      clear_screen();
      info = &r_base[i].rule_list[j];
      strcpy(rule_ind, info->rule_name);
      if(infer(info, rule_ind))
         {
         printf("\n\nAccording to the rule %s: ", rule_ind);
```

```
        temp = info->then_list;
        while(temp)
          {
          printf("\n\n      %s ", temp->property);
          if(temp->next) printf("and ");
          temp = temp->next;
          }
        printf(".");

        printf("\n\n\nContinue? (Y/N):  ");
        ch = getche();
        ch = tolower(ch);
        printf("\n");
        if(ch == 'n') return;
        else break;
        }
      }
    }
comment2("No (more) object(s) found.\n");
}

/*
**
---------
  infer
---------
**
** Examine rule list and true and false stacks.
**
*/
int infer(struct rule *info, char *object)
{
struct record *curr, *temp, *pres;
char ind;
int i;

pres = info->if_list;
/* examine false stack */
if(!examine_false_stack(pres, object)) return 0;

/* examine true stack */
if(!examine_true_stack(pres, object)) return 0;

/* examine rule list */
i = 0;
while(pres)
  {
  if(check_stack(pres->property))  /* property not in true stack */
    {
    i++;
    printf("\n\nQuestion [%d]: %s.", i, pres->property);
    printf("\n\n              Is it true? ((T)rue, (F)alse, (W)hy):  ");
    ind = getche();
    ind = tolower(ind);
    printf("\n");
    curr = (struct record *) malloc(sizeof(rd));
    if(!curr)
      {
      printf("Out of memory.\n");
      return;
```

```
      }

    curr->next = NULL;
    switch(ind)
      {
      case 'f':
      /* False property. Store it into false_stack and try next rule. */
        strcpy(curr->property, pres->property);
        if(!false_stack)
          {
          false_stack = curr;
          false_temp = false_stack;
          }
        else
          {
          false_temp->next = curr;
          false_temp = curr;
          }
        store_prop_stack(object, pres->property,'f');
        return 0;
      case 't':
      /* True property. Store it into true_stack and try next property. */
        strcpy(curr->property, pres->property);
        if(!true_stack)
          {
          true_stack = curr;
          true_temp = true_stack;
          }
        else
          {
          true_temp->next = curr;
          true_temp = curr;
          }
        pres = pres->next;
        break;
      case 'w':
        explain_why(object);
        i--;
        break;
      }
    }
  else
    pres = pres->next;
  }
return 1;  /* All properties in if-list satisfy condition. */
}

/*
**
-----------------------
 examine_false_stack
-----------------------
**
** Examine false stack.
**
*/
int examine_false_stack(struct record *info, char *object)
{
struct record *stack, *temp;
```

```
stack = false_stack;
while(stack)
   {
   temp = info;
   while(temp)
      {
      if(!strcmp(temp->property, stack->property))
         {
         store_prop_stack(object, temp->property, 'f');
         return 0;
         }
      temp = temp->next;
      }
   stack = stack->next;
   }
return 1;
}

/*
**
----------------------
  examine_true_stack
----------------------
**
** Examine true stack.
**
*/
int examine_true_stack(struct record *info, char *object)
{
struct record *stack, *temp;
char ind;

stack = true_stack;
while(stack)
   {
   ind = 0;
   temp = info;
   while(temp)
      {
      if(!strcmp(temp->property, stack->property))
         {
         ind = 1;
         }
      temp = temp->next;
      }
   if(!ind)
   /* If-list doesn't have required true property. Try next rule. */
      {
      store_prop_stack(object, stack->property, 't');
      return 0;
      }
   stack = stack->next;
   }
return 1;  /* need more check */
}

/*
**
---------------
  check_stack
```

```
---------------
**
** Check stack.
**
*/
int check_stack(char *property)
{
struct record *info;

/* check true property stack */
info = true_stack;
while(info && strcmp(property, info->property)) info = info->next;

if(!info) return 1; /* Property not in true_stack. */
else return 0;      /* Property in true_stack. Check next property. */
}

/*
**
--------------------
  store_prop_stack
--------------------
**
** Examine true stack.
**
*/
void store_prop_stack(char *object, char *prop, char ind)
{
ps_pos++;

strcpy(prop_stack[ps_pos].item, object);
strcpy(prop_stack[ps_pos].property, prop);
prop_stack[ps_pos].true_or_false = ind;
}

/*
**
---------------
  explain_why
---------------
**
** Explain why.
**
*/
void explain_why(char *object)
{
struct record *temp;
int i;

clear_screen();
goto_xy(0, 4);
printf("Trying rule %s.\n\n", object);

if(true_stack) printf("It is true that: \n");
temp = true_stack;
while(temp)
   {
   printf("                %s\n", temp->property);
   temp = temp->next;
   }
```

```
printf("\n\n");
if(false_stack) printf("It is not: \n");
temp = false_stack;
while(temp)
  {
  printf("          %s\n",temp->property);
  temp = temp->next;
  }

for(i=0; i<=ps_pos; i++)
  {
  printf("\n\nRule %s was rejected because ", prop_stack[i].item);
  if(prop_stack[i].true_or_false == 'f')
    printf("%s is not a property.\n", prop_stack[i].property);
  else
    printf("%s is a required property.\n", prop_stack[i].property);
  }
}

/*
**
---------------
 save_r_base
---------------
**
** Save rule-list base.
**
*/
void save_r_base(char *file_name)
{
struct rule *cur;
struct record *temp;
FILE *fp;
char name[FILE_NAME_SIZE];
int i, j, k;

/* store the rule data in a .RUL file */
strcpy(name, file_name);
strcat(name, ".rul");
fp = fopen(name, "w");
if(!fp)
  {
  printf("Cannot open file\n");
  return;
  }

comment2("Saving rule-list base... \n");

for(i=0; i<=ob_pos; ++i)
  {
  for(j=0; j<sizeof(r_base[i].object_name); j++)
    putc(r_base[i].object_name[j], fp);
  for(k=0; k<r_base[i].num_of_rules; k++)
    {
    cur = &r_base[i].rule_list[k];
    for(j=0; j<sizeof(cur->rule_name); j++) putc(cur->rule_name[j], fp);

    /* if-list are stored in *.if file */
    temp = cur->if_list;
```

```
    while(temp)
      {
      for(j=0; j<sizeof(temp->property); j++)
        putc(temp->property[j], fp);
      temp = temp->next;
      }
    for(j=0; j<sizeof(temp->property); j++) putc('\0', fp);

    /* then-list are stored in *.thn file */
    temp = cur->then_list;
    while(temp)
      {
      for(j=0; j<sizeof(temp->property); j++)
        putc(temp->property[j], fp);
      temp = temp->next;
      }
    for(j=0; j<sizeof(temp->property); j++) putc('\0', fp);
    }
  putc('$', fp);
  fprintf(fp, "%d", r_base[i].num_of_rules);
  }
putc('@', fp);
fclose(fp);
}

/*
**
---------------
 load_r_base
---------------
**
** Load rule-list base.
**
*/
void load_r_base(char *file_name)
{
struct record *info, *temp;
FILE *fp;
char name[FILE_NAME_SIZE];
int i, j, k;

strcpy(name, file_name);
strcat(name, ".rul");
fp = fopen(name, "r");
if(!fp)
  {
  printf("Cannot open file\n");
  return;
  }

comment2("Loading rule-list base...\n");

/* load data file */
clear_r_base();
for(i=0; i<MAX_OBJ; ++i)
  {
  if((r_base[i].object_name[0] = getc(fp)) == '@') break;
  for(j=1; j<sizeof(r_base[i].object_name); j++)
    r_base[i].object_name[j] = getc(fp);
```

```
for(k=0; k<MAX_ITEM; ++k)
  {
  if((r_base[i].rule_list[k].rule_name[0] = getc(fp)) == '$') break;
  for(j=1; j<sizeof(r_base[i].rule_list[k].rule_name); j++)
    r_base[i].rule_list[k].rule_name[j] = getc(fp);

  /* load if-list */
  r_base[i].rule_list[k].if_list = (struct record *) malloc(sizeof(rd));
  info = r_base[i].rule_list[k].if_list;
  if(!info)
    {
    printf("Out of memory\n");
    return;
    }
  for( ; ;)
    {
    for(j=0; j<sizeof(info->property); j++)
      info->property[j] = getc(fp);
    if(!info->property[0])
      {
      temp->next = '\0';
      break;
      }
    info->next = (struct record *) malloc(sizeof(rd));
    if(!info->next)
      {
      printf("Out of memory\n");
      break;
      }
    temp = info;
    info = info->next;
    }

  /* load then-list */
  r_base[i].rule_list[k].then_list = (struct record *) malloc(sizeof(rd));
  info = r_base[i].rule_list[k].then_list;
  if(!info)
    {
    printf("Out of memory\n");
    return;
    }
  for( ; ;)
    {
    for(j=0; j<sizeof(info->property); j++)
      info->property[j] = getc(fp);
    if(!info->property[0])
      {
      temp->next = '\0';
      break;
      }
    info->next = (struct record *) malloc(sizeof(rd));
    if(!info->next)
      {
      printf("Out of memory\n");
      break;
      }
    temp = info;
    info = info->next;
    }
  }
```

```
  fscanf(fp, "%d", &r_base[i].num_of_rules);
  }
fclose(fp);
ob_pos = i-1;
}

/*
**
----------------
 clear_r_base
----------------
**
** Clear rule-list base before loading.
**
*/
void clear_r_base(void)
{
struct rule *cur;
struct record *info, *temp;
int i, j;

for(i=0; i<=ob_pos; i++)
  {
  for(j=0; j<r_base[i].num_of_rules; j++)
    {
    cur = &r_base[i].rule_list[j];

    info = cur->if_list;
    while(info)
      {
      temp = info->next;
      free(info);
      info = temp;
      }

    info = cur->then_list;
    while(info)
      {
      temp = info->next;
      free(info);
      info = temp;
      }
    }
  }
}
```

2. Model Based Fault Diagnosis 프로그램

```
$STORAGE:2
$NOFLOATCALLS
C       LIQUID PIPELINE MODEL USING THE METHOD OF CHARACTERISTICS
        PROGRAM SIMGLEAK
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        DIMENSION X(36),F(18)
        COMMON /SPEC/ B(18),R(18),HMS(2,2),QLEAK,JLEAK
        COMMON /SQNC/ IPM(12),NPRBS,PPRBS,PU
        DATA GC,PAI/9.80?,3.1416/

        OPEN(1,FILE='SIMLEAK.DAT')
        OPEN(2,FILE='SIMLEAK.REC',STATUS='UNKNOWN')
        OPEN(3,FILE='SIMH.REC',STATUS='UNKNOWN')
        READ(1,100) NOS,NPRBS,PPRBS,PTIME,PLENGTH,PDIAM,HININ,HEXIN,
     &              STIME,VINIT,TWAVE,ETIME,ZLEAK,PLEAK,TLEAK,RSCALE
100     FORMAT(2(10x,I5/),14(10x,D15.5/))

        AF=PAI*PDIAM**2/4.0
        JLEAK=ZLEAK*NOS/PLENGTH
        QINIT=AF*VINIT
        WAVESP=PLENGTH/TWAVE
        RAMDOM=1.2345678D+09
        DO 10 I=1,12
        IPM(I)=1
10      CONTINUE
        TIME=0.0

        DO 20 I=1,NOS+1
        X(I)=QINIT
20      CONTINUE
        DO 30 I=NOS+2,2*NOS
        X(I)=HININ-(HININ-HEXIN)/NOS*(I-NOS-1)
30      CONTINUE
        CALL    MEASURE(HMS,HININ,HEXIN,TIME,PTIME,X,NOS)
        F(1)=2.0*GC*PDIAM*NOS*AF**2*(HMS(1,1)-X(NOS+2))/
     &      (PLENGTH*X(1)**2)
        DO 40 I=2,NOS-1
        F(I)=2.0*GC*PDIAM*NOS*AF**2*(X(NOS+I)-X(NOS+I+1))/
     &       (PLENGTH*X(I)**2)
40      CONTINUE
        F(NOS)=2.0*GC*PDIAM*NOS*AF**2*(X(2*NOS)-HMS(2,1))/
     &         (PLENGTH*X(NOS)**2)

        DO 50 I=1,NOS
        B(I)=WAVESP/GC/AF
        R(I)=F(I)*PLENGTH/(2.0*GC*PDIAM*AF**2*NOS)
50      CONTINUE

200     CALL    MEASURE(HMS,HININ,HEXIN,TIME,PTIME,X,NOS)

        IF(TIME.GE.TLEAK) THEN
        QLEAK=X(JLEAK)*PLEAK/100.0
        X(JLEAK)=X(JLEAK)-QLEAK
        ENDIF

        CALL    PROCESS(X,NOS)
        CALL    OUTPRT(TIME,X,HMS,NOS)

        TIME=TIME+STIME
        IF(TIME.LT.ETIME) GO TO 200
        CLOSE(2)
        STOP
        END
```

```
      SUBROUTINE PROCESS(X,NOS)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(36),XS(36),XN(36),AM(36,36),BM(36,36),U(36)
      COMMON /SPEC/ B(18),R(18),HMS(2,2),QLEAK,JLEAK

      DO 10 I=1,NOS*2
      XS(I)=X(I)**2
      XN(I)=0.0
      U(I)=0.0
      DO 10 J=1,NOS*2
      AM(I,J)=0.0
      BM(I,J)=0.0
10    CONTINUE

      AM(1,2)=1.0
      DO 20 I=2,NOS
      AM(I,I-1)=B(I-1)/(B(I-1)+B(I))
      AM(I,I+1)=B(I)/(B(I-1)+B(I))
20    CONTINUE
      AM(NOS+1,NOS)=1.0

      AM(1,NOS+2)=-1.0/B(1)
      AM(2,NOS+3)=-1.0/(B(1)+B(2))
      DO 30 I=3,NOS-1
      AM(I,I+NOS-1)=1.0/(B(I-1)+B(I))
      AM(I,I+NOS+1)=-AM(I,I+NOS-1)
30    CONTINUE
      AM(NOS,NOS*2-1)=1.0/(B(NOS-1)+B(NOS))
      AM(NOS+1,NOS*2)=1.0/B(NOS)

      DO 40 I=1,NOS-1
      AM(I+NOS+1,I)=B(I)*B(I+1)/(B(I)+B(I+1))
      AM(I+NOS+1,I+2)=-AM(I+NOS+1,I)
40    CONTINUE

      AM(NOS+2,NOS+3)=B(1)/(B(1)+B(2))
      DO 50 I=2,NOS-2
      AM(I+NOS+1,I+NOS)=B(I+1)/(B(I)+B(I+1))
      AM(I+NOS+1,I+NOS+2)=B(I)/(B(I)+B(I+1))
50    CONTINUE
      AM(NOS*2,NOS*2-1)=B(NOS)/(B(NOS-1)+B(NOS))

      BM(1,2)=-R(1)/B(1)
      DO 60 I=2,NOS
      BM(I,I-1)=-R(I-1)/(B(I-1)+B(I))
      BM(I,I+1)=-R(I)/(B(I-1)+B(I))
60    CONTINUE
      BM(NOS+1,NOS)=-R(NOS-1)/B(NOS-1)

      DO 70 I=1,NOS-1
      BM(I+NOS+1,I)=-B(I+1)*R(I)/(B(I)+B(I+1))
      BM(I+NOS+1,I+2)=B(I)*R(I+1)/(B(I)+B(I+1))
70    CONTINUE

      U(1)=HMS(1,1)/B(1)
      U(2)=HMS(1,2)/(B(1)+B(2))
      U(NOS)=-HMS(2,2)/(B(NOS-1)+B(NOS))
      U(NOS+1)=-HMS(2,1)/B(NOS)
      U(NOS+2)=HMS(1,2)*B(2)/(B(1)+B(2))
      U(NOS*2)=HMS(2,2)*B(NOS-1)/(B(NOS-1)+B(NOS))

      DO 80 I=1,NOS*2
      DO 90 J=1,NOS*2
      XN(I)=XN(I)+AM(I,J)*X(J)+BM(I,J)*XS(J)
90    CONTINUE
      XN(I)=XN(I)+U(I)
```

```
80      CONTINUE
        X(JLEAK)=X(JLEAK)+QLEAK
        XS(JLEAK)=X(JLEAK)**2
        XN(JLEAK-1)=0.0
        XN(JLEAK+NOS-1)=0.0
        DO 100 J=1,NOS*2
        XN(JLEAK-1)=XN(JLEAK-1)+AM(JLEAK-1,J)*X(J)+BM(JLEAK-1,J)*XS(J)
        XN(JLEAK+NOS-1)=XN(JLEAK+NOS-1)+AM(JLEAK+NOS-1,J)*X(J)+
     &                  BM(JLEAK+NOS-1,J)*XS(J)
100     CONTINUE
        DO 110 I=1,NOS*2
        X(I)=XN(I)
110     CONTINUE
        RETURN
        END

        SUBROUTINE MEASURE(HMS,HININ,HEXIN,TIME,PTIME,X,NOS)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        DIMENSION HMS(2,2),X(36)
        COMMON /SQNC/ IPM(12),NPRBS,PPRBS,PU
C
C       HMS(1,1)=NEW Hin
C       HMS(1,2)=OLD Hin
C       HMS(2,1)=NEW Hex
C       HMS(2,2)=OLD Hex
C
        RESIDUE=DMOD(TIME,PTIME)
        IF(RESIDUE.LT.1.D-8) CALL     PRBS(NPRBS,IPM,PU)
        IF(TIME.EQ.0.0) THEN
                PU=0.0
                HMS(1,1)=HININ
                HMS(2,1)=HEXIN
                HMS(1,2)=HMS(1,1)
                HMS(2,2)=HMS(2,1)
        ELSE
                HINOLD=HMS(1,1)
                HEXOLD=HMS(2,1)
                HMS(1,1)=X(NOS+2)+(HININ-HEXIN)/NOS
C                HMS(1,1)=HININ*(1.0+PU*PPRBS/100.0)
                HMS(2,1)=X(NOS*2)-(HININ-HEXIN)/NOS
                HMS(1,2)=HINOLD
                HMS(2,2)=HEXOLD
        ENDIF
        RETURN
        END

        SUBROUTINE OUTPRT(TIME,X,HMS,NOS)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        DIMENSION X(36),HMS(2,2)

        WRITE(*,'('' TIME='',D12.4)') TIME
        WRITE(2,'('' TIME='',D12.4)') TIME
        WRITE(*,100) (X(I),I=1,NOS+1)
        WRITE(2,100) (X(I),I=1,NOS+1)
100     FORMAT(2x,'Q:',19D12.5)
        WRITE(*,200) (X(I),I=NOS+2,NOS*2)
        WRITE(2,200) (X(I),I=NOS+2,NOS*2)
200     FORMAT(2x,'H:',17D12.5/)
        WRITE(3,300) TIME,X(1),X(NOS+1),X(NOS+2),X(NOS*2),HMS(1,1)
300     FORMAT(5x,6D12.5)

        RETURN
        END

        SUBROUTINE PRBS(NDIM,IPM,PU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      DIMENSION IPM(12)

      GO TO(5,6,7,8,9,10,11),NDIM-4
5     IRAN=MOD(IPM(NDIM)+IPM(3),2)
      GO TO 100
6     IRAN=MOD(IPM(NDIM)+IPM(5),2)
      GO TO 100
7     IRAN=MOD(IPM(NDIM)+IPM(4),2)
      GO TO 100
8     IRAN=MOD(MOD(MOD(IPM(NDIM)+IPM(4),2)+IPM(3),2)+IPM(2),2)
      GO TO 100
9     IRAN=MOD(IPM(NDIM)+IPM(5),2)
      GO TO 100
10    IRAN=MOD(IPM(NDIM)+IPM(7),2)
      GO TO 100
11    IRAN=MOD(IPM(NDIM)+IPM(9),2)
100   DO 200 I=2,NDIM
200   IPM(NDIM-I+2)=IPM(NDIM-I+1)
      IPM(1)=IRAN
      if(IRAN.EQ.0) IRAN=-1
      PU=FLOAT(IRAN)
      RETURN
      END

      SUBROUTINE RNOISE(D,SCALE,V)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)

      A=2147483647.
      B=2147483648.
      D=DMOD(16807.*D,A)
      V=SCALE*(2.*D/B-1.)
      RETURN
      END
```

```
NOS      :   18                  (NUMBER OF SECTIONS)
NPRBS    :   10                  (PRBS DIMENSION)
PPRBS    :    5.00000D+00        (MAGNITUTE OF PRBS)
PTIME    :    5.00000D-02        (PRBS INTERVAL)
PLENGTH  :    1.20000D+02        (PIPE LENGTH, m)
PDIAM    :    1.00000D-02        (PIPE DIAMETER, m)
PIN      :    5.50000D+01        (INLET PIEZOMETRIC HEAD, m)
PEX      :    1.90000D+01        (OUTLET PIEZOMETRIC HEAD, m)
STIME    :    5.00000D-03        (SAMPLING TIME, sec)
VINIT    :    1.20000D+00        (FLUID VELOCITY, m/sec)
TWAVE    :    9.00000D-02        (TRAVEL TIME OF PRESSURE WAVE, sec)
ETIME    :    1.00000D+00        (SIMULATION END TIME, sec)
ZLEAK    :    4.00000D+01        (LOCATION OF LEAKING, m)
PLEAK    :    8.00000D-01        (LEAKAGE PERCENT)
TLEAK    :    3.00000D-01        (LEAKING START TIME, sec)
RSCALE   :    1.00000D-09        (NOISE SCALE)
1
```

```
$STORAGE:2
$NOFLOATCALLS
C------- LEAKAGE DETECTION USING CROSS-CORRELATION METHOD
         PROGRAM LEAK_DETECTION
         IMPLICIT DOUBLE PRECISION(A-H,O-Z)
         DIMENSION PM(2),QM(2),DQ(2,-50:300),PAI_QQ(-50:50,2),C(2,2)
         DIMENSION RQ2(2),DQ2(2),DQ2_SUM(2),QSTAR(2),X(36),F(18)
         COMMON /SPEC/ B(18),R(18),HMS(2,2),RANDNO(4),RSCALE(4)
         COMMON /LEAK/ TLEAK,QLEAK,PLEAK,JLEAK
         DATA GC,PAI/9.807,3.1416/

C        OPEN DATA AND RECORD FILES
         OPEN(1,FILE='DETLEAK.DAT',STATUS='OLD')
         OPEN(5,FILE='DETLEAKT.REC',STATUS='UNKNOWN')
         OPEN(6,FILE='DETLEAKZ.REC',STATUS='UNKNOWN')

C        READ INPUT DATA
         READ(1,500) NOS,IP,PLENGTH,PDIAM,HININ,HEXIN,VINIT,
     &   STIME,ETIME,TWAVE,TLEAK,ZLEAK,PLEAK,CAPPA,RAMBDA,
     &   PAI_BBD,PAI_INI,(RSCALE(I),I=1,4)
500      FORMAT(2(10x,I5/),19(10x,D15.5/))
         CLOSE(1)

C        CALCULATE FUNDAMENTAL DATA
         AF=PAI*PDIAM**2/4.0
         JLEAK=ZLEAK*NOS/PLENGTH
         QINIT=AF*VINIT
         WAVESP=PLENGTH/TWAVE

C        INITIALIZATION
         TIME=0.0
         ITER=0
         K=2
         QLEAK=0.0
         ZLE=0.0
         DO 10 I=1,NOS+1
         X(I)=QINIT
10       CONTINUE
         DO 20 I=NOS+2,2*NOS
         X(I)=HININ-(HININ-HEXIN)/NOS*(I-NOS-1)
20       CONTINUE
         DO 30 I=1,2
         QSTAR(I)=QINIT
30       CONTINUE
         HMS(1,1)=HININ
         HMS(2,1)=HEXIN
         DO 40 I=1,2
         C(I,1)=QSTAR(I)**2/(HININ-HEXIN)
40       CONTINUE
         DO 50 I=-IP,IP
         PAI_QQ(I,1)=PAI_INI
50       CONTINUE
         DO 60 I=1,2
         DQ2_SUM(I)=0.0
60       CONTINUE
         F(1)=2.0*GC*PDIAM*NOS*AF**2*(HININ-X(NOS+2))/(PLENGTH*X(1)**2)
         DO 70 I=2,NOS-1
         F(I)=2.0*GC*PDIAM*NOS*AF**2*(X(NOS+I)-X(NOS+I+1))/
     &        (PLENGTH*X(I)**2)
70       CONTINUE
         F(NOS)=2.0*GC*PDIAM*NOS*AF**2*(X(2*NOS)-HEXIN)/
     &         (PLENGTH*X(NOS)**2)
         DO 80 I=1,NOS
         B(I)=WAVESP/GC/AF
```

```
      R(I)=F(I)*PLENGTH/(2.0*GC*PDIAM*AF**2*NOS)
80    CONTINUE
      DO 90 I=1,4
      RANDNO(I)=I*1.2345678D+09
90    CONTINUE

200   ITER=ITER+1
      CALL    PROCESS(TIME,X,PM,QM,HININ,HEXIN,NOS,TWAVE)

C     CALCULATE THE CROSS-CORRELATION FUNCTION
      DO 100 I=1,2
      DQ(I,ITER)=QM(I)-QSTAR(I)
100   CONTINUE
      DO 110 ITAU=-IP,IP
      IF(ITAU.LE.0) THEN
              PAI_QQ(ITAU,K)=RAMBDA*PAI_QQ(ITAU,K-1)+
     &                (1.0-RAMBDA)*(DQ(1,ITER)*DQ(2,ITER+ITAU))
      ELSE
              PAI_QQ(ITAU,K)=RAMBDA*PAI_QQ(ITAU,K-1)+
     &                (1.0-RAMBDA)*(DQ(1,ITER-ITAU)*DQ(2,ITER))
      ENDIF
      PAI_QQ(ITAU,K-1)=PAI_QQ(ITAU,K)
110   CONTINUE

C     CALCULATE THE AVERAGE OF THE CROSS-CORRELATION FUNCTION
      PAI_SUM=0.0
      DO 120 I=-IP,IP
      PAI_SUM=PAI_SUM+PAI_QQ(I,K)
120   CONTINUE
      PAI_SUM=PAI_SUM/(2.0*IP+1)

C     CHECK THE THRESHOLD
      WRITE(5,'('' THRESHOLD:'',D12.4)') PAI_SUM
      IF(PAI_SUM.GT.PAI_BBD) THEN
C     ESTIMATE THE PRESSURE GRADIENTS
      DO 130 I=1,2
      C(I,K)=CAPPA*C(I,K-1)+(1.0-CAPPA)*QM(I)**2/(PM(1)-PM(2))
      C(I,K-1)=C(I,K)
130   CONTINUE
      ENDIF

C     DETERMINE THE REFERENCE MASS FLOWS
      DO 140 I=1,2
      RQ2(I)=C(I,K)*(PM(1)-PM(2))
140   CONTINUE

      IF(PAI_SUM.GT.PAI_BBD) GO TO 300
C     ESTIMATE THE LEAK LOCATION AND THE LEAK FLOW
      DO 150 I=1,2
      DQ2(I)=QM(I)**2-RQ2(I)
      DQ2_SUM(I)=DQ2_SUM(I)+DQ2(I)
150   CONTINUE

      ZLE=PLENGTH*(1.0-DQ2_SUM(1)/DQ2_SUM(2))
      WRITE(6,'('' TIME , ZLE :'',2D12.4)') TIME,ZLE
300   TIME=TIME+STIME
      IF(TIME.LT.ETIME) GO TO 200

      CLOSE(5)
      CLOSE(6)
      STOP
      END

C-----------------------------------------------------------------------
      SUBROUTINE PROCESS(TIME,X,PM,QM,HININ,HEXIN,NOS,TWAVE)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
```

```
      DIMENSION X(36),PM(2),QM(2)
      COMMON /SPEC/ B(18),R(18),HMS(2,2),RANDNO(4),RSCALE(4)
      COMMON /LEAK/ TLEAK,QLEAK,PLEAK,JLEAK

      CALL    MEASURE(HMS,HININ,HEXIN,X,NOS)

      IF(TIME.GE.TLEAK) THEN
      QLEAK=X(JLEAK)*PLEAK/100.0
      X(JLEAK)=X(JLEAK)-QLEAK
      ENDIF
      CALL    MODEL(X,NOS)

      DO 10 I=1,2
      CALL    RNOISE(RANDNO(I),RSCALE(I),W)
      PM(I)=HMS(I,1)+W
10    CONTINUE
      CALL    RNOISE(RANDNO(3),RSCALE(3),W)
      QM(1)=X(1)+W
      CALL    RNOISE(RANDNO(4),RSCALE(4),W)
      QM(2)=X(NOS+1)+W
      RETURN
      END

C-----------------------------------------------------------------------
      SUBROUTINE MODEL(X,NOS)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(36),XS(36),XN(36),AM(36,36),BM(36,36),U(36)
      COMMON /SPEC/ B(18),R(18),HMS(2,2),RANDNO(4),RSCALE(4)
      COMMON /LEAK/ TLEAK,QLEAK,PLEAK,JLEAK

      DO 10 I=1,NOS*2
      XS(I)=X(I)**2
      XN(I)=0.0
      U(I)=0.0
      DO 10 J=1,NOS*2
      AM(I,J)=0.0
      BM(I,J)=0.0
10    CONTINUE

      AM(1,2)=1.0
      DO 20 I=2,NOS
      AM(I,I-1)=B(I-1)/(B(I-1)+B(I))
      AM(I,I+1)=B(I)/(B(I-1)+B(I))
20    CONTINUE
      AM(NOS+1,NOS)=1.0

      AM(1,NOS+2)=-1.0/B(1)
      AM(2,NOS+3)=-1.0/(B(1)+B(2))
      DO 30 I=3,NOS-1
      AM(I,I+NOS-1)=1.0/(B(I-1)+B(I))
      AM(I,I+NOS+1)=-AM(I,I+NOS-1)
30    CONTINUE
      AM(NOS,NOS*2-1)=1.0/(B(NOS-1)+B(NOS))
      AM(NOS+1,NOS*2)=1.0/B(NOS)

      DO 40 I=1,NOS-1
      AM(I+NOS+1,I)=B(I)*B(I+1)/(B(I)+B(I+1))
      AM(I+NOS+1,I+2)=-AM(I+NOS+1,I)
40    CONTINUE

      AM(NOS+2,NOS+3)=B(1)/(B(1)+B(2))
      DO 50 I=2,NOS-2
      AM(I+NOS+1,I+NOS)=B(I+1)/(B(I)+B(I+1))
      AM(I+NOS+1,I+NOS+2)=B(I)/(B(I)+B(I+1))
50    CONTINUE
      AM(NOS*2,NOS*2-1)=B(NOS)/(B(NOS-1)+B(NOS))
```

```
      BM(1,2)=-R(1)/B(1)
      DO 60 I=2,NOS
      BM(I,I-1)=-R(I-1)/(B(I-1)+B(I))
      BM(I,I+1)=-R(I)/(B(I-1)+B(I))
60    CONTINUE
      BM(NOS+1,NOS)=-R(NOS-1)/B(NOS-1)

      DO 70 I=1,NOS-1
      BM(I+NOS+1,I)=-B(I+1)*R(I)/(B(I)+B(I+1))
      BM(I+NOS+1,I+2)=B(I)*R(I+1)/(B(I)+B(I+1))
70    CONTINUE

      U(1)=HMS(1,1)/B(1)
      U(2)=HMS(1,2)/(B(1)+B(2))
      U(NOS)=-HMS(2,2)/(B(NOS-1)+B(NOS))
      U(NOS+1)=-HMS(2,1)/B(NOS)
      U(NOS+2)=HMS(1,2)*B(2)/(B(1)+B(2))
      U(NOS*2)=HMS(2,2)*B(NOS-1)/(B(NOS-1)+B(NOS))

      DO 80 I=1,NOS*2
      DO 90 J=1,NOS*2
      XN(I)=XN(I)+AM(I,J)*X(J)+BM(I,J)*XS(J)
90    CONTINUE
      XN(I)=XN(I)+U(I)
80    CONTINUE
      X(JLEAK)=X(JLEAK)+QLEAK
      XS(JLEAK)=X(JLEAK)**2
      XN(JLEAK-1)=0.0
      XN(JLEAK+NOS-1)=0.0
      DO 100 J=1,NOS*2
      XN(JLEAK-1)=XN(JLEAK-1)+AM(JLEAK-1,J)*X(J)+BM(JLEAK-1,J)*XS(J)
      XN(JLEAK+NOS-1)=XN(JLEAK+NOS-1)+AM(JLEAK+NOS-1,J)*X(J)+
     &                BM(JLEAK+NOS-1,J)*XS(J)
100   CONTINUE
      DO 110 I=1,NOS*2
      X(I)=XN(I)
110   CONTINUE
      RETURN
      END

C-----------------------------------------------------------------------
      SUBROUTINE MEASURE(HMS,HININ,HEXIN,X,NOS)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION HMS(2,2),X(36)
C
C     HMS(1,1)=NEW Hin
C     HMS(1,2)=OLD Hin
C     HMS(2,1)=NEW Hex
C     HMS(2,2)=OLD Hex
C
      HINOLD=HMS(1,1)
      HEXOLD=HMS(2,1)
      HMS(1,1)=X(NOS+2)+(HININ-HEXIN)/NOS
      HMS(2,1)=X(NOS*2)-(HININ-HEXIN)/NOS
      HMS(1,2)=HINOLD
      HMS(2,2)=HEXOLD
      RETURN
      END

C-----------------------------------------------------------------------
      SUBROUTINE RNOISE(D,SCALE,V)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)

      A=2147483647.
      B=2147483648.
```

```
      D=DMOD(16807.*D,A)
      V=SCALE*(2.*D/B-1.)
      RETURN
      END
```

```
NOS         :    18                    (NUMBER OF SECTIONS)
IP          :    10                    (NUMBER OF TERMS FOR CROSS-CORRELATION(CC))
PLENGTH     :     1.20000D+02          (PIPE LENGTH, m)
PDIAM       :     1.00000D-02          (PIPE DIAMETER, m)
HININ       :     5.50000D+01          (INLET PIEZOMETRIC HEAD, m)
HEXIN       :     1.90000D+01          (OUTLET PIEZOMETRIC HEAD, m)
VINIT       :     1.20000D+00          (FLUID VELOCITY, m/sec)
STIME       :     5.00000D-03          (SAMPLING TIME, sec)
ETIME       :     1.50000D+00          (SIMULATION END TIME, sec)
TWAVE       :     9.00000D-02          (TRAVEL TIME OF PRESSURE WAVE, sec)
TLEAK       :     3.00000D-01          (LEAKAGE START TIME, sec)
ZLEAK       :     4.00000D+01          (LOCATION OF LEAKAGE, m)
PLEAK       :     0.80000D-00          (LEAKAGE PERCENT)
CAPPA       :     0.92500D+00          (FORGETTING FACTOR FOR PRESSURE GRADIENT)
RAMBDA      :     0.95000D+00          (FORGETTING FACTOR FOR CROSS-CORRELATION FN)
PAI-BBD     :    -9.00000D-16          (BOUNDARY VALUE OF CROSS-CORRELATION FN)
PAI-INI     :     0.00000D+00          (INITIAL GUESS OF CROSS-CORRELATION FN)
SCALE(Pi):      2.20000D-02           (MEASUREMENT NOISE SCALE OF Pin) 2.2E-2
SCALE(Po):      7.60000D-03           (MEASUREMENT NOISE SCALE OF Pout)7.6E-3
SCALE(Qi):      3.77000D-08           (MEASUREMENT NOISE SCALE OF Qin) 3.77e-8
SCALE(Qo):      3.77000D-08           (MEASUREMENT NOISE SCALE OF Qout)   "
1
```

3. Fuzzy Logic Control 프로그램

```
      PROGRAM FUZZYSERVO
      COMMON Y(-2:500),U(-2:500),SP(-1:500),E(-1:500),ITIME
      COMMON ESTAR,ESTAR1,ESTAR2,DELE,SIGN

      OPEN(5,FILE='FUZSER.DAT',STATUS='UNKNOWN')
      OPEN(6,FILE='FUZSER.OUT',STATUS='UNKNOWN')

      READ(5,*) TOL
      READ(5,*) ESTAR1,ESTAR2
      READ(5,*) ESTAR,N,DELE,UMAX,UMIN
      READ(5,*) SP0,SP1,SP2,SP3,SP4
      READ(5,*) IC1,IC2
      CLOSE(5)

      IF (ABS(ESTAR1-ESTAR2).LE.TOL) STOP 'ESTAR CONVERGES'

      OPEN(5,FILE='FUZSER.DAT',STATUS='UNKNOWN')

      IF (ESTAR1.EQ.ESTAR) THEN
          DELE=DELE/2
      ENDIF

      ESTAR1=ESTAR2

      DO 20 I=1,IC1
         SP(I)=SP0
20    CONTINUE
      DO 30 I=IC1+1,IC1+IC2
         SP(I)       = SP1
         SP(I+IC2)   = SP2
         SP(I+IC2*2) = SP3
         SP(I+IC2*3) = SP4
30    CONTINUE

      SP(-1)=0
      SP(0)=0
      E(-1)=0
      Y(-2)=0
      Y(-1)=0
      U(-2)=0
      U(-1)=0
      ITIME=0

40    IF ((SP(ITIME)-SP(ITIME-1)).GT.0) THEN

         SIGN=1

41       U(ITIME)=UMAX
         CALL PATH1

         IF (E(ITIME-1).GT.ESTAR) GOTO 41

         DO 42 I=1,N
            U(ITIME)=UMIN
42          CALL PATH1

         CALL PATH2

      ENDIF

      IF ((SP(ITIME)-SP(ITIME-1)).LT.0) THEN
```

```
          SIGN=-1

51        U(ITIME)=UMIN
          CALL PATH1

          IF (-E(ITIME-1).GT.ESTAR) GOTO 51

          DO 52 I=1,N
             U(ITIME)=UMAX
52           CALL PATH1

          CALL PATH2

       ENDIF

       U(ITIME)=SP(ITIME)
       CALL PATH1

       IF (ITIME.EQ.IC1+IC2*4+1) THEN
          WRITE(*,*) ESTAR
          WRITE(5,*) TOL
          WRITE(5,*) ESTAR1,ESTAR2
          WRITE(5,*) ESTAR,N,DELE,UMAX,UMIN
          WRITE(5,*) SP0,SP1,SP2,SP3,SP4
          WRITE(5,*) IC1,IC2
          STOP
       ENDIF

       GOTO 40

       END

       FUNCTION PLANT(YM1,YM2,UM1,UM2)
       PLANT=1.7567*YM1-.7788*YM2+.0115*UM1+.0106*UM2
       RETURN
       END

       SUBROUTINE PATH1
       COMMON Y(-2:500),U(-2:500),SP(-1:500),E(-1:500),ITIME
       COMMON ESTAR,ESTAR1,ESTAR2,DELE,SIGN
       Y(ITIME)=PLANT(Y(ITIME-1),Y(ITIME-2),U(ITIME-1),U(ITIME-2))
       E(ITIME)=SP(ITIME)-Y(ITIME)
       WRITE(6,*) ITIME,SP(ITIME),Y(ITIME),U(ITIME-1)
       ITIME=ITIME+1
       WRITE(*,*) ITIME
       RETURN
       END

       SUBROUTINE PATH2
       COMMON Y(-2:500),U(-2:500),SP(-1:500),E(-1:500),ITIME
       COMMON ESTAR,ESTAR1,ESTAR2,DELE,SIGN
       ESTAR2=ESTAR
       IF (Y(ITIME-1).GT.SP(ITIME-1)) THEN
          ESTAR=ESTAR+DELE*SIGN
       ELSEIF (Y(ITIME-1).LT.SP(ITIME-1)) THEN
          ESTAR=ESTAR-DELE*SIGN
       ENDIF
       RETURN
       END
```

```
      PROGRAM FUZZYID

C *** ISIZE = N * ( K + 1 )
C *** IPRE  = K
C *** NIMP  = N
C *** NRD   = N * (K-NIUV) * 2
C *** NDS   = M

      PARAMETER (ISIZE=60,IPRE=3,NIMP=15,NRD=60,NDS=201)
      COMMON /B1/ X(IPRE,NDS),YY(NDS),DB(NDS),UB(ISIZE,NDS)
      COMMON /B2/ BETA(ISIZE,NDS),S(ISIZE,ISIZE),SRXT(ISIZE,1)
      COMMON /B3/ RX(1,ISIZE),RXT(ISIZE,1)
      COMMON /B4/ RXS(1,ISIZE),US(ISIZE,ISIZE),RXSRXT(1,1)
      COMMON /B5/ DS(1,1),PX(ISIZE,1),OP(ISIZE,1),RXP(1,1)
      COMMON /B6/ ZERO(NIMP,IPRE),ONE(NIMP,IPRE)
      COMMON /B7/ RZERO(NIMP,IPRE),RONE(NIMP,IPRE)
      COMMON /B8/ ZEROU(NIMP,IPRE),ZEROL(NIMP,IPRE)
      COMMON /B9/ ONEU(NIMP,IPRE),ONEL(NIMP,IPRE)
      COMMON /B10/ ZEROB(NIMP,IPRE),ONEB(NIMP,IPRE)
      COMMON /B11/ SZERO(NIMP,IPRE),SONE(NIMP,IPRE)
      COMMON /B12/ RD(NRD,NRD),FV(IPRE)
      COMMON /B13/ M,N,K,IUV(IPRE),NKP1,PI,NIUV
      REAL   Z(NRD)

      WRITE(*,*) 'INPUT WAIT : IWAIT'
      READ(*,*) IWAIT
      WRITE(*,*) 'INPUT TOLERANCE : TOL '
      READ(*,*) TOL
      WRITE(*,*) 'INPUT BOUNDARY DECREASE : CB'
      READ(*,*) CB
      WRITE(*,*) 'INPUT NUMBER OF IMPLICATIONS : N '
      READ(*,*) N
      WRITE(*,*) 'INPUT NUMBER OF DATA SETS : M '
      READ(*,*) M
      WRITE(*,*) 'INPUT NUMBER OF PREMISE VARIABLES : K '
      READ(*,*) K
      WRITE(*,*) 'INPUT NUMBER OF UNCONDITIONED PREMISE V. : NIUV'
      READ(*,*) NIUV
      WRITE(*,*) 'INPUT UNCONDITIONED PREMISE VARIABLE NUMBER : IUV '
      READ(*,*) (IUV(I),I=1,NIUV)

      DO 5 I=1,N
         WRITE(*,*) 'PREMISE PARAMETERS BOUND OF IMPLICATION ',
     &              I,'   (0,1)'
         DO 5 L=1,K
            DO 88 ICUV=1,NIUV
               IF (L.EQ.IUV(ICUV)) GOTO 5
88          CONTINUE
            READ(*,*) RZERO(I,L),ZEROB(I,L),RONE(I,L),ONEB(I,L)
5     CONTINUE

      WRITE(*,*) 'INPUT ALPHA : ALPHA '
      READ(*,*) ALPHA

      OPEN(5,FILE='A.PRN',STATUS='OLD')
```

```
      DO 10 J=1,M
         READ(5,*) (X(L,J),L=1,K),YY(J)
10    CONTINUE

      NKP1=N*(K+1)

      DO 100 I=1,NKP1
         PX(I,1)=0
         DO 110 J=1,NKP1
            IF (I.EQ.J) S(I,J)=ALPHA
            IF (I.NE.J) S(I,J)=0
110      CONTINUE
100   CONTINUE


      PIMIN=1E5
      IT=0
      ITT=0
200   IT=IT+1
      ITT=ITT+1
      ISEED=IT
      DO 300 I=1,N*(K-NIUV)*2
         RD(I,I)=RAN(ISEED)
300   CONTINUE

      IC=0
      DO 6 I=1,N
         DO 7 L=1,K
            DO 77 ICUV=1,NIUV
               IF (L.EQ.IUV(ICUV)) GOTO 7
77          CONTINUE
            ZEROL(I,L)=RZERO(I,L)-CB*ZEROB(I,L)
            ZEROU(I,L)=RZERO(I,L)+CB*ZEROB(I,L)
            ONEL(I,L)=RONE(I,L)-CB*ONEB(I,L)
            ONEU(I,L)=RONE(I,L)+CB*ONEB(I,L)
            IC=IC+1
            Z(IC)=ZEROL(I,L)+RD(IC,IC)*(ZEROU(I,L)-ZEROL(I,L))
            ZERO(I,L)=Z(IC)
            IC=IC+1
            Z(IC)=ONEL(I,L)+RD(IC,IC)*(ONEU(I,L)-ONEL(I,L))
            ONE(I,L)=Z(IC)
7        CONTINUE
6     CONTINUE

      PI=FP(Z)

      PIMINP=PIMIN
      PIMIN=MIN(PI,PIMIN)
      IF (PIMINP.NE.PIMIN) THEN
         ITO=IT
         DO 209 I=1,NKP1
            OP(I,1)=PX(I,1)
209      CONTINUE
         DO 210 I=1,N
            DO 210 L=1,K
```

```
               DO 222 ICUV=1,NIUV
                  IF (L.EQ.IUV(ICUV)) GOTO 210
222            CONTINUE
               SZERO(I,L)=ZERO(I,L)
               SONE(I,L)=ONE(I,L)
210         CONTINUE
         ENDIF

         WRITE(*,*) 'ITERATION : ',IT
         WRITE(*,*) 'PERFORMANCE INDEX : ',PI
         WRITE(*,*) (PX(I,1),I=1,NKP1)
         WRITE(*,*)


         IF (ITT.EQ.IWAIT) THEN
            IF (CB.LE.TOL) THEN
               OPEN(6,FILE='MODEL.REP',STATUS='UNKNOWN')
               WRITE(*,*) '***** PARAMETER(S) IDENTIFIED ***** '
               WRITE(*,*) (OP(I,1),I=1,NKP1)
               WRITE(*,*)

               WRITE(6,*) '??? AFTER ',ITO,' ITERATIONS '
               WRITE(6,*) '=====          REPORT          ===== '
               WRITE(6,*)
               WRITE(6,*) '*****    PERFORMANCE INDEX    ***** '
               WRITE(6,*) PIMIN
               WRITE(6,*)
               WRITE(6,*) '***** PARAMETER(S) IDENTIFIED ***** '
               WRITE(6,*) (OP(I,1),I=1,NKP1)
               WRITE(6,*)
               DO 250 I=1,N
                  WRITE(*,*) '*** IMPLICATION ',I
                  WRITE(*,*) '*** PREMISE PARAMETER(S) : ZERO,ONE '
                  WRITE(*,*) ' (+ , -) ',CB
                  WRITE(6,*) '*** IMPLICATION ',I
                  WRITE(6,*) '*** PREMISE PARAMETER(S) : ZERO,ONE '
                  WRITE(6,*)
                  DO 250 L=1,K
                     DO 555 ICUV=1,NIUV
                        IF (L.EQ.IUV(ICUV)) GOTO 250
555                  CONTINUE
                     WRITE(*,*) SZERO(I,L),SONE(I,L)
                     WRITE(6,*) ' VARIABLE (',L,') '
                     WRITE(6,*) ' ZERO RANGE : ',RZERO(I,L),' (+,-)',
     *                           CB,' *',ZEROB(I,L)
                     WRITE(6,*) ' ** ZERO  :: ',SZERO(I,L)
                     WRITE(6,*)
                     WRITE(6,*) ' ONE  RANGE : ',RONE(I,L),' (+,-)',
     *                           CB,' *',ONEB(I,L)
                     WRITE(6,*) ' ** ONE   :: ',SONE(I,L)
                     WRITE(6,*)
250               CONTINUE
               STOP
            ENDIF
```

```
          CB=CB/2.
          DO 220 I=1,N
             DO 220 L=1,K
                DO 777 ICUV=1,NIUV
                   IF (L.EQ.IUV(ICUV)) GOTO 220
777             CONTINUE
                RZERO(I,L)=SZERO(I,L)
                RONE(I,L)=SONE(I,L)
220          CONTINUE

             WRITE(*,*) '--- AUTOMATIC SEARCH BOUND DECREASE --- '
             WRITE(*,*) 'CURRENT SEARCH BOUND : ',CB
             WRITE(*,*)
             ITT=0
          ENDIF

          GOTO 200

          END


          REAL FUNCTION FA(I,K,X,ZERO,ONE,L,M)
          REAL ZERO(L,M),ONE(L,M)
          SLOPE=1./(ONE(I,K)-ZERO(I,K))
          SINTE=-SLOPE*ZERO(I,K)
          FA=SLOPE*X+SINTE
          RETURN
          END


C
C         T(M,L) = A(M,N)*B(N,L)
C
          SUBROUTINE MULT(A,B,T,M0,N0,L0,M,N,L)
          DIMENSION A(M0,N0),B(N0,L0),T(M0,L0)
          DO 10 I = 1, M
          DO 10 J = 1, L
10           T(I,J) = 0.
          DO 20 I = 1, M
          DO 20 J = 1, L
          DO 20 K = 1, N
20           T(I,J) = A(I,K)*B(K,J) + T(I,J)
          RETURN
          END
C
C         B(N,M) = A(M,N)
C
          SUBROUTINE TRNS(A,B,M0,N0,M,N)
          DIMENSION A(M0,N0),B(N0,M0)
          DO 10 I = 1, M
          DO 10 J = 1, N
10           B(J,I) = A(I,J)
          RETURN
          END
```

```
      FUNCTION RAN(ISEED)
      PARAMETER(IA=7141,IC=54773,IM=259200)
      ISEED=MOD(ISEED*IA+IC,IM)
      RAN=FLOAT(ISEED)/FLOAT(IM)
      RETURN
      END

      FUNCTION FP(Z)
      PARAMETER (ISIZE=60,IPRE=3,NIMP=15,NRD=60,NDS=201)
C      PARAMETER (ISIZE=32,IPRE=3,NIMP=8,NRD=32,NDS=201)
      COMMON /B1/ X(IPRE,NDS),YY(NDS),DB(NDS),UB(ISIZE,NDS)
      COMMON /B2/ BETA(ISIZE,NDS),S(ISIZE,ISIZE),SRXT(ISIZE,1)
      COMMON /B3/ RX(1,ISIZE),RXT(ISIZE,1)
      COMMON /B4/ RXS(1,ISIZE),US(ISIZE,ISIZE),RXSRXT(1,1)
      COMMON /B5/ DS(1,1),PX(ISIZE,1),OP(ISIZE,1),RXP(1,1)
      COMMON /B6/ ZERO(NIMP,IPRE),ONE(NIMP,IPRE)
      COMMON /B7/ RZERO(NIMP,IPRE),RONE(NIMP,IPRE)
      COMMON /B8/ ZEROU(NIMP,IPRE),ZEROL(NIMP,IPRE)
      COMMON /B9/ ONEU(NIMP,IPRE),ONEL(NIMP,IPRE)
      COMMON /B10/ ZEROB(NIMP,IPRE),ONEB(NIMP,IPRE)
      COMMON /B11/ SZERO(NIMP,IPRE),SONE(NIMP,IPRE)
      COMMON /B12/ RD(NRD,NRD),FV(IPRE)
      COMMON /B13/ M,N,K,IUV(IPRE),NKP1,PI,NIUV
      REAL   Z(NRD)

      IC=0
      DO 6 I=1,N
         DO 7 L=1,K
            DO 88 ICUV=1,NIUV
               IF (L.EQ.IUV(ICUV)) GOTO 7
88          CONTINUE
            IC=IC+1
            ZERO(I,L)=Z(IC)
            IC=IC+1
            ONE(I,L)=Z(IC)
7        CONTINUE
6     CONTINUE
      PI=0

      DO 20 J=1,M
         DBSUM=0
         DO 30 I=1,N
            DO 40 L=1,K
               DO 44 ICUV=1,NIUV
                  IF (L.EQ.IUV(ICUV)) GOTO 40
44             CONTINUE
               FV(L)=FA(I,L,X(L,J),ZERO,ONE,NIMP,IPRE)
40          CONTINUE
            SMALL=1.
            DO 41 L=1,K
               DO 11 ICUV=1,NIUV
                  IF (L.EQ.IUV(ICUV)) GOTO 41
11             CONTINUE
```

```
                IF (FV(L).GT.1.OR.FV(L).LT.0) THEN
                   SMALL=0
                   GOTO 42
                ELSE
                   SMALL=MIN(SMALL,FV(L))
                ENDIF
41           CONTINUE

42           UB(I,J)=SMALL
             DBSUM=DBSUM+SMALL
30        CONTINUE

C *** ?????
          IF (DBSUM.EQ.0) GOTO 20
C *** ?????

          DB(J)=DBSUM
          DO 50 I=1,N
             BETA(I,J)=UB(I,J)/DB(J)
50        CONTINUE
20     CONTINUE


       DO 60 J=1,M
          IC=0
          DO 70 I=1,N
             IC=IC+1
             RX(1,IC)=BETA(I,J)
70        CONTINUE
          DO 80 L=1,K
             DO 90 I=1,N
                IC=IC+1
                RX(1,IC)=BETA(I,J)*X(L,J)
90           CONTINUE
80        CONTINUE


          NKP=NKP1
          NKP1=ISIZE
          CALL TRNS(RX,RXT,1,NKP1,1,NKP)
          CALL MULT(S,RXT,SRXT,NKP1,NKP1,1,NKP,NKP,1)
          CALL MULT(RX,S,RXS,1,NKP1,NKP1,1,NKP,NKP)
          CALL MULT(SRXT,RXS,US,NKP1,1,NKP1,NKP,1,NKP)
          CALL MULT(RXS,RXT,RXSRXT,1,NKP1,1,1,NKP,1)
          DS(1,1)=-1/(1+RXSRXT(1,1))

          NKP1=NKP
          DO 120 I=1,NKP1
             DO 130 J1=1,NKP1
                S(I,J1)=S(I,J1)+DS(1,1)*US(I,J1)
130          CONTINUE
120       CONTINUE
```

```
          NKP=NKP1
          NKP1=ISIZE
          CALL MULT(RX,PX,RXP,1,NKP1,1,1,NKP,1)
          ERROR=YY(J)-RXP(1,1)

          PI=PI+ERROR**2

          CALL MULT(S,RXT,SRXT,NKP1,NKP1,1,NKP,NKP,1)

          NKP1=NKP
          DO 140 I=1,NKP1
             PX(I,1)=PX(I,1)+SRXT(I,1)*ERROR
140       CONTINUE

60     CONTINUE

       PI=SQRT(PI)/SQRT(NDS)
       FP=PI
       RETURN
       END
```

```
C****** WOBE DISTILLATION COLUMN MODEL SIMULATION
C****** FLC + PROCESS : SP ------> Y
C****** WB2.FOR + FLC
C****** FLC : PARAMETER FROM FUZIDFN.FOR
C****** DATA FILE : FLC3.DAT
C****** # OF IMPLICATIONS : 8
C****** RESULT : REDUCED OVERSHOOT
C****** (+,-) SYMMETRY MUST BE CONSIDERED TO GET IMPROVED RESPONSE
C****** INTRODUCE MEASUREMENT NOISE (-.05,.05)

      IMPLICIT REAL (M)
      PARAMETER (ISIZE=44,IPRE=2,NIMP=11)
      COMMON /BL1/ P(2,ISIZE,1)
      COMMON /BL2/ ZERO(2,NIMP,IPRE),ONE(2,NIMP,IPRE)
      COMMON /BL3/ N,K,NKP1,UM
      COMMON /BL4/ VMAX(2),VMIN(2),SP0(2)
      REAL U1(-8:500),U2(-4:500),Y1(-1:500),Y2(-1:500)
      REAL G1(-1:500),G2(-1:500),G3(-1:500),G4(-1:500)
      REAL SP1(0:500),SP2(0:500),E1(-1:500),E2(-1:500)
      REAL M1(-5:500),M2(-3:500),M3(-1:500),M4(-1:500)
      REAL PAIN(2),ESIGN(2,2)

      DO 10 I=-8,-1
10       U1(I)=0
      DO 11 I=-4,-1
11       U2(I)=0
      DO 12 I=-5,-1
12       M1(I)=0
      DO 13 I=-3,-1
13       M2(I)=0


      G1(-1)=0
      G2(-1)=0
      G3(-1)=0
      G4(-1)=0
      E1(-1)=0
      E2(-1)=0
      Y1(-1)=0
      Y2(-1)=0
      M3(-1)=0
      M4(-1)=0


      WRITE(*,*) '***** DISTURBANCE : CD,DM,UM *****'
      READ(*,*) CD,DM,UM
      WRITE(*,*) '***** INPUT : SP1,SP2,DURATION *****'
      READ(*,*) SP0(1),SP0(2),ITER
      WRITE(*,*) 'INPUT NUMBER OF IMPLICATIONS : N '
      READ(*,*) N
      WRITE(*,*) 'INPUT NUMBER OF PREMISE VARIABLES : K '
      READ(*,*) K
      WRITE(*,*) 'INPUT NUMBER OF CONSEQUENCE VARIABLES : NCV '
      READ(*,*) NCV
```

```
      WRITE(*,*) '***** IMPLICATION VARIABLE MAX, MIN ***** '
      DO 1 I=1,K
1     READ(*,*) VMAX(I),VMIN(I)

      OPEN(6,FILE='A1.PRN',STATUS='UNKNOWN')

      DO 5 II=1,NCV
      DO 5 I=1,N
         WRITE(*,*) 'PREMISE PARAMETERS  OF IMPLICATION ',
     &              I,'   (0,1)'
         READ(*,*) (ZERO(II,I,L),ONE(II,I,L),L=1,K)
5     CONTINUE


      DO 6 II=1,NCV
      DO 6 I=1,N
         READ(*,*) (P(II,J,1),J=4*(I-1)+1,4*I)
6     CONTINUE

      NKP1=N*(K+2)

      DO 20 I=0,ITER
         SP1(I)=SP0(1)
         SP2(I)=SP0(2)
20    CONTINUE

      C1G1=9.418773E-1
      C2G1=7.439702E-1
      C1G2=9.329119E-1
      C2G2=-1.301508
      C1G3=9.123395E-1
      C2G3=5.785594E-1
      C1G4=9.534969E-1
      C2G4=-8.789076E-1
      C1M3=.9123
      C2M3=.4494
      C3M3=-.4196
      C1M4=.9535
      C2M4=1.1742
      C3M4=-1.1055

      ITIME=0

      iseed=0

30    CONTINUE

      E1(ITIME)=SP1(ITIME)-Y1(ITIME-1)
c      ESIGN(1,1)=SIGN(1,E1(ITIME))
c      ESIGN(1,2)=SIGN(1,E1(ITIME-1))
c      IF (ESIGN(1,1).LT.0) E1(ITIME)=-E1(ITIME)
c      IF (ESIGN(1,2).LT.0) E1(ITIME-1)=-E1(ITIME-1)
      PAIN(1)=E1(ITIME)
      PAIN(2)=E1(ITIME-1)
      CALL FLC(PAIN,M1(ITIME),M1(ITIME-1),1,ESIGN)
```

```
c       IF (ESIGN(1,1).LT.0) E1(ITIME)=-E1(ITIME)
c       IF (ESIGN(1,2).LT.0) E1(ITIME-1)=-E1(ITIME-1)



        E2(ITIME)=SP2(ITIME)-Y2(ITIME-1)
c       ESIGN(2,1)=SIGN(1,E2(ITIME))
c       ESIGN(2,2)=SIGN(1,E2(ITIME-1))
c       IF (ESIGN(2,1).LT.0) E2(ITIME)=-E2(ITIME)
c       IF (ESIGN(2,2).LT.0) E2(ITIME-1)=-E2(ITIME-1)
        PAIN(1)=E2(ITIME)
        PAIN(2)=E2(ITIME-1)
        CALL FLC(PAIN,M2(ITIME),M2(ITIME-1),2,ESIGN)
c       IF (ESIGN(2,1).LT.0) E2(ITIME)=-E2(ITIME)
c       IF (ESIGN(2,2).LT.0) E2(ITIME-1)=-E2(ITIME-1)

C         M1(ITIME)=M1(ITIME-1)+.2278*E1(ITIME)-.2*E1(ITIME-1)
C         M2(ITIME)=M2(ITIME-1)-.0569*E2(ITIME)+.047*E2(ITIME-1)

        M3(ITIME)=FM(M3(ITIME-1),M1(ITIME-4),M1(ITIME-5),C1M3,C2M3,C3M3)
        M4(ITIME)=FM(M4(ITIME-1),M2(ITIME-2),M2(ITIME-3),C1M4,C2M4,C3M4)

        U1(ITIME)=M1(ITIME)+M4(ITIME)
        U2(ITIME)=M2(ITIME)+M3(ITIME)

        G1(ITIME)=FG(G1(ITIME-1),U1(ITIME-2),C1G1,C2G1)
        G2(ITIME)=FG(G2(ITIME-1),U2(ITIME-4),C1G2,C2G2)
        G3(ITIME)=FG(G3(ITIME-1),U1(ITIME-8),C1G3,C2G3)
        G4(ITIME)=FG(G4(ITIME-1),U2(ITIME-4),C1G4,C2G4)

        iseed=iseed+1
        Y1(ITIME)=G1(ITIME)+G4(ITIME)+(.5-ran(iseed))*DM+CD
        iseed=iseed+1
        Y2(ITIME)=G2(ITIME)+G3(ITIME)+(.5-ran(iseed))*DM+CD

        WRITE(6,*) ITIME,Y1(ITIME),Y2(ITIME),M1(ITIME),M2(ITIME)

        IF (ITIME.EQ.ITER) STOP

        ITIME=ITIME+1

        GOTO 30

        END

        FUNCTION FG(P1,P2,C1,C2)
        FG=C1*P1+C2*P2
        RETURN
        END

        FUNCTION FM(P1,P2,P3,C1,C2,C3)
        FM=C1*P1+C2*P2+C3*P3
        RETURN
        END
```

```
      SUBROUTINE FLC(X,RXP,UM1,IZ,ESIGN)
      PARAMETER (ISIZE=44,IPRE=2,NIMP=11)
      REAL X(IPRE),DB,UB(ISIZE),BETA(ISIZE)
      REAL RX(1,ISIZE),RXP(1,1)
      COMMON /BL1/ P(2,ISIZE,1)
      COMMON /BL2/ ZERO(2,NIMP,IPRE),ONE(2,NIMP,IPRE)
      COMMON /BL3/ N,K,NKP1,UM
      REAL FV(IPRE),ESIGN(2,2)

      DBSUM=0
      DO 30 I=1,N
         DO 40 L=1,K
            FV(L)=FA(I,L,X(L),IZ,NIMP,IPRE)
40       CONTINUE
         SMALL=1.
         DO 41 L=1,K
            IF(FV(L).GT.1.OR.FV(L).LT.0) THEN
               SMALL=0
               GOTO 42
            ELSE
               SMALL=MIN(SMALL,FV(L))
            ENDIF
41       CONTINUE
42       UB(I)=SMALL
         DBSUM=DBSUM+SMALL
30    CONTINUE

      if (dbsum.eq.0) then
         rxp(1,1)=um1
         return
      endif

      DB=DBSUM

      DO 50 I=1,N
         BETA(I)=UB(I)/DB
50    CONTINUE

      DO 55 L=1,K
         X(L)=X(L)
55    CONTINUE

      IC=0
      DO 70 I=1,N
         IC=IC+1
         RX(1,IC)=BETA(I)
70    CONTINUE
      DO 80 L=1,K
         DO 90 I=1,N
            IC=IC+1
            RX(1,IC)=BETA(I)*X(L)
90       CONTINUE
80    CONTINUE
      DO 75 I=1,N
         IC=IC+1
```

```
          RX(1,IC)=BETA(I)*UM1
75      CONTINUE

        CALL MULT(RX,IZ,RXP,1,NKP1,1)
c        IFLAG=0

c        if (esign(iz,1).lt.0) rxp(1,1)=rxp(1,1)*UM

        RETURN
        END


        REAL FUNCTION FA(I,K,X,IZ,L,M)
        COMMON /BL2/ ZERO(2,11,2),ONE(2,11,2)
        COMMON /BL4/ VMAX(2),VMIN(2),SP0(2)
        SLOPE=1./(ONE(IZ,I,K)-ZERO(IZ,I,K))/SP0(IZ)
        SINTE=-SLOPE*ZERO(IZ,I,K)*SP0(IZ)
c        IF (X.LT.VMIN(IZ)) THEN
c           FA=0
c        ELSEIF (X.GT.VMAX(IZ)) THEN
c           FA=1
c        ELSE
           FA=SLOPE*X+SINTE
c        ENDIF
        RETURN
        END

        FUNCTION RAN(ISEED)
        PARAMETER(IA=7141,IC=54773,IM=259200)
        ISEED=MOD(ISEED*IA+IC,IM)
        RAN=FLOAT(ISEED)/FLOAT(IM)
        RETURN
        END

C
C       T(M,L) = A(M,N)*B(N,L)
C
        SUBROUTINE MULT(A,IZ,T,M,N,L)
        COMMON /BL1/ B(2,44,1)
        DIMENSION A(M,N),T(M,L)
        DO 10 I = 1, M
        DO 10 J = 1, L
10         T(I,J) = 0.
        DO 20 I = 1, M
        DO 20 J = 1, L
        DO 20 K = 1, N
20         T(I,J) = A(I,K)*B(IZ,K,J) + T(I,J)
        RETURN
        END
```

```
0,0,1
1,1,200
11
2
2
1 -.1819
1 -.1633
-.1819 0 -.18   0
-.1819 0  .8443 0
.6703 0  -.18  0
.6703 0  .8443 0
.6703 0  .2986 1
.2117 .4586  .8443 0
.2117 .4586  .2986 1
.8196 .4586  .8443 0
.8196 .4586  .2986 1
.3638 1  .8443 0
.3638 1  .2986 1
-.0856 0 -.1633 0
-.0856 0  .5855 0
.7072 0 -.1633 0
.7072 0  .5855 0
.7072 0  .2728 1
.1201 .4618  .5855 0
.1201 .4618  .2728 1
.8672 .4618  .5855 0
.8672 .4618  .2728 1
.3627 1  .5855 0
.3627 1  .2728 1
    9.083098E-06    3.060412E-04    2.521482E-03   -2.572884E-04
    9.286980E-02    2.434622E-02    1.045023E-01    1.678540E-01
    1.358987E-01    1.022834E-01    1.901114E-03    2.276451E-01
    2.264430E-01    2.288217E-01    2.280312E-01    4.980905E-02
    2.276957E-01    2.107476E-01    2.479026E-02    1.510522E-01
    1.255228E-01    2.273882E-01   -1.998743E-01   -1.981687E-01
   -1.953659E-01   -2.007091E-01    6.081709E-02   -1.828377E-01
   -1.010641E-01    5.089072E-02   -5.452921E-02   -3.297240E-02
   -1.999194E-01    9.999325E-01    9.977873E-01    9.820902E-01
    1.001848        2.496811E-01    8.461466E-01    3.178515E-01
    3.529404E-02    1.819319E-01    3.133413E-01    9.929867E-01
   -3.984950E-05   -2.112143E-04   -4.545483E-03   -4.299697E-05
   -7.240065E-02   -9.787425E-02   -9.551924E-02   -1.017420E-01
   -6.448774E-02   -7.116760E-02   -1.442773E-04   -5.664988E-02
   -5.558926E-02   -5.889896E-02   -5.758184E-02   -2.381591E-02
   -3.543465E-02   -2.038205E-02    2.479633E-03    9.635296E-02
    1.427214E-02   -5.703261E-02    4.680140E-02    4.492212E-02
    5.045434E-02    4.749054E-02    2.399387E-02    4.215046E-02
    2.282067E-02    1.003337E-03   -8.724957E-02   -4.133355E-03
    4.719295E-02    9.995689E-01    9.976844E-01    9.508845E-01
    9.995335E-01    3.058110E-01    8.664101E-02    8.931703E-02
    2.125727E-02    3.449558E-01    3.342425E-01    9.984765E-01
```

??? AFTER 60 ITERATIONS
===== REPORT =====

***** PERFORMANCE INDEX *****
6.340078E-06

***** PARAMETER(S) IDENTIFIED *****

9.083098E-06	3.060412E-04	0.000000E+00	2.521482E-03
-2.572884E-04	9.286980E-02	0.000000E+00	2.434622E-02
1.045023E-01	1.138101E-01	1.678540E-01	1.358987E-01
1.138960E-01	1.022834E-01	1.901114E-03	2.276451E-01
2.264430E-01	0.000000E+00	2.288217E-01	2.280312E-01
4.980905E-02	0.000000E+00	2.276957E-01	2.107476E-01
1.138102E-01	2.479026E-02	1.510522E-01	1.139022E-01
1.255228E-01	2.273882E-01	-1.998743E-01	-1.981687E-01
0.000000E+00	-1.953659E-01	-2.007091E-01	6.081709E-02
0.000000E+00	-1.828377E-01	-1.010641E-01	0.000000E+00
5.089072E-02	-5.452921E-02	0.000000E+00	-3.297240E-02
-1.999194E-01	9.999325E-01	9.977873E-01	0.000000E+00
9.820902E-01	1.001848	2.496811E-01	0.000000E+00
8.461466E-01	3.178515E-01	0.000000E+00	3.529404E-02
1.819319E-01	0.000000E+00	3.133413E-01	9.929867E-01

*** IMPLICATION 1
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (1)
ZERO RANGE : -2.061407E-01 (+,-) 3.125000E-02 * 2.000000E-01
** ZERO :: -2.015866E-01

ONE RANGE : 0.000000E+00 (+,-) 3.125000E-02 * 0.000000E+00
** ONE :: 0.000000E+00

VARIABLE (2)
ZERO RANGE : 1.234297E-01 (+,-) 3.125000E-02 * 2.000000E-01
** ZERO :: 1.239524E-01

ONE RANGE : 0.000000E+00 (+,-) 3.125000E-02 * 0.000000E+00
** ONE :: 0.000000E+00

*** IMPLICATION 2
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (1)
ZERO RANGE : -1.614212E-01 (+,-) 3.125000E-02 * 2.000000E-01
** ZERO :: -1.620710E-01

ONE RANGE : 0.000000E+00 (+,-) 3.125000E-02 * 0.000000E+00
** ONE :: 0.000000E+00

VARIABLE (2)
ZERO RANGE : 7.922933E-01 (+,-) 3.125000E-02 * 3.000000E-01
** ZERO :: 7.980146E-01

ONE RANGE : 0.000000E+00 (+,-) 3.125000E-02 * 0.000000E+00

```
 ** ONE    ::     0.000000E+00

*** IMPLICATION            3
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (           1)
 ZERO RANGE :   -1.271646E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO   ::   -1.313054E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

 VARIABLE (           2)
 ZERO RANGE :    3.170809E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO   ::    3.157255E-01

 ONE  RANGE :        1.000000 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::        1.000000

*** IMPLICATION            4
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (           1)
 ZERO RANGE :    6.616101E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO   ::    6.673153E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

 VARIABLE (           2)
 ZERO RANGE :   -8.255064E-02 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO   ::   -7.812572E-02

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

*** IMPLICATION            5
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (           1)
 ZERO RANGE :    7.066060E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO   ::    7.059637E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

 VARIABLE (           2)
 ZERO RANGE :        1.198739 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO   ::        1.205522

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

*** IMPLICATION            6
*** PREMISE PARAMETER(S) : ZERO,ONE
```

```
VARIABLE (          1)
ZERO RANGE :    6.359076E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    6.377372E-01

ONE  RANGE :    0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::    0.000000E+00

VARIABLE (          2)
ZERO RANGE :    2.514438E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    2.526083E-01

ONE  RANGE :      1.000000 (+,-)      3.125000E-02 *    0.000000E+00
** ONE    ::      1.000000

*** IMPLICATION           7
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    1.607875E-01 (+,-)    3.125000E-02 *    2.000000E-01
** ZERO   ::    1.667571E-01

ONE  RANGE :    5.224448E-01 (+,-)    3.125000E-02 *    2.000000E-01
** ONE    ::    5.162844E-01

VARIABLE (          2)
ZERO RANGE :   -1.826976E-01 (+,-)    3.125000E-02 *    2.000000E-01
** ZERO   ::   -1.839539E-01

ONE  RANGE :    0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::    0.000000E+00

*** IMPLICATION           8
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    2.260626E-01 (+,-)    3.125000E-02 *    2.000000E-01
** ZERO   ::    2.287727E-01

ONE  RANGE :    4.267709E-01 (+,-)    3.125000E-02 *    2.000000E-01
** ONE    ::    4.323626E-01

VARIABLE (          2)
ZERO RANGE :    7.139356E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    7.074051E-01

ONE  RANGE :    0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::    0.000000E+00

*** IMPLICATION           9
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    1.975414E-01 (+,-)    3.125000E-02 *    2.000000E-01
** ZERO   ::    1.945384E-01
```

```
 ONE  RANGE :    3.567450E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    3.521907E-01

 VARIABLE (          2)
 ZERO RANGE :    3.445566E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    3.526161E-01

 ONE  RANGE :         1.000000 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::        1.000000

*** IMPLICATION          10
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :         1.003002 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::        1.001554

 ONE  RANGE :    6.234784E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    6.257688E-01

 VARIABLE (          2)
 ZERO RANGE :   -1.270113E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::   -1.310320E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::    0.000000E+00

*** IMPLICATION          11
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    7.785544E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::    7.748195E-01

 ONE  RANGE :    4.464155E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    4.405969E-01

 VARIABLE (          2)
 ZERO RANGE :    8.003817E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    8.036629E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::    0.000000E+00

*** IMPLICATION          12
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    8.686444E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::    8.643352E-01

 ONE  RANGE :    6.033341E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    6.091750E-01
```

```
VARIABLE (          2)
ZERO RANGE :    4.089194E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    4.094989E-01

ONE  RANGE :        1.000000 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::       1.000000

*** IMPLICATION          13
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    1.515737E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    1.426513E-01

ONE  RANGE :        1.000000 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::       1.000000

VARIABLE (          2)
ZERO RANGE :   -6.549161E-02 (+,-)    3.125000E-02 *    2.000000E-01
** ZERO   ::   -6.936010E-02

ONE  RANGE :    0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::    0.000000E+00

*** IMPLICATION          14
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    8.428196E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    8.506746E-01

ONE  RANGE :        1.000000 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::       1.000000

VARIABLE (          2)
ZERO RANGE :    7.080780E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    7.067958E-01

ONE  RANGE :    0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::    0.000000E+00

*** IMPLICATION          15
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :   -1.191289E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::   -1.231769E-01

ONE  RANGE :        1.000000 (+,-)    3.125000E-02 *    0.000000E+00
** ONE    ::       1.000000

VARIABLE (          2)
ZERO RANGE :    1.820323E-01 (+,-)    3.125000E-02 *    3.000000E-01
** ZERO   ::    1.795067E-01
```

```
ONE  RANGE :        1.000000 (+,-)   3.125000E-02 *   0.000000E+00
** ONE    ::       1.000000
```

```
??? AFTER          60 ITERATIONS
=====          REPORT          =====

*****    PERFORMANCE INDEX    *****
   1.038696E-05

***** PARAMETER(S) IDENTIFIED *****
  -3.984950E-05   -2.112143E-04    0.000000E+00   -4.545483E-03
  -4.299697E-05   -7.240065E-02    0.000000E+00   -9.787425E-02
  -9.551924E-02   -2.838715E-02   -1.017420E-01   -6.448774E-02
  -2.845020E-02   -7.116760E-02   -1.442773E-04   -5.664988E-02
  -5.558926E-02    0.000000E+00   -5.889896E-02   -5.758184E-02
  -2.381591E-02    0.000000E+00   -3.543465E-02   -2.038205E-02
  -2.838710E-02    2.479633E-03    9.635296E-02   -2.844994E-02
   1.427214E-02   -5.703261E-02    4.680140E-02    4.492212E-02
   0.000000E+00    5.045434E-02    4.749054E-02    2.399387E-02
   0.000000E+00    4.215046E-02    2.282067E-02    0.000000E+00
   1.003337E-03   -8.724957E-02    0.000000E+00   -4.133355E-03
   4.719295E-02    9.995689E-01    9.976844E-01    0.000000E+00
   9.508845E-01    9.995335E-01    3.058110E-01    0.000000E+00
   8.664101E-02    8.931703E-02    0.000000E+00    2.125727E-02
   3.449558E-01    0.000000E+00    3.342425E-01    9.984765E-01

*** IMPLICATION          1
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :   -2.226708E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO   ::   -2.181167E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

 VARIABLE (          2)
 ZERO RANGE :   -5.115595E-02 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO   ::   -5.063324E-02

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

*** IMPLICATION          2
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    4.760428E-02 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO   ::    4.695444E-02

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE    ::    0.000000E+00

 VARIABLE (          2)
 ZERO RANGE :    9.937481E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO   ::    9.994693E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
```

```
 ** ONE   ::     0.000000E+00

*** IMPLICATION            3
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (           1)
 ZERO RANGE :    -9.169656E-04 (+,-)    3.125000E-02 *    2.000000E-01
 ** ZERO  ::    -5.057735E-03

 ONE  RANGE :     0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
 ** ONE   ::     0.000000E+00

 VARIABLE (           2)
 ZERO RANGE :     4.443691E-01 (+,-)    3.125000E-02 *    3.000000E-01
 ** ZERO  ::     4.430137E-01

 ONE  RANGE :         1.000000 (+,-)    3.125000E-02 *    0.000000E+00
 ** ONE   ::         1.000000

*** IMPLICATION            4
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (           1)
 ZERO RANGE :     5.643149E-01 (+,-)    3.125000E-02 *    3.000000E-01
 ** ZERO  ::     5.700201E-01

 ONE  RANGE :     0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
 ** ONE   ::     0.000000E+00

 VARIABLE (           2)
 ZERO RANGE :    -2.804697E-01 (+,-)    3.125000E-02 *    2.000000E-01
 ** ZERO  ::    -2.760448E-01

 ONE  RANGE :     0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
 ** ONE   ::     0.000000E+00

*** IMPLICATION            5
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (           1)
 ZERO RANGE :     7.601443E-01 (+,-)    3.125000E-02 *    3.000000E-01
 ** ZERO  ::     7.595020E-01

 ONE  RANGE :     0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
 ** ONE   ::     0.000000E+00

 VARIABLE (           2)
 ZERO RANGE :     8.401942E-01 (+,-)    3.125000E-02 *    3.000000E-01
 ** ZERO  ::     8.469771E-01

 ONE  RANGE :     0.000000E+00 (+,-)    3.125000E-02 *    0.000000E+00
 ** ONE   ::     0.000000E+00

*** IMPLICATION            6
*** PREMISE PARAMETER(S) : ZERO,ONE
```

```
VARIABLE (          1)
ZERO RANGE :    7.902790E-01 (+,-)   3.125000E-02 *   3.000000E-01
** ZERO   ::    7.921087E-01

ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
** ONE    ::    0.000000E+00

VARIABLE (          2)
ZERO RANGE :    2.687320E-01 (+,-)   3.125000E-02 *   3.000000E-01
** ZERO   ::    2.698965E-01

ONE  RANGE :      1.000000 (+,-)     3.125000E-02 *   0.000000E+00
** ONE    ::      1.000000

*** IMPLICATION          7
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    7.259076E-02 (+,-)   3.125000E-02 *   2.000000E-01
** ZERO   ::    7.856033E-02

ONE  RANGE :    4.843870E-01 (+,-)   3.125000E-02 *   2.000000E-01
** ONE    ::    4.782266E-01

VARIABLE (          2)
ZERO RANGE :   -1.789499E-01 (+,-)   3.125000E-02 *   2.000000E-01
** ZERO   ::   -1.802061E-01

ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
** ONE    ::    0.000000E+00

*** IMPLICATION          8
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    1.634214E-01 (+,-)   3.125000E-02 *   2.000000E-01
** ZERO   ::    1.661315E-01

ONE  RANGE :    3.809352E-01 (+,-)   3.125000E-02 *   2.000000E-01
** ONE    ::    3.865269E-01

VARIABLE (          2)
ZERO RANGE :    3.203905E-01 (+,-)   3.125000E-02 *   3.000000E-01
** ZERO   ::    3.138600E-01

ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
** ONE    ::    0.000000E+00

*** IMPLICATION          9
*** PREMISE PARAMETER(S) : ZERO,ONE

VARIABLE (          1)
ZERO RANGE :    7.712241E-02 (+,-)   3.125000E-02 *   2.000000E-01
** ZERO   ::    7.411937E-02
```

```
 ONE  RANGE :    3.197982E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    3.152439E-01

 VARIABLE (          2)
 ZERO RANGE :    2.518448E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    2.599043E-01

 ONE  RANGE :       1.000000 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::       1.000000

*** IMPLICATION          10
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    8.414716E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::    8.400239E-01

 ONE  RANGE :    7.120872E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    7.143776E-01

 VARIABLE (          2)
 ZERO RANGE :   -7.159695E-02 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::   -7.561764E-02

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::    0.000000E+00

*** IMPLICATION          11
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    8.675799E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::    8.638450E-01

 ONE  RANGE :    4.522465E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    4.464279E-01

 VARIABLE (          2)
 ZERO RANGE :    7.468367E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    7.501178E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::    0.000000E+00

*** IMPLICATION          12
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    8.748921E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::    8.705829E-01

 ONE  RANGE :    6.930540E-01 (+,-)   3.125000E-02 *   2.000000E-01
 ** ONE   ::    6.988950E-01
```

```
 VARIABLE (          2)
 ZERO RANGE :    5.437076E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    5.442871E-01

 ONE  RANGE :         1.000000 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::        1.000000

*** IMPLICATION          13
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    2.492785E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    2.403561E-01

 ONE  RANGE :         1.000000 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::        1.000000

 VARIABLE (          2)
 ZERO RANGE :   -3.341059E-02 (+,-)   3.125000E-02 *   2.000000E-01
 ** ZERO  ::   -3.727908E-02

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::    0.000000E+00

*** IMPLICATION          14
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    6.413578E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    6.492128E-01

 ONE  RANGE :         1.000000 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::        1.000000

 VARIABLE (          2)
 ZERO RANGE :    7.320329E-01 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    7.307507E-01

 ONE  RANGE :    0.000000E+00 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::    0.000000E+00

*** IMPLICATION          15
*** PREMISE PARAMETER(S) : ZERO,ONE

 VARIABLE (          1)
 ZERO RANGE :    8.024263E-02 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    7.619467E-02

 ONE  RANGE :         1.000000 (+,-)   3.125000E-02 *   0.000000E+00
 ** ONE   ::        1.000000

 VARIABLE (          2)
 ZERO RANGE :    1.932045E-02 (+,-)   3.125000E-02 *   3.000000E-01
 ** ZERO  ::    1.679491E-02
```

```
  ONE  RANGE :        1.000000 (+,-)   3.125000E-02 *   0.000000E+00
  ** ONE   ::       1.000000
```

주 의

1. 이 보고서는 과학기술처에서 시행한 특정연구개발사업의 연구보고서이다.

2. 이 연구개발내용을 대외적으로 발표할 때에는 반드시 과학기술처에서 시행한 특정연구개발사업의 연구결과임을 밝혀야 한다.