2002 추계학술발표회 논문집
한국원자력학회

원전 고등제어 알고리즘 설계를 위한 소규모 시뮬레이터 개발

Development of a Small-Scale Simulator for Designing Advanced Control Algorithms at Nuclear Power Plants

심영록, 나만균
조선대학교
광주 동구 서석동 375

노희천
한국과학기술원
대전 유성구 구성동 373-1

요 약

제어계통의 개발 및 시험을 위해 설계된 고등 제어 알고리즘을 여러 기존 시뮬레이터에서 구현하는데 어려움이 있어 왔다. 특히, 고등제어 알고리즘을 설계하여 구현하는 것은 거의 불가능하다. 따라서 고등제어 알고리즘을 구현하기 쉽고, 프로그램 상호간 인터페이스가 강화된 원전 시뮬레이터를 개발하는 것이 필요하게 되었다. 본 연구의 목적은 고리2호기 원자력발전소의 특성에 맞는 운전원 사고 대처 훈련을 위한 실시간 시뮬레이터 개량화를 통한 고등제어 알고리즘을 적용할 수 있는 시뮬레이터를 개발하는 것이다. 개발된 원전 시뮬레이터는 FORTRAN, Visual C++ 및 MATLAB 프로그램으로 구성되어 있으며, 한 예로써 증기발생기 수위를 제어하기 위하여 고등제어 알고리즘의 하나인 이동구간제어방법을 이용하여 구현하였다.

Abstract

There has been a difficulty in implementing control algorithms (including even proportional-integral-derivative control algorithms) designed to verify and test to many small-scale simulators. In particular, it was almost impossible to implement advanced control algorithms to them. Therefore, it is required to develop a new simulator that facilitates the implementation of advanced control algorithms and the interface between different kinds of application programs. The objective of this work is to improve the existing small-scale simulator (FISA-2/WS) for the Kori unit 2 nuclear power plant and thus enable advanced control algorithms to be tested and verified by being applied to it. The developed simulator consists of FORTRAN, Visual C++, and MATLAB programs and an advanced control method using a receding horizon control method was applied to it to control the steam generator water level.

1. 서 론

원자력발전소 정상 및 비정상 조건하에서 발전소 거동의 정확한 예측은 정확한 열수력 모델뿐만 아니고 보조계통, 기계적 부품 및 제어 함수의 시뮬레이션을 위한 설비가 필요하다. 이런 능력은 MARS[1], RELAP[2], TRAC[3] 코드 등과 같은 대규모 안전해석코드를 통하여 어느 정도 달성될

수 있다 할지라도 이런 코드들은 대부분 LOCA(Loss of Coolant Accident)와 같은 설계기준사고의 원전 안전해석을 하기 위한 시뮬레이션을 수행하도록 개발되었다. 그러나 이런 사고상황에서는 보통 제어기능이 동작하지도 않으며, 동작한다 할지라도 정상적인 성능을 기대할 수가 없다. 또한 이런 코드들은 제어계통의 개발 및 시험을 위해 최적화되지 않아서 기존 PID 제어 알고리즘을 코딩하여 구현하는데도 상당한 어려움을 유발하며, 최신 고등제어기를 설계하여 구동되게 하는 것은 거의 불가능하다. 따라서 고등제어 알고리즘을 개발하기 위하여 널리 이용되고 있는 MATLAB[4] 소프트웨어 등과 같은 어플리케이션 소프트웨어의 인터페이스가 강화된 시뮬레이터를 개발할 필요가 제기되었다.

본 연구에서는 고리2호기 원자력발전소의 특성에 맞는 운전원의 사고 대처 훈련을 위한 실시간 시뮬레이터(FISA-2/WS)[5]를 일반 데스크탑에서 사용할 수 있도록 개량화 하고, 고등제어 알고리즘을 적용할 수 있도록 NPPS(Nuclear Power Plant Simulator) 프로그램을 개발하였다. NPPS에 사용된 제어기는 이동구간제어방법(Receding Horizon Control Method)을 이용한 증기발생기 수위제어기가 적용되었고, FORTRAN(플랜트 시뮬레이션 계산)[6], Visual C++ (그래픽 디자인)[7], MATLAB(제어기 설계) 등의 다양한 언어들 사이에 인터페이스가 이루어졌다.

2. 원전 시뮬레이터(NPPS)

2.1 시뮬레이터 주요 특성

NPPS는 포트란언어, Visual C++언어 및 MATLAB으로 구성되어 있으며, 상호 인터페이스 되어 있다. 소스프로그램은 2개의 실행프로그램과 3개의 입력데이터로 구성되어 있고, 주 실행프로그램(NPPS)는 39개의 포트란 소스프로그램과 23개의 Visual C++ 소스 프로그램 및 4개의 MATLAB(고등제어 알고리즘) 파일로 구성되어 있다. 입력파일 생성 프로그램(ssi)는 새로운 발전소 조건에 대한 입력파일을 만들 때 쓰이며, 필요한 입력파일들은 이미 주어져 있기 때문에 일상 시뮬레이션을 수행하는 데에는 필요하지 않다. 그리고 초기 운전 상태를 모의하여 NPPS의 입력파일로 만드는 실행프로그램으로 6개의 포트란 소스프로그램(ssi.f, ctrl.f, lib.f, rcs.f, trns.f, init,f)으로 구성되어 있다. 입력데이터는 다음과 같이 구성되어 있다.

- Data.in : 고리 2호기의 기하학적 특성 등을 기억하고 있는 데이터
- Out1 ~ Out40 : 고리 2호기의 운전상태를 기억하고 있는 데이터
- Rx_input : 일차원 노심 계산을 행할 때 필수적인 중성자 단면적에 대한 정보들과 노심의 초기화에 필요한 정보를 저장하며 노심의 연소 상태에 따라 BOL, MOL, EOL의 세가지로 구분되어 있다.

발전소 모델에서 발전소를 모의하여 그 결과를 Graphic Display Module로 넘기면 모니터의 화면에 발전소의 운전 상태가 표시된다. 이것을 본 사용자가 Interactive Communication Module을 통해서 어떤 조치를 취할 수 있다. 메뉴 구조상 가장 상부 구조에 해당되는 부분으로 "SIMULATION", "SYSTEM", "TREND", "SAFETY Ps", "ALARM_TRIP", "ACCITENTS", "CONTROL DEMO", "DIAGNOSIS", "NETWORK"의 9개의 메뉴가 있으며 그 밑에는 하부메뉴가 계층 구조로 되어 있다. 본 시뮬레이터를 실행하여 시뮬레이션을 시작하기 전에 필요한 옵션을 선택하게 되는데 이미 사용자를 위해 만들어진 입력 파일은 10 ~ 100% 출력, BOL, MOL, EOL에 30개의 파일이 존재한다. Default 값으로 BOL 100% 출력으로 선택되어 있으며, 이중 사용자가 원하는 출력을 선택할 수 있도록 되어 있다. 그리고 Trend 곡선을 보고 싶은 시간 간격은 200 ~ 1000초까지 임의로 선택할 수 있고, 사용자가 원하는 시간 간격을 입력할 수 있도록 되어 있으며, default 값으로 600초가 선택되어 있다. 그리고 노심 모델을 점운동방정식 혹은 1차원 확산 모델 중 하나를 선택할 수 있으며 default 값은 점운동방정식이다. 그리고 Timer는 실시간 계산 여부를 확인한다. NPPS는 Real-Time

Mode와 Fast-Time Mode의 2가지 작동 양식이 있다. Real-Time Mode에서는 실제 발전소에서 사건이 진행되는 시간과 모의하는데 걸리는 시간, 즉 계산되어 모니터에 출력되는 시간이 같은 경우를 말하며 1초마다 계산 결과를 모니터 화면에 디스플레이한다. Fast-Time Mode에서는 1초 간격의 모의 시간을 매 0.1초마다 계산 결과를 모니터 화면에 디스플레이하므로써, 실시간보다 10배 빨리 모의할 수가 있는데, default 값으로 Real-Time Mode로 선택되어 있다.

NPPS에서 추가적인 옵션 중에 하나는 네트워크 통신을 통해서 분산 시스템 환경 기능을 가지고 있다. 내장된 임의의 어플리케이션을 수행할 수 있는 Off-Line Mode와 네트워크에 연결된 외부 Control Station에 내장된 임의의 어플리케이션을 수행할 수 있는 On-Line Mode가 있다. 따라서 서버와 클라이언트간의 통신을 통한 임의의 어플리케이션을 수행할 수 있도록 되어 있다. 발전소 모델과 입력 데이터는 BTU(British Thermal Unit)를 사용했으나, Man-Machine Interaction Model(Graphic Display Module과 Interactive Communication Module)에서는 SI 단위를 사용했다. 따라서 사용자는 SI 단위로 발전소의 운전 상태를 파악하게 된다.

2.2 NPPS의 주요 메뉴

(1) 핵증기 공급계통(NSSS)

원자력발전소 시스템 계통 중 핵증기 공급 계통을 개략화하여 표현했는데 그림 1에서와 같이 주요 구성 시스템은 원자로, 원자로 냉각펌프(RCP), 증기발생기, 가압기 등이 있으며 각 시스템은 냉각재관으로 연결되어 있다. 본 계통의 디스플레이는 고리 2호기 원자력발전소의 도면[8]을 참조하여 그렸으며, 2차 계통의 일부를 포함할 만큼 매우 광범위하다. 이렇게 한 이유는 운전자가 이 화면을 통해 거의 모든 발전소 정보를 한눈에 쉽게 파악하도록 하며 현재 벌어지고 있는 상황을 올바로 판단하는데 도움을 주기 위해서다. 그러므로 이 화면 하나만 가지고도 다른 시스템을 보지 않고 상황 판단을 할 수 있기 때문에 시스템 계통의 대표적 화면으로 선택하였다.

(2) 원자로 냉각재 계통(RCS)

원자력발전소 시스템 계통 중 원자로 냉각재 계통에 관련된 사항을 사용자가 알기 쉽도록 개략화하여 그림 2에서와 같이 일목 요원하게 표현하였으며, 주요한 시스템은 원자로, 증기발생기, 냉각재 펌프로 구성되어 있다. 각 시스템은 관을 나타내는 선으로 연결되어 있다. 원자로 냉각재 계통은 핵증기 공급 계통 중에서 냉각재 계통을 좀더 자세히 표현한 것으로 냉각재가 유입, 방출되는 양과 그에 연관된 물리변수가 디스플레이 된다. 그러므로 이 계통은 NSSS에 보조적으로 사용될 수 있다.

(3) 가압기와 방출 탱크 계통(PRZ)

발전소 시스템 계통 중 가압기와 방출탱크 계통을 그림 3에서와 같이 사용자가 한눈에 알아볼 수 있도록 실제 발전소 도면을 참고하여 개략화 했다. 시스템은 크게 가압기와 방출 탱크로 구성되며, 이를 연결하는 관들과 밸브들로 이루어져 있으며, 시뮬레이션 되는 부분만 밸브의 개폐나 물리변수의 값이 디스플레이 된다. 가압기 주변 밸브상태를 나타내는 실제 고리 2호기 발전소 밸브 개수와 모델링된 결과가 다르므로 이를 극복하기 위해서 PORV 두개 밸브 중 50% 이하로 열린 경우는 그림 아래쪽 PORV를 연 것으로 표시하고, 50% 이상 열렸을 경우 두개가 다 열리도록 해두었다. 이는 포트란 프로그램에서 시뮬레이션시 넘어오는 값에 따라 결정되며, 안전밸브와 스프레이 밸브는 두개가 동시에 열리는 것으로 화면상에 나타낸다.

(4) 핵증기 공급 계통 기포비(VOID FRAC)

포트란 프로그램에서는 핵증기 공급 계통 기포비를 시뮬레이션하기 위해 그림 4에서와 같이

몇 개의 노드(node)로 나누었는데, 사고시 각 노드에서 기포비가 주요한 관심사이므로, 이를 각 노드에서의 색깔로 표시해 줌으로써 사용자가 쉽게 인식할 수 있도록 했다. 노드를 나눈 형태로 핵증기 공급 계통을 세분하여, 시뮬레이션 결과 넘어오는 기포비를 색깔별로 표시해 준다. 이는 정상상태에서는 일정하므로 변화를 볼 수 없고, LOCA와 같은 냉각재 유출 사고시 시간에 따라 기포비를 각 노드에서 관찰할 수 있다. 그리고 정상상태에서는 가압기를 제외한 부분의 각 노드의 기포비는 0이고, 가압기에는 두 개의 노드로 나누어져 있으나, 여기서 색깔로 표시할 때는 두개의 평균값을 취해 디스플레이하게 된다.

(5) ECCS, RHRS 계통

실제 발전소의 ECCS, RHR 계통을 개략화하여 그림 5에서와 같이 시뮬레이션에 맞게 계통을 그렸다. 그러므로 실제 발전소 도면과는 아주 미세한 차이가 있으나 중요한 계통들은 모두 포함시켰다. 이 화면은 ECCS 계통을 이해하는 계통도로도 사용할 수 있고 각 밸브의 상태, 유량, 펌프상태가 표시되므로, 시뮬레이션시 이 계통의 변화를 점검할 수 있다. 구성은 Accumulator, RWST, RHR 펌프, SI 펌프, 열교환기, 각종 밸브, 이를 연결하는 선으로 이루어져 있다.

(6) OPdT, OTdT 마진

안전성에 관련된 주요 변수의 하나인 OP T, OT T의 Trip 마진을 그림 6에서와 같이 시각적으로 표시함으로써 운전자에게 발전소 제어에 대한 정보를 제공하고 운전에 따른 마진의 변화를 측정할 수 있게 한다. 시뮬레이션 되어 넘어오는 T와 OP T, OT T 설정치를 화면상의 좌표로 전환하여 각 루프의 정성적인 값으로 디스플레이 해주는데, 마진은 각 설정치를 각 루프의 T로 나누어 백분율로 환산된 값으로, T 값이 100%로 기준이 되고 100%를 넘는 부분이 trip 마진이 된다.

(7) 2차 계통(BOP)

NSSS에서 2차 계통은 대충 증기관 부분만 그려졌으므로 이를 상세히 나타내주는 도면이 필요하다. 이를 위해서 2차 계통을 개략화하여 따로 그림 7에서와 같이 그렸다. 그러나 2차 계통은 터빈 이후 부분과 급수 계통 부분이 시뮬레이션이 안되므로 이 부분은 생략되었다. 그래서 구조는 증기발생기, 증기관, 터빈, 급수 계통 일부, 각종 밸브들로만 되어 있다. 밸브의 상태 및 각종 물리변수 값이 정량적으로 디스플레이 된다. 여기서 2차 계통을 보고 있을 때 1차 계통의 중요변수인 출력과 가압기 압력을 볼 수 없기 때문에 이를 참조하도록 덧붙였다. 이중 급수 계통의 밸브는 급수 밸브와 bypass 밸브의 2개로 구성되는데, 이는 터빈의 출력에 따라 조정된다. 즉, 출력이 15% 이상일 때는 급수 밸브만 열리고, 15% 이하일 때는 bypass 밸브만 열리도록 되어 있다. 따라서 이 제어논리에 따라 밸브의 개폐 상태가 바뀐다.

(8) 증기발생기 계통(S/G)

시스템 계통 중 증기발생기와 연관되어 있는 사항을 모아서 그림 8에서와 같이 나타내준다. 증기발생기 광역, 협역 수위와 증기 계통, 급수 계통의 중요한 변수 값과 고온관, 저온관의 상태를 포함하는데 증기발생기 수위를 제외한 모든 동적 디스플레이는 정량적인 값을 나타낸다. 동적 디스플레이는 물리변수 중 수위, 온도, 유량, 압력 등을 포트란 프로그램에서 시뮬레이션된 결과를 받아 디스플레이 해준다. 이 중 수위는 광역 수위를 증기발생기 내부에, 협역 수위는 증기발생기 옆에 정량적인 값과 정성적인 값을 동시에 나타내주었다.

(9) 추이곡선(TREND)

발전소 동향을 파악하는데 꼭 필요하다고 생각되는 변수들의 시간별 자료를 표현하는 방법이 필요하다. 이는 시간에 대한 물리량의 변화를 그래프로 나타내주는 방법이 일반적이므로 이 방법

으로 변수의 추이를 알 수 있도록 했다. 발전소 상황을 판단하는데 필요하거나 중요한 의미를 가지는 물리변수만을 선정하여 추이곡선을 디스플레이하도록 했다. 이러한 변수들을 표현하기 위해서 한 화면을 4등분하여 추이곡선 틀을 만들고 그 내부에 여러 변수를 색깔로 구분해 디스플레이 했다. 우선 디스플레이할 변수는 시스템 별로 크게 나누고 선정된 변수의 단위가 같은 것끼리 묶어 디스플레이가 편하도록 했다. 여기서 나누는데 기준이 되는 단위는 온도(℃), 압력(bar), 유량(% 또는 ton/h)이며, 시스템은 크게 원자로, 증기발생기, 가압기, 냉각재 시스템, 증기 시스템, 급수 시스템의 6가지로 크게 나누었다.

(10) ACCIDENT

시뮬레이션 도중 사고를 일으키기 위해 만들어진 메뉴로서 메뉴 항목에 따라 즉, 냉각재 유출사고(LOCA), 증기관 파열사고(SL BREAK), U 튜브 파열사고(S/G TUBE), 주급수관 파열사고(FL BREAK), 원자로 Trip, 터빈 Trip, Trip bypass, Instrument Fail, 냉각 펌프 등으로 구성되어 있다. 냉각펌프(RCP)를 제외하곤 사고의 위치와 크기를 입력하거나 슬라이더로 조절하게 되어 있어서 한번 일으키고 난 후 다시 일으킬 수 없다는 점이다. Instrument Fail 메뉴는 센서의 고장을 모의하기 위한 메뉴로서 고장 모드는 High Stuck, Low Stuck 두 경우만을 가정한다. 이 메뉴를 선택하게 되면 17가지 중요 파라메타에 대한 고장 선택을 할 수 있는 윈도우가 나타난다. 여기서 17개의 변수 이름 오른쪽에는 '1', '2', … 등의 숫자가 보이는데 이는 그 변수를 측정하는 센서를 나타내며 선택하도록 되어 있다.

(11) CONTROL DEMO

이 메뉴는 고등제어기를 설계하고 검증 구현하는 것을 쉽게 하기 위하여 만들어졌다. 하부메뉴로 SG Level Control과 Power Level Control으로 구성되어 있다. 여기에서 SG Level Control에는 PI Control과 Advanced Control로 구성되어 있으며, 사용자가 Advanced Control를 선택하여 수위 설정치 변경에 따른 제어기의 응답을 관찰할 수가 있다.

(12) NETWORK

실시간 시스템은 다양한 분야에 대한 그 필요성 확대와 범용화로 대형 소프트웨어의 설계 및 재사용을 위한 객체지향성과 컴퓨팅 파워의 향상을 위한 분산 시스템 환경을 요구하게 되었다. 분산 시스템 환경(Distributed System Environment)이란 하나의 프로세서에서 중앙 집중적으로 수행되던 시스템을 여러 개의 프로세서로 모듈화 하여, 네트워크를 통하여 연결된 여러 대의 프로세서에 나누어서 동시에 수행하는 것을 의미한다. NPPS는 각 시스템을 연결하는 프로토콜로는 TCP/IP가 사용되었고, 인터페이스를 위해 소켓(Socket) 프로그램을 이용하였다. 그림 9는 서버와 클라이언트 간의 통신상태를 나타낸다. NPPS는 실행함과 동시에 클라이언트들과 통신을 수행할 수 있도록 서버를 오픈하게 된다. 여기에서 클라이언트는 제어기 설계 어플리케이션을 위한 것이다. 서버에서는 클라이언트에게 수행하는데 필요한 데이터를 보내주면 클라이언트는 받은 데이터를 계산하여 서버에게 수행된 데이터를 보내게 된다.

2.3 시뮬레이터 인터페이스

(1) Visual C++과 포트란과의 인터페이스

NPPS에서는 포트란 프로그램과 Visual C++ 프로그램으로 구성되어 있는데, 그 중에서 포트란 프로그램은 발전소를 모의하고 그 결과를 Visual C++ 프로그램에 전달하는 역할을 한다. 또한 사용자의 조치를 Visual C++ 프로그램에서 읽고 그와 관련된 정보를 포트란 프로그램에서 받기도 한다. 이 시뮬레이터에서 사용하는 포트란 프로그램은 7개의 패키지(package)로 구성되어 있고, 각 패키지는 여러 개의 부 프로그램이나 함수를 포함하고 있다. 각각의 부 프로그램과 함수는 가능한

작게 만들었다. 이것은 개발과 보존을 용이하게 할 뿐만 아니라 프로그램의 수정에도 편리한 점이 있기 때문이다. 프로그램간의 정보전달 방법으로 포트란 프로그램에서는 주로 라벨(label)이 있는 common문을 이용하였다. 여기에서 주의해야 할 것은 포트란에서는 대문자와 소문자가 구분되지 않기 때문에, 포트란 common문의 라벨을 Visual C++ 프로그램에서 구조체명의 정의시 혹은 변수 호출시 대문자로 표기해야 한다는 것이다.

포트란과의 정보교환은 포트란 프로그램인 "fstart()", "fsimul()"을 통해서만 이루어진다. 따라서 부 프로그램 fstart와 fsimul의 포트란 프로그램은 계층 구조상 가장 상부에 속해 있으며, 다른 부 프로그램을 호출해서 고유기능을 수행한다. 부 프로그램 fstat는 발전소를 모의하기 전에 입력데이터를 읽어 들이고, 계산에 필요한 여러가지 변수의 초기치를 설정하며, 부 프로그램 fsimul은 발전소를 모의하고 그 결과를 Visual C++ 프로그램으로 전달한다.

따라서 이 두함수에서 common문으로 선언된 부분을 Visual C++의 헤더 파일안에 extern문 구조체 배열을 사용하여 시뮬레이션에 해당되는 변수들을 부르는 방법을 설명하면 다음과 같다.

- 포트란 프로그램 "sun.f"파일에서 common문으로 선언된 부분

```
      subroutine fstart
 … 이하 생략
      common /almi/ itrip_alm(100),ialarm_a(100),ialarm_b(100)
                    ,ialarm_c(100),ialarm_d(100),ialarm_e(100)
 …이하 생략
```

- Visual C++ 프로그램에서 헤더 파일에 선언된 부분

```
extern "C" void FSTART()
extern "C" void FSIMUL()

extern "C" struct{
     int itrip_alm[100], ialarm_a[100], ialarm_b[100]
     int ialarm_c[100], ialarm_d[100], ialarm_e[100]
} ALMI
…이하 생략
```

(2) Visual C++과 MATLAB과의 인터페이스

NPPS에서는 증기발생기 수위(급수)를 제어하기 위한 MATLAB과 Visual C++사이의 데이터를 주고받기 위해서 사용되었다. MATLAB은 자체의 내장 인터프리터에 의하여 명령과 함수를 해석하여 실행이 이루어진다. 그래서 위의 컴파일러와 링커를 사용해서 생성된 외부 함수는 특정 형식을 가지게 된다. Visual C++에서 MATLAB으로 만들어진 고등제어 알고리즘을 사용하기 위해서 MATLAB External Interface Library를 이용해서 Visual C++에서 호출할 수 있도록 Mex-File을 사용하였다.

Mex-File이란 MATLAB Interpreter가 자동으로 load하고 MATLAB 환경 내에서 실행할 수 있는 Dynamically Linked Subroutine이다. 즉, mex.m 함수로 C언어나 FORTRAN source code를 MATLAB에서 사용할 수 있거나, 반대로 사용할 수 있도록 Compile하면, Windows 환경에서는 확장자가 *.dll인 것으로 바뀐다. DLL(Dynamic Link Library) 형식은 Widows 환경에서 사용하는 표준 16-bit DLL을 가리키는 것으로서 일반적인 Windows DLL 파일을 만드는 방법과 똑같은 과정을 거치게 된다.

위와 같이 Visual C++에서 MATLAB을 사용하기 위해서는 MATLAB에서 "mex –setup"을 이용하

여 Complier를 설정해주면 된다. 그리고 Visual C++에서 "MATLAB Add-in Project Setup"에서 "M-MEX DLL"을 사용하여 설계된 MATLAB파일을 불러들이면 관련된 *.c파일 및 *.h파일이 생성되고, 관련된 *.c 파일 및 *.m 파일들은 Visual C++ 프로젝트에 추가가 된다. 생성된 파일 중에서 메인 M-파일의 헤더 파일을 링크하여 사용하면 된다. 그리고 NPPS에서 MATLAB과 Visual C++사이의 데이터를 주고 받기 위해서 작성된 소스부분은 다음과 같다.

```
void CMainFrame::SgCtrl(int in_flag)
{
        mxArray *init_flag;          // initial flag (in_flag)
        mxArray *ii;                 // loops A or B
        mxArray *x1_ptr;             // water level
        mxArray *x2_ptr;             // steam flowrate
        mxArray *x3_ptr;             // desired water level
        mxArray *x4_ptr;             // ictrl_flag[0]
        mxArray *y_ptr;              // feedwater flowrate

    /* Create an mxArray to input into mlfSgCtrl */

        for (int i = 0; i < 2; ++i)
        {

                init_flag = mlfScalar(in_flag);

                ii = mlfScalar(i);
                x1_ptr = mlfScalar(CTRL.wlevel[i]);
                if(in_flag == 0) {
                        x2_ptr = mlfScalar(BOPCON.wfwo);
                }
                else {
                        x2_ptr = mlfScalar(CTRL.stflow[i]);
                }
                x3_ptr = mlfScalar(CTRL.dlevel);
                x4_ptr = mlfScalar(CTRL.ictrl_flag[0]);

                y_ptr = mlfSg_ctrl(init_flag, ii, x1_ptr, x2_ptr, x3_ptr, x4_ptr);

                // The return value from mlfSgCtrl is an mxArray so we must extract the data from it
                y = mxGetPr(y_ptr);

                CTRL.fwflow[i] = *y;

        }
}
```

그리고 MATLAB에 있는 매개변수들은 Visual C++에서 mxArray로 선언된 변수로 사용할 수 있다. mxArray는 MATLAB에서 제공되는 매개변수에 대한 다음과 같은 정보를 저장하는 Structure이다.

- 매개변수에서 사용되는 데이터 type의 종류
- 매개변수의 Dimensions
- 매개변수에 주어진 실질적인 데이터
- 만일 매개변수가 수치(numeric) Data이면, 실수인지 허수인지에 대한 정보

mlfScalar 함수는 입력 매개변수로 주어진 mxArray에 대한 1-by-1 배열의 형태로 매개변수 double data값이 입출력된다. 사용방법은 "mxArray *mlfScalar(double v)"와 같은 형태로 사용하는데, double v값은 1-by-1 배열에 입력되어 사용된다. 그리고, mxGetPr 함수는 입력 매개변수로 주어진 mxArray에 대한 real data의 starting address를 돌려준다. 사용방법은 "double *mxGetPr(const mxArray

*array_ptr)와 같은 형태로 사용하는데, 여기서 array_ptr는 mxArray type의 pointer 변수이다. mxGetPr 함수가 성공적으로 작동하면, array_ptr이 가리키는 매개변수의 첫번째 real data의 address를 돌려준다. 그러나, array_ptr이 가리키는 매개변수가 real data를 가지고 있지 않으면, NULL을 돌려준다.

3. 시뮬레이터를 이용한 고등제어기 설계 및 구현

3.1 고등제어기 설계

본 연구에서는 이동구간제어방법(Receding Horizon Control Method)을 이용하여 구현하였으므로 이 방법에 대하여 간단히 소개하기로 한다.

RHC는 최적제어, Stochastic 제어, 시간지연을 갖는 공정제어 및 다변수 제어 등을 통합한다 [9-14]. RHC의 또 다른 장점은 일정 제어범위(Finite Control Horizon)가 이용되므로 인하여 산업현장에서 흔히 존재하는 제한조건이나 비선형 공정 등에 적절히 적용될 수 있다. 이런 RHC의 특성으로 인하여 원자력발전소 증기발생기의 수위제어에 적절히 응용될 수 있을 것으로 판단된다. RHC의 기본 개념은 현재 시간에 미래의 일정 구간에 대한 최적화를 통한 제어입력을 구한 연후에 현재의 제어입력으로 단지 첫번째 제어입력 만을 구현하는 것이다. RHC의 기본 요소는 미래의 한 시점의 프로세스 출력을 예측하기 위한 명확한 모델 (예측모델), 목적 함수 그리고 제어법칙을 위한 최적화 등이 포함된다. 예측모델로서 초기에는 FIR(Finite Impulse Response)모델과 FSR(Finite Step Response)모델 등이 사용되었으나, 전달함수 모델, 상태공간 모델 그리고 비선형 모델 등도 이용될 수 있다. 이 예측모델을 이용하여 예측구간 N 을 위한 출력값이 예측되며, 예측된 출력값들은 과거의 입출력값 및 미래의 제어신호에 의존한다. 출력값이 가급적 기준 출력값을 따르도록 주어진 목적함수를 최적화함으로써 일련의 최적화된 제어신호가 계산된다.

목적함수가 이차함수(Quadratic) 형태이고, 모델이 선형이며, 제한조건이 존재하지 않는 경우는 제어입력은 분석적으로 구해질 수 있으나, 대부분의 프로세스에서 실제 제어입력은 수치적(Numerically)으로 얻어진다. 이때 시스템에 가해지는 제한조건들을 알고리즘에 포함시킴으로써 제한조건들을 만족시키는 범위에서 최적의 제어입력이 구해진다. 최적화된 제어신호 중에서 단지 첫번째 입력만 프로세스 입력에 보내진다. 그리고 다음 샘플주기에서는 $\mathbf{y}(t+1)$ 은 이미 알려졌기 때문에 앞 샘플주기마다 반복적으로 하여 제어입력이 계산된다. 이와 같은 과정이 그림 10에 주어져 있으며, RHC 알고리즘을 도시화하여 나타내면 그림 11과 같다[11].

다음의 시불변 이산계통을 고려하자:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\Delta u_{uv}(k), \\ y(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned} \tag{1}$$

여기서

$\Delta u_{ux}(k) = \Delta u(k) - \Delta v(k)$

$x(k) \in R^n$; 상태벡터

$\Delta u(k) = u(k) - u(k-1)$: 제어입력 (급수 유량)

$\Delta v(k) = v(k) - v(k-1)$: 측정 가능한 외란 (증기 유량)

$y(k)$: 프로세스 출력 (증기발생기 수위).

식(1)에서 제어입력 자체 $u(k)$ 가 아닌, 제어입력의 변화량 $\Delta u(k)$ 가 옵셋 에러를 제거하기 위하여 이용되었다. 관련 성능지수는 다음의 이차 함수로 주어진다:

$$J = \frac{1}{2}\sum_{J=0}^{N-1}\left[Q(\mathbf{Cx}(k+j) - r(k+j)^2 + m\Delta u_{uv}(k+j)^2\right] + \frac{1}{2}Q_F(\mathbf{Cx}(k+N) - r(k+N))^2, \tag{2}$$

여기서 Q (Positive Semi-Definite), Q_F (Positive Semi-Definite) 그리고 m (Positive Definite)는 어떤 미래 시간 구간에서 $(y-r)$ 와 Δu 의 대칭 행렬의 가중치들이며, r 은 기준 입력(목표 입력)이다. 일반적인 최적 추적제어에서는 문제(2)를 풀어 얻어진 $u(k+j)$, $j \in [0, N]$가 시간 $[k, k+N]$ 구간 동안 시스템을 제어하는데 사용되고, 시간 $k+N$ 에서는 위의 과정이 구간 $[k+N, k+N+N]$에 대하여 반복된다.

그러나 RHC제어의 개념은 $\Delta u(k+M+1) = \Lambda = \Delta u(k+N) = 0$ 을 가정하여 시간 k 에서 비용지수 식(2)를 최소화하는 최적화 문제를 풀어 얻어진 제어입력 $\Delta u(k), \Delta u(k+1), \Lambda, \Delta u(M-1)$ 을 구하는 것이다.

제한조건이 존재하는 경우에 적용되는 강력한 도구인 Lagrange-multiplier 기법을 이용하여 제어 입력을 구하면 다음과 같다:

$$\begin{aligned}\Delta u_{uv}(k+j) &= \left[m + \mathbf{B}^T\mathbf{F}(j+1)\mathbf{B}\right]^{-1}\mathbf{B}^T\left[\mathbf{g}(k+j+1) - \mathbf{F}(j+1)\mathbf{Ax}(k+j)\right] \\ &= -\mathbf{K}(j)\mathbf{x}(k+j) + \mathbf{K}_g(j)\mathbf{g}(k+j+1), \quad j = 0, 1, \Lambda, M-1\end{aligned} \tag{3}$$

여기서

$$\mathbf{F}(j) = \mathbf{C}^T Q\mathbf{C} + \mathbf{A}^T\mathbf{F}(j+1)\left[\mathbf{A} - \mathbf{BK}(j)\right], \tag{4}$$

$$\mathbf{g}(k+j) = \begin{cases}\mathbf{C}^T Qr(k+j) + \left[\mathbf{A} - \mathbf{BK}(j)\right]^T\mathbf{g}(k+j+1) & for\ \ j \le M-1 \\ \mathbf{A}^T\mathbf{g}(k+j+1) + \mathbf{C}^T Qr(k+j) & for\ \ M \le j \le N-1\end{cases} \tag{5}$$

$$\mathbf{F}(N) = \mathbf{C}^T Q_F\mathbf{C}$$

$$\mathbf{g}(k+j) = \mathbf{C}^T Q_F r(k+N)$$

참고로 $\Delta u_{uv}(k+j) = 0$, $j = M, \Lambda, N$ 을 아는 것이 중요하다. $\mathbf{K}(j)$ 는 기준입력에 의존하지 않기 때문에 위와 같이 산술적 차분 방정식을 통하여 Off-Line으로 쉽게 구해진다. 즉 제어할 시스템과 성능지수에 포함된 설계변수들 (m, Q, Q_F) 그리고 예측구간 N 이 결정되면 이들 이득 값은 완전히 결정된다. 그러나 식(3)에서 마지막 항과 관련된 이득 $\mathbf{K}_g(j)$ 는 현재 시간 k 에서 $[k, k+N]$사이에서 알 수 있다고 가정한 미래의 기준입력 및 측정 가능한 외란, 그리고 $\mathbf{K}(j)$ 를 이용하여 매시간마다 On-Line으로 계산해야 한다. 그리고 식(3)에서 단지 첫번째 제어 입력이 구현되기 때문에, RHC 제어기의 제어 입력은 다음과 같다:

$$\Delta u_{uv}(k) = -\mathbf{K}(0)\mathbf{x}(k) + \mathbf{K}_g(0)\mathbf{g}(k+1) \tag{6}$$

식 (6)에서 $\mathbf{K}(0)$ 와 $\mathbf{K}_g(0)$ 는 시불변 계통을 위해 상수이나, $\mathbf{g}(k+1)$ 은 출력 설정치에 의존하기 때문에 매 주기마다 다시 계산되어야 한다. $\mathbf{x}(k)$ 는 측정되는 변수가 아니며, 모델링의 부정확성에 의하여 정확히 알려지지 않기 때문에 본 연구에서는 잘 알려진 칼만 필터를 이용하여 상태를 관측하였다. 설계된 RHC 제어기를 블록 다이어그램으로 나타내면 그림 12과 같이 표현될 수 있다.

3.2 데모구현

원전 시뮬레이터에서 이용된 비선형 모델은 본 제어기를 설계하는데 직접 이용될 수 없기 때

문에, 증기발생기의 모든 출력 시뮬레이션에서 다음의 선형모델의 100% 출력에 해당하는 제어기 설계모델을 이용하여 RHC 제어기가 설계되었다:

$$Y(s) = \frac{G_1}{s}[U(s) - V(s)] - \frac{G_2}{1 + t_2 s}[U(s) - V(s)] + \frac{G_3 s}{t_1^{-2} + 4p^2 T^{-2} + 2t_1^{-1} s + s^2} U(s) \qquad (7)$$

여기서 $Y(s)$, $U(s)$ 및 $V(s)$ 는 증기발생기 수위, 급수유량 및 증기유량의 라플라스 변환된 값을 나타낸다.

앞서 설명한 제어기를 검증 및 비교를 위해 시뮬레이터에 기본으로 내장된 비례적분 제어기가 이용되었다. 다음과 같은 2가지 경우에 대한 데모를 수행하였다:

1) 동일 컴퓨터내 이동구간제어기 적용 (Off-Line) : 동일 컴퓨터내 같은 프로젝트 파일안에 고등제어 알고리즘을 MATLAB으로 구현하여 MATLAB 제어기 알고리즘과 Visual C++ 및 FORTRAN 프로그램의 시뮬레이터 사이에 인터페이스가 이루어져 수행이 되는 경우를 일컫는다.

2) 다른 컴퓨터와의 네트워크 통신에 의한 이동구간제어기 적용 (On-Line) : 서버 컴퓨터(시뮬레이터)와 클라이언트 컴퓨터(고등제어 어플리케이션 내장) 사이에 네트워크 통신에 의한 고등제어기 적용을 검증하기 위함이며, 실제 적용된 예는 동일 컴퓨터내에서 적용한 경우와 똑같은 시나리오를 가정하여 수행하였다(그림 9 참조).

위 두 가지 경우에 대하여 당연히 동일한 결과를 얻어야 하듯이 컴퓨터간 네트워크 통신이 원활히 이루어져 모의가 잘 되는 것을 확인할 수 있었다. 그림 13은 네트워크 통신에 의하여 얻어진 고등 제어기를 적용한 결과를 그림으로 보여주고 있다. 이 그림은 원자로 출력 100%에서 증기발생기 수위를 처음 600초는 50% 수위 설정치를 유지하다가 설정치를 55%로 증가시키는 시나리오에 대하여 고등제어기의 응답을 보여준다. 그림 14는 같은 시나리오에 대하여 PID 제어기의 성능을 보여준다. 원래 고리2호기는 수위 설정치를 출력준위에 상관없이 50%로 일정하게 유지하도록 되어 있으며, 5% 만큼 벗어나면 경보가 울리게 되어 있다. 따라서 고의로 설정치를 55%로 증가시키면 경보가 울리는 것을 확인할 수가 있었다. 그림 15는 원자로 열출력이 30%인 경우의 고등제어기의 응답을 보여준다. 여기에서 고등제어기를 설계할 때는 앞의 100%에서 설계한 증기발생기 수학적 모델과 같은 것을 이용하였다. 그림 16은 PI 제어기에 대한 그림이다. 이 그림에서 보여지듯이 제어가 거의 되지 않는 것을 알 수 있으며, 이 출력에 맞추어 PI 이득치를 다시 잘 조정해야 될 것으로 판단된다.

NPPS는 증기발생기 수위제어기 검증만을 위해 만들어진 것이 아니기 때문에 결과 비교를 그림으로 확연히 비교할 수 없어 아쉬웠지만 고출력에서는 PID 제어와 거의 비슷한 성능을 보이는 것으로 나타났다. 또한 PID 이득값들도 최적화되어 있기 때문에 고등제어 알고리즘이 PID와 비슷한 성능을 갖는 것만으로 성공적이라고 판단할 수 있다. 왜냐하면 고등제어 알고리즘(이동구간제어방법)은 고리2호기 발전소와 아주 다른 Irving 모델[15]을 이용하여 구현되었기 때문이다. 만일 이용된 플랜트 모델을 좀더 한층 정교하게 한다면 훨씬 좋은 결과를 얻을 수 있을 것으로 판단된다.

4. 요약 및 결론

본 연구에서 개발된 원전 시뮬레이터(NPPS)는 FORTRAN, Visual C++ 및 MATLAB 프로그램들을 사용하였다. 따라서 NPPS는 MATLAB 프로그램을 이용한 고등제어 알고리즘을 구현하기 쉽고,

프로그램 상호간 인터페이스가 강화된 시뮬레이터이다. 예로써 NPPS에 고등제어방법의 구현 용이성을 검증·시험하기 위해서, 증기발생기 수위를 제어하기 위하여 설계된 고등제어 알고리즘의 하나인 이동구간제어방법을 시뮬레이터에 구현하였다. 구현이 아주 쉽게 이루어졌으며, 단지 제어 알고리즘의 설계시 흔히 이용되는 MATLAB 언어로 코딩한 후 프로젝트 프로그램에 불러들이는 것으로 완결되었다. 그리고 NPPS는 분산 시스템 환경을 가지고 있어서 중앙집중적으로 수행되던 시스템을 여러 개의 프로세서로 모듈화하여, 네트워크를 통하여 연결된 여러 대의 프로세서에 나누어서 동시에 수행할 수가 있다.

감사의 글

본 연구는 과학기술부 선진기술확보사업의 지원으로 수행되었다.

참고문헌

[1] 이원재 외, Improved Features of MARS 1.4 and Verification, KAERI/TR-1386-99, KAERI, 1999.

[2] K.O. Pasamehmetoglu et al., "TRAC-PF1/MOD2 Theory Manual," LA-12031-M, Vol 1, NUREG/CR-5693, Los Alamos National Laboratory (1992).

[3] K.E. Carlson et al. "RELAP5/MOD3 Manual, "NUREG/CR 5535, EG-2546, EG&G Idaho, Inc (1990).

[4] *MATLAB Ver. 6.1*, Math Work Inc. (2001).

[5] 노희천 외, 고리2호기 마이크로 시뮬레이터용 소프트웨어 개발(사용자 지침서), KRC-90N-J07, KEPCO (1992).

[6] *Compaq Visual Fortran Professional Edition Ver. 6.5*, Compaq Computer Corporation (1999).

[7] 이상엽, *Visual C++ Programming Bible Ver. 6.x*, 영진출판사(2000)

[8] 고리 원자력발전소 2호기 계통도, KEPCO

[9] W. H. Kwon and A. E. Pearson, "A Modified Quadratic Cost Problem and Feedback Stabilization of a Linear System," IEEE Trans. Automatic Control, 22, 838 (1977).

[10] J. Richalet, A. Rault, J. L. Testud, and J. Papon, "Model Predictive Heuristic Control: Applications to Industrial Process," Automatica, 14, 413 (1978).

[11] C. E. Garcia, D. M. Prett, and M. Morari, "Model Predictive Control: Theory and Practice a Survey," Automatica, 25, 335 (1989).

[12] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust Constrained Model Predictive Control Using Linear Matrix Inequality," Automatica, 32, 1361 (1996).

[13] J. W. Lee, W. H. Kwon and J. H. Lee, "Receding Horizon H^{∞} Tracking Control for Time-Varying Discrete Linear Systems," Int. J. Control, 68, 385 (1997).

[14] J. W. Lee, W. H. Kwon, and J. Choi, "On stability of Constrained Receding Horizon Control with Finite Terminal Weighting Matrix," Automatica, 34, 1607 (1998).

[15] E. Irving, C. Miossec, and J. Tassart, "Toward Efficient Full Automatic Operation of the PWR Steam Generator with Level Adaptive Control," BNES, London, Boiler Dynamic and Control in Nuclear Power Stations, pp. 309-329 (1980).

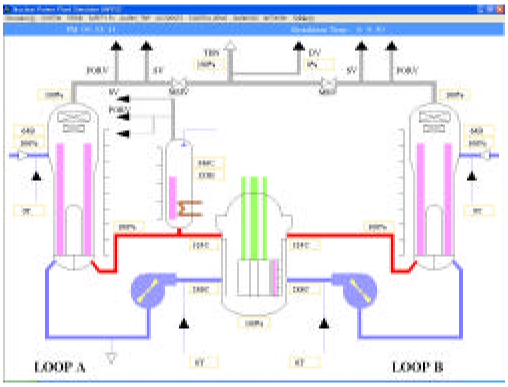


Fig. 1. NSSS.

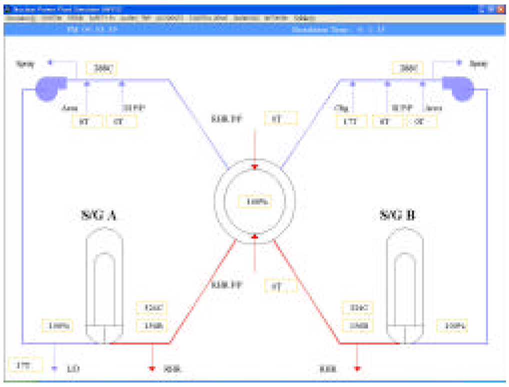


Fig. 2. RCS.

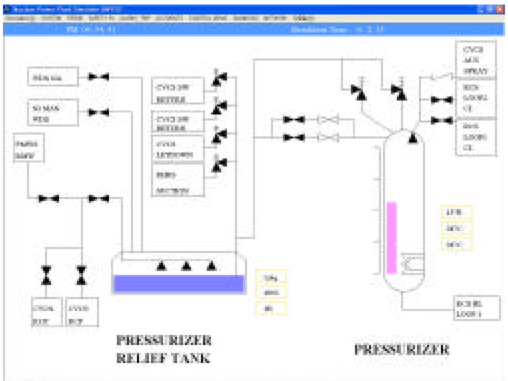


Fig. 3. PRZ and relief tank system.

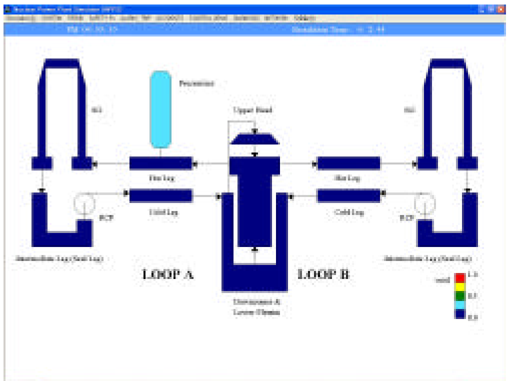


Fig. 4. NSSS void fraction.

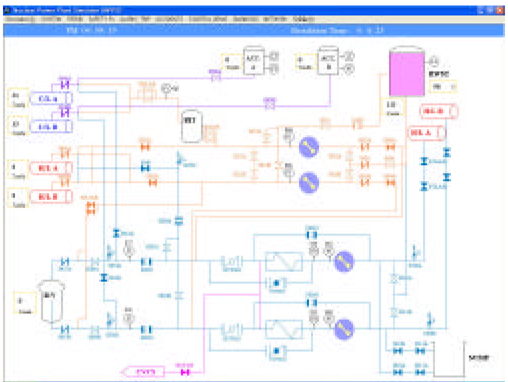


Fig. 5. ECCS and RHRS.

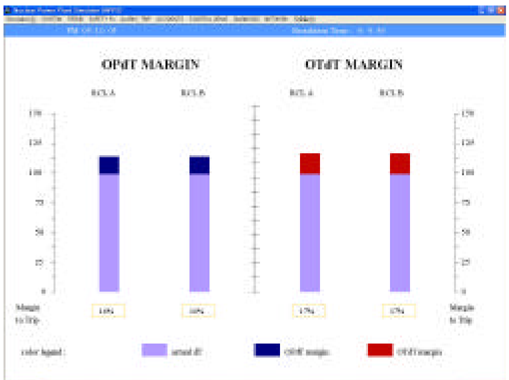


Fig. 6. OP△T and OT△T margin

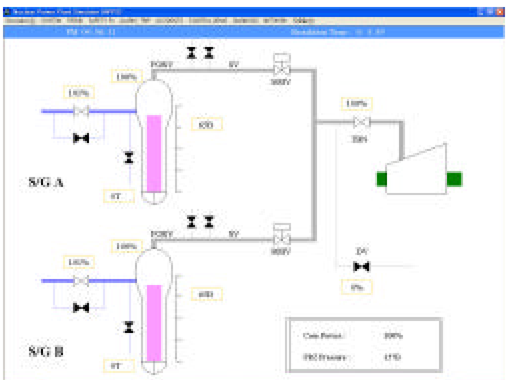


Fig. 7. Secondary system.

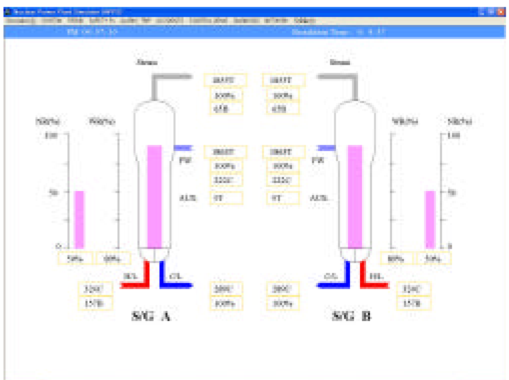
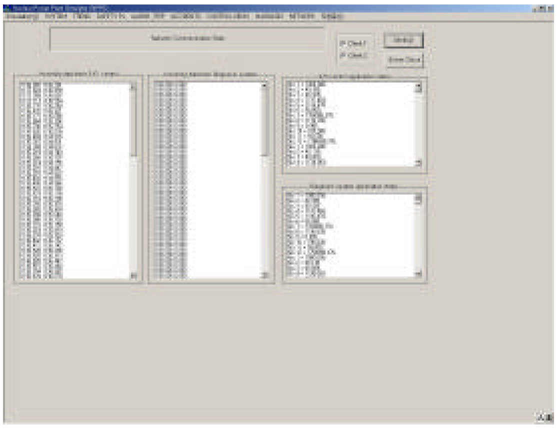
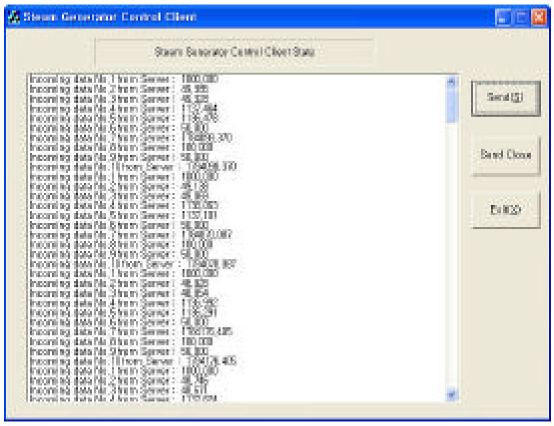


Fig. 8. Steam generator system.



(a) server communication status



(b) client communication status

Fig. 9. Network communication.

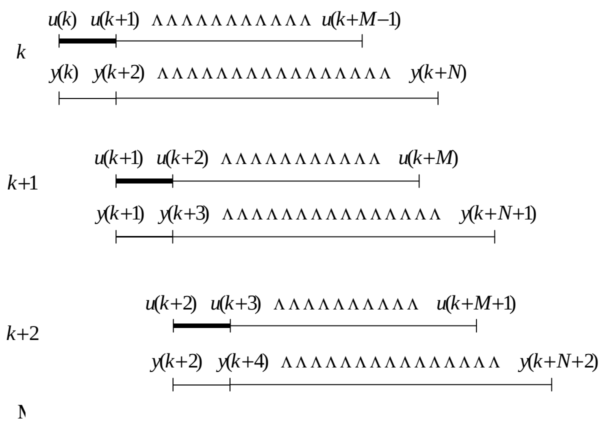


Fig. 10. Optimal solutions and manipulated variables for control implementation.

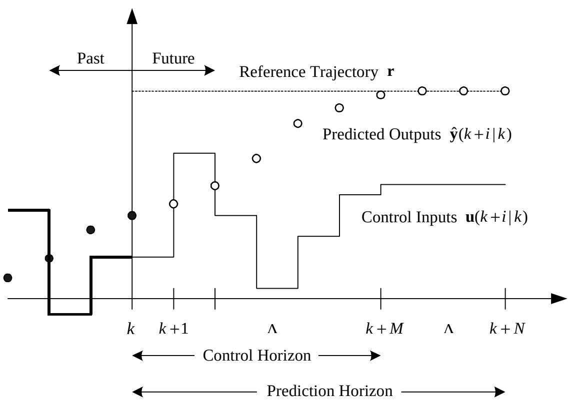


Fig. 11. Basic concept of a receding horizon control method.

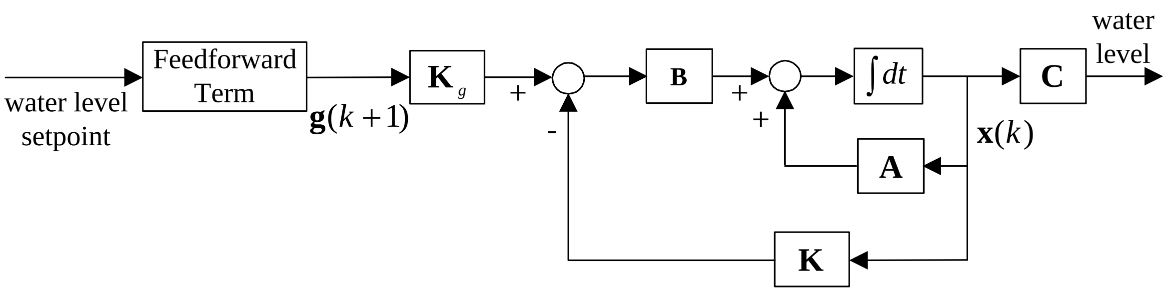
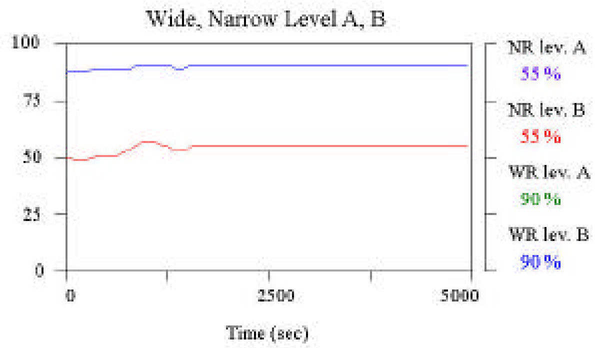
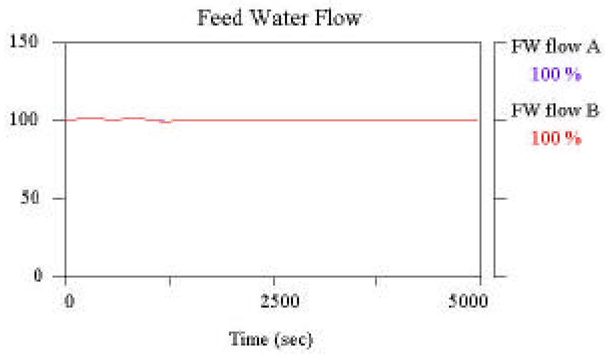


Fig. 12. Block diagram of the designed receding horizon controller.

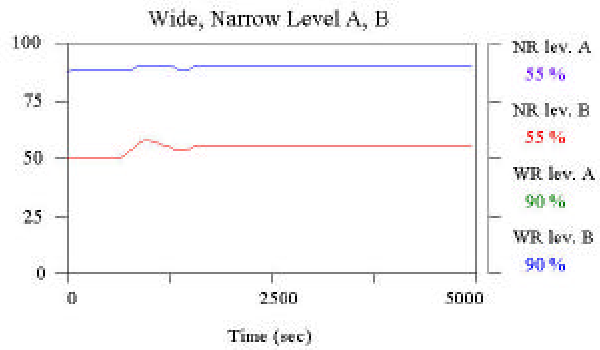


(a) water level

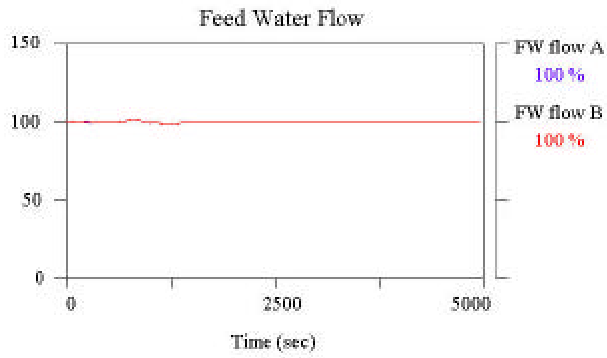


(b) feedwater flowrate

Fig. 13. Demonstration of an advanced controller(high power).

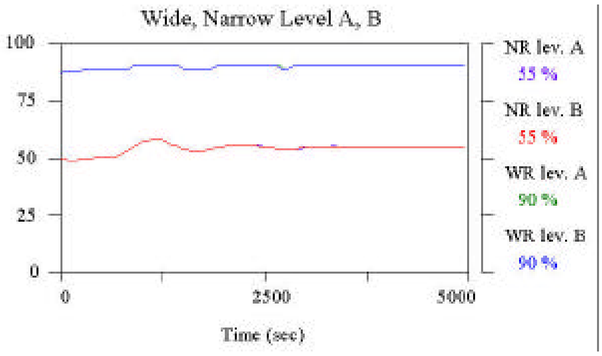


(a) water level

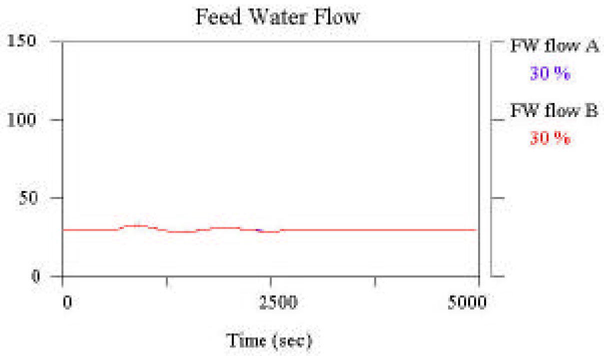


(b) feedwater flowrate

Fig. 14. Demonstration of a PID controller(high power).

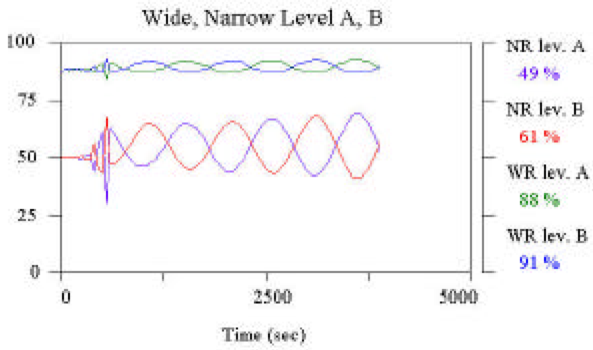


(a) water level

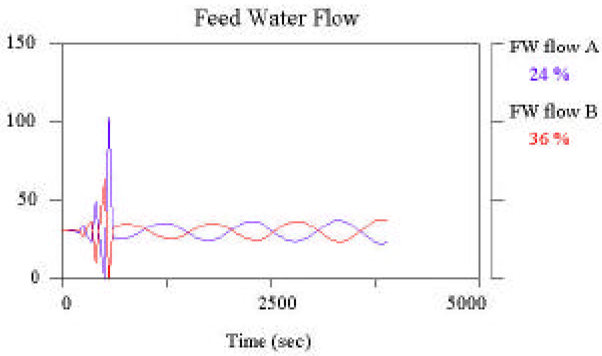


(b) feedwater flowrate

Fig. 15. Demonstration of an advanced controller (low power).



(a) water level



(b) feedwater flowrate

Fig. 16. Demonstration of a PID controller (low power).