HWAHAK KONGHAK Vol. 29, No. 2, April, 1991, pp. 211-221
(Journal of the Korean Institute of Chemical Engineers)

다목적 화학공정 동적모사기 개발
I. 시스템의 일반적 개요 및 모사전략

이강주 · 윤인섭

서울대학교 공과대학 화학공학과
(1990년 9월 22일 접수, 1991년 3월 11일 채택)

Development of General Purpose Dynamic Chemical Process Simulator
I. General Overview of the System and Simulation Strategy

Kang Ju Lee and En Sup Yoon

Dept. of Chem. Eng., College of Eng., Seoul National University
(Received 22 September 1990; accepted 11 March 1991)

요 약

범용 화학공정 동적모사기에 적합한 모사기의 구조, 적분전략, 입력언어, 계산효율 향상을 위한 방법들을 제시하였고, SCI(Sequential Clustered Integration)를 기본으로 하는 원형(prototype) 모사기를 개발하였다. SCI 방식은 SMI(Sequential Modular Integration)의 일반적인 형태인데, SEI(Simultaneous Equation Integration)의 장점을 수용하였기 때문에 SMI나 SEI에 비하여 계산의 효율성과 응용의 유연성 면에서 유리하다. SCI는 단일 모사기 구조 안에서 SMI로부터 SEI에 이르기까지의 다양한 수준의 커플링이 가능하다. 예제를 통하여 SCI의 성능에 많은 영향을 미치는 clustering의 기준에 대하여 연구하였고, 모사기의 성능을 평가하였다.

Abstract—A simulator structure, integration strategy, input language and simulation technique are presented in order to develop a general purpose chemical process dynamic simulator, and a SCI(Sequential Clustered Integration)-based prototype simulator is implemented. The SCI is a general form of SMI(Sequential Modular Integration) but has better performance than SMI and SEI(Simultaneous Equation Integration) with respect to computational efficiency and flexibility of applications. The SCI approach is capable of introducing various mode of coupling of the process units, actually, from SMI to SEI within a single simulator structure. Examples are presented to study the clustering criteria which have significant influence on the performance of SCI and to illustrate the system performance.

1. 서 론

화학공정 동적모사기는 공정의 여러 가지 특성을 연구하는 훌륭한 도구가 되고 있다. 즉, 공정의 안정성(stability) 및 민감도(sensitivity) 해석연구, 제안된 제어 알고리듬의 시험 등에 대한 제어연구, 그리고 운전 시작(start-up), 운전중지(shut-down) 또는 조업전환(switching) 등과 같은 비정상상태 공정의 연구 등에 크게 활용될 수 있다. 최근에는 공정의 가상적인 여러 상황에 대하여 적절한 대응조치를 하기 위한 조업자 훈련용(training simulator)으로 각광을 받고 있으며, 고장 진단용 전문가 시스템 등과 같은 인공지능 시스

템과 직접 또는 간접으로 연결되어 사용되기도 한다.

1960년대 말부터 시작되었던 동적모사에 대한 연구는 정상상태 모사에 비하여 발전속도가 매우 느린 편이었다. 이는 화학공정의 동특성에 대한 이해 부족과, 대규모의 복잡하고 비선형성을 갖는 모델식들을 풀 수 있는 알고리듬의 복잡성 등에 기인하였다. 또한 당시에는 컴퓨터 장비의 가격과 성능의 제약때문에 실제적으로 대규모 화학공정의 동적모사는 매우 어려웠다. 그러나 최근에는 컴퓨터 기술의 급속한 발달과 복잡한 공정모델식들을 풀 수 있는 효과적인 알고리듬의 발전에 힘입어 화학공정 동적모사가 현실적으로 가능해지기 시작하였으며, 이미 상당한 수준의 프로그램이 개발되어 있다.

동적모사가 정상상태모사와 다른 점은 공정상호간의 동적 커플링(dynamic coupling)을 실현하는 점에 있다. 이러한 동적 커플링은 공정의 시간에 따른 변화를 나타내게 되는데, 모델식중의 미분방정식들에 의해서 표현된다. 따라서 동적모사에 있어서 모사전략은 주로 미분방정식들의 적분방식에 따라 분류되는 것이 일반적인데, 크게 다음의 두 가지 방법으로 구분될 수 있다. 먼저 단위공정의 미분식들을 순차적으로 적분해 가는 방식과 전체 공정의 미분식들을 모아서 동시에 적분하는 방식이 그것이다. 본 논문에서는 전자의 방식을 SMI (Sequential Modular Integration), 후자를 SEI(Simultaneous Equation Integration)라고 나타내었다. SMI는 정상상태모사의 순차모듈접근법에 의해 축적된 기술을 그대로 활용할 수 있기 때문에 개발이 비교적 수월하다. 그러나 이 방식은 복잡하게 상호커플링이 되어 있는 공정의 경우에는 해의 정확성과 계산의 효율성 측면에서 문제점을 나타낸다. 발표된 대표적인 프로그램으로는 DYFLOW[1], DYNSYS, DYNSYL[2] 등이 있다. 반면에 SEI에서는 수학적으로 일관된 수식을 사용할 수 있기에 해의 정확성 면에서 우수하고, 설계문제나 최적화문제 등의 응용에도 용이한 장점이 있다. 그러나 이 방식은 개발비용이 많이 들고, 아직까지는 크고 복잡한 방정식들을 다룰 수 있는 완벽한 수치해법적 기술이 부족하다는 면에서 문제점을 가지고 있다. 대표적인 시스템으로는 SPEEDUP[3], ASCEND-II[4, 5], QUASILIN [6], DIVA[7], DASP[8] 등을 들 수 있다. 현재의 추세로는 학계에서 SEI 방식이 선호되고 있지만, 대규모 동적모사를 위해서는 여전히 전자의 방식이 유리하다고 할 수 있다. 특히 병렬처리 컴퓨터를 이용하거나, 네트워크 컴퓨터를 이용하기 위해서는 모듈중심기법이 유리하다고 할 수 있다[9, 10].

이[23]는 범용의 화학공정 동적모사기를 개발하기 위하여 가장 적절한 구조로서 SMI와 SEI의 중간 형태인 계산방법을 도입하였다. 이 방식은 SCI(Sequential-Clustered Integration)라고 불리우며, Fagley[11, 12] 등이 처음 제안하여 계산효율을 입증하였다. SCI에서는 상호작용이 심한 단위공정의 묶음을 하나의 cluster로 정의한 후, 하나의 cluster 안에서는 SEI를, cluster 끼리는 SMI를 사용하는 기법이다. 실제로 이 방식은 cluster의 크기에 따라 SMI로부터 SEI까지의 다양한 방식의 커플링을 허용한다. 결국 이 방식은 가장 일반적인 형태의 적분방식이라고 할 수 있으며, 동적모사에 있어서 화학공정의 다양한 특성을 고려하기에 좋은 방식임에 틀림없다. 그런데 SCI의 성능은 대상공정의 특성이나 모사 목적에 적합한 clustering 구조를 찾는 것에 의존된다. 박 등[22]이 clustering에 대한 기초적인 기준을 설정하였으나, 아직까지 적절한 clustering 기준에 대한 연구는 미흡한 실정이다.

본 논문에서는 SCI 방식에 기본을 둔 범용 화학공정 동적모사에 대하여 연구하였으며, 시험용으로 개발되고 있는 모사기의 전반적인 소개와 모사기법에 대하여 기술하였다. 또한 동적모사의 계산효율과 성능을 높이기 위한 여러 측면의 기법들을 제시하였다. 마지막으로 예제를 통하여 구현된 모사기의 성능을 평가하였으며, clustering 기준에 대하여 연구하였다.

2. 본 론

화학공정을 모델링하면 대규모의 대수식과 미분식의 혼합형 수식을 얻는다. 상미분방정식에 대한 수치해법은 제법 잘 알려져 있고, 활용할 수 있는 코드도 많은 편이다. 그래서 대부분의 범용 동적모사기는 미분식의 형태로써 상미분방정식을 요구한다. 결국 식 (1), (2)와 같은 방정식 시스템의 해를 시간에 따라 구하는 것이 동적모사기의 역할이다. 편의상 모사시스템은 미분방정식의 해를 구하는 적분루틴과 공정의 특징을 나타내고 적분루틴에 여러 함수값들을 제공해 주는 함수루틴으로 구분된다. 이 때 이들의 상호 관계를 살펴보는 것이 구현된 모사전략을 이해하는데 필요하다. 식 (1)은 공정의 dynamics를 나타내는 미분식이고 식 (2)는 미분식과 커플링되어 있는 대수식을 나타내고 있다. 이 때 식 (2)를 대수변수 $\underline{x}$를 구하기 위한 절차로 생각하여 식 (1), (2)를 식(3), (4)와 같이 나타내기도 한다.

$$\frac{d\underline{y}}{dt}=\underline{f}(\underline{x}, \underline{y}, t) \tag{1}$$

$$\underline{0}=\underline{g}(\underline{x}, \underline{y}, t) \tag{2}$$

$$\frac{d\underline{y}}{dt}=\underline{f}(P, \underline{y}, t) \tag{3}$$

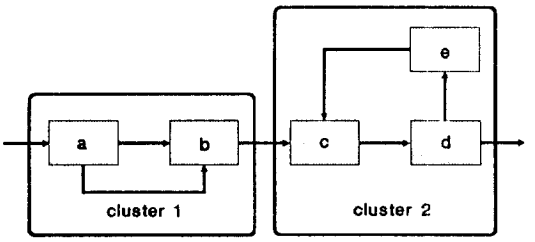

Fig. 1. Schematic diagram of a plant with 5 units.

$$P=g(\underline{x}, \underline{y}, t) \quad (4)$$

$$\underline{y}(0)=\underline{y}_o \quad (5)$$

여기에서

t=시간

$\underline{y}$=미분방정식의 상태변수(differential variables)

$\underline{x}$=대수방정식의 대수변수(algebraic variables)

P=절차함수(computational procedures)

SMI 기법에서는 개별적인 단위공정은 하나의 작은 독립적인 모사기 역할을 수행한다. 각각의 단위모듈 안에서 적분기는 식 (1)을 적분하여 $\underline{y}$값을 구하고, 이후에 매 적분간격마다 식 (2)에서 $\underline{x}$를 구한다. 실제로 다른 모듈에 대하여 독립적으로 수행되기 때문에, 모듈간의 심한 커플링이 존재하면 부정확성이 초래될 수 있고 계산성능도 떨어지게 된다. 그러나 모듈의 특성에 가장 적합한 적분방법이나 오차제어 등의 설정이 가능하다는 장점도 있다[13-17]. 반면에 SEI 기법에서는 식 (2)는 각각의 단위모듈 안에서 풀어지고 있으나, 식 (1)은 공정전체를 동시에 적분하는 방식을 취하고 있다. 본 연구에서 사용하는 SCI는 위의 두 방법의 혼합이라고 할 수 있다. 진정한 의미의 방정식 중심기법은 식 (1), (2)를 전공정에 걸쳐 동시에 수렴시키는 것이다. 그러나 실제로 식 (2) 형태의 대부분의 식들은 열역학 관계식 또는 유체역학적 식들인데, 이들을 함께 고려한다는 것은 현실적으로 어려움이 있다. 다음으로는 본 연구에서 사용하는 SCI 방식의 모사전략과 적분전략을 설명하였다. 또한 이들 방법이 다른 방법에 비하여 우수함을 보였다.

2-1. 모사 및 적분전략-SCI

방정식중심 기법들에서는 종종 효율을 위하여 전체의 방정식들을 몇 개의 작은 부분으로 분할(decomposition)하여 해석을 한다. 이러한 기법은 SCI 방법에서 공정을 cluster들로 분할하는 것과 비교될 수 있다. 전체의 모사문제는 작고 독립성이 강한 cluster들로 분할되며 각각의 cluster는 하나 이상의 단위공정의 모듈로

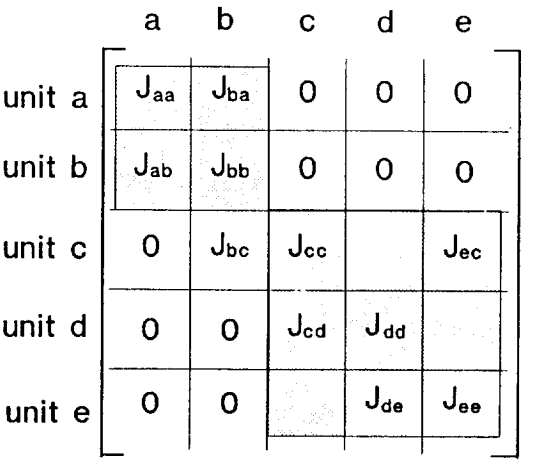

Fig. 2. The pattern of Jacobian matrix J.

이루어져 있다. 각 모듈은 대수식 계산의 단위가 되고, 각 cluster는 미분식 계산의 단위가 된다. 결국 문제의 크기는 공정 전체가 아닌 cluster에 달려 있다.

SCI의 가장 기본적인 장점은 공정의 특성에 맞는 clustering을 선택하여 모사의 효율을 증가시킬 수 있다는 것이다. Fig. 1과 같은 가상의 공정을 사용하여 SCI의 효율성을 살펴보았다. 공정의 clustering은 크게 두 가지 기준에 의해서 행해질 수 있다. 일반적인 정상상태 모사문제에서의 tearing과 유사하게 공정의 topology를 고려한 것과, 여기에 dynamics를 고려한 경우가 있다. Fig. 11의 공정을 2개의 cluster로 분할한 경우에 대하여 생각해 보자. 공정에 2개의 recycle이 존재하여 SMI 방법이라면 적어도 2개의 tear stream을 설정하여야 한다. 그러나 2개의 cluster로 분할을 하면 cluster 사이에는 recycle이 없게 된다. 이것은 tearing 방법을 사용하지 않아도 됨을 의미하며, cluster안의 적분은 SEI와 같이 동시에 커플링되어 행해진다. SCI 방법은 SEI와 같은 정도의 정확성을 유지하면서도 상대적으로 작은 크기의 Jacobian 행렬을 다루기 때문에 계산측면에서 유리하다. Fig. 1의 공정에 대한 Jacobian 행렬의 형태를 Fig. 2에 간략히 도시하였다.

Fig. 1에 대한 전체 Jacobian의 크기는 $(n_a+n_b+n_c+n_d+n_e)^2$이다. 예를 들어 각각의 차원이 10이라면, Jacobian 원소의 수는 $50^2=2500$이 된다. 그러나 2개의 서로 독립된 cluster로 분리하면 각각 $20^2+30^2=1300$으로 Jacobian 계산뿐만 아니라 기억장소 측면에서도 유리함을 알 수 있다. 물론 각각의 단위공정을 순차모듈식으로 풀면 Jacobian은 J_a, J_b, J_c, J_d, J_e부분만이 존재함으로 500개의 원소만 필요하다. 그러나 공정의 커플링을 나타내는 정보인 off-diagonal 항들이 무시되기 때문에

순차모듈식은 수렴하기 어렵게 된다.

다음으로 본 연구에서 사용하는 적분전략을 소개하면 다음과 같다. 사용하는 적분방법은 Gear method[21]에 기본을 두고 있고, 실제코드는 IMSL의 DGEAR이다. 이 코드를 여러 개의 cluster로 이루어진 방정식을 동시에 다룰 수 있도록 수정하였고, 전체적인 감독기능의 supervisory 루틴을 개발하였다[22, 23].

먼저 식 (1) 또는 (3)을 사용하는 적분방법에 따라 대응하는 대수식으로 변환할 수 있는데, 식 (6)과 같이 쓸 수 있다. 이 때 아래첨자는 cluster를 나타내고 c는 cluster의 갯수를 나타낸다.

$$\underline{F}_i(\underline{y}_1, \underline{y}_2, \ldots, \underline{y}_c, t) = 0 \qquad (6)$$

식 (6)은 (i)-cluster에 대한 모델식을 나타내는데, 이 때 (i)-cluster내부의 변수 $\underline{y}_i$와 다른 cluster의 변수 $\underline{y}_j$로 구분하여 다시 식 (7)과 같이 쓸 수 있다.

$$\underline{F}_i(\underline{y}_i, \underline{y}_j, t) = 0,\ i \neq j \text{ and } j = 1, 2, \cdots, c \qquad (7)$$

여기에서 $i = 1, 2, \cdots, c$

시간 t_n일 때 식 (7)에서 $\underline{y}_{j,n}$을 구하기 위해서는 $\underline{y}_{j,n}$을 미리 알아야 한다. Recycle이 없는 직렬 공정이라면 $\underline{y}_{j,n}$은 선행 cluster에서 풀어진 값이므로 이미 알고 있는 값이 되나, 일반적인 상황에서는 아직 정확한 $\underline{y}_{j,n}$ 값을 이용할 수 없다. 그래서 식 (8)을 사용하여 t_{n-1}에서의 정보를 가지고 t_n에서의 모든 $\underline{y}$에 대하여 예측값을 구한다. 이 때 Nordsieck history 배열 $\underline{z}$를 사용한다.

$$\underline{\tilde{z}}_{k,n} = D\underline{z}_{k,n-1} \qquad (8)$$

$$\underline{z}_{k,n} = [\underline{y}_{k,n}, h\underline{y}'_{k,n}, h^2\underline{y}^{(2)}_{k,n}/2!, \cdots, h^q\underline{y}^{(q)}_{k,n}/q!]^T \qquad (9)$$

이 때 $h = t_n - t_{n-1}$이고 q는 method order, D는 Pascal triangle matrix이다.

식 (7)을 quasi-Newton Raphson 방법으로 풀 경우에는 초기값을 (8)식에 의하여 구한다. 식 (7)이 수렴을 하면 오차시험을 하여 한 스텝의 적분을 끝내게 된다. 정리하면 아래와 같은 전략이 본 연구에서 사용하는 SCI 방법이다.

알고리듬

1. 모든 cluster를 순차적으로 예측한다(predictor).
2. 예측값을 가지고 각 모듈에서 대수식 계산을 한다.
3. 모든 cluster에 대하여
   a. 수정계산(corrector)을 한다. 이 때 매 corrector 반복계산이 시행될 때마다 대수변수도 함께 갱신한다.
   b. 만약 현재의 cluster가 수렴을 하지 않거나, 수렴을 하였다고 해도 오차시험에 실패하면 현재의 시간 horizon에서의 모든 cluster의 수정된 값들을 취소하고, 적절한 적분간격을 다시 구하여 1번 스텝으로 간다.
4. 모든 cluster가 성공적으로 적분이 되었으면, 매 적정 간격마다 적분차수를 고려하여 최대 적분간격을 구한다.

하나의 cluster는 모듈방식에서의 모듈에 해당하는 개념이므로, 독립적인 적분방법을 사용할 수 있다. 즉, 각각의 모듈특성에 따라 stiff 또는 nonstiff 적분방법이나, implicit 또는 explict 방법을 사용할 수 있을 것이다. 그리고 현재의 시스템에서와 같이 동일한 적분방법을 사용할 경우에는 각각의 cluster는 서로 다른 적분간격과 적분차수를 가질 수 있다. 이러한 경우는 서로 다른 시간에 모듈간의 통신을 제어하기 위하여 coordinator가 필요하다[13, 16].

현재 채택하고 있는 방식은 모든 cluster를 통하여 같은 크기의 적분간격과 적분차수를 갖는다. 이 때 최적 적분간격을 구하는 방법은 다음과 같다. 즉, 각각의 cluster가 성공적으로 적분되면 현재의 차수와 이보다 낮은 차수, 높은 차수에 대하여 다음 시간에 적용가능한 적분간격, h_i, h_{i-}, h_{i+}를 구한다. 모든 cluster가 성공적이고 다음 적분간격을 정하기 위해서, 각각의 cluster에서는 가장 큰 h값을 제안하고 시스템에서는 이들 중에서 가장 작은 값을 제안한 cluster의 h값과 그 때의 적분차수를 선택하게 된다. 결국 가장 빠른 응답을 보이는 cluster에 의해 공정의 적분간격이 좌우되지만, 이 보다 큰 적분간격을 제안한 cluster는 explicit 형태인 예측계산만으로 수렴이 될 수 있으므로 크게 손해보는 일은 아니다. 일반적으로 방정식 중심기법에서는 이와 같은 모듈의 특성을 고려할 수 없다. 즉 가장 빠른 응답을 보이는 공정에 좌우되어 전체의 Jacobian 계산은 증가하게 된다. Guru 등[5]은 방정식 중심기법에서 이와 같은 비효율성을 개선하기 위하여 Jacobian의 부분적 갱신을 도입하기도 했다.

2-2. 시스템의 구현

SCI 전략을 사용하는 다목적 화학공정 동적모사기를 개발하기 위한 시스템의 구조를 연구하고 구현하였다. 시스템 본체 및 모델들은 FORTRAN 77을 사용하여 구현하였고, 사용자 인터페이스 일부는 DEC사의 DCL을 사용하였다. 그래픽 부분은 DEC사의 그래픽 명령어인 ReGIS를 사용하였다.

시스템 구현에 있어서 다음과 같은 주요 인자들을 고려하였다. 즉, 자료구조의 효율성과 유연성, 시스템의 확장성과 모듈성, 사용자 편리, 계산효율과 기억장소의

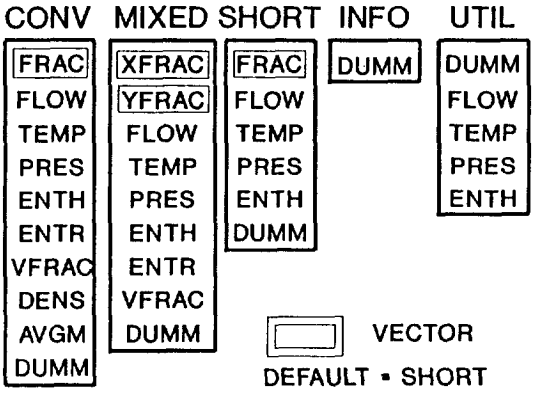

Fig. 3. Stream classification.

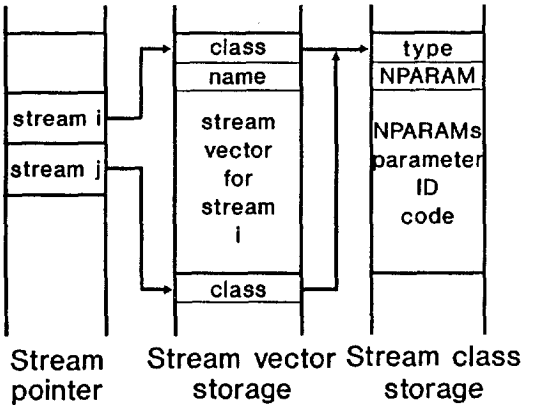

Fig. 4. Stream data structure.

절약 등이다.

2-2-1. 자료구조(Data structure)

대부분의 공정모사기들은 배열형의 데이타구조를 갖고 있다. 모든 스트림과 장치 데이타를 저장하기 위하여 2차원 또는 그 이상의 고차원배열을 사용한다. 행렬형의 데이타구조는 쉽게 이해할 수 있으며, 개발하기 쉽지만, 새로운 변수나 스트림을 정의하려면 모든 프로그램을 수정하여야 한다. 그리고 기존의 기-액 평형공정에서 벗어나, 고체를 다루는 공정이나 제어나 설계문제를 위한 정보흐름을 원활히 다룰 수가 없다[18, 19].

본 연구에서는 이와 같은 점에 주목하여 모든 데이타구조를 일차원으로 저장하는 Plex 데이타구조를 채택하였다. 이 구조는 값을 저장하기 위한 bead storage와 특정 데이타의 위치를 가르키는 포인터로 구성되어 진다.

2-2-1-1. 스트림 자료구조(Stream data structure)

공정에 나타나는 스트림을 몇 개의 서로 다른 특성을 가지는 클래스로 분류하면 계산상의 효율은 물론이고 기억장소도 절약된다. 모든 스트림은 하나의 클래스에 속하게 되는데, 모사시에 사용자는 스트림들의 클래스를 정해야 한다. 일반적으로 물질 스트림(material stream), 정보 스트림(informatoin stream), 유틸리티 스트림(utility stream) 등의 세 가지 클래스를 생각할 수 있다. Fig. 3에 현재 정의되어 있는 스트림 클래스를 나타내었다. 대부분의 keyword는 약어로 표시되었고, 특히 FRAC, XFRAC, YFRAC는 각각 bulk, 액체, 기체의 조성을 나타낸다.

모든 스트림들은 하나의 스트림 클래스를 가진다. Fig. 4에 스트림 자료구조를 도식적으로 나타내었다. 실제의 스트림 정보값들은 stream bead에 저장되고 그 스트림의 종류(class)에 따라 저장된 값들의 의미는 class bead에 저장되어 있다.

2-2-1-2. 장치 자료구조(Equipment data structure)

각 단위공정(장치)에 관련되는 인자들을 저장한다. 실수형값을 가지는 인자와 정수형값을 가지는 인자로 구분된다. 이들의 데이타구조는 스트림 데이타구조와 동일하다. 각각의 단위공정이 계산되면, 연결되어 있는 공정과의 정보 교환은 스트림을 통하여 하게 된다. 그러나 스트림에 저장되지 못하는 기타의 값들은 모듈(단위공정) 내부에 장치 데이타로 저장된다. 예를 들어 탱크를 보면, 사용자가 지정한 값(단면적), 상태변수(탱크안의 물질의 양), 또는 상태변수에 커플링이 되어 있는 변수(높이, 밀도) 등은 실수형 데이타에 저장된다. 그리고 해당 단위공정으로 들어오고 나가는 스트림의 수와 그 스트림들의 번호들이나 기타 여러 계산 선택사양들이 정수형값을 가진다.

2-2-1-3. 기타 모사 시스템의 자료구조

크게 스트림 자료와 장치 자료 이외에도 동적모사를 위한 여러 종류의 자료구조가 정의되어야 한다. 공정이 여러 개의 cluster로 구분되어야 하므로 이들을 기억하고 계산순서 등을 기억하는 cluster 자료구조가 있다. 또한 모듈구조를 가지는 모사기를 가지고 방정식 중심기법을 적용시키기 위해서, cluster의 방정식을 이루는 각각의 단위모듈들의 위치를 기억해야 한다. 즉, 특정 모듈은 방정식의 어떠한 부분에 위치하는지를 기억해야 한다. 다음으로 열역학적 계산을 위하여 데이타베이스의 자료들을 읽어오기 위한 구조가 있다. 또한 화학공정의 특징중의 하나가 모델식들이 희소행렬을 이룬다는 것이다. 그래서 희소형태(sparsity pattern)을 기억하여 Jacobian 행렬을 생성할 때 효율을 대폭적으로 높일 수 있다.

2-2-2. 입출력 구조

소프트웨어의 생명력은 입출력 구조에 가장 직결된

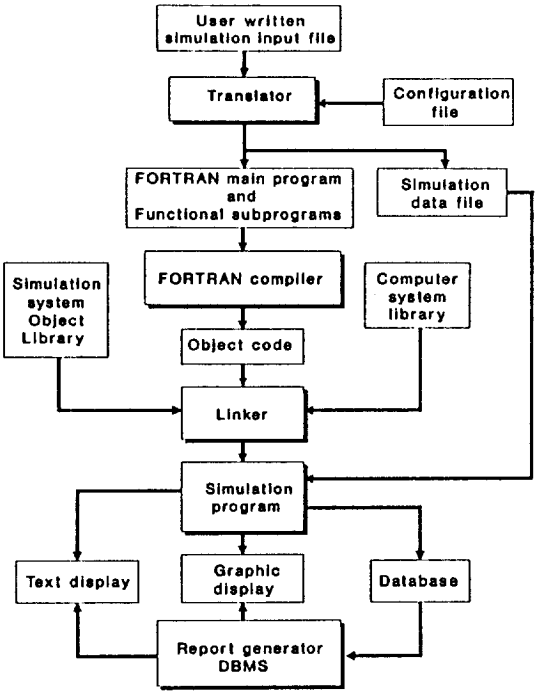

Fig. 5. Global structure of dynamic simulation system.

다고 할 수 있다. 사용하기 어렵고 결과분석이 용이하지 않으면 아무리 훌륭한 알고리듬이 사용되었을지라도 곧 사장되는 경우가 허다하다. 본 모사기는 다음에 설명할 입력언어와 더불어 최적 입출력 구조를 설계하였다. 모사기 전체의 데이타구조가 Plex 구조이므로 모든 자료들의 논리적인 접근이 쉽다. 이 점은 입력언어의 유연성과 효율성에 많은 도움이 된다. 모사결과의 출력은 다음의 두 경우로 나뉘어 진다. 첫째로 실행중에 계속 원하는 변수값들의 변화가 모니터될 수 있고, 그 변화 추이는 실행중에 그래픽 터미날로 출력된다. 두번째는 모사결과 발생하는 모든 결과를 잘 설계된 데이타베이스에 출력하는 것이다. 이것은 실행이 종료된 후에 데이타베이스 인터페이스를 통해 결과를 분석하거나 특정변수들의 추이를 그래프로 그릴 수 있게 한다. 또한 원하는 경우 모사를 특정 시간부터 다시 시작시킬 수 있게 한다. 거의 모든 응용 프로그램은 어느 정도의 일반성을 가진다. 다시 말하면, 프로그램의 몇 가지 인자들에 의해서 다른 문제를 실행시키게 되어 있다. 예를 들어 기본적인 계산상의 알고리듬, 기억장소 구조, 입출력 방법, 컴퓨터 자원의 할당 및 배분, 포맷옵션, 그리고 그밖의 여러 가지 등이 인자들에 의해서 결정될 수 있다. 일반성을 가지는 프로그램은 코드의 길이가 길어지고, 따라서 실행속도가 약간 떨어지는 것이 보통이다. 그래서 특정문제에 맞는 프로그램을 자동으로 생성하는 방식이 generator이다. 본 모사기는 입력화일로부터 실행하고자 하는 모사문제에 맞는 프로그램을 생성하도록 되어 있고 또한 원하는 양식의 출력을 생성하도록 하는 방식을 택하고 있다. Fig. 5에 전체적인 구조를 보였다.

Table 1. Functional descriptions of input language

Section	Descriptions
Title section	Problem title statement
Component	Define chemical components in data base
Unit operation	Define flowsheet and unit operation
Stream	Supply stream vector
Stream class	Classification of stream
Initial	Supply initial conditions
Cluster	Clustering of flowsheet
Integration option	Integration options
Action	User defined operation command or spec.
End	Indicates end of file

2-3. 입력언어(Input language)

모사시스템이 전체적으로 완성되어 있지 않아도 입력언어와 이를 번역하는 프로그램의 개발은 전체 시스템 개발에 아주 유리하다. 입력언어의 설계는 한번에 끝나는 작업이 아니고 점차적으로 그 기능이 추가될 수 있도록 설계하는 것이 중요하다. 번역 프로그램은 입력한 텍스트 화일을 읽어들여 문장 분석을 하게 되므로, 일반적으로 문자 처리나 리스트 처리에 뛰어난 컴퓨터 언어를 사용한다. 범용성이 뛰어난 언어중 Fortran 77도 문자 처리에 충분한 능력을 발휘한다. 그렇지만 기본적으로 문자 처리를 위한 내장함수가 많지 않으므로 몇몇 유용한 유틸리티 프로그램을 개발하여 사용하였다. 이러한 유틸리티 프로그램과 잘 구조화된 번역기는 비단 본 연구의 동적모사기에만 사용되는 것이 아니라, 다른 시스템의 입력언어의 개발에 사용될 수 있을 것이다.

Table 1에 본 연구에서 개발된 모사기에서 현재까지 갖추고 있는 입력언어의 기능을 나타내었다.

2-4. 계산효율 향상

화학공정 동적모사는 정상상태모사의 경우보다 일반적으로 계산량이 훨씬 많고 또한 어려운 면이 있다. 여기에서는 이러한 문제를 몇 가지로 나누어 기술하였다. 현재 모사시스템에 적용된 것도 있고 적용시키기

위하여 계속 연구중에 있는 것도 있다.

2-4-1. 수치적분

화학공정의 동적모사를 위한 모델식들은 대체로 비선형의 stiff 시스템이고, 빈번한 불연속점을 갖는다. Stiffness 문제는 BDF(backward differentiation formula) 방식을 사용하여 어느 정도 극복이 되고 있다. 또한, 공정을 clustering할 때 장치의 동특성이 현격히 다른 것은 다른 cluster로 구분하는 것도 stiffness를 해결할 수 있는 한 방법이다. 이것에 대한 시험은 예제문제로 살펴보았다. 제어전략이나 기타 조업상의 이유 등으로 발생하는 불연속은 모사기의 계산성능을 현격히 저하시키고 원하는 모사 자체를 불가능하게 하기도 한다. 이에 대처하는 방안은 몇 가지로 나누어 볼 수 있는데, 기본적으로는 불연속점을 찾아내어 적분을 다시 시작하는 것이 그 해결방법이다. 한편, 불연속점에서는 과거값에 영향을 받지 않는 Runge-Kutta류의 적분방법으로 전환하는 방식도 생각할 수도 있다. 그러나 한 종류의 적분방법을 일관성있게 사용하는 방법에서는 적분차수를 1로 하고 작은 적분간격으로 불연속점을 통과하기도 한다[3].

본 연구에서는 불연속을 자동적으로 검출하여 정확한 불연속점을 찾은 다음 그 점을 기점으로 모사를 다시 시작하는 방식을 쓰고 있다[22]. 불연속점이 시간으로 나타나는 explicit 경우는 비교적 쉽게 해결이 되지만, 불연속점이 적분이 진행되어야만 알 수 있는 상태변수의 조합으로 나타나는 implicit 경우는 불연속을 유발시키는 함수를 계속 모니터하여 그 부호가 변하는 점을 찾아야 한다. 일단 불연속점을 찾고 나면, 그 후는 explicit 경우와 같이 처리된다.

2-4-2. Sparse Jacobian 기법

방정식 중심이나 clustering을 사용하는 기법에서는 대상공정이 복잡하고 커질수록 구성되는 Jacobian 행렬은 희소행렬이 된다. 희소행렬이란 전체 행렬의 원소중에서 값이 0인 원소가 차지하는 비율이 매우 큰 행렬을 말한다. 알려진 바와 같이 희소행렬은 그 희소형태를 분석하여 이용하면 상당량의 계산을 절약할 수 있다. 많은 연구들에서 Harwell MA28코드를 성공적으로 사용하고 있으며, 본 연구에서도 이 코드를 사용하고 있다. 이 때 희소행렬의 계산뿐만 아니라, 희소행렬의 갱신이나 생성시에도 희소형태를 이용하면 상당량의 계산을 절약할 수 있다. 일반적으로 유한 차분법에 의해서 Jacobian을 구할 경우에 식 (10)을 사용한다.

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\underline{x}+\delta_j \underline{e}_j) - f_i(\underline{x})}{\delta_j} \quad i, j=1,\cdots, N \qquad (10)$$

이 때 $\underline{e}_j$는 단위벡터이고 N은 Jacobian의 크기이다.

식 (10)에 의하면 N^2 Jacobian 원소들을 구하려면 N^2+N번의 함수계산이 필요한 셈이다. 즉, N개의 변수를 차례로 perturbation시켜야 한다. 그러나 행렬을 잘 분석하면 두 개 이상의 변수를 동시에 perturbation시킬 수 있어서 계산횟수를 줄일 수 있다. 즉, 같은 식에 동시에 영향을 주지 않는 변수들의 그룹을 찾아내면 같은 그룹의 변수들은 동시에 perturbation 시킬 수 있다. 만약에 변수들의 그룹수가 N_g라면 $N_g(N)+N$번의 함수계산만이 필요하게 된다. 즉, 이 방식을 사용했을 때의 효율은 다음과 같다.

$$\% \text{ Saving} = \frac{N_g+1}{N+1} \times 100 \qquad (11)$$

다음에 변수들의 그룹을 발견하는 알고리듬을 pseudo-code 형태로 나타내었다. 이 때 R, C는 희소형태를 저장하기 위한 배열이며, G는 변수들의 그룹짓는 정보를, P는 각각의 그룹의 위치를 기억하는 배열이다.

Tridiagonal 행렬에 제안된 알고리듬을 적용시키면 3개의 변수 그룹을 얻는다. 즉 {1, 4, 7,,,}, {2, 5, 8,,,}, {3, 6, 9,,,}. 이 경우에는 행렬의 크기에 관계없이 단 3번의 perturbation 만이 필요하게 된다.

알고리듬

```
(1) Initially G={1, 2,···, N} and D={0, 0,···, 0}
              L, P, R, C=φ
    Select Pivot_col in G set; pivot_col=1
(2) Do i=1, N; Check equations
      IF element(i, pivot_col) is non_zero THEN
        push i to R set;store row position
        push pivot_col to C set;store column position
        Do j=pivot_col+1, N    ;check variables
          IF element(i,j) is non_zero THEN
            delete j in G set
            add j in D set;j must be other group
          ENDIF
        ENDDO; for all variables after pivot_col
      ENDIF
    ENDDO; for all equations
(3) IF there is no element to select pivot_col in G set
      THEN
      save G set in L set; one group is created.
      make pointer to this group in L set
      IF D set is empty THEN
        exit; all group are found
      ELSE
        empty G set and move D set to G set
```

```
        empty D set
        select pivot_col in new G set
      ENDIF
    ELSE
      select new pivot_col in G set
    ENDIF
    goto step(2)
```

Table 2. CPU time for typical tray column simulation[11]

	Simultaneous	Sequential
Physical properties	79.4	82.9
Unit module routine calculations	5.7	4.7
Storage operation	4.2	4.3
Sparse LU strategy/ decomposition	3.9	0.8
Sparse LU solution	3.5	3.0
Gear integrator	1.6	2.2
Supervisory routine	1.7	2.1
Total	100.0(%)	100.0(%)

2-4-3. 열역학 계산

많은 사람들의 연구결과 화학공정 모사시 대부분의 계산은 열역학루틴에서 소요되는 것으로 보고되고 있다. 특히 비정상상태모사의 경우는 Table 2에서 보는 바와 같이 물성치 계산이 차지하는 비중이 더욱 크다.

물성치 계산의 효율을 높이는 방법으로는 LAM(Local Approximation Method)과 같은 shortcut 기법을 사용하거나, 물성계산 루틴은 그대로 둔채 전체적으로 적은 수의 물성치 계산루틴을 호출하도록 Jacobian의 갱신 방법이나 적분방법의 효율을 높이는 간접적인 방법이 있을 수 있다. 본 연구에서는 앞으로 후자의 방법을 적극적으로 사용할 것이며, LAM의 사용도 고려하고 있다.

3. 예 제

개발된 시스템의 성능 평가와 제안된 전략의 평가를 위해서 몇 가지 간단한 예제를 다루었다. Fagley 등[11]은 처음으로 sequential-clustered approach의 개념을 도입하였는데, 아직까지 cluster를 정하는 문제는 언급하지 않고 있다. 다만 적절한 크기의 cluster가 모사에 유리하다는 것만을 보였다. 첫번째 예제에서는 일반적인 공정에 대한 적절한 clustering 기준을 설정하기 위하여 간단한 4개의 혼합탱크를 여러 방식으로 연결하여 모사를 시행하였다. 각 탱크의 holdup을 조정하여 동특성의 차이를 나타내었고 일정시간에 입력의 불연속을 도입하여 문제를 더욱 어렵게 만들었다. 두번째 예제는 제어연구에 적용하기 위해 Fletcher 등[8]이 보인 탱크의 액위 제어문제를 보였다.

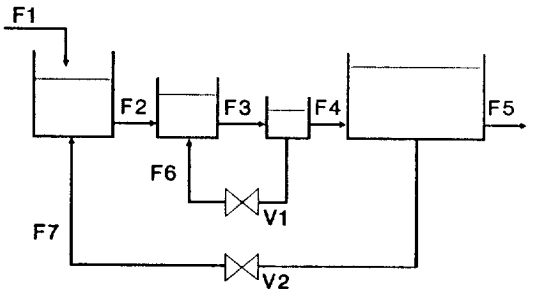

Fig. 6. Schematic diagram of 4-tank plant.

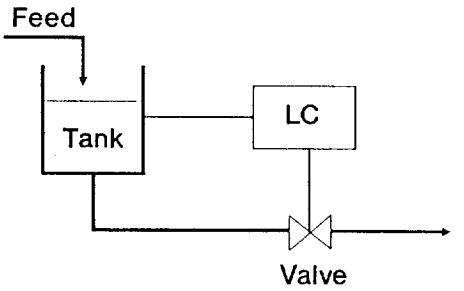

Fig. 7. Liquid tank level control.

3-1. 4-탱크문제

각각의 유량은 각 탱크의 holdup이 시간에 따라 약간 증가하거나 일정하게 되도록 시간의 함수형태로 주었다. 그림에서 탱크의 크기는 대략적인 탱크의 holdup을 나타낸다. 탱크의 holdup 초기값으로 각각 10, 0.1, 0.01, 50,000으로 설정했다. 모든 경우에 오차한계를 0.0001로 하였고 각 clustering에 대한 계산효율의 비교에 함수계산을 위한 함수들의 호출 횟수를 사용하였다. 성능인자인 PI(performance index)는 SEI에 해당하는 하나의 cluster로 이루어진 방법을 기준으로 하여 계산한 것이다.

Case 1. **재순환흐름이 없는 경우**($F_6=F_7=0$)

재순환흐름이 없는 공정이 되므로 SMI 방식이 가장 유리할 것임을 즉시 알 수 있다. Table 3에 clustering에 대한 모사결과를 비교하였다.

이 문제의 경우는 계산속도는 단지 모듈의 동특성의 차이와 방정식수에 관계된다. 전체를 동시에 수렴시키는 S7의 경우에 비하여 순차모듈식으로 수렴시킨 S1의 경우가 2배 이상의 효율을 보인다. 일반적으로 S9의

Table 3. Computational efficiency versus clustering(Case 1) TOL=0.0001 TEND=10(hr)

구분	Clustering	Nstep	PI
S1	(1), (2), (3), (4)	206	2.08
S2	(1, 2), (3), (4)	195	1.79
S3	(1), (2, 3), (4)	196	1.79
S4	(1, 2), (3, 4)	196	1.51
S5	(1, 2, 3), (4)	193	1.38
S6	(1), (2, 3, 4)	201	1.30
S7	(1, 2, 3, 4)	190	1.00
S8	(4), (1), (2), (3)	206	2.07
S9	(1), (3), (2), (4)	634	0.58
S10	(3), (4), (1), (2)	355	1.14
S11	(2), (3), (4), (1)	306	1.32

경우와 같이 공정의 topology를 무시하고 계산순서를 정하면 상당한 손해를 본다. 그러나 S8의 경우도 topology를 무시하고 4번 탱크를 먼저 풀었지만 효율은 S1의 경우와 거의 같다. 이유는 공정의 동특성에 기인한다. 즉 4번 탱크는 용량이 무척 큰 용기로 그 시간상수가 500에 이른다. 그러므로 하나의 큰 버퍼로 작용하여 공정에 영향을 거의 미치지 않게 된다. 이러한 성질을 latency라 하고 이러한 모듈의 발견은 모사를 유리하게 한다. 즉, 이 공정은 적은 횟수의 반복계산으로 쉽게 수렴하므로 이를 이용하면 계산효율을 높일 수 있다. S10, S11의 경우에는 입력을 받는 모듈(1번 탱크)를 먼저 풀지 않았기 때문에 순수한 공정의 특성을 살리지 못한 것이라고 볼 수 있다. 실제로 모사를 수행하여 살펴보면 대부분의 clustering이 S7보다 함수계산 횟수에서 유리하지만 Jacobian 계산에서는 S1을 제외하고는 매우 불리하다. 즉 Jacobian 계산 비용이 상대적으로 아주 비싸다면, S1만이 가장 훌륭한 모사 방법이다.

Case 2. 하나의 재순환흐름으로 커플링 도입($F_6 \neq 0$, $F_7 = 0$)

F_6에 의해서 탱크 3과 탱크 2의 커플링을 도입하였다. 적절한 clustering을 예상해 보면 (1), (2, 3), (4)가 우선 후보가 될 것임을 쉽게 알 수 있다. Table 4에 결과를 나타내었다.

빠른 응답을 보이면서 서로 커플링이 되어 있는 2, 3번 탱크를 하나의 클러스터로 정의하는 것이 가장 유리함을 알 수 있다. S5와 S6의 경우는 모두 커플링 공정인 (2), (3)을 tearing한 것인데, 방정식 크기가 작은 S6이 조금 유리하다. 일반적으로 재순환 흐름 등이 존재하여 공정이 커플링되면 하나의 클러스터로 정의하는 것이 유리하다. 그러나 재순환 흐름이 존재하여 공정간의 커플링을 유발하더라도 반드시 하나의 클러스터로 포함시키는 것만이 유리한 것은 아니다. 즉, cluster의 크기가

Table 4. Computational efficiency versus clustering(Case 2) TOL=0.0001 TEND=10(hr)

구분	Clustering	Nstep	PI
S1	(1), (2, 3), (4)	241	1.45
S2	(1, 2, 3), (4)	204	1.41
S3	(1), (2, 3, 4)	226	1.08
S4	(1, 2, 3, 4)	197	1.00
S5	(1, 2), (3, 4)	460	0.59
S6	(1), (2), (3), (4)	446	0.82

Table 5. Computational efficiency versus clustering(Case 3) TOL=0.0001 TEND=10(hr)

구분	Clustering	Nstep	PI
S1	(1), (2, 3), (4)	203	1.68
S2	(1, 2, 3), (4)	197	1.39
S3	(1), (2, 3, 4)	259	0.89
S4	(1, 2, 3, 4)	196	1.00
S5	(1, 2), (3, 4)	314	0.84
S6	(1), (2), (3), (4)	395	0.98

너무 커지거나, 공정의 동특성상 하나의 클러스터로 정의하지 않는 것이 유리할 경우가 있다. 다음의 Case 3에서 이와 같은 현상을 알 수 있다.

Case 3. 두 개의 재순환흐름의 도입($F_6 \neq 0$, $F_7 \neq 0$)

두 개의 recycle을 도입하여 상호간의 커플링을 심하게 하였다. 이러한 경우에 단순히 topology 만을 고려한다면, 동시해법이 유리할 것 같지만 공정의 동특성을 살펴보면 F_7에 의하여 도입된 탱크 1과 탱크 4와의 커플링은 너무 미약하여 실제로는 재순환 흐름이 없는 것과 유사한 결과가 나온다. 위의 모든 경우에 S2와 같은 경우가 S3와 같은 경우보다 효율이 좋았다. 외관상으로는 두 경우의 차이는 없지만 공정의 동특성에 기인한다. 즉 (2), (3)번 탱크는 시간상수가 1×10^{-3}, 1×10^{-4}이고 (4)번 탱크는 5×10^{2}정도인데, 현격한 동특성의 차이가 있는 공정들을 하나의 cluster로 하는 경우는 stiffness 문제를 더욱 노출하는 결과를 가져온다.

위의 간단한 4개의 탱크문제를 풀면서 다음과 같은 일반적인 clustering 기준을 제시할 수 있다.

기준 1 : 재순환없는 공정이거나 재순환 흐름의 시간지연이 적분간격보다 훨씬 크다면 순차모듈방식을 택한다. 이 때 계산순서는 정보의 흐름을 따라간다.

기준 2 : 재순환 흐름 등으로 공정이 상호 커플링이 되면 재순환 흐름을 포함하는 공정을 하나의 클러스터로 설정한다.

기준 3 : 외란이 도입되는 공정은 되도록 고립시킨다.

기준 4 : 장치의 용량이나 시간 상수들이 현격하게 차이가 나는 공정들은 하나의 클러스터로 묶지 않는

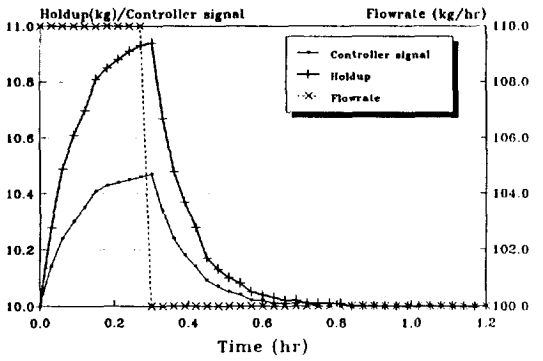

Fig. 8. Results of simulation(liquid level control).

것이 좋다.

기준 5 : Tear 스트림을 정할 때는 동적으로 가장 느린 공정의 출력스트림을 정한다. 이는 tear 스트림의 공정에의 영향을 최소화하려는 이유에서이다. 이 tear 스트림은 cluster 간의 경계에 해당한다.

기준 6 : 각 cluster의 계산속도는 방정식의 Jacobian 행렬의 크기와 관련되므로 각 cluster는 거의 비슷한 계산량을 가지도록 정한다.

위와 같은 기준들은 정량화하기는 곤란하지만 어느 정도의 기준을 제시하고 있다. 실제로 공정의 모사에 앞서서 미리 공정의 동특성을 알기는 어렵다. 그러나 방정식의 수, 공정의 topology, 단위공정의 특성, 사용하는 화학물질의 성질, 컴퓨터의 능력 등을 고려하여 타당한 clustering을 자동적으로 제시하도록 연구해야 한다.

3-2. Level controller-control study

일반적으로 대부분의 제어 알고리듬은 서브루틴 형태로 모사기에 내장시킬 수 있다. 또한 특별한 제어 알고리듬이나 계산식 등은 입력언어에서 직접 정의하여 사용할 수 있다. 이 문제에서는 모사기에 내장된 PID 제어기의 P모드를 사용하였고 제어밸브는 입력 화일에서 직접 정의하였다. Fig. 8에 제어신호와 탱크의 holdup을 나타내었다. Fig. 9에서는 이 문제의 입력 화일을 나타내었다. ACTION문에서 FORTRAN 형태로 제어밸브를 정의하고 있다.

```
* Simple liquid level tank
*
COMPONENTS = 1, 2
* Unit Operation and Flowsheet section
UNIT OPERATION
     LTANK 1 = TANK  1  -2 -3
          HOLDup = 10.0
     PID_CTRL 2 = LC, 3 -4
          SETPOINT = 10. GAIN = 0.5  RESET = 10.
STREAM CLASS
   class1 = ALL
CLUSTER
   clus1 = ALL
INITIAL
    WITHIN STREAM = 1 2
          FLOW = 100
PLOT
    DEFINE SC SIGNAL STREAM 4
    DEFINE HB HOLDUP EQUIP 1
    DEFINE FEED FLOW STREAM 1
    MAX_X = 1.2
    MIN_Y = 10.10,100
    MAX_Y = 11.11,110
    GRID_ON = ON
    DEVICE = TTA3:
ACTION
    DEFINE HB HOLD EQUIP = 1
    DEFINE FOUT FLOW STRE  = 2
    DEFINE FEED FLOW STREAM 1
    DEFINE SC   SIGN STREAM = 4
*
*  Valve action (SPECIFIC CASE IN THIS PROBLEM)
    CV = 3.162277
    FOUT = SC * CV*DSQRT( HB )
    IF ( TIME .LE. 0.3 ) THEN
         FEED = 110.0
    ELSE
         FEED = 100
    ENDIF
* INTEGRATION OPTIONS
INTEGRATION OPTIONS
    STEP = 0.0001   FINAL = 1.2  FREQ = 0
    TOLER = 0.00001 METH = 22    DELT = 0.03
END
```

Fig. 9. List of input file: level tank control.

4. 결론 및 제안사항

범용 화학공정 동적모사기 개발에 관한 연구의 일환으로 개발중인 prototype의 전반적인 구조와 모사기법에 관하여 소개하였다. 개발비용, 계산속도, 정확성, 응용성 등의 여러 면에서 유리한 모사기법인 SCI를 구현하였다. 이 방식의 가장 큰 장점은 유연한 구조로 인하여 다양한 방식의 모사를 할 수 있다는 것이다. 예제에서 살펴본 바와 같이 이 방식은 적절한 크기의 cluster가 정의되면 최대의 계산효율을 보인다.

Cluster의 크기가 커지면 SEI 기법에 가까와지는데, 이 때에는 적분스텝수와 Jacobian 행렬의 갱신횟수에 있어서 다른 방법에 비하여 유리하지만 Jacobian의 크기가 커지므로 갱신에 많은 비용이 소요된다. 그러나 본 연구에서 제시한 Jacobian 갱신 기법을 사용하면 상당량의 계산상의 비용을 절감시킬 수 있다.

SCI의 성능을 높이기 위해서는 예제에서 제시된 clustering 기준들을 더욱 보강하여야 하며, 모사문제에 따라 clustering이 자동으로 이루어질 수 있도록 연구해야 한다. 동적모사에 관련하여 수치해법적 측면에서 stiffness, 불연속, 시간지연, index 문제 등에 관해 계속적인 연구가 필요하다. 또한 공정 소프트웨어를 개발하는 입장에서는 다목적의 응용성과 사용의 편리성, 계산의

정확성들을 고려해야 한다.

감 사

본 연구는 목적기초연구의 일환으로 수행되었으며, 이를 지원해 준 한국과학재단에 감사를 드립니다.

NOMENCLATURE

D : Pascal triangle matrix
e : unit vector
h : step size($t_n - t_{n-1}$)
h_i : proposed next step size for current method order, q
h_{i-} : proposed next step size for q−1
h_{i+} : proposed next step size for q+1
J : Jacobian matrix
N : number of ODE equations
N_g : number of independent variable group
P : computational procedures
q : method order
t : time
x : algebraic variable
y : state variable
$\underline{z}$: Nordsieck history array
$\underline{\tilde{z}}$: predicted Nordsieck history array

REFERENCES

1. Franks, R. G. E.: "Modeling and Simulation in Chemical Engineering", John Wiley & Sons, New York(1972).
2. Patterson, G. K. and Rozsa, R. B.: *Comput. Chem. Engng.*, **4**, 1(1980).
3. Pantelides, C. C.: *Comput. Chem. Engng.*, **12**, 745 (1988).
4. Kuru, S.: Ph. D. Dissertation, Carnegie-Mellon Univ. (1981).
5. Kuru, S. and Westerberg, A. W.: *Comput. Chem. Engng.*, **9**, 175(1985).
6. Smith, G. J. and Morton, W.: *Comput. Chem. Engng.*, **12**, 469(1988).
7. Holl, P., Marquardt, W. and Gilles, E. D.: *Comput. Chem. Engng.*, **12**, 421(1988).
8. Fletcher, J. P. and Ogbonda, J. E.: *Comput. Chem. Engng.*, **12**, 401(1988).
9. Ponton, J. W., Johnston, E. M., Forsyth, J. M., Matheson, A. and Rutherford, F. M.: *Comput. Chem. Engng.*, **13**, 1245(1989).
10. Cera, G. D.: *Comput. Chem. Engng.*, **13**, 737(1989).
11. Fagley, J. C.: Ph. D. Dissertation, Univ. of Michigan (1984).
12. Fagley, J. C. and Carnahan, B.: *Comput. Chem. Engng.*, **14**, 161(1990).
13. Liu, Yin-Chang and Brosilow, Coleman B.: *Comput. Chem. Engng.*, **11**, 241(1987).
14. Cook, W. J., Klatt, John and Brosilow, Coleman B.: *Comput. Chem. Engng.*, **11**, 255(1987).
15. Hillestad, M. and Hertzberg, T.: *Comput. Chem. Engng.*, **10**, 377(1986).
16. Hillestad, M. and Hertzberg, T.: *Comput. Chem. Engng.*, **12**, 407(1988).
17. Ponton, J. W.: *Comput. Chem. Engng.*, **7**, 12(1983).
18. Britt, H. I.: *Found. of Computer-aided Chem. Proc. Des.*, **I**, 471(1980).
19. Husain, A.: "Chemical Process Simulation", John Wiley & Sons, (1986).
20. Westerberg, A. W., Hutchison, H. P., Motard, R. L. and Winter, P.: "Process Flowsheeting", Cambridge University Press(1979).
21. Gear, C. W.: "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, (1971).
22. 박정애 : 석사학위 논문, 서울대학교(1990).
23. 이강주 : 석사학위 논문, 서울대학교(1989).