'89 韓國自動制御學術會議論文集 1989. 10. 27~28

화학공정 비정상상태 모사기의 최적 적분전략에 대한 고찰

박 정애°, 이 강주, 윤 인섭

서울대학교 공과대학 화학공학과

화학공정 비정상상태 모사에 있어서 계산상 불리한 특성인 불연속성과 stiff한 성질에 대처할 수 있도록 sequential-clustered 구조를 기본으로 하는 모사기에 불연속 처리 루틴을, 구현하였고, stiff 성질의 완화를 위해 공정의 동특성 차이에 기인하는 latency를 이용하여 적절한 clustering 기법으로 cluster 크기를 결정하는 pre-processor를 개발하였다.

1. 서 론

화학공정의 비정상 상태 모사(dynamic simulation)는 정상 상태 모사만으로는 불가능한 프로세스 외란에 대한 시스템의 변화, 제어변수의 영향, 조업조건 연구, 콘트롤 시스템 평가, operability 연구, 프로세스 변수의 영향, 재해연구, 공정의 start-up과 shut-down등에 대한 많은 연구를 수행하고, 공정의 설계와 개발시 사용되는 유용한 분석 도구로서 점차 그 자리를 확립하게 되었다. 1960년대 이후로 계속 개발이 이루어져 왔지만 정상상태 모사기에 비해 그 발전 정도가 매우 느리게 진행되어 왔는데 그것은 화학공정의 동특성 모델링에 대한 어려움과 더불어 대규모의 DAE (differential algebraic equation)으로 표현 되며 비선형성, sparseness, stiffness, 불연속 등의 계산상 불리한 특성을 지닌 모델 방정식들을 풀기 위한 알고리즘의 복잡성에 기인한 것이다. 이러한 문제들은 적절한 조처를 취하지 않으면 부정확하고 비효율적인 결과를 초래하며 원하는 모사를 불가능하게 하므로, 계산의 안정성과 효율성에 기초하여 수치적 문제점들에 대처 가능한 적분 루틴의 개발이 요구된다.

본 연구는 비정상 상태 모사에 있어서 가장 흥미있는 문제 중의 하나인 불연속을 극복하는 알고리즘을 개발하는 것으로 현재 개발 단계에 있는 범용 비정상상태 모사기의 적분기와 루틴 제어기의 일부를 보강하였다. 불연속에 대한 연구는 그 특성을 구명하는 것을 시작으로 하여, 불연속 상황을 발견하여 그것이 발생하는 시간을 정확히 구해내고 그 점을 기점으로 다시 적분을 개시하는 방법들에 대하여 단계적인 연구가 진행되어 왔다. Gear등[3]이 불연속 접근처에서 LMS(linear mutistep) 방법과 single step 방법인 Runge-Kutta 방법을 연결해 주는 RK-stater를 개발하고, Kuru등[6]이 적분간격(step-size)을 변수로 취급하여 모든 방정식들과 함께 푸는 방법(ASCEND II)을 적용하였으며, Pantelides등[7]이 적분경로를 따른 탐색에 의해 불연속 점에 정확히 일치하는 적분간격을 구해 내는 locking mechanism을 구현하였다(SPEEDUP). 대부분의 모사기에는 불연속 처리를 위한 특별한 루틴이 부가되어 있다.

또한 공정을 구성하는 모듈 사이에 상당한 동특성 차이를 가질 경우 latency를 이용하여 불필요하고 과도한 연산을 줄일 수 있다. 동특성이 느린 공정에 대해서는 explicit 적분 방식을 사용하거나 predictor 계산만으로 연산을 하는 방법[6]이 제안되고 있다. 본 시스템에 있어서는 기본 구조로 채택하고 있는 sequential-clustered 구조를 활용하여 특정 unit를 고립화시키는 방식으로 latency를 이용하여 stiff의 완화를 꾀하였다.

2. 불연속 (**discontinuity**)

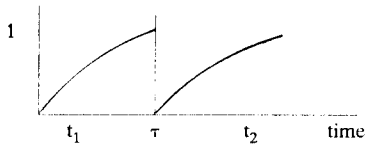
화학공정 모사에 있어서 빈번한 불연속의 원인은 임의의 disturbance의 도입, 조업 상황에 따른 조건의 변화, 위험한 상황에서의 제어 전략의 변화, start-up과 shut-down등의 batch 공정의 조업 절차 상의 변화 등이다. 이러한 상황은 미분식 dy/dt=f(y,t)의 RHS(right-hand-side)항의 불연속을 초래한다. RK류의 식을 사용하면 해결이 수월하지만, 과거 값에 영향을 많이 받는 LMS 방법은 취약성을 드러낸다. 하지만, stiff-integrator로서 가장 좋은 것으로 보고되어 있는 BDF 방식의 Gear method를 기본으로 채택하였기 때문에 LMS 방법에 적용 가능한 알고리즘이 요구된다.

LMS 방식에서는 불연속이 발생하면, 불연속 이전에 구축되어진 polynomial interpolating points에 의해 stream 변수의 표현이 불가능하기 때문에 적분기에서의 corrector 연산이 반복적인 실패를 거듭하고, jacobian matrix의 재계산이 행해지며 step-size의 상당한 감소로 인한 과도한 연산을 행하게 되는 결과를 보인다.

불연속은 explicit와 implicit로 양분할 수 있는데, explicit 불연속은 사용자가 지정하는 forcing function이나 혹은 시간에 따른 함수 형태로 명확히 그 불연속 점을 알 수 있는 경우이고, implicit 불연속은 ODE 상태 변수나 대수식의 대수 변수의 불연속을 의미한다.

2.1 Explicit 불연속 (time event)

불연속 값을 제공하는 변수가 시간의 함수 형태로 주어진 경우로서 언제 불연속이 일어나게 될지를 미리 알수 있는 경우이다. 다음의 그림에서 그 간단한 예를 보이고 있다. 어떤 단위공정으로 입력되는 feed stream의 조성이 정해진 시간에 따라 불연속으로 변하는 경우를 가정해보자. 초기의 한 성분의 조성을 다음과 같이 시간의 함수로 놓고 일정한 시간 τ 에 불연속을 도입하는 경우 다음 그림의 경우와 같다.



$$x(t) = 1.0 - \exp(-\tau) \qquad 0<t<\tau$$

$$x(t) = 1.0 - \exp(-(t-\tau)) \qquad t > \tau$$

Fig. 1. Explicit Discontinuity at time τ

위와 같은 explicit discontinuity를 극복하기 위해서는 다음과 같은 방법이 적절하다. 즉, 적분 스텝이 성공적으로 수행되었을때 다음 적분 스텝을 시도하기 전에 explicit discontinuity의 활성화를 점검한다. 즉 제안된 다음 적분시간 사이에 불연속점이 포함되는 지를 검토하여, 불연속이 포함되면 적분간격을 축소하여 정확한 불연속점에 다음 적분시간의 끝을 일치시킨다. 다음으로 한단계 적분을 시행한 후 불연속점에서 다시 적분기를 재가동 시킨다. 위의 그림에서 현재 적분이 시간 t1까지 진행되었고 제안된 적분간격, h_{next}가 t2까지 적분을 시도하려 한다. 이때 불연속점 τ가 이 구간에 포함되므로 h_{next}를 조절하여 적분 끝을 정확히 τ에 일치시킨다. 결국 불연속 점까지 간단하게 적분이 시행되고 나면, τ점에서 적분기를 재가동시킨다.

Fig. 2에서 보여주는 바와 같이, explicit discontinuity를 극복하는 방법을 사용하지 않으면 적분기는 여러번의 실패를 거듭하여 점차 적분간격을 축소하여 불연속 점을 탐지하여 적분차수 1로 불연속 점을 지나갈 수 있다. 그러나 이러한 과정은 계산상 아주 비효율적이다. 위에서 설명한 방법을 사용하면 적분기는 항상 불연속을 만나지 않게 된다. 이때에도 한가지 문제점은 불연속 점에서 재시작을 할 경우 적절한 적분간격을 다시 정하는 문제가 남아있다. 물론 아주 작게 주면 항상 오차 시험을 통과 할 수 있지만 이 또한 계산의 효율성에 반대되는 경우가 된다. 본 모사기에서는 재시작시 step change에 상당하는 action이 없을 때는 10^{-3} 정도로, 그러한 action이 존재할 때는 10^{-6} 정도로 적분간격의 초기치를 설정해 주도록 되어 있다.

Algorithm-Explicit

```
while ( time < finish time ) do
  if ( RESTART ) then
    SET RESTART = false value
    SET Integration order = 1
    SELECT step size small enough to pass error test
    START Initial value problem
  else
    CHECK if discontinuity is activated
    If active then
      SET step size for next time to match
      the exact discontinuity point
      IMPLEMENT one step integration
      SET RESTART = true value
    else
      CONTINUE integration
    endif
  endif
enddo
```

Fig. 2. Pseudocode for implementing explicit discontinuity handler

2.2 Implicit 불연속 (state event)

일반적으로 불연속이 발생하는 시간을 미리 알 수 없는 경우를 말하는데 대개 불연속을 발생시키는 조건이 적분결과 나오는 ODE 상태변수들로 나타내어 진다. 예를 들어 반응기안의 반응물의 온도가 얼마 이상이면 어떠한 조작을 취하려 한다든지, 또는 반응물의 농도가 얼마이면 반응기의 조작조건을 변경하는 예를 들 수 있다. 이들의 경우는 불연속의 조건을 온도나 농도와 같은 상태변수로 나타내고 있기 때문에 실제로 어느 시간에 불연속을 도입하게 될지 미리 알 수 없게 된다. 항상 적분기가 불연속을 당하지 않도록 조처하려면, 매 적분스텝마다 implicit 불연속의 활성화를 결정한다. 이미 불연속의 조건을 만족하면 정확한 불연속점을 찾기위하여 적분경로를 따라 탐색을 한다. 이러한 탐색 과정은 f(x)=0의 해를 구하는 방법과 유사하다. 불연속 점을 찾은다음 현재 수행했던 적분을 취소하고 다시 불연속 점까지만 적분을 시행하고 explicit 경우와 유사하게 계속 진행시킨다. 다음의 예와 알고리듬을 통하여 자세한 방법을 검토하였다. Fig. 3은 임의의 ODE 상태변수 y의 값이 y_{set}에 도달할 때, 불연속을 도입하는 경우를 나타낸다. 현재 적분이 t_c까지 완료 되었고 이때 불연속 조건을 검토한 결과 y가 지정된 y_{set}을 통과하여 불연속의 활성화를 초래하게 되었다. 적분기는 시간 τ에서 불연속을 도입하여야 하는데 이를 무시하고 t_c 까지 진행 하였으므로 정확한 모사가 이루어진 것이 아니다.

따라서 한 적분구간에서 정확한 불연속 점을 찾기 위하여, 다음의 식을 만족시키는 시간 t를 구해낸다. 이 시간이 곧, 불연속 점(τ)이 된다.

$$y_{set} = I(t, NQ, \underline{y}) \quad , \quad t_{old}<t<t_c \qquad (1)$$

이때 I 는 interpolation formula를 표시하며, 다음과 같이 정의된다.

$$I = \sum_{j=0}^{nq} y(i,j+1)*S^{**}j \quad , \quad S=-(t-t_c)/(t_c-t_{old}) \qquad (2)$$

이때 $\underline{y}$ 는 Nordsieck history array이다.

식 (1), (2)를 이용하여 불연속 점 τ를 구한다음, 시행했던 한 구간의 적분을 모두 취소하여 적분 시작위치를 t_{old}로 옮긴다음 정확히 불연속 점 까지의 적분을 시행한다.

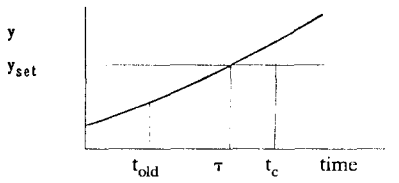


Fig. 3 Implicit discontinuities at time τ

DGEAR 적분기는 Nordsieck array[3]를 사용하여 predictor-corrector 계산을 행하는데 이 Nordsieck array는 적분 차수와 적분간격의 변화를 조정하기가 용이하다. 위에서 언급한 알고리즘을 적용하기 위해서는 step-size 변화시 요구되는 모든 매개변수(parameter)들을 다시 조정하고, array Z_n를 rescale시킨다.

$$Z_n=[y_n, hy_n^{(1)}, (h^2y_n^{(2)})/2!, (h^3y_n^{(3)})/3!, \ldots, (h^qy_n^{(k)})/k!]$$

이때 적분차수(q)는 1로 setting되고, 구해진 step-size는 truncation error $\underline{e}_{t,q} = C_q(h^{q+1}y^{(q+1)})/(q+1)!$ 하에서

$$[\ (1/n)\Sigma(e_{t,j}/y_{max,j})^2\ ]^{1/2} \ .<=. \ tol \qquad (3)$$

error test를 만족시켜야 한다.

Predictor : $Z_n^P = DZ_{n-1}$ (4)

Corrector :

$$Z_n = Z_n^P + E_nL \qquad (5)$$

$$E_n = \sum_{s=0}^{m-1}(I-hb_0J^*)^{-1}F_n^{(s)} \qquad (6)$$

$$F_n^{(s)} = F(Z_n^{(s)}) = hf(y_n^{(s)}, t_n) - hy_n'^{(s)} \qquad (7)$$

$$hy_n'^{(s+1)} = hy_n'^{(s)} + (I-hb_0J^*)^{-1}F_n^{(s)} \qquad (8)$$

$$y_n^{(s+1)} = y_n + L_0(hy_n'^{(s+1)} - hy_n'^{(s)}) \qquad (9)$$

s = iteration counter

corrector 계산의 식 (7),(8),(9)을 통해 수렴조건을 만족시키게 되면 식(5)의 Z_n을 갱신한다. 불연속 처리를 위한 방법을 적분기에 적용시키면 불연속 점 근처에서 거듭되는 function evaluation과 jacobian evaluation의 수를 줄일 수 있다. 근본적으로 explict 와 implicit의 두 경우의 대처 방법은 동일하다. 다만 미리 불연속 점을 알고 있는 경우와 불연속 점을 찾아야 하는 점이 다를 뿐이다.

Algorithm-Implicit

if current y values satisfy discontinuity conditions then
  SEARCH discontinuity point
  CANCEL one step integration
  SET integration step size to hit discontinuity point
  IMPLEMENT one step integration
  SET RESTART = true value
  GOTO EXPLICIT algorithm
else
  CONTINUE integration
endif

Fig. 4 Pseudocode for implementing implicit discontinuity handler

2.3 알고리즘의 구현과 확장

Fig. 5는 본 모사 시스템의 개략적인 구조와 연결된 불연속 처리 루틴을 보인다. 공정의 제어시 빈번히 발생하는 explicit 불연속은 특정 unit에 관계없이 시스템 전체를 통괄해 주는 **Supervisor**와 연결된**Handler**를 통해 처리된 후, **Action**만 특정 unit에 작용한다. Implicit 불연속 또한 **Supervisor**의 제어를 받지만 disturbance가 도입되거나 특정한 조업 수행을 요구하는 unit 모듈에 독립적인 단위 모듈로 부합된다. 이러한 implicit 불연속 모듈은 불연속 활성화 조건을 검색하는 **Check** 부분과 원하는 action을 취하는 **Action** 부분으로 나뉘어지는데, 이 모듈은 input data를 **Interpreter**가 해석하여 자동적으로 만들어진다. **Action** 부분의 실행 명령은 논리 연산자인 IF/AND/OR/NOT로 그 조건이 구현된다.

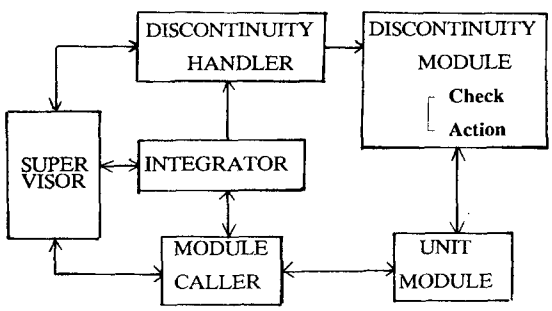


Fig. 5 Block diagram of system structure

모사기는 startup과 shutdown 등의 다양한 batch 공정에 대한 시뮬레이션도 수행하게 되는데 이때 time event와 state event, 즉 불연속 처리 루틴을 연결하여 각각의 조업 절차에 따라, 모사하는 unit의 연결 상태의 변화와 그에 뒤따르는 방정식 수와 초기 치의 재 설정, 연산 순서의 재 배열, 장치 벡터와 stream 벡터의 크기 설정 등이 다시 이루어짐으로서 batch 공정으로의 확장이 가능하다. [2]

3. Latency

Flowsheet 상에 동적으로 둔감하거나 아예 동특성이 없는 units가 존재 하는 것을 말한다. 이러한 latency가 존재하는 원인은 다음과 같다.

1) unit들의 time constant 차이

2) 공정 상에 도입된 disturbance의 한계적인 영향

3) 특정 unit에 대한 control action의 한계적인 작용

결국 latency는 근본적으로 stiff 시스템을 이루고 있는 것과 거의 동일한 의미가 된다. Stiff 시스템이란 구성 ODE 방정식들의 시간 상수의 크기가 매우 커다란 차이를 보임으로서, 적분간격이 accuracy에 의해 결정되는 것이 아니라 stability에 의해 결정되는 시스템을 의미한다. 따라서 동특성이 비슷한 식들끼리 풀게 되면 stiff 문제가 덜 발생하게 되며, 이러한 latency를 발견하여 이용하면 상당한 연산 효율을 증가시킬 수 있다. 즉, 동특성이 느린 공정에 대해서는 explicit 적분방식을 사용하거나, LMS 방식을 쓰고 있을때는 predictor 계산 만으로 모사를 하면 된다. 또는 spline collocation의 경우에는 다른 units 에 대하여 다른 grid 밀도를 주게 된다.[6]

일반적으로 latency라 함은 EOS(equation oriented system)에서 출발한 것으로 동적으로 민감한 단위공정의 식들과 상당히 둔감한 단위공정의 식들을 동시에 모아놓으면 전체적으로는 꽤 까다로운 문제가 된다. 즉 간단히 풀 수 있었던 단위공정도 어려운 부분과 결합되어 전체적으로 어려운 문제를 이루기 때문이다. 그래서 EOS에서는 계산의 효율을 높이기 위하여 전체 Jacobian matrix 중에서 latency가 발견된 부분에 대해서는 update를 하지않거나 LU-factorization부분의 pivoting을 생략하여 계산효율을 얻으려 하는 시도를 하고 있다[ASCEND]. 그러나 본 연구의 기본 구조인 sequential clustered system에서는 빠른 응답을 보이는 부분은 하나 또는 몇개의 단위공정으로 고립화되기 때문에 stiff 문제를 제거하거나 줄일 수 있게 된다.

3.1 Latency 적용 방법

수학적 방법이 아닌 clustering 기법으로 latency를 이용하기 위해 적절한 cluster의 크기 선정, 즉 unit들의 grouping에 대한 정성적인 제안 사항을 사용자에게 제시하도록 pre-processor를 개발하였다. 모사를 위한 input data와 disturbance의 도입, 초기 설정 치를 가지고 해석한다. 각 unit의 time constant에 가장 큰 영향을 미치는 변수는 hold-up으로서, 이 변수 초기 치의 상대적인 값을 비교하여, 그 차이가 적은 unit들을 한 cluster안에 위치하게 하며 disturbance가 도입될 경우 material flow 정보에 따라 interconection이 밀접한 unit들을 한데 grouping한다. 하지만, time constant의 상대적인 값의 차이가 적을 지라도 process topology를 무시하면서 바로 옆에 근접한 unit가 아닌 것끼리 grouping하는 것은 sequential modular 구조에서는 불가능한 일이다.

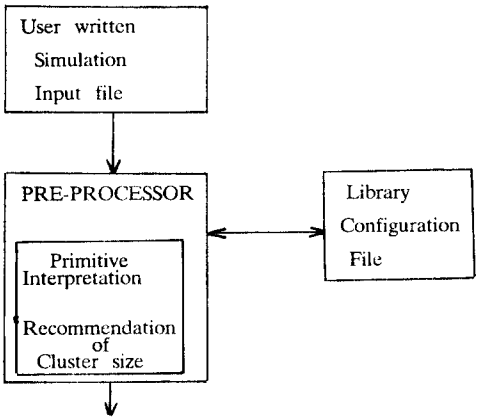
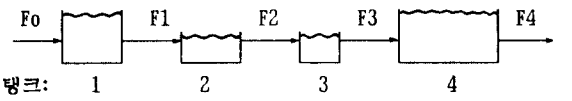


Fig. 6 Structure of Pre-Processor

4. 적용사례

4.1 Four Tank Problem [1]

Fig. 7은 각 stream 마다 유량(F)이 시간의 함수로 나타내어지고 feed 조성이 Fig. 1 의 형태로 τ=1.9 시간에서 disturbance를 보인다. 구성하는 가상 공정은 간단하지만 stiffness ratio가 1.0E+6에 달하는 매우 stiff한 경우로 풀기 어려운 예제이다. Table 2는 보통의 적분기와 개선된 적분기를 사용하였을 경우를 비교한 것으로 모사 시간에 따라, 그 정도가 달라지기는 하겠으나 불연속 점 근처에서 40%에서 50%에 달하는 연산이 행해지던 것이 10% 정도로 감소를 보인다.



각 stream의 유량

F0=F1=100.
F2=100.-0.03t
F3=100.-0.04t
F4=100.-0.05t (moles/hr)

각 Tank의 초기 holdup

M1=10.
M2=0.1
M3=0.01
M4=50000

Fig. 7 Four Tank Flowsheet

Table 1. Four Tank Problem

total simulation time=2.5 (hr)
tol=1.0E-6

	total evaluation no.		near the discontinuity	
	nfe	nje	nfe	nje
conventional	1100	118	432(39%)	57(48%)
modified	757	74	80(11%)	11(15%)

Table 2. Isothermal Flash Vapotization

total simulation time=18 (min)
tol=1.0E-5

	total evaluation no.		near the discontinuity	
	nfe	nje	nfe	nje
conventional	439	80	229(52%)	49(61%)
modified	283	46	55(19%)	13(28%)

4.2 Isothermal Flash Vaporization

Fig. 8은 모사 시작 후 1 분 후에 C_2H_4와 C_2H_6의 도입으로 feed 조성의 step 변화가 일어나는 isothermal-flash vaporization 시스템이다.

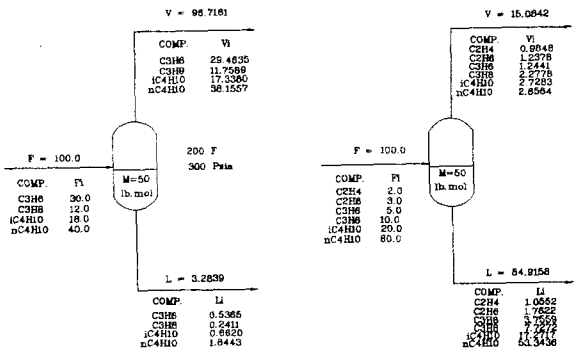


Fig. 8 Isothermal Flash Vaporization

이 flash는 완전히 혼합되며, vapor hold-up은 무시 가능하며, molar liquid hold-up이 일정하게 유지되는 간단한 간단한 모델이지만, explicit 불연속의 도입으로 vapor(V)와 liquid(L)의 양이 각각 96.7161에서 15.0842로 3.2839에서 84.9158로의 급격한 변화를 보인다. 모사 시작 후 0.75분 내에 총 vapor flow rate의 98%에 달하는 변화가 일어난다. 이 예는 disturbance의 도입이 시스템을 stiff 하게 변화시킨 것으로 Table 2.에서 보이는 바와 같이 개선된 적분기를 사용한 결과, 불연속 점 근처에서 function evaluation수와 jacobian evaluation수의 감소를 보인다.

4.3 Double Effect Evaporator [7]

Fig. 9의 evaporator 공정은 물에 glycol 3.5%가 섞인 혼합물이 evaporator로 유입되어 물을 증발시켜 glycol의 농도를 높히는 예로서 feed pump에서 leaking이 발생함에 따라 시스템이 어떤 responce를 나타내는가를 알아보는 것이다. 이 예제는 feed pump leakig으로 표현되는 explicit 불연속 발생과 feed flow rate 감소로 뒤따르는 Evap1에서의 hold-up의 감소, steam과 열교환되는 heat area의 변화로 표현되는 implicit 불연속 상황의 예로서, 불연속 점 근처에서 발생되는 system의 환경 변화는 급격한 jump를 보이는 것이 아니기 때문에 적분기는 큰 어려움을 겪지는 않지만 위의 두 예제와는 달리 처리 루틴을 통과하지 않으면 원하는 모사 자체가 불가능한 그러한 예이다. 모사시 사용되는 모듈들은 VALVE, PI-CONTROLLER, CENTRIFUGAL PUMP, EVAPORATOR이고 disturbance의 영향이 시스템에 신속하고 밀접하게 작용하고, 풀어야 하는 미분식의 갯수가 적으므로 모든 unit를 하나의 cluster로 하여 한꺼번에 적분하는 것이 효율적이다.

개선한 적분기를 사용하면 위의 경우뿐만 아니라 multiple 불연속의 자동적인 극복이 가능하고, 원하는 모사

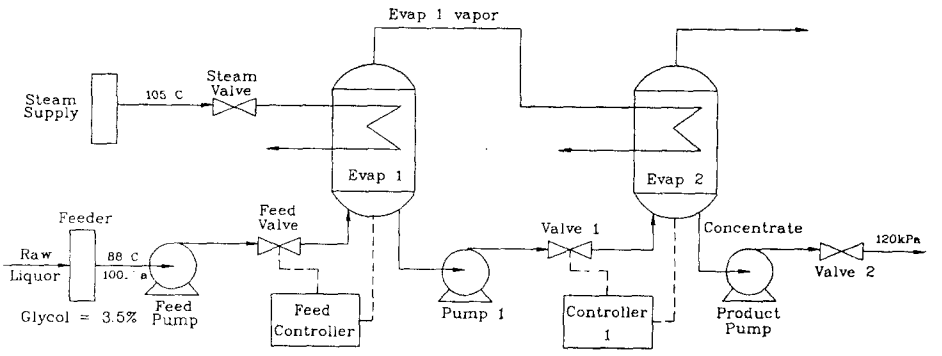


Fig. 9 Evaporator Flowsheet

를 원활히 시행할 수 있을 뿐아니라 연산시 효율성도 개선시킬 수 있다.

5. 결 론

화학공정의 비정상상태 모사는 정상상태 모사에 비해 여러가지 어려운 수치적(numerical) 문제점들을 갖고 있다. 따라서 범용 모사기를 개발해 나가는데 있어서도 수반되는 어려움은 그 배가 될 것이다. 기본으로 하는 구조 -equation oriented system 또는 sequential modular system-가 무엇이냐에 따라 요구되는 처리 루틴의 알고리즘도 변화할 것이며, 풀고자 하는 그 문제의 특성에 따라 쓰이는 방법이 각각 장단점을 갖고 있음으로, 어느 것이 비교우위에 있는지 판단 기준을 설정하기가 모호하다. 하지만, 본 모사기의 기본 구조인 sequential clustered system은 적절한 cluster-size만 설정되면 stiff화를 유발시킬 가능성을 완화시킴으로서 latency를 이용할 수 있고, 한 cluster내에 있는 unit들을 한꺼번에 적분함으로서 적절한 coupling이 도입됨으로 계산의 정확성과 안정성 향상에 도움을 준다. 그러므로, cluster-size를 정하는 판별 기준의 확장과 보강이 요구되는 것은 당연한 일일 것이다.

한편, 사용자의 편리를 위한 도구의 개발이 절대적으로 필요하다. 항상 특수한 상황에 대한 모사를 원하기 때문에 moldule library에 존재하는 module 만으로는 어떤 상황에 대한 구현이 쉽지 않으므로, 사용자가 필요에 따라 module을 쉽게 작성할 수 있도록 simulation language의 개발이 빨리 이루어져야 할 것으로 본다.

감 사

본 연구를 수행하는데 도움을 주신 주식회사 유공과 과학기술원의 송 형근 박사님께 깊은 감사를 드립니다.

참고 문헌

1. Fagley, J.C., "Flexibility and Efficiency in Modular Dynamic Chemical Plant Simulation," Ph.D. Thesis, University of Michigan, (1984).
2. Fletcher J.P. and J.E. Ogbonda, "A Modular Equation-Oriented Approach to Dynamic Simulation of Chemical Processes," Comput. Chem. Eng., Vol. 12, No.5, 401-405 (1988).
3. Gear, C.W., "Numerical Initial Value Problems in Ordinary Differential Equations," Prentice-Hall, Englewood Cliffs, New Jersey (1971)
4. Hillestard, M. and T. Hertzberg, "Dynamic Simulation of Chemical Engineering Systems by the Sequential Modular Approach," Comput. Chem. Eng., Vol. 10, No.4, 377-388 (1986).
5. Hillestad. M and Hertzberg. T, "Convergence and Stability of the Sequential Modular Approach to Dynamic Process Simulation," Comput. Chem. Eng., Vol. 12, No.5, 407-414 (1988).
6. Kuru, s. and A.W. Westerberg, "A Newton-Raphson Based Strategy for Exploiting Latency in Dynamic Simulation," Comput. Chem. Eng., Vol. 9, No.2, 175-182 (1985).
7. Pantelides, C.C., "SPEEDUP-Recent Advances in Process Simulation," Comput. Chem. Eng., Vol. 12, No. 7, 745-755 (1988).
8. Patterson, G.K. and R.B. Rozsa, "DYNSYL : A General-Purpose Dynamic Simulator for Chemical Processes," Comput. Chem. Eng., Vol. 4, 1-20 (1980).
9. Smith G.J. and W. Morton, "Dynamic Simulation using an Equation- Orientated Flowsheeting Package," Comput. Chem. Eng., Vol. 12, No.5, 469-473 (1988).