Control System Design Package를 위한 Dead Time의 처리에 관한 제안

이 지 태
경북대학교. 화학공학과

1. 서론

Dead time은 유체 이동을 수반하는 공정에서는 실제적으로 나타나며, 이런 실제적인 시스템 외에도 우리가 공정을 dead time을 포함하는 모델로 근사하게 되면 나타나게 된다. 이 dead time이 매우 빈번히 공정모델에 나타나지만 이를 잘 표현하고 처리하는 방법은 아직 보편화되어 있지 못하다. 이런 까닭으로 dead time은 주로 Pade근사를 통하여 선형시스템의 기본 모델로 변형하여 처리를 하게 된다. 그러나 Pade근사가 잘 통하지 않는 경우가 있다. 예를 들어 다음 증류탑공정

$$G(s)=\begin{bmatrix} \frac{-1.986\exp(-0.71s)}{66.67s+1} & \frac{5.24\exp(-60s)}{400s+1} & \frac{-5.984\exp(-2.24s)}{14.29s+1} \\ \frac{0.0204\exp(-0.59s)}{(7.14s+1)^2} & \frac{-0.33\exp(-0.68s)}{(2.38s+1)^2} & \frac{2.38\exp(-0.42s)}{(1.43s+1)^2} \\ \frac{0.374\exp(-7.75s)}{2222s+1} & \frac{-11.3\exp(-3.79s)}{(21.74s+1)^2} & \frac{-9.811\exp(-1.59s)}{11.36s+1} \end{bmatrix}$$

에 다중루프 제어기

$$C(s)=diag\left\{-\frac{24.9s+4.40}{s},-\frac{6.08s+0.507}{s},-\frac{0.303s+0.0247}{s}\right\}$$

가 장착된 시스템의 모사를 하여보면, dead time의 (5,5) Pade까지는 모사 오차가 줄어드는데 좀더 정확한 모사를 위해 Pade근사 차수를 그 이상으로 늘리면 계산오차가 더 나빠지는 것을 볼 수 있다[1]. 이는 공정의 (1,2) 요소에 있는 매우 큰 dead time 때문에 일어나는데 dead time의 다른 근사로도 해결할 수 없다. 또 다른 Pade 근사의 어려움은 근사 선형시스템의 모델차수가 매우 커질 수 있다는 것이다. 위의 예에서(5,5) Pade근사를 하면 전 시스템의 차수는 61차가 된다. 이 정도는 쉽게 처리할 수 있으나 시스템의 크기가 커지면 어려움이 있게 된다.

또한 공정 제어시스템의 자료를 입력하고 Pade근사를 하며 realization하여 모사를 위한 상태방정식 형태를 얻는 과정이 매우 번거롭다. Simulink같은 block형태의 모사 프로그램으로 위의 시스템을 모사할 수도 있지만 이 또한 자료 입력은 매우 번거롭다.

본 연구는 위의 시스템 같은 대규모 dead time을 갖는 시스템의 모사 및 분석을 위한 방법을 제안하고자 한다.

2. Dead time을 갖는 시스템의 자료구조 제안

상태공간 형태의 선형시스템

$$\dot{x}=Ax+Bu$$
$$y=Cx+Du$$

은 네 개의 행렬 (A, B, C, D)로 표현할 수 있다. 이 표현은 매우 편리하여 쉽게 여러 상태공간 형태의 시스템들이 연결된 경우에도 확장되며, 분석을 위한 빠르고 효과적인 방법들도 구비되어 있다. 행렬(A, B, C, D)를 계속 가지고 가는 대신에

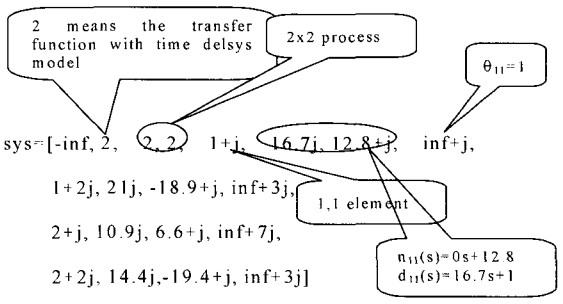
$$S=\begin{bmatrix} A & B & n \\ C & D & 0 \\ 0 & 0 & -\inf \end{bmatrix}$$

처럼 축약하여 하나의 행렬 S로 처리하는 방법이 개발되어 있다. 여기서 inf는 무한대를 나타내는 MATLAB에서 사용하는 변수이다.

산업현장의 공정모델은 주로 다음과 같은 시간지연을 갖는 전달함수의 행렬로 주어진다.

$$G(s)=\begin{bmatrix} \frac{12.8\exp(-s)}{16.7s+1} & \frac{-18.9\exp(-3s)}{21s+1} \\ \frac{6.6\exp(-7s)}{10.9s+1} & \frac{-19.4\exp(-3s)}{14.4s+1} \end{bmatrix}$$

이를 나타내는 자료구조를 제안한다. 행렬의 각 요소 $g_{ij}(s)=n(s)\exp(-qs)/d(s)$를 나타내는데 i,j, n(s) 및 d(s)자료, q가 필요하다. 예를 들어 $g_{11}(s)=12.8\exp(-s)/(16.7s+1)$를 표현하는데 i,j,n(s),d(s),q의 1, 1, [12.8], [16.7 1], 1를 기억해야 한다. 이를 위해 [1+j, 0+16.7j, 12.8+1j, inf+1j]를 이용한다. 첫 1+j는 1,1을 의미하고, 0+16.7j, 12.8+1j는 [0 12.8], [16.7 1]의 n(s)와 d(s)를 의미하며, 마지막의 inf+1j는 자료 끝과 q을 의미한다. 각 요소를 1차원 벡타로 결합하고 G(s)의 차원을 결합하여 위의 공정을 다음과 같이 표현한다.



이 자료구조는 공정을 결정하는 모든 자료를 가지고 있으며 매우 간결하다. 이런 형태의 다른 공정 표현을 결합하면 매우 편리한 도구가 될 것이다.

3. 시스템 자료처리 프로그램

위의 공정 표현에 근거하는 자료 처리 프로그램 le_cs을 작성하였다. 부록에 소스가 첨부되어 있는데 주요 기능은

- Pade근사를 통한 상태변수 공정의 생성
- 상수 공정의 생성
- 시간지연을 포함하는 전달함수 공정의 생성 및 처리
- 문자로 표현되는 공정의 생성
- 공정 시스템의 연결을 통한 복잡한 시스템의 생성이다.

4. Frequency Response 계산 프로그램

부록에 첨부되어 있는 프로그램 le_s2fr은 위의 공정 표현에 근거한 복잡한 시스템의 주파수응답을 계산한다. 한 주파수마다 하나씩 계산을 하기 때문에 상태방정식에 의한 방법에 비해서는 계산 속도는 느리지만 Pade근사 같은 근사를 하지 않아 정밀한 계산을 할 수 있다.

5. Time Response 계산 프로그램

부록에 첨부되어 있는 프로그램 le_step은 위의 공정 표현에 근거한 복잡한 시스템의 계단응답을 계산한다. 시스템의 주파수응답을 먼저 계산하고 이를 FFT방법을 적용하여 계단응답을 얻는다. FFT법을 적용하는데 있어 어려움을 주는 요소들을 Series/Parallel Compensator를 이용하여 제거하였다[2].

6. 예제

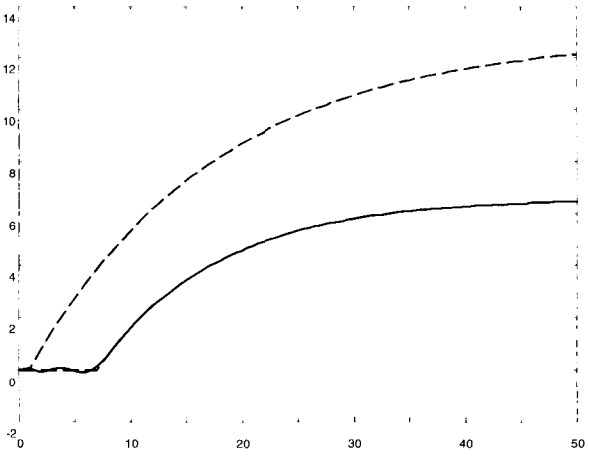
6.1. Time Response계산 비교

화학공정의 제어문제에서 자주 나타나는 공정

$$g_{wb}(s)=\begin{bmatrix}\dfrac{12.8\exp(-s)}{16.7s+1} & \dfrac{-18.9\exp(-3s)}{21s+1}\\ \dfrac{6.6\exp(-7s)}{10.9s+1} & \dfrac{-19.4\exp(-3s)}{14.4s+1}\end{bmatrix}$$

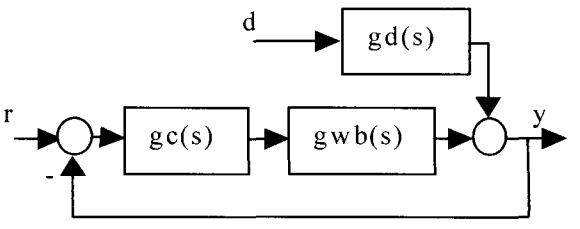
의 개루프 계단응답을 Pade근사를 통한 상태방정식방법과 FFT에 근거한 방법을 비교하였다.

```
gwb=le_cs(2,2);
gwb=le_cs(gwb,1,1,[12.8],[16.7 1],1);
gwb=le_cs(gwb,1,2,[-18.9],[21 1],3);
gwb=le_cs(gwb,2,1,[6.6],[10.9 1],7);
gwb=le_cs(gwb,2,2,[-19.4],[14.4 1],3);

t=0:.5:50;
[a,b,c,d]=le_cs(gwb,5);  % Pade approximation
ypade=step(a,b,c,d,1,t);

yfft=le_step(gwb,1,t);    % FFT method

plot(t,ypade,'-',t,yfft,'--')
```



6.2. 복잡한 시스템의 모사

다음의 시간지연이 있는 다변수 공정의 제어시스템의 모사를 하였다.

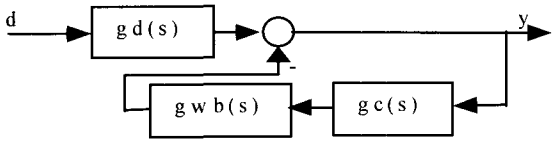


$$gwb(s)=\begin{bmatrix}\frac{12.8\exp(-s)}{16.7s+1} & \frac{-18.9\exp(-3s)}{21s+1}\\ \frac{6.6\exp(-7s)}{10.9s+1} & \frac{-19.4\exp(-3s)}{14.4s+1}\end{bmatrix}$$

$$gc(s)=\begin{bmatrix}0.375(1+\frac{1}{8.29s}) & 0\\ 0 & -0.075(1+\frac{1}{23.6s})\end{bmatrix}$$

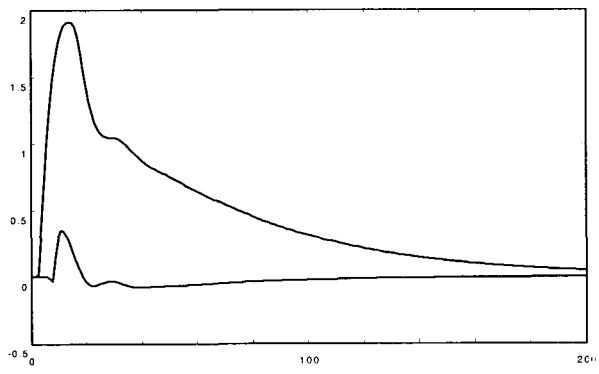
$$gd(s)=\begin{bmatrix}\frac{3.8\exp(-8s)}{14.9s+1}\\ \frac{4.9\exp(-3s)}{13.2s+1}\end{bmatrix}$$

외란 d의 변화에 대한 시간 응답을 보기 위하여 위의 시스템을 다음과 같이 변형한다.



이 시스템은 다음과 같이 해석되고 모사 되어진다.

```
gwb=le_cs(2,2);      % wb process
gwb=le_cs(gwb,1,1,[12.8],[16.7 1],1);
gwb=le_cs(gwb,1,2,[-18.9],[21 1],3)
gwb=le_cs(gwb,2,1,[6.6],[10.9 1],7);
gwb=le_cs(gwb,2,2,[-19.4],[14.4 1],3);

gc=le_cs(2,2);       % controller
gc=le_cs(gc,1,1,[0.375 0.375/8.29],[1 0],0);
gc=le_cs(gc,2,2,[-0.075 -0.075/23.6],[1 0],0);

gd=le_cs(2,1);       % disturbance
gd=le_cs(gd,1,1, 3.8, [14.9 1], 8);
gd=le_cs(gd,2,1, 4.9, [13.2 1], 3);
g1=le_cs(eye(2)); g2=le_cs(gc,gwb,'*');
g3=le_cs(g1,g2,'n'); g4=le_cs(gd,g3,'*');

t=0:1:200; y=le_step(g4,1,t); plot(t,y,'-')
```



7. 결론

시간지연을 갖는 시스템의 기술을 위한 편리한 방법을 제안하였다. 이 자료구조와 프로그램들은 제어시스템 연구의 마지막에 항상 나타나는 모사를 통한 검증 단계 등에 매우 편리하게 사용할 수 있을 것이다.

참고문헌

1. J. T. Lee and T. F. Edgar, "Subspace Identification Method for Simulation of Large Dimensional Closed Loop Systems with Time Delays," *AIChE J.*, vol. 48, pp. 417-420, 2002.
2. J. T. Lee and T. F. Edgar, "Simulations of Stable Non-conventional and Large Dimensional Linear Processes Using the FFT Method," *Computers and Chemical Engineering*, in press, 2004.

부록(MATLAB 프로그램)

```
function [sa,sb,sc,sd]=le_cs(s1,s2,s3,s4,s5,s6)
%
% s=le_cs(a,b,c,d)
%   generate a system matrix of (a,b,c,d).
% [a,b,c,d]=le_cs(s1)
% [a,b,c,d]=le_cs(s1,npade)
%   (a,b,c,d) with Pade approximation of time delays.
%   npade is the Pade approximation order.
% s=le_cs(K)
%   a constant matrix.
%
% s=le_cs(m,1)
%   initiallize a (m x 1) transfer function matrix.
% s=le_cs(s1,i,j,num,den,theta)
%   add/replace the (i,j) element in s1.
% [num,den,theta]=le_cs(s1,i,j)
%   extract the (i,j) element in s1.
%
% s=le_cs(num,den,th)
% s=le_cs(s1,i,j)
%   SISO system of (num,den,th) or (i,j) subsystem of s1.
%
% s=le_cs(s1)
%   s1 is a series of string for the transfer function matrix
%     in the Laplace variable s.
%     ex: s1='diag([exp(-s)/(s+1), exp(-2*s)/(2*s+1)])'
%
```

```
% s=le_cs(s1,s2,op)
%   op=='+'   s=s1+s2          (parallel connection)
%      '-'    s=s1-s2          (parallel subtraction)
%      '*'    s=s2*s1          (series connection)
%      'n'    s=inv(1+s1*s2)*s1 (negative feedback)
%      'p'    s=inv(1-s1*s2)*s1 (positive feedback)
%

% state space model
% s=le_cs(a,b,c,d)
if nargin==4,
  n=length(s1); [l,m]=size(s4);
   [q1,q2]=size(s1); if q1~=q2, error('A is not square'); end
   [q3,q4]=size(s2); if q3~=q1, error('Dimension mismatch in B'); end
   [q5,q6]=size(s3); if q6~=q1, error('Dimension mismatch in C'); end
   [q7,q8]=size(s4);
    if q4~=q8 | q5~=q7, error('Dimension mismatch in D'); end
  sa=[-inf,1,n,m,l];
  if n==0,
    for k1=1:l, sa=[sa, s4(k1,:)]; end
  else
    for k1=1:n, sa=[sa, s1(k1,:)]; end
    for k1=1:n, sa=[sa, s2(k1,:)]; end
    for k1=1:l, sa=[sa, s3(k1,:)]; end
    for k1=1:l, sa=[sa, s4(k1,:)]; end
  end
  return
end
% [a,b,c,d]=le_cs(s1)
% [a,b,c,d]=le_cs(s1,npade)
if nargout==4
  if nargin==1, s2=1; end
  [sa,sb,sc,sd]=le_s2ss(s1,s2);
  return
end

% transfer function model
% s=le_cs(m,l)
if nargin==2, sa=[-inf,2,s1,s2]; return, end
% s=le_cs(s1,i,j,num,den,theta)
if nargin==6,
    if length(find(s1==-inf))>1, error('not a system matrix'), end
    if s1(2)~=2, error('not a transfer function matrix'), end
  sa=s1;
  s23=s2+j*s3;
  bp=[5, find(real(s1)==inf)+1]; ne=find(s1(bp(1:length(bp)-1))==s23);
  if ~isempty(ne), sa(bp(ne):bp(ne+1)-1)=[]; end
  if length(s5)>=length(s4),
    sa=[sa, s23,[zeros(1,length(s5)-length(s4)),s4]+j*s5, inf+j*s6];
  else
    sa=[sa, s23,s4+[zeros(1,length(s4)-length(s5)),j*s5], inf+j*s6];
  end
  return
end
% [num,den,theta]=le_cs(s1,i,j)
if nargout==3,
    if length(find(s1==-inf))>1, error('not a system matrix'), end
    if s1(2)~=2, error('not a transfer function matrix'), end
  s23=s2+j*s3;
  bp=[5, find(real(s1)==inf)+1]; ne=find(s1(bp(1:length(bp)-1))==s23);
  if isempty(ne)
    sa=[]; sb=[]; sc=[];
  else
    s=s1(bp(ne)+1:bp(ne+1)-1); ns=length(s);
    sa=real(s(1:ns-1)); sb=imag(s(1:ns-1)); sc=imag(s(ns));
    while sa(1)==0, sa(1)=[]; end
    while sb(1)==0, sb(1)=[]; end
  end
  return
end

if nargout==1 & nargin==1,
  if ischar(s1), sa=[-inf,3,double(s1)]; return, % string or constant matrix
  else                                           % constant system
    [l,m]=size(s1); sa=[-inf,1,0,m,l];
    for k1=1:l, sa=[sa, s1(k1,:)]; end
  end
  return
end

if nargin==3
  if ischar(s3), sa=[s1,s2,-inf,double(s3)];  % system operation
    [n1,m1]=size(le_s2fr(s1,1.e-6));          % checking error
    [n2,m2]=size(le_s2fr(s2,1.e-6));
    if s3=='+' | s3=='-',
      if n1~=n2 | m1~=m2, error('dimension mismatch'), end
    elseif s3=='*',
      if m2~=n1, error('dimension mismatch'), end
    elseif s3=='n' | s3=='p',
      if m2~=n1 | n2~=m1, error('dimension mismatch'), end
    end
  else                                        % SISO system
    if isempty(find(s1==-inf)),               % transfer function system
      sa=[-inf,2,1,1, 1+j];
      if length(s2)>=length(s1),
        sa=[sa, [zeros(1,length(s2)-length(s1)),s1]+j*s2, inf+j*s3];
      else
        sa=[sa, s1+[zeros(1,length(s1)-length(s2)),j*s2], inf+j*s3];
      end
    else                                      % (i,j) subsystem
      [m,l]=size(le_s2fr(s1,1.e-6));
      ei=zeros(1,m); ei(s2)=1; sa=[s1,[-inf,1,0,m,1,ei],-inf,double('*')];
      ej=zeros(1,l); ej(s3)=1; sa=[[-inf,1,0,1,l,ej],sa,-inf,double('*')];
    end
```

```
    end
    return
end


%-----------------------------------------
function [sa,sb,sc,sd]=le_s2ss(s1,npade)
%
% [a,b,c,d] : state space matrices for the system s1.
% npade is the Pade approximation order of time delay.
%
st=[];
bp=[find(s1==-inf), length(s1)+1];
for k1=1:length(bp)-1
    ip1=bp(k1); ip2=bp(k1+1)-1;
    sp=s1(ip1:ip2);

    if sp(2)==1,
        n=sp(3); m=sp(4); l=sp(5);
        a=zeros(n,n); b=zeros(n,m); c=zeros(l,n); d=zeros(l,m);
        ik1=6;
        if n==0,
            a=[]; b=[]; c=[];
            for k2=1:l, ik2=ik1+m-1; d(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
        else
            for k2=1:n, ik2=ik1+n-1;  a(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
            for k2=1:n, ik2=ik1+m-1; b(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
            for k2=1:l, ik2=ik1+n-1;  c(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
            for k2=1:l, ik2=ik1+m-1; d(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
        end
        st=le_pushs(st,a,b,c,d);
    elseif sp(2)==2,
        m=sp(3); l=sp(4); sa=zeros(m,l);
        bp1=[5, find(real(sp)==inf)+1];
        a=[]; b=[]; c=[]; d=zeros(m,l); na=0; nb=0; nc=0; nd=0;
        for k2=1:length(bp1)-1,
            sb=sp(bp1(k2):bp1(k2+1)-1);
            nsb=length(sb); ni=real(sb(1)); nj=imag(sb(1));
            num=real(sb(2:nsb-1)); den=imag(sb(2:nsb-1)); theta=imag(sb(nsb));
            [a1,b1,c1,d1]=tf2ss(num,den);
            if theta>0,
                [a2,b2,c2,d2]=pade(theta,npade);
                [a1,b1,c1,d1]=series(a1,b1,c1,d1,a2,b2,c2,d2);
            end
            n1=length(a1); na=length(a);
            if n1>0
                a=[a zeros(na,n1);zeros(n1,na) a1];
                b2=zeros(n1,l); b2(:,nj)=b1; b=[b;b2];
                c2=zeros(m,n1); c2(ni,:)=c1; c=[c c2];
            end
            d(ni,nj)=d1;
        end
        st=le_pushs(st,a,b,c,d);
    elseif sp(2)==3,
        error('cannot be converted to the state space model!')
    end

    if sp(2)>10, [a2,b2,c2,d2,st]=le_pops(st); [a1,b1,c1,d1,st]=le_pops(st); end
    if char(sp(2))=='+',
        [a,b,c,d]=parallel(a1,b1,c1,d1,a2,b2,c2,d2); st=le_pushs(st,a,b,c,d);
    elseif char(sp(2))=='-',
        [a,b,c,d]=parallel(a1,b1,c1,d1,a2,b2,-c2,-d2); st=le_pushs(st,a,b,c,d);
    elseif char(sp(2))=='*',
        [a,b,c,d]=series(a1,b1,c1,d1,a2,b2,c2,d2); st=le_pushs(st,a,b,c,d);
    elseif char(sp(2))=='n',
        [a,b,c,d]=feedback(a1,b1,c1,d1,a2,b2,c2,d2,-1); st=le_pushs(st,a,b,c,d);
    elseif char(sp(2))=='p',
        [a,b,c,d]=feedback(a1,b1,c1,d1,a2,b2,c2,d2,1); st=le_pushs(st,a,b,c,d);
    end
end
[sa,sb,sc,sd,st]=le_pops(st);


function sa=le_pushs(s1,a,b,c,d)
%
[n,m]=size(b); [l,m]=size(d);
sp=zeros(1,(n+m)*(n+1)); ik1=1;
if n~=0,
  for k1=1:n, ik2=ik1+n-1; sp(ik1:ik2)=a(k1,:); ik1=ik2+1; end
  for k1=1:n, ik2=ik1+m-1; sp(ik1:ik2)=b(k1,:); ik1=ik2+1; end
  for k1=1:l, ik2=ik1+n-1; sp(ik1:ik2)=c(k1,:); ik1=ik2+1; end
end
for k1=1:l, ik2=ik1+m-1; sp(ik1:ik2)=d(k1,:); ik1=ik2+1; end
sa=[n,m,l,sp,s1];


function [a,b,c,d, sa]=le_pops(s1)
%
n=s1(1); m=s1(2); l=s1(3);
a=zeros(n,n); b=zeros(n,m); c=zeros(l,n); d=zeros(l,m);
ik1=4;
if n==0
  a=[]; b=[]; c=[];
  for k1=1:l, ik2=ik1+m-1; d(k1,:)=s1(ik1:ik2); ik1=ik2+1; end
else
  for k1=1:n, ik2=ik1+n-1; a(k1,:)=s1(ik1:ik2); ik1=ik2+1; end
  for k1=1:n, ik2=ik1+m-1; b(k1,:)=s1(ik1:ik2); ik1=ik2+1; end
  for k1=1:l, ik2=ik1+n-1; c(k1,:)=s1(ik1:ik2); ik1=ik2+1; end
  for k1=1:l, ik2=ik1+m-1; d(k1,:)=s1(ik1:ik2); ik1=ik2+1; end
end
sa=s1(ik1:end);
```

```
function [hs,hs1,hs2,hs3]=le_s2fr(s1,s)
%
% hs : s1(s)
% hs1,hs2,hs3 : the first to third constant matrices for series
%      expansion of system around s
%

st={};
bp=[find(s1==-inf), length(s1)+1];
ns=nargout; if ns==0, ns=1; end
for k1=1:length(bp)-1
   ip1=bp(k1); ip2=bp(k1+1)-1;
   sp=s1(ip1:ip2);

   if sp(2)==1,
      n=sp(3); m=sp(4); l=sp(5); sz=zeros(l,m*ns);
      sa=zeros(n,n); sb=zeros(n,m); sc=zeros(l,n); sd=zeros(l,m);
      ik1=6;
%        retrieve a,b,c,d / find value and derivatives
      if n==0,
         for k2=1:l, ik2=ik1+m-1;sd(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
         sz(:,1:m)=sd;
      else
         for k2=1:n, ik2=ik1+n-1;sa(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
         for k2=1:n, ik2=ik1+m-1;sb(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
         for k2=1:l, ik2=ik1+n-1;sc(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
         for k2=1:l, ik2=ik1+m-1;sd(k2,:)=sp(ik1:ik2); ik1=ik2+1; end
         sa=inv(sa-s*eye(n)); sz(:,1:m)=sd-sc*sa*sb;
         for k2=2:ns, sz(:,k2*m+1:(k2+1)*m)=-sc*sa^k2*sb; end
      end
      st=le_push(st,sz);

   elseif sp(2)==2,
      m=sp(3); l=sp(4); sd=zeros(1,ns); sz=zeros(m,l*ns);
      bp1=[5 find(real(sp)==inf)+1];
%       retrieve num,den,th
      for k2=1:length(bp1)-1,
         sb=sp(bp1(k2):bp1(k2+1)-1);
         nsb=length(sb); ni=real(sb(1)); nj=imag(sb(1));
         num=real(sb(2:nsb-1)); den=imag(sb(2:nsb-1)); th=imag(sb(nsb))
%           series expansion
         [ax,bx,cx,dx]=tf2ss(num,den);
         if isempty(ax), sd(1)=dx;
         else
            ax=inv(ax-s*eye(length(ax)));
            sd(1)=dx-cx*ax*bx; for k3=2:ns, sd(k3)=-cx*ax^k3*bx; end
         end
         sc=conv(sd,[1, -th, th^2/2, -th^3/6]); sc=exp(-th*s)*sc;
         for k3=1:ns, sz(ni,nj+(k3-1)*l)=sc(k3); end
      end
      st=le_push(st,sz);

   elseif sp(2)==3,
      sa=eval(char(sp(3:end))); [m,l]=size(sa); sz=zeros(m,l*ns);
      sz(:,1:l)=sa;
      s=s-1.e-6; sa=eval(char(sp(3:end)));
      s=s+2.e-6; sa=eval(char(sp(3:end)))-sa; s=s-1.e-6;
      if ns>=2, sz(:,l+1:2*l)=sa/(2.e-6); end
      st=le_push(st,sz);
   end
%      operation
   if sp(2)>10, [sb,st]=le_pop(st); [sa,st]=le_pop(st); end

   if char(sp(2))=='+', st=le_push(st,sa+sb);
   elseif char(sp(2))=='-', st=le_push(st,sa-sb);
   elseif char(sp(2))=='*',
      [m,n]=size(sb); [n,l]=size(sa); l=l/ns;
      sz=sb(:,1:n)*sa;
      for k2=2:ns, sz(:,(k2-1)*l+1:l*ns)=sz(:,(k2-1)*l+1:l*ns)+...
            sb(:,(k2-1)*n+1:k2*n)*sa(:,1:l*(ns-k2+1)); end
      st=le_push(st,sz);
   elseif char(sp(2))=='n' | char(sp(2))=='p',
      if char(sp(2))=='p', sb=-sb; end
      [m,n]=size(sb); [n,l]=size(sa);
      sz=sa(:,1:n)*sb;
      for k2=2:ns, sz(:,(k2-1)*n+1:n*ns)=sz(:,(k2-1)*n+1:n*ns)+...
            sa(:,(k2-1)*m+1:k2*m)*sb(:,1:n*(ns-k2+1)); end
      sz(:,1:n)=sz(:,1:n)+eye(n); sc=inv(sz(:,1:n));
      sa=sc*sa; sz=sc*sz;
      for k2=2:ns,
         for k3=1:ns-k2+1
            k23=k2+k3-1;
            sa(:,(k23-1)*m+1:k23*m)=sa(:,(k23-1)*m+1:k23*m)-...
               sz(:,k3*n+1:(k3+1)*n)*sa(:,(k2-2)*m+1:(k2-1)*m);
         end
      end
      st=le_push(st,sa);
   end
end
sa=le_pop(st); [m,l]=size(sa); l=l/ns;
hs=sa(:,1:l); if nargout==1, return, end
hs1=sa(:,l+1:2*l); if nargout==2, return, end
hs2=sa(:,2*l+1:3*l); if nargout==3, return, end
hs3=sa(:,3*l+1:4*l); if nargout==4, return, end

%-------------------------------------------------------
function sa=le_push(s1,s2)
%
[m,l]=size(s2); s3=zeros(1,m*l);
for k1=1:m, s3(1+(k1-1)*l: k1*l)=s2(k1,:); end
sa=[m,l,s3,s1];


function [sa,sb]=le_pop(s1)
```

```
%
m=sl(1); l=sl(2)
sb=sl(m*l+3:length(sl));
sa=zeros(m,l); for k1=1:m, sa(k1,:)=sl(3+(k1-1)*l:2+k1*l); end


function y=le_step(hs,nu,t,nw)
% Step responses for t=0:dt:tf
%
% y=le_step(sys,nu,t)
% y=le_step(sys,nu,t,nw)
%   sys : system
%   nu : step input number
%   t : time (0:dt:tf)
%   nw : number of frequency responses (power of 2)
%       default=512
%       if negative, a series compensation is applied.
% y — step responses
%       y(:,nt) is output at the time t(nt).

if nargin==3, nw=512; end
[ny,nnu]=size(le_s2fr(hs,1.e-6));

isf=0; al=zeros(ny,1);
if nw<0,
   nw=-nw; isf=1;
   [h0,h1]=le_s2fr(hs,1.e-6); al=-h1(:,nu)./h0(:,nu);
end

dt=t(2)-t(1); tf=t(end); nt=length(t);
nfft=2*pow2(nextpow2(nt-1)); tff=nfft*dt;

% maximum frequency
dw=2*pi/tff; nw=pow2(nextpow2(nw)); if nw<nfft, nw=nfft;, end
w=dw*(0:nw); s=j*w; w(1)=1.e-6;

hswt=zeros(ny,nw+1);
for k1=1:nw+1, hsw=le_s2fr(hs,j*w(k1)); hswt(:,k1)=hsw(:,nu); end

y=zeros(ny,nt);
for k1=1:ny
   hw=hswt(k1,:);
   if isf==1, hw=hw.*(al(k1)*s+1)./(al(k1)/10*s+1); end
% size of pulse
   hzero=inf;
   for nz=1:8
      q=tff/2^nz; z=(1-exp(j*q*w(2:nw+1)))./w(2:nw+1);
      hq=[q/2 -j*conj(z)]/tff.*hw;  hz=abs(real(sum(hq;)*2^nz);
      if hzero<hz,
         q=q*2; z=(1-exp(j*q*w(2:nw+1)))./w(2:nw+1);
         hq=[q/2 -j*conj(z)]/tff.*hw; nz=nz-1; break;
      else, hzero=hz;
      end
   end
   nfold=nw/nfft; hz=hq(1:nfft);
   for k=1:nfold-1, hz=hz+hq(k*nfft+1:k*nfft+nfft); end
% pulse response
   hq=[conj(hz)+fliplr([hz(2:nfft) hz(1)]);
   hq=fft(hq(1:2:nfft)+hq(2:2:nfft));
   hz=exp(j*2*pi/nfft*(0:nfft/2-1)); iq=2:nfft/2;
   hq=[real(hq(1)) ...
     real(hq(iq))+imag(hq(iq)).*(1-real(hz(iq)))./imag(hz(iq))];
% step response
   for k=nz:-1:1,
      na=nfft/2^k; hz=[zeros(1,na), hq(1:nfft/2-na)]; hq=hq+hz; end
   hq=[hq hq(nfft/2)];
   y(k1,:)=hq(1:fix(tf/dt)+1);
   if isf==1, y(k1,:)=lsim([al(k1)/10 1],[al(k1) 1],y(k1,:),t)'; end
end
```

저자약력

《이 지 태》

- 1979년 서울대학교 화학공학과 졸업 (학사).
- 1981년 KAIST 화학공학과 졸업 (석사).
- 1986년 KAIST 화학공학과 졸업(박사).
- 1983년~현재 경북대학교 화학공학과 교수.
- 관심분야 : 공정제어.