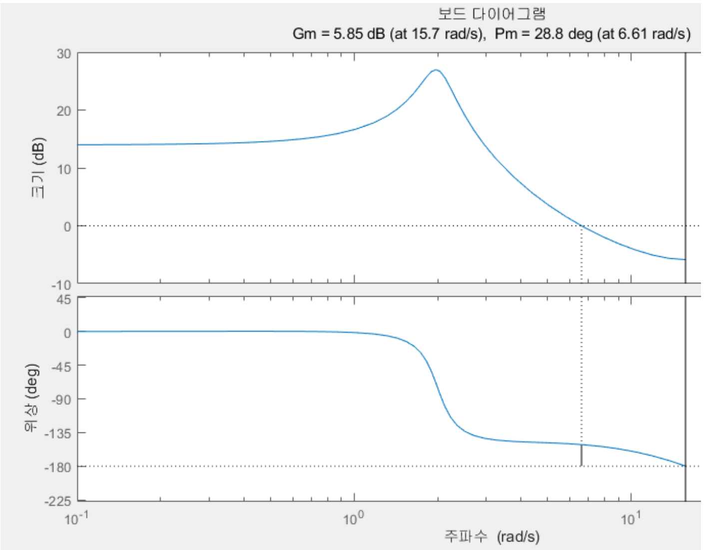
1.a.

우선 제어기를 적용하지 않은 open loop의 bode plot을 봅니다.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

margin(Gd_tf);
```

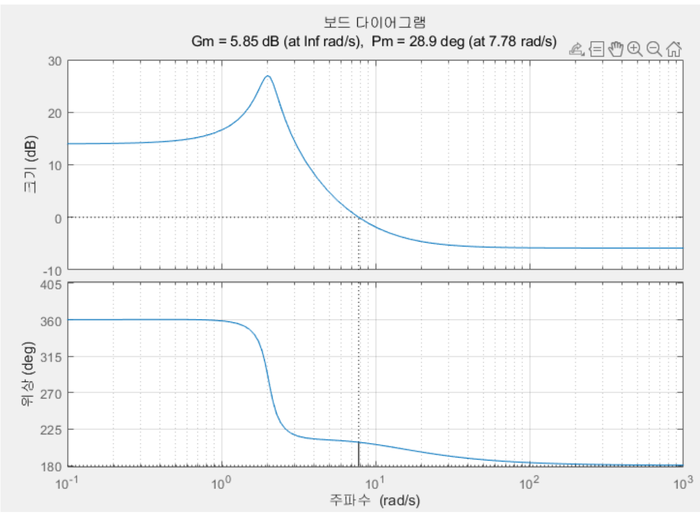
결과 :

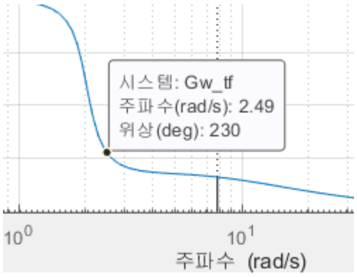


phase margin을 45도 이상 확보하기 위해 w 도메인 Bode diagram을 통해 $\angle G^W(j\omega_{w1}) \approx -180^\circ + 45^\circ(= \phi_m) + 5^\circ = -130^\circ = 230^\circ$ 인 ω_{w1}을 찾습니다.

```
Gw_tf = d2c(Gd_tf,'tustin');
figure();
margin(Gw_tf); grid on;
```

결과 :





따라서 $\omega_{w1} = 2.49$입니다. 이를 사용하여 $\omega_{w0}(< 0.1\omega_{w1})$를 정합니다.

```
om_w1 = 2.49;
k_w0 = 0.005;% 0.1 이하로 설정.
om_w0 = k_w0*om_w1;
```

또한 a_0는 임의로 설정, $\omega_{wp}\left(\leq \frac{\omega_{w0}}{a_0|G^W(j\omega_{w1})|}\right)$는 제시된 범위에 맞춰 임의로 설정합니다.

```
a0 = 1.2; % 임의로 설정
k_wp = 0.3; % 1이하로 설정
om_wp = k_wp*om_w0/(a0*exp(20.1/20));
```

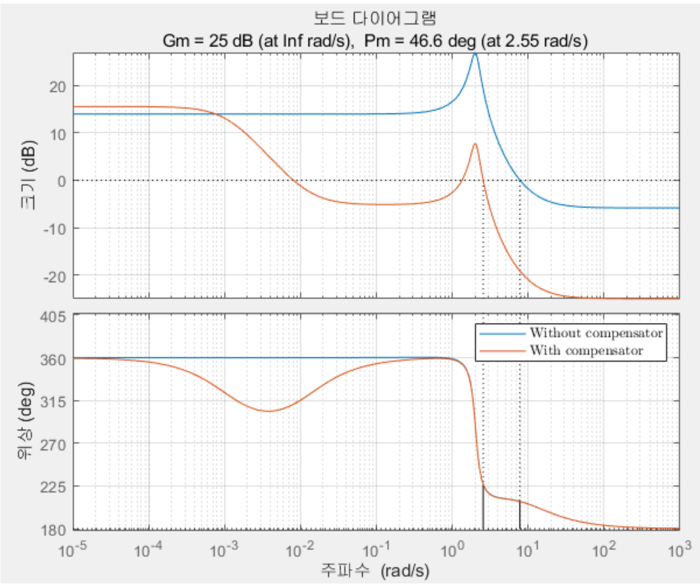
이제 설정한 상수들로 phase lag 보상기를 구성합니다.

```
Dw_lag_tf = a0*tf([1/om_w0 1], [1/om_wp 1]);
```

보상이 잘 되었는지 확인하기 위해 보상기를 적용하기 전과 후를 비교합니다.

```
figure();
margin(Gw_tf); hold on;
margin(Gw_tf * Dw_lag_tf); grid on;
legend('Without compensator', 'With compensator', 'Interpreter', ' latex');
```

결과 :



Phase margin이 45°이상임을 알 수 있습니다.

z 도메인에서의 보상기를 출력해봅니다.

```
% Dw_lag_tf를 이산시간으로 변환
Dd_lag_tf = c2d(Dw_lag_tf, T, 'zoh');
```

```
Dd_lag_tf

fprintf('Kc = %f\n', Dd_lag_tf.Numerator{1}(1));
fprintf('z0 = %f\n', -Dd_lag_tf.Numerator{1}(2)/Dd_lag_tf.Numerator{1}(1));
fprintf('zp = %f\n', -Dd_lag_tf.Denominator{1}(2));
```

결과 :

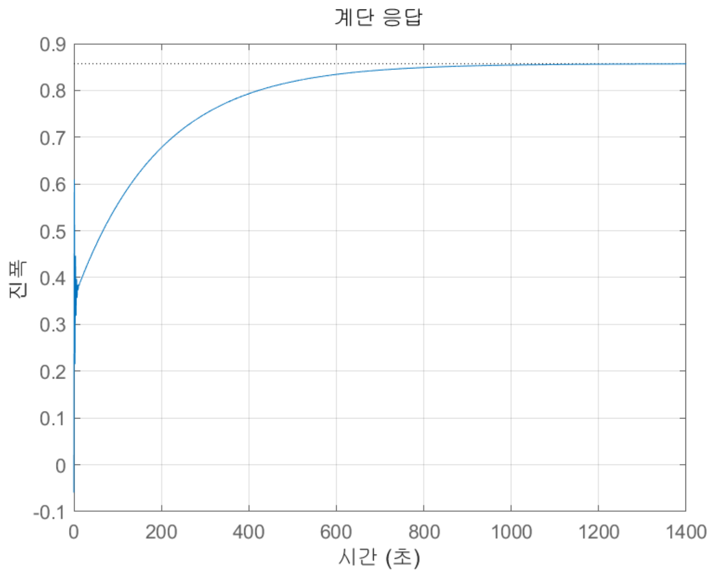
```
Dd_lag_tf =

  0.1098 z - 0.1095
  -----------------
     z - 0.9998

샘플 시간: 0.2 seconds
이산시간 전달 함수입니다.
모델 속성
Kc = 0.109813
z0 = 0.997510
zp = 0.999772
```

$$C_{lag}^{d}(z) = 0.109813\frac{z - 0.997510}{z - 0.999772}$$

1.b.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% margin(Gd_tf);

Gw_tf = d2c(Gd_tf,'tustin');
% figure();
% margin(Gw_tf); grid on;

om_w1 = 2.49;

k_w0 = 0.005;% 0.1 이하로 설정.

om_w0 = k_w0*om_w1;

a0 = 1.2; % 임의로 설정

k_wp = 0.3; % 1이하로 설정

om_wp = k_wp*om_w0/(a0*exp(20.1/20));

Dw_lag_tf = a0*tf([1/om_w0 1], [1/om_wp 1]);

open_loop = Gw_tf * Dw_lag_tf;

closed_loop = feedback(open_loop, 1);

step(closed_loop); grid on;
```

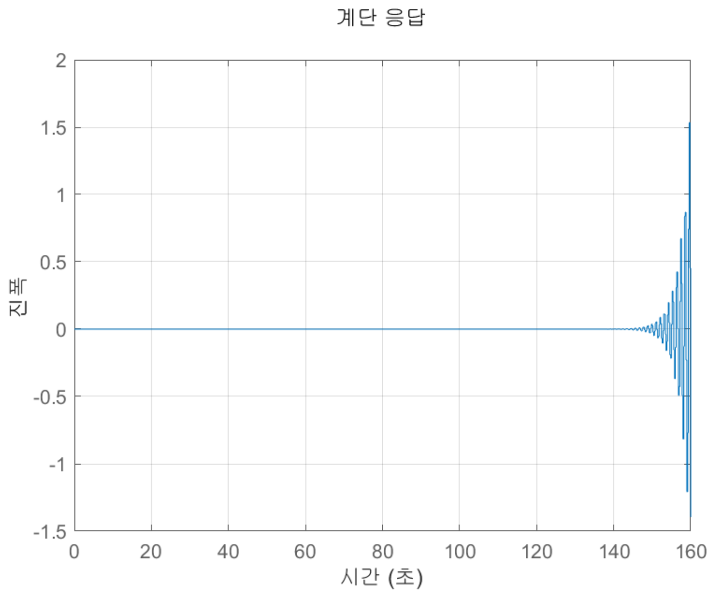
결과 :



1.c.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 불확실성 모델 추가
num = [-0.05 1];
den = [0.05 1];
Gp_uncertain_tf = tf(num,den);
Gp_tf = Gp_tf * Gp_uncertain_tf;

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% bilinear transform을 이용하여 연속시간 시스템으로 변환
Gw_tf = d2c(Gd_tf,'tustin');

% closed loop 생성
closed_loop = feedback(Gd_tf, 1);

% step rseponse 확인
step(closed_loop); grid on;
```

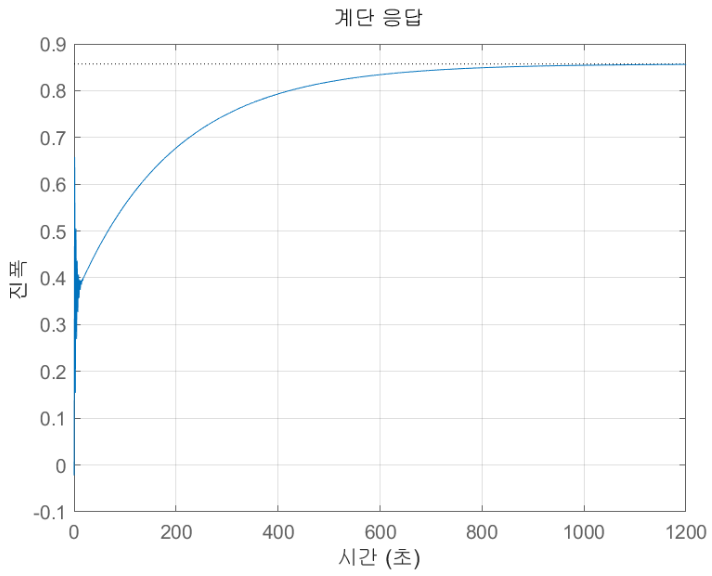
결과 :



1.d.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 불확실성 모델 추가
num = [-0.05 1];
den = [0.05 1];
Gp_uncertain_tf = tf(num,den);
Gp_tf = Gp_tf * Gp_uncertain_tf;

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% bilinear transform을 이용하여 연속시간 시스템으로 변환
Gw_tf = d2c(Gd_tf,'tustin');

% lag compensator 설계
om_w1 = 2.49;
k_w0 = 0.005;% 0.1 이하로 설정.
om_w0 = k_w0*om_w1;
a0 = 1.2; % 임의로 설정
k_wp = 0.3; % 1이하로 설정
om_wp = k_wp*om_w0/(a0*exp(20.1/20));
Dw_lag_tf = a0*tf([1/om_w0 1], [1/om_wp 1]);

% 이산시간으로 변환
Dd_lag_tf = c2d(Dw_lag_tf, T, 'zoh');

% lag compensator 적용
open_loop = Gd_tf * Dd_lag_tf;

% closed loop 생성
closed_loop = feedback(open_loop, 1);

% step rseponse 확인
step(closed_loop); grid on;
```

결과 :



1.e.

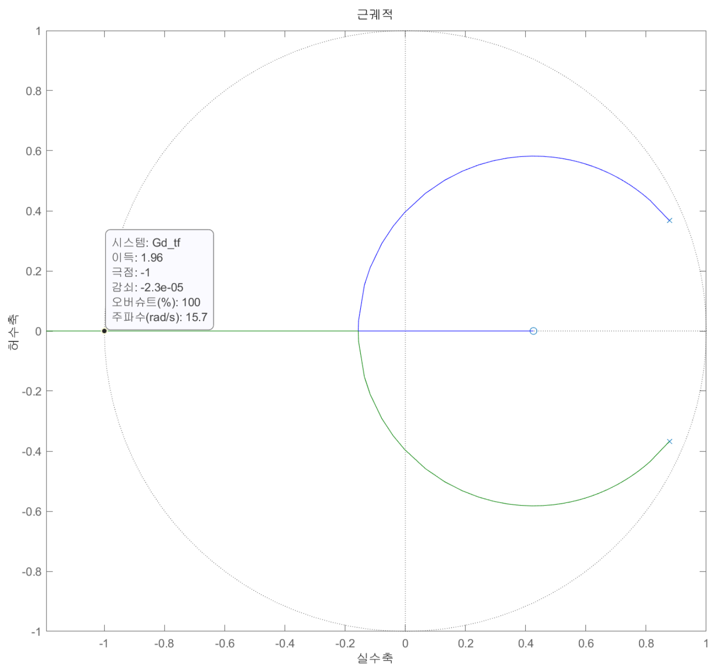
$C^d_{lag}(z)$는 Phase margin을 키워 시스템이 어느정도 불확실해도 안정성을 보장해주는 역할을 합니다.

2.a.

$C^d(z) = K_p$의 최댓값 $K^\star_p$를 찾기 위해 root locus를 그립니다.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

rlocus(Gd_tf);
```

결과 :



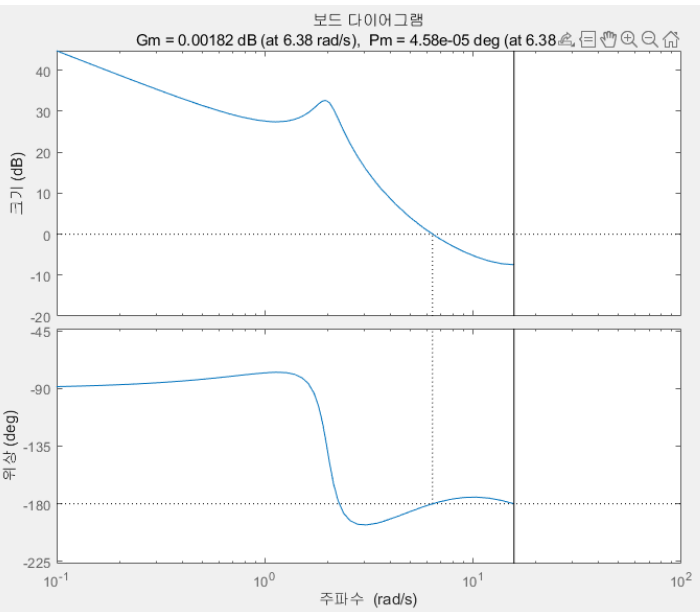
이에 따르면 $K^\star_p = 1.96$입니다.

2.b.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% 제어기 생성
Ksp = 1.96;
Cdp = 0.6*tf(Ksp);
Cdi = tf(T, [1 -1], T);

% Ksi를 찾는다.
Ksi = 0;
for digit = 0:1:5 % 소수점 5자리까지 찾는다.
    for g = 1:1:10
        k = Ksi + (g*(0.1^digit));
        Cd = Cdp + k*Cdi;
        [Gm,Pm,Wcg,Wcp] = margin(Gd_tf * Cd);
        if Pm < 0
            Ksi = Ksi + (g-1)*(0.1^digit);
            break;
        end
    end
end

format long
Ksi

Cd = Cdp + Ksi*Cdi;
margin(Gd_tf * Cd)
```

결과 :

```
Ksi =

   3.452230000000000
```

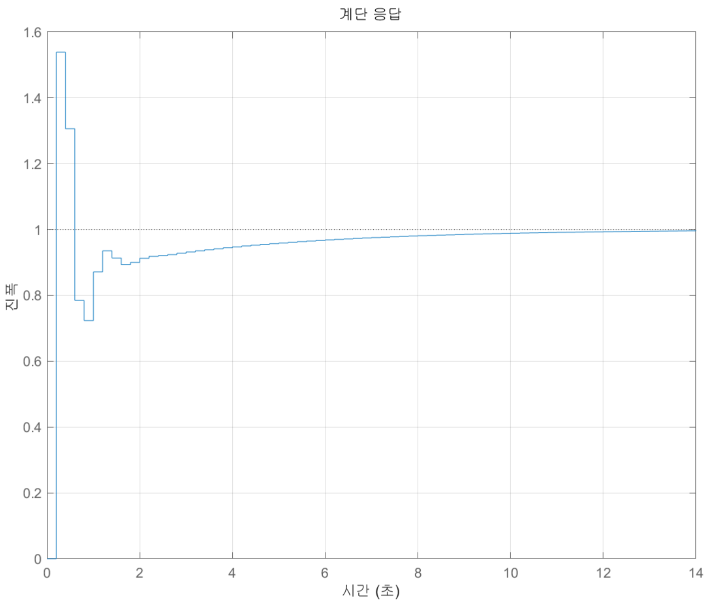


따라서 $K_i^\star = 3.45223$ 입니다.

2.c.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% 제어기 생성
Ksp = 1.96;
Ksi = 3.45223;
Cdp = 0.6*tf(Ksp);
Cdi = 0.1*Ksi*tf(T, [1 -1], T);
Cd = Cdp + Cdi;

step(feedback(Gd_tf * Cd, 1)); grid on;
```

결과 :



3.a.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% 상태 변수 모델로 변환
Gd_ss = ss(Gd_tf);

Gd_ss
```

결과 :

```
Gd_ss =

  A = 
             x1       x2
   x1     1.755  -0.9048
   x2         1        0

  B = 
       u1
   x1   1
   x2   0

  C = 
             x1       x2
   y1     1.308  -0.5571

  D = 
       u1
   y1   0

샘플 시간: 0.2 seconds
이산시간 상태공간 모델입니다.
```

3.b.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% 상태 변수 모델로 변환
Gd_ss = ss(Gd_tf);

% igenvalue를 p 벡터로 설정하는 벡터 K생성
p = [0.2 0.5];
K = place(Gd_ss.A, Gd_ss.B, p);

disp('K:');
disp(K);
```

결과 :

```
K:
   1.054596289762177  -0.804837418035960
```

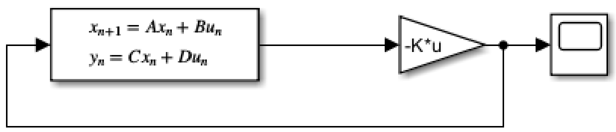
3.c.

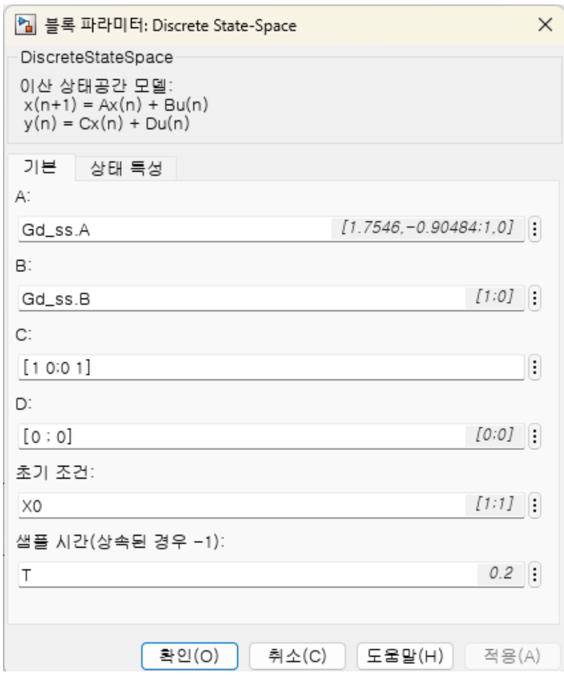
hw3_3_c.m :

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
```

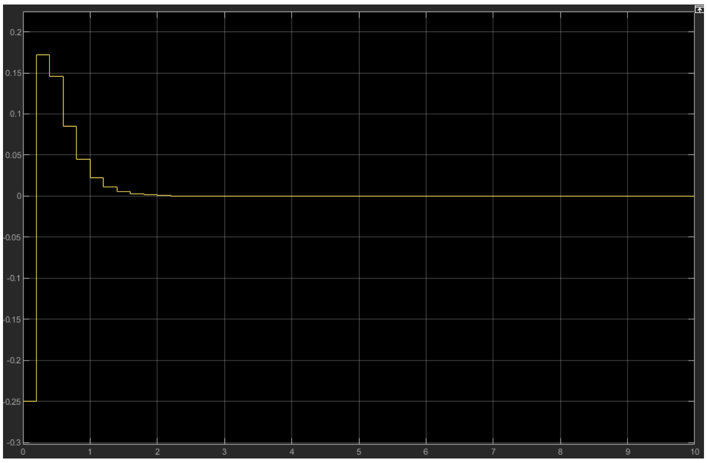
```
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% 상태 변수 모델로 변환
Gd_ss = ss(Gd_tf);

% igenvalue를 p 벡터로 설정하는 벡터 K생성
p = [0.2 0.5];
K = place(Gd_ss.A, Gd_ss.B, p);

% 초기 상태 벡터 X0 설정
X0 = [1; 1];

hw3_3_c_sim
```

hw3_3_c_sim.slx :





scope :



3.d.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% 상태 변수 모델로 변환
Gd_ss = ss(Gd_tf);

% igenvalue를 p 벡터로 설정하는 벡터 L생성
p = [0.2 0.5];
L = place(Gd_ss.A', Gd_ss.C', p)';

L
```

결과 :

```
L =

   1.147607170162977
   0.802082744596030
```

$$\mathbf{L}^d = \begin{bmatrix} 1.147607170162977 \\ 0.802082744596030 \end{bmatrix}$$

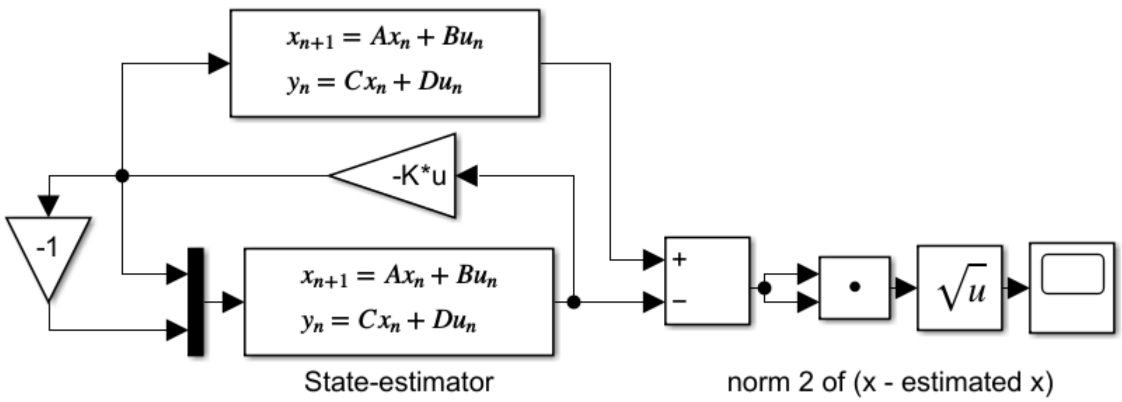
3.e.

hw_3_3_e.m :

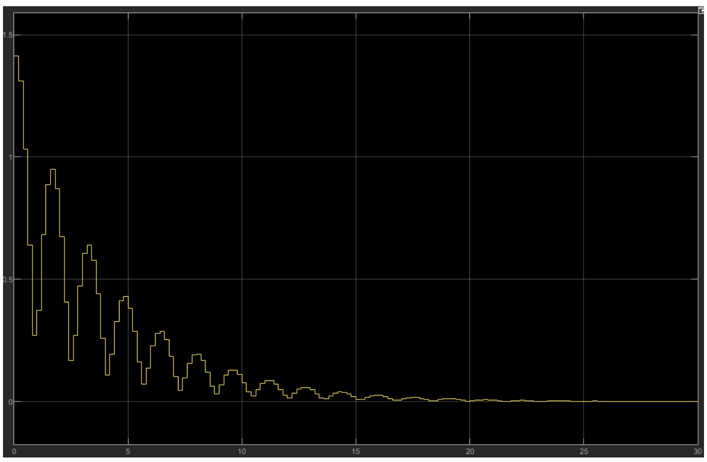
```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');
```

```
% 상태 변수 모델로 변환
Gd_ss = ss(Gd_tf);

% igenvalue를 p 벡터로 설정하는 벡터 K,L생성
p = [0.2 0.5];
K = place(Gd_ss.A, Gd_ss.B, p);
L = place(Gd_ss.A', Gd_ss.C', p)';

% 초기 상태 벡터 X0 설정
X0 = [1; 1];

A_est = Gd_ss.A - L*Gd_ss.C;
B_est = [Gd_ss.B L];

hw3_3_e_sim
```

hw3_3_e_sim.slx :



scope :



실제 상태 벡터를 추정 상태 벡터로 빼서 만든 벡터의 norm이 0으로 수렴하는 것을 scope를 통해 볼 수 있습니다.
따라서 state estimator이 잘 동작함을 알 수 있습니다.

3.f.

$$
\begin{aligned}
\hat{\mathbf{x}}^d(k+1) &= \mathbf{A}^d\hat{\mathbf{x}}^d(k) + \mathbf{B}^d u^d(k) + \mathbf{L}^d(y^d(k) - \mathbf{C}^d\hat{\mathbf{x}}^d(k)) && \because (4) \\
\hat{\mathbf{x}}^d(k+1) &= \mathbf{A}^d\hat{\mathbf{x}}^d(k) + \mathbf{B}^d(-\mathbf{K}^d\hat{\mathbf{x}}^d(k)) + \mathbf{L}^d(y^d(k) - \mathbf{C}^d\hat{\mathbf{x}}^d(k)) && \because (5) \\
&= (\mathbf{A^d} - \mathbf{B^d}\mathbf{K^d} - \mathbf{L^d}\mathbf{C^d})\hat{\mathbf{x}}^d(k) + \mathbf{L}^d y^d(k)
\end{aligned}
$$

$$
\begin{aligned}
z\hat{\mathbf{X}} &= (\mathbf{A^d} - \mathbf{B^d}\mathbf{K^d} - \mathbf{L^d}\mathbf{C^d})\hat{\mathbf{X}} + \mathbf{L}^d Y^d \\
-U^d &= \mathbf{K}^d\hat{\mathbf{X}}
\end{aligned}
$$

위 식에서 유도한 상태 변수 모델을 가지고 MATLAB을 통해 전달함수를 구합니다.

```
clc; clear; close all;
% 모델 생성
num = [5 20];
den = [1 0.5 4];
Gp_tf = tf(num,den);

% 이산시간 시스템으로 변환
T=0.2;
Gd_tf = c2d(Gp_tf,T,'zoh');

% 상태 변수 모델로 변환
Gd_ss = ss(Gd_tf);

% igenvalue를 p 벡터로 설정하는 벡터 L, K생성
p = [0.2 0.5];
K = place(Gd_ss.A, Gd_ss.B, p);
p = [-0.1 0.2211];
L = place((Gd_ss.A - Gd_ss.B*K)', Gd_ss.C', p)';


% 전체 전달함수 생성
An = Gd_ss.A - Gd_ss.B*K - L*Gd_ss.C;
Bn = L;
Cn = K;
tfn = tf(ss(An,Bn,Cn,0,T));
tfn
```

결과 :

```
tfn =


  4.895e-17 z + 3.551e-17
  ------------------------
  z^2 - 0.1211 z - 0.02211

샘플 시간: 0.2 seconds
이산시간 전달 함수입니다.
```