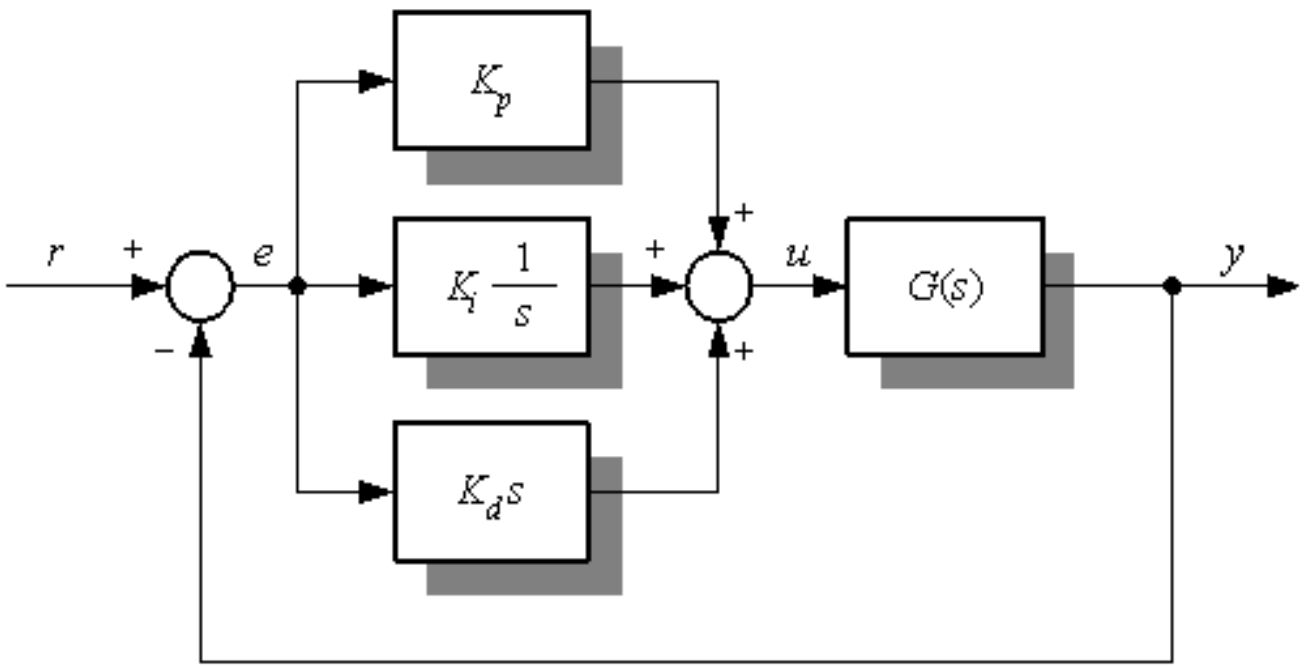
What is PI control

시간영역

P 제어기 : Kp*$e(t)$

I 제어기 : Ki*$\int_0^t e(\tau)d(\tau)$

D 제어기 : Kd* $\frac{de(t)}{dt}$



주파수 영역

P 제어기 : Kp

I 제어기 : $\frac{Ki}{s}$

D 제어기 : Kd*s

위의 블록선도에서 Kd = 0 이면 PI제어기가 된다.

따라서 PI 제어기의 전달함수는 C(s) = Kp + $\frac{Ki}{s}$ 가 되고 개로 전달함수는 L(s) = G(s)*C(s) 가 된다

우선 각 제어기의 특징을 알아보도록 하자

P 제어 : 기준 신호와 되먹임 신호 사이의 차인 오차 신호에 적당한 비례상수 이득을 곱해 서 제어 신호를 만들어내는 방법

- P gain이 높아지면 상승 시간이 줄어들고 오버 슈트가 증가한다. 무엇보다 정상상태 오차를 완전히 0으로 만들 수 없다. (0형 시스템)

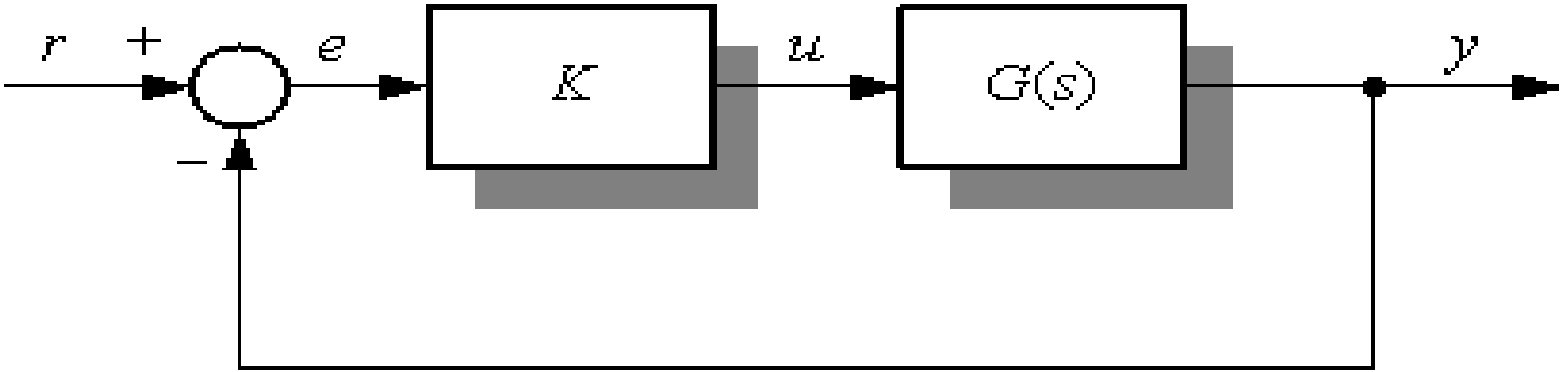
I 제어 : 비례 제어에서 발생하는 잔류 편차를 적분 동작으로 제거해 오차를 줄이기 위한 제어 방법

PI 제어 : P제어만 할 경우에 정상상태 오차를 완전히 없앨 수 없기 때문에 전달함수의 분모에 원점을 지나는 극점을 추가해 주어서 계단 입력에 대한 정상상태오차를 0으로 만들 수 있다.

이해를 위해 동영상을 보자

https://www.youtube.com/watch?v=fusr9eTceEo

이제 시뮬레이션으로 확인해보자

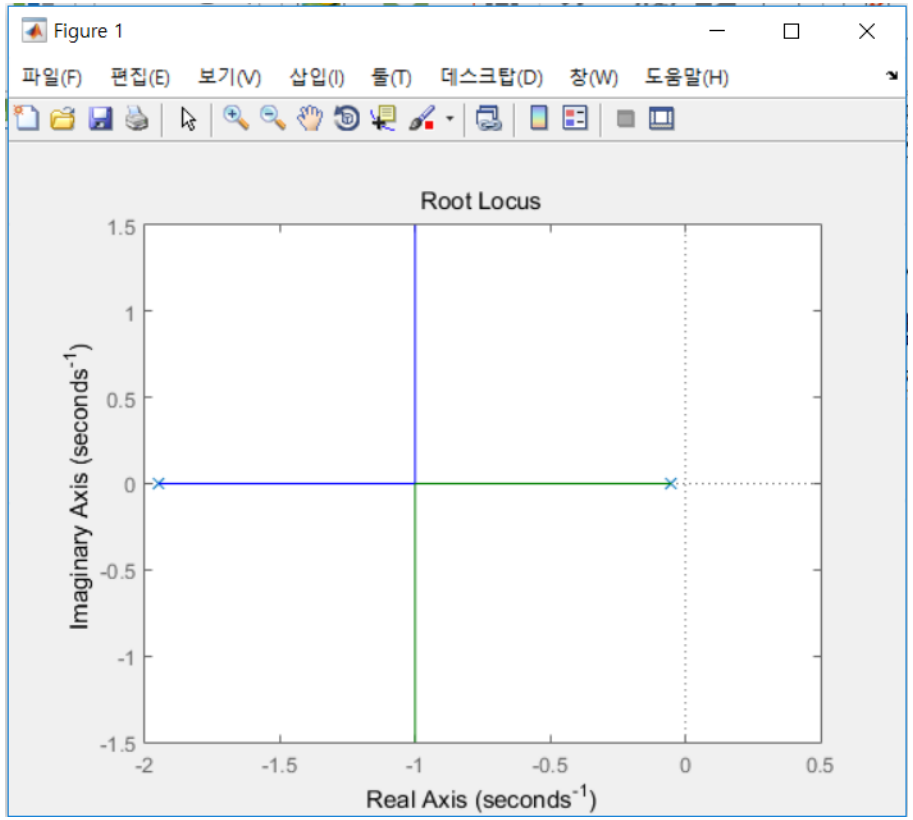


가장먼저 시스템이 안정성을 확인하자 폐로시스템에서 개로전달함수 G(s) = 0.1/s^2+2s+0.1 로 정하고 Kp = K 라고 치고 개로 전달함수 K*G(s)에 대한 근 궤적을 구해보면

PID_control_2.m | Untitled

```
clear
clc

num = [0.1];
den = [1 2 0.1];

G = tf(num,den)

Kp = 1;
Ki = 0
Kd = 0;

K = pid(Kp,Ki,Kd)

rlocus(K*G)
```

(Matlab code)



(근 궤적)

극점이 두 개다 좌 반면에 있고 K값을 아무리 변화시켜도 좌 반면에 위치하므로 안정한 시스템이라는 것을 확인할 수 있다

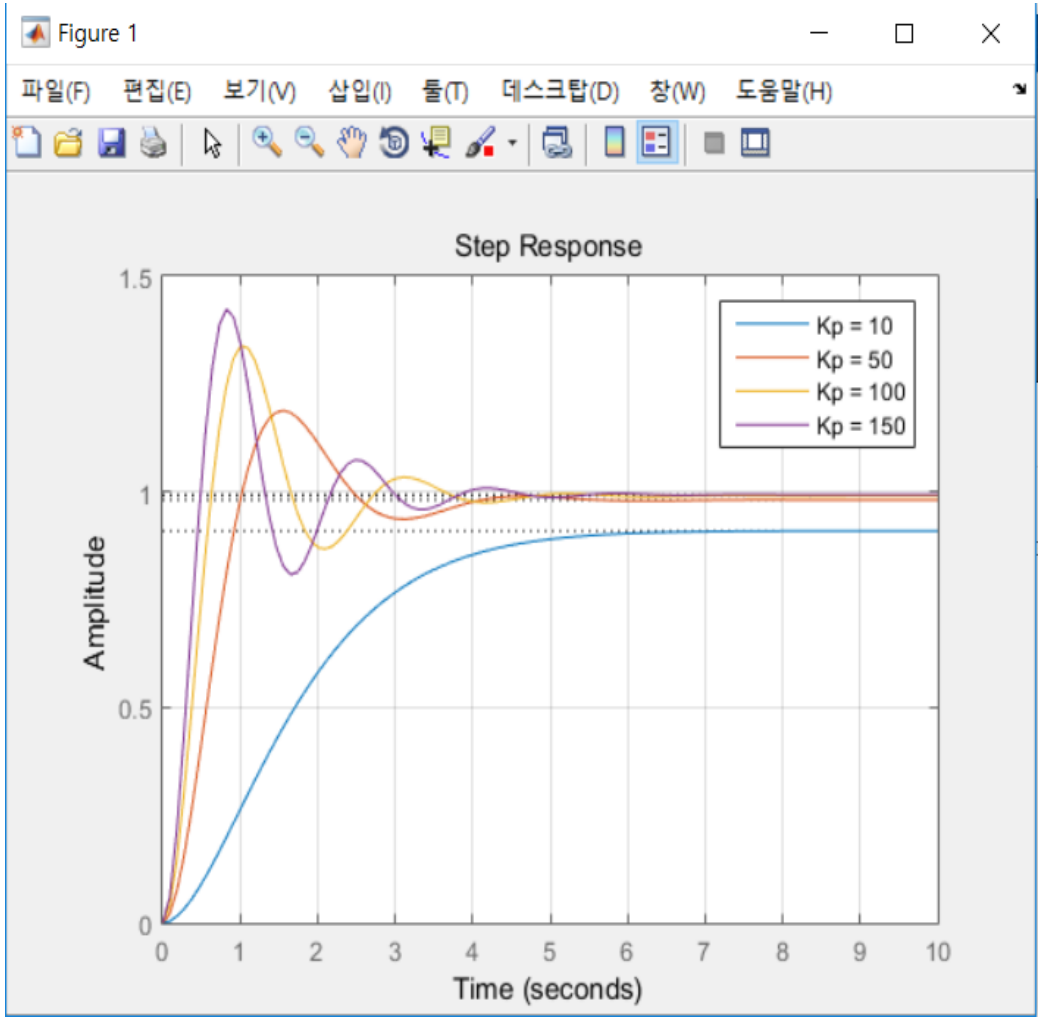
그 다음 matlab에서 Kp 값을 변경해보면서 계단 입력에 대한 시간 응답을 보자

편집기 - C:₩Users₩Hyun-Ho₩PID_control_2.m

PI_control.m | PID_control_2.m

```
clf
clear
clc

num = [0.1];
den = [1 2 0.1];
G = tf(num,den)
H = 1;

hold on
Kp = [10 50 100 150];
Ki = 0;
Kd = 0;

for i=1:4
K = pid(Kp(i),Ki,Kd)
T = feedback(K*G,H)
step(T,10)
end

legend('Kp = 10','Kp = 50','Kp = 100','Kp = 150');
grid;
hold off
```

(Matlab code)



(시간 응답)

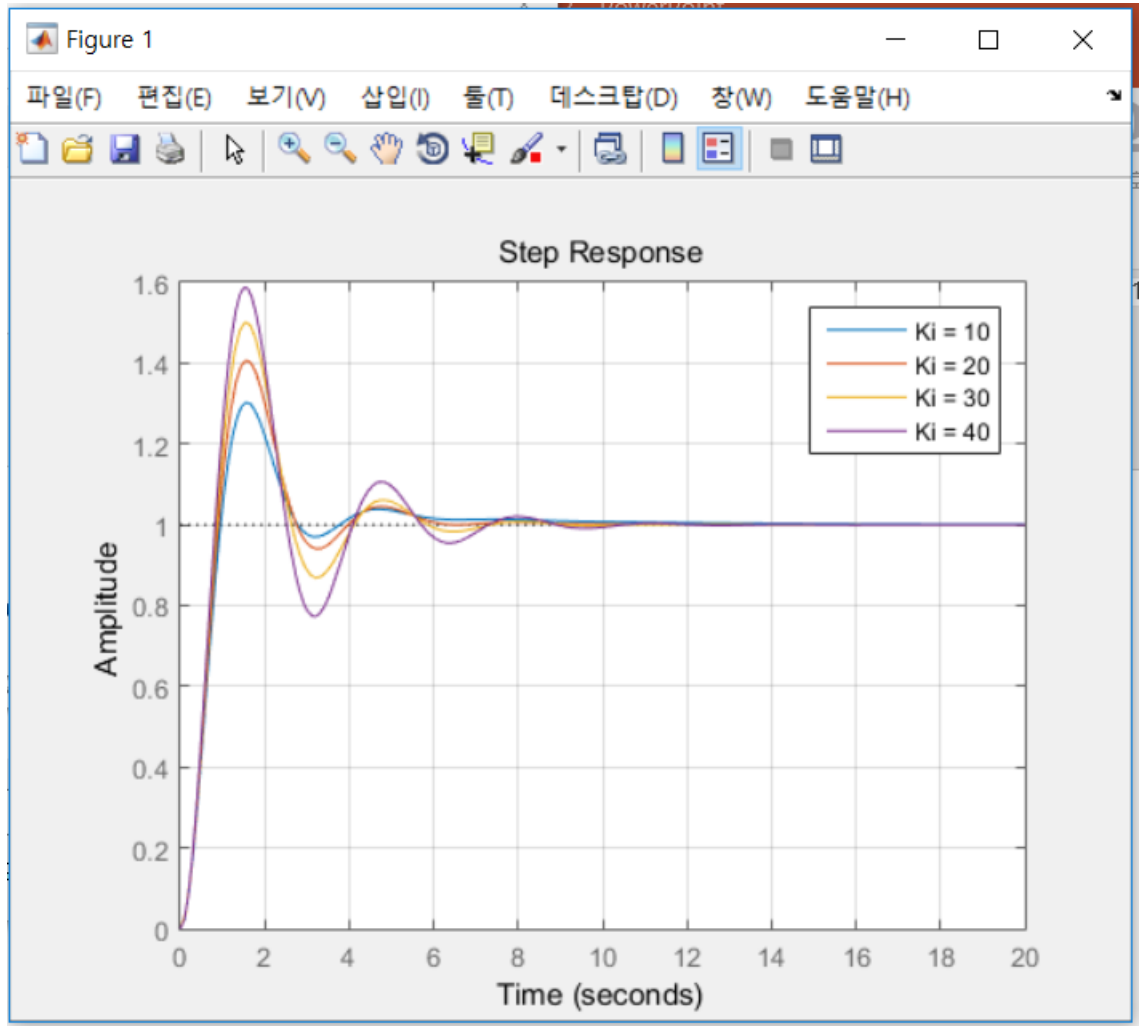
이번에는 matlab에서 Ki 값을 변경해보면서 계단입력에 대한 시간 응답을 보자

편집기 - C:₩Users₩Hyun-Ho₩PID_control_2.m

PI_control.m | PID_control_2.m

```
1 –    clf
2 –    clear
3 –    clc
4
5 –    num = [0.1];
6 –    den = [1 2 0.1];
7 –    G = tf(num,den)
8 –    H = 1;
9
10 –   hold on
11 –   Kp = 50;
12 –   Ki = [10 20 30 40];
13 –   Kd = 0;
14
15 –   for i=1:4
16 –   K = pid(Kp,Ki(i),Kd)
17 –   T = feedback(K*G,H)
18 –   step(T,20)
19 –   end
20
21 –   legend('Ki = 10','Ki = 20','Ki = 30','Ki = 40');
22 –   grid;
23 –   hold off
24
```

(Matlab code)



(시간 응답)

Applied to motors

180RPM
입력

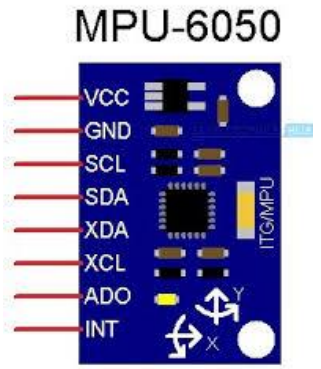
엔코더에서 RPM 측정
오차 발생

PI controller 에서 나온 제어량을
전압으로 전환

PI controller

Applied to motors

각도
입력

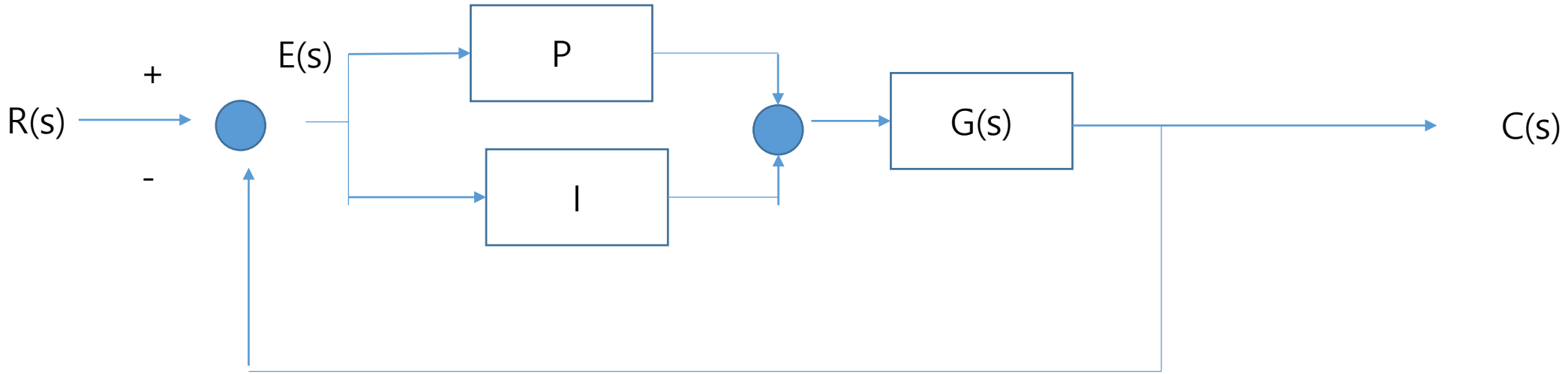


MPU에서 각도 측정 후
오차 발생

PI controller

PI_control code with C

```c
/******P,I,D Gain Set******/
double Kp = 1.0;
double Ki = 0.0;
double Kd = 0.0;
double error;

double error_previous; // Previous error

double desired_rpm = 180; // BLDC RPM

double current_rpm;

double P_control, I_control, D_control;

double Time = 0.004; // Loop Time

double PID_control;

void main()
{

}

void pid_control()
{
    current_rpm = 엔코더 측정값;

    error = desired_rpm - current_rpm;

    P_control = Kp * error;
    I_control += Ki * error * Time;
    D_control = Kd * (error - error_previous) / Time;


    PID_control = P_control + I_control + D_control;
    error_previous = error;
}
```



다음과 같은 개로전달함수를 가지는 시스템에서 아래 Spec을 만족시키는 PI 제어기를 설계해보자.

$$G(s) = \frac{1}{(s+1)(0.5s+1)}$$

1) 계단입력시 정상상태 오차 = 0
2) 최대 오버 슈트 < 5%
3) 정착시간(5%) < 6초
4) 램프입력 시 오차 상수 Kv >= 9

K = Kp + $\frac{Ki}{s}$ = $\frac{Kp(s+\frac{Ki}{Kp})}{s}$ 먼저 PI 제어기를 개로 전달함수를 구하면

K*G(s) = $\frac{Kp(s+\frac{Ki}{Kp})}{s}$*$\frac{1}{(s+1)(0.5s+1)}$ 램프입력시 정상상태 오차는 $\lim_{s \to 0} \frac{Kp(s+\frac{Ki}{Kp})}{(s+1)(0.5s+1)}$ = Kp 이며 4번 조건을 만족하기 위해 Kp >= 9 여야 한다.

이제 오버 슈트를 줄여보자

개로 전달함수 L(s) = K*G(s) = $\frac{Kp(s+\frac{Ki}{Kp})}{s}$*$\frac{1}{(s+1)(0.5s+1)}$ 에서 영점은 $-\frac{Ki}{Kp}$ 이다 영점을 극점보다 0 에 가깝게 놓기위해 $\frac{Ki}{Kp}$ = 0.8 로 설정한다.

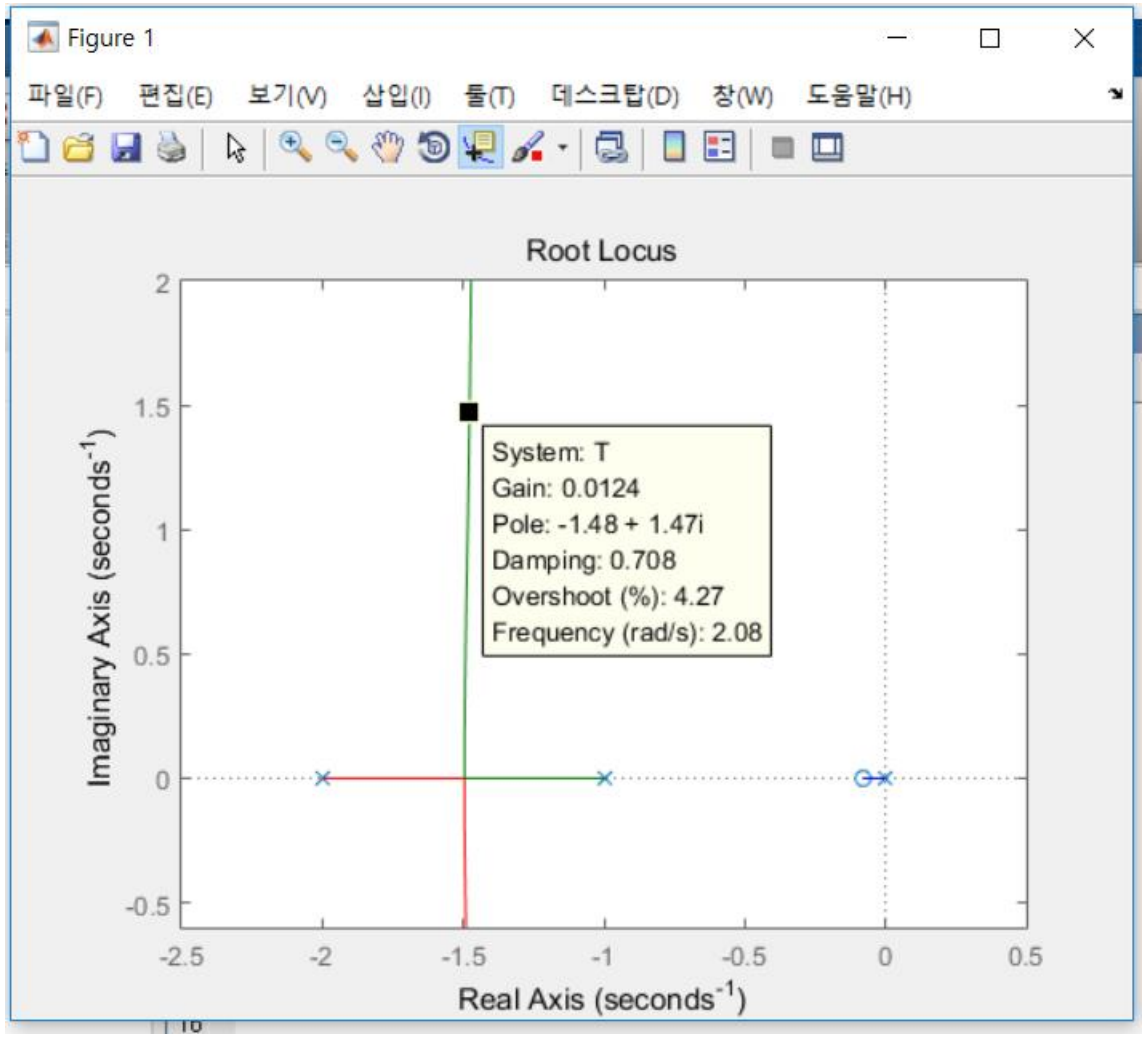
개로 전달함수 L(s) = K*G(s) = $\frac{Kp(s+\frac{Ki}{Kp})}{s} * \frac{1}{(s+1)(0.5s+1)}$ 에서 영점은 $-\frac{Ki}{Kp}$ 이다 영점을 극점보다 0 에 가깝게 놓기 위해 $\frac{Ki}{Kp}$ = 0.08 로 설정한다.

근궤적을 그려 보자

편집기 - C:₩Users₩Hyun-Ho₩PID_control_3.

PI_control.m | PID_control_2.m

```
1 -     clf
2 -     clc
3
4 -     syms s
5
6 -     Kp = 100;
7 -     Ki = 8;
8
9
10 -    num3 = [Kp Ki];
11 -    den3 = [0.5 1.5 1 0]
12
13 -    T = tf(num3,den3);
14
15 -    rlocus(T);
16
```



여기서 s = -1.48+1.47i 일 때 감쇠비가 0.708이 되는 Kp를 선택하였으며
Kp를 구하려면1+Kp*L(s) = 0 방정식을 풀면 Kp를 구할 수 있다. 이식에서

Kp = 1.1446,
Ki = 0.8*Kp = 0.0916 을 구할 수 있다.

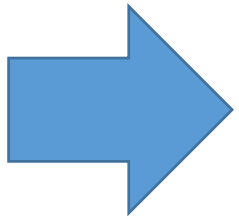
하지만 우리는 위에서 4번 조건을 만족 시키기 위하여 Kp >= 9 조건을 맞춰야 한다.

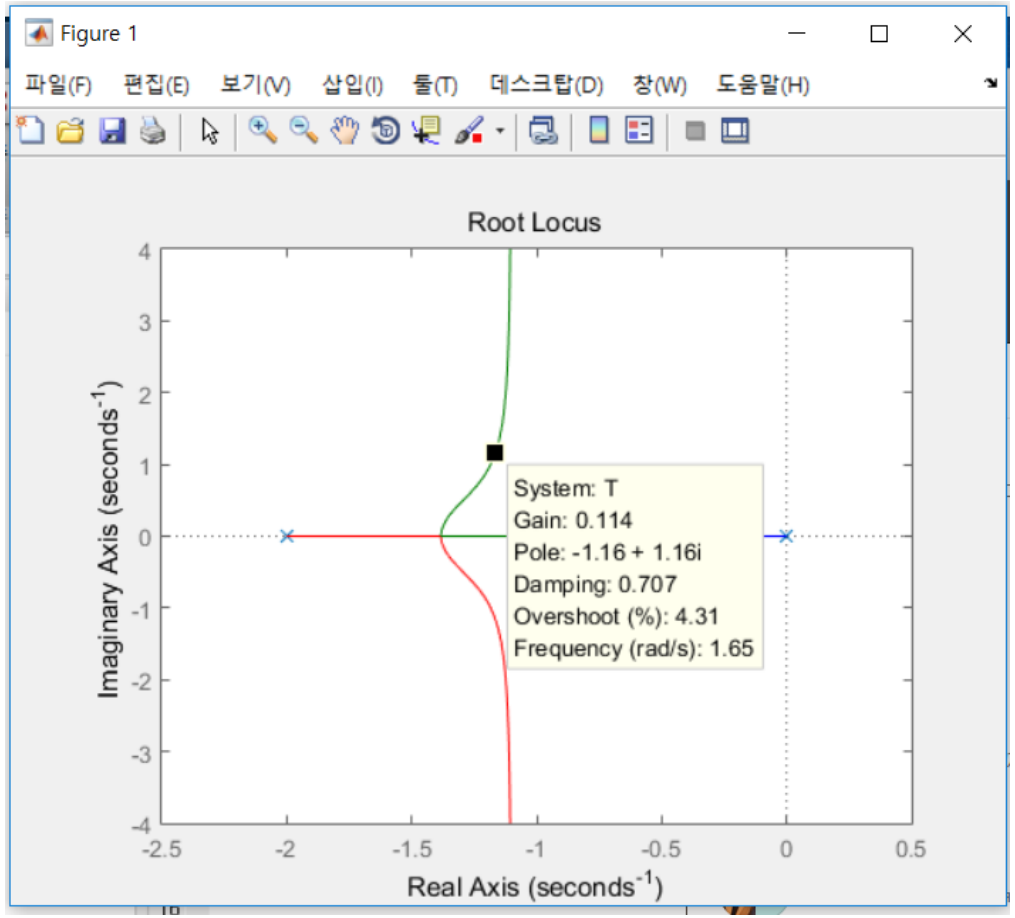
따라서 $\frac{Ki}{Kp}$ 값을 0.8 로 수정한 후 근 궤적을 다시 그리면

편집기 - C:₩Users₩Hyun-Ho₩PID_control_3.m

PI_control.m | PID_control_2.m | PID

```
1 –  clf
2 –  clc
3
4 –  syms s
5
6 –  Kp = 10;
7 –  Ki = 8;
8
9
10 –  num3 = [Kp Ki];
11 –  den3 = [0.5 1.5 1 0]
12
13 –  T = tf(num3,den3);
14
15 –  rlocus(T);
16
17
18
19
```

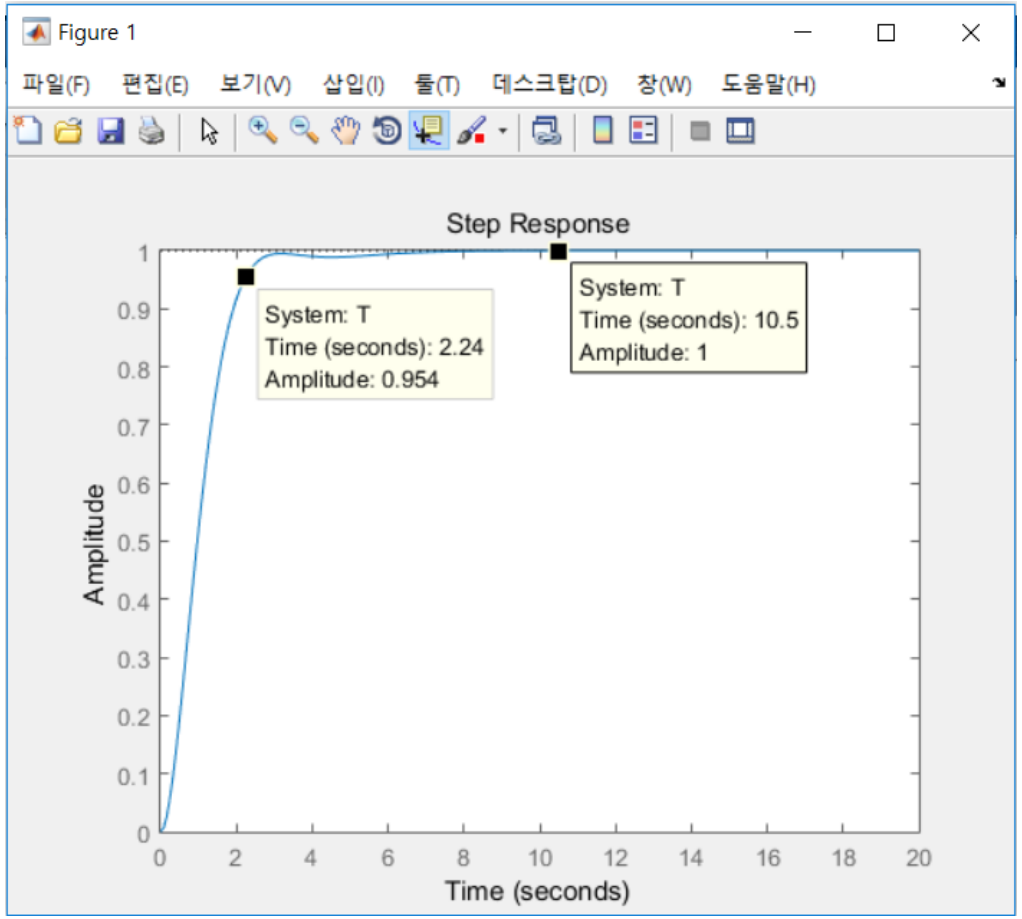




여기서 다시 감쇠비가 0.707인 부분을 찾으면 s = -1.16 + 1.16i 임을 알 수 있다.
또다시 Kp를 구하기 위하여 1+Kp*L(s) = 0 방정식을 풀면 우리가 원하던 Kp,Ki값을 구할 수 있다.

Kp = 1.1435 , Ki = 0.9148

구한 Kp,Ki 값이 맞는지 확인해보자



오버 슈트는 0% 이며
Settling Time = 2.24 sec
정상상태 오차 = 0

처음에 설계한 spec에 모두 맞추었다.

Reference

http://cemtool.co.kr/products/control/Chap8/Sec8.5/Sec8.5.htm

http://www.academia.edu/13298003/PID_%EA%B8%B0%EC%B4%88_%EC%A0%95%EB%A6%AC

http://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwj6jJ2suYLaAhWDQpQKHZpyCEkQFggmMAA&url=http%3A%2F%2Fprof.gwangju.ac.kr%2Fbbs%2Fdownload1.php%3Fgid%3Dyongmin%26no%3D781%26type%3Dlecture_note&usg=AOvVaw2liM6_N0MxTUhEt2mtZKg4

http://hyongdoc.tistory.com/49