rldesign.PI Proportional–integral (PI) controller design

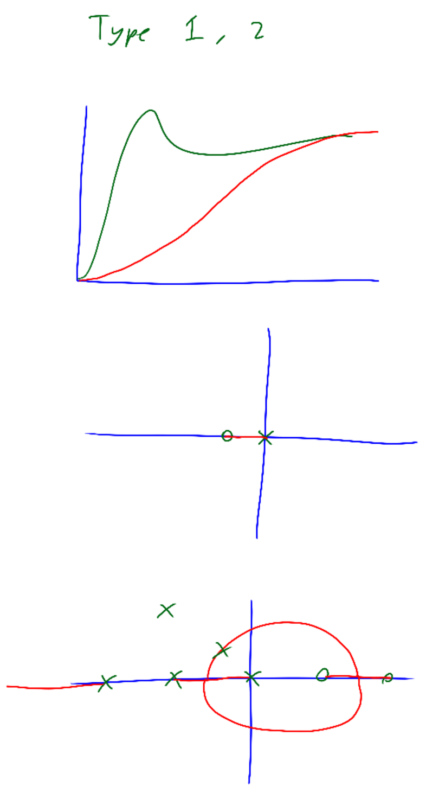
When studying steady-state error, we discovered that the more integrators (s^{-1}) in the open-loop transfer function, the better the steady-state error. PI control includes integrator compensation to a proportional controller without significantly affecting the transient response. Later, we will deal with how to design for transient response.

Here's the plan: include an integrator (i.e. pole at the origin) in the controller and a nearby zero to counter the pole's (slowing) effects on the transient response.

Why does the integrator affect the transient response? Adding a pole at the origin completely changes the root locus, and therefore the location of the closed-loop poles, and therefore the transient response.

In order to mitigate this, we place a zero near the origin, which nearly cancels the integrator's effect on the root locus. To see this, recall that the root locus must meet the phase criterion (Lec. rlocus.def). Let us meditate on Fig. PI.1, in which a system is presented that initially contained the three left half-plane poles and no zeros. Including the integral pole at the origin (integrator) and zero nearby, we obtain the root locus shown. From the phase criterion,

$$-\theta_1 + \theta_2 - \theta_3 - \theta_4 - \theta_5 = \pi \pm 2\pi m \quad (m \in \mathbb{Z}). \tag{1}$$



If the compensator zero is placed close to the pole, then $\theta_2 - \theta_1 \approx 0$ and the root locus is mostly unchanged from its pre-compensation state.

Design procedure

The following design procedure can guide us through this typically straightforward controller design.

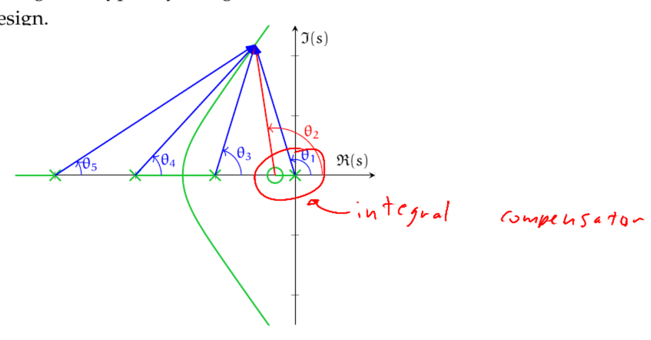


Figure PI.1: the effect of the integral compensator on the root locus's angle condition.

1. Design a proportional controller to meet transient response requirements by choosing the gain K for the dominant closed-loop poles to be $p_{1,2}$.
2. Include cascade integral compensation and a real zero near $\mathrm{Re}(p_{1,2})/10$.
3. Tune the gain K such that the close-loop poles are as desirable as possible.
4. Simulate the time response to see if it meets specs. Tune. If the steady-state compensation is too slow, try moving the zero leftward.

Example rldesign.PI–1 re: PI control for percent overshoot

For a plant with transfer function

$$\frac{10}{(s+2)(s+5)}$$

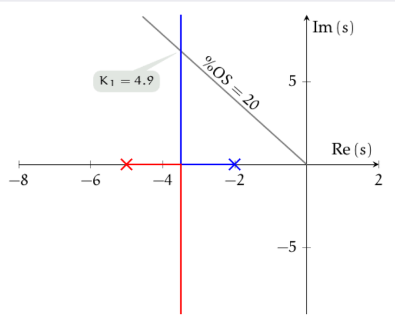
design a unity feedback PI controller such that the system has $\%OS = 20$ and zero steady-state error for step inputs.

We will use MATLAB. First, let us observe that with no integrators in the plant (Type 0 system), the system will have a finite steady-state error to step inputs. Therefore, we require integral compensation. Let's define the transfer function.

```
sys1 = zpk([],[-2,-5],10);
```

The desired closed-loop pole location is along the ray corresponding to 20 percent overshoot. Since this is available with the data cursor in the `rlocus` plot, there is no need to compute the damping ratio or the angle of the ray. Let us consider the root locus.

```
figure;
rlocus(sys1,sort([0,.225,4:.1:10,Inf])
    );
ylim([-10,10])
grid on
```



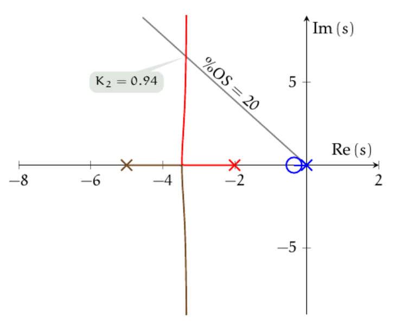
From the figure, we can see that when the gain is $K_1 = 4.9$, according to the second-order approximation, the $\%OS$ is 20. This occurs at the test point $s_{tp} = -3.5 + j6.84$. If we were designing simply a P controller, we would now simulate the closed-loop response with this gain. Before we simulate, let's apply integral compensation. We put a pole at the origin as the integrator and compensate with a nearby zero. We start with that zero at $\mathrm{Re}(s_{tp})/10 = -0.35$. Our compensator has the transfer function

$$\frac{s+0.35}{s}$$

and can be applied to the open-loop transfer function as follows.

```
sReal = -3.5;
zeroc = sReal/10;
comp = zpk([zeroc],[0],1); %
    compensator
sys2 = K1*comp*sys1; % controlled open
    -loop tf
```

Now a new root locus analysis is required in order to determine the new gain required to get back near the test point. This is shown below.



We see that the cascade gain required to return to the overshoot ray is $K_2 = 0.94$.

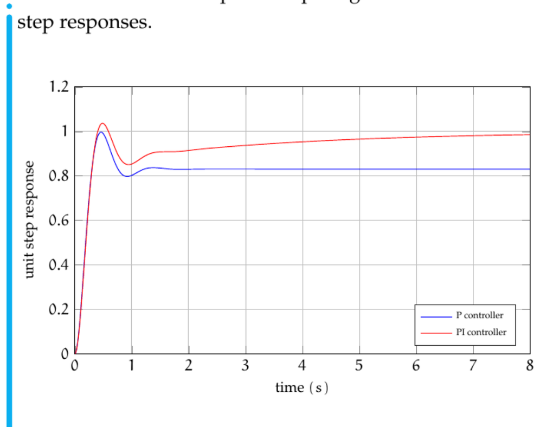
Now we must find the closed-loop transfer functions for each controller design, which can be found as follows.

```
sys1cl = feedback(K1*sys1,1);
sys2cl = feedback(K2*sys2,1);
```

Now we are ready to simulate the closed-loop step response to evaluate the actual step response for each controller design.

```
tvec = 0:.01:8;
y1 = step(sys1cl,tvec);
y2 = step(sys2cl,tvec);
stepinfo(y1,tvec)
```

The command `stepinfo` computes the simulated transient response characteristics. The result is $\%OS = 20.0$, good! Note that we used the P controller for this evaluation. The strict definition of this gives a skewed value due to the steady-state error compensation. Let's take a look at a plot comparing the two step responses.



Note that the PI-controlled system has steady-state error approaching zero and that the two systems have similar transient response characteristics, per our expectation. The steady-state error does respond relatively "slowly" because the third closed-loop pole introduced by the integral compensator is relatively close to the imaginary axis. Moving the compensator zero leftward can speed this response, but transient responses will be increasingly effected. In this case, the settling time determined by the complex closed-loop poles (we could call this the "transient settling time") will increase as we move the zero leftward. However, the settling time determined by the integrator (we could call this the "steady-state settling time") will simultaneously decrease. Specific system requirements would determine how we balance these considerations.

Our final controller design has transfer function

$$K_1 K_2 \cdot \frac{s+0.35}{s} = 4.61 \cdot \frac{s+0.35}{s}.$$