

06.6 rldesign.PD Proportional-derivative (PD) controller design

Thus far, our designs have been restricted to closed-loop pole locations on the original root locus. We could add integral or lag compensation for steady-state error performance and vary the gain for transient response performance. But what if we desire closed-loop poles $p_{1,2}$ to be in a location that the root locus does not intersect?

Among many possible methods to address this, we pursue the following: a **derivative compensator** with zero location z_c chosen such that the root locus intersects $p_{1,2}$, with form

$$K(s - z_c), \quad (1)$$

where $K \in \mathbb{R}$ is a gain. This compensator is called “derivative” because its primary effect on the overall controller’s operation on the error e is a new factor of s , yielding a scaling of the term $sE(s) = \dot{e}(t)$.

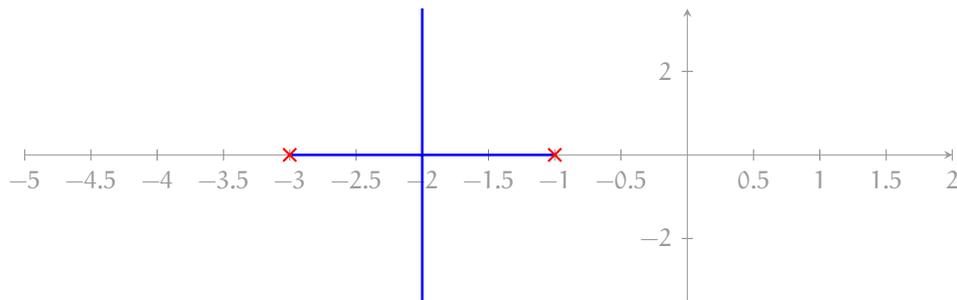


Figure PD.1: root locus for a simple plant with two poles.

The effect of this zero is to pull the locus toward it. Consider the simple plant of Fig. PD.1. Suppose we would like to speed up the closed-loop response, but cannot because, no matter how much gain we use, the settling time is fixed by the vertical asymptotes. If we use a compensator zero at z_c , we can pull the locus leftward, as shown in Fig. PD.2. Varying z_c from $-\infty$

to 0, we see that any location left of -2 can be intersected. In fact, if we consider both positive and negative gains for this example, we can place a desired closed-loop pole at any location in the complex plane!

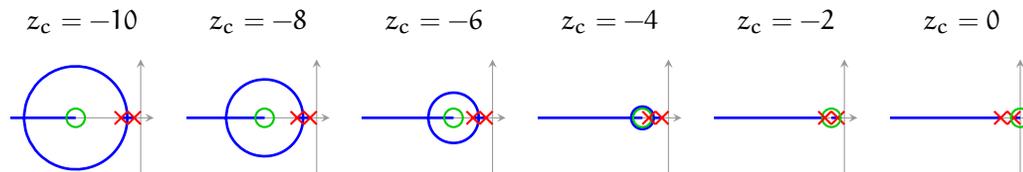


Figure PD.2: root locus (blue) for plant with poles (red) compensated with a zero (green) at z_c . Note that varying z_c yields root loci that can intersect any point in the complex plane if negative gains are considered. An animation corresponding to this figure can be found at https://youtu.be/VZbT_2bT2xU.

A way to approach designing a controller for a plant G with a derivative compensator C is to consider the compensator zero's effect on the phase criterion, which must always be satisfied at points on the root locus:

$$\angle(G(s)C(s)) = \pi. \tag{2}$$

In order for a desired point $s = \psi$ to be on the root locus, then,³



Let this angle $\angle(\psi - z_c)$, called the **compensator angle**, be given the symbol

$$\theta_c \equiv \angle(\psi - z_c). \tag{3}$$

Then

$$z_c = \text{Re}(\psi) - \text{Im}(\psi) / \tan \theta_c \quad (\theta_c \in [-\pi, \pi]), \tag{4}$$

³The 2π modulo in these expressions is suppressed for clarity.

where we have limited the application of this result to $\theta_c \in [-\pi, \pi]$ because a single zero can contribute angles in this interval only.^{4,5} This result is to be used in the design procedure that follows. It can be understood geometrically as the position of z_c such that the angle of the vector with tail at z_c and head at ψ is θ_c .

Design procedure

The following procedure provides a starting-point for proportional-derivative controller design. Let's assume the transient response specification is such that we desire a closed-loop pole to be located at $s = \psi$.

1. Design a proportional controller to meet transient response requirements by choosing the gain K_1 for the dominant closed-loop poles to be as close as possible to ψ .
2. Include a cascade derivative compensator of the form

$$K_2(s - z_c), \quad (5)$$

where, initially, $K_2 = 1$ and z_c is a real zero that satisfies Eq. 4. For convenience, we repeat the two key formulas:

$$\begin{aligned} \theta_c &= \pi - \angle G(\psi) \quad \text{and} \\ z_c &= \text{Re}(\psi) - \text{Im}(\psi) / \tan \theta_c \quad (\theta_c \in [-\pi, \pi]). \end{aligned}$$

3. Use a new root locus to tune the gain K_2 such that a closed-loop pole is at ψ .
4. Construct the closed-loop transfer function with the controller

$$K_1 K_2 (s - z_c). \quad (6)$$

5. Simulate the time response to see if it meets specifications. Tune.

⁴See [Lec. 06.10 rldesign.mutld](#) for how to handle required angle compensations beyond $\pm\pi$.

⁵Note that $\theta_c \in [-\pi, 0)$ is possible only when $\text{Im} \psi < 0$ and $\theta_c \in (0, \pi]$ is possible only when $\text{Im} \psi > 0$.

A design example

Let a system have plant transfer function

$$\frac{1}{(s+2)(s+6)(s+11)}. \quad (7)$$

Design a PD controller such that the closed-loop settling time is about 0.8 seconds and the overshoot is about 15%.

Determining ψ

We use Matlab for the design.⁶ First, we must determine what the specified transient response criteria imply for the locations of our closed-loop poles. Let one of these desired pole locations be called ψ . The transient response performance criteria are as follows.

```
Ts = .8; % sec ... spec settling time
OS = 15; % percent ... spec overshoot
```

The second-order approximation from [Chapter 03 trans](#) tells us that the settling time specification implies a specific $\text{Re}(\psi)$ and the overshoot a specific angle $\angle\psi$. The real part is found from the expressions

$$T_s = \frac{4}{\zeta\omega_n} \quad \text{and} \quad \text{Re}(\psi) = -\zeta\omega_n \Rightarrow \quad (8)$$

$$\text{Re}(\psi) = -\frac{4}{T_s}. \quad (9)$$

The angle is found via the equations

$$\zeta = \frac{-\ln(\%OS/100)}{\sqrt{\pi^2 + \ln^2(\%OS/100)}}, \quad (10)$$

$$\tan(\angle\psi) = \frac{\sqrt{1-\zeta^2}}{\zeta}, \quad \text{and} \quad \tan(\angle\psi) = -\text{Im}(\psi)/\text{Re}(\psi). \quad (11)$$

A remarkably simple expression results:

⁶See ricopic.one/control/source/pd_controller_design_example.m for the source.

$$\text{Im}(\psi) = -\text{Re}(\psi) \frac{\sqrt{1-\zeta^2}}{\zeta} \quad (12a)$$

$$\text{Im}(\psi) = -\text{Re}(\psi) \frac{\pi}{\ln(100/\%OS)}. \quad (12b)$$

So, in the final analysis, the desired pole location ψ (assuming the second-order approximation is valid) is given by the expression

$$\psi = -\frac{4}{T_s} \left(1 - j \frac{\pi}{\ln(100/\%OS)} \right). \quad (13)$$

This formula holds beyond the scope of this problem. We define it as an anonymous function.

```
psi_fun = @(Ts,pOS) -4/Ts*(1-1j*pi/log(100/pOS));
psi = psi_fun(Ts,OS);
disp(sprintf('psi = %0.3g + j %0.3g',real(psi),imag(psi)))
```

```
| psi = -5 + j 8.28
```

P control

We design a proportional controller that gets us as close as possible to ψ . The root locus is shown in [Figure PD.3](#).

```
G = zpk([], [-2, -6, -11], 1);
figure
rlocus(G)
```

Although we cannot get close to ψ on the root locus, we can at least meet our $\%OS$ specification by choosing a gain of about

$$K_1 = 240. \quad (14)$$

Let's construct the compensator and corresponding closed-loop transfer function G_P for gain control.

```
K1 = 240;
G_P = feedback(K1*G, 1);
```

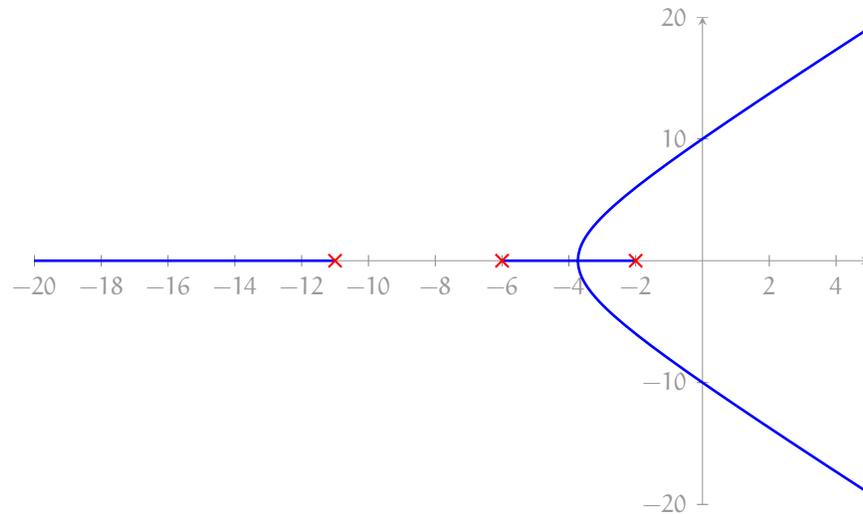


Figure PD.3: root locus without compensation.

Derivative compensation

Now, we use cascade derivative compensation with compensator

$$K_2(s - z_c). \quad (15)$$

For now, we set $K_2 = 1$. From Equation 4, we compute the compensator zero

$$z_c = \text{Re}(\psi) - |\text{Im}(\psi)| / \tan \theta_c \quad \text{and} \quad \theta_c = \pi - \angle G(\psi).$$

```
theta_c = pi - angle(evalfr(G,psi));
z_c = real(psi) - abs(imag(psi))/tan(theta_c);
disp(sprintf('theta_c = %0.3g deg',rad2deg(theta_c)))
disp(sprintf('z_c = %0.3g',z_c))
```

```
theta_c = 67.1 deg
z_c = -8.5
```

Let's construct the compensator sans tuned gain K_2 and tune it up using another root locus.

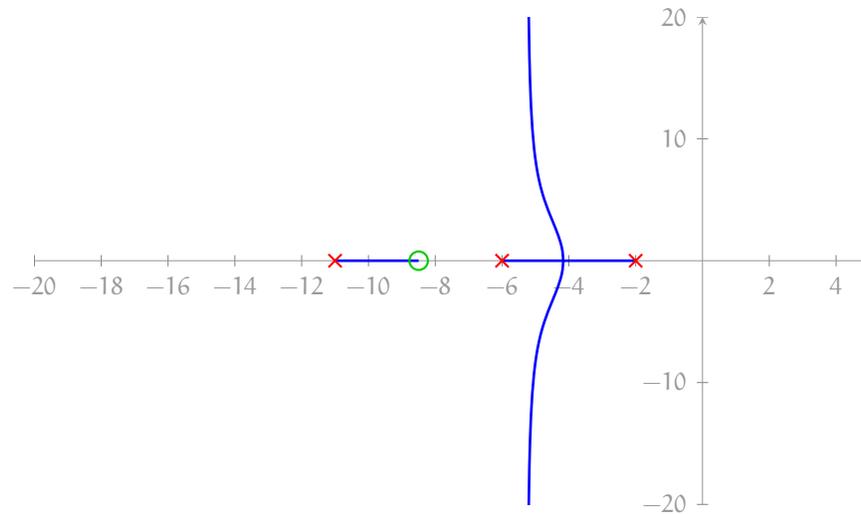


Figure PD.4: root locus with compensation.

```
C_sans = zpk(z_c, [], 1);
figure
rlocus(K1*C_sans*G)
```

The resulting root locus of [Figure PD.4](#) intersects ψ ! (I mean, we knew it would, but we had our doubts.) The corresponding gain is, from [Equation 2](#) (or we could use the data cursor),

$$K_2 = \frac{1}{|(\psi - z_c)G(\psi)|}. \quad (16)$$

Let's compute it, the controller C_{PD} , and the closed-loop transfer function G_{PD} .

```
K2 = 1/abs(evalfr(K1*C_sans*G,psi));
C = K1*K2*C_sans;
G_PD = feedback(C*G,1);
```

Simulate

Our placement of the ψ depended on the second-order approximation's accuracy, which in this case is questionable, due to the proximity of a third closed-loop pole. In any case, we simulate the step response to test the efficacy of the PD controller design and to compare it with the P controller.

```
t_a = linspace(0,2.5,200); % s ... sim time
y_P = step(G_P,t_a); % P controlled step response
y_PD = step(G_PD,t_a); % PD controlled step response
```

```
figure
plot(t_a,y_P);
hold on;
plot(t_a,y_PD);
xlabel('time (s)');
ylabel('step response');
grid on
legend('P control','PD control','location','southeast');
```

The responses, shown in [Figure P Lag.3](#), suggest the PD controller is at least close to meeting the transient specifications. It is a happy accident that the steady-state error also improved; derivative compensation does not always do this. Let's use `stepinfo` to compute more accurate transient response characteristics of the PD-controlled system.

```
si_PD = stepinfo(y_PD,t_a);
disp(sprintf('settling time: %0.3g',si_PD.SettlingTime))
disp(sprintf('percent overshoot: %0.3g',si_PD.Overshoot))
```

```
settling time: 0.82
percent overshoot: 16.2
```

This is quite close to the specification. If desired, the gain K_2 and the zero location z_c could be tuned, iteratively.

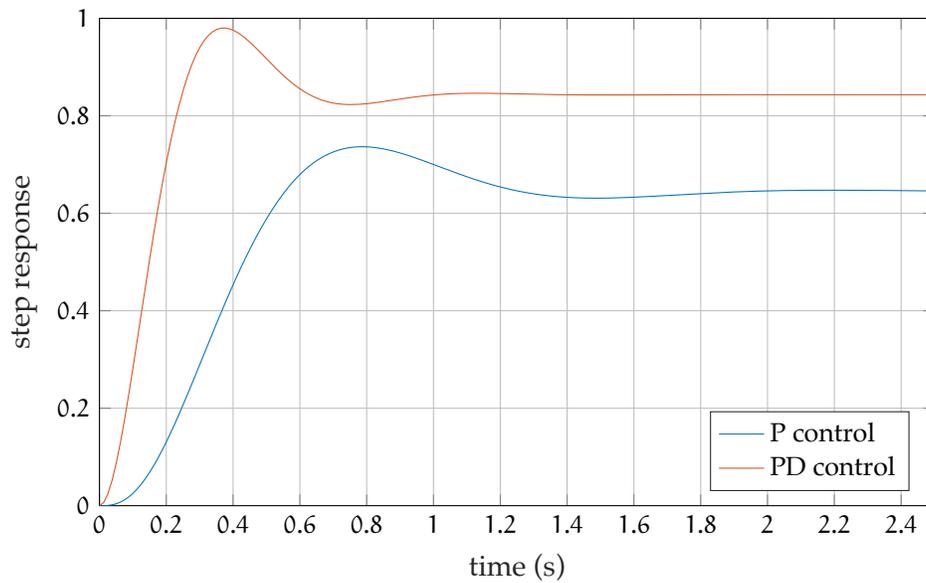


Figure PD.5: step responses for proportional and proportional-derivative controllers.