VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HCM UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING - MECHATRONICS DEPARTMENT

ME3011

PID Controller Design for Self-excited Brushed DC Motor

Submitted To:
Nguyen Quoc Chi
Asst. Professor
Department of Mechatronics Engineering

Submitted By:
Nguyen Quy Khoi
1852158
CC02
HK201

HCMC, January 9, 2021

Contents

List of Figures

Chapter 1

Modeling the motor

1.1 Electrical modeling

In this modeling project, the motor is DC brushed type. The type of control is field control, which means the armature supply voltage is fixed. Before deriving the transfer function, it is worth to mention that all variables are in standard SI units.

The magnetic torque of the motor is generated by the interaction of the stator field and the rotor field and is given by

$$T_m(t) = k_m i_g(t) \tag{1.1}$$

where

- $T_m(t)$ is the motor torque.
- k_m is a constant. The value depends on the motor's characteristics (number of poles, number of conductors, etc.)
- $i_g(t)$ is the rotor current (i.e. armature current).

Using Kirchoff's voltage law, the armature circuit equation is

$$v_g(t) = (R_g + R_f)i_g(t) + (L_g + L_f)\frac{di_g(t)}{dt} \tag{1.2}$$

where

- v_g is the supply voltage to the rotor (i.e. armature voltage). The value is constant since this is field control design.
- R_g is the resistance of the armature windings.
- R_f is the resistance of the field windings.
- L_g is the leakage inductance of the armature.
- L_f is the inductance of the field windings.

Assuming the generator voltage $v_g(t)$ is proportional to the field current $i_f(t)$, the relation between $V_g(t)$ and $V_f(t)$ using Laplace transform is

$$V_g(s) = k_g I_f(s) = \frac{k_g}{R_f + L_f s} V_f(s) \tag{1.3}$$

Using Laplace transform for equations 1.1 and 1.2 combining with 1.3, the relation is obtained

$$\frac{T_m(s)}{V_g(s)} = \frac{k_g k_m}{R_g + R_f + (L_g + L_f)s} = \frac{k_g k_m}{R_e + L_e s} \tag{1.4}$$

1.2 Mechanical modeling

The motor has its own inertia J_m and viscous damping coefficient b_m. Therefore, equivalent parameters are necessary to form relations between the rotational output load $\theta_L(t)$ and motor torque $T_m(t)$.

From machine element course, it is known that the relation between torque, rotational displacement and number of teeth follows the formula (assuming negligible backlash):

$$\frac{T_1}{T_2} = \frac{\theta_2}{\theta_1} = \frac{N_1}{N_2} \tag{1.5}$$

where

- T_1 is the torque generated by gear 1.
- T_2 is the torque generated by gear 2.
- θ_2 is the rotational displacement of gear 2.
- θ_1 is the rotational displacement of gear 1.
- N_1 is the number of teeth of gear 1.
- N_2 is the number of teeth of gear 2.

Using equation 1.5, the motor and load is related by

$$\frac{T_m(t)}{T_L(t)} = \frac{\theta_L(t)}{\theta_m(t)} = \frac{N_1}{N_2} = \frac{T_m(t)}{\left(J_e \dfrac{d^2\theta_L(t)}{dt^2} + b_e \dfrac{d\theta_L(t)}{dt}\right)} = n \tag{1.6}$$

where

- $T_L(t)$ is the torque of the load.
- $\theta_L(t)$ is the rotational displacement of the load.
- $\theta_m(t)$ is the rotational displacement of the motor.
- $J_e = J_m/n^2 + J_L$ is the equivalent inertia of the motor and load. The inertia is adjusted with gear ratio N_2/N_1.
- $b_e = b_m/n^2 + b_L$ is the equivalent viscous damping coefficient of the motor and load. The inertia is adjusted with gear ratio N_2/N_1.
- $n = N_1/N_2$ is the gear ratio.

Using Laplace transform, equation 1.6 becomes

$$T_m(s) = \Theta_L(s) n \left(J_e s^2 + b_e s\right) \tag{1.7}$$

1.3 Forming the transfer function

Combining equation 1.4 and 1.7, the required transfer function is

$$\frac{\Theta_L(s)}{V_f(s)} = \frac{k_g k_m/(nL_eJ_e)}{s(\frac{R_e}{L_e}+s)(s+\frac{b_e}{J_e})} \tag{1.8}$$

This is the transfer function of the plant in the unity feedback system. For illustrating how designing a PID controller is implemented, the parameters are chosen as follows (selected with consultations from the internet to resemble real-case scenario):

k_g	0.1
k_m	1
L_f	0.05
n	1/10
J_L	0.1
J_m	0.01
b_L	0.5
b_m	0.5
R_f	1000
R_g	1000

Thus,
$R_e = R_g + R_f = 1000 + 1000 = 2000$
$L_e = L_g + L_f = 0.025 + 0.025 = 0.05$
$J_e = J_m/n^2 + J_L = 0.01/0.1^2 + 0.1 = 1.1$
$b_e = J_m/n^2 + J_L = 0.5/0.1^2 + 0.5 = 5.5$
and the transfer function is

$$\frac{\Theta_L(s)}{V_f(s)} = \frac{18.2}{s\,(s+5)\,(s+40000)} \tag{1.9}$$

Since the pole $s = -40000$ has negligible effect compares to $s = 0$ and $s = -5$, the second order approximation of the transfer function is

$$\frac{\Theta_L(s)}{V_f(s)} = G(s) = \frac{18.2}{s\,(s+5)} \tag{1.10}$$

Chapter 2

Proportional controller design

2.1 Specifying the problem

In designing a proportional controller, the only parameter that can be improved is percent overshoot $\%OS$. Since there is no design requirement, the problem is randomly specified as:

Design the value of gain, K_p to yield 10% overshoot for a unit step input.

2.2 Solving the problem

Since the plant is approximated as a second-order system, the damping ratio ζ is

$$\zeta = \frac{-\ln(\%OS)}{\sqrt{\pi^2 + \ln^2(\%OS)}} = \frac{-\ln(10/100)}{\sqrt{\pi^2 + \ln^2(10/100)}} = 0.591 \tag{2.1}$$

A program is written for drawing the root locus and will be shown later. The purpose is to find the intersection between the desired damping ratio and and root locus. From the figure, it is found that $K_p = 0.9827$ at the closed-loop poles $s = -2.5 \pm j3.411$. The closed-loop transfer function with unity feedback is:

$$\frac{\Theta_L(s)}{V_f(s)} = \frac{17.885}{s^2 + 5s + 17.885} \tag{2.2}$$

The natural frequency is

$$\omega_n = \sqrt{17.885} = 4.23 \tag{2.3}$$

The settling time is

$$T_s = \frac{4}{\zeta\omega_n} = \frac{4}{0.591 \times 4.23} = 1.6 \tag{2.4}$$

The peak time is

$$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} = \frac{\pi}{4.23\sqrt{1-0.591^2}} = 0.92 \tag{2.5}$$

Since the system is Type 1, the static error constant for ramp input is

$$K_v = \lim_{s\to 0} sK_pG(s) = \frac{0.9827 \times 18.2}{5} = 3.577 \tag{2.6}$$

and the steady-state error is

$$e_{ramp}(\infty) = \frac{1}{K_v} = \frac{1}{3.577} = 0.28 \tag{2.7}$$

For unit step input, the steady-state error is $e_{step}(\infty) = 0$ (type 1 system).

2.3 Graphic results

2.3.1 Transfer function of the gain controller

```
from control import *
from numpy import *
from matplotlib.pyplot import *


G = tf(18.2, [1, 5, 0])
rlocus(G)

zeta = -log(0.1)/sqrt(pi**2 + log(0.1)**2)
x = array([-5, 0])

plot(x, tan(pi-arccos(zeta))*x)
plot(*step_response(feedback(G, 1)))
```

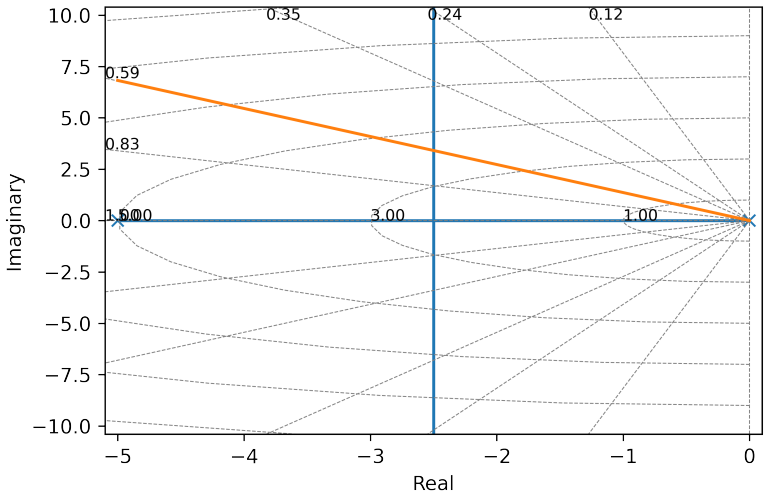

Figure 2.1: Root locus of the open-loop uncompensated system

2.3.2 Transient response of the system

```
from control import *
from numpy import *
from matplotlib.pyplot import *
```

```
G = tf(18.2, [1, 5, 0])
sys = feedback(0.9827*G, 1)

plot(*step_response(sys))
```

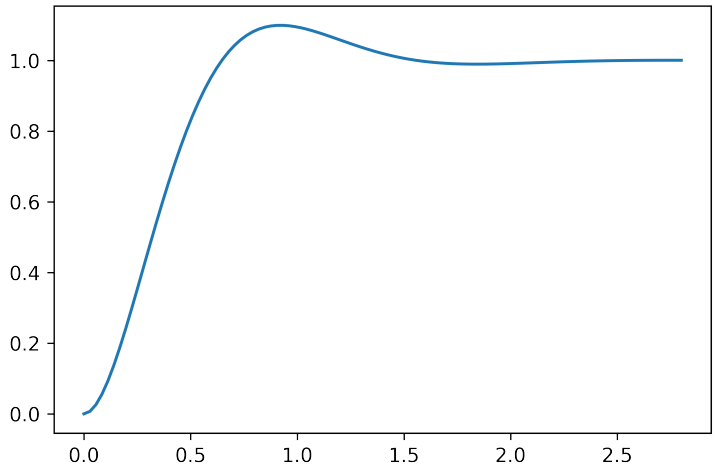

Figure 2.2: Step response of the system using proportional gain controller

Chapter 3

Ideal integral (PI) controller design

3.1 Specifying the problem

An ideal integral controller improves the steady-state error of the system. For a step input, the error is proven to be 0 in the previous chapter. Therefore, it is of interest to improve the error of a ramp input by designing an active controller. The design problem is

Design an active controller to yield 10% **overshoot and zero stead-state error for a ramp input.**

3.2 Solving the problem

Using the results from the previous chapter, the closed-loop poles $s = -2.5 \pm j3.411$ are used to find the appropriate zero in the ideal integral fraction $(s+z_c)/s$. Arbitrarily choose $z_c = 0.1$ and compare if the gain K is closed to the previous value $K_p = 0.9827$. Using $s = -2.5 + j3.411$ as the value for calculation, the angle is

$$\begin{aligned}\theta &= \angle(s+0.1) - 2[\angle(s+0)] - \angle(s+5)\\ &= \angle(-2.5+j3.411+0.1) - 2[\angle(-2.5+j3.411+0)] - \angle(-2.5+j3.411+5)\\ &= 125.13^\circ - 2\times 126.24^\circ - 53.76^\circ\\ &= -181.11^\circ \approx -180^\circ\end{aligned}$$

which is near the root locus but does not significantly affect the transient response. The corresponding gain is

$$\begin{aligned}K &= \frac{|s|}{|s+0.1|}\frac{|s||s+5|}{18.2}\\ &= \frac{|-2.5+j3.411|}{|-2.5+j3.411+0.1|}\frac{|-2.5+j3.411||-2.5+j3.411+5|}{18.2}\\ &= 0.996 \approx 1\end{aligned}$$

Thus, the transfer function for the ideal integral controller is

$$G_c(s) = K\frac{s+0.1}{s} = 0.996 + \frac{0.1}{s} \tag{3.1}$$

In summary, the proportional gain is $K_p = 1$ and integral gain is $K_i = 0.1$. From the figure, the response uses an active PI controller converges to ramp input, which satisfies the requirement.

3.3 Graphic results

3.3.1 Transfer function of PI controller

```
from control import *
from numpy import *
from matplotlib.pyplot import *


s = tf('s')
G = tf([18.2],[1,5,0])
rlocus(G)

zeta = -log(0.1) / sqrt(pi**2 + log(0.1)**2)
x = array([-10,0])

plot(x, tan(pi-arccos(zeta))*x)
print(0.996*array([1,0.1]))
```

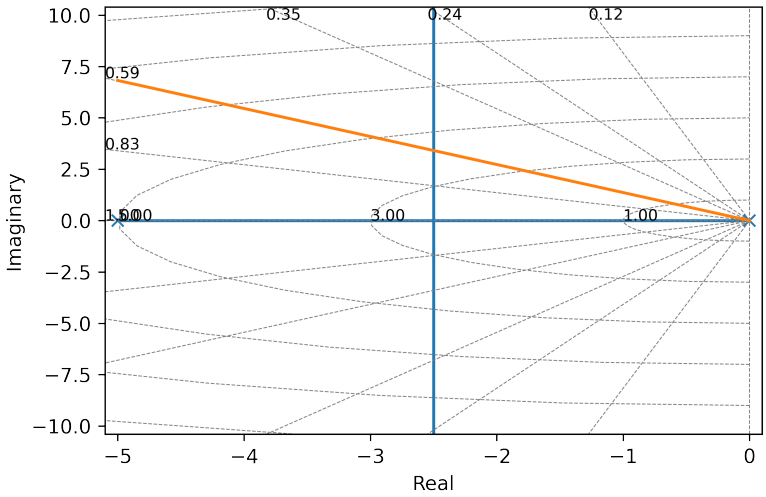

Figure 3.1: Root locus of the open-loop uncompensated system

3.3.2 Transient response of the system

Step response

```
from control import *
from numpy import *
from matplotlib.pyplot import *


s = tf('s')
G = tf([18.2],[1,5,0]) * (1+0.1/s)
```

```
sys = feedback(0.996*G, 1)

T = linspace(0,5,1000)

plot(*step_response(sys, T=T))
```

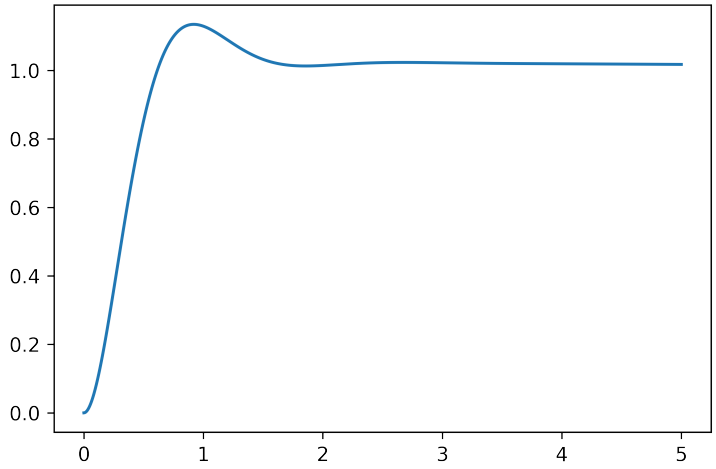

Figure 3.2: Step response of the system using PI controller

Ramp response

```
from control import *
from numpy import *
from matplotlib.pyplot import *


s = tf('s')
G = tf([18.2],[1,5,0]) * (1+0.1/s)
sys = feedback(0.996*G, 1)

U = T = linspace(0, 2, 1000)
yout, T, xout = lsim(sys, U, T)

plot(T, U, color='gray', label='ramp input')
plot(T, yout, color='blue', label='ramp response')
legend()
```

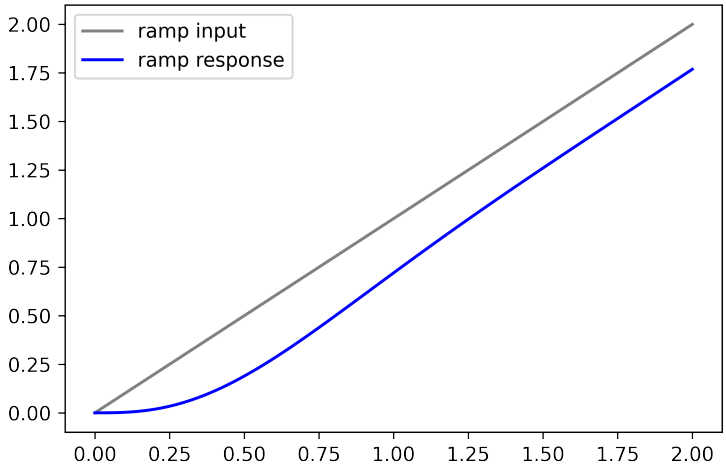

Figure 3.3: Ramp response of the system using PI controller

Chapter 4

PID controller design

4.1 Specifying the problem

There are 2 simple approaches in designing a controller: active (PD design then PI design) or passive (lead compensator design then lag compensator design), which are referred to as PID and lag-lead compensator. In this modeling project, the active controller PID is used. The design technique consists of the following steps:

1. Evaluate the performance of the uncompensated system to determine how much improvement in transient response is required.
2. Design the PD controller to meet the transient response specifications. The design includes the zero location and the loop gain.
3. Simulate the system to be sure all requirements have been met.
4. Redesign if the simulation shows that requirements have not been met.
5. Design the PI controller to yield the required steady-state error.
6. Determine the gains, K_p, K_i and K_d.
7. Simulate the system to be sure all requirements have been met.
8. Redesign if simulation shows that requirements have not been met.

Combining all design problems from previous chapters, the design requirements for PID controller is

Design a PID controller so that the system can operate with a 2 fold improvement in settling time at 10% **overshoot and zero steady-state error for a ramp input.**

4.2 Ideal derivative (PD) controller

Following the guidelines above, the ideal derivative controller is designed first. Using the calculated result, the uncompensated closed-loop poles are $s = -2.5 \pm j3.411$ with a gain $K = 0.9827$. The settling time must be a half of the uncompensated one. Therefore,

$$T_s = \frac{4}{\zeta\omega_n} = \frac{4}{0.591\omega_n} = 1.6/2 \Rightarrow \omega_n = 8.46 \tag{4.1}$$

The real part of the compensated pole is:

$$\sigma_d = -\zeta\omega_n = -0.591 \times 8.46 = -5 \tag{4.2}$$

The imaginary part of the compensated pole is:

$$\omega_d = \omega_n\sqrt{1-\zeta^2} = 8.46\sqrt{1-0.591^2} = 6.824 \tag{4.3}$$

Therefore, the desired compensated closed-loop pole are $s = -5 \pm j6.824$. Assume that the compensator zero is located at $-z_c$ and the value $s = -5 + j6.824$, the total angle is

$$\begin{aligned}\theta &= \angle(s+z_c) - \angle(s+0) - \angle(s+5)\\ &= \angle(-5+j6.824+z_c) - 126.23^\circ - 90^\circ = -180^\circ\\ &\Leftrightarrow \angle(-5+j6.824+z_c) = 36.23^\circ\\ &\Leftrightarrow \frac{6.824}{-5+z_c} = \tan 36.23^\circ\\ &\Leftrightarrow z_c = 14.31\end{aligned}$$

Thus, the PD controller is $G_{PD} = s + 14.31$. With the new zero introduced to the open-loop transfer function, another gain K must be found such that the damping ratio $\zeta = 0.591$ intersects the root locus of equation $K(s+14.31)\frac{18.2}{s(s+5)}$. The corresponding gain K is

$$\begin{aligned}K &= \frac{1}{|s+14.31|}\frac{|s||s+5|}{18.2}\\ &= \frac{1}{|-5+j6.824+14.31|}\frac{|-5+j6.824||-5+j6.824+5|}{18.2}\\ &= 0.042\end{aligned} \tag{4.4}$$

Thus, the transfer function of PD controller is

$$G_{PD} = K(s+14.31) = 0.042s + 0.6 \tag{4.5}$$

4.3 Ideal integral (PI) controller

The PI controller reduces the steady-state error of ramp input $e_{ramp}(\infty)$ to 0. Arbitrarily choose the zero $z_c = -0.1$ so that the controller does not significantly affect the transient response of the PD compensated system. Thus, the transfer function is

$$G_{PI} = K\frac{s+0.1}{s} \tag{4.6}$$

Using the root locus program, $s = -4.422 \pm j6.034$ is selected. Comparison is necessary to see if the gain K is closed to the previous value $K_p = 0.9827$. Using $s = -4.422 + j6.034$ as the value for calculation, the angle is

$$\begin{aligned}\theta &= \angle(s+14.31) + \angle(s+0.1) - 2[\angle(s+0)] - \angle(s+5)\\ &= \angle(-4.422+j6.034+14.31) + \angle(-4.422+j6.034+0.1)\\ &\qquad -2[\angle(-4.422+j6.034+0)] - \angle(-4.422+j6.034+5)\\ &= 31.393^\circ + 125.613^\circ - 2\times 126.236^\circ - 84.528^\circ\\ &= -179.994^\circ \approx -180^\circ\end{aligned}$$

which is near the root locus but does not significantly affect the transient response. The corresponding gain K is

$$\begin{aligned} K &= \frac{|s|}{|s+0.1|}\frac{|s||s+5|}{18.2|s+14.31|} \\ &= \frac{|-4.422+j6.034|}{|-4.422+j6.034+0.1|}\frac{|-4.422+j6.034||-4.422+j6.034+5|}{18.2|-4.422+j6.034+14.31|} \\ &= 0.217 \end{aligned}$$

In summary, the transfer function of the PID controller is

$$G_{PID} = 0.217\frac{(s+0.1)(s+14.31)}{s} = 0.217s + 3.127 + \frac{0.31}{s} \tag{4.7}$$

with $K_p = 3.127$, $K_i = 0.31$ and $K_d = 0.217$.

4.4 Graphic results

4.4.1 Compensated pole of the PD controller z_c

```
from control import *
from numpy import *
from matplotlib.pyplot import *
import sympy as sp


G = tf([18.2], [1,5,0])
rlocus(G)

zeta = -log(0.1) / sqrt(pi**2 + log(0.1)**2)
x = array([-10,0])

plot(x,tan(pi-arccos(zeta))*x)

Ts = 1.6/2
omg_n = 4/Ts/zeta
sig_d = -zeta*omg_n
omg_d = omg_n*sqrt(1-zeta**2)
s_d = sig_d + 1j*omg_d

z_c = sp.symbols('z_c')
f = omg_d/(sig_d+z_c) - tan(angle(s_d+0) + angle(s_d+5) - pi)
z_c = sp.solve(f)
print(z_c)
```

4.4.2 Transfer function of the PID controller

```
from control import *
from numpy import *
```

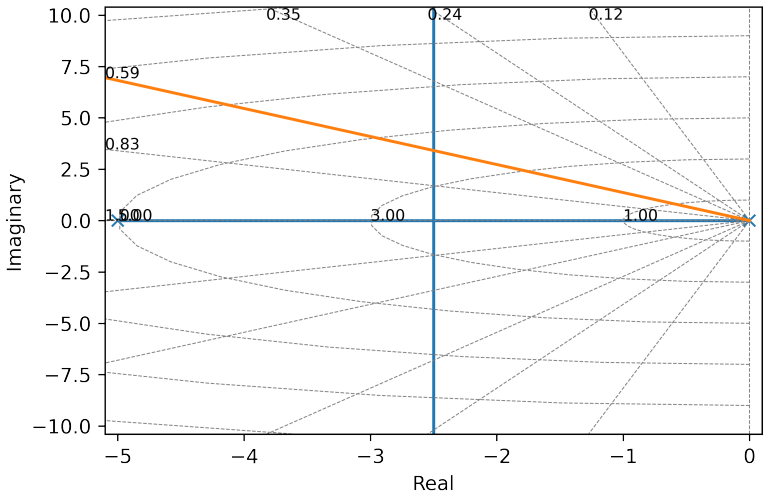

Figure 4.1: Root locus of the open-loop uncompensated system

```
from matplotlib.pyplot import *


s = tf('s')
G = tf([18.2],[1,5,0]) * (s+14.31) * (1+0.1/s)
rlocus(G)

zeta = -log(0.1) / sqrt(pi**2 + log(0.1)**2)
x = array([-10,0])

plot(x, tan(pi-arccos(zeta))*x)
print(0.217*poly([-0.1,-14.31]))
```

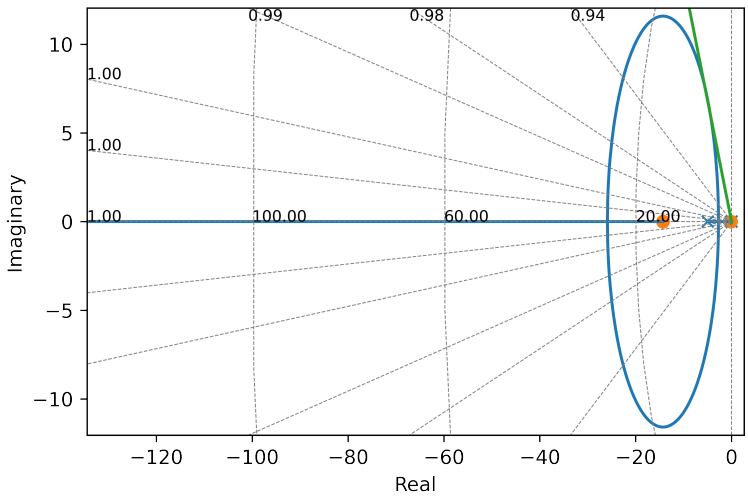

Figure 4.2: Root locus of the open-loop PD compensated system

4.4.3 Transient response of the system

Step response

```
from control import *
from numpy import *
from matplotlib.pyplot import *


s = tf('s')
G = tf([18.2],[1,5,0]) * (s+14.31) * (1+0.1/s)
sys = feedback(0.217*G, 1)

T = linspace(0, 3, 1000)

plot(*step_response(sys, T=T))
```

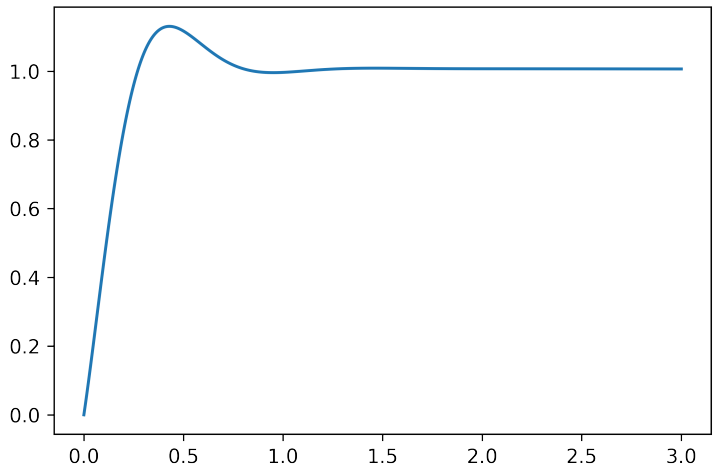

Figure 4.3: Step response of the system using PID controller

Ramp response

```
from control import *
from numpy import *
from matplotlib.pyplot import *


s = tf('s')
G = tf([18.2],[1,5,0]) * (s+14.31) * (1+0.1/s)
sys = feedback(0.217*G, 1)

U = T = linspace(0, 0.8, 1000)
yout, T, xout = lsim(sys, U, T)
```

```
plot(T, U, color='gray', label='ramp input')
plot(T, yout, color='blue', label='ramp response')
legend()
```

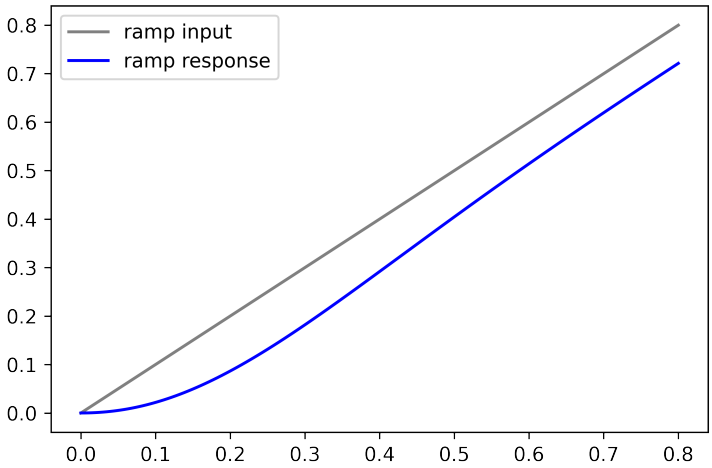

Figure 4.4: Ramp response of the system using PID controller