CHEN 461 HW9 - Mark Levchenko

March 29, 2023

[ ]:
```python
import control

import numpy as np
import matplotlib.pyplot as plt

from sympy.abc import t, s
from sympy import symbols, simplify, expand
from sympy.series import limit
```

1 Problem 10.8

1.1 Part a

Real PD:

$G(s) = k_c \left(\frac{1+\tau_D s}{1+\alpha\tau_D s} \right)$

$E(s) = \frac{1}{s}$

$U(s) = k_c \left(\frac{1+\tau_D s}{s(1+\alpha\tau_D s)} \right)$

$U(s) = k_c \left(\frac{1}{s} + \frac{\tau_D(1-\alpha)}{(1+\alpha\tau_D s)} \right)$

$u(t) = k_c \left(1 + \frac{\tau_D(1-\alpha)}{\alpha\tau_D} \right) \exp\left(-\alpha^{-1} \frac{t}{\tau_D} \right)$

Real PD response:

$$\frac{u(t)}{k_c} = 1 + \frac{(1-\alpha)}{\alpha} \exp\left(-\alpha^{-1} \frac{t}{\tau_D} \right)$$

Ideal PD:

$G(s) = k_c (1 + \tau_D s)$

$U(s) = k_c \left(\frac{1}{s} + \tau_D \right)$

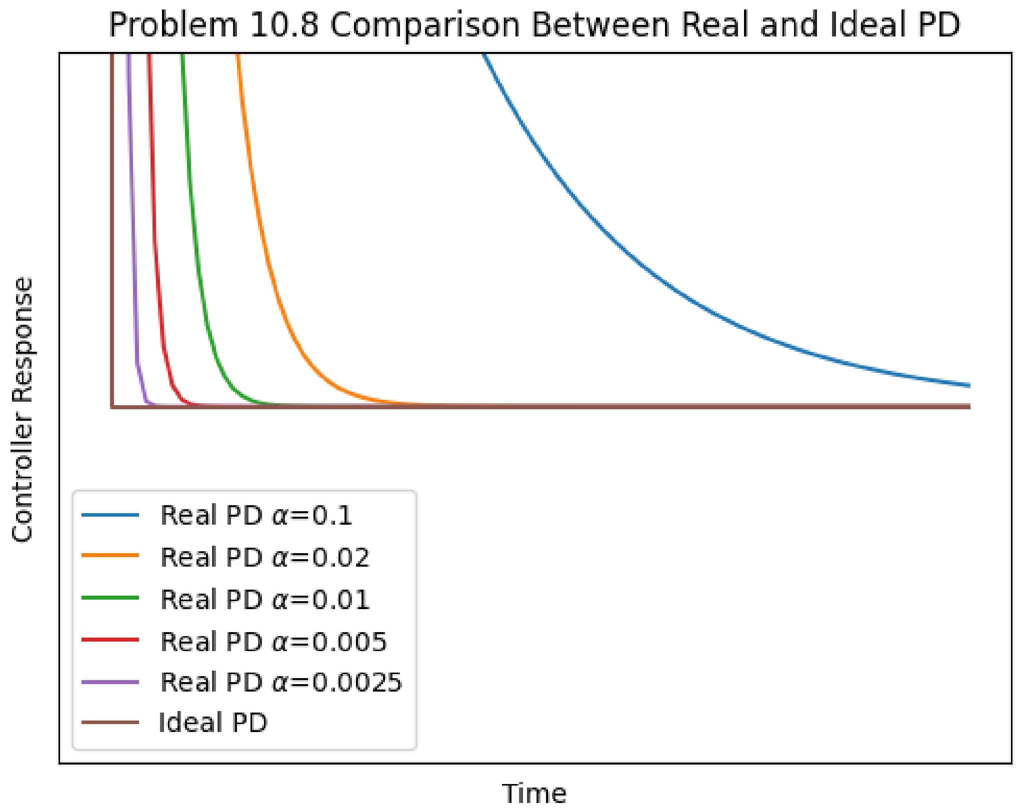
Ideal PD response:

$$\frac{u(t)}{k_c} = 1 + \delta(t)$$

1.1.1 Simulation

[ ]:
```python
alpha = [0.1, 0.02, 0.01, 0.005, 0.005/2]

def real_pd(t, alpha):
    return 1 + (1 - alpha) / alpha * np.exp(-t / alpha)

t_values = np.linspace(0, .5, 100)

for a in alpha:
    plt.plot(t_values, real_pd(t_values, a), label=r"Real PD $\alpha$="+f"{a}")

ideal_pd_response = np.ones(t_values.shape[0])
ideal_pd_response[0] = 1e300

plt.plot(t_values, ideal_pd_response, label="Ideal PD")

plt.ylim([0, 2])
plt.xlabel("Time")
plt.ylabel("Controller Response")
plt.title("Problem 10.8 Comparison Between Real and Ideal PD")
plt.xticks([], [])
plt.yticks([], [])
plt.legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x21a89221550>
```



The closer that α is to zero, the closer closer that the Real PD is to the Ideal PD. The smaller the α value, the faster that the Real PD responds to the error.

1.2 Part B

Real PD:

$$G(s) = k_c \left(\frac{1 + \tau_D s}{1 + \alpha \tau_D s} \right)$$

Ideal PD:

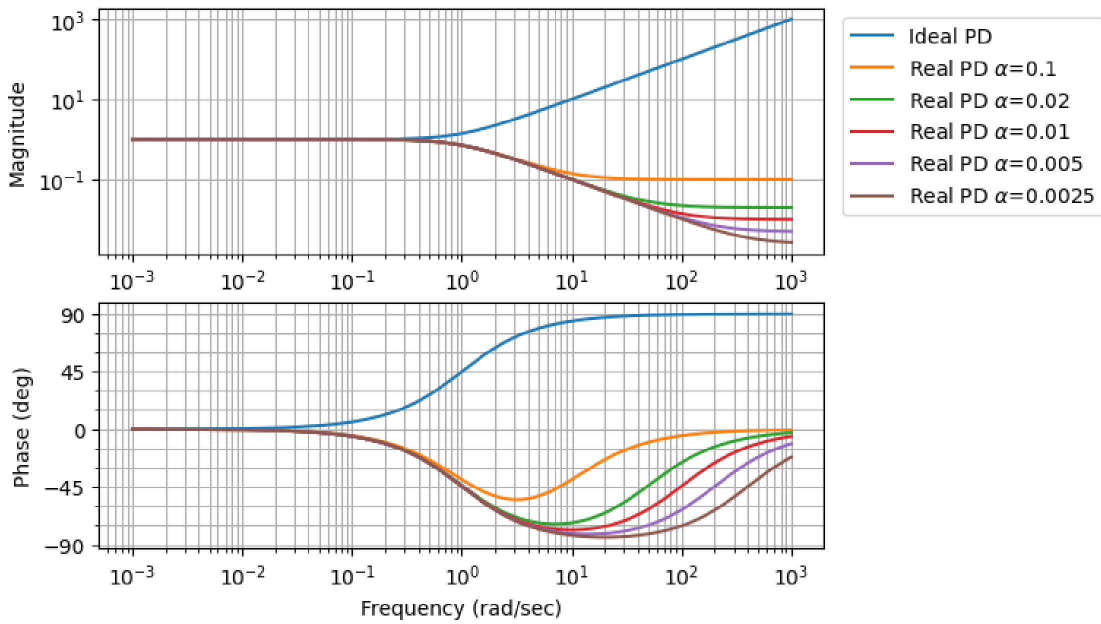
$$G(s) = k_c \left(1 + \tau_D s \right)$$

1.2.1 Bode Diagrams

[ ]:
```
alpha = [0.1, 0.02, 0.01, 0.005, 0.005/2]

ideal_pd = control.tf([1, 1], [1])

w = np.linspace(1e-3, 1e3, int(1e5))
```

```
mag, phase, omega = control.bode_plot(ideal_pd, omega=w, wrap_phase=True,␣
↪label="Ideal PD")

for a in alpha:
    real_pd = control.tf([a, 1], [1, 1])
    mag, phase, omega = control.bode_plot(real_pd, omega=w, label=r"Real PD␣
↪$\alpha$="+f"{a}")

plt.legend(loc="upper right", bbox_to_anchor=(1.4, 2.2))
```

[ ]: <matplotlib.legend.Legend at 0x21a8b9711d0>



2 Problem 10.10

$$\frac{de_I}{dt} = e$$

$$\frac{de_F}{dt} = \frac{e - e_F}{\tau_F}$$

$$u = k_c \left(e + \frac{e_I}{\tau_I} + \frac{\tau_D}{\tau_F} (e - e_F) \right)$$

3 Problem 11.8

3.1 Part A

Feedback loop transfer function:

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

Process transfer function:

$$G_p(s) = \frac{k_p}{\tau^2 s^2 + 2\zeta\tau s + 1}$$

Controller transfer function:

$$G_c(s) = k_c \left(\frac{1 + \tau_D s}{1 + \alpha\tau_D s} \right)$$

3.1.1 Define feedback transfer function symbolically

[ ]:
```
tau, zeta, tau_D, alpha = symbols('tau, zeta, tau_D, alpha')
k_c, k_p = symbols('k_c, k_p')

G_p = k_p / (tau**2 * s**2 + 2 * zeta * tau * s + 1)
G_c = k_c * (1 + tau_D * s) / (1 + alpha * tau_D * s)

G_feedback = simplify(G_c * G_p / (1 + G_c * G_p))

G_feedback
```

[ ]:
$$\frac{k_c k_p (s\tau_D + 1)}{k_c k_p (s\tau_D + 1) + (\alpha s\tau_D + 1)(s^2\tau^2 + 2s\tau\zeta + 1)}$$

3.2 Part B

Find offset by Final Value Theorem:

$$\lim_{s \to 0^+} sY(s) = \lim_{t \to \infty} y(t)$$

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

$Y(s) = G(s) Y_{sp}(s)$

In deviation form:

$y_{sp}(t) = 1$

$Y_{sp}(s) = \frac{1}{s}$

$Y(s) = \frac{G(s)}{s}$

$sY(s) = s\frac{G(s)}{s} = G(s)$

Offset is $y_{sp}(t) - y(t)$ at ∞

$y(t) = \lim_{s \to 0^+} G(s)$

Offset:

$$1 - \lim_{s \to 0^+} G(s)$$

[ ]:
```
simplify(1 - limit(G_feedback, s, 0))
```

[ ]: $\frac{1}{k_c k_p + 1}$