Lead Compensation

With student number 03116, the transfer function is

$$G(s) = \frac{3}{s^2 + s} \tag{1}$$

with a desired peak at 0.25 s after excitation with an overshoot of no more than 40%. The overall transfer function of (1) in a unity gain negative feedback is

$$H(s) = \frac{G(s)}{1 + G(s)} \tag{2}$$

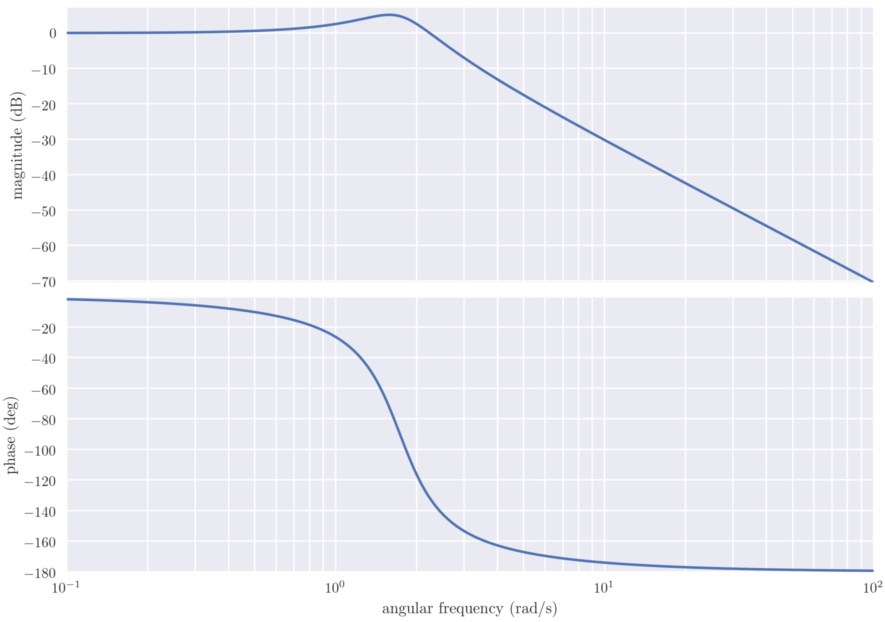

Figure 1: Bode plot of (2).

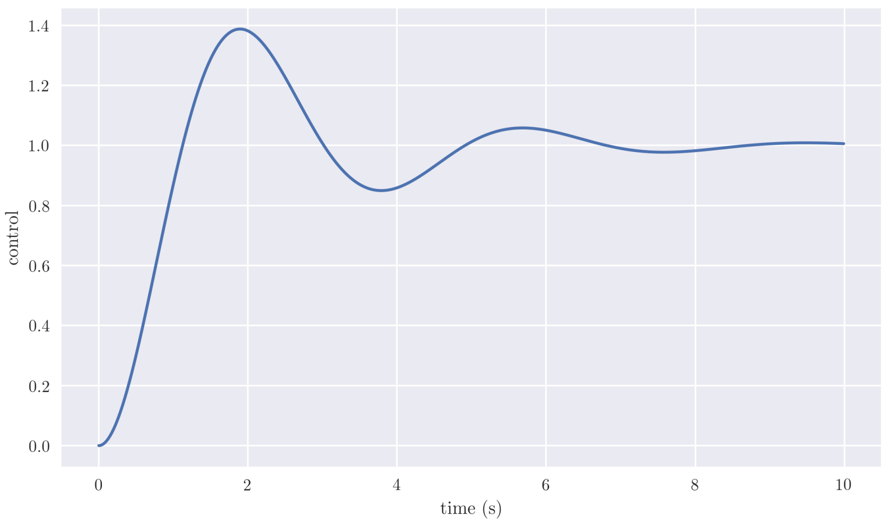

Figure 2: Step function response of (2).

(a) **Desired pole location**

$$s_d = -\sigma_d \pm j\omega_d \tag{3}$$
$$= -\zeta\omega_n + j\omega_n\sqrt{1-\zeta^2} \tag{4}$$
$$\zeta = \frac{-\ln(\%OS)}{\sqrt{\pi^2 + \ln^2(\%OS)}} \tag{5}$$
$$= \frac{-\ln(0.4)}{\sqrt{\pi^2 + \ln^2(0.4)}} \tag{6}$$
$$\boxed{\zeta = 0.28} \tag{7}$$
$$\omega_n = \frac{\pi}{T_p\sqrt{1-\zeta^2}} \tag{8}$$
$$= \frac{\pi}{0.25\sqrt{1-0.28^2}} \tag{9}$$
$$\boxed{\omega_n = 13.09} \tag{10}$$
$$\boxed{s_d = -3.67 + 12.57j} \tag{11}$$

(b) **Angle deficiency**

$$G(s_d) = \frac{3}{s_d^2 + s_d} \tag{12}$$
$$= -0.02 + 0.01j \tag{13}$$
$$\measuredangle G(s_d) = 203.81^\circ \tag{14}$$
$$\Phi_d = 180 - \measuredangle G(s_d) \tag{15}$$
$$\boxed{\Phi_d = 0.49 \text{ rad} = 28.23^\circ} \tag{16}$$

(c) **Compensator poles and zeros**

$$\alpha = \arctan\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right) \tag{17}$$

$$= \arctan\left(\frac{\sqrt{1-0.28^2}}{0.28}\right) \tag{18}$$

$$\boxed{\alpha = 1.29} \tag{19}$$

$$z_c = -\omega_n\sqrt{1-\zeta^2}\tan\left(\frac{\alpha-\Phi_d}{2}\right) - \zeta\omega_n \tag{20}$$

$$= -(13.09)\sqrt{1-0.28^2}\tan\left(\frac{1.29-0.49}{2}\right) - (0.28)(13.09) \tag{21}$$

$$\boxed{z_c = -8.94} \tag{22}$$

$$p_c = -\omega_n\sqrt{1-\zeta^2}\tan\left(\frac{\alpha+\Phi_d}{2}\right) - \zeta\omega_n \tag{23}$$

$$= -(13.09)\sqrt{1-0.28^2}\tan\left(\frac{1.29+0.49}{2}\right) - (0.28)(13.09) \tag{24}$$

$$\boxed{p_c = -19.18} \tag{25}$$

(d) **Compensator gain**

$$K_c = \frac{1}{\left|G(s_d)\dfrac{s_d+z_c}{s_d+p_c}\right|} \tag{26}$$

$$= \frac{1}{\left|-0.02+0.01j\dfrac{-3.67+12.57j-8.94}{-3.67+12.57j-19.18}\right|} \tag{27}$$

$$\boxed{K_c = 82.11} \tag{28}$$

(e) **Simulink verification**

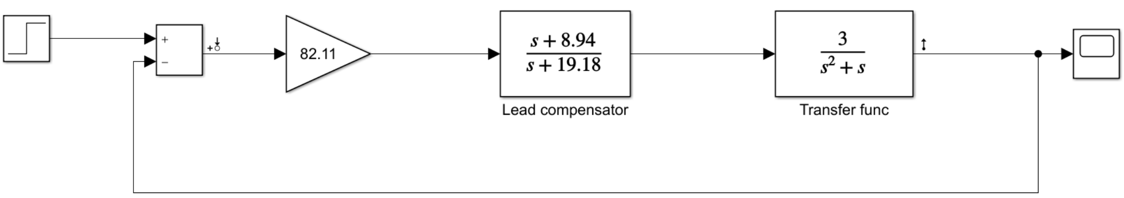

Figure 3: Block diagram design of (2) with a lead compensator.

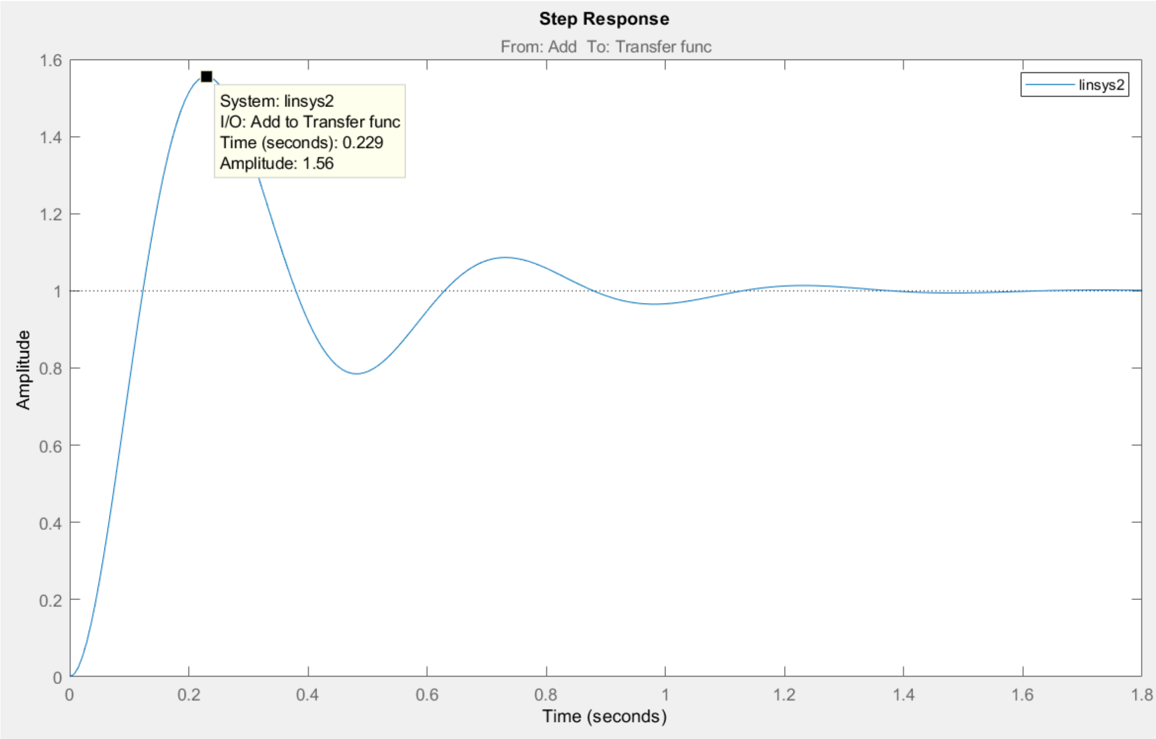

Figure 4: Step function response of the system with a lead compensator. Due to rounding-off errors, the peak time occurs at $t = 0.229$ s.

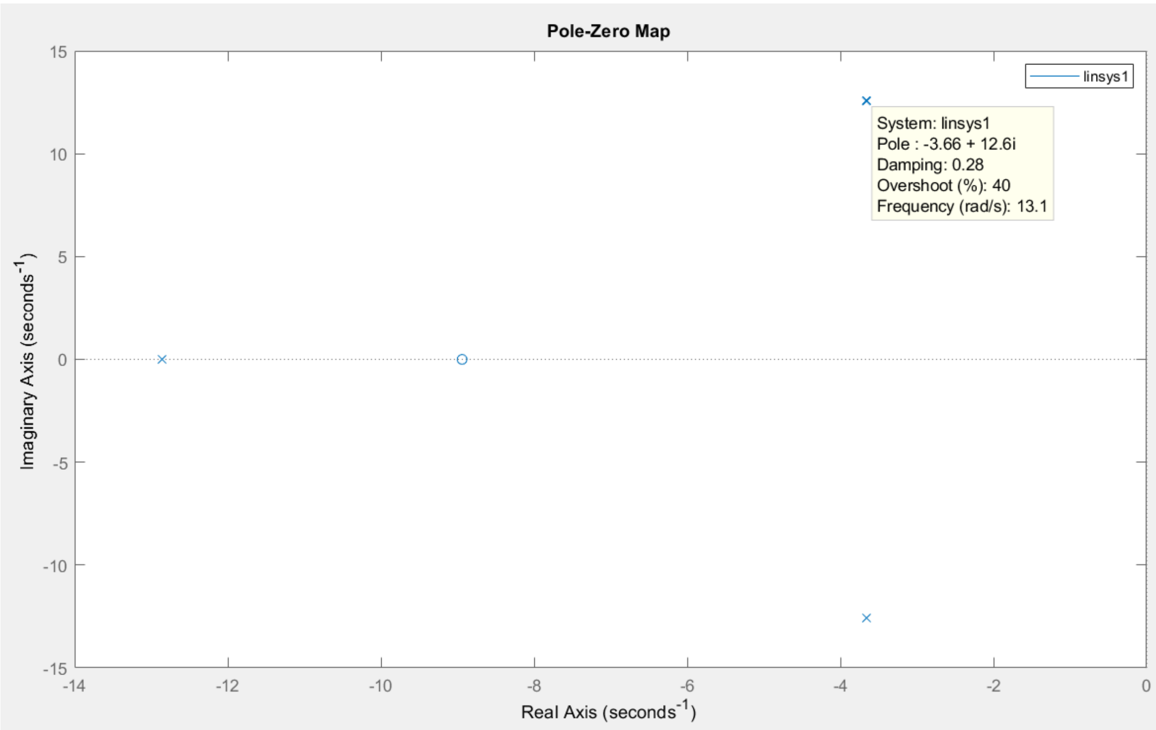

Figure 5: Pole-zero map of the system with a lead compensator. The percent overshoot is exactly 40%.

Appendix

Listing 1: Source code.

```
import numpy as np
import matplotlib.pyplot as mp
import matplotlib.ticker as tick


class LeadCompensator:

    def __init__(self, w):
        self.w = w
        self.s = 1j*w

    def initTransferFunc(self, G):
        self.G = G

    def initNegativeFeedback(self, gain):
        self.k = gain
        self.H = self.G(self.s)/(1 + self.k*self.G(self.s))
        self.magnitude = 20*np.log10(self.H)
        self.phase = np.degrees(np.arctan2(self.H.imag, self.H.real))

    def BodePlot(self, save=False, savename=None):
        fig = mp.figure(figsize=(5*16/9, 5*1.25))

        ax = fig.add_subplot(211)
        ax.plot(self.w, self.magnitude)
        ax.set_xscale("log")
        ax.grid(True, which="both")
        ax.set_ylabel("magnitude␣(dB)")
        ax.set_xlim(self.w.min(), self.w.max())
        ax.set_ylim(self.magnitude.min(), self.magnitude.max()+2)
        ax.xaxis.set_major_formatter(tick.NullFormatter())

        ax = fig.add_subplot(212)
        ax.plot(self.w, self.phase)
        ax.set_xscale("log")
        ax.grid(True, which="both")
        ax.set_xlabel("angular␣frequency␣(rad/s)")
        ax.set_ylabel("phase␣(deg)")
        ax.set_xlim(self.w.min(), self.w.max())
        ax.set_ylim(self.phase.min()-1, self.phase.max()+1)

        mp.tight_layout()
        if save:
            mp.savefig(savename, dpi=300, bbox_inches="tight")
        mp.show()

    def initDesired(self, percent_overshoot, Tp):
        self.zeta = -np.log(percent_overshoot)/np.sqrt(np.pi**2 + \
                                            np.log(percent_overshoot)**2)
        self.wn = np.pi/(Tp * np.sqrt(1 - self.zeta**2))
        self.sd = -self.zeta*self.wn + 1j*self.wn*np.sqrt(1 - self.zeta**2)
        self.Gsd = self.G(self.sd)
        phiGsd = np.arctan2(self.Gsd.imag, self.Gsd.real)
        self.phid = np.pi - phiGsd
```

```
    def initCompensator(self):
        self.alpha = np.arctan2(np.sqrt(1 - self.zeta**2), self.zeta)
        self.zc = -self.wn*np.sqrt(1 - self.zeta**2) * \
                    np.tan((self.alpha - self.phid)/2) - self.zeta*self.wn
        self.pc = -self.wn*np.sqrt(1 - self.zeta**2) * \
                    np.tan((self.alpha + self.phid)/2) - self.zeta*self.wn
        self.K = 1/abs(self.Gsd*(self.sd + self.zc)/(self.sd + self.pc))

def G(s):
    return 3/(s**2 + s)

w = np.logspace(-1, 2, 500)
sys = LeadCompensator(w)
sys.initTransferFunc(G)
sys.initNegativeFeedback(1)
sys.BodePlot(True, "sys_bode.png")
sys.initDesired(0.4, 0.25)
sys.initCompensator()
```