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%.

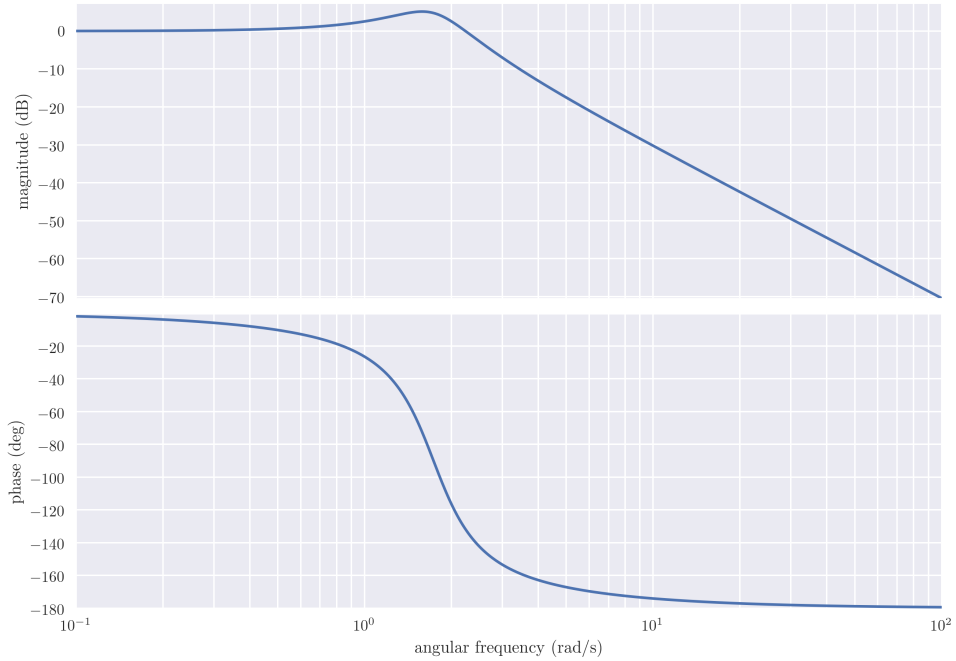

Figure 1: Bode plot of (1) in a unity gain negative feedback.

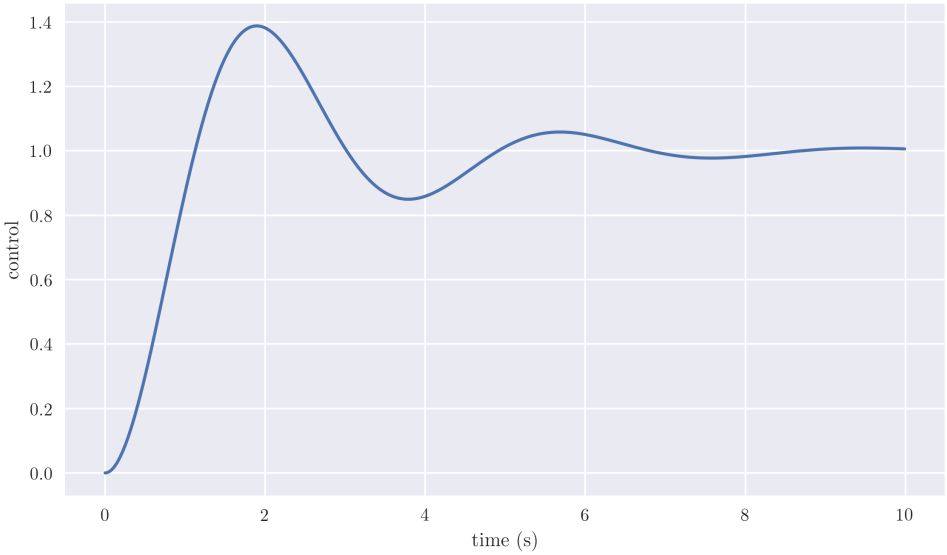

Figure 2: Step function response of (1).

(a) **Desired pole location**

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

(b) **Angle deficiency**

$$G(s_d) = \frac{3}{s_d^2 + s_d} \tag{11}$$
$$= -0.02 + 0.01j \tag{12}$$
$$\measuredangle G(s_d) = 203.81^\circ \tag{13}$$
$$\Phi_d = 180 - \measuredangle G(s_d) \tag{14}$$
$$\boxed{\Phi_d = 23.81^\circ} \tag{15}$$

(c) **Compensator poles and zeros**

$$\alpha = \arctan\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right) \tag{16}$$
$$z_c = -\omega_n\sqrt{1-z^2}\tan\left(\frac{\alpha - \Phi_d}{2}\right) - \zeta\omega_n \tag{17}$$
$$p_c = -\omega_n\sqrt{1-z^2}\tan\left(\frac{\alpha + \Phi_d}{2}\right) - \zeta\omega_n \tag{18}$$

Appendix

Listing 1: Source code

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


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.degrees(np.arctan2(self.Gsd.imag, self.Gsd.real))
```

```
55          self.phid = 180 - phiGsd
56
57      def initCompensator(self, z):
58          alpha = np.arctan2(np.sqrt(1 - self.zeta**2), self.zeta)
59          self.zc = -self.wn*np.sqrt(1 - z**2) * \
60                      np.tan((alpha - self.phid)/2) - self.zeta*self.wn
61          self.pc = -self.wn*np.sqrt(1 - z**2) * \
62                      np.tan((alpha + self.phid)/2) - self.zeta*self.wn
63          self.K = 1/abs(self.Gsd*(self.sd + self.pc)/(self.sd + self.zc))
64
65
66  def G(s):
67      return 3/(s**2 + s)
68
69
70  w = np.logspace(-1, 2, 500)
71  sys = LeadCompensator(w)
72  sys.initTransferFunc(G)
73  sys.initNegativeFeedback(1)
74  sys.BodePlot(True, "sys_bode.png")
75  sys.initDesired(0.4, 0.25)
```