```
# ************************* HW13 ************************
from control import tf, bode, margin, step_response, mag2db, tf2ss
import matplotlib.pyplot as plt
import numpy as np
import loopshape_tools as lt
import massParam as P

# Define Plant
Plant = tf([1/P.m],[1, P.b/P.m, P.k1/P.m])

# Define PID Controller
kd = 2.9
kp = 5
ki = 0.1
C_pid = tf([kd, kp, ki], [1.0, 0])
C = C_pid

# ************************************************************
'''
1) get_control_integral(ki):            controls system type at low-freq
2) get_control_proportional(kp):        controls w_co
3) get_control_lead(omega_lead, M):     controls PM
4) get_control_lpf(p):                  controls noise reduction - also does
 prefilter
5) get_control_lag(z, M):               controls ref tracking
'''

# # Integral control
# Ki = 1
# C_int = lt.get_control_integral(Ki)
# C = C*C_int

# Proportional control
Kp = 0.05
C_prop = lt.get_control_proportional(Kp)
C = C*C_prop

# Lead Compensator
omega_z = 5
M = 100
C_lead = lt.get_control_lead(omega_z, M)
C = C*C_lead

# Low-pass filter
omega_lpf = 15
C_lpf = lt.get_control_lpf(omega_lpf)
C = C*C_lpf

# Low-pass filter
omega_lpf = 20
```

```
C_lpf = lt.get_control_lpf(omega_lpf)
C = C*C_lpf

# Phase Lag
omega_z = 1
M = 200
C_lag = lt.get_control_lag(omega_z, M)
C = C*C_lag

# Get Prefilter
omega_pre = 1
F = lt.get_control_lpf(omega_pre)
# F = tf([1], [1])

# Get TF num and den
C_num = np.asarray(C.num[0])
C_den = np.asarray(C.den[0])
F_num = np.asarray(F.num[0])
F_den = np.asarray(F.den[0])

print("Controller: ", C)
print("Prefilter: ", F)


if __name__=="__main__":
    dB_flag = False

    # -------------------- Plotting --------------------
    # Plant
    mag, phase, omega = bode(Plant,
                             dB=dB_flag,
                             omega=np.logspace(-4, 5),
                             label='P(s) - Plant',
                             plot=True)

    gm, pm, Wcg, Wcp = margin(Plant)
    print("for original system:")
    print(" pm: ", pm, " Wcp: ", Wcp, "gm: ", gm, " Wcg: ", Wcg)

    # -------------------- Design Specifications --------------------
    # ----------- Reference Tracking -----------
    omega_r = 0.001
    gamma_r = 10**-4

    lt.add_spec_ref_tracking(gamma_r, omega_r)

    # ----------- Noise Attenuation -----------
    omega_n = 1000
    gamma_n = 10**-5
    lt.add_spec_noise(gamma_n, omega_n)
```

```
    # ------------------------------------------------------------------

    # -------------------- Plotting --------------------
    # Plant + PID Controller
    mag, phase, omega = bode(Plant*C_pid,
                             dB=dB_flag,
                             omega=np.logspace(-4, 5),
                             label='P(s)C_pid(s) - Open Loop',
                             plot=True,
                             margins=True)

    # Plant + Controller
    mag, phase, omega = bode(Plant*C,
                             dB=dB_flag,
                             omega=np.logspace(-4, 5),
                             label='P(s)C(s) - Open Loop',
                             plot=True,
                             margins=True)

    gm, pm, Wcg, Wcp = margin(Plant * C)
    print("for final C*P:")
    print(" pm: ", pm, " Wcp: ", Wcp, "gm: ", gm, " Wcg: ", Wcg)

    print(" ")
    omega_n = 20
    magN, _, _ = bode(Plant*C, dB=dB_flag, omega=omega_n, plot=False)
    print("Noise attenuation at wn >= 20 rad/s: ", magN[0])

    print(" ")
    omega_r = 0.2
    magR, _, _ = bode(Plant*C, dB=dB_flag, omega=omega_r, plot=False)
    print("Reference Tracking at wr <= 0.2 rad/s: ", 1/magR[0])

    # # Closed-Loop Controller
    # mag, phase, omega = bode((Plant*C / (1 + Plant*C)),
    #                            dB=dB_flag,
    #                            omega=np.logspace(-4, 5),
    #                            label=r'$\frac{P(s)C(s)}{1+P(s)C(s)}$ -
Closed-loop',
    #                            plot=True,
    #                            margins=True)

    fig = plt.gcf()
    fig.axes[0].legend()
    fig.axes[0].grid(True)
    fig.axes[1].grid(True)
    fig.axes[0].set_title('Bode Diagram')
    plt.savefig('part5.png')
    # plt.show()
```

```
############################################
# now check the closed-loop response with prefilter
############################################
# Closed loop transfer function from R to Y - no prefilter
CLOSED_R_to_Y = (Plant * C / (1.0 + Plant * C))
# Closed loop transfer function from R to Y - with prefilter
CLOSED_R_to_Y_with_F = (F * Plant * C / (1.0 + Plant * C))
# Closed loop transfer function from R to U - no prefilter
CLOSED_R_to_U = (C / (1.0 + Plant * C))
# Closed loop transfer function from R to U - with prefilter
CLOSED_R_to_U_with_F = (F*C / (1.0 + Plant * C))

plt.figure(4)
plt.clf()
plt.grid(True)
plt.subplot(311)
mag, phase, omega = bode(CLOSED_R_to_Y, dB=dB_flag, plot=False)
if dB_flag:
    plt.semilogx(omega, mag2db(mag), color=[0, 0, 1],
        label='closed-loop $\\frac{Y}{R}$ - no pre-filter')
else:
    plt.loglog(omega, mag, color=[0, 0, 1],
        label='closed-loop $\\frac{Y}{R}$ - no pre-filter')
mag, phase, omega = bode(CLOSED_R_to_Y_with_F,
                         dB=dB_flag, plot=False)
if dB_flag:
    plt.semilogx(omega, mag2db(mag), color=[0, 1, 0],
        label='closed-loop $\\frac{Y}{R}$ - with pre-filter')
else:
    plt.loglog(omega, mag, color=[0, 1, 0],
        label='closed-loop $\\frac{Y}{R}$ - with pre-filter')
plt.ylabel('Closed-Loop Bode Plot')
plt.grid(True)
plt.legend()

plt.subplot(312), plt.grid(True)
T = np.linspace(0, 2, 100)
_, yout_no_F = step_response(CLOSED_R_to_Y, T)
_, yout_F = step_response(CLOSED_R_to_Y_with_F, T)
plt.plot(T, yout_no_F, color=[0,0,1],
         label='response without prefilter')
plt.plot(T, yout_F, color=[0,1,0],
         label='response with prefilter')
plt.legend()
plt.ylabel('Step Response')


plt.subplot(313)
plt.grid(True)
```

```
    _, Uout = step_response(CLOSED_R_to_U, T)
    _, Uout_F = step_response(CLOSED_R_to_U_with_F, T)
    plt.plot(T, Uout, color=[0, 0, 1],
             label='control effort without prefilter')
    plt.plot(T, Uout_F, color=[0, 1, 0],
             label='control effort with prefilter')
    plt.ylabel('Control Effort')
    plt.legend()

    plt.show()
```