```python
# VTOL Parameter File
import sys
sys.path.append('..')  # add parent directory
import VTOLParam as P
sys.path.append('../hw10')  # add parent directory
sys.path.append('../hw15')  # add parent directory
import VTOLParamHW10 as P10
import VTOLParamHW15 as P15
from control import tf, bode
import matplotlib.pyplot as plt

# flag to define if using dB or absolute scale for M(omega)
dB_flag = False

# Assign transfer functions from previous HW solution
P_lon = P15.P_lon
P_lat_in = P15.P_lat_in
P_lat_out = P15.P_lat_out

# Compute the controller transfer functions from HW10

# PID xfer function for longitudinal control
C_lon = tf([(P10.kd_h+P10.kp_h*P.sigma), (P10.kp_h+P10.ki_h*P.sigma), P10.ki_h],
           [P.sigma, 1, 0])

# PD control xfer function for inner loop of lateral control
C_lat_in = tf([(P10.kd_th+P.sigma*P10.kp_th), P10.kp_th], [P.sigma, 1])

# based on performance from HW 10, we get best performance with
# no integrator term, therefore, this is just a PD controller.
# PD xfer function for outer loop of lateral control
C_lat_out = tf([(P10.kd_z+P.sigma*P10.kp_z), P10.kp_z], [P.sigma, 1])

# display bode plots of transfer functions
fig1 = plt.figure()
bode([P_lon, P_lon*C_lon], dB=dB_flag)
plt.legend(['$P_{lon}(s)$', '$C_{lon}(s)P_{lon}(s)$'])
fig1.axes[0].set_title('VTOL Longitudinal Loop')

# display bode plots of transfer functions
fig2 = plt.figure()
bode([P_lat_in, P_lat_in*C_lat_in], dB=dB_flag)
plt.legend(['$P_{lat,in}(s)$', '$C_{lat,in}(s)P_{lat,in}(s)$'])
fig2.axes[0].set_title('VTOL Lateral Inner Loop')

fig3 = plt.figure()
bode([P_lat_out, P_lat_out*C_lat_out, tf([1.0], [1.0, 0.0])], dB=dB_flag)
plt.legend(['$P_{lat,out}(s)$', '$C_{lat,out}(s)P_{lat,out}(s)$', '$\\frac{1}{s}$'])
fig3.axes[0].set_title('VTOL Lateral Outer Loop')

##########################
```

```
52 # for parts a)-b)
53 ##########################
54
55 # For part a), the transfer function C*P is shown in figure, this is a
56 # shortcut to rendering latex from python and could be done in other
   ways.
57 # We can use this result to find e_ss to any input.
58 plt.figure()
59 xfer_func = (P_lon*C_lon)._repr_latex_()
60 plt.text(0.1, 0.5,'$%s$'%xfer_func[2:-2], fontsize='xx-large')
61 plt.tick_params(axis='both', which='both', bottom=False, top=False,
62                 left=False, right=False, labelbottom=False, labelleft=
   False)
63 plt.title("xfer function for $C_{lon}(s)P_{lon}(s)$")
64
65 omegas = [30.0]  # omega_no
66 mag_CP_lon, phase, omegas = bode(P_lon*C_lon, plot=False, dB=False,
   omega = omegas)
67
68 print("\n\n\nvalue for metric calculation (part b):")
69 # part b)
70 print("gamma_n = ", mag_CP_lon[0])
71
72
73 ##########################
74 # for parts c)-d)
75 ##########################
76 omegas = [2.0]  # omega_d_in
77 mag_CP_lat_in, phase, omegas = bode(P_lat_in*C_lat_in, plot=False, dB=
   False, omega = omegas)
78 mag_P_lat_in, phase, omegas = bode(P_lat_in, plot=False, dB=False, omega
    = omegas)
79
80
81 print("\n\n\nvalue for metric calculation (part c):")
82 # part c)
83 print("gamma_d_in = ", mag_P_lat_in[0]/mag_CP_lat_in[0])
84
85 # part d)
86 print('for part d), look at plot directly for C_{lat,in}P_{lat,in}')
87
88
89 ##########################
90 # for parts e)-f)
91 ##########################
92 omegas = [0.01, 0.1]  # omega_d_out, omega_r
93 mag_CP_lat_out, phase, omegas = bode(P_lat_out*C_lat_out, plot=False, dB
   =False, omega = omegas)
94
95 print("\n\n\nvalue for metric calculation (parts e and f):")
96 # part e)
97 print("gamma_r = ", 1.0/mag_CP_lat_out[1])
98
```

```
 99 # part f)
100 print("gamma_d_out =", 1.0/mag_CP_lat_out[0])
101
102
103
104 print('\n\nClose window(s) to end program')
105 plt.show()
106
107 # # Closes plot windows when the user presses a button.
108 # plt.pause(0.0001)  # not sure why this is needed for both figures to
    display
109 # print('Press key to close')
110 # plt.waitforbuttonpress()
111 # plt.close()
112
```