Assignment 3 - PID Controllers

March 6, 2024

In this assignment we will design a PID controller to increase the responsiveness of a system and reject disturbances. The main objective in this assignment is to gain experience tuning a PID controller since this is a common controller that you may encounter in industry. The system we will consider throughout this assignment has transfer function $P(s) = \frac{s+3}{s^4+8s^3+27s^2+50s+50}$. We will tune the controller by studying the closed-loop systems' step response. The procedure outlined in this assignment is very roughly the Ziegler-Nichols Method of PID tuning. The advantage of this method is that you can perform the method directly on a real system. There is no need for a model of the system.

Use the following code snippet to set up the simulation. Note that we have defined a reference signal, disturbance and measurement noise to be used throughout the assignment.

```
import numpy as np
import control as ct
import matplotlib.pyplot as plt
import scipy as sp

P = ct.tf([1, 3], [1, 8, 27, 50, 50])

N = 5000
duration = 10
t = np.linspace(0,duration,N)
n = 0.05*np.random.randn(N)

b1, a1 = sp.signal.butter(5, 0.1, 'low')
b2, a2 = sp.signal.butter(5, 0.01, 'low')
d = 10*sp.signal.lfilter(b2, a2, np.random.randn(N))
d += sp.signal.lfilter(b1, a1, np.random.randn(N))
d += np.linspace(0,5,N)

r = 5*np.ones(N)
r[0:100] = np.zeros(100)

fig, ax = plt.subplots(3,1)
ax[0].plot(t,r)
```

```
ax.set_title('Reference Signal')
ax.set_xlabel('Time (s)')
ax[1].plot(t,d)
ax.set_title('Disturbance')
ax.set_xlabel('Time (s)')
ax[2].plot(t,n)
ax.set_title('Measurement Noise')
ax.set_xlabel('Time (s)')
```

1. Select a value for K_p such that the closed-loop system is stable, but nearly unstable. Use the following code snippet to try different values of K_p. Start with a small value, and gradually increase it until the closed-loop system seems like it is almost unstable. In your report, note the value of K_p that you found. Include a plot of the Plant output in the closed-loop system in your report.

    ```
    Kp = # enter your value here
    K = ct.tf(Kp,1)
    T = K*P/(1+K*P)
    S = 1/(1+K*P)

    t,y_r = ct.forced_response(T,t,r)
    t,y_n = ct.forced_response(S,t,n)
    t,y_d = ct.forced_response(S*P,t,d)

    y = y_r + y_n + y_d

    fig, ax = plt.subplots(1)
    ax.plot(t,y)
    ax.set_title('Plant Output')
    ax.set_xlabel('Time (s)')
    ```

2. Make a Root Locus plot of the plant. Mark the final closed-loop poles that you selected in Question 1 on the plot. Use the following code snippet:

    ```
    fig,ax = plt.subplots(1)
    cl_poles, K_vec = ct.root_locus(P)

    den_closed_loop = np.array([]) # enter the denominator of your
                                   # closed-loop system from Q1 here

    cl_poles_selected = np.roots(den_closed_loop)
    ax.plot(np.real(cl_poles_selected), np.imag(cl_poles_selected), 'rx')
    ```

3. Divide that value of K_p that you found by 2. Use this new value as the value for K_p for the remainder of this assignment. Plot the closed-loop poles corresponding to this value of K_p on the Root-Locus plot as well.

4. Add derivative action to your controller. Select a value for K_d such that the step response has a 2% settling time that is roughly 3 seconds (doesn't need to be exact, just close). Start with small values of K_d and slowly increase it until you satisfy the requirement. Record the value of K_d you select and a plot of the Plant output of the closed-loop system in your report. Use the following code snippet:

```
Kd = # enter your value for Kd here
K = ct.tf(Kp,1) + ct.tf([Kd, 0], [0.01, 1])

T = K*P/(1+K*P)
S = 1/(1+K*P)

t,y_r = ct.forced_response(T,t,r)
t,y_n = ct.forced_response(S,t,n)
t,y_d = ct.forced_response(S*P,t,d)

y = y_r + y_n + y_d

fig, ax = plt.subplots(1)
ax.plot(t,y)
ax.set_title('Plant Output')
ax.set_xlabel('Time (s)')
```

5. What is the steady state error of the closed-loop system you have designed when the input is a step with magnitude 5? Use exact calculations and show your work.

6. Now we will add integral action to eliminate the steady state error when the input is a step with magnitude 5. Select a value for K_i such that the step response reaches 5 within 4 seconds. Start with small values of K_i and slowly increase it until you satisfy the requirement. Record the value of K_i you select and a plot of the Plant output of the closed-loop system in your report. Use the following code snippet:

```
Ki = # enter your value here
K = ct.tf(Kp,1) + ct.tf([Kd, 0], [0.01, 1]) + ct.tf([Ki], [1, 0])

T = K*P/(1+K*P)
S = 1/(1+K*P)

t,y_r = ct.forced_response(T,t,r)
t,y_n = ct.forced_response(S,t,n)
t,y_d = ct.forced_response(S*P,t,d)
```

```
y = y_r + y_n + y_d

fig, ax = plt.subplots(1)
ax.plot(t,y)
ax.set_title('Plant Output')
ax.set_xlabel('Time (s)')
```

7. How effectively is the controller rejecting the disturbance signal?

8. Make a Bode plot of the sensitivity, complementary sensitivity and loop transfer functions. Use the following code snippet:

```
S = 1/(1+K*P)
T = K*P/(1+K*P)
L = K*P
mag, phase, w = ct.bode_plot([S,T,L])
```

9. Simulate the closed-loop system with a different reference. Based on the Bode plots of the previous question, does the response to this input make sense? Include the plot of the desired vs. actual output in your report.

```
n = 0.05*np.random.randn(N)
d = 10*sp.signal.lfilter(b2, a2, np.random.randn(N))
d += sp.signal.lfilter(b1, a1, np.random.randn(N))
d += 5*np.linspace(0,5,N)

b3, a3 = sp.signal.butter(5, 0.001, 'low')
r = 10*sp.signal.lfilter(b3, a3, np.random.randn(N))

fig, ax = plt.subplots(3,1)
ax[0].plot(t,r)
ax[1].plot(t,d)
ax[2].plot(t,n)

t,y_r = ct.forced_response(T,t,r)
t,y_n = ct.forced_response(S,t,n)
t,y_d = ct.forced_response(S*P,t,d)

y = y_r + y_n + y_d

fig, ax = plt.subplots(1)
ax.plot(t,y, label='actual output')
ax.plot(t,r, label='desired output')
ax.set_title('Desired Output compared to Actual Output')
ax.set_xlabel('Time (s)')
```