ELE 503 Final Project

Table of Contents

Noah Johnson

(1) Derivation of SSM for Virtual Plant

See attached page for derivation

(2) Calculation of feedback gain vector K

```
Ts = 3; % Given
run models; % Get numerical vals for system constants
load sroots;

plantPoles = eig(A); % find plant poles

adp = s1/Ts + imag(plantPoles(2))*j; % calculate adp type poles

sPoles = [plantPoles(1) adp conj(adp) -plantPoles(4) s1/Ts]; % one of
 the poles is unstable and must be reflected.

K = place(A,B,sPoles); % Find K using Place

[d1,d2] = rb_regsf(A,B,K,0) % find stability margins
```

```
d1 =

    0.6101


d2 =

     1
```

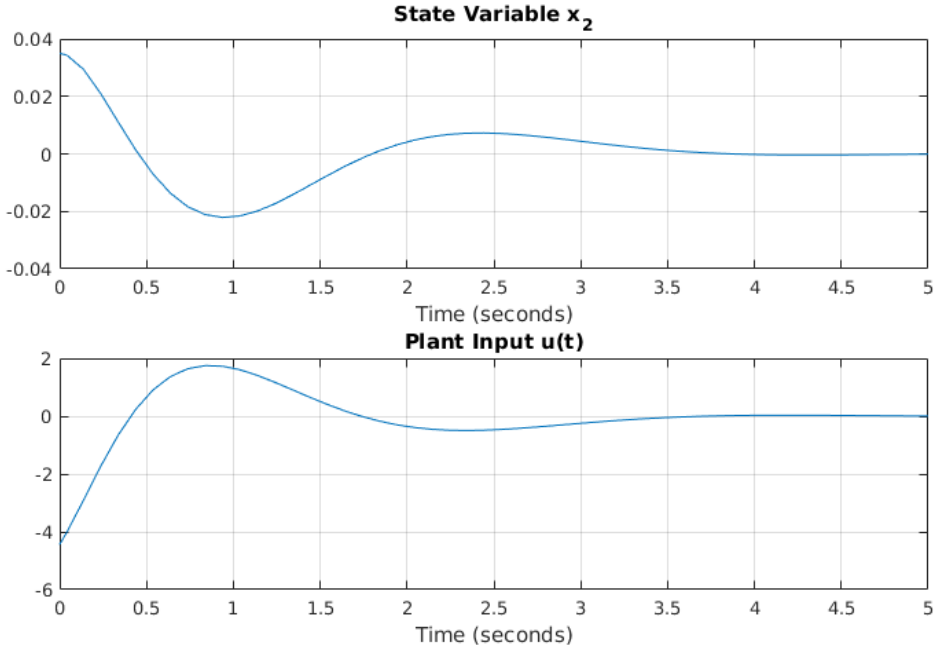
(3) Simulation for x2(0) = 2 degrees

The settling time of $x_2(t)$ is very close to that in figure 8 on pg. 30 of [1], as is the case for the plant input, $u(t)$. The graph of $u(t)$ we found is actually nearly identical to that in figure 8. However, the magnitude of the graph of $x_2(t)$ seems to be wrong, though the shape is correct. I was unable to find the cause of this error.

```
x20 = 2*pi/180;
yref = 0;

% run simulation
results = sim('Rsim1','StopTime', '5');
tout = results.tout;
u = results.u;
x = results.x;

% run reg_asfp
open('figs/sim1.fig')
```

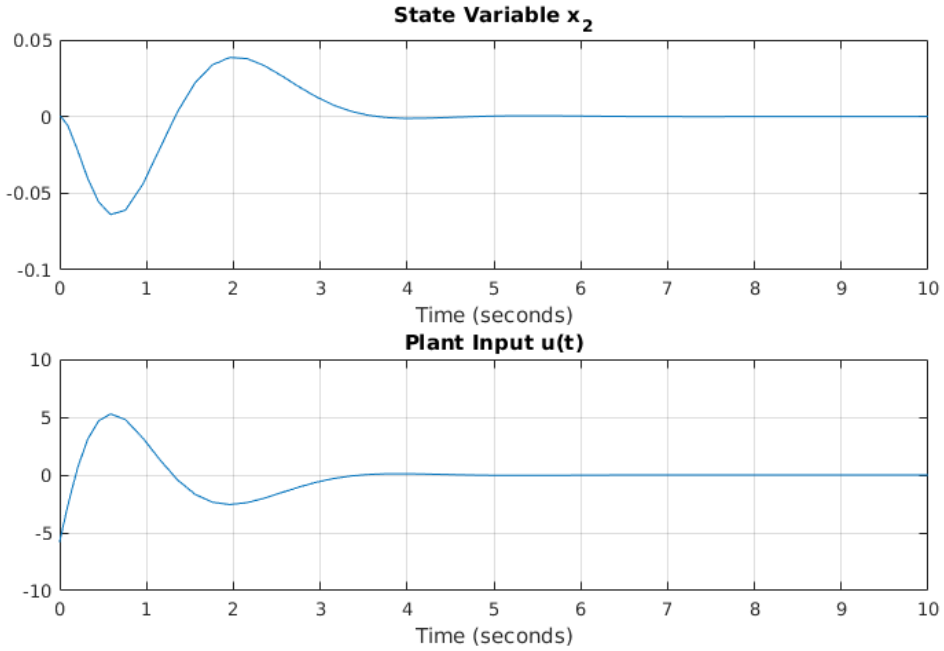


(4) Steady State Simulation

```
x20 = 0;
yref = 0.12626; % required yref value to get steady state value of 0.4

% run simulation
results = sim('Rsim1','StopTime', '10');
tout = results.tout;
u = results.u;
x = results.x;
```

```
% run reg_asfp
open('figs/sim2.fig')
```



(5a) Steady State value for x1, (yref = 1)

```
yss = inv(A-B*K)*-B*K*[1;0;0;0;0] % see attached scratch notes for
 derivation.

x1_ss = yss(1) % derived steady state displacement value for yref = 1.
```

```
yss =

    3.1678
   -0.0000
         0
         0
    3.1678


x1_ss =

    3.1678
```

(5b) Steady State value for yref (x1 = 0.4)

```
yss = inv(A-B*K)*-B*K*[0.12626;0;0;0;0] %yref for x1 = 0.4 is 0.0782.
```

```
x1_ss = yss(1) % derived steady state displacement value for yref = 1.
```

```
yss =

    0.4000
   -0.0000
         0
         0
    0.4000


x1_ss =

    0.4000
```

(6) Control System Stability Robustness Bounds

The stability robustness bounds for the entire control system are: $\delta_1 = 0.3661$ and $\delta_2 = 0.5555$. These differ from the previous bounds $(\delta_1 = 0.6101, \delta_2 = 1)$ because the first considered the stability of the virtual plant only, and ignored to outer feedback loop. This second set of bounds is the correct one, as it considered the whole system. The (A,B,C,D) values that should be used are A_sys, B_sys, C_sys, and D_sys, respectively.

```
A_star = ([(Abar - Bbar*Kpi) (Bbar*ki)] - Bbar*kp*K);
C_star = (-kp*K + [-Kpi ki]);

% Delta 1
A_sys = [A_star; -K];
B_sys = [Bbar; 0];
C_sys = C_star;
D_sys = 0;

sys = ss(A_sys,B_sys,C_sys, D_sys);

delta1 = inv(norm(sys,inf))


% Delta 2
A_sys = [A_star; -K];
B_sys = [Bbar; 0];
C_sys = C_star;
D_sys = 1;

sys = ss(A_sys,B_sys,C_sys, D_sys);

delta2 = inv(norm(sys,inf))
```

```
delta1 =
```

```
    0.3661

delta2 =

    0.5555
```

(7) Tracking System Block Diagram

See attached page for block diagram.

(8) Tracking System Design

The tracking system gain values (K1 and K2) must be calculated using >>tsd(A,B,C,Aa,Ba,poles,T,place). The A,B,C values are those given for the virtual plant. The Aa and Ba values are defined in (7) to both be 1. The selection of the tracking system poles would be as follows:

- We would select 5 poles based on the plant poles, exactly like what we did in (2)
- We would then chose 1 pole for the tracking system, as the tracking system is first order. We chose to use >>tzero() to find the zeros of the virtual plant, and find that there is one zero with a negative real part that is less than s1/Ts, at -4.9664. This will be chosen as the last pole.

After this we would run >>tsd() with the values we just found, taking care to use the place algorithm, as the rfbg algorithm will give "good" feedback gain based on the stability roubustness bounds. These are wrong as we will see in (9).

(9) tsd Stability Robustness Bounds

The stability robustness bounds returned by the tsd command would not be usable, because the algorithm inserts the perturbation at the wrong place (in front of the virtual plant), causing these values to be incorrect. This is because we are using tsd to track the virtual plant and not the hardware plant. The gain values are still usable, but the delta bounds are incorrect.

(10) Method for finding stability robustness bounds

The tracking system stability robustness bounds can be found similarly to how they were calculated in (6). We would insert a perturbation right before the hardware plant, set r(t) = 0, and find the state space model from w to v. we could then find delta 1 by taking the reciprocal of the system infinity norm. Delta 2 would be found similarly.

Published with MATLAB® R2017a