MECH467 Lab 2 Pre-lab
Ratthamnoon Prakitpong
63205165

1.
We can derive it by hand:

Neglect T_d because ignore coulomb friction

$$\underbrace{X_d(s) \cdot \left(\frac{K_p}{s}\right)\left(\frac{1}{(J_e s + B_e)}\right)(K_t K_a)}_{G(s)} \quad V(s) = \frac{K_e K_T K_a}{J_e s^2 + B_e s}$$

$$G_d(z) = ZOH|G(s)| = (1-z^{-1})\, Z\left\{\frac{G(s)}{s}\right\} = (1-z^{-1})\, Z\left\{\frac{(K_e K_t K_a)}{s^2(J_e s + B_e)}\right\}$$

$$= (1-z^{-1})(K_e K_t K_a)\, Z\left\{\frac{A}{s(J_e s + B_e)} + \frac{B}{s^2}\right\}$$

where $A = -J_e/B_e$
$B = 1/B_e$

$$= (1-z^{-1}) K_e K_t K_a\, Z\left\{\left(\frac{-1}{B_e}\right)\left(\frac{J_e}{B_e}\right)\frac{(B_e/J_e)}{s\left(s+\frac{B_e}{J_e}\right)} + \left(\frac{1}{B_e}\right)\frac{1}{s^2}\right\}$$

$$= (1-z^{-1}) K_e K_t K_a \left(\left(\frac{-J_e}{B_e^2}\right)\frac{z^{-1}(1-e^{-B_eT/J_e})}{(1-z^{-1})(1-e^{-B_eT/J_e}z^{-1})} + \left(\frac{1}{B_e}\right)\frac{Tq^{-1}}{(1-z^{-1})^2}\right)$$

Use MATLAB symplify function

From MATLAB simplify function:
G(z) = (887*(13845538983972269071115320 79*z +
1383747136427847856170981841))/(2348798729487646720*(9007199254740992*z -
8991463285215455)*(z – 1))
Divide everything by 2348798729487646720*9007199254740992 to get:
G(z) = ((0.5805e-4)z + 0.5802e-4) / (z^2-1.9983z+0.9983)

From c2d:

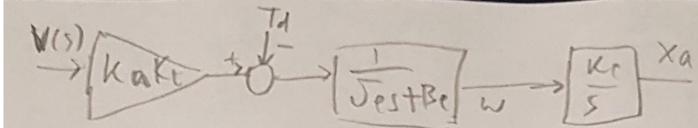
```
>> q1
   1.0e-04 *

         0    0.5805    0.5802

    1.0000   -1.9983    0.9983
```

As we can see, the derived equation and output from c2d are the same.

2.



$$\frac{X_a}{w} = \frac{K_e}{s} \rightarrow sX_a = K_e w$$

$$\frac{dX_a(t)}{dt} = K_e w(t)$$

$$\frac{w}{K_aK_tV - T_d} = \frac{1}{J_es + B_e} \rightarrow sw = \frac{-B_e}{J_e}w + \frac{K_aK_t}{J_e}V - \frac{1}{J_e}T_d$$

$$\frac{dw(t)}{dt} = \frac{-B_e}{J_e}w(t) + \frac{K_aK_t}{J_e}V(t) + \frac{-1}{J_e}T_d(t)$$

where $[x] = \begin{bmatrix} w(t) \\ X_a(t) \end{bmatrix}$ & $[u] = \begin{bmatrix} V(t) \\ T_d(t) \end{bmatrix}$

$$[\dot{x}] = \underbrace{\begin{bmatrix} \frac{-B_e}{J_e} & 0 \\ K_e & 0 \end{bmatrix}}_{[A]}[x] + \underbrace{\begin{bmatrix} \frac{K_aK_t}{J_e} & \frac{-1}{J_e} \\ 0 & 0 \end{bmatrix}}_{[B]}[u]$$

$$\downarrow$$

$$[x(k+1)] = [\Phi(T)][x(k)] + [H(T)][u(k)]$$

$$[\Phi(T)] = e^{[A]T} \approx [I] + [A]T = \begin{bmatrix} -TB_e/J_e + 1 & 0 \\ K_eT & 1 \end{bmatrix}$$

$$[H(T)] = \left[\int_0^T e^{[A]t}dt\right]\cdot[B] = \begin{bmatrix} \int_0^T(\frac{-B_e}{J_e}t + 1)\,dt & 0 \\ \int_0^T K_e t\,dt & \int_0^T dt \end{bmatrix}\cdot\begin{bmatrix} \frac{K_aK_t}{J_e} & \frac{-1}{J_e} \\ 0 & 0 \end{bmatrix}$$

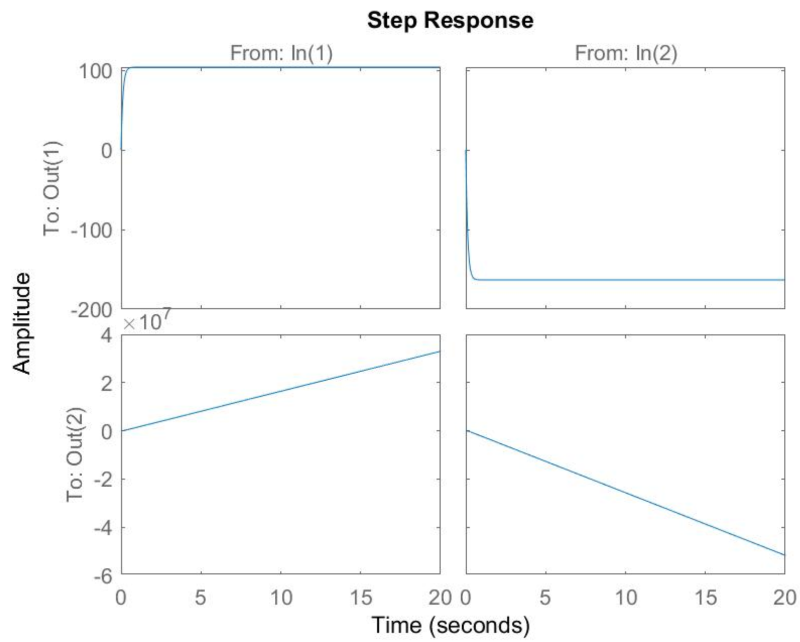
$$= \begin{bmatrix} T - \frac{B_e}{J_e}\frac{T^2}{2} & 0 \\ K_e\frac{T^2}{2} & T \end{bmatrix}\cdot\begin{bmatrix} \frac{K_aK_t}{J_e} & \frac{-1}{J_e} \\ 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{K_aK_t}{J_e}\left(T - \frac{B_eT^2}{J_e 2}\right) & \frac{-1}{J_e}\left(T - \frac{B_e}{J_e}\frac{T^2}{2}\right) \\ \frac{K_aK_t}{J_e}\left(K_e\frac{T^2}{2}\right) & \frac{-T}{J_e} \end{bmatrix}$$

and since we want X_a & w as output, we can just say

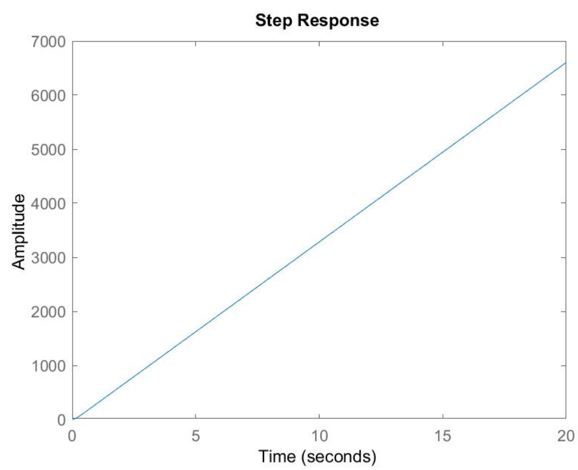
$$[y] = \begin{bmatrix} w(k) \\ X_a(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}[x] + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}[u]$$

We get this plot when simulated in MATLAB:



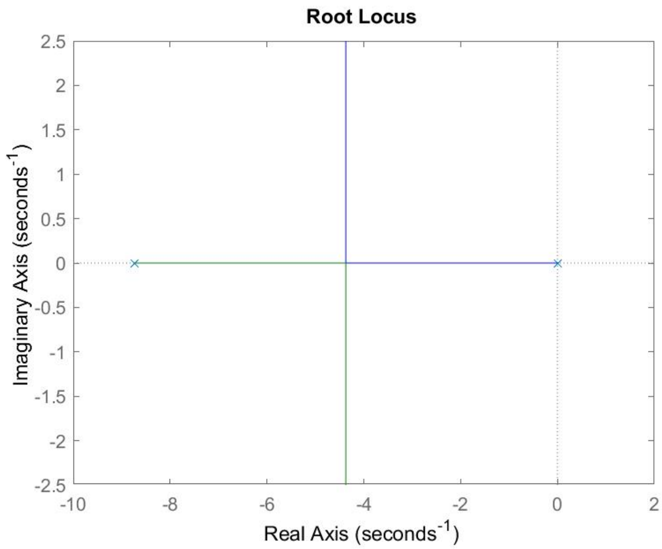
The plot to note is bottom-left, which is Xa as a result of V.

This is the resulting plot from just the z-domain transfer function:



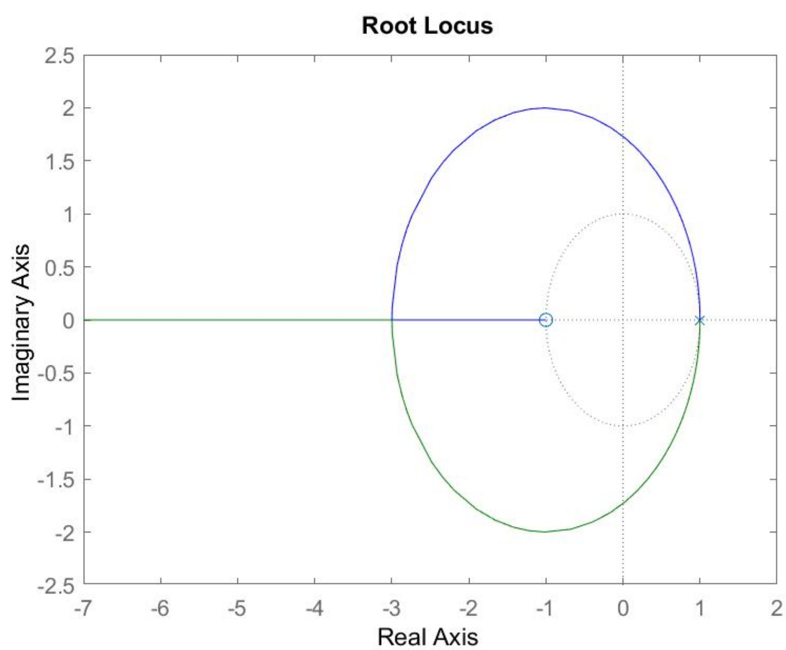
The two plots have the same trend, although the scaling are different. This may result from different derivation methods.

3a.
For continuous system:

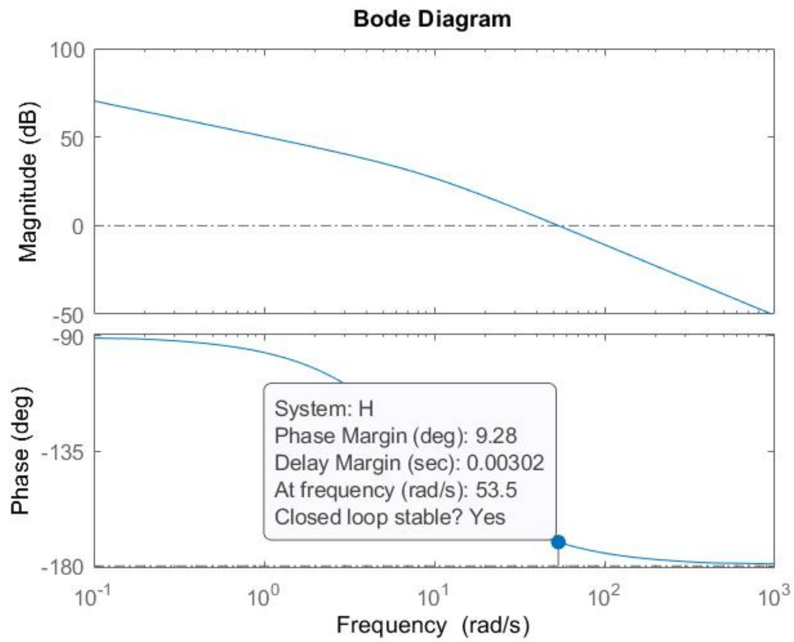


We can write poles of continuous function as a piece-wise function.
For 0 <= K <= 0.00658:
Pole 1 = (-4.37 / 0.00658) * K
Pole 2 = -8.74 + ((8.74 – 4.37) / 0.00658) * K
For K > 0.00658:
Pole 1 = -4.37 + i * (1.24 / (0.00711 – 0.00658)) * (K – 0.00658)
Pole 2 = -4.37 – i * (1.24 / (0.00711 – 0.00658)) * (K – 0.00658)

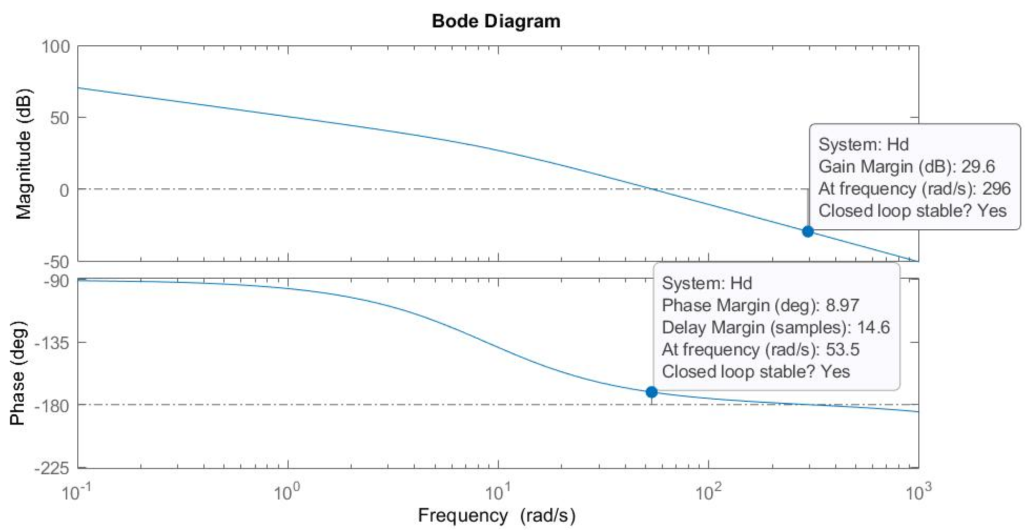
For discrete system:



3b.
For continuous system:

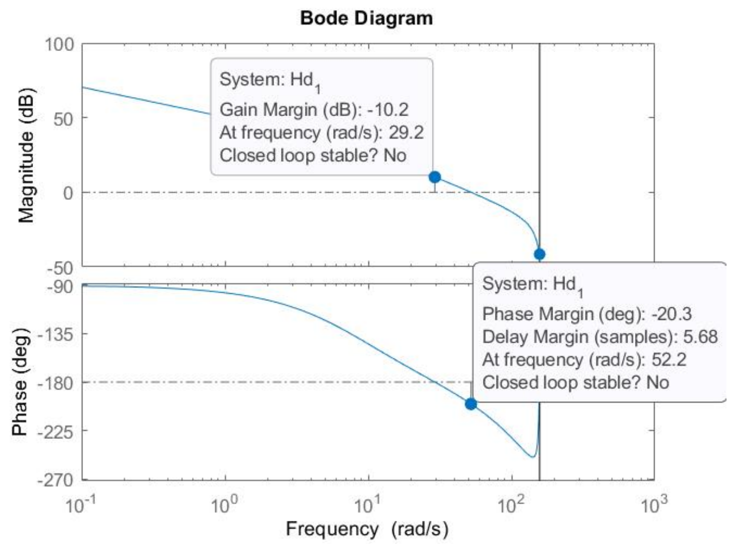


For discrete system where T = 0.0002 s:

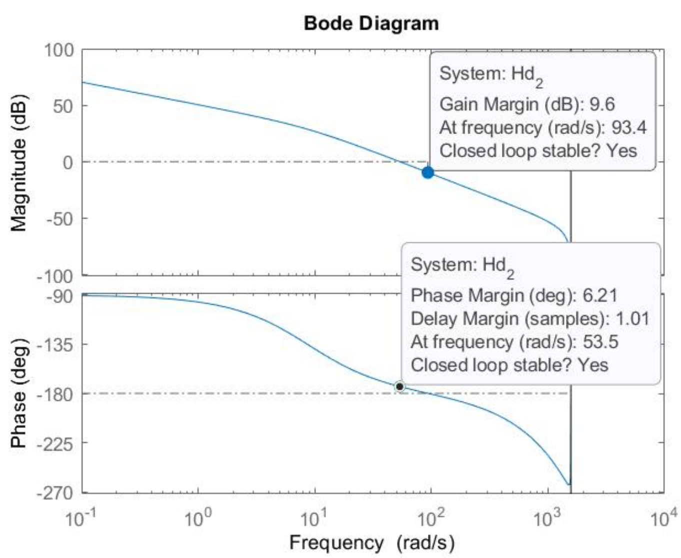


Gain and phase margin are shown in the plot. Continuous system's phase never reaches -180 deg, so there's no unstable gain and no gain margin. Since the phase margin is small (~10 deg), we can say that the system is unstable. There's a difference in phase margin when converting from continuous time to discrete time, which is likely introduced when the system was discretized.

3c.
For discrete system where T = 0.02 s:

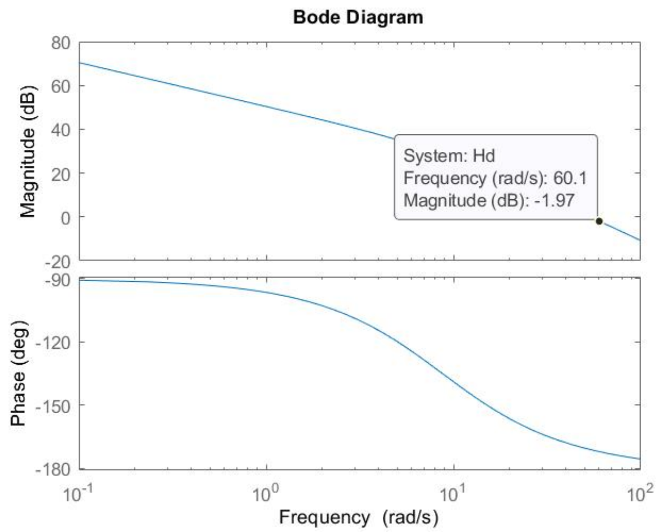


For discrete system where T = 0.002 s:



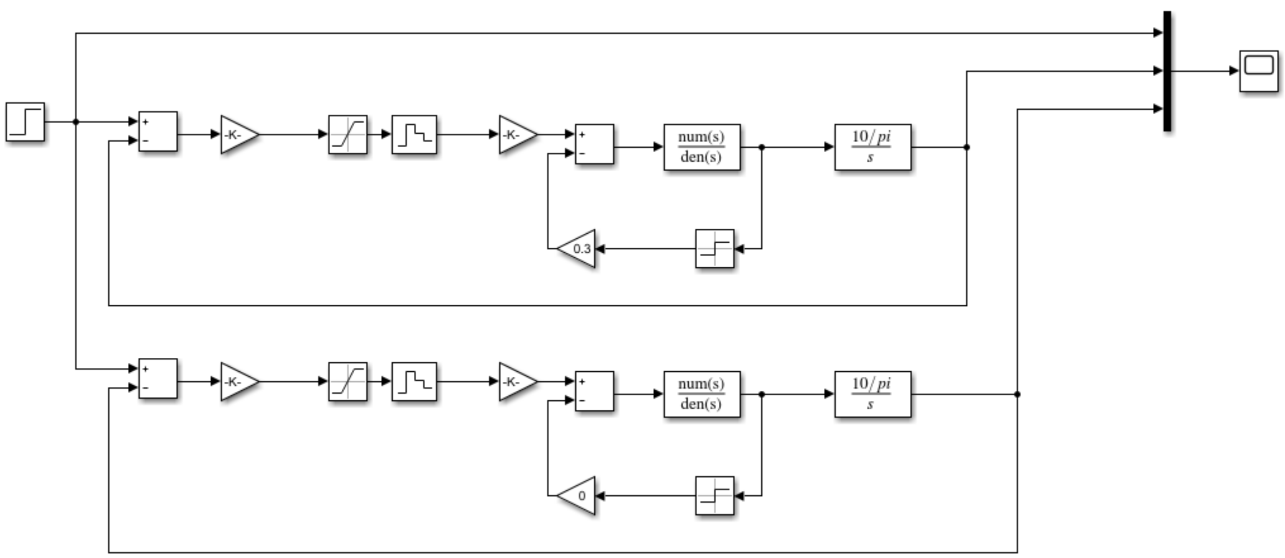
Stability in continuous and discrete time are not always equivalent, since you lose some information when converting function from continuous to discrete domain. The smaller sampling period, the closer discrete model is to the continuous model. Gain and phase margins are largest when T = 0.02 s, we can that it is most stable.

4.
As we can see, gain is -1.97dB at around 60 rad/s.

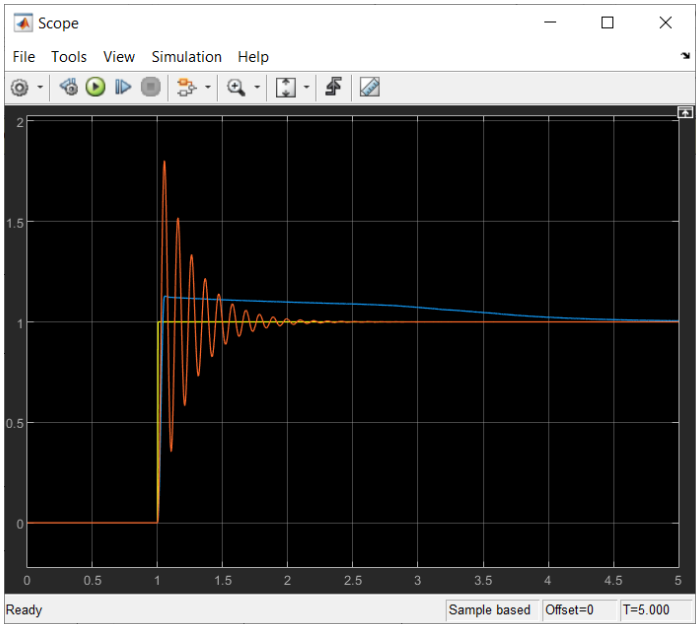


Therefore, we need to find K such that 20log(K) = 1.97dB to shift the magnitude plot up so it is 0dB at 60 rad/s. After solving, K = 1.2546.

Putting this into Simulink, we get this.

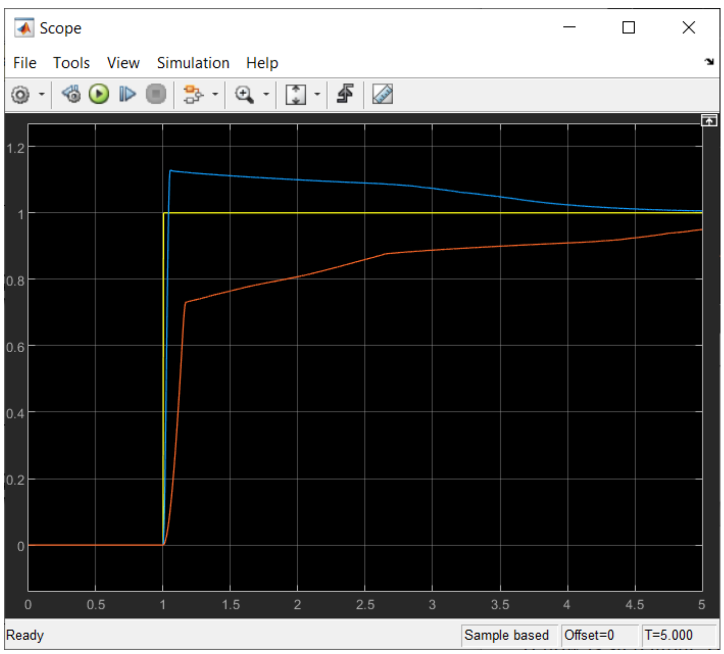


Simulating cases where uk are different:



Yellow is step input. Blue is the system where uk = 0.3. Red is the system where uk = 0. Since Coulomb friction goes against input torque and reduces amplitude of response, it makes sense that overshoot is less where friction is present. Similarly, it causes rise time to be delayed by a bit and causes settling time to delay; without friction, the system could settle much faster.

Simulating cases were saturation limits are different:



Yellow is step input. Blue is the system where saturation is +/-3. Red is the system where saturation is +/- 0.5. Because high input values are eliminated, the system doesn't have inputs that are high enough to produce high outputs (ie outputs that is close to steady-state value), and it takes more feedback cycles for the system to reach settling point. Therefore, overshoot is decreased, and rise time is increased, and settling time is increased.

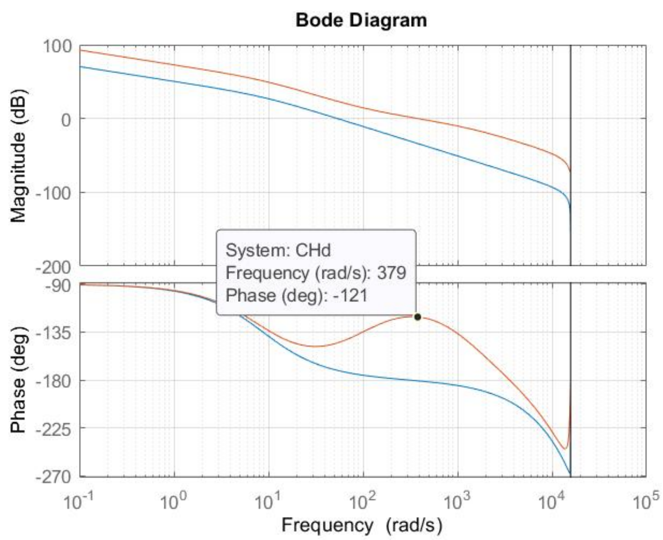
5a.

Lead-lag controller is a function of K, a, and T, where a and T can be solved by these equations.

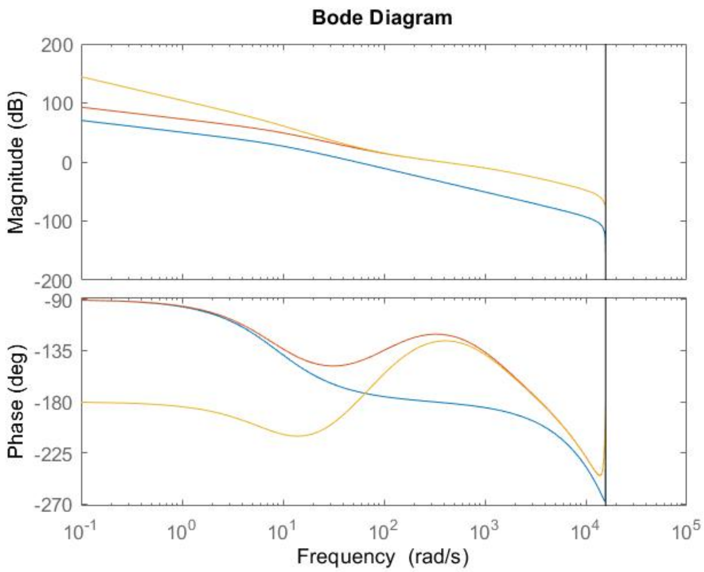
$$C(s) = K\frac{1+\alpha Ts}{1+Ts}$$

$$\alpha = \frac{1+\sin(\phi_m)}{1-\sin(\phi_m)}$$
$$T = \frac{1}{\sqrt{\alpha}\omega_m}$$

Since phi_m and w_m are given as 60 deg and 377 rad/s, a and T can simply be solved. To get K, I've plotted C(s)H(s) on bode plot, found gain margin at 377 rad/s, then solve for K to shift gain at that frequency back to 0dB. Following those steps, I've found that I need to shift gain by +22.4dB, so K = 10^(22.4/20) = 13.18. Plotting everything on the bode plot, we can see that we've created 60 deg margin at 377 rad/s, and we've shifted magnitude such that the system is stable.



5b.
By applying our integrator, we get this bode plot (yellow is the plot with integrator cascaded).



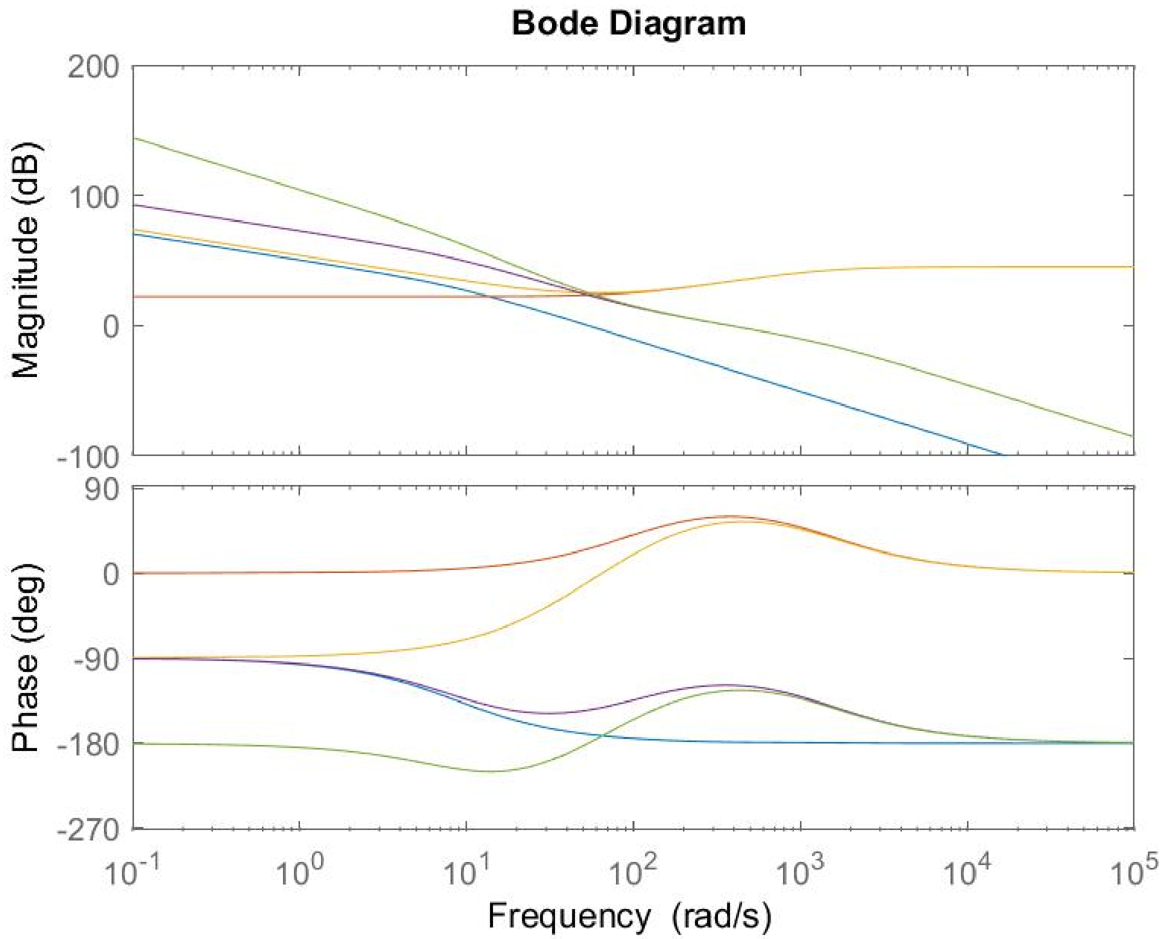
As we can see, the phase is not affected where the crossover frequency. However, now the steady-state error is eliminated, because we now have infinite gain at zero frequency (see equation of steady state error for clarification).

$$E(s) = \left(1 - \frac{G_o(s)}{1 + G_o(s)}\right) U(s)$$

$$E(s) = \frac{U(s)}{1 + G_o(s)}$$

$$e(\infty) = \lim_{s \to 0} sE(s)$$

6.



The colours correspond to plots:

- Blue: G_ol(s)
- Red: LL(s)
- Yellow: LLI(s)
- Purple: G(s)*LL(s)
- Green: G(s)*LLI(s)

The plot is practically the same as its discrete version. Lead-lag compensator, with K calibrated properly, shifts magnitude to 0dB, and increases phase, resulting in greater stability at crossover frequency. Integrator increases gain as frequency approaches zero, so to reduce steady state error to zero; integrator changes phase but not at crossover frequency.

Since steady state error is inversely proportional to DC gain and steady state error is error where frequency approaches zero, it's best to increase DC gain to minimize steady state error. Keeping gain at crossover frequency at 1 reduces changes to rise time, since there's no gain and therefore no changes to how the system behaves when transformed.

Appendix:

Q1 code:

```
Ke = 10/3.1415;
Ka = .887;
Kt = .72;
Je = .0007;
Be = .00612;
T = .0002;

H = tf((Ke*Ka*Kt),[Je Be 0]);
Hd = c2d(H,T);
[num, dem] = tfdata(Hd, 'v');
disp(num);
disp(dem);

syms z;
f =
Ke*Ka*Kt*(1-1/z)*(((-Je/(Be*Be))*(1/z)*(1-exp(-Be*T/Je))/((1-1/
z)*(1-exp(-Be*T/Je)/z)))+((1/Be)*T*(1/z)/((1-1/z)*(1-1/z))));
s = simplify(f);
disp(s);
```

Q2 code:

```
Ke = 10/3.1415;
Ka = .887;
Kt = .72;
Je = .0007;
Be = .00612;
T = .0002;

H = tf((Ke*Ka*Kt),[Je Be 0]);
Hd = c2d(H,T);
step(Hd);
saveas(gcf, 'q2-function.jpg');

A = [-T*Be/Je+1, 0; Ke, 1];
B = [(Ka*Kt/Je)*(T-(Be/Je)*(T*T/2)),
(-1/Je)*(T-(Be/Je)*(T*T/2)); (Ka*Kt/Je)*(Ke*T*T/2), -T/Je];
C = [1, 0; 0, 1];
D = [0, 0; 0, 0];
sys = ss(A, B, C, D, T);
figure(2);
step(sys);
saveas(gcf, 'q2-ss.jpg');
```

Q3 code:

```
Ke = 10/3.1415;
Ka = .887;
Kt = .72;
Je = .0007;
Be = .00612;
T = .0002;

H = tf((Ke*Ka*Kt),[Je Be 0]);    % continuous time
Hd = c2d(H,T);                   % discrete time

% a
rlocus(H);
saveas(gcf, 'q3a-cont.jpg');
rlocus(Hd);
saveas(gcf, 'q3a-disc.jpg');

% b
bode(H);
bode(Hd);

%c
Hd_1 = c2d(H, 0.02);
Hd_2 = c2d(H, 0.002);
bode(Hd_1);
bode(Hd_2);
```

Q5 code:

```
Ke = 10/3.1415;
Ka = .887;
Kt = .72;
Je = .0007;
Be = .00612;
T = .0002;

w = 377;
phi = 60; % deg
phi = phi * pi/180;
a = (1+sin(phi))/(1-sin(phi));
t = 1/(sqrt(a)*w);
K = 13.18; % found that with K=1, mag at w is -22.4dB, so we
need to shift by +22.4dB
C = tf([K*a*t K], [t 1]);


Ki = w/10;
G = tf([1 Ki], [1 0]);
```

```
H = tf((Ke*Ka*Kt),[Je Be 0]);    % continuous time
Hd = c2d(H,T);                   % discrete time
hold on;
bode(Hd);

% a
CHd = c2d(C*H, T);
bode(CHd);

% b
GCHd = c2d(G*C*H, T);
bode(GCHd);
```

Q6 code:

```
Ke = 10/3.1415;
Ka = .887;
Kt = .72;
Je = .0007;
Be = .00612;
T = .0002;

w = 377;
phi = 60; % deg
phi = phi * pi/180;
a = (1+sin(phi))/(1-sin(phi));
t = 1/(sqrt(a)*w);
K = 13.18; % found that with K=1, mag at w is -22.4dB, so we
need to shift by +22.4dB
C = tf([K*a*t K], [t 1]);

Ki = w/10;
G = tf([1 Ki], [1 0]);

H = tf((Ke*Ka*Kt),[Je Be 0]);    % continuous time
CG = C*G;
CH = C*H;
CGH = C*G*H;

hold on;
bode(H);
bode(C);
bode(CG);
bode(CH);
bode(CGH);
saveas(gcf, 'q6.jpg');
```