ELECENG 3CL4 Lab 4 Report

Aaron Pinto
pintoa9
L02

Raeed Hassan
hassam41
L02

March 27, 2021

Member Contributions

Both group members contributed an even amount to both the exercises and the report. Both members went through the exercises together and contributed to all sections of the report.

Objective

To use the root locus technique to design a phase lead compensator for a marginally-stable servomotor.

2 Design of Phase-Lead Compensator

The procedure that was applied to design a phase lead compensator with desired design requirements was the same procedure followed in the pre-lab exercises.

The phase lead compensator was designed in a MATLAB script, with the code relevant to the design of the compensator found in Listing 1. The procedure is detailed in comments found in the MATLAB script. The parameters of the desired phase lead compensator were found to be $z = 8$, $p = 20.7149$, and $k_c = 1.8154$. The resulting phase lead compensator controller $G_c(s) = 1.8154\frac{s+8}{s+20.7149}$.

Listing 1: Design of phase lead compensator

```
%% EXP1.1; DESIGN PHASE LEAD COMPENSATOR
% AVERAGE TAU_M AND ALPHA VALUES FROM LAB 2
tau_m = (0.133 + 0.155)/2;
A = (25.877 + 30.303)/2;
% CREATE SYSTEM FROM VALUES
sysG = tf(1, [1 1/tau_m 0]);
polesG = pole(sysG); % DETERMINE POLES OF THE SYSTEM

% DESIGN REQUIREMENTS
po = 20;
settle_time = 0.5;

% CALCULATE REAL PART OF DESIRED ROOT LOCUS USING 2ND-ORDER SYS
    APPROXIMATION
sigma = 4 / settle_time;
% CALCULATE ZETA USING PERCENT OVERSHOOT
zeta = -log(po/100)/sqrt(pi^2 + log(po/100)^2);
phi = acos(zeta); % DETERMINE ANGLE FROM ZETA
omega = sigma * tan(phi); % CALCULATE IMAGINARY PART OF DESIRED
    ROOT LOCUS

poles_rlocus = [(-sigma + omega*1j) (-sigma - omega*1j)];
```

```
s0 = poles_rlocus(1); % PICK POSITIVE IMAGINARY ROOT FOR
   CALCULATIONS

% PLACE ZERO UNDER ROOT LOCUS
z = sigma;

% PHASE CONDITION
% DETERMINE PHASE CONTRIBUTIONS FROM POLES OF PLANT
phase0 = rad2deg(atan2(imag(s0) - imag(polesG(1)), real(s0) -
   real(polesG(1))));
phase1 = rad2deg(atan2(imag(s0) - imag(polesG(2)), real(s0) -
   real(polesG(2))));
angle_G = -phase0  - phase1;

% DETERMINE REQUIRED PHASE CONTRIBUTION FROM CONTROLLER;
   DETERMINE POSITION OF POLE
angle_Gc = 180 - angle_G - 360;
angle_Gc_pole = 90 - angle_Gc;
p = (sigma - omega*tan(angle_Gc_pole));

% PLACE ZEROS AND POLES FOR CONTROLLER AT -Z AND -P
Gc_zero = -z;
Gc_pole = -p;

% MAGNITUDE CONDITION
% DETERMINE MAGNITUDE CONTRIBUTIONS FROM POLES AND ZEROS
mag_poles = norm(s0 - polesG(1)) * norm(s0 - polesG(2)) * norm(
   s0 - Gc_pole);
mag_zeros = omega;

kG = A/tau_m; % PLANT GAIN

% DETERMINE CONTROLLER GAIN FROM MAGNITUDE CONDITION
kc = (1/kG) * (mag_poles/mag_zeros);
```

The velocity error constant is found using the MATLAB code in Listing 2. The velocity error constant was found to be 19.6943.

Listing 2: Velcoity error constant computation

```
%% EXP1.2; CALCULATE VELOCITY ERROR CONSTANT
kv = (kG * kc * z) / (p/tau_m);
```

The transfer function for the closed-loop system was generated in Listing 3. The transfer function $T_s = \frac{354.1s+2833}{s^3+27.66s^2+498s+2833}$. The poles of the transfer function were placed at $-9.6115 \pm 15.6025j$ and -8.4364, and the zero of the transfer function was placed at -8.

Listing 3: Transfer function computation

```
%% EXP1.3; COMPUTE CLOSED-LOOP TRANSFER FUNCTION T(s)
sysGc = tf([1 -Gc_zero],[1 -Gc_pole]); % SYSTEM FOR CONTROLLER
G_s = series(sysG, kG); % INTRODUCE PLANT GAIN
Gc_s = series(sysGc, kc); % INTRODUCE CONTROLLER GAIN
scaled_P_s = series(G_s, Gc_s); % GET SERIES OF PLANT AND
   CONTROLLER
T_s = feedback(scaled_P_s, 1); % GET TRANSFER FUNCTION
cl_poles = pole(T_s); % POLES OF TRANSFER FUNCTION
cl_zeros = zero(T_s); % ZEROS OF TRANSFER FUNCTION
```

The step response is plotted using the MATLAB code shown in Listing 4. The plot of the step response is shown in Figure 1a.

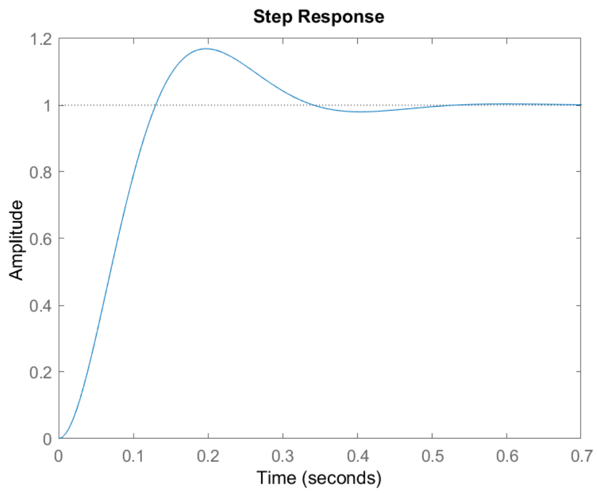
Listing 4: Step response

```
%% EXP1.4; PLOT STEP RESPONSE
fig_step = figure(1);
step(T_s);
```

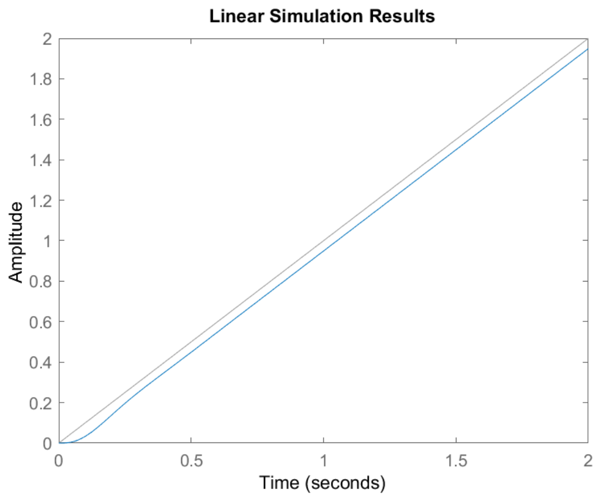
The unit ramp response is plotted using the MATLAB code shown in Listing 5. The plot of the unit ramp response is shown in Figure 1b.

Listing 5: Unit ramp response

```
% EXP1.5; PLOT UNIT RAMP RESPONSE
fig_unit_ramp = figure(2);
t = 0:0.001:2; % UNIT RAMP
lsim(T_s, t, t);
```



(a) Step Response



(b) Unit Ramp Response

Figure 1: Responses of Closed-Loop Transfer Function $T(s)$

We can see from the step response in Figure 1a that we are close to our desired design requirements. The percent overshoot is slightly less than 20% at around 17%, and the settling time is around the 0.5 sec requirement. The steady-state error for the unit ramp response in Figure 1b is around 0.05, which matches the calculated steady-state error value for the unit ramp response (1 / velocity error constant = 0.508).

As our design was done using approximations for a second-order system with no zeros, it is expected that there will be some discrepancies in the results as the system in use for the servomotor is a third-order system with one zero. However, we can see that the approximations still allow us to design a system that is close to meeting the design requirements. Such a system could be a good initial design that could later be adjusted to get closer to the design requirements.

3 Experiment with Phase Lead Compensator

The amplitude of the disturbance was set to 0. The numerator coefficients of the controller were k_c for s term and $k_c z$ for the constant term, and for the denominator they were 1 for the s term and p for the constant term. We entered them into the controller block in descending order of power.

The measured peak overshoot and 2% settling time were _ and _, respectively.

Potential sources of the discrepancies could be the inherent non-linear effects of the simulation model