MECH467 Lab 2

Ratthamnoon Prakitpong
#63205165

Abstract

Continuing from Lab 1, we learn to control a ball-screw linear actuator. To design a more accurate and stable control, we need to modify signal using controllers. Three controllers we'll use today are proportional, lead-lag compensator, and lead-lag compensator with integrator cascaded, with rise time, overshoot, and steady state error being main parameters analyzed.

Introduction

To customize drive systems to specific functions in each industries, each with their own performance and stability criterias, controllers need to be designed. Controllers control specific parameters of a system which result in certain criterias being satisfied (for example, increasing DC gain decreases steady state error and improves input-output accuracy). A popular software for designing and modelling controllers is Simulink. After obtaining quantitative data for comparison, you often analyze and compare it against theory to see if the numbers make sense or not.

This report's layout mirrors that process, where the first part is quantitative (comparison of experimental data and model), and second part is qualitative analysis.

Comparison of Experiments and Simulation

1.

	Experiment	Simulink	Experiment	Simulink	Experiment	Simulink
	LLI	LLI	LL	LL	P	P
Rise time	0.0096	0.0112	0.019	0.0196	0.0294	0.0266
Overshoot	0.2275	0.26	0	0	0.0827	0.1228

Table 1: Experimental and Simulink step results

Experimental and Simulink results are very similar and consistent. In terms of rise time, P controller takes longest, followed by LL controller, and followed by LLI controller. In terms of overshoot, LLI controller overshot by the most, followed by P controller, and followed by LL controller. Output of system with LL controller doesn't have overshoot because the system behaves like an overdamped system. In P and LLI controller, overshoot of experimental data are consistently smaller than Simulink's by ~0.04. The simplest explanation is due to underestimating friction in Simulink model, which we did when simplifying friction model. More friction in real life means that response would be more dampen, hence less overshoot.

Simulink models are lifted from prelab, with minor modifications. Those, along with scripts, are in Appendix for reviewing.

2.

	Experiment	Simulink	Experiment	Simulink	Experiment	Simulink
	LLI	LLI	LL	LL	P	P
SS Error	0.0003	0	0.0091	0.0359	0.1214	0.3768

Table 2: Experimental and Simulink ramp results

Experimental and Simulink results are very similar and consistent. P controller has the largested steady state error, followed by LL controller, followed by LLI controller. This makes sense since P controller is used to set unity gain, without attention to steady state error. In LL controller, gain magnitude is shifted up, which increases gain at zero frequency and decreases steady state error. In LLI controller, the cascaded integrator is picked such that gain at zero frequency is infinite, hence zero (or practically, very small) steady state error. Consistently, experimental results are approximately 1/3 of Simulink results. Since steady state error is proportional to U(s) and inversely proportional to G(s), we either have some hidden 1/3 gain in system, or our experimental steady state error is 3 times larger.

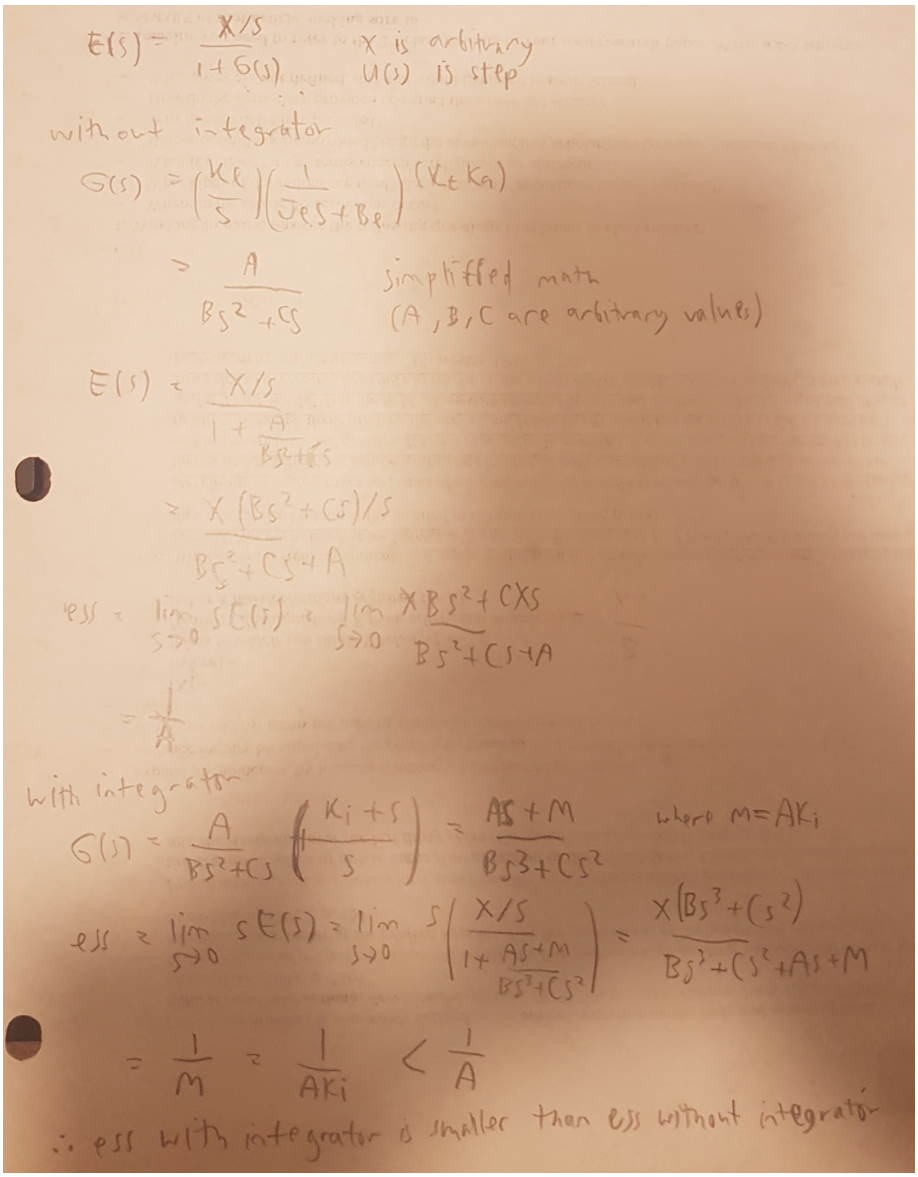
Simulink model and scripts are in Appendix.

Comparison of Controllers

3.
Both P controller's rise time of step response and steady state error of ramp response are larger than LL controller's. LL controller increasing bandwidth means less phase difference passover frequency, so rise time will decrease. LL controller's gain is also higher than P controller's, which means higher gain magnitude at zero frequency and lower steady state gain.

4.
It's not possible to design a lead-lag compensator with a very large bandwidth, because parts with faster response time cost more, and that meant shift gain up infinitely which is not possible for a physical system to do, and there are inherited delay from zero order hold. Without that delay, it still isn't possible due to other factors mentioned.

5.



Integrator decreases steady state error.

6.
Integrator causes overshoot due to large change in setpoint. Cascading a differentiator reduces overshoot, although it can add phase shift to system. A designer needs to weigh these trade-offs carefully.

7.
Some systems may be simple enough that you can get away with out a lead-lag integrator controller. Additionally, lead-lag integrator is much more difficult to design, so in a budget or time constrained project, a simple proportional controller may be preferred.

Conclusion

Proportional controller, while simple and quick to implement, provides less optimal control than lead-lag controller. Lead-lag controller reduces system rise time, overshoot, and steady state error. To get an even lower steady state error, you can cascade an integrator onto the system, at cost of increased overshoot. An engineer has to consider the requirements and trade-offs of the project and design a controller that fits the project. Designing and testing can be done with modelling and verified experimentally.

Appendix:

q1experimental.m

```
list = dir("*.mat");
allNames = {list.name};
for k = 1 : length(allNames)
   % load all data from each file
   name = allNames{k};
   data = load(name);
   time = data.output.time;
   in = data.output.CH1in;
   out = data.output.CH1out;
   sig = data.output.CH1sig;

   % steady state at ~7686
   % rise time = 10% to 90%
   steadystate = out(1,7686);
   [~,idx] = min(abs(out(1, 1:7686)-steadystate*0.9));
   p90 = time(1, idx);
   [~,idx] = min(abs(out(1, 1:7686)-steadystate*0.1));
   p10 = time(1, idx);
   risetime = p90 - p10;

   % overshoot = max - steady state
   maxVal = max(out);
   overshoot = maxVal - steadystate;

   % print
   disp(name);
   disp(risetime);
   disp(overshoot);
end
```

q1simulink.m

```
% p controller
pController
scopeConfig =
get_param('pController/Scope','ScopeConfiguration');
scopeConfig.DataLogging = true;
scopeConfig.DataLoggingSaveFormat = 'Dataset';
out = sim('pController');
printRiseAndOvershoot('p controller', out);

% LL controller
% from prelab q5a
```

```
% a = 13.9282
% T = 7.1074e-04 = .00071074
% K = 13.1800
llController
scopeConfig =
get_param('llController/Scope','ScopeConfiguration');
scopeConfig.DataLogging = true;
scopeConfig.DataLoggingSaveFormat = 'Dataset';
out = sim('llController');
printRiseAndOvershoot('ll controller', out);

% LLI controller
% from prelab q5b
% Ki = 37.7000
lliController
scopeConfig =
get_param('lliController/Scope','ScopeConfiguration');
scopeConfig.DataLogging = true;
scopeConfig.DataLoggingSaveFormat = 'Dataset';
out = sim('lliController');
printRiseAndOvershoot('lli controller', out);

function printRiseAndOvershoot(name, out)
    x1_data = out.ScopeData{1}.Values.Data(:,2);
    x1_time = out.ScopeData{1}.Values.Time;

    % rise time = 10% to 90%
    steadystate = x1_data(25001,1);
    [~,idx] = min(abs(x1_data-steadystate*0.9));
    p90 = x1_time(idx);
    [~,idx] = min(abs(x1_data-steadystate*0.1));
    p10 = x1_time(idx);
    risetime = p90 - p10;

    % overshoot = max - steady state
    maxVal = max(x1_data);
    overshoot = maxVal - steadystate;

    disp(name);
    disp(risetime);
    disp(overshoot);
end
```

q2experimental.m

```
list = dir("*.mat");
```

```
allNames = {list.name};
for k = 1 : length(allNames)
    % load all data from each file
    name = allNames{k};
    data = load(name);
    time = data.output.time;
    in = data.output.CH1in;
    out = data.output.CH1out;
    sig = data.output.CH1sig;

    % steady state at 25000
    sse = out(1, 25000) - in(1, 25000);

    disp(name);
    disp(sse);
end
```

q2simulink.m

```
% p controller
pController
scopeConfig =
get_param('pController/Scope','ScopeConfiguration');
scopeConfig.DataLogging = true;
scopeConfig.DataLoggingSaveFormat = 'Dataset';
out = sim('pController');
printSSE('p controller', out);

% LL controller
llController
scopeConfig =
get_param('llController/Scope','ScopeConfiguration');
scopeConfig.DataLogging = true;
scopeConfig.DataLoggingSaveFormat = 'Dataset';
out = sim('llController');
printSSE('ll controller', out);

% LLI controller
lliController
scopeConfig =
get_param('lliController/Scope','ScopeConfiguration');
scopeConfig.DataLogging = true;
scopeConfig.DataLoggingSaveFormat = 'Dataset';
out = sim('lliController');
printSSE('lli controller', out);
```

```
function printSSE(name, out)
    ramp = out.ScopeData{1}.Values.Data(:,1);
    response = out.ScopeData{1}.Values.Data(:,2);
    sse = abs(response(25001, 1) - ramp(25001, 1)); % use last
value, guaranteed steady state
    disp(name);
    disp(sse);
end
```