Lab 4- System Identification and Digital Control System Design of a Swinging Pendulum

EEE 481, TA: Vishwam Aggrawal

Abhigya Raval, 1210333485

***Abstract*—The lab was an introductory experiment to the students about the catastropic complexities that arise in real life situations and how they can be avoided by simulations. The major concept in this lab was about system identification and design and implementations of a controller based on identified approximate models of plants. The lab is organized into two parts, each reasoning with the previous one. Knowledge from the previous one experiment made it very easy two finish the second part. Once implementing the controller on the boards, it was found that the results were good. This was a very insightful lab experiment overall. First part of the experiment was where we learned how to identify the model of the system based on simple input commands and using advanced mathematics. It was interesting to trial and error on the guessing of the plant transfer function and more interesting to notice how accurate the output was to the guessed function once the one with minimum error was identified. Second part of the experiment was relatively simpler where we designed and tested the controller which was designed using the identified transfer function. The controller performed well, given the fact that the design was based on mere identification based on data.Overall, this was a very informative lab which put in the foundations for the future labs for this class.**

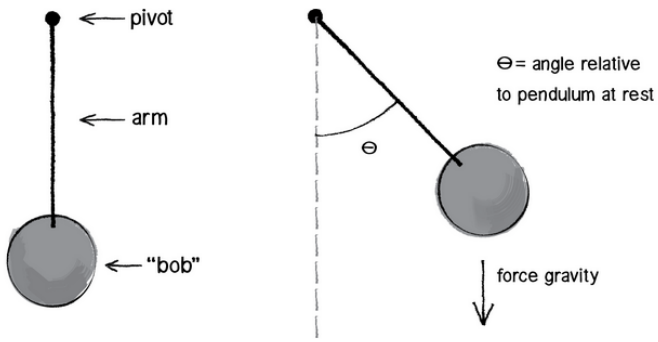

Fig. 1. Plant FBD

I. Introduction

THE purpose of this lab experiment was to introduce students to system identification of dynamic models through MATLAB & SIMULINK. Moreover, discrete PID control is also implemented on external hardware to emphasize of computer control and demonstrating the very crucial and amazing effect of control in the world today. The goal of this lab is to identify the plant transfer function for a given SIMULINKK model using complex mathematical computations and analysis. We also design a control system for maintaining the position of a swinging pendulum. Once the performance is reviewed, a PID controller is implemented and the controlled model is simulated. The model of the swinging pendulum can be defined by the diagram shown in Fig. 1.

A pendulum with its weave hanging straightforwardly underneath the help rotate is at a steady harmony point; there is no torque on the pendulum so it will stay still, and whenever uprooted from this position will encounter a reestablishing torque which returns it toward the harmony position. A pendulum with its weave in an upset position, upheld on an unbending bar legitimately over the rotate, 180 from its steady harmony position, is at a shaky balance point. Now again there is no torque on the pendulum, however the smallest relocation away from this position will cause a attraction torque on the pendulum which will quicken it away from harmony, and it will fall over. This model yields the point an incentive in radians. The useful square outline of the pendulum framework is appeared in Figure 1 and further subtleties of this model can be found here. The System comprises of a Motor Driver square which takes in PWM values between - 255 and 255 (8-piece in one course) and turns an engine, the pole of which is joined to a pendulum.

The following model is provided and needs to be analyzed by implementing a Pseudo Random Binary sequence on as the input to the system and then recodring the outputs given by it.

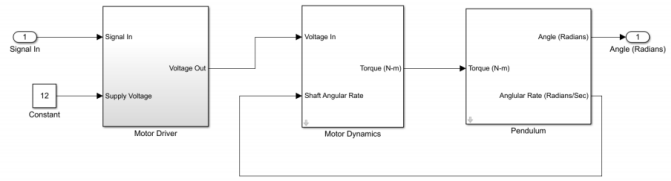

Fig. 2. Model needed to identify

The lab includes using two Arduino Dues that are interconnected over Serial Ports on the Boards. One of these boards is supposed to be programmed to be the controller and the other to be set up to act as a virtual model of the plant. This is a vastly practiced norm in the industry as it eliminates the need for an actual plant and saves tremendous amounts of money.

The controller board shall be programmed in SIMULINK's external mode, meaning it maintains communication with the SIMULINK. Whereas, the plant model board shall be programmed in SIMULINK standalone mode, meaning it is self sufficient.

The first step in designing robust computer control systems is to understand the communications protocols and the possible errors associated with it. This lab serves that purpose by two successive experiments:

1) **Experiment 4.1:**
   Use provided MATLAB scripts to use the data and identify a proper working transfer function of the plant system.
2) **Experiment 4.2:**
   Design and implement a discrete PID controller that was designed from the identified plant in the previous part.

II. Setup and Procedure

For all parts of the experiment with Hardware-In-the-Loop (HIL) testing, board 1 will be programmed in external mode and will be the board primarily functioning as the controller of the system. Board 2 will be programmed in standalone mode and will run the plant model infinitely.

The wiring for these experiemnts looks as follows:

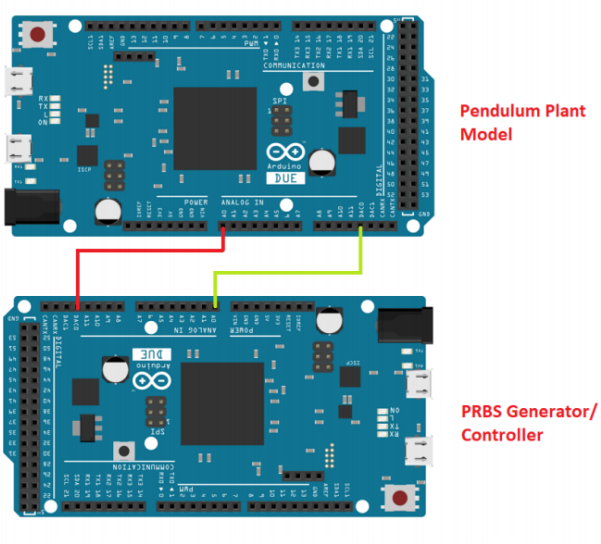

Fig. 3. Wiring schematics

1) The setup for experiment 4.1 is as follows:

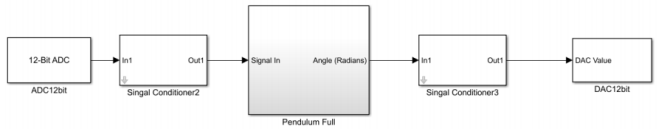

Fig. 4. SIMULINK Model for Plant Board

The model consists of a non-linear model that shall be used in the given MATLAB script **SysIdFunction.m** function to identify the transfer function of the plant based on the data that will be recorded on the PRBS generator side. The equilibrium conditions are set under the mask of the plant model shown above and the output of the MATLAB function is then converted to the transfer function in the s-domain.

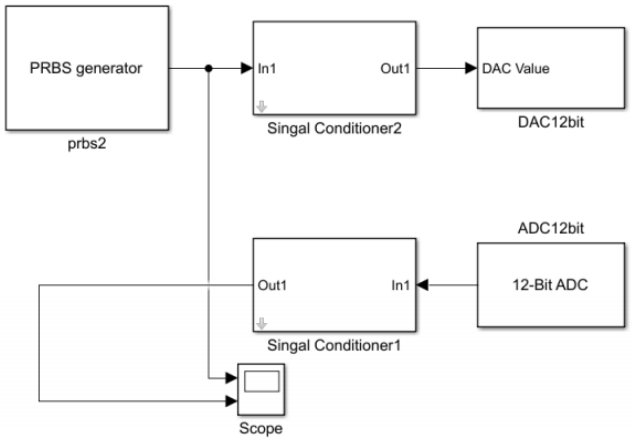

Fig. 5. SIMULINK Model for PRBS Board

The model consists of a Pseudo Random Binary Sequence generator and some signal conditioning blocks provided by the TA. This baord will be run in external mode and the data will be recorded on the scope set up in the model.

2) The setup for Experimental setup for experiment 4.2 is as follows:

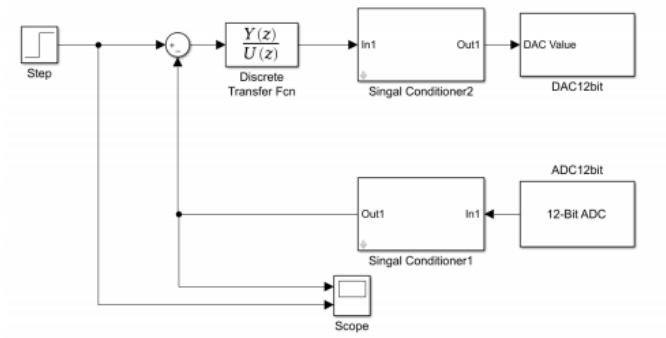

Fig. 6. SIMULINK Model for Controller

Everything else remains the same in the simulation model except for the fact that we have added a controller now. The controller is designed using the identified plant from experiment 4.1.

III. Results & Discussion

1) **Experiment 4.1** The results from this part of the experiment were very interesting. We find that the MATLAB script predicts the plant model very closely by comparing the errors between the actual data and the idetified plant data. Plant identification was done for 1 to 6 orders of power and it was found that the identified plant with the order of 2 was the most robust one. Following are the results.

$$P(s) = \frac{-0.0003333s + 0.03398}{s^2 + 1.52s + 9.922} \tag{1}$$

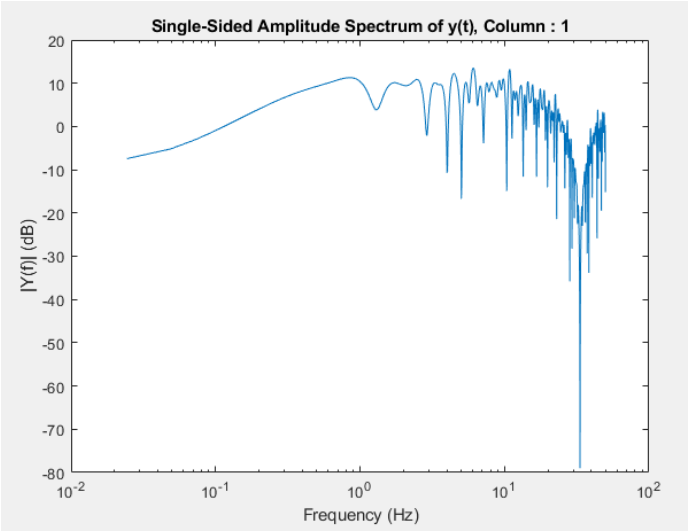

Fig. 7. Fast Fourier Transform

The step response of the system is also shown below:

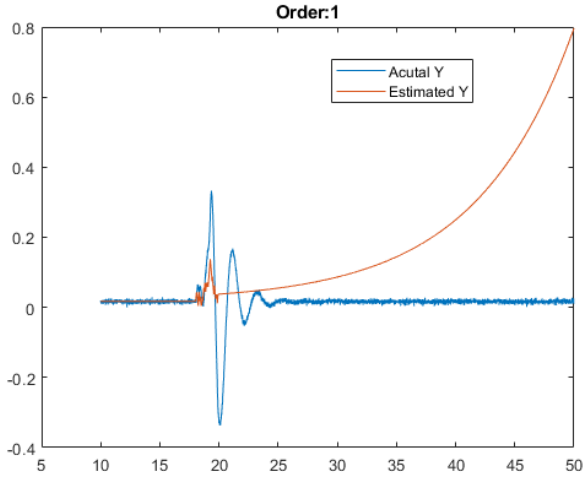

Fig. 8. First order identified plant

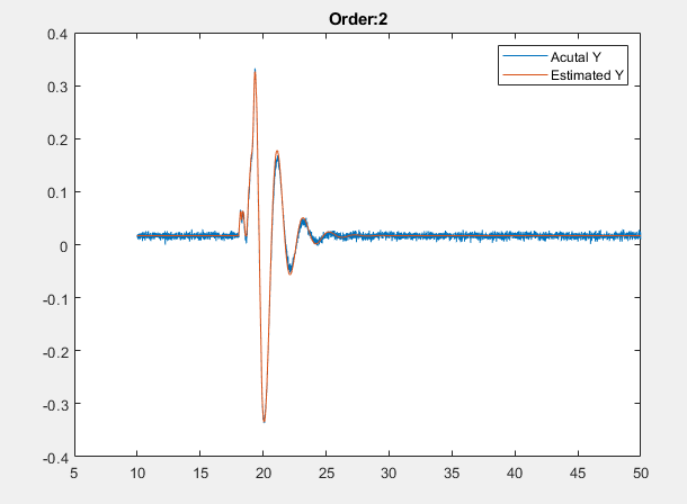

Fig. 9. Second order identified plant

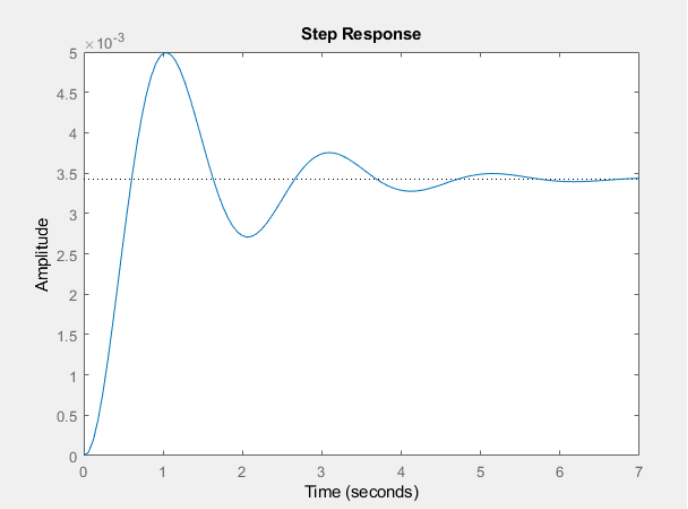

Fig. 10. Open loop Step response

```
>> stepinfo(IdentifiedPlant2)

ans =

  struct with fields:

        RiseTime: 0.3992
    SettlingTime: 5.1712
     SettlingMin: 0.0027
     SettlingMax: 0.0050
       Overshoot: 45.7946
      Undershoot: 0
            Peak: 0.0050
        PeakTime: 1.0300
```

Fig. 11. Open loop step properties

PID Controller design

The controller is designed for a Phase margin of 60 degrees and a bandwidth of 20 rad/s. The code attached in the appendix is used to design the controller. ZOH phase is also included in the design to account for the lag that will be caused when discretizing the plant and controller.

2) **Experiment 4.2**

The transfer function for the second order plant identified model was found to be the following:

$$P(s) = \frac{-0.0003333s + 0.03398}{s^2 + 1.52s + 9.922} \tag{2}$$

The controller is designed to have a phase margin of 60 degrees and to be able to give out a zero steady state error from the plant. A PID controller is chosen and the controller gains are found to be as follows:

$$C(s) = \frac{357.6s^2 + 1836s + 2357}{s(0.005s + 1)} \tag{3}$$

$$k_I = 1836$$

$$k_P = 2357$$

$$k_D = 357.6$$

Here's the code used for getting the exact controller specifications:

```
    % Experiment 4.2- Abhigya Raval
clear all; clc

s = tf('s');
%  Givens

P = (-0.0003333*s + 0.03398)/(s^2 + 1.52*s + 9

BW= 20;
PM = 60; Wgc = BW/1.5;
Ts = 0.01;

% Controller

tau = Ts/2;
zohLag = -Wgc*Ts/2*180/pi; %compute lag from Z
[~,phP]=bode(P,Wgc);%phase of plant
if phP>0
phP = mod(phP,360); % bring phP within +/-360
phP = phP-360; % provide negative shift
end
IntTau = tf(1,[tau 1 0]); % integrator and tau
[~,phIntTau]=bode(IntTau,Wgc);

% phZ = (PM-180-phP-phIntTau)/2; %phase of eac
phZ = (PM-180-phP-phIntTau-zohLag)/2; %phase o

a = Wgc/tand(phZ);% compute zero location
% Get mag for PID with pseudo pole filter at W
[mPintTauZ,~]=bode(P*IntTau*tf([1 2*a a^2],1),
K = 1/mPintTauZ;%find K;
```

```
C_DT=K*(s+a)^2/(s*(tau*s+1)); % PID cont
TryCT = feedback(C_DT*P,1);

CdTustin = c2d(C_DT,Ts,'tustin');
Pzoh = c2d(P,Ts,'zoh');
L_DT=CdTustin*Pzoh;
allmargin(L_DT)
TryTus = feedback(L_DT,1);

figure(1)
step(TryCT,TryTus)
```

Following are the outputs of the step response of the system with the controller of the new system.

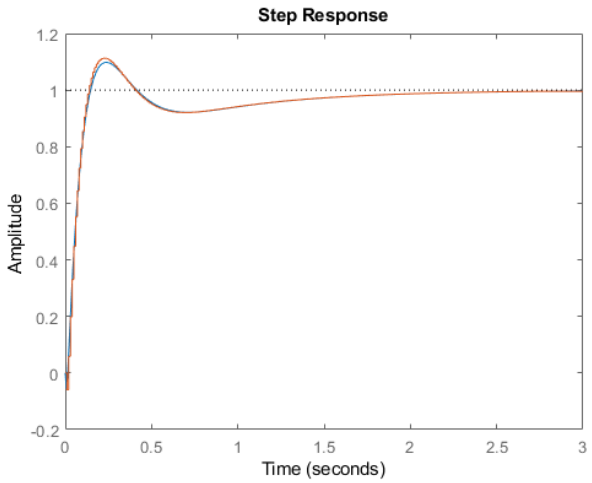

Fig. 12. Close-loop expected Step Response

Now the controller is discretized and applied to the actual plant board and following are the results for a Step input at 25 seconds from 0 to $\frac{\pi}{6}$. As we can see the system settles down without any trouble and the controller has worked.

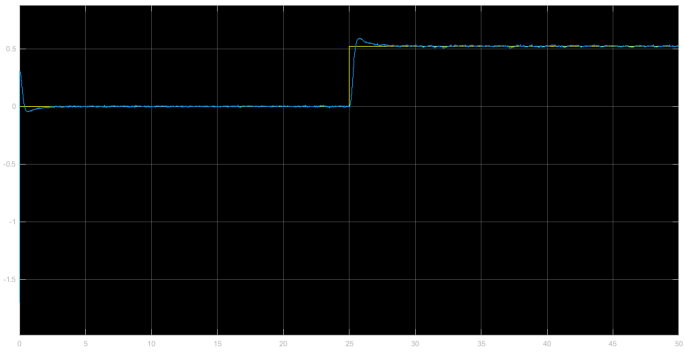

Fig. 13. Step Response of Controlled System

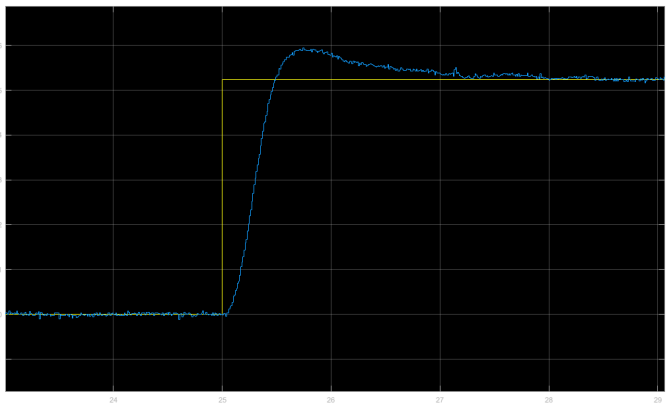

Fig. 14. Step Response- ZOOMED in for better view

And finally, the sensitivity and complementary sensitivity plots are as follows:

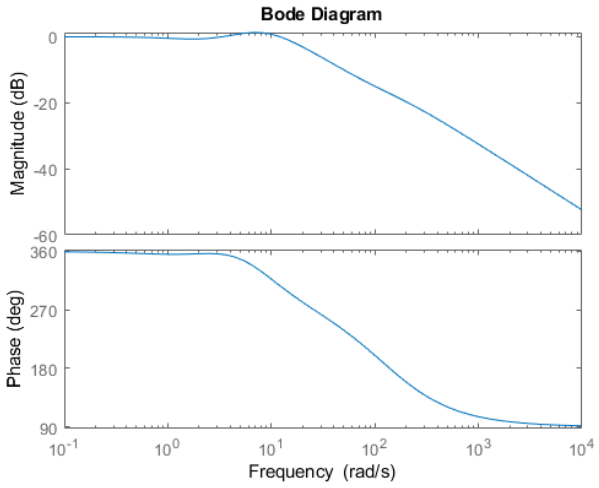

Fig. 15. Sensitivity

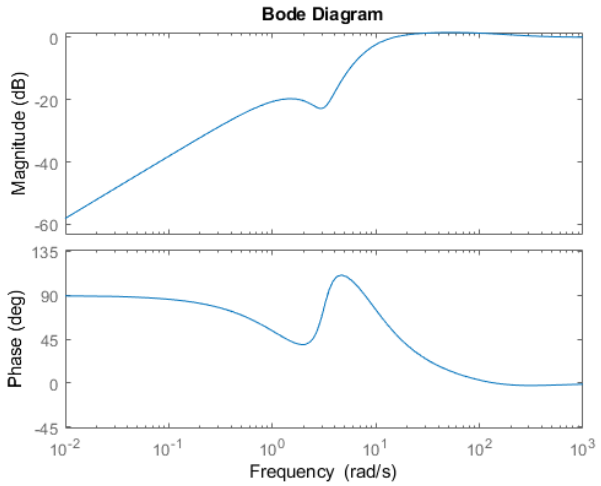

Fig. 16. Complementary Sensitivity

The plots above are a little off due to the fact that the controller is designed with the ZOH lag in it, however the bode plots are plotted using the plant without ZOH and continuous.
Just for fun, I added a filter to the system to see if we can minimize the overshoot and it worked. Following are the results:

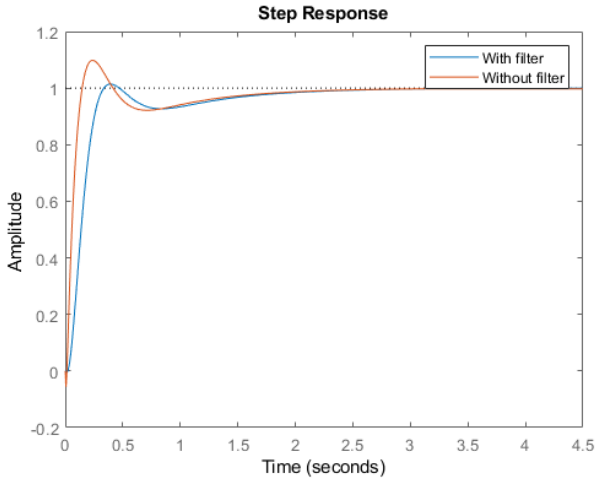

Fig. 17. Filter added step response with a controller

IV. Conclusion

It is self evident from the results above that, designing and building control systems consists of a lot more than just analytical work. The purpose of having a feedback system is to do tasks autonomously, as fast as possible and some of the applications wouldn't even allow for a margin of delay as small as one sampling time. It was very interesting to see these phenomenon in first person. It was seen that adding a controller that is well designed about a system can drastically improve the system stability, settling time, overshoot and error mitigation. This means time and money are saved and technologies are made better.

In the first part of the experiment, the prediction of the plant model was one of the most interesting projects I have gotten to do. On my free time, I will definitely dissect and understand the code behind the masks of the scripts provided. The idea of plotting the fast fourier trasnforms and seeing the energy of the system at the certain bandwidtch was another one of the new things we got to learn.

For the second part of the experiment, a controller designed with the approxnimated plant from the previous experiemnt was implemented and it was seen that the controller worked fine. This proves the reliability of the identified plant model and how this works on toward the approximation of an approximation.

Secondly, the theory taught in class about PID control, plant modelling, disturbance mitigation, limitations of hardware, voltage regulations, quantization and analysis all came to life in this experiment. Understanding the control properties of a PID controller was eye opening. The challenge to figure out the plant model just by using the datasets open up the scope of machine learning and how control properties can be implemented there.

Overall, it was realized that better design of a controller and more importantly a better design of the plant model can significantly improve the control on the system and make the system more robust to changes and disturbances and the controller more efficient!

Appendix A
MATLAB Script

```
% Abhigya Raval Lab 4 EEE481

%% Inputting the constants
clear all; clc
Ts = 0.01;     % Sample time
Tsw = 0.03; % Switching time

% Motor
R = 2.4;
K = 0.9;

% Pendulum

m = 0.5; % mass
L = 1;       % length of bob

epsilon = 0.5;

theta_0 = 0;   % init
theta_dot_0 = 0; % init



%% data
```

```
T = ScopeData.time(1000:end);
U = ScopeData.signals(1).values(1000:end);
Y = ScopeData.signals(2).values(1000:end);

% % figure(1)
% plot(T,U)
% hold on
%
% figure(2)
% plot(T,Y)


%% fft
figure(3)
plotfft(1/Ts,U,1,'logdb');

%% System identifications

BW = 20;
%
% [IdentifiedPlant1, error1] = SysIdFunction(T,U,Y,BW,1);
% mse1 = mean(error1.^2);

[IdentifiedPlant2, error2] = SysIdFunction(T,U,Y,BW,2);
mse2 = mean(error2.^2);

% [IdentifiedPlant3, error3] = SysIdFunction(T,U,Y,BW,3);
% mse3 = mean(error3.^2);
%
% [IdentifiedPlant4, error4] = SysIdFunction(T,U,Y,BW,4);
% mse4 = mean(error4.^2);

% [IdentifiedPlant5, error5] = SysIdFunction(T,U,Y,BW,5);
% mse5 = mean(error5.^2);

% [IdentifiedPlant6, error6] = SysIdFunction(T,U,Y,BW,6);
% mse6 = mean(error6.^2);


%% Some more stuff
step(IdentifiedPlant2)


%% Experiment 4.2

% Experiment 4.2- Abhigya Raval
clear all; clc

s = tf('s');
%  Givens

P = (-0.0003333*s + 0.03398)/(s^2 + 1.52*s + 0.922);

BW= 20;
PM = 60; Wgc = BW/1.5;
Ts = 0.01;

% Controller
tau = Ts/2;
zohLag = -Wgc*Ts/2*180/pi; %compute lag from ZOH
[~,phP]=bode(P,Wgc);%phase of plant
if phP>0
phP = mod(phP,360); % bring phP within +/-360
phP = phP-360; % provide negative shift
end
IntTau = tf(1,[tau 1 0]); % integrator and tau pha
[~,phIntTau]=bode(IntTau,Wgc);

% phZ = (PM-180-phP-phIntTau)/2; %phase of each ze
phZ = (PM-180-phP-phIntTau-zohLag)/2; %phase of ea

a = Wgc/tand(phZ);% compute zero location
% Get mag for PID with pseudo pole filter at Wgc
[mPintTauZ,~]=bode(P*IntTau*tf([1 2*a a^2],1),Wgc)
K = 1/mPintTauZ;%find K;

C_DT=K*(s+a)^2/(s*(tau*s+1)); % PID controller
TryCT=feedback(C_DT*P,1);

CdTustin = c2d(C_DT,Ts,'tustin');
Pzoh = c2d(P,Ts,'zoh');
L_DT=CdTustin*Pzoh;
allmargin(L_DT)
TryTus=feedback(L_DT,1);

figure(1)
step(TryCT,TryTus)
%% Sensitivity
figure(4)
S = 1/(1+P*C_DT);
CS = 1 -S;

bode(S, BW, 6);
figure(5)
bode(CS)

%% Part 5
M = 4;
F = (a*M)/(s+(a*M));

newTR = F*TryCT;
step(newTR, TryCT)
legend('With filter','Without filter')
```

APPENDIX B
REFERENCES

[1] Aggarwal Vishwam, "EEE481- Computer Controlled Systems Lab 4 Manual", Arizona State University, Tempe AZ, March. 2020

[2] Charles L. Phillips, H. Troy Nagle, Aranya Chakrabortty, Digital Control System Analysis and Design, Pearson, 2015

APPENDIX C
ACKNOWLEDGEMENTS

I'd like to thank my lab teaching assistant, Vishwam Aggarwal and my peers on Hangouts class group for helping me understand the concepts better and tackling the labs with a critical approach.