ECEN315 - Open Loop Response of a Motorised, Propellor Driven Pendulum

Joshua Benfell - 300433229

Abstract—We propose

1 Introduction

The aim of this report is to derive a model for the open loop response of a motorised, propellor driven pendulum arm. This is being derived for the eventual development of controllers for this system.

2 Background

3 Methods

4 Results

5 Discussion

6 Conclusion

Appendix A
Matlab Code

```
%Script for ECEN315 Lab 2
clear
clf
R_a = 6.3; %Ohms
L_a = 0.797; %H
K_b = 0.0043; %Vs/rad
K_t = K_b; %Nm/A
D_m = 0.00000553; %Nms/rad
J_m = 0.00000241; %Kgm^2

numerator = K_t/(L_a*J_m);
sSqCoef = 1;
sCoef = (L_a * D_m + R_a * J_m)/(L_a * J_m);
coef3 = (R_a * D_m + K_t * K_b)/(L_a * J_m);

sys = tf(numerator , [sSqCoef sCoef coef3])
stepCoefs = [1 2 3 4 5 6];
steps = 6;

Legend = cell(steps, 1);

for i = 1:steps
    step(i * sys, 10)
    hold on
    Legend{i} = strcat(num2str(i), 'V Step');
end
hold off
ylabel("\omega_m (rad/s)");
```

```
xlabel("Time (s)");
title("Step Response of the Motor");
legend(Legend)

stepResponseInfo = stepinfo(stepCoefs.*sys)%, 'SettlingTimeThreshold', 0.000005); %Include this setting to know approx when oscillation ends

%units of steady state gains rad/Vs
Gain = numerator/coef3
steadyStateValue = Gain * stepCoefs

SettlingTime = [stepResponseInfo(1:6).SettlingTime] %The inductance of the motor is likely too high which is causing the critical damping/overdamping and thus quick settling time.
%steadyStateValue = [stepResponseInfo(1:6).Peak]
%steadyStateGain = [stepResponseInfo(1:6).Peak] ./ stepCoefs %Not sure about this method
```

```
% Finding Coefficients from un-driven Damped pendulum
clear
clf

m = 0.168; %kg
g = 9.81;  %m/s^2
d = 0.14; %m distance from pivot to cog
r = 0.165; %m length of pendulum arm
T = 0.97; %Period in seconds
f = 1/T; %freq
w = 2*pi*f; %angular frequency

data = readtable("Data.csv"); % load csv data

figure(1)
plot(data.Var1, data.Var2); % plot csv data
xlabel("Time (s)")
ylabel("y(t)")

positiveValues = abs(data.Var2); % make all peaks positive

[peaks, locs] = findpeaks(positiveValues); % find all peaks
peaksTable = table(data.Var1(locs), peaks); % convert to table

hold on
plot(peaksTable.Var1, peaksTable.peaks); % plot line connecting all peaks
hold off

%Fitting coefficients to the function
modelfun = @(b,x) b(1)*exp(-b(2).*x(:, 1)); % function to model
beta0 = [200, 1]; % initial Guesses
model = fitnlm(peaksTable, modelfun, beta0); % the non-linear model we get out
coefs = model.Coefficients{:, 'Estimate'}; % Coefficients we are looking for


A = coefs(1); % Our A coefficient
B = coefs(2); % Our B coefficient

j_p = (m*d*g) / (power(w,2)+power(B,2)) % moment of inertia we are looking for
c = 2 * B * j_p % damping coeff we are looking for

% Transfer Function

numerator = 1/j_p;
coefB = c/j_p;
```

```
coefC = (d*m*g)/j_p;

sys = tf(numerator, [1, coefB, coefC])
figure(2)
step(sys)


% Combined Transfer function
% Lab 2 Transfer Function
R_a = 6.3; %Ohms
L_a = 0.797; %H
K_b = 0.0043; %Vs/rad
K_t = K_b; %Nm/A
D_m = 0.00000553; %Nms/rad
J_m = 0.00000241; %Kgm^2

numeratorLab2 = K_t/(L_a*J_m);
coefALab2 = 1;
coefBLab2 = (L_a * D_m + R_a * J_m)/(L_a * J_m);
coefCLab2 = (R_a * D_m + K_t * K_b)/(L_a * J_m);

sysLab2 = tf(numeratorLab2 , [coefALab2 coefBLab2 coefCLab2]);
ylabel("\theta (rads)")

k_p = 0.0053;

sys3 = sysLab2 * sys * k_p * r
Leg = cell(3, 1);
figure(3)
hold on
for i = 3:5
    step(i * sys3);
    Leg{i-2} = strcat(num2str(i), 'V Step');
end
hold off
ylabel("\theta (rads)")
legend(Leg)
```

Appendix B
Risk Assessment