ME EN 6230
Lab 1
Ryan Dalby

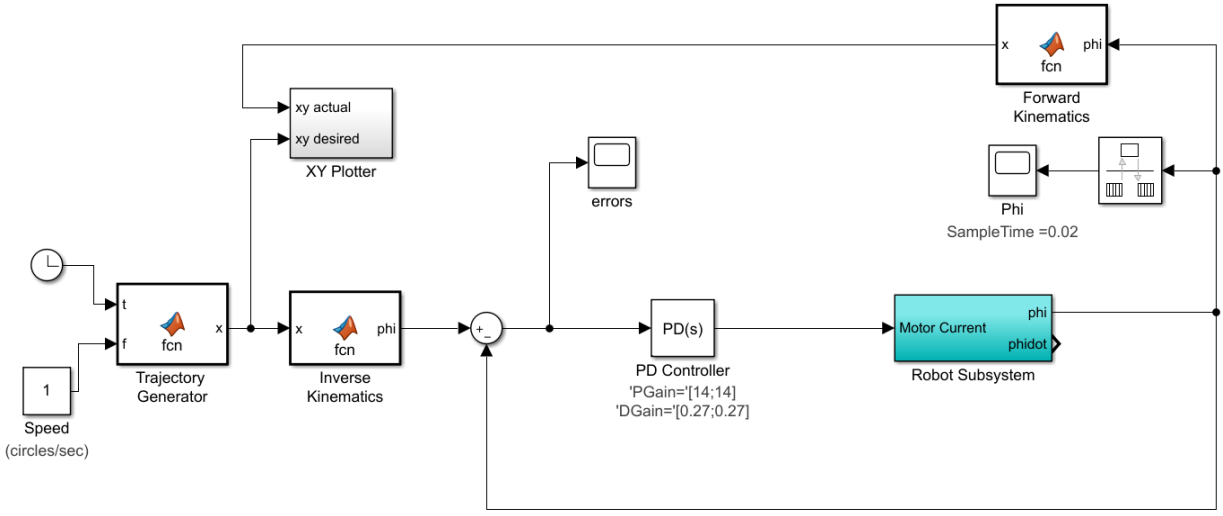
Note that lab station 2 was used for this lab. The amp bias used was [-0.042; 0.065] and the current sense bias used was [0.02; 0.037].

1. Decentralized Control

Note: PD gains are Kp = 14 and Kd = 0.27 for all decentralized control. (These were found in PS5)

1.1. PD Control Only

1.1.1. Model

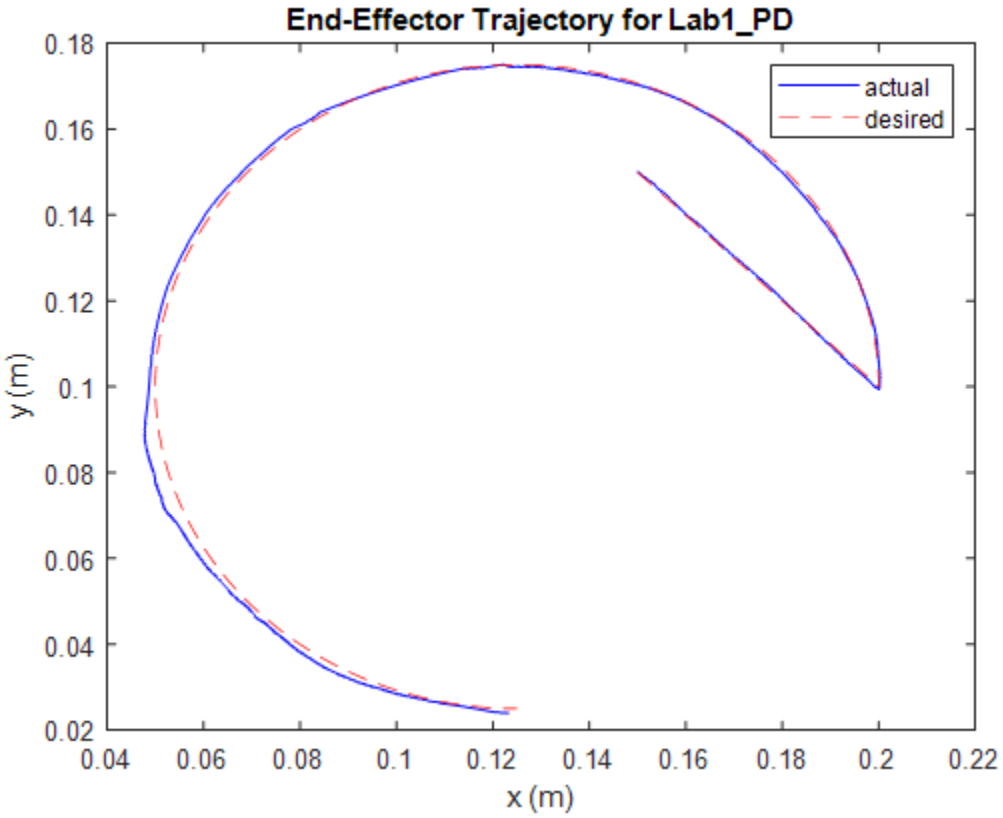


1.1.2. Code

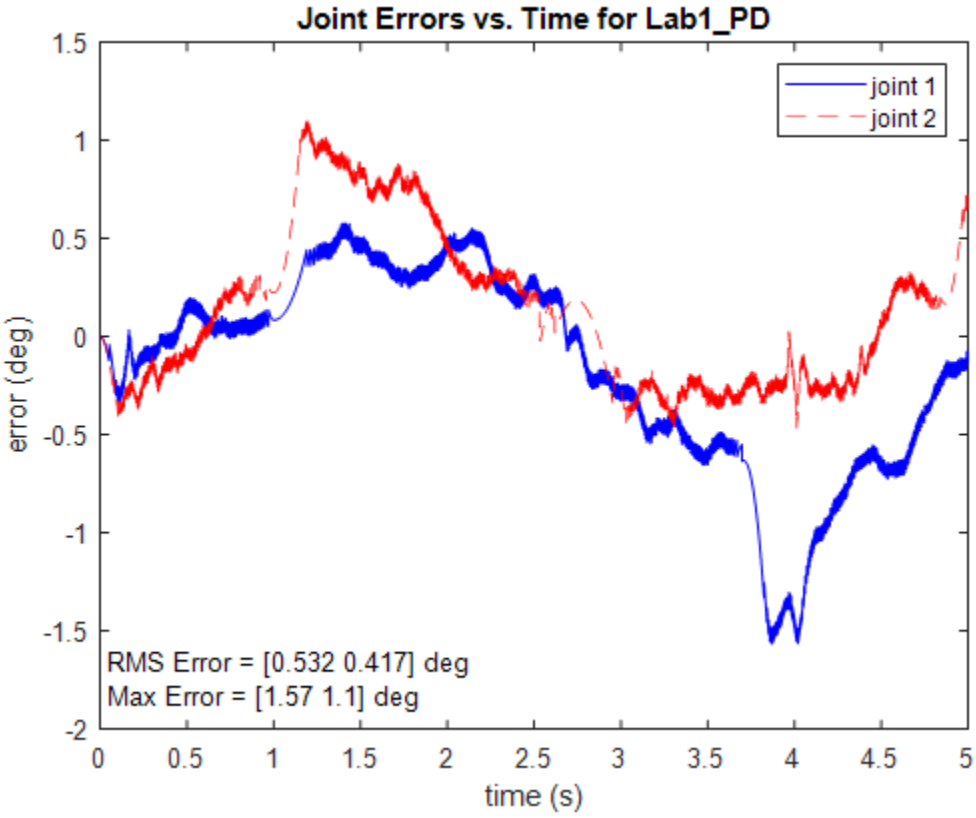
No code for model besides plotting code which is attached at the end of this document.

1.1.3. Low Speed Simulation (f=0.2 circles/s)

1.1.3.1. X-y trajectory

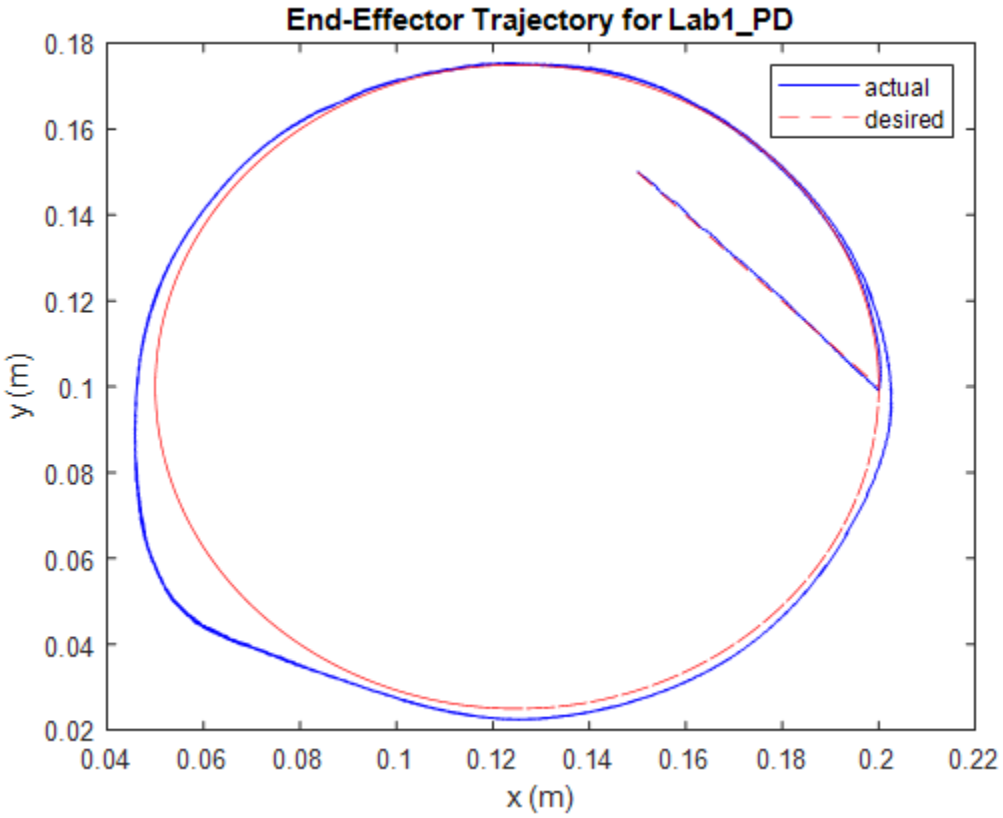


1.1.3.2. Joint angle errors

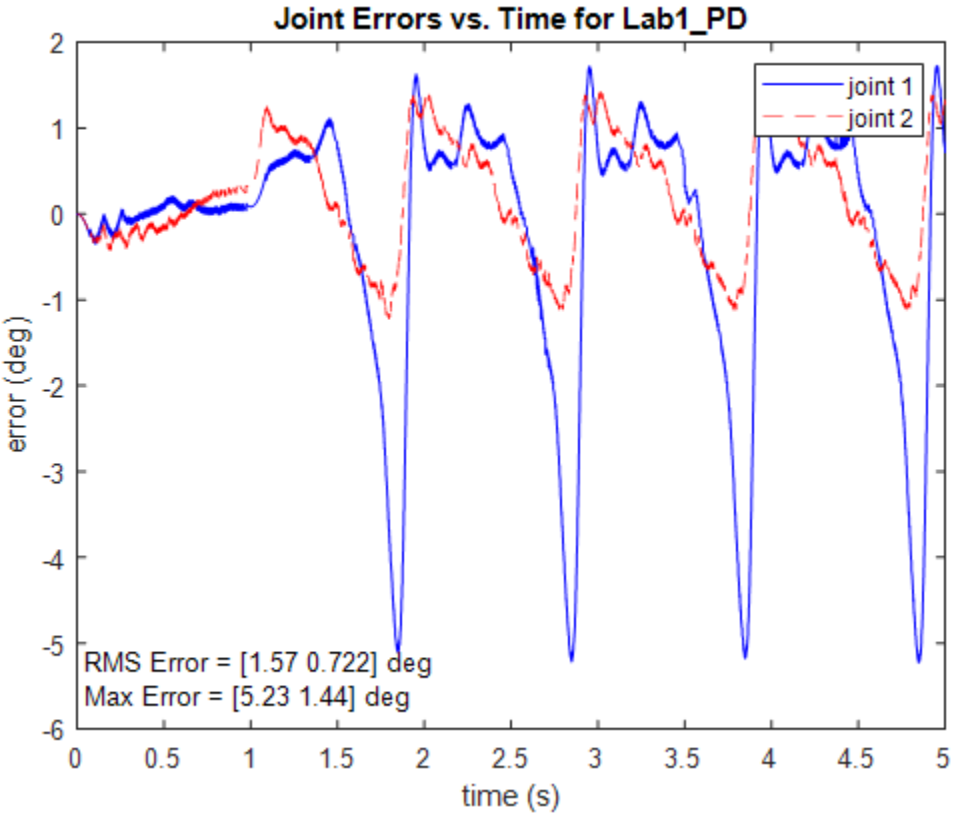


1.1.4. High Speed Simulation (f=1 circles/s)

1.1.4.1. X-y trajectory

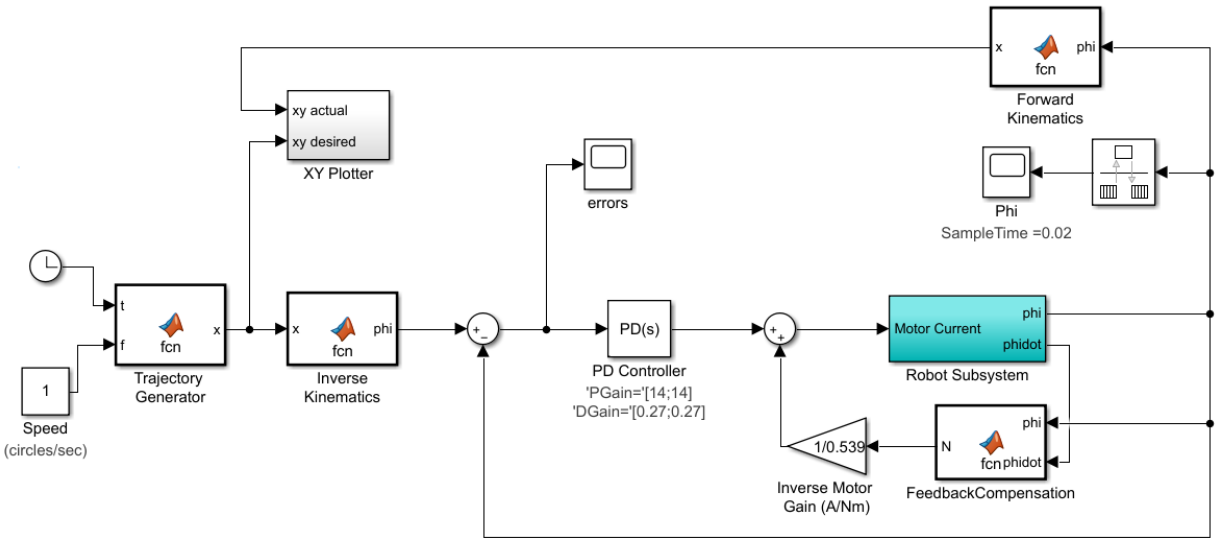


1.1.4.2. Joint angle errors



1.2. PD Control with gravity, friction, and centripetal/coriolis feedback compensation

1.2.1. Model



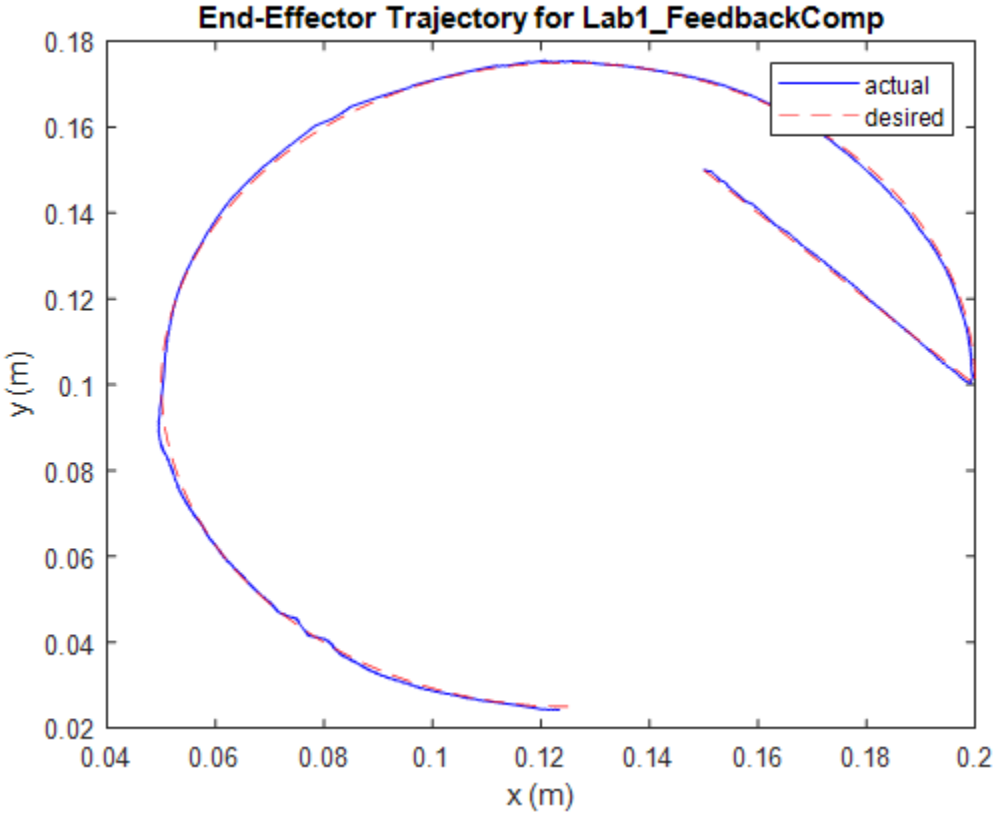
1.2.2. Code

Code for FeedbackCompensation Block:

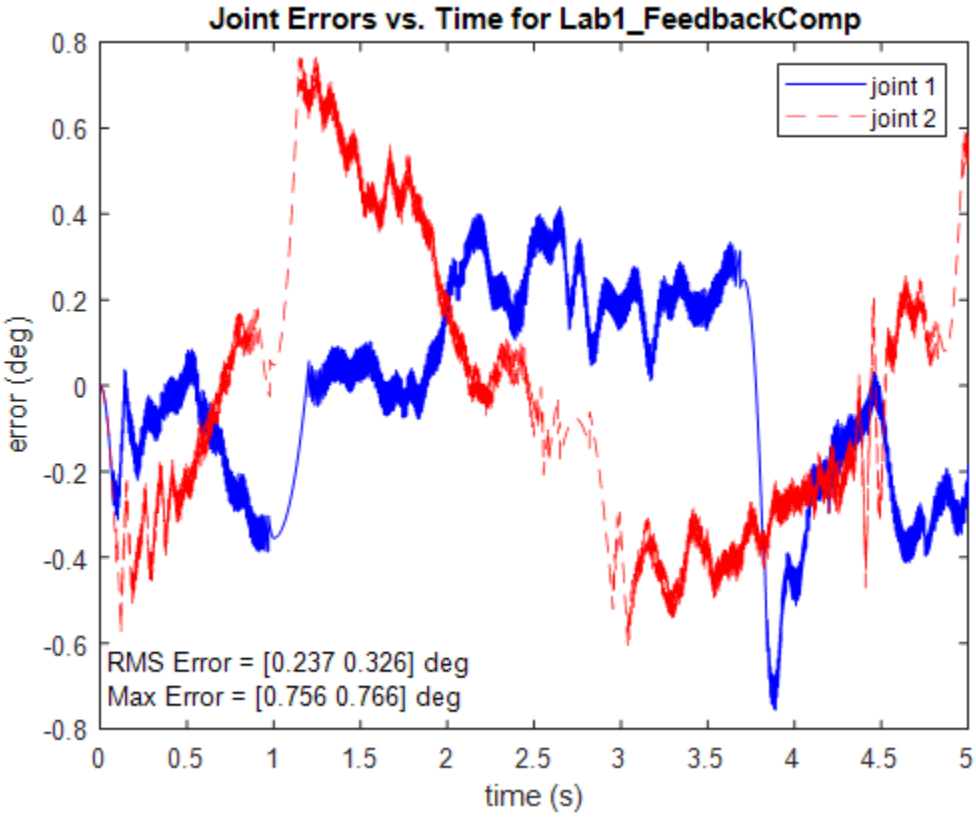
```
function N = fcn(phi, phidot)

% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

a1 = 0.15;  % link 1 length
a2 = 0.15;  % link 2 length
m1 = 0.092;  % link 1 mass
m2 = 0.077; % link 2 mas
r01 = 0.062; % link 1 center of mass
r12 = 0.036; % link 2 COM
I1 = 0.64e-3;  % link 1 inertia
I2 = 0.30e-3;  % link 2 inertia
Jm1 = 0.65e-6; % motor inertias
Jm2 = 0.65e-6;
b1 = 3.1e-6; % viscous damping constants
b2 = 3.1e-6;
c1 = 0.0001; % coulomb friction constants
c2 = 0.0001;
g = 9.8; % gravitational constant
N1 = 70; % gear ratios
N2 = 70;

h = a1*r12*m2*sin(phi(2)-phi(1));
G1 = (r01*m1+a1*m2)*g*cos(phi(1));
G2 = r12*m2*g*cos(phi(2));
F1 = N1^2*b1*phidot(1) + N1*c1*sign(phidot(1));
F2 = N2^2*b2*phidot(2) + N2*c2*sign(phidot(2));

V = [0 -h ;h 0]*[phidot(1)^2;phidot(2)^2]; % centripetal torques
G = [G1;G2]; % gravity torques
F = [F1;F2]; % frictional torques

N = V + G + F;
```

1.2.3. Low Speed Simulation (f=0.2 circles/s)

1.2.3.1. X-y trajectory

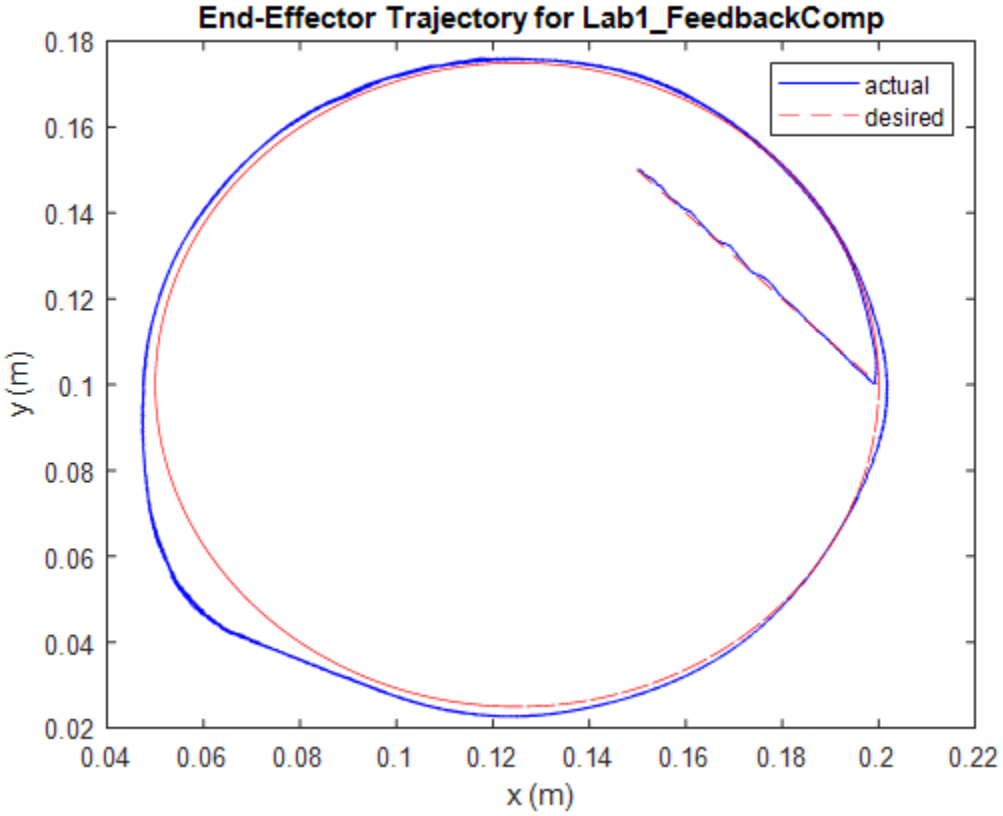


1.2.3.2. Joint angle errors

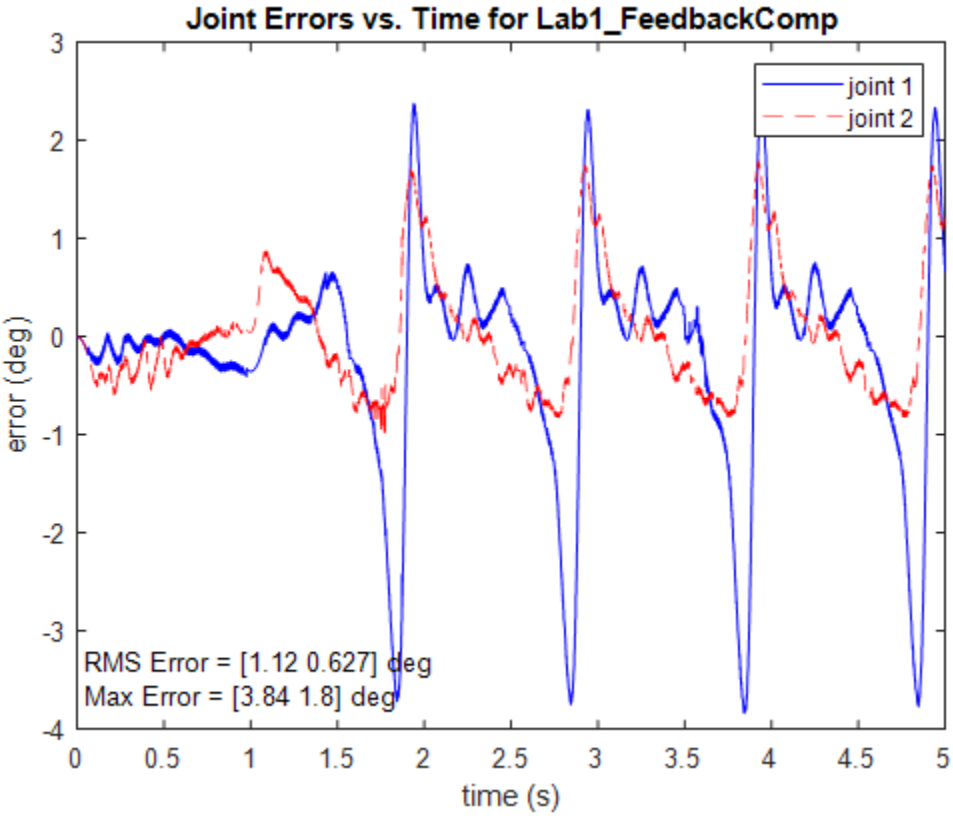


1.2.4. High Speed Simulation (f=1 circles/s)

1.2.4.1. X-y trajectory

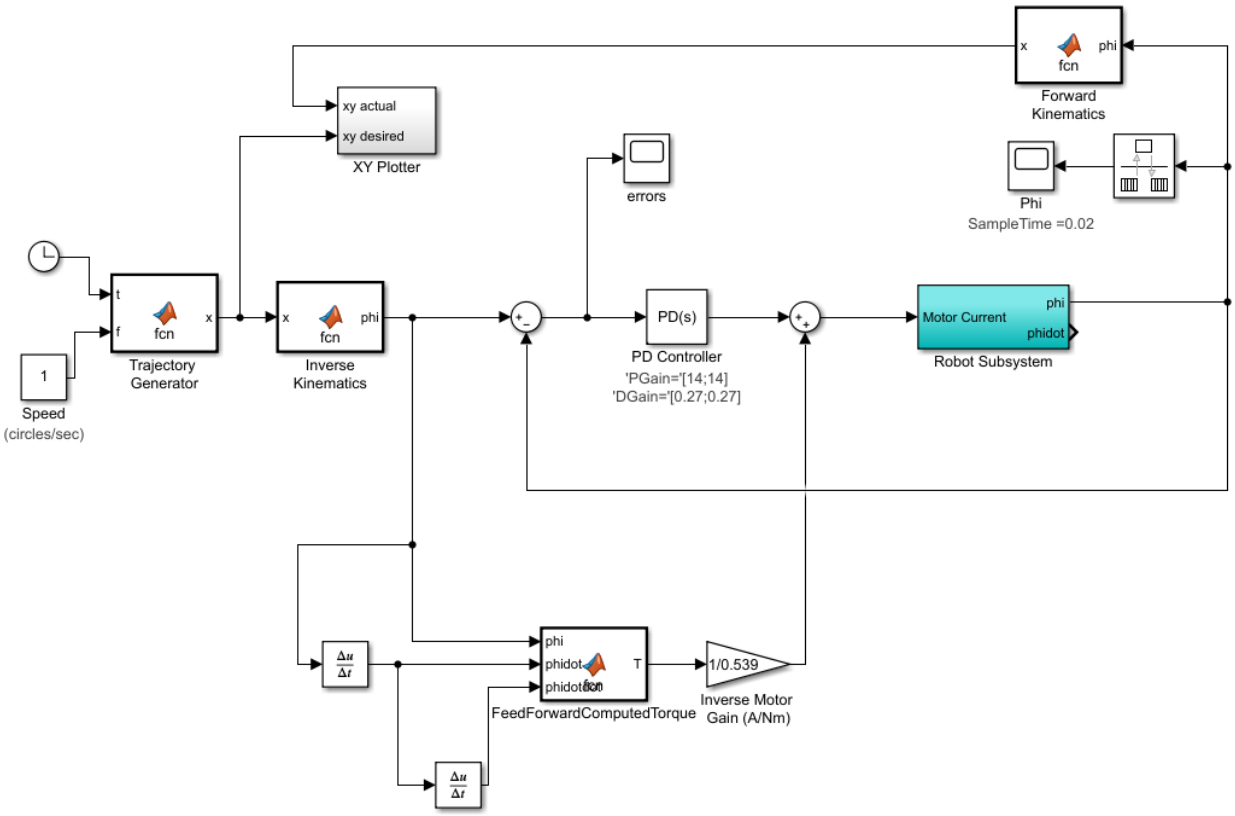


1.2.4.2. Joint angle errors



1.3. PD Control with Computed Torque Feedforward Control

1.3.1. Model



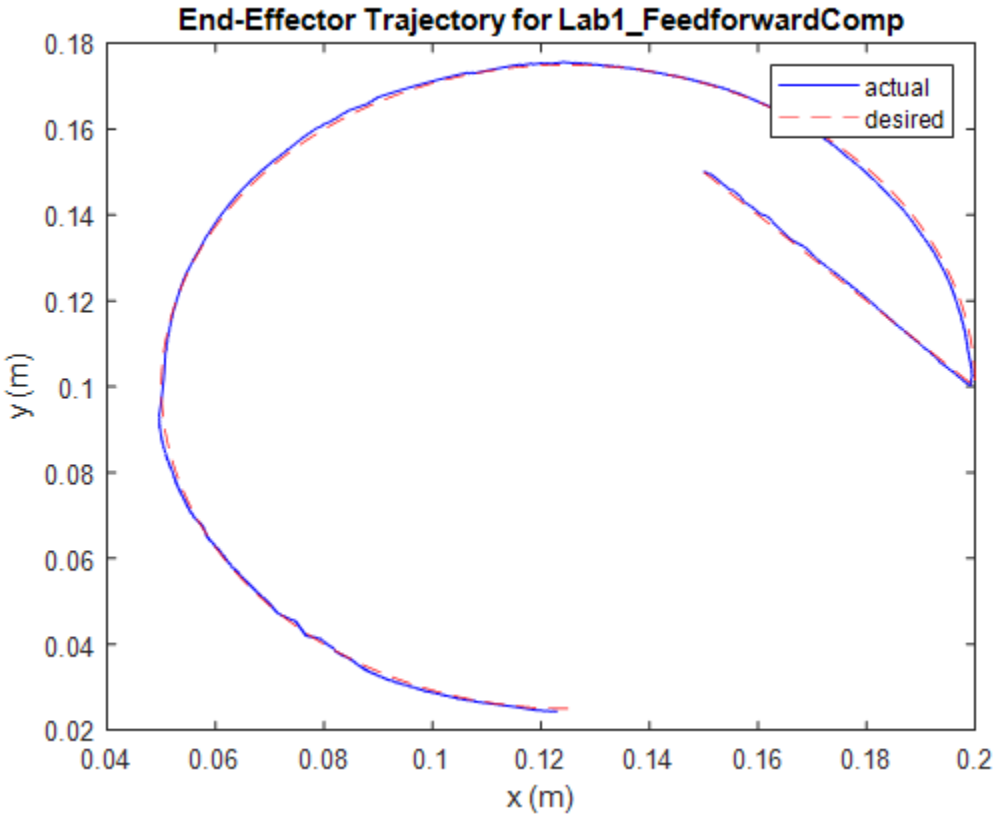
1.3.2. Code

Code for FeedForwardComputedTorque Block:

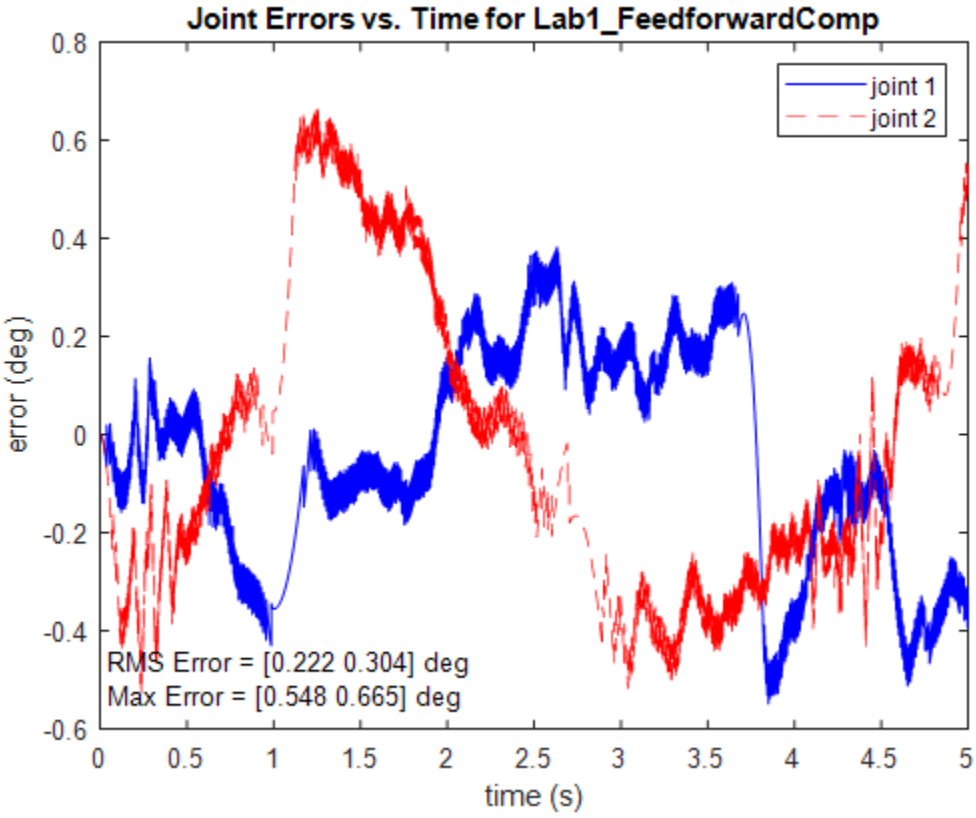
```
function T = fcn(phi,phidot,phidotdot)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

a1 = 0.15;  % link 1 length
a2 = 0.15;  % link 2 length
m1 = 0.092;  % link 1 mass
m2 = 0.077; % link 2 mas
r01 = 0.062; % link 1 center of mass
r12 = 0.036; % link 2 COM
I1 = 0.64e-3;  % link 1 inertia
I2 = 0.30e-3;  % link 2 inertia
Jm1 = 0.65e-6; % motor inertias
Jm2 = 0.65e-6;
b1 = 3.1e-6; % viscous damping constants
b2 = 3.1e-6;
c1 = 0.0001; % coulomb friction constants
c2 = 0.0001;
g = 9.8; % gravitational constant
N1 = 70; % gear ratios
N2 = 70;

H11 = N1^2*Jm1 + I1 + m2*a1^2;
H12 = a1*r12*m2*cos(phi(2)-phi(1));
H21 = H12;
H22 = N2^2*Jm2 + I2;
h = a1*r12*m2*sin(phi(2)-phi(1));
G1 = (r01*m1+a1*m2)*g*cos(phi(1));
G2 = r12*m2*g*cos(phi(2));
F1 = N1^2*b1*phidot(1) + N1*c1*sign(phidot(1));
F2 = N2^2*b2*phidot(2) + N2*c2*sign(phidot(2));

H = [H11 H12; H21 H22]; % inertia matrix
V = [0 -h ;h 0]*[phidot(1)^2;phidot(2)^2]; % centripetal torques
G = [G1;G2]; % gravity torques
F = [F1;F2]; % frictional torques

T = H*phidotdot + V + G + F;
```

1.3.3. Low Speed Simulation (f=0.2 circles/s)

1.3.3.1. X-y trajectory

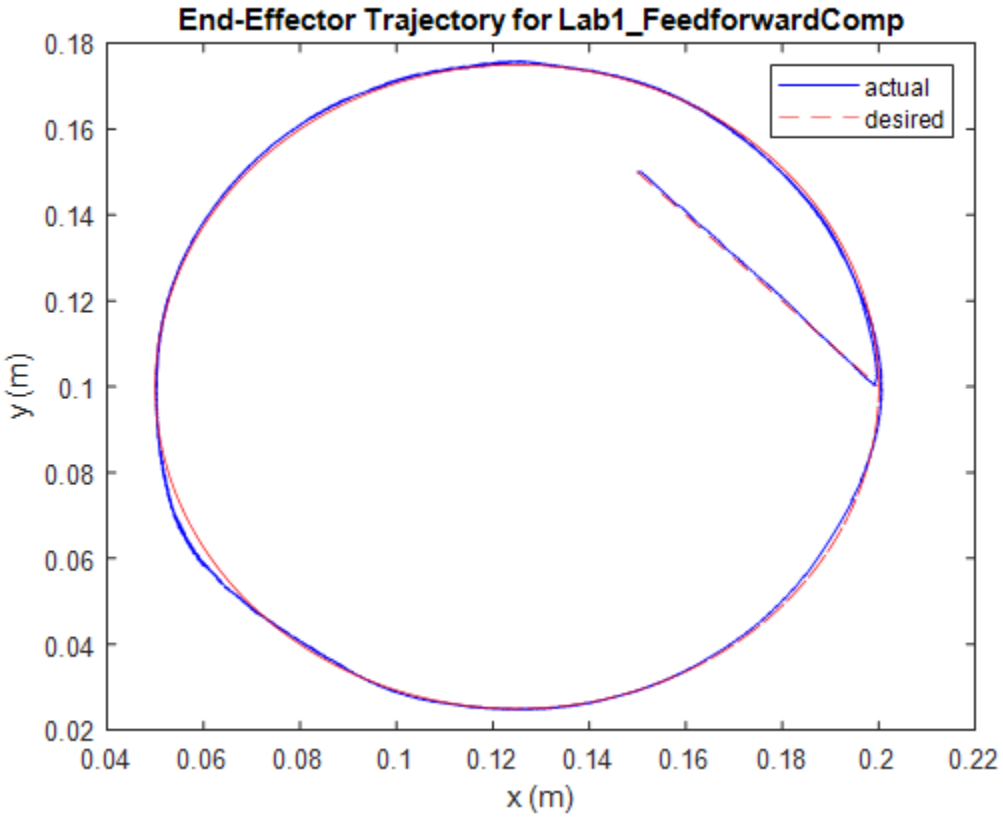


1.3.3.2. Joint angle errors

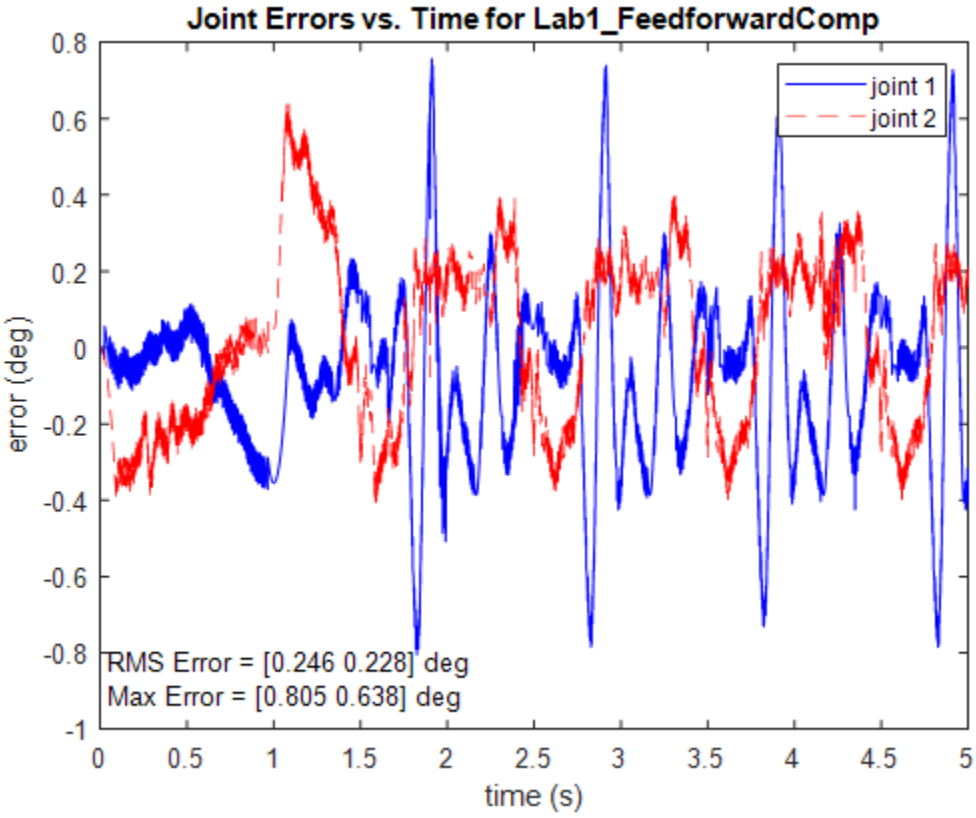


1.3.4. High Speed Simulation (f=1 circles/s)

1.3.4.1. X-y trajectory



1.3.4.2. Joint angle errors

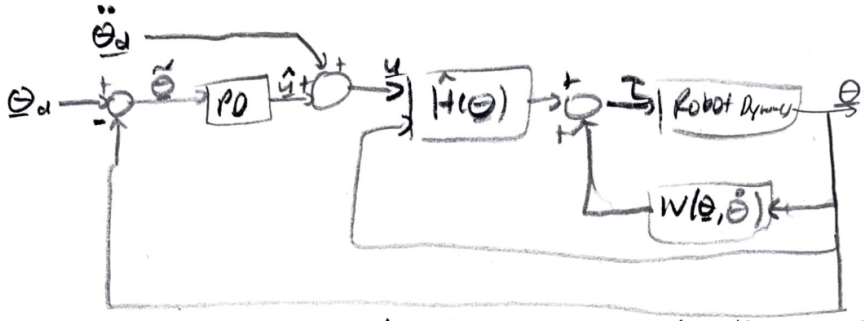


2. Centralized Control

2.1. Inverse Dynamics Controller

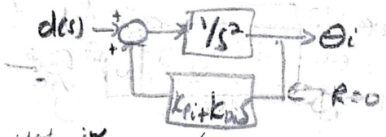
2.1.1. PD gain design

2 2.1 IDC controller



b) Assuming perfect model, indexed notation and $PD = K_{Pi} + K_{Pi}s$, w/ disturbance, $d(s)$

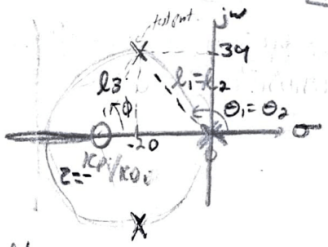
will design for disturbance rejection:



loop gain:

$$\frac{K_{Pi}+K_{Di}s}{s^2} = \frac{K_{Di}\left(s+K_{Pi}/K_{Di}\right)}{s^2}$$

↖ can do root locus now



Poles: $s=0$ $s=0$

Zeros: $s=-K_{Pi}/K_{Di}$

For $\%OS = 20\%$ and $T_S = 0.2s$

↳ Desired CLP: $s = -20 \pm 34.04j$

Angle Condition:

$$\phi_1 - 2\theta_1 = \pm 180°$$

$$\phi = \pm 180° + 2\tan^{-1}\left(\frac{34}{-20}\right) \quad \phi = 54.25°$$

$$\tan(\phi_1) = \left(\frac{34}{-20-z}\right) \quad z = -48.07 = -K_{Pi}/K_{Di}$$

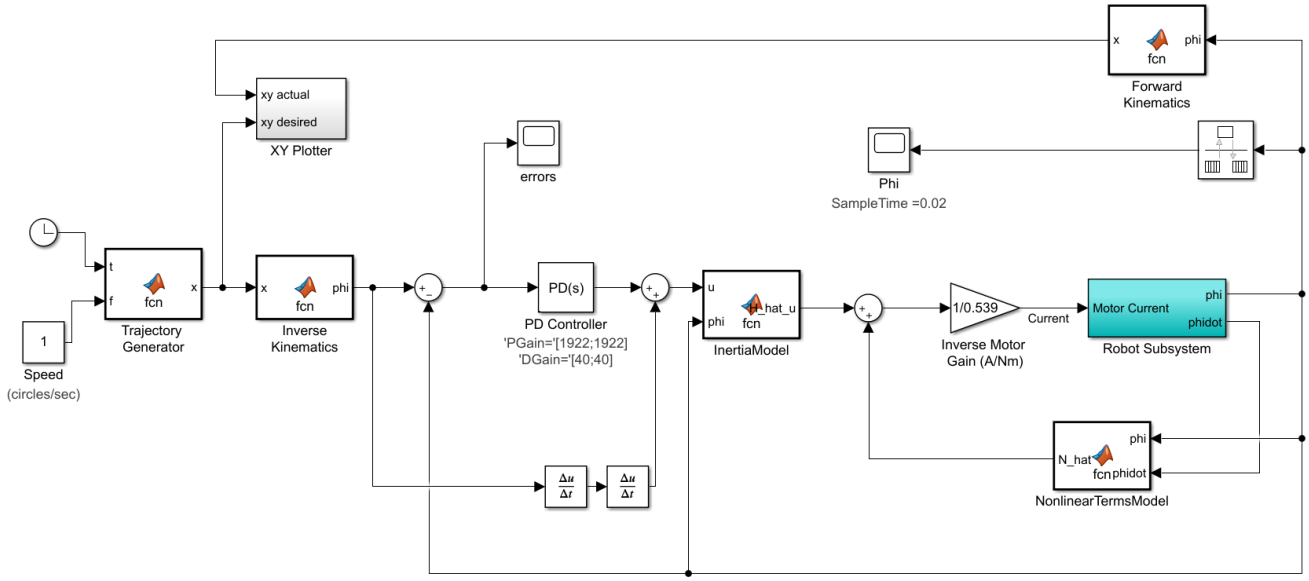
Magnitude Condition:

$$K_{Di} = \frac{(l_1)^2}{l_3} = \frac{\sqrt{(34)^2+(20)^2}^2}{\sqrt{(28.02)^2+(\infty)^2}} = \frac{40}{34} = 40$$

$$K_{Pi} = K_{Di}(48.07) = 1922$$

$K_{Di} = 40$ $K_{Pi} = 1922$

2.1.2. Model


x
fcn
phi
Forward
Kinematics
xy actual
xy desired
XY Plotter
errors
Phi
SampleTime =0.02
t
f
fcn
x
Trajectory
Generator
1
Speed
(circles/sec)
x
fcn
phi
Inverse
Kinematics
PD(s)
PD Controller
'PGain='[1922;1922]
'DGain='[40;40]
u
phi
N_hat_u
fcn
InertiaModel
1/0.539
Inverse Motor
Gain (A/Nm)
Current
Motor Current
phi
phidot
Robot Subsystem
Δu
Δt
Δu
Δt
phi
N_hat
fcn
phidot
NonlinearTermsModel

2.1.3. Code

Code for InertiaModel Block:

```
function H_hat_u = fcn(u,phi)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

a1 = 0.15;  % link 1 length
a2 = 0.15;  % link 2 length
m1 = 0.092;  % link 1 mass
m2 = 0.077; % link 2 mas
r01 = 0.062; % link 1 center of mass
r12 = 0.036; % link 2 COM
I1 = 0.64e-3;  % link 1 inertia
I2 = 0.30e-3;  % link 2 inertia
Jm1 = 0.65e-6; % motor inertias
Jm2 = 0.65e-6;
b1 = 3.1e-6; % viscous damping constants
b2 = 3.1e-6;
c1 = 0.0001; % coulomb friction constants
c2 = 0.0001;
g = 9.8; % gravitational constant
N1 = 70; % gear ratios
N2 = 70;

H11 = N1^2*Jm1 + I1 + m2*a1^2;
H12 = a1*r12*m2*cos(phi(2)-phi(1));
H21 = H12;
H22 = N2^2*Jm2 + I2;

H_hat = [H11 H12; H21 H22]; % inertia matrix

H_hat_u = H_hat*u;
```

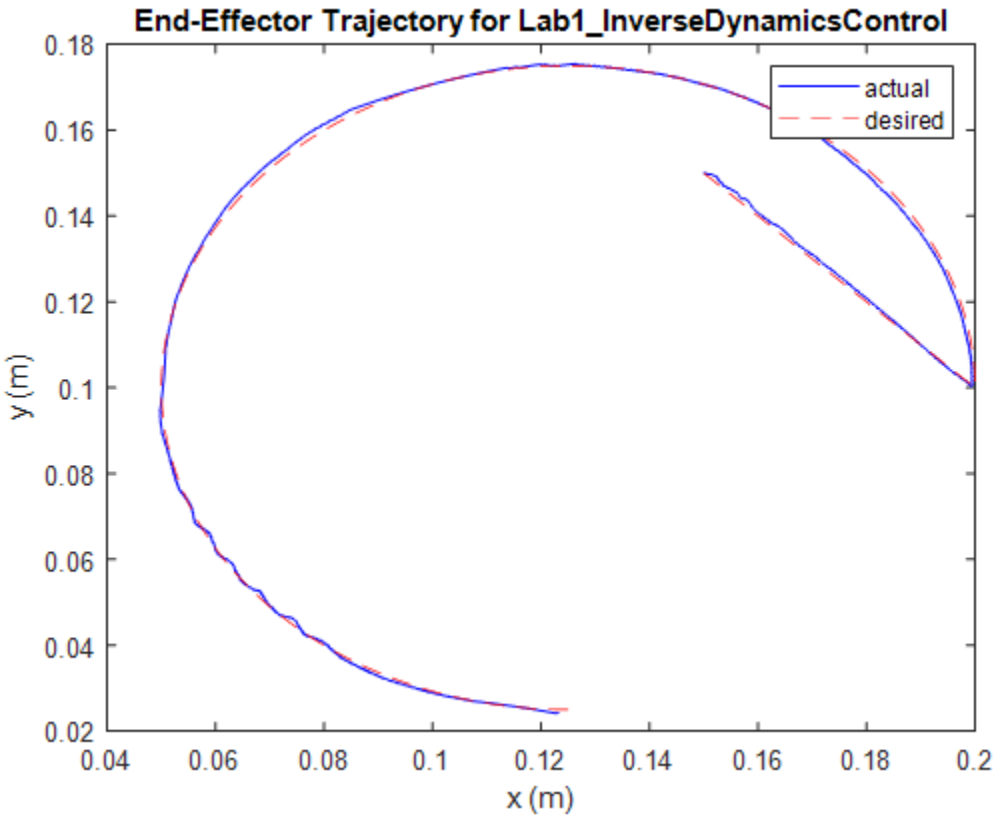
Code for NonlinearTermsModel Block:

```
function N_hat = fcn(phi,phidot)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

a1 = 0.15;  % link 1 length
a2 = 0.15;  % link 2 length
m1 = 0.092;  % link 1 mass
m2 = 0.077; % link 2 mas
r01 = 0.062; % link 1 center of mass
r12 = 0.036; % link 2 COM
I1 = 0.64e-3;  % link 1 inertia
I2 = 0.30e-3;  % link 2 inertia
Jm1 = 0.65e-6; % motor inertias
Jm2 = 0.65e-6;
b1 = 3.1e-6; % viscous damping constants
b2 = 3.1e-6;
c1 = 0.0001; % coulomb friction constants
c2 = 0.0001;
g = 9.8; % gravitational constant
N1 = 70; % gear ratios
N2 = 70;

h = a1*r12*m2*sin(phi(2)-phi(1));
G1 = (r01*m1+a1*m2)*g*cos(phi(1));
G2 = r12*m2*g*cos(phi(2));
F1 = N1^2*b1*phidot(1) + N1*c1*sign(phidot(1));
F2 = N2^2*b2*phidot(2) + N2*c2*sign(phidot(2));

V_hat = [0 -h ;h 0]*[phidot(1)^2;phidot(2)^2]; % centripetal torques
G_hat = [G1;G2]; % gravity torques
F_hat = [F1;F2]; % frictional torques

N_hat = V_hat + G_hat + F_hat;
```

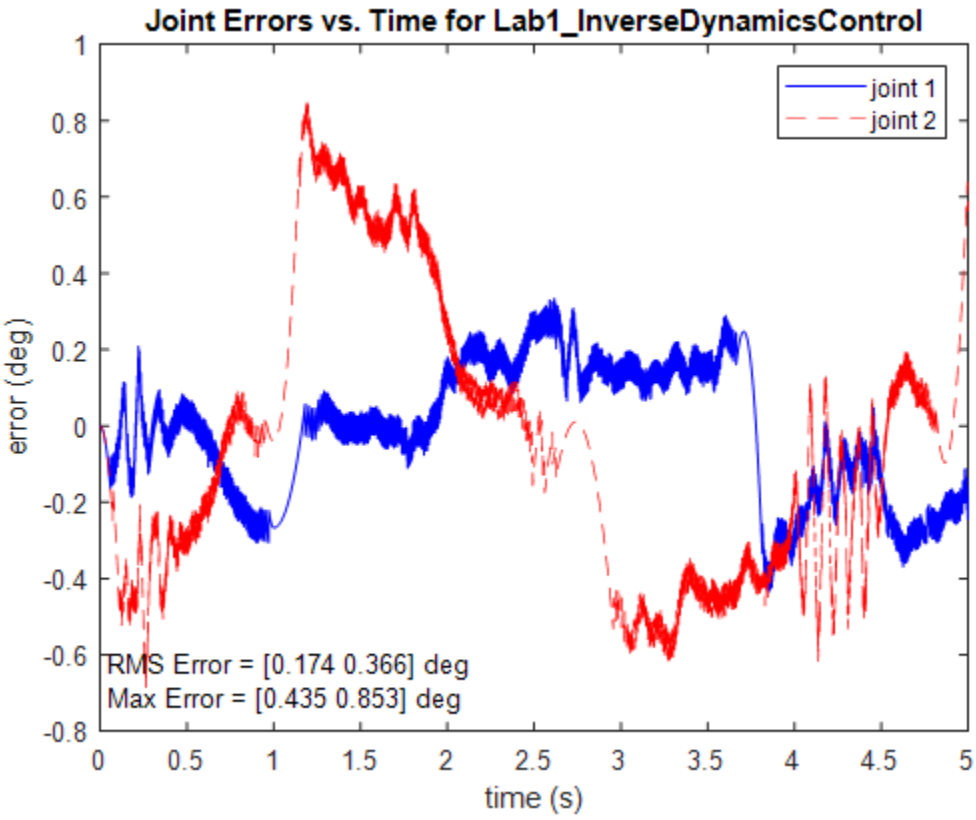
2.2. Inverse Dynamics Controller Simulation

2.2.1. Low Speed Simulation (f=0.2 circles/s)

2.2.1.1. X-y trajectory

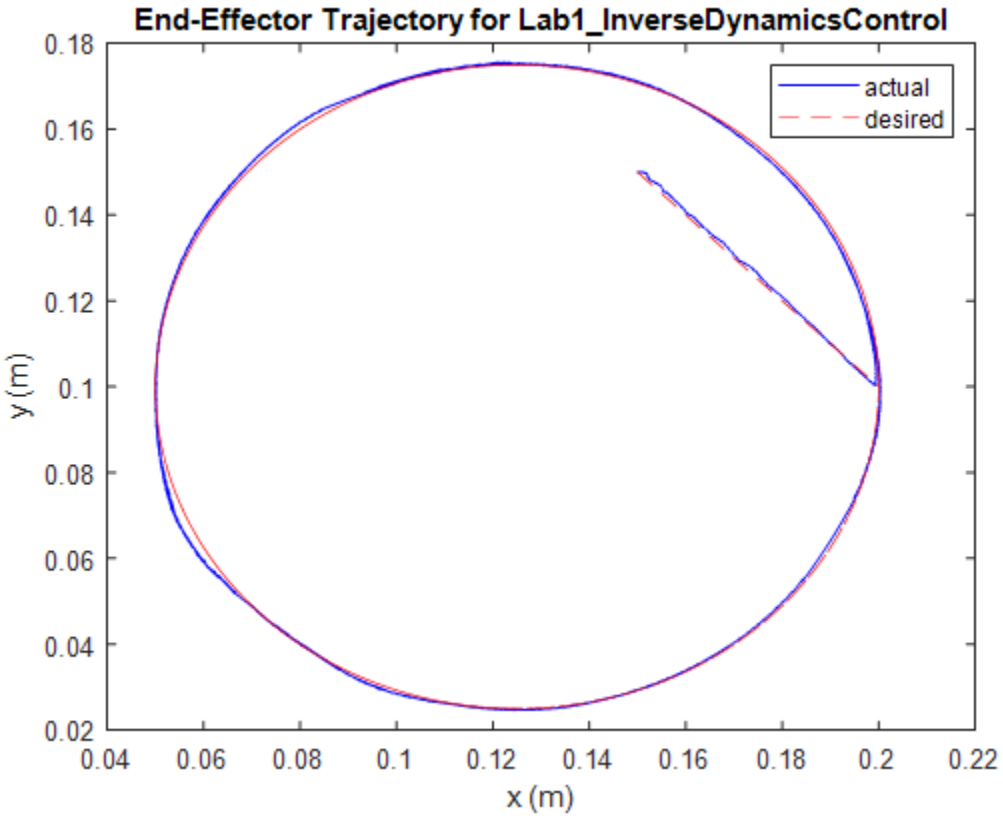


2.2.1.2. Joint angle errors

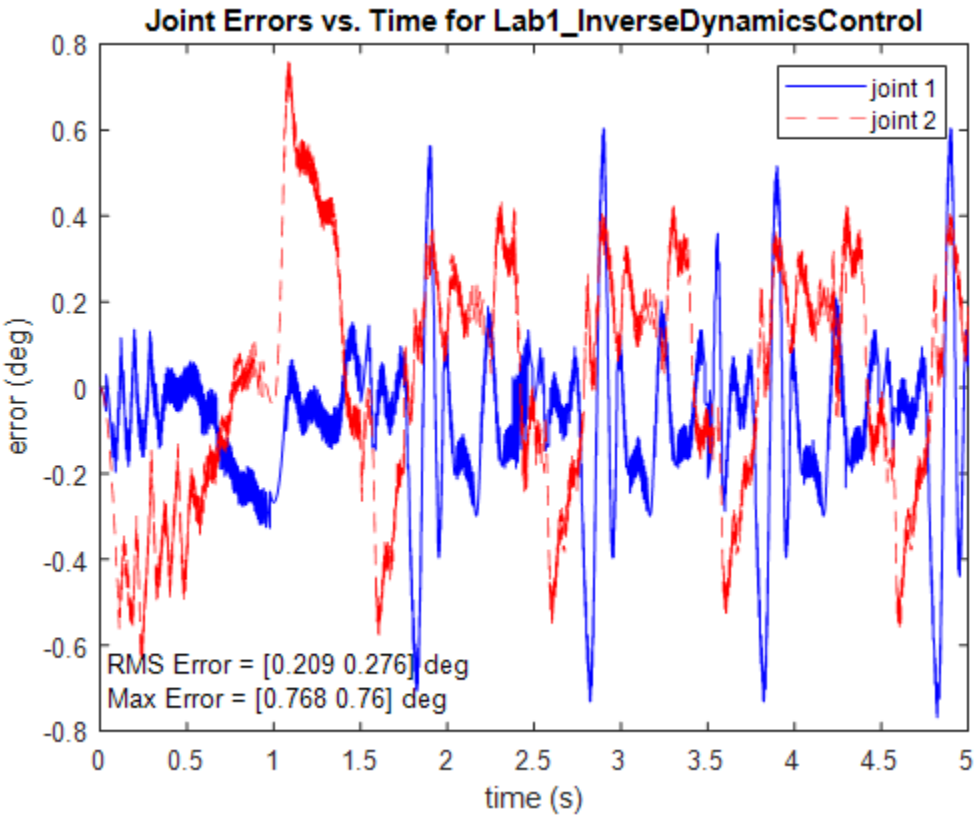


2.2.2. High Speed Simulation (f=1 circles/s)

2.2.2.1. X-y trajectory

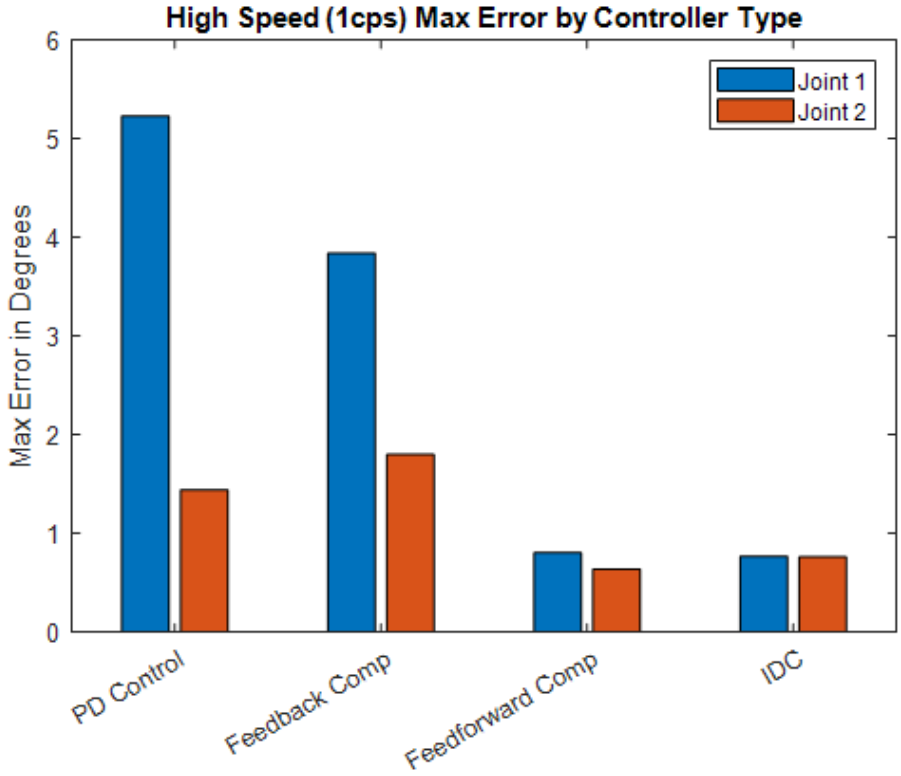


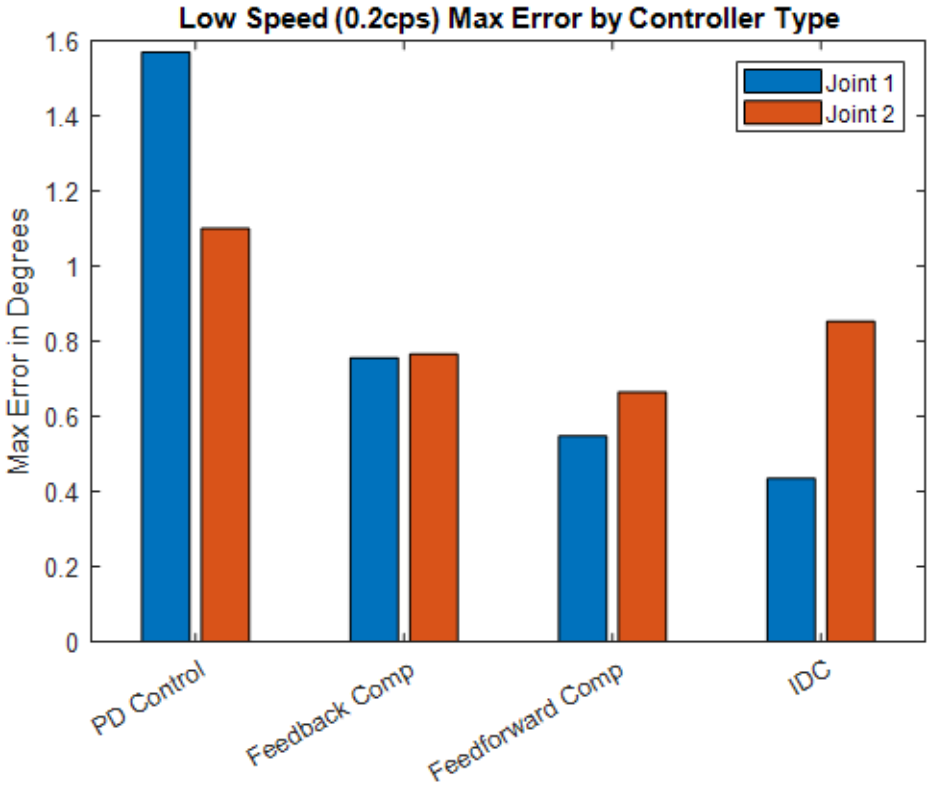
2.2.2.2. Joint angle errors



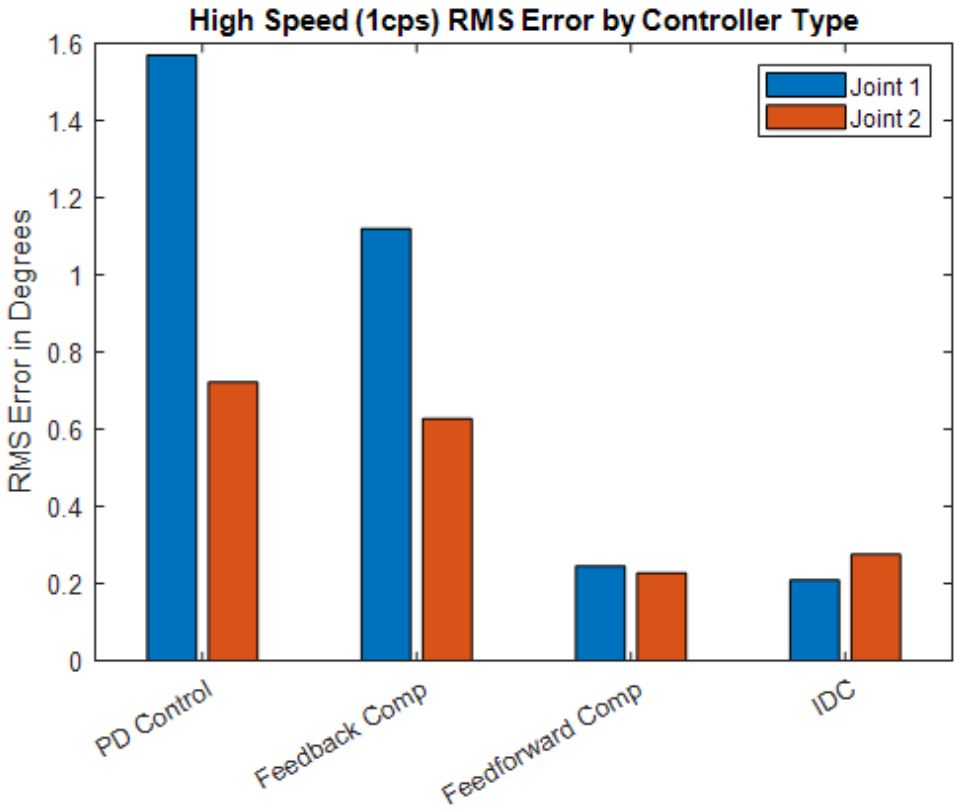
3. Controller Comparison

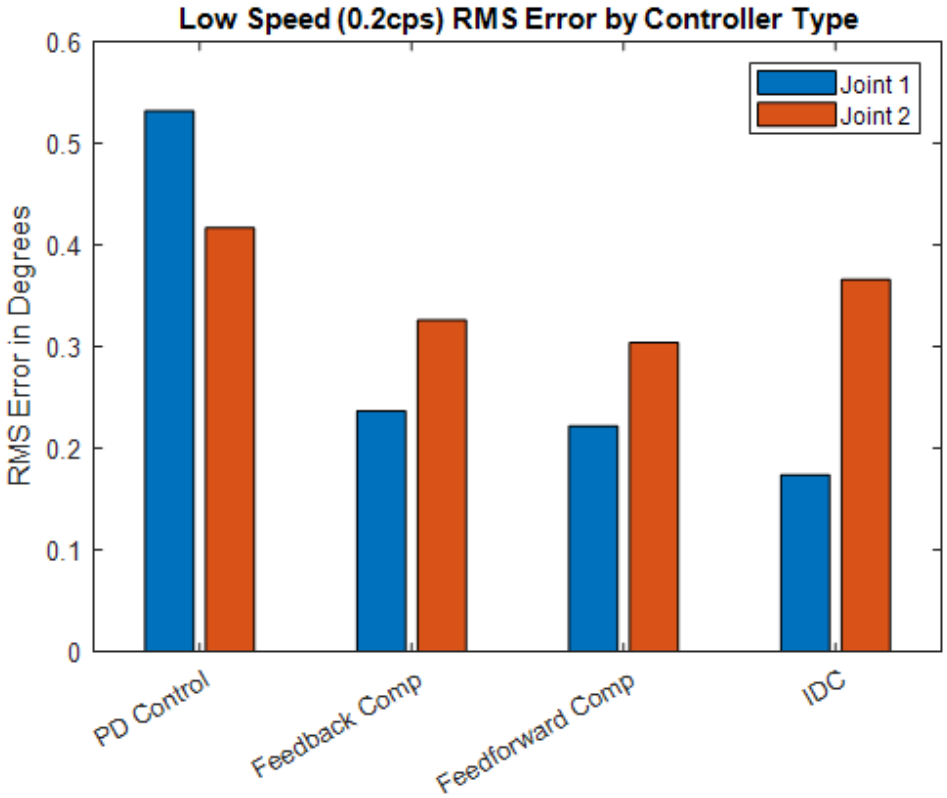
3.1. Peak joint errors





3.2. Root-Mean-Square joint errors





3.3. Comparison

As can be seen above for joint 1 the controllers followed the expected trends of IDC having the least RMS and max error then feedforward compensation, feedback compensation, and lastly with the most RMS and max error the PD controller (for both low and high speed).
For joint 2 the trends were slightly different:

- For low speed: Feedforward compensation had the lowest RMS and max error then feedback compensation, IDC, and lastly with the most RMS and max error the PD controller.
- For high speed: Feedforward compensation had the lowest RMS and max error then IDC, feedback compensation, and lastly with the most RMS and max error the PD controller.

For joint 2 it is likely that the trends for controller performance did not follow the trends of the simulation and joint 1 due to the noise of actual joint readings used in feedback compensation becoming dominant. The errors of joint 2 were likely dominated by sensor-related error while the error of joint 1 was likely dominated by uncertainties which outweigh the sensor-related errors since joint 1 has a more difficult task of tracking its trajectory since it supports the mass and inertia of both link 1 and link 2. Since the feedforward compensation is not affected by the sensor noise of the encoder the error was less pronounced for the simpler task of joint 2 tracking its joint trajectory.
It is also very possible that this difference is because of inherent uncertainties in the real-world. Unlike simulation, each trajectory tracking run cannot be exactly replicated and thus there are variations between each run.

3.4. Comparison Code

```
%% ME EN 6230 Lab 1 Ryan Dalby
% Controller Comparison
close all;
set(groot, 'DefaultTextInterpreter', 'none') % Prevents underscore from
becoming subscript

% errors are in degrees
% PD Controller
m11_slow_rms = [0.532 0.417];
m11_slow_max = [1.57 1.1];
m11_fast_rms = [1.57 0.722];
m11_fast_max = [5.23 1.44];

% PD feedback comp
m12_slow_rms = [0.237 0.326];
m12_slow_max = [0.756 0.766];
m12_fast_rms = [1.12 0.627];
m12_fast_max = [3.84 1.8];

% PD feedforward comp
m13_slow_rms = [0.222 0.304];
m13_slow_max = [0.548 0.665];
m13_fast_rms = [0.246 0.228];
m13_fast_max = [0.805 0.638];

% IDC
m21_slow_rms = [0.174 0.366];
m21_slow_max = [0.435 0.853];h
m21_fast_rms = [0.209 0.276];
m21_fast_max = [0.768 0.760];

slow_rms = [m11_slow_rms; m12_slow_rms; m13_slow_rms; m21_slow_rms];
fast_rms = [m11_fast_rms; m12_fast_rms; m13_fast_rms; m21_fast_rms];
slow_max = [m11_slow_max; m12_slow_max; m13_slow_max; m21_slow_max];
fast_max = [m11_fast_max; m12_fast_max; m13_fast_max; m21_fast_max];

controller_labels = {'PD Control', 'Feedback Comp', 'Feedforward Comp',
'IDC'};
controller_labels_cat = categorical(controller_labels);
controller_labels_cat = reordercats(controller_labels_cat,
string(controller_labels_cat));
```

```
figure;
bar(controller_labels_cat,slow_rms);
title('Low Speed (0.2cps) RMS Error by Controller Type');
legend('Joint 1', 'Joint 2');
ylabel('RMS Error in Degrees');

figure;
bar(controller_labels_cat,fast_rms);
title('High Speed (1cps) RMS Error by Controller Type');
legend('Joint 1', 'Joint 2');
ylabel('RMS Error in Degrees');

figure;
bar(controller_labels_cat,slow_max);
title('Low Speed (0.2cps) Max Error by Controller Type');
legend('Joint 1', 'Joint 2');
ylabel('Max Error in Degrees');

figure;
bar(controller_labels_cat,fast_max);
title('High Speed (1cps) Max Error by Controller Type');
legend('Joint 1', 'Joint 2');
ylabel('Max Error in Degrees');
```

Appendix- Plotting Code (Used to make simulation plots)

```
%% ME EN 6230    Lab 1 Plot Joint Error and Trajectory    Ryan Dalby
set(groot, "DefaultTextInterpreter", "none") % Prevents underscore from
becoming subscript

% Extract necessary data, will error if the data does not exist
time = errors.time; % s
time_datapoints = length(time);
model_title = extractBefore(errors.blockName, "/errors");
joint_errors = rad2deg(transpose(reshape(errors.signals.values, [2,
time_datapoints]))); % deg
actual_trajectory = xy; % m
desired_trajectory = xy_d; % m

% RMS Joint Errors
RMS_error = rms(joint_errors);
% Max Joint Errors
max_error = max(abs(joint_errors));

% Plot Joint Errors vs Time
figure;
plot(time, joint_errors(:,1), "b-");
hold on;
plot(time, joint_errors(:,2), "r--");
hold on;
rms_string = mat2str(RMS_error,3);
text(0.01,0.10,strcat("RMS Error = ", rms_string," deg"), "Units",
"normalized");
hold on;
max_string = mat2str(max_error,3);
text(0.01,0.05,strcat("Max Error = ", max_string," deg"), "Units",
"normalized");
title(strcat("Joint Errors vs. Time for ", model_title));
xlabel("time (s)");
ylabel("error (deg)");
legend("joint 1", "joint 2");

% Plot End-Effector Trajectory
figure;
plot(xy(:,1), xy(:,2), "b-");
hold on;
plot(xy_d(:,1), xy_d(:,2), "r--");
title(strcat("End-Effector Trajectory for ", model_title));
```

```
xlabel("x (m)");
ylabel("y (m)");
legend("actual", "desired");
```