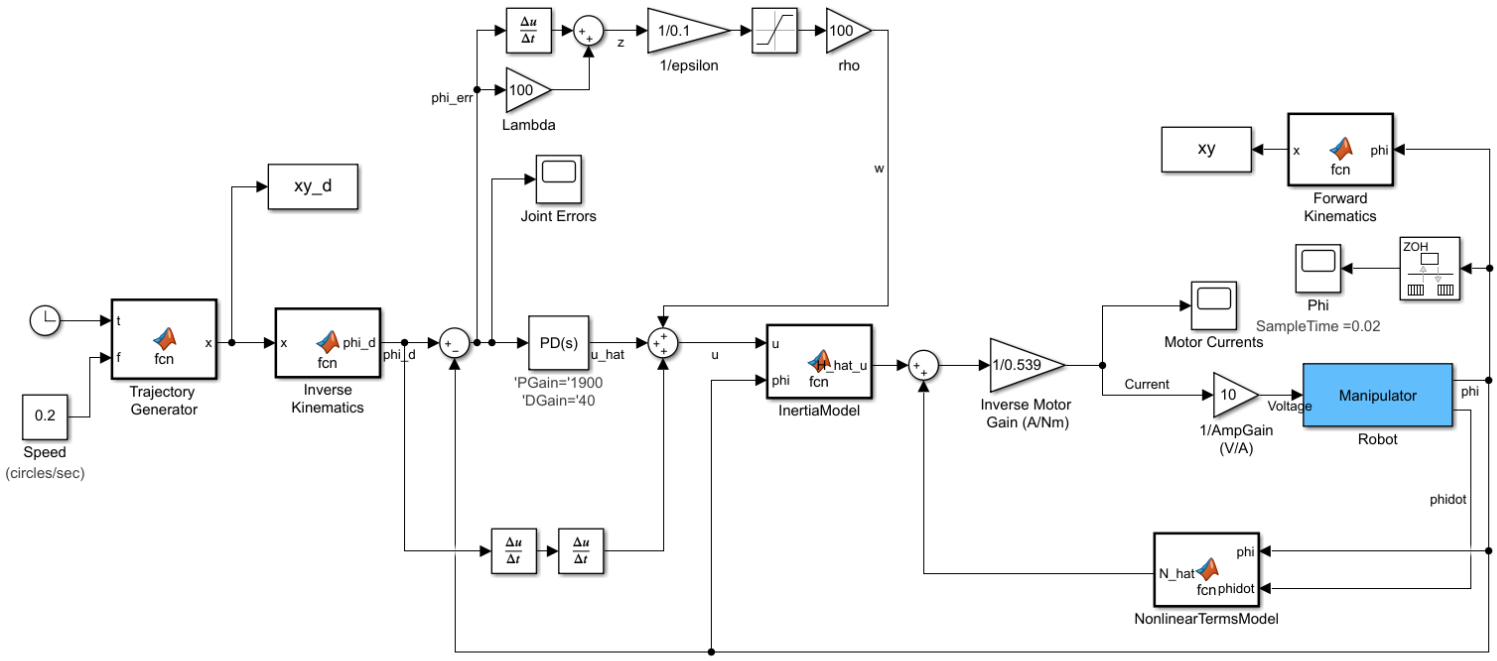
ME EN 6230
Problem Set 7
Ryan Dalby

1. Sliding Mode Control

1.1. Model



1.2. Code

Code for InertiaModel Block:

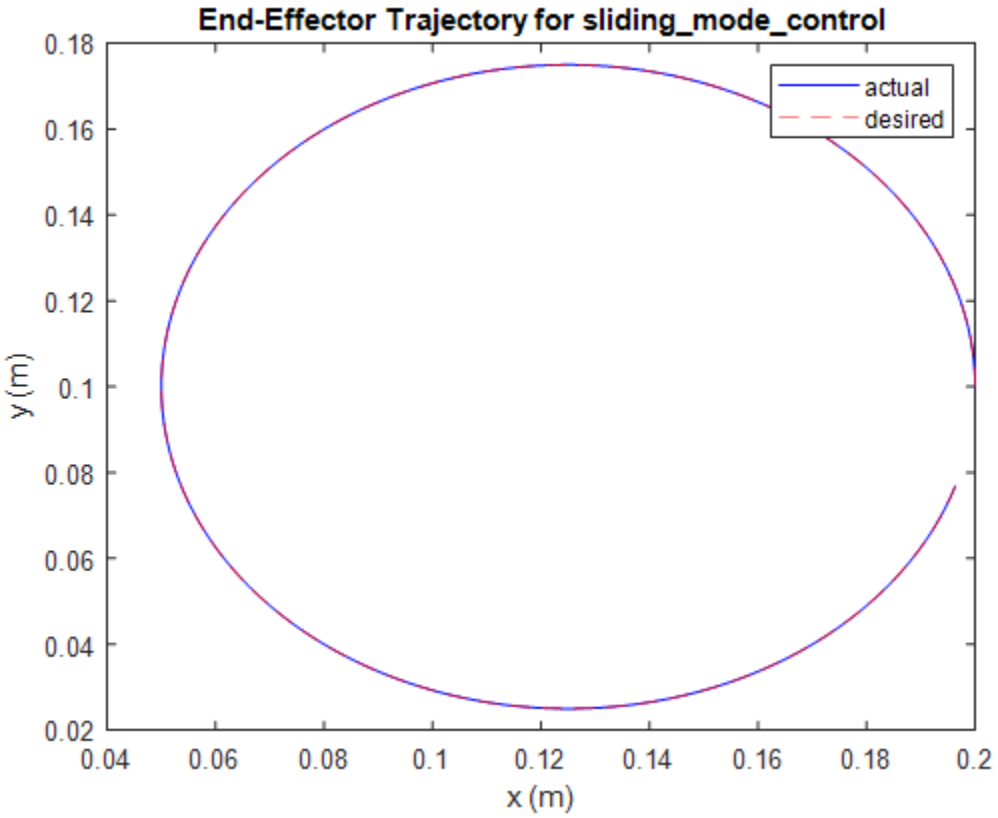
```
function H_hat_u = fcn(u,phi)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

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:

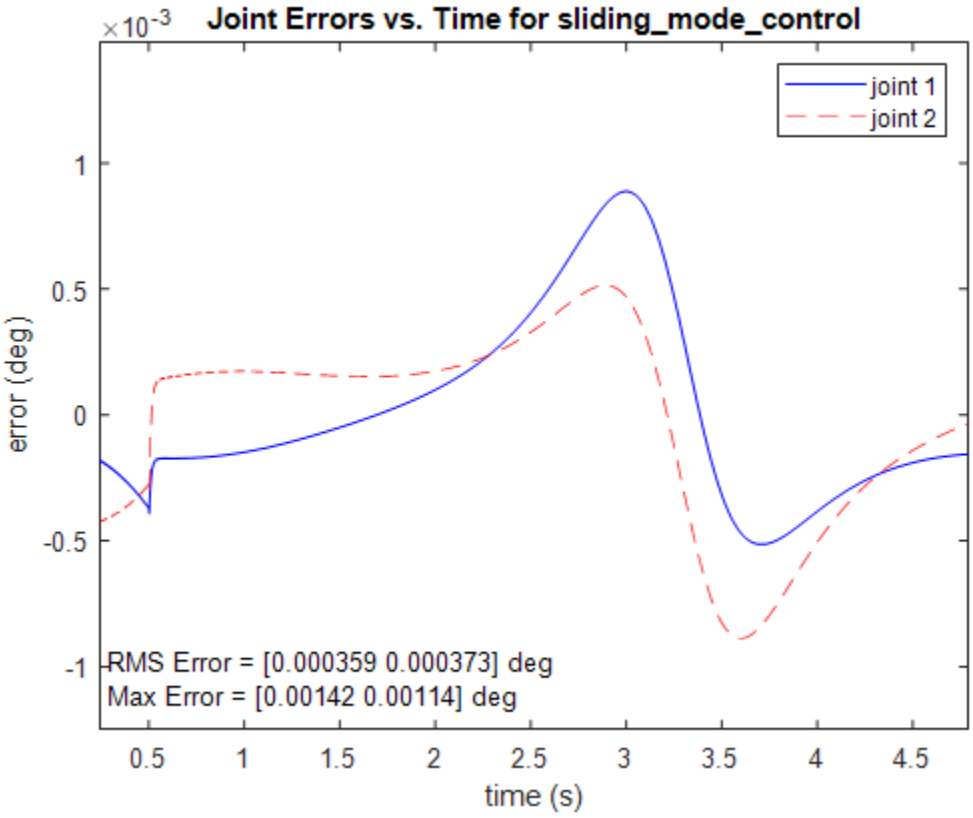
```
function N_hat = fcn(phi,phidot)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

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;
```

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

1.3.1. X-y trajectory

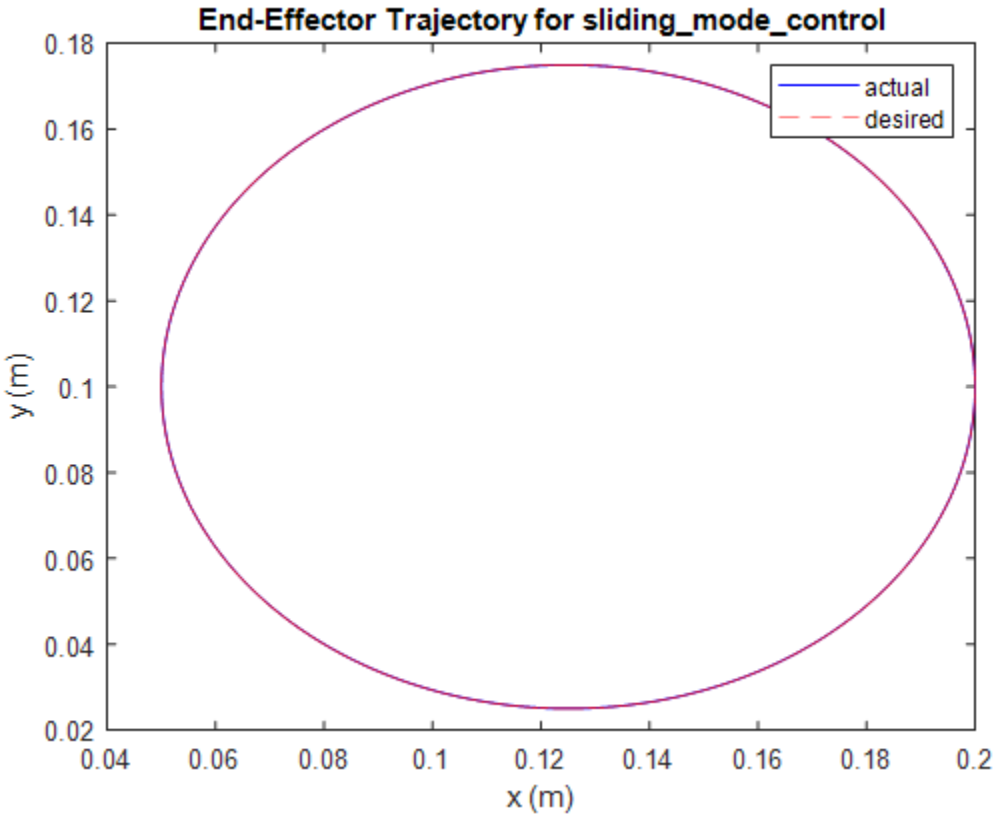


1.3.2. Joint angle errors

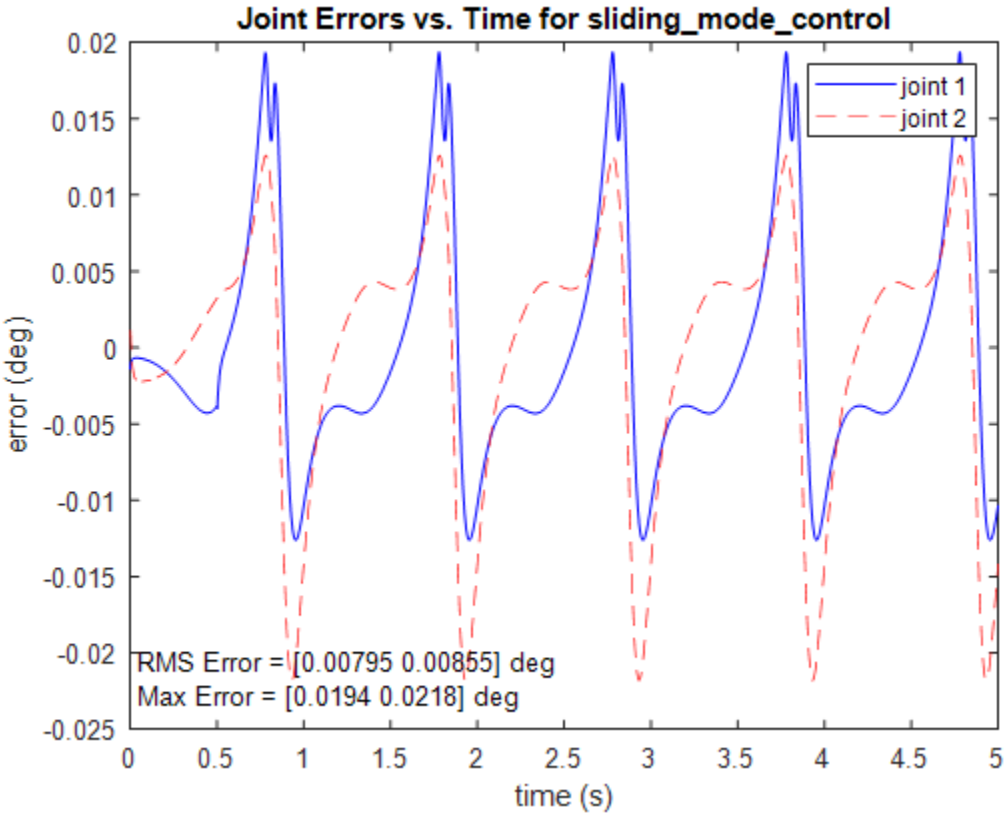


1.4. High Speed Simulation (f=1 circles/s)

1.4.1. X-y trajectory

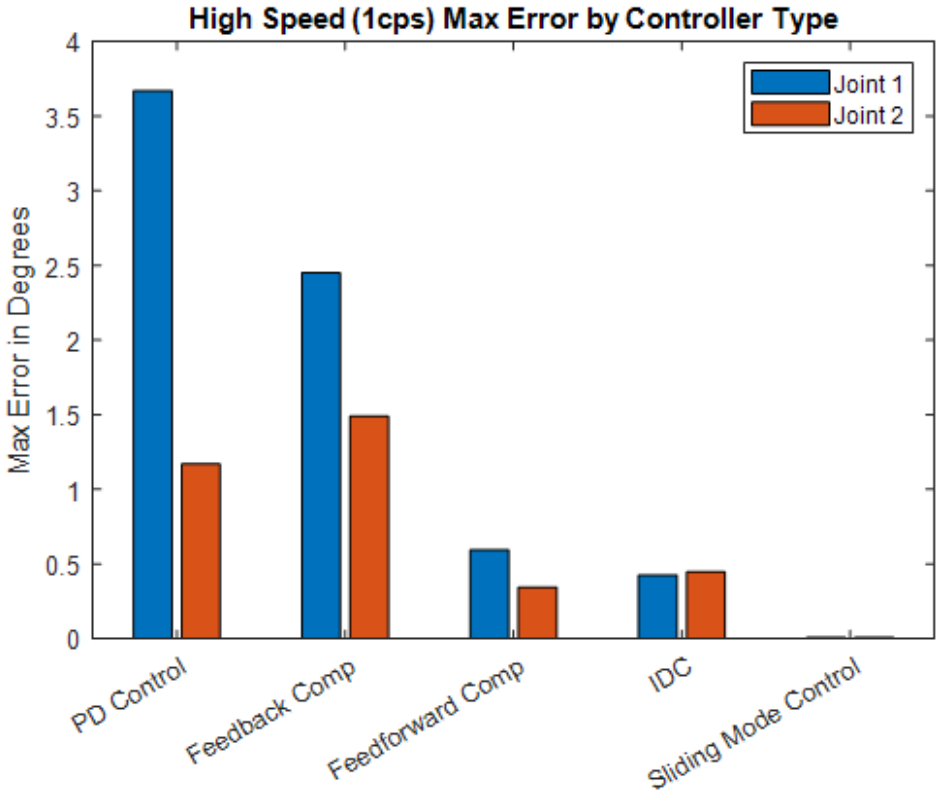


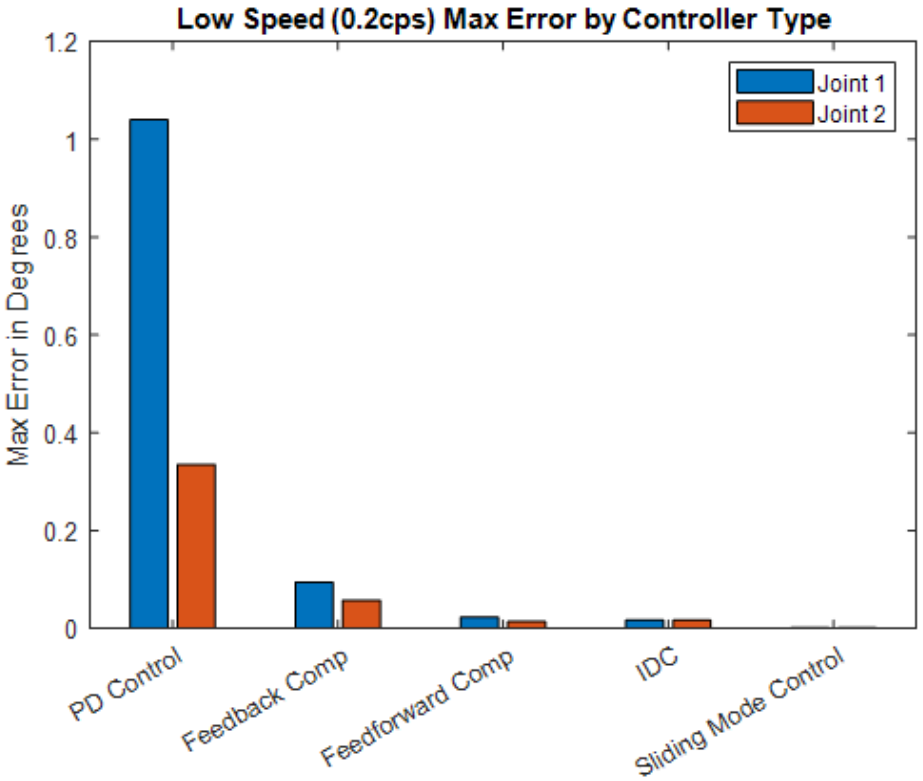
1.4.2. Joint angle errors



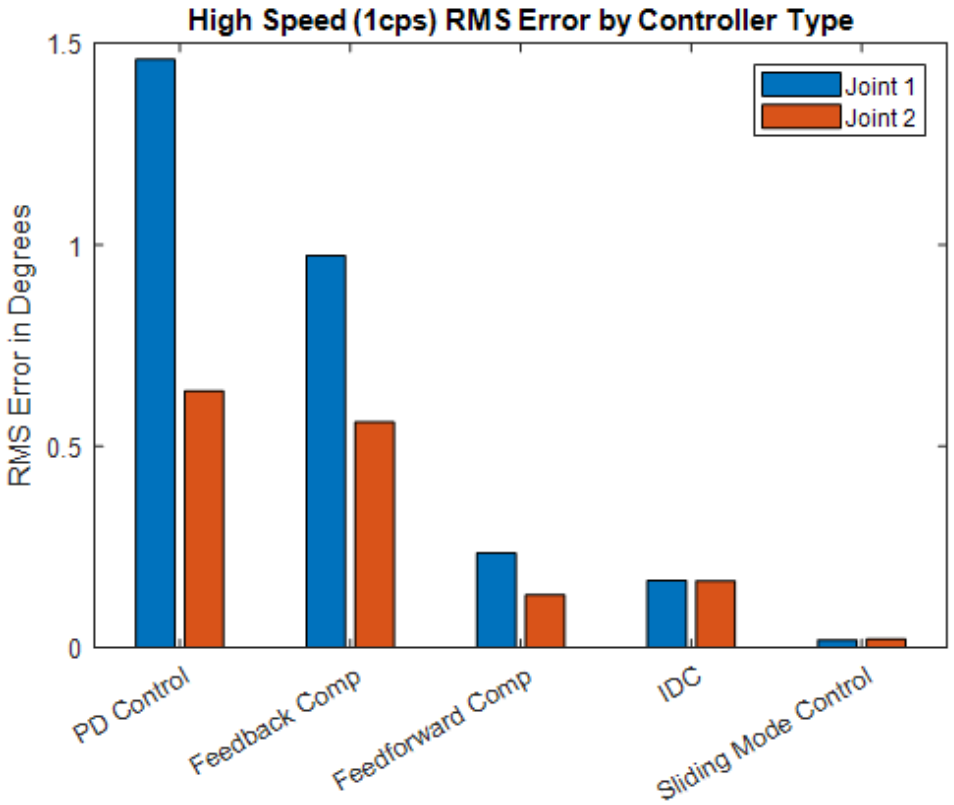
1.5. Controller comparison

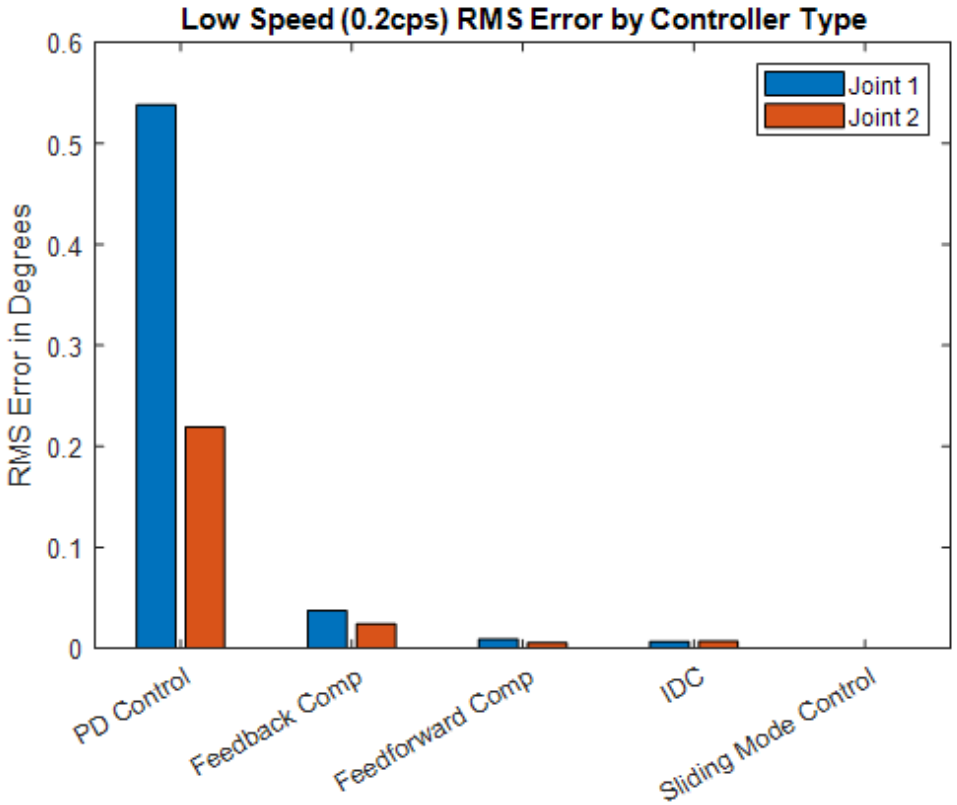
1.5.1. Peak joint errors





1.5.2. Root-Mean-Square joint errors





1.5.3. Comparison

As can be seen in the error plots above comparing the 5 controllers analyzed thus far, sliding mode control is the controller with the least RMS and max error at both low and high speed trajectories. This is because of the controller's ability to robustly deal with model uncertainties which actually deals with the numerical uncertainties that were preventing the inverse dynamics controller from tracking the trajectory perfectly. The least effective controller by both RMS and max error is the PD controller as it doesn't account for the model dynamics at all. The controllers from least error to most error (both RMS and max errors) were the sliding mode controller, inverse dynamics controller, feedforward compensation controller, feedback compensation controller, and lastly the PD controller.

1.5.4. Comparison Code

```
%% ME EN 6230 Problem Set 7 Ryan Dalby
% Controller Comparison
% close all;
set(groot, 'DefaultTextInterpreter', 'none') % Prevents underscore from
becoming subscript

% errors are in degrees
% PD Controller
pd_slow_rms = [0.538 0.219];
pd_slow_max = [1.04 0.335];
pd_fast_rms = [1.46 0.637];
pd_fast_max = [3.67 1.17];

% PD feedback comp
feedbackcomp_slow_rms = [0.0373 0.0241];
feedbackcomp_slow_max = [0.0941 0.0576];
feedbackcomp_fast_rms = [0.973 0.56];
feedbackcomp_fast_max = [2.45 1.49];

% PD feedforward comp
feedforwardcomp_slow_rms = [0.00907 0.00573];
feedforwardcomp_slow_max = [0.0229 0.0137];
feedforwardcomp_fast_rms = [0.235 0.131];
feedforwardcomp_fast_max = [0.595 0.345];

% IDC (Note this was re-run with same PD gain values as sliding control for
% fair comparison)
idc_slow_rms = [0.00668 0.00702];
idc_slow_max = [0.0167 0.0167];
idc_fast_rms = [0.167 0.166];
idc_fast_max = [0.426 0.447];

% Sliding mode control
slidingmodecontrol_slow_rms = [0.0003585,0.0003734];
slidingmodecontrol_slow_max = [0.00142,0.00114];
slidingmodecontrol_fast_rms = [0.0194,0.0218];
slidingmodecontrol_fast_max = [0.00795,0.00855];

slow_rms = [pd_slow_rms; feedbackcomp_slow_rms; feedforwardcomp_slow_rms;
idc_slow_rms; slidingmodecontrol_slow_rms];
fast_rms = [pd_fast_rms; feedbackcomp_fast_rms; feedforwardcomp_fast_rms;
idc_fast_rms; slidingmodecontrol_fast_rms];
```

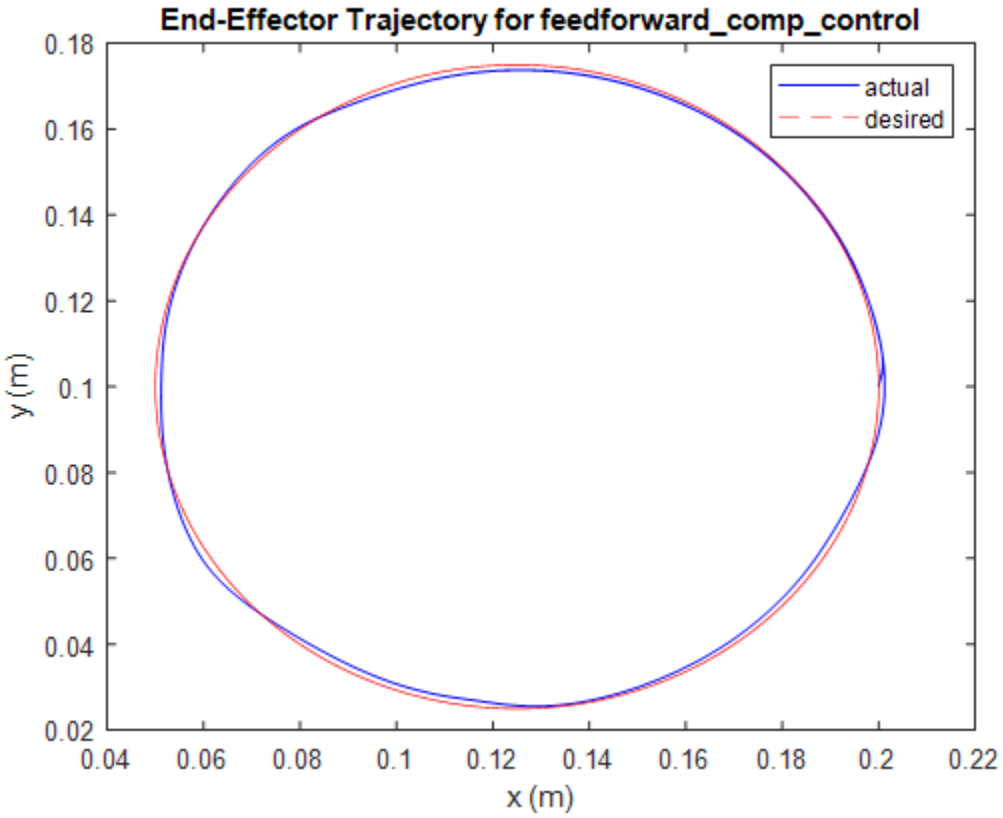
```
slow_max = [pd_slow_max; feedbackcomp_slow_max; feedforwardcomp_slow_max;
idc_slow_max; slidingmodecontrol_slow_max];
fast_max = [pd_fast_max; feedbackcomp_fast_max; feedforwardcomp_fast_max;
idc_fast_max; slidingmodecontrol_fast_max];

controller_labels = {'PD Control', 'Feedback Comp', 'Feedforward Comp',
'IDC', 'Sliding Mode Control'};
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');
```

2. Robustness

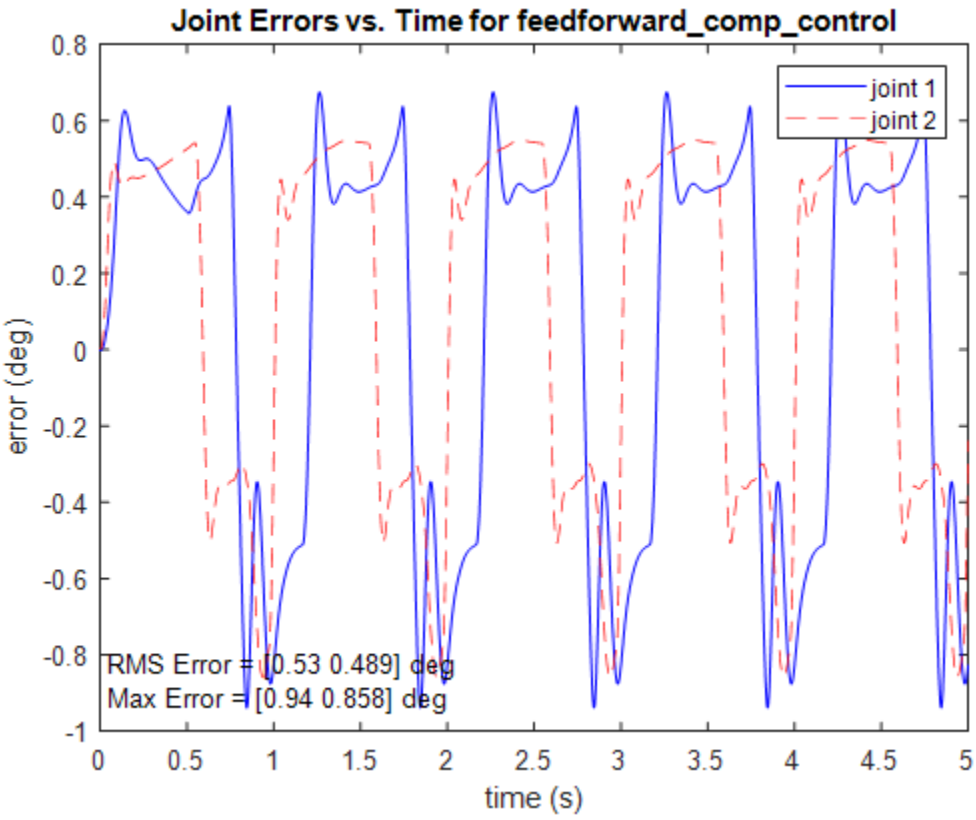
2.1. Feedforward compensation with disturbance

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

2.1.1.1. X-y trajectory



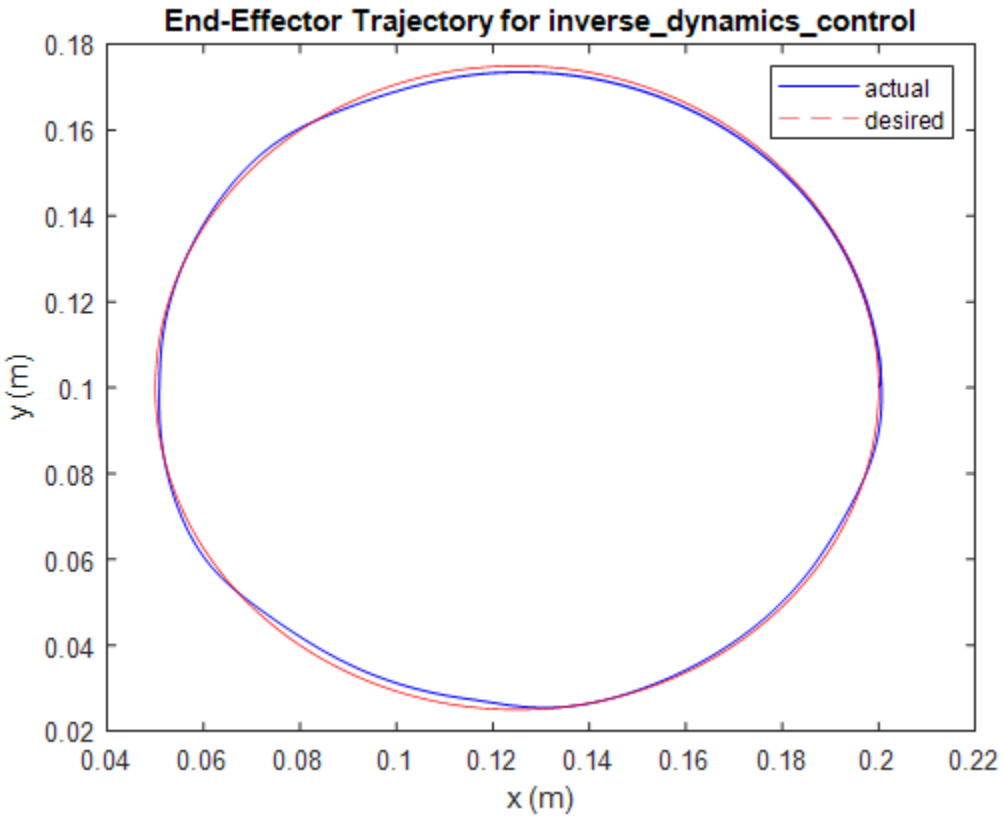
2.1.1.2. Joint angle errors



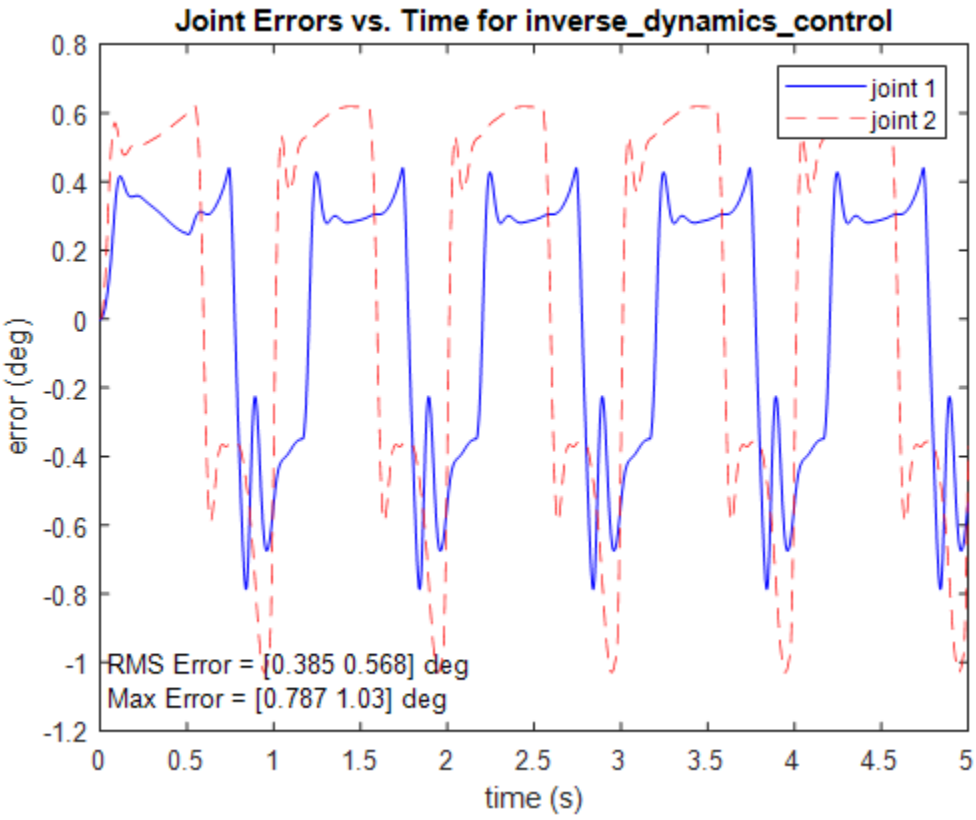
2.2. Inverse dynamics control with disturbance

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

2.2.1.1. X-y trajectory



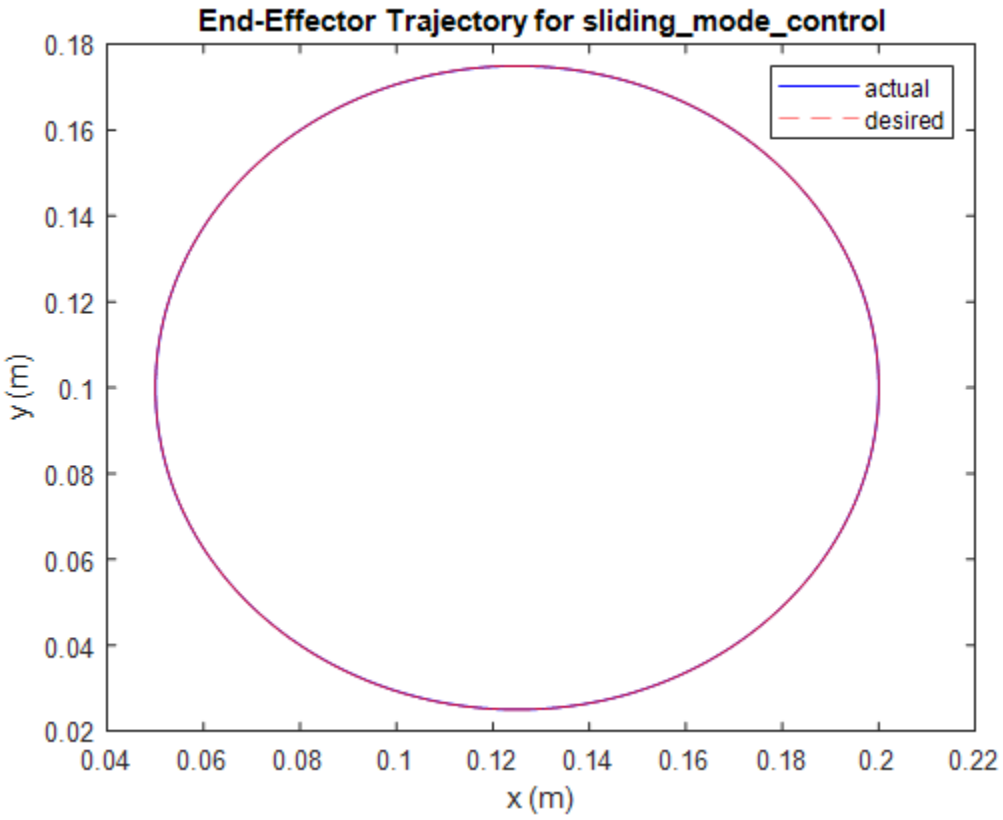
2.2.1.2. Joint angle errors



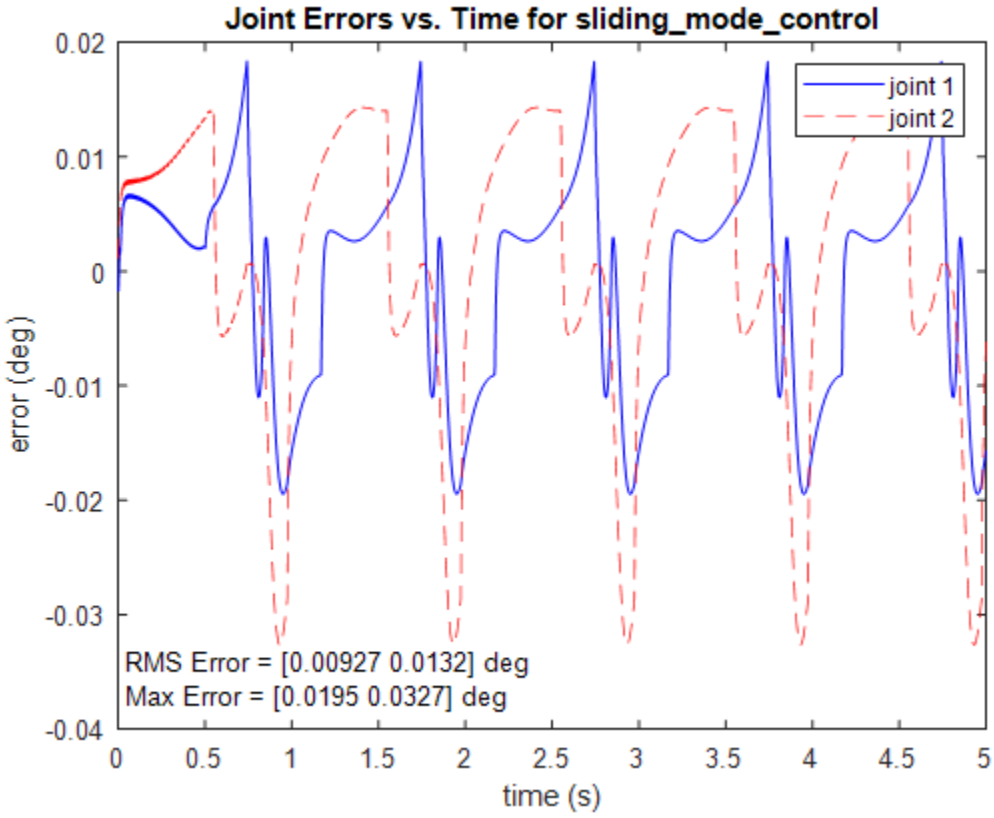
2.3. Sliding mode control with disturbance

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

2.3.1.1. X-y trajectory

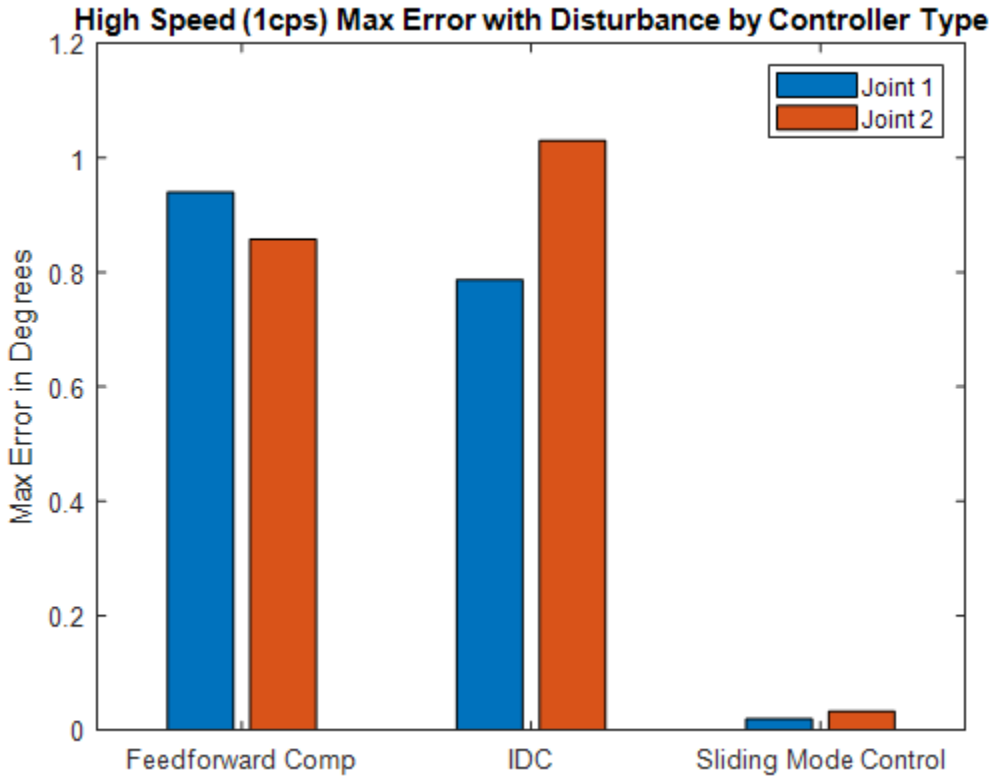


2.3.1.2. Joint angle errors

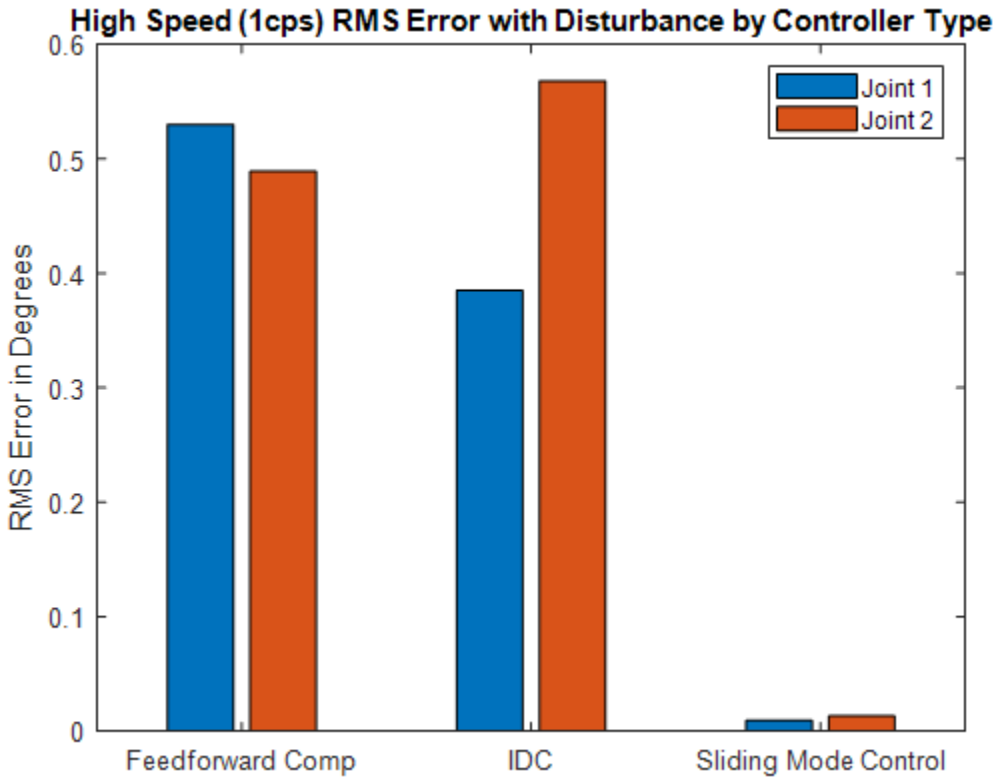


2.4. Controller robustness comparison

2.4.1. Peak joint errors



2.4.2. Root-Mean-Square joint errors



2.4.3. Comparison

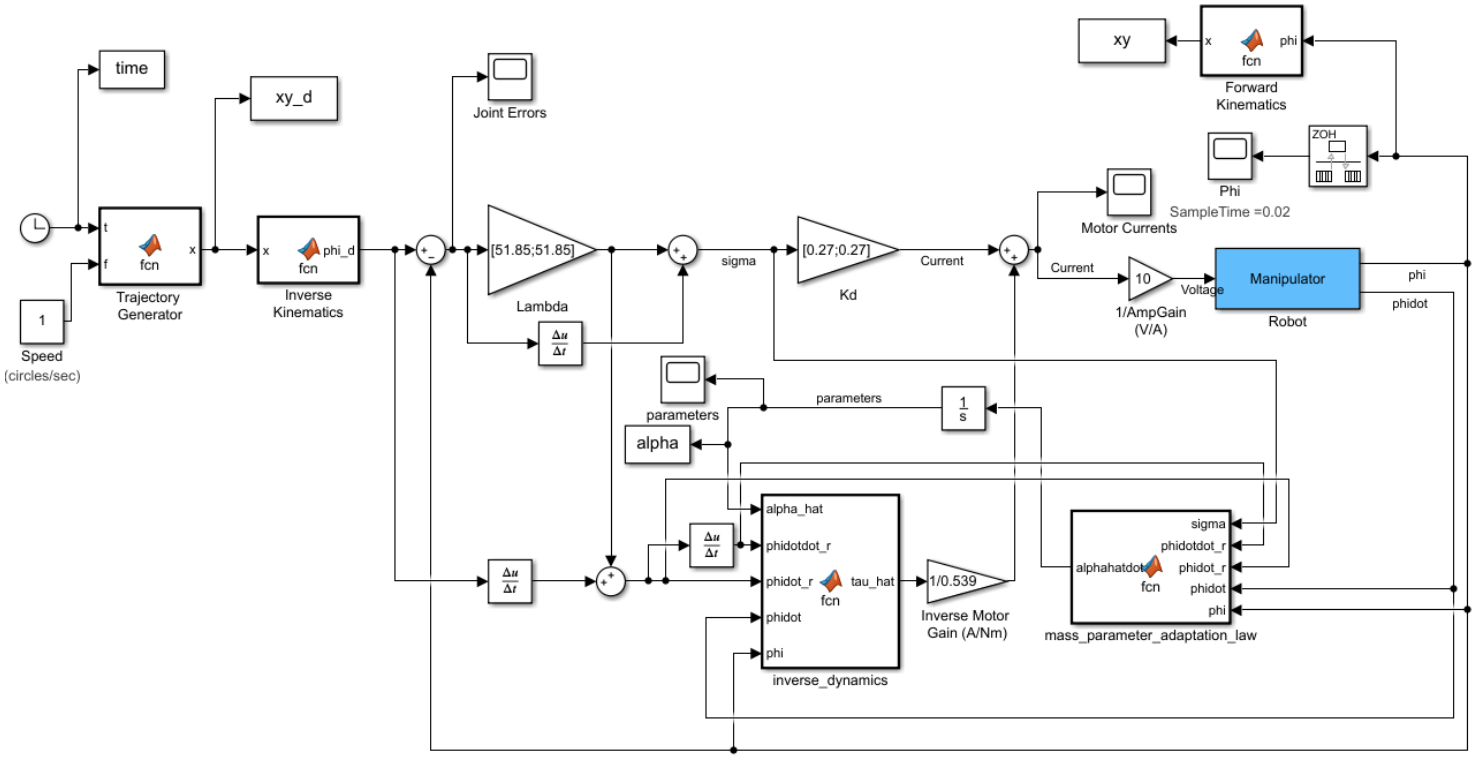
As can be seen in the plots above the sliding mode controller is the best controller in terms of least RMS and max error by a large margin. The inverse dynamics controller and the feedforward compensation controller and comparable to each other in terms of both RMS and max error. The sliding mode controller has the ability to deal with model uncertainty using the sliding surface function w(z) which provides an ability to deal with model uncertainty/disturbances. The other controllers lack this ability and clearly suffer in the presence of model uncertainty/disturbances.

2.4.4. Comparison Code

```
%% ME EN 6230 Problem Set 7 Ryan Dalby
% Robustness Comparison
% close all;
set(groot, 'DefaultTextInterpreter', 'none') % Prevents underscore from
becoming subscript

% errors are in degrees

% PD feedforward comp w/ disturbance
feedforwardcomp_fast_rms = [0.53 0.489];
feedforwardcomp_fast_max = [0.94 0.858];

% IDC w/ disturbance
idc_fast_rms = [0.385 0.568];
idc_fast_max = [0.787 1.03];

% Sliding mode control w/ disturbance
slidingmodecontrol_fast_rms = [0.00927 0.0132];
slidingmodecontrol_fast_max = [0.0195 0.0327];

fast_rms = [feedforwardcomp_fast_rms; idc_fast_rms;
slidingmodecontrol_fast_rms];
fast_max = [feedforwardcomp_fast_max; idc_fast_max;
slidingmodecontrol_fast_max];
controller_labels = {'Feedforward Comp', 'IDC', 'Sliding Mode Control'};
controller_labels_cat = categorical(controller_labels);
controller_labels_cat = reordercats(controller_labels_cat,
string(controller_labels_cat));

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

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

3. Adaptive Control

3.1. Model



Note: A diagonal 4x4 gamma matrix was used with values of 15.

3.2. Code

Code for inverse_dynamics Block:

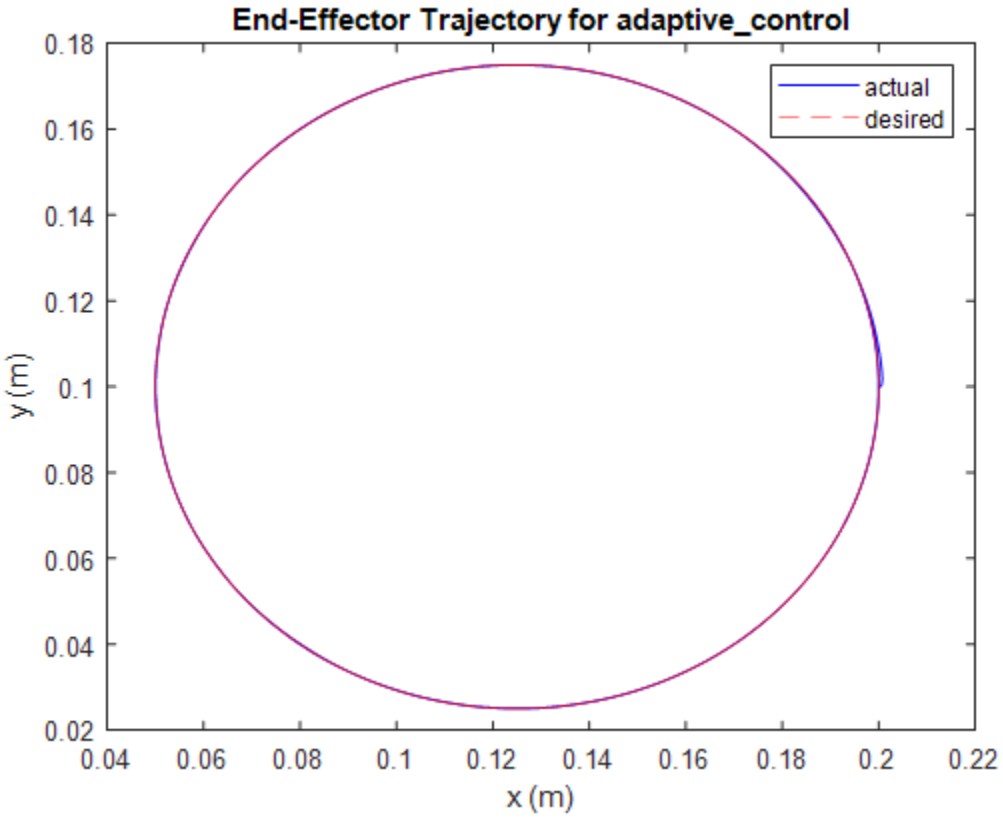
```
function tau_hat = fcn(alpha_hat,phidotdot_r,phidot_r,phidot,phi)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

a1 = 0.15;  % link 1 length
a2 = 0.15;  % link 2 length
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;

F1 = N1^2*b1*phidot(1) + N1*c1*sign(phidot(1));
F2 = N2^2*b2*phidot(2) + N2*c2*sign(phidot(2));

F = [F1;F2]; % frictional torques

Y11 = phidotdot_r(1);
Y12 = a1*cos(phi(2)-phi(1))*phidotdot_r(2) -
a1*sin(phi(2)-phi(1))*phidot(2)*phidot_r(2);
Y13 = g*cos(phi(1));
Y14 = 0;
Y21 = 0;
Y22 = a1*cos(phi(2)-phi(1))*phidotdot_r(1) +
a1*sin(phi(2)-phi(1))*phidot(1)*phidot_r(1) + g*cos(phi(2));
Y23 = 0;
Y24 = phidotdot_r(2);

Y = [Y11 Y12 Y13 Y14; Y21 Y22 Y23 Y24];

tau_hat = Y*alpha_hat + F;
```

Code for mass_parameter_adaptation_law Block:

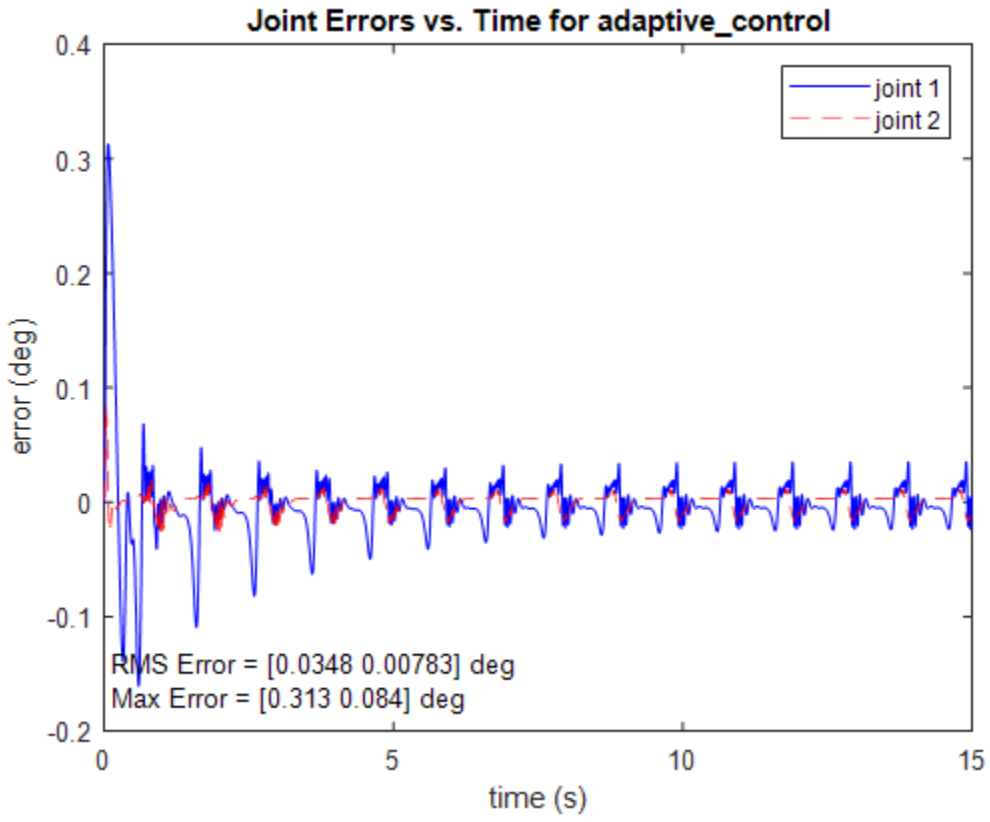
```
function alphahatdot = fcn(sigma,phidotdot_r,phidot_r,phidot,phi)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

a1 = 0.15;  % link 1 length
a2 = 0.15;  % link 2 length

g = 9.8; % gravitational constant

Y11 = phidotdot_r(1);
Y12 = a1*cos(phi(2)-phi(1))*phidotdot_r(2) -
a1*sin(phi(2)-phi(1))*phidot(2)*phidot_r(2);
Y13 = g*cos(phi(1));
Y14 = 0;
Y21 = 0;
Y22 = a1*cos(phi(2)-phi(1))*phidotdot_r(1) +
a1*sin(phi(2)-phi(1))*phidot(1)*phidot_r(1) + g*cos(phi(2));
Y23 = 0;
Y24 = phidotdot_r(2);

Y = [Y11 Y12 Y13 Y14; Y21 Y22 Y23 Y24];

gamma_val = 15;
gamma = [gamma_val 0 0 0; 0 gamma_val 0 0; 0 0 gamma_val 0; 0 0 0
gamma_val];

alphahatdot = inv(gamma) * transpose(Y) * sigma;
```

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

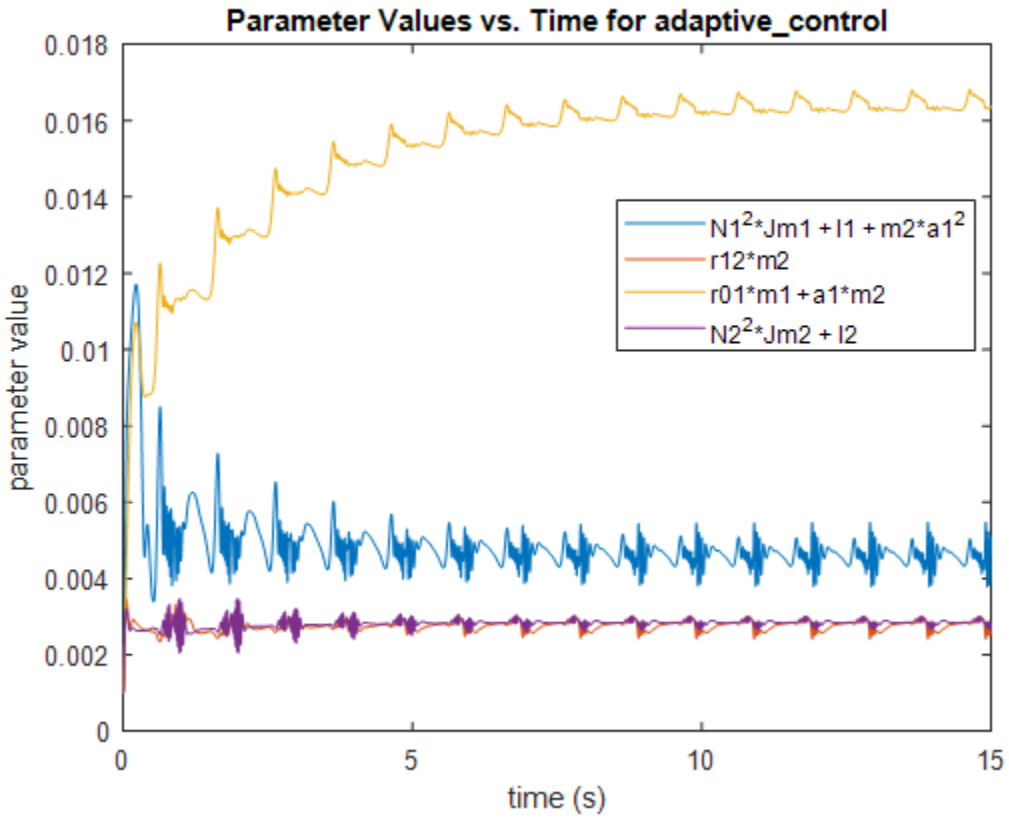
3.3.1. X-y trajectory



3.3.2. Joint angle errors



3.3.3. Parameter estimates vs. time



3.4. Analysis

Does the tracking error converge to zero?

No the tracking error does not exactly converge to zero. It oscillates around zero tracking error as seen in the joint angle errors plot.

Do the mass/inertia parameters converge to their true values?

The true parameter values in vector form are [0.0055575; 0.002772; 0.017254; 0.003485] which does not exactly match the approximate converged values of [0.0045; 0.0027; 0.0162; 0.0027] but is close to the true parameter values. This is because the adaptation law finds values that work well not necessarily the true values.

Appendix- Plotting Code (Used to make simulation plots)

```
%% ME EN 6230 Problem Set 7 Ryan Dalby
% close all;
set(groot, 'DefaultTextInterpreter', 'none') % Prevents underscore from
becoming subscript

% Extract necessary data, will error if the data does not exist
time = errors.time; % s
model_title = extractBefore(errors.blockName, "/Joint Errors");
joint_errors = rad2deg(errors.signals.values); % 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;
text(0.01,0.10,append('RMS Error = ', mat2str(RMS_error,3),' deg'),
'Units', 'normalized');
hold on;
text(0.01,0.05,append('Max Error = ', mat2str(max_error,3),' deg'),
'Units', 'normalized');
title(append('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(append('End-Effector Trajectory for ', model_title));
xlabel('x (m)');
ylabel('y (m)');
legend('actual', 'desired');
```

```
% If model_title is adaptive_control plot parameter adaptation
if strcmp(model_title, 'adaptive_control')
    figure;
    for i = 1:size(alpha, 2)
        plot(time, alpha(:,i));
        hold on;
    end
    title(append('Parameter Values vs. Time for ', model_title))
    xlabel('time (s)');
    ylabel('parameter value');
    legend('N1^2*Jm1 + I1 + m2*a1^2','r12*m2','r01*m1 + a1*m2','N2^2*Jm2 +
I2');
end
```