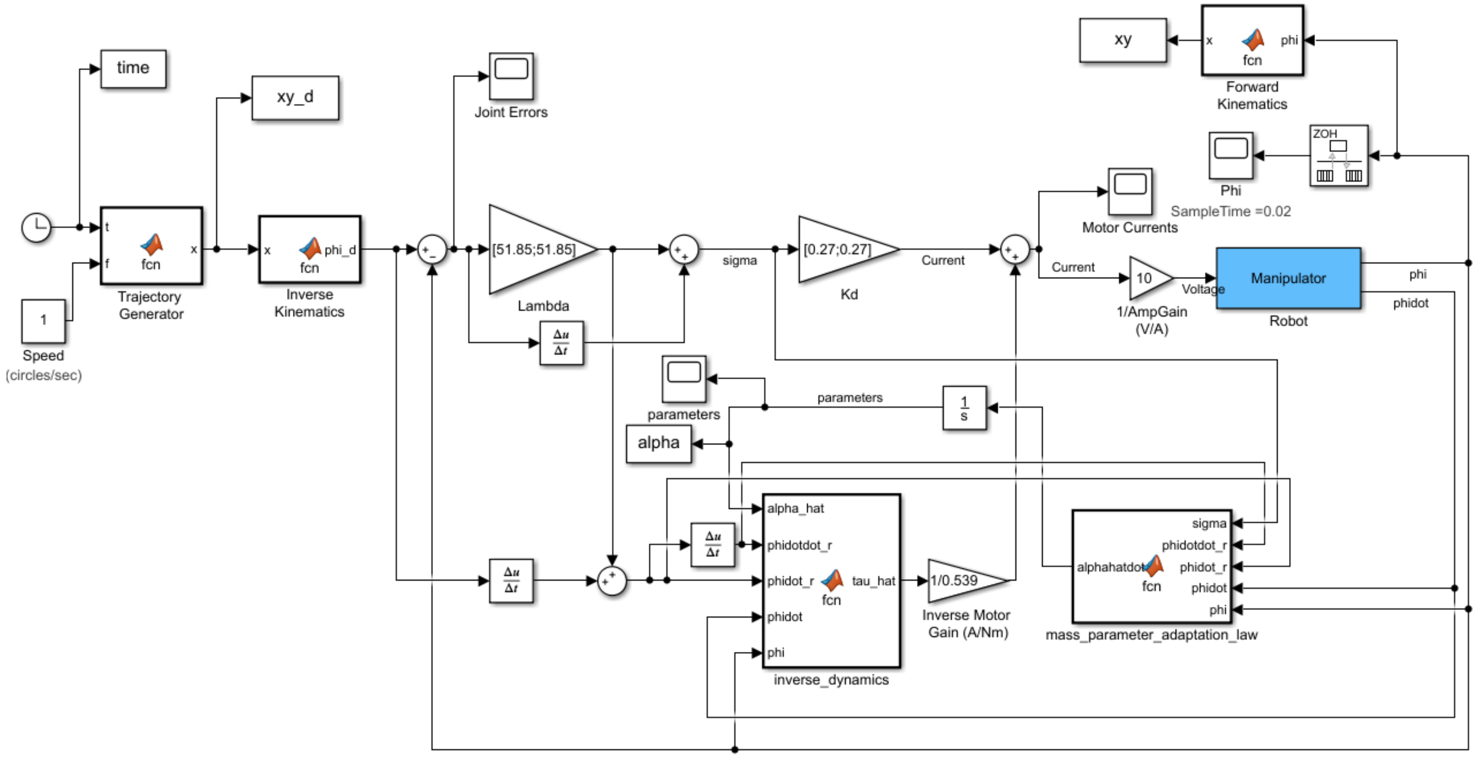
3. Adaptive Control

3.1. Model



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

3.2. Code

Code for inverse_dynamics Block:

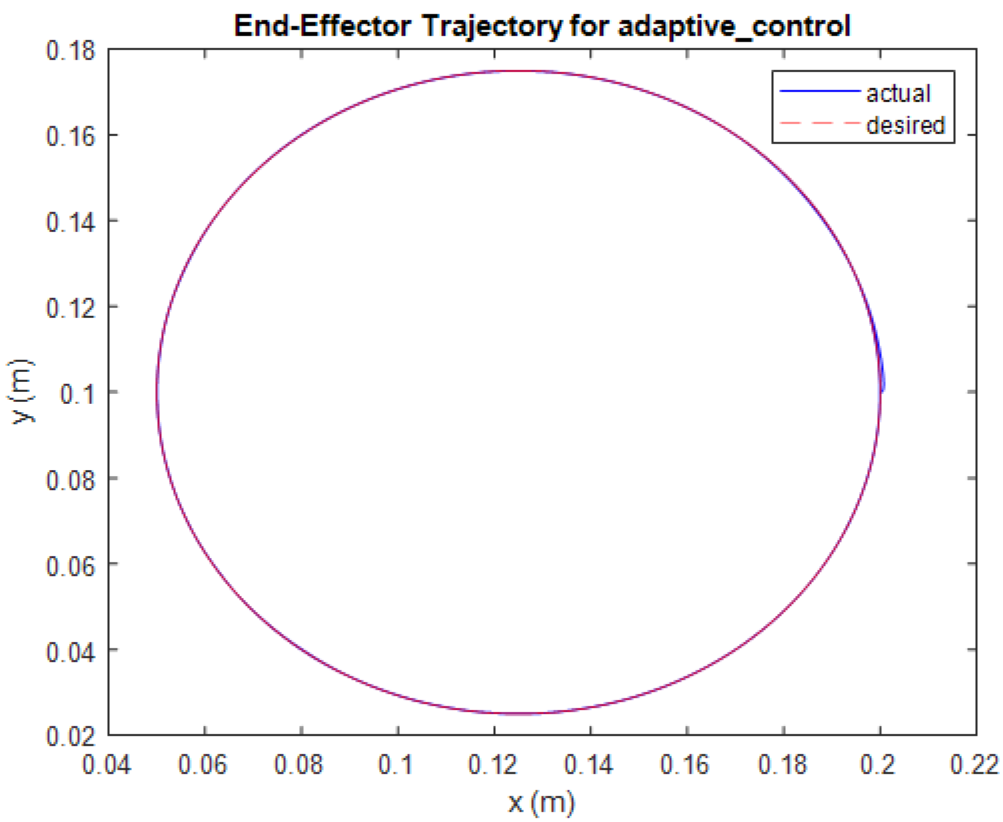
```
function tau_hat = fcn(alpha_hat,phidotdot_r,phidot_r,phidot,phi)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

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:

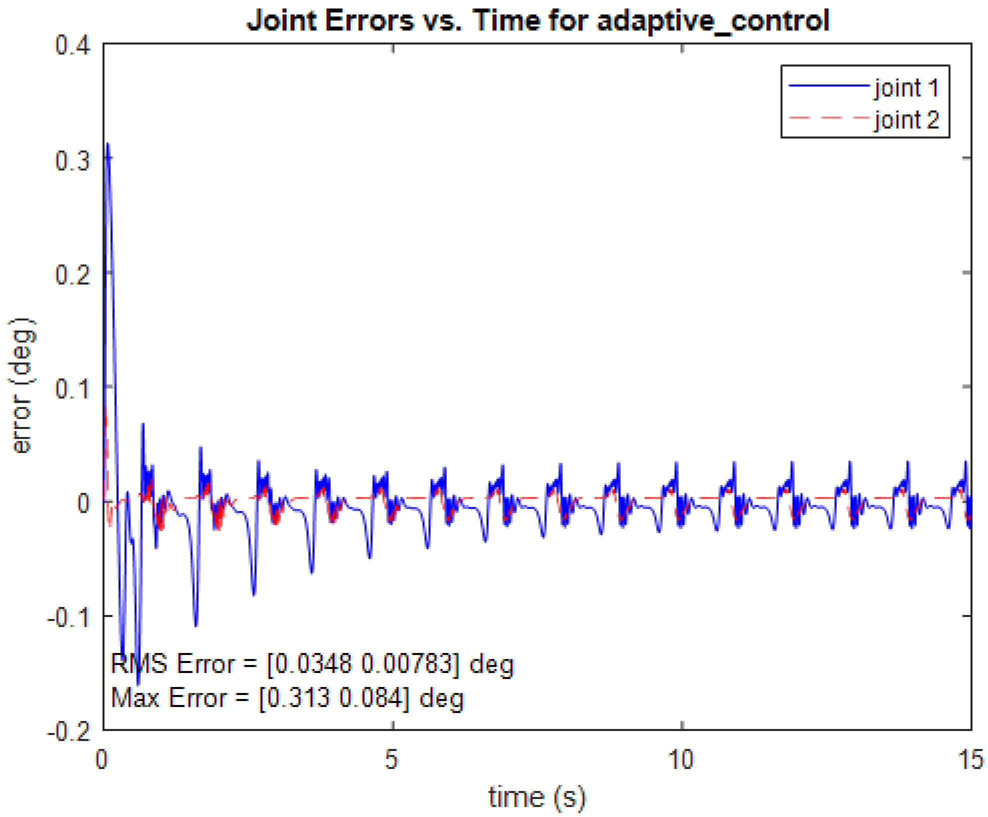
```
function alphahatdot = fcn(sigma,phidotdot_r,phidot_r,phidot,phi)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

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)

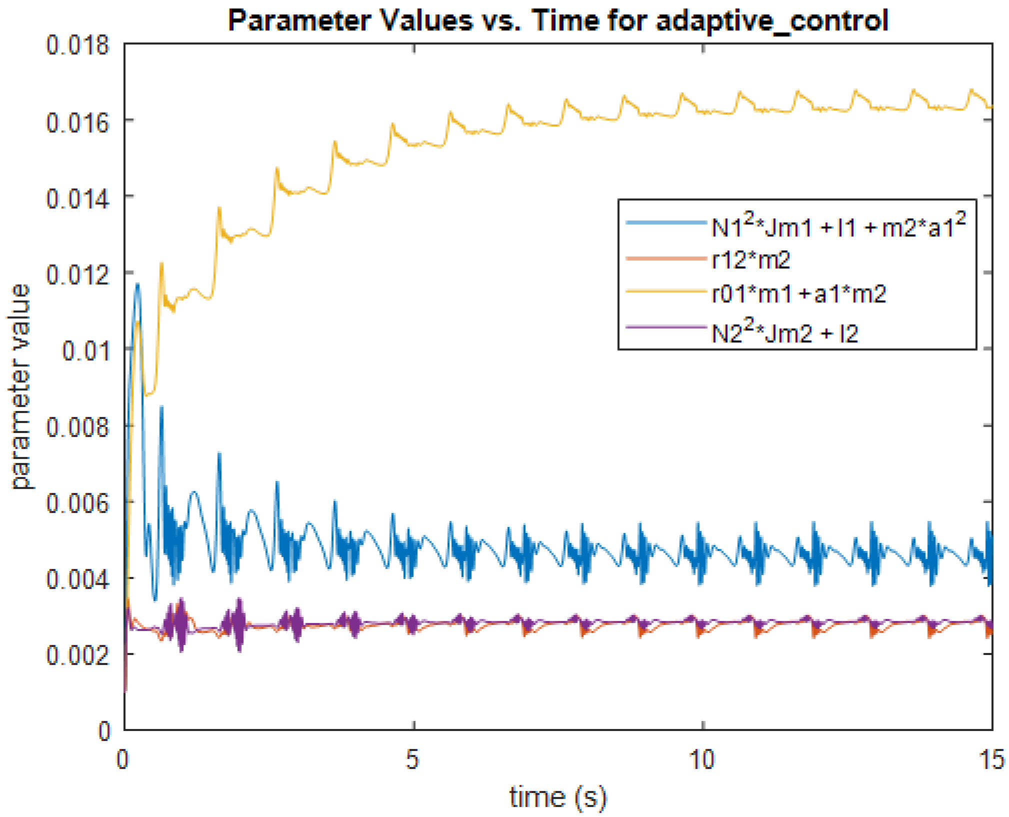
3.3.1. X-y trajectory



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
```