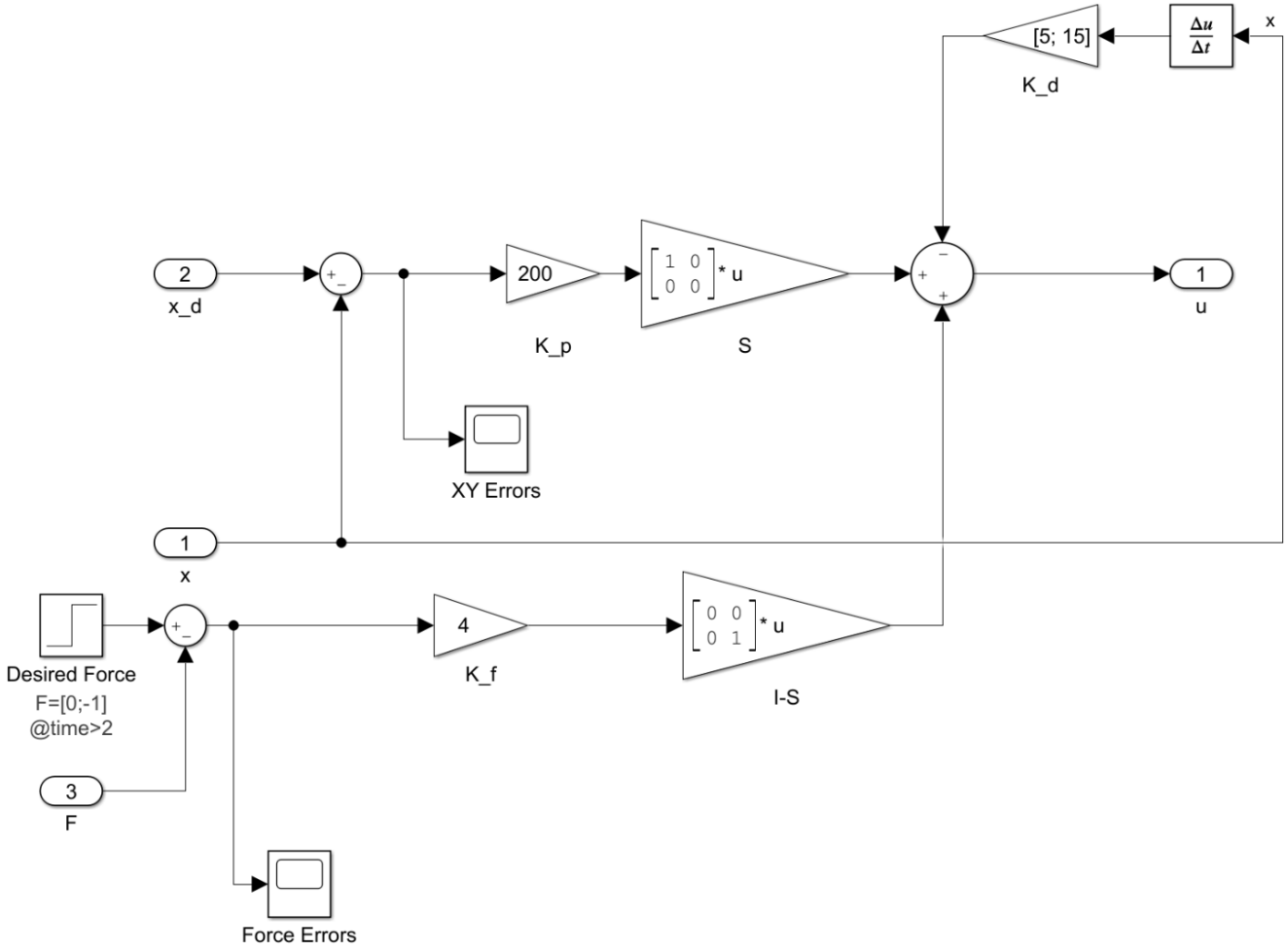
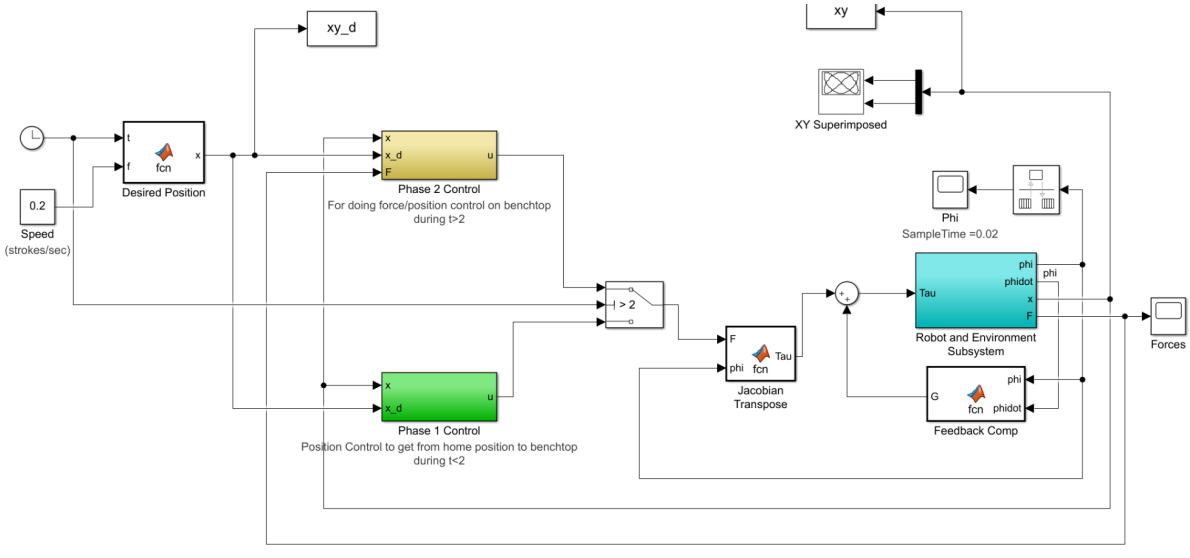
ME EN 6230
Lab 5
Ryan Dalby

Note that lab station 1 was used for this lab.

1. Hybrid Position/Force Control with P Force Control

1.1. Model



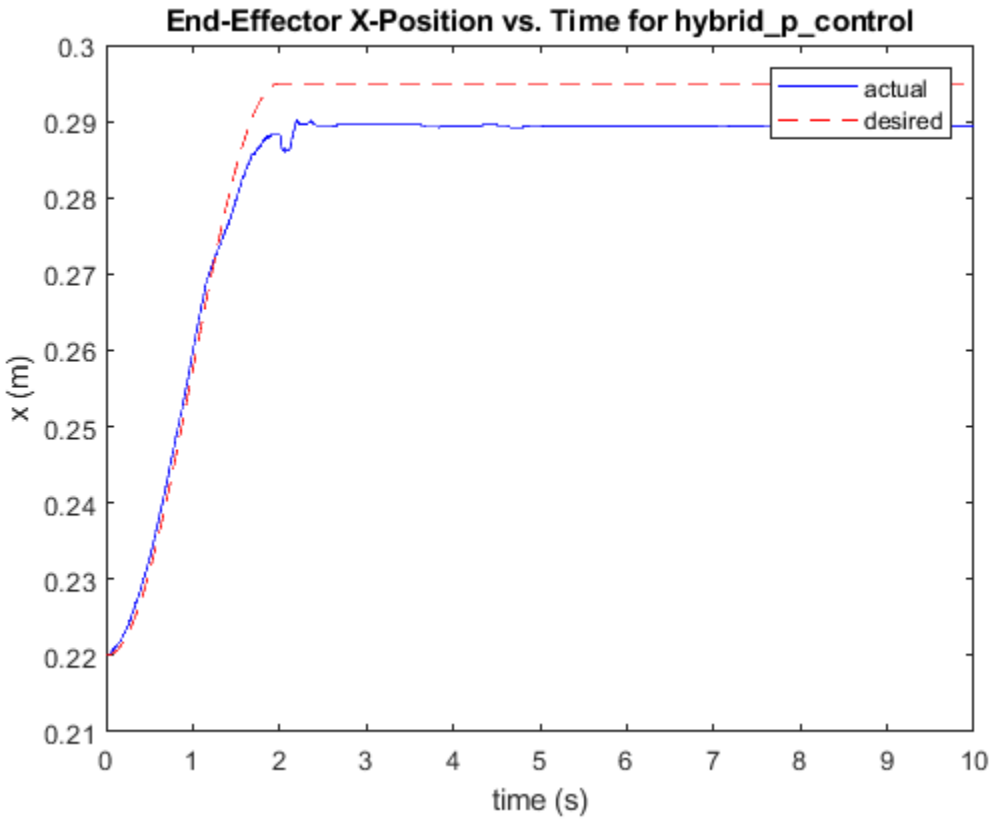


1.2. Code

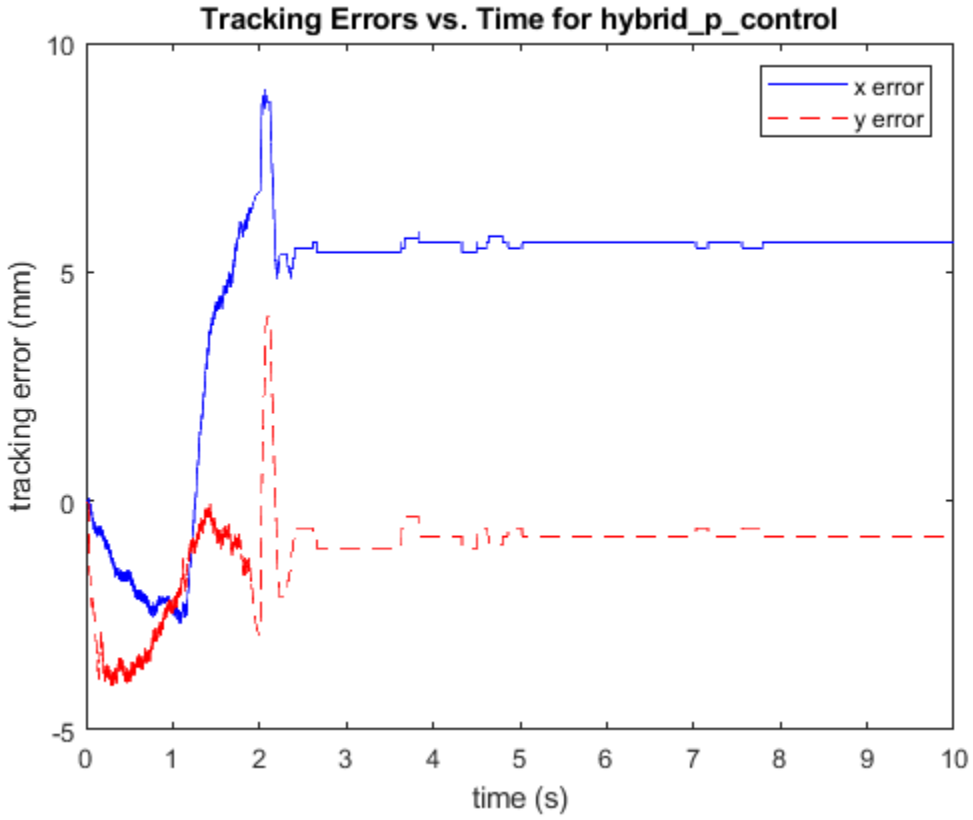
No additional MATLAB function blocks were needed for this controller that were not already described by the template or above.

1.3. Quasi-static Trajectory (0 strokes/sec)

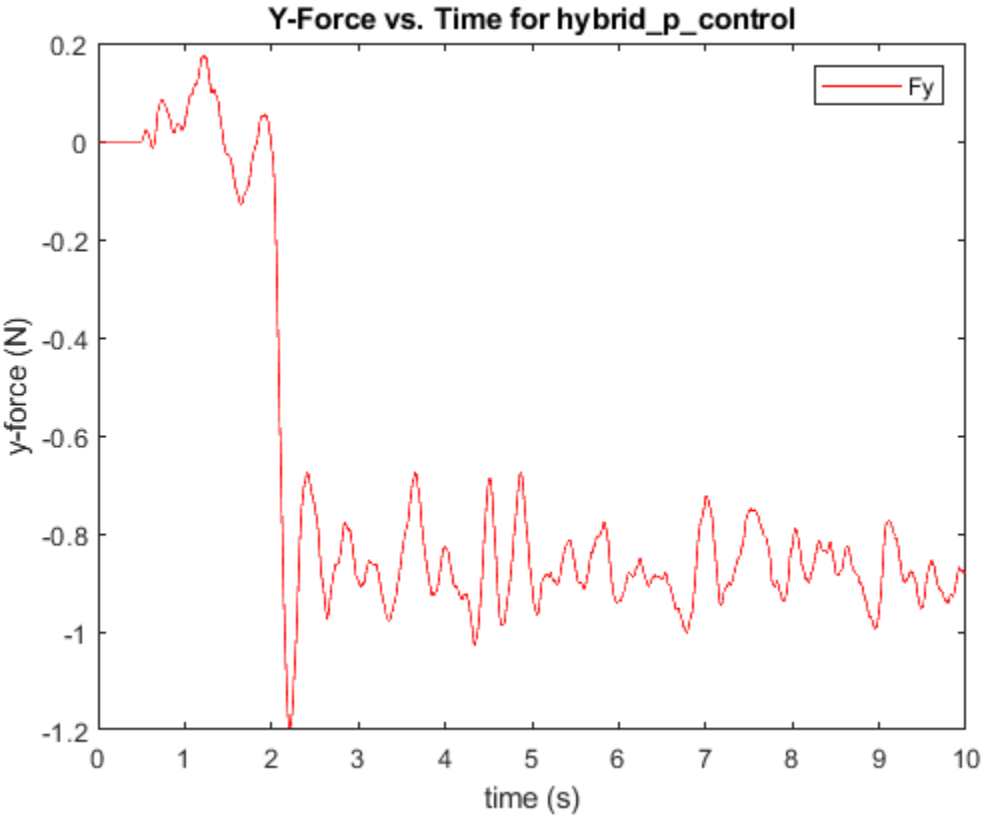
1.3.1. End-Effector X-Position



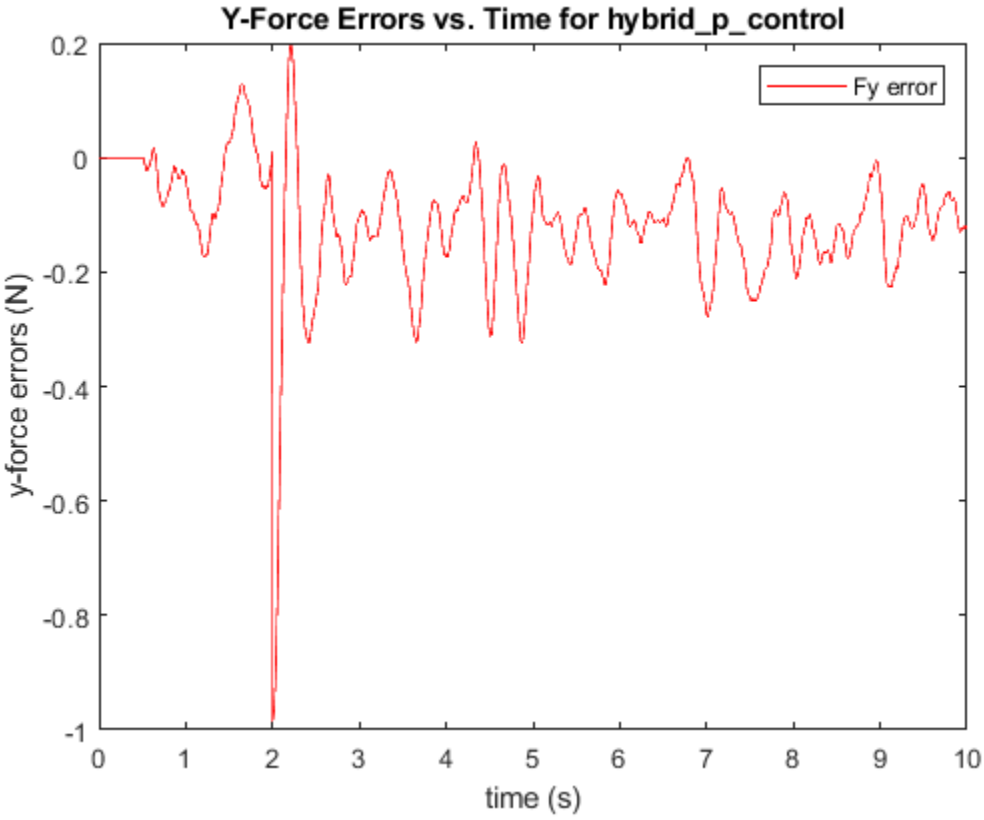
1.3.2. Tracking Errors



1.3.3. Y-Force

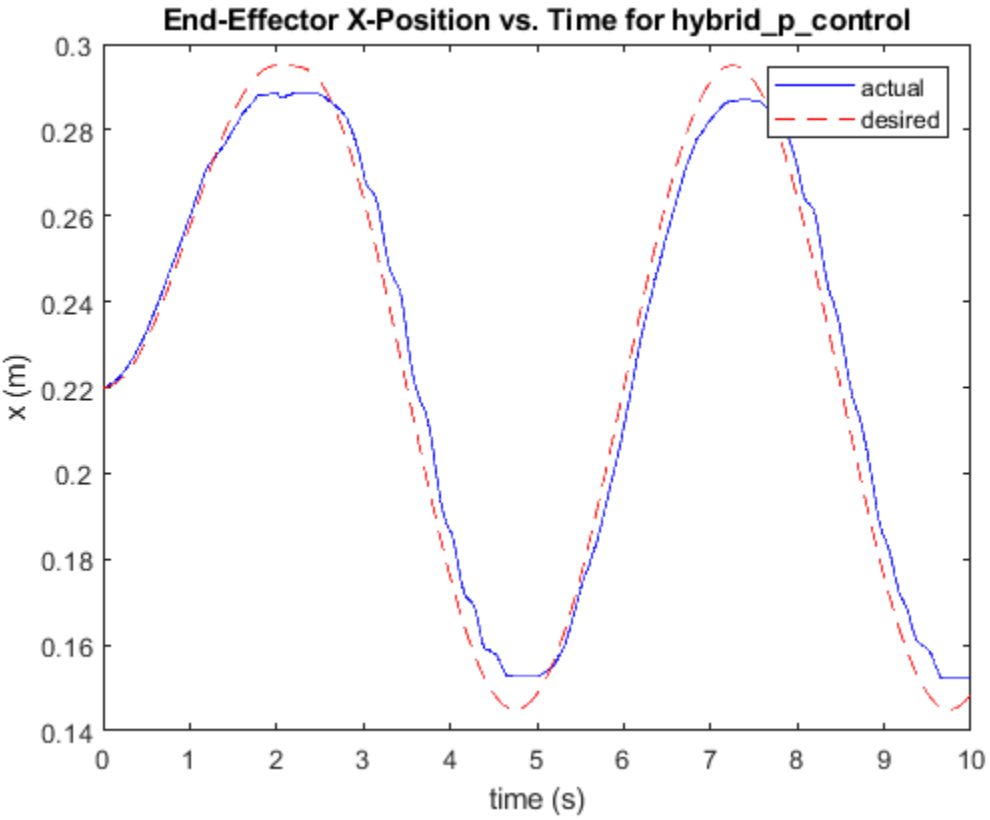

Y-Force vs. Time for hybrid_p_control
Fy
y-force (N)
time (s)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
1
2
3
4
5
6
7
8
9
10

1.3.4. Y-Force Error

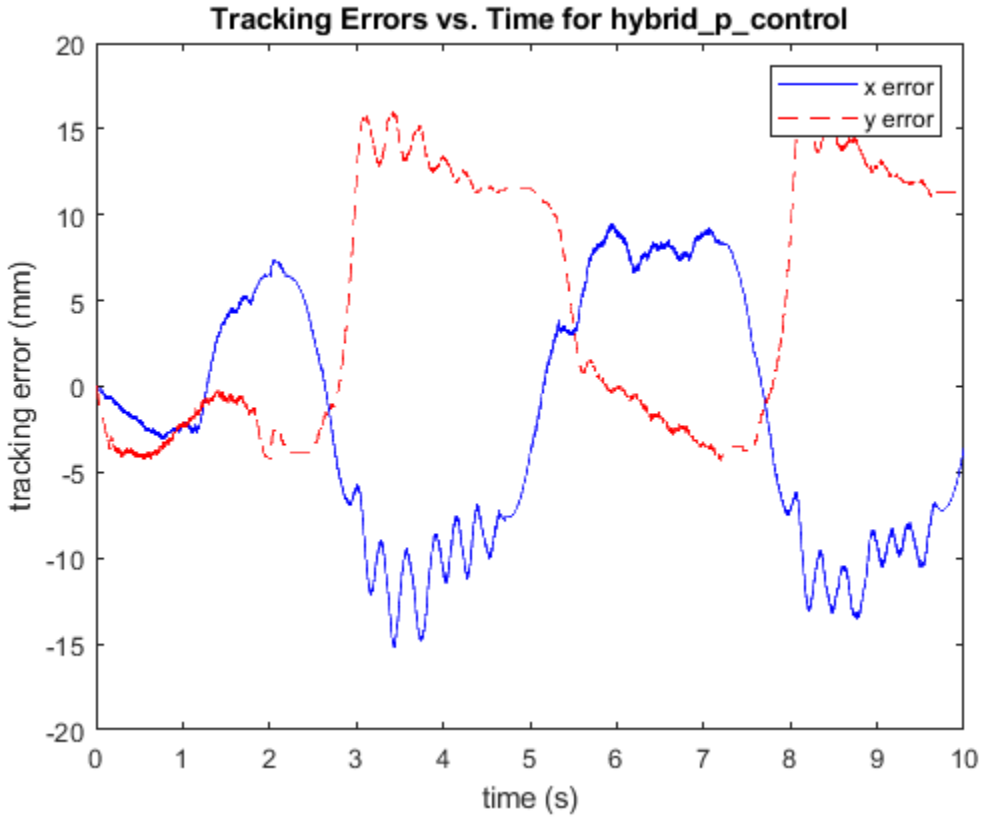


1.4. Slow Trajectory (0.2 strokes/sec)

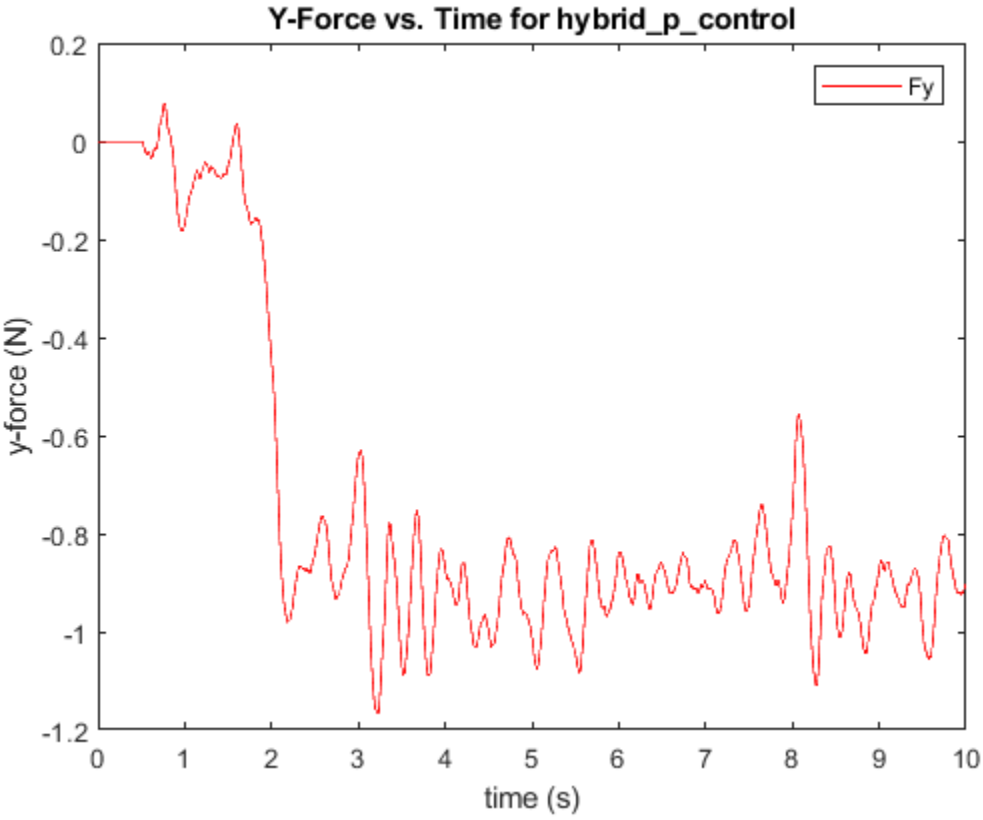
1.4.1. End-Effector X-Position



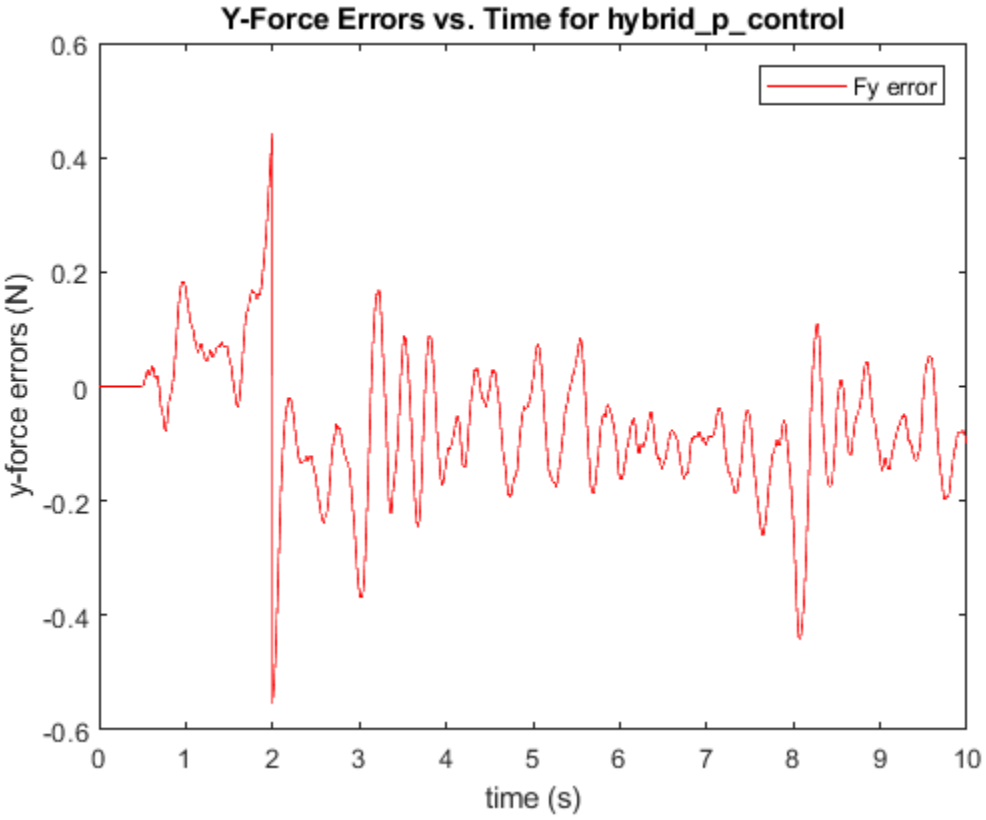
1.4.2. Tracking Errors



1.4.3. Y-Force


Y-Force vs. Time for hybrid_p_control
Fy
y-force (N)
time (s)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
1
2
3
4
5
6
7
8
9
10

1.4.4. Y-Force Error



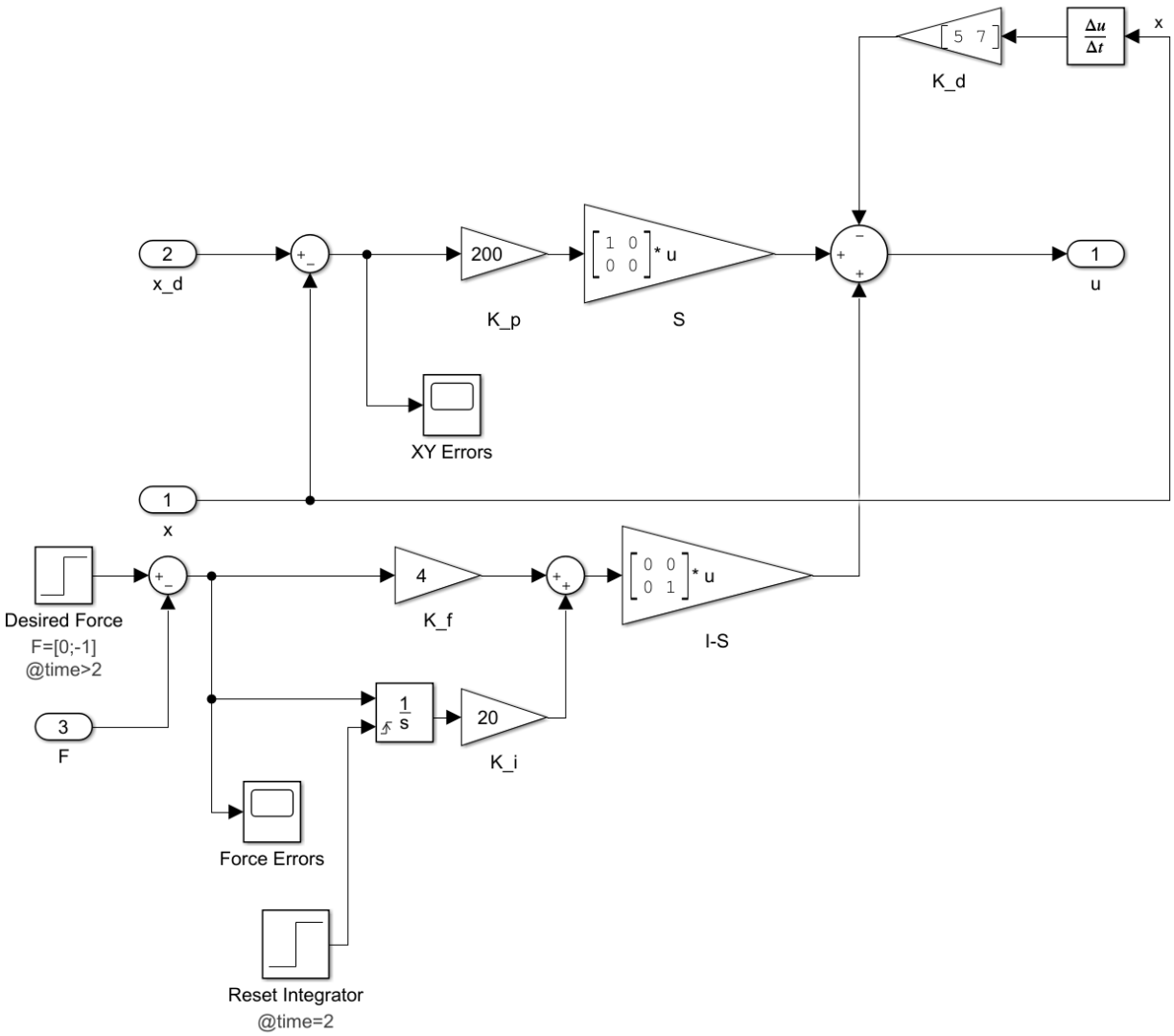
1.5. Analysis

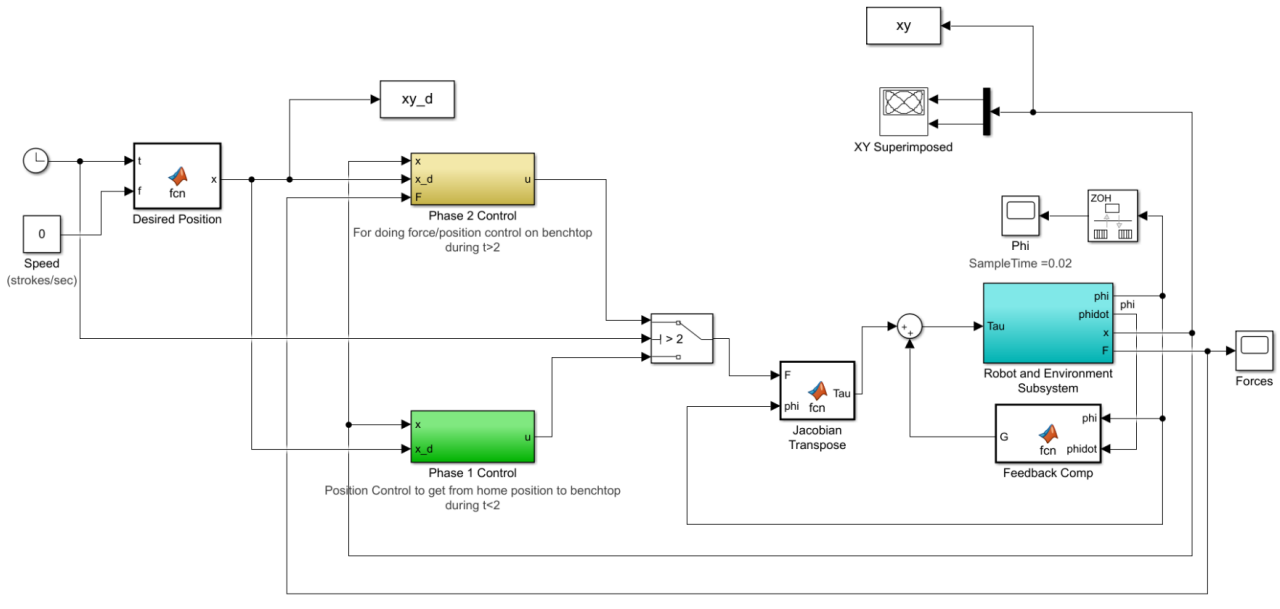
Using hybrid position/force control with only P force control results in stable control for both the quasi-static and low speed trajectory. The K_p and K_f gains did have to be lowered from their simulation values (500 to 200 and 10 to 4 respectively) in order to get stable trajectory and force tracking although these values were still acceptable for good performance. The K_d gain for the y-direction was able to be raised with no effect to the stability and to provide better dampening to the y component of the trajectory.

In terms of performance, the controller was able to track the x-position and the y-force but with some steady-state error. This steady state error was present for both the quasi-static and low speed trajectory and can be seen primarily in the y-force error plot which converges to a non-zero force error value. Do note that the force and force error plots have some oscillations due to the force sensor.

2. Hybrid Position/Force Control with PI Force Control

2.1. Model



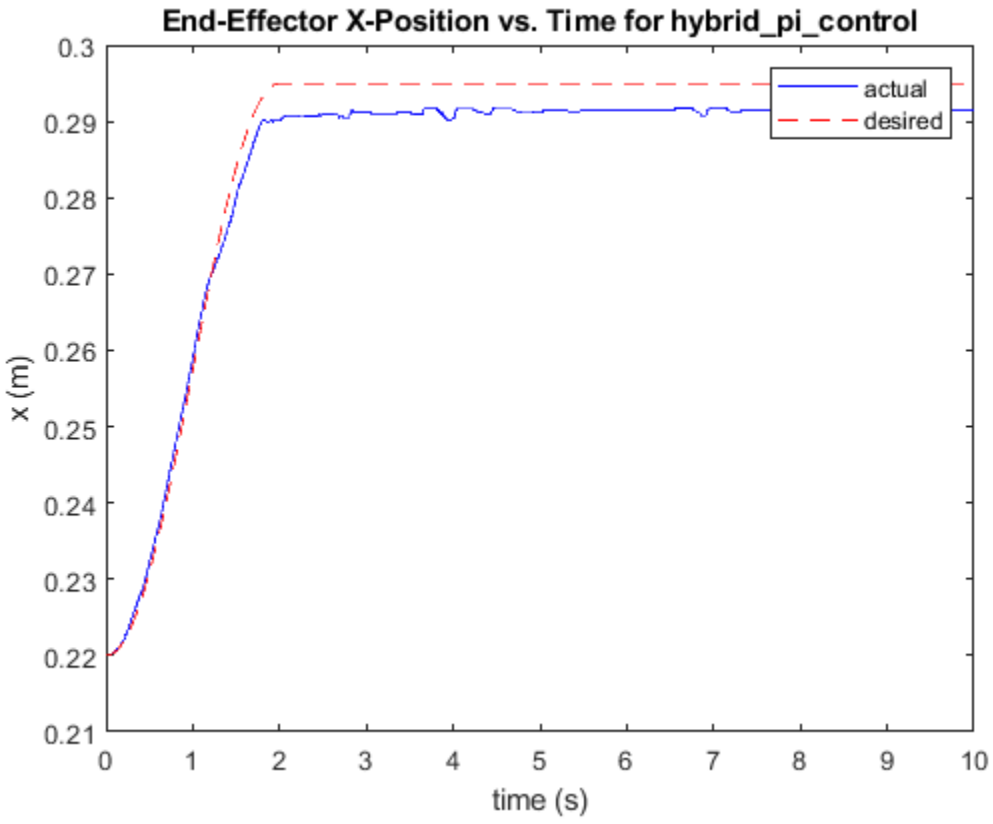


2.2. Code

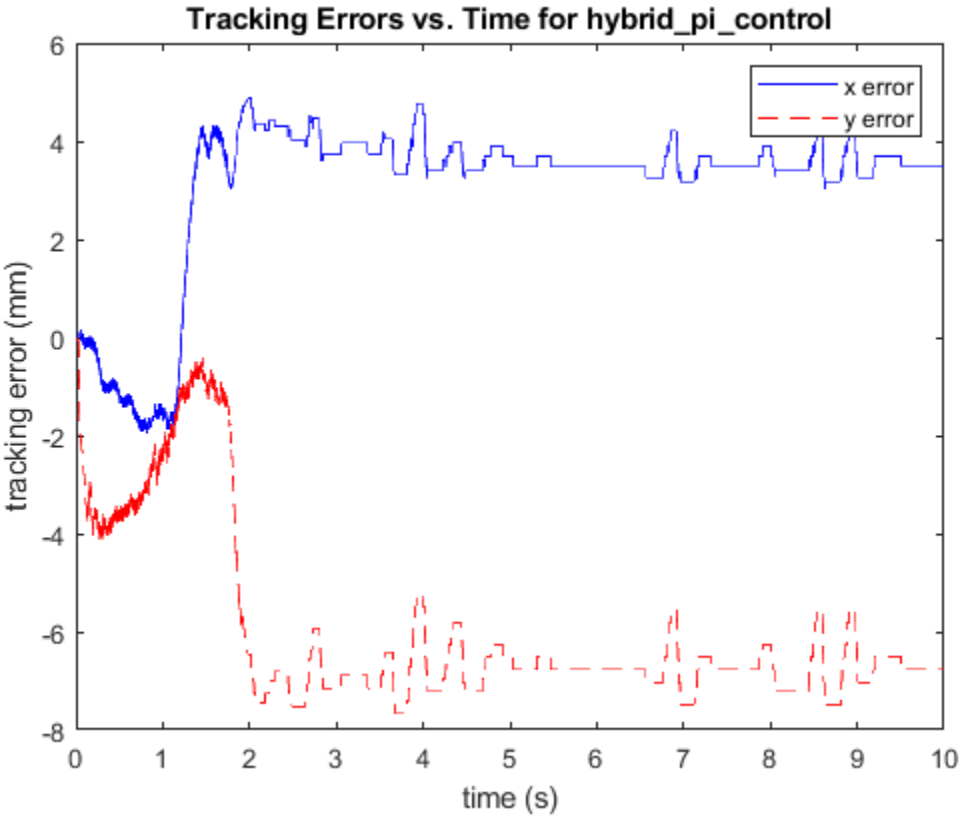
No additional MATLAB function blocks were needed for this controller that were not already described by the template or above.

2.3. Quasi-static Trajectory (0 strokes/sec)

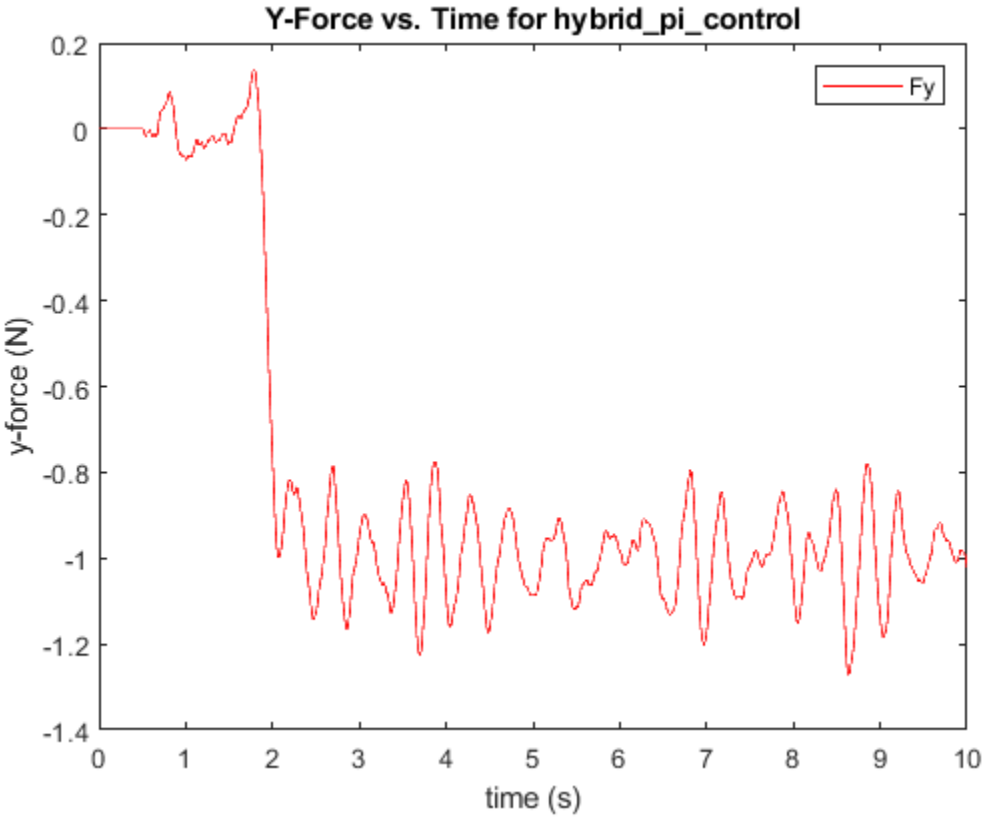
2.3.1. End-Effector X-Position



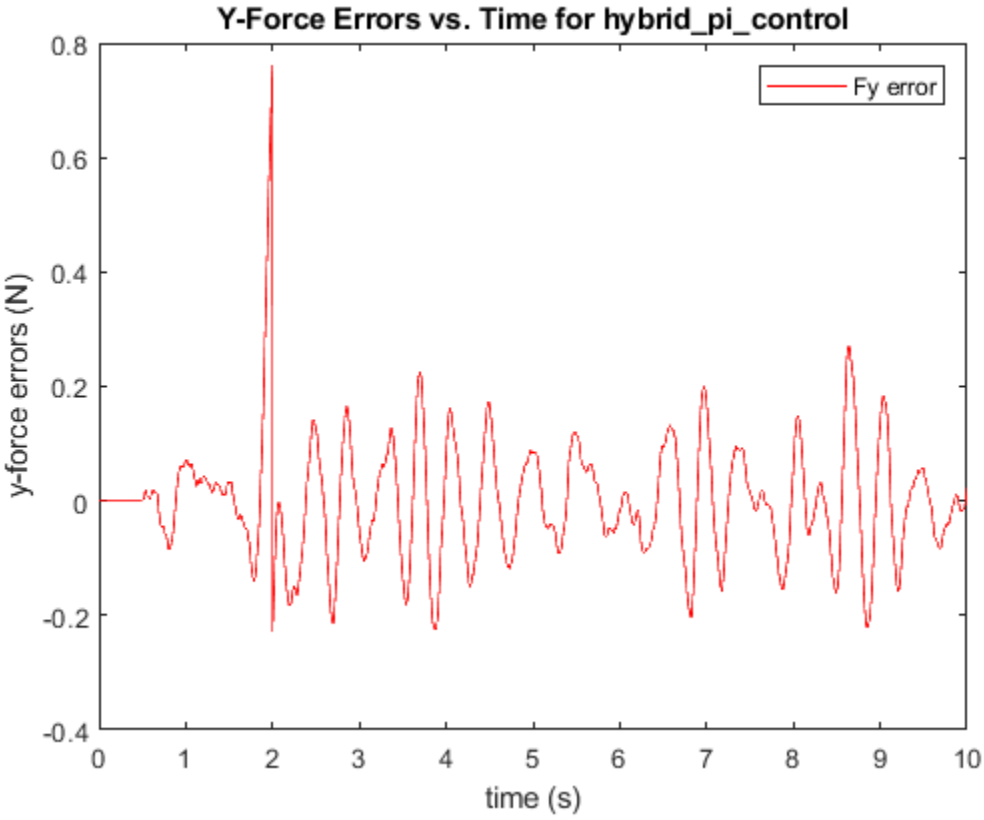
2.3.2. Tracking Errors



2.3.3. Y-Force

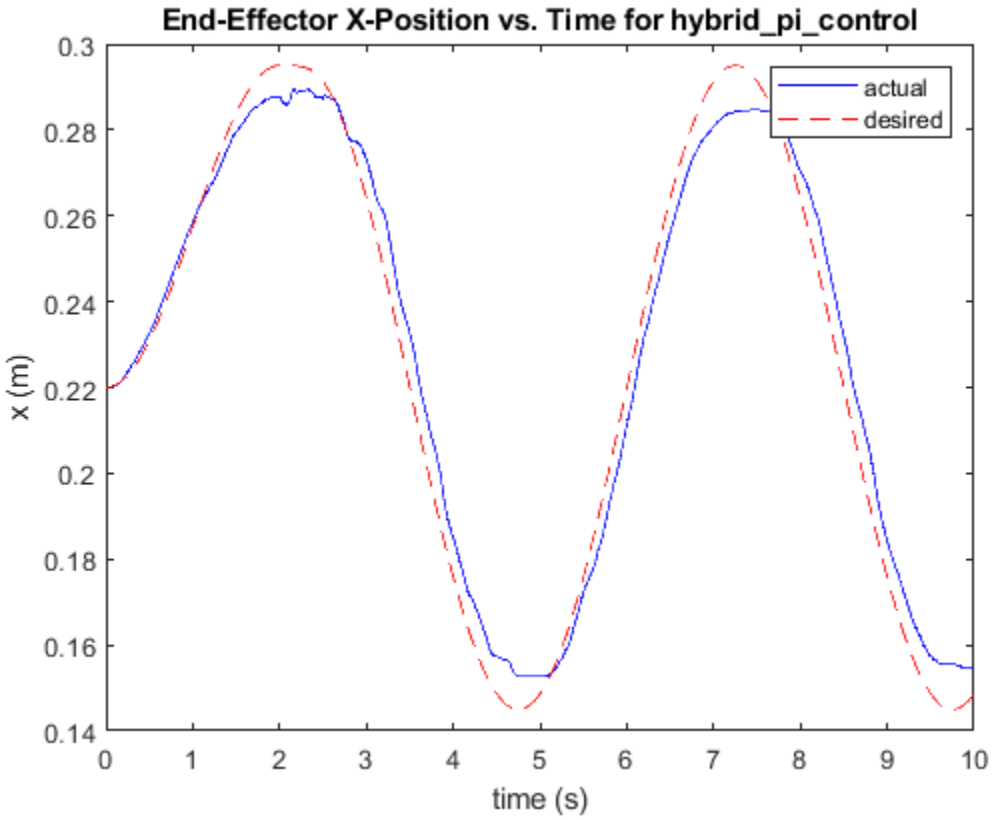

Y-Force vs. Time for hybrid_pi_control
Fy
y-force (N)
time (s)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
0
1
2
3
4
5
6
7
8
9
10

2.3.4. Y-Force Error

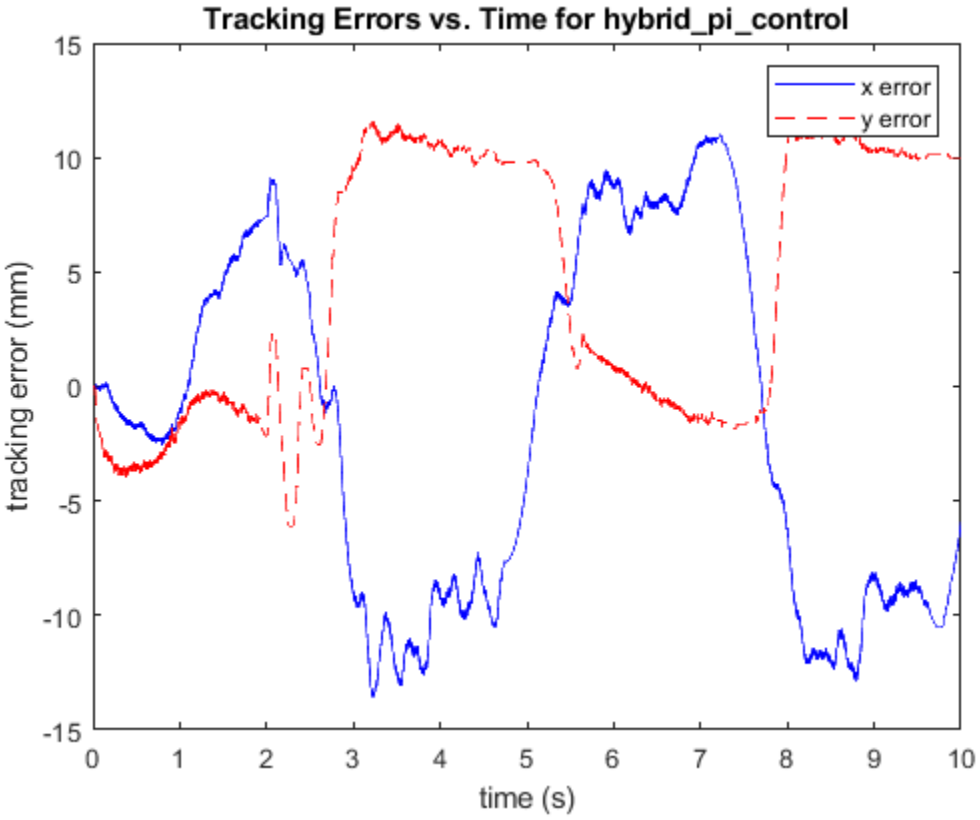


2.4. Slow Trajectory (0.2 strokes/sec)

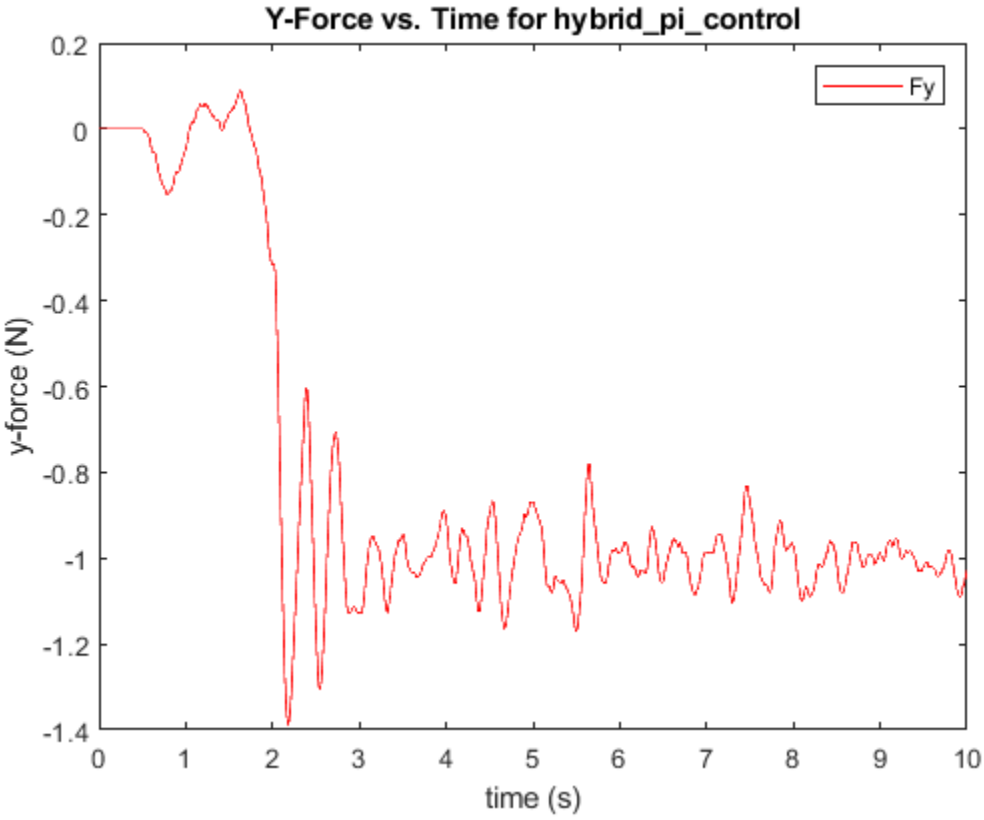
2.4.1. End-Effector X-Position



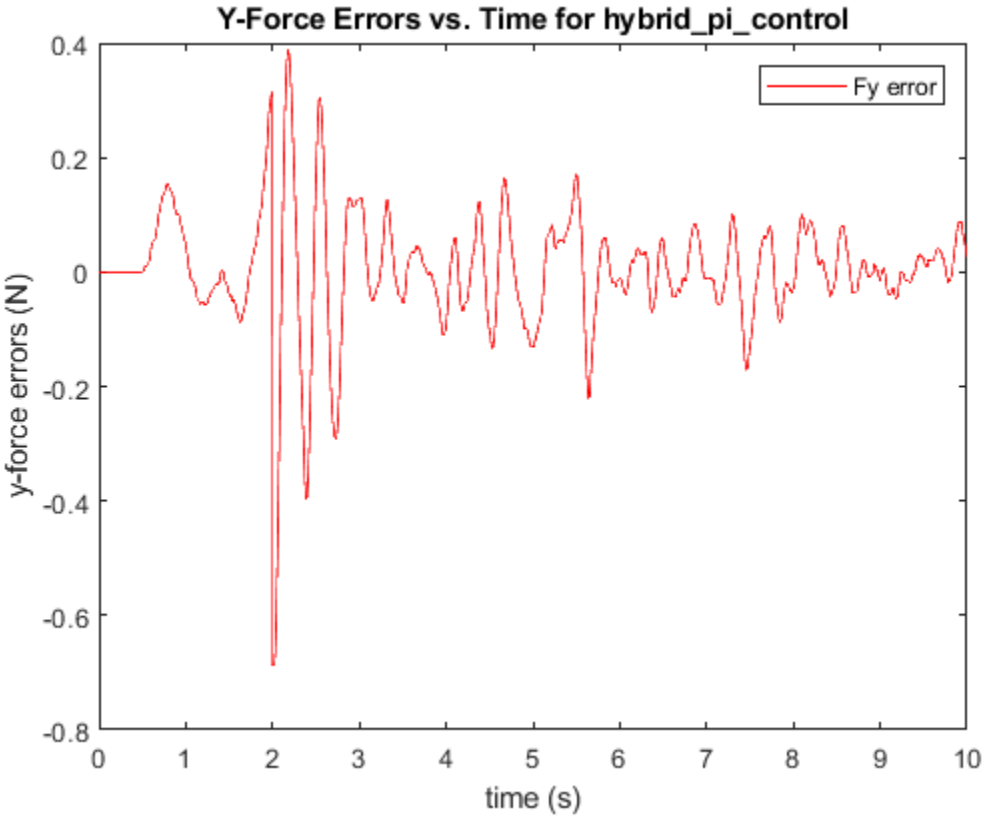
2.4.2. Tracking Errors



2.4.3. Y-Force


Y-Force vs. Time for hybrid_pi_control
Fy
y-force (N)
time (s)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
0
1
2
3
4
5
6
7
8
9
10

2.4.4. Y-Force Error



2.5. Analysis

Using hybrid position/force control with P and I force control results in stable control for both the quasi-static and low speed trajectory once tuned properly. The K_p and K_f gains did have to be lowered from their simulation values (500 to 200 and 10 to 4 respectively) and the y-value of K_d had to be lowered when compared to the P force control only hybrid controller to maintain stable control. This shows that when compared to P force control only this controller is more unstable because of the integration of the force which is inherently noisy. Thus the K_i value had to be lowered dramatically when compared to the simulation value (from 1000 to 20).

In terms of performance, even with the low integral gain the controller still has better y-force tracking than P force control alone since the value the y-force oscillates about is now the desired -1 N. Thus the steady-state error in the force control is reduced through the integral control. In terms of tracking position the performance is virtually identical since the position tracking of the hybrid controller is unchanged.

3. Appendix- Plotting Code (Used to make simulation plots)

```
%% ME EN 6230 Lab 5 Ryan Dalby
% close all;
set(groot, 'DefaultTextInterpreter', 'none') % Prevents underscore from
becoming subscript

% Do note that for this lab the XYErrors and ForceErrors were changed from
% outputting as a dataset to outputting as a structure with time and a
block
% to output xy_d was added when compared to Lab5_template2021.slx

% Extract necessary data, will error if the data does not exist
time = XYErrors.time; % s
model_title = extractBefore(XYErrors.blockName, "/");
actual_trajectory = xy; % m
desired_trajectory = xy_d; % m
tracking_errors = XYErrors.signals.values*1000; % mm
forces = F(:,2:3); % N
force_errors = ForceErrors.signals.values; % N

% Plot End-Effector X-Position
figure;
plot(time, xy(:,1), 'b-');
hold on;
plot(time, xy_d(:,1), 'r--');
title(strcat("End-Effector X-Position vs. Time for ", model_title));
xlabel("time (s)");
ylabel("x (m)");
legend("actual", "desired");

% Plot Tracking Errors
figure;
plot(time, tracking_errors(:,1), 'b-');
hold on;
plot(time, tracking_errors(:,2), 'r--');
title(strcat("Tracking Errors vs. Time for ", model_title));
xlabel("time (s)");
ylabel("tracking error (mm)");
legend("x error", "y error");


% Plot Y-Force
figure;
```

```
plot(time, forces(:,2), 'r-');
title(strcat("Y-Force vs. Time for ", model_title));
xlabel("time (s)");
ylabel("y-force (N)");
legend("Fy");


% Plot Y-Force Error
figure;
plot(time, force_errors(:,2), 'r-');
title(strcat("Y-Force Errors vs. Time for ", model_title));
xlabel("time (s)");
ylabel("y-force errors (N)");
legend("Fy error");
```