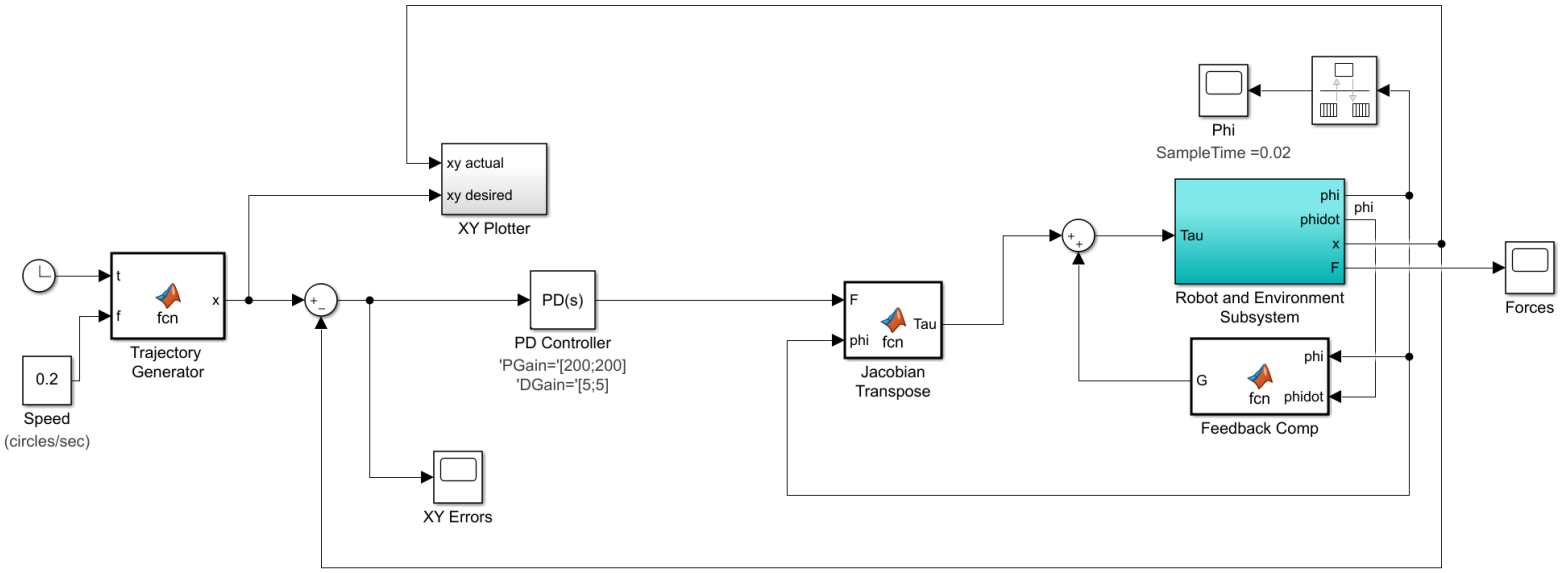
ME EN 6230
Lab 4
Ryan Dalby

Note that lab station 1 was used for this lab. All trajectories were run at 0.2 circles/sec.

1. Template Baseline

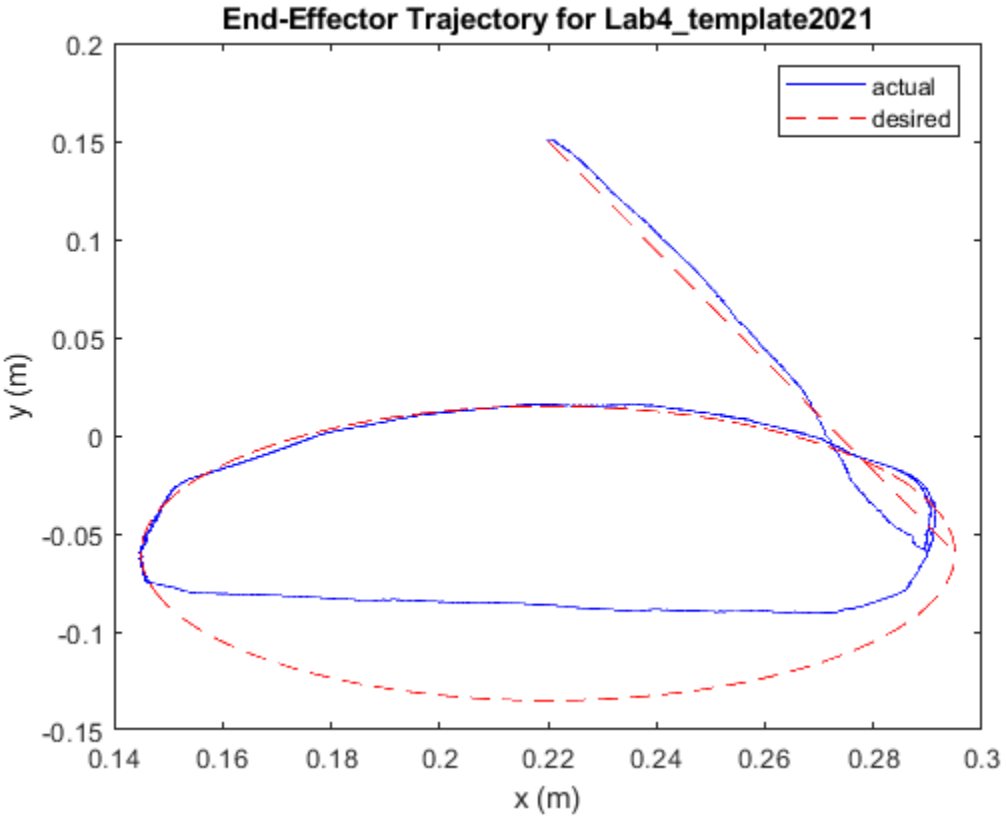
1.1. Model



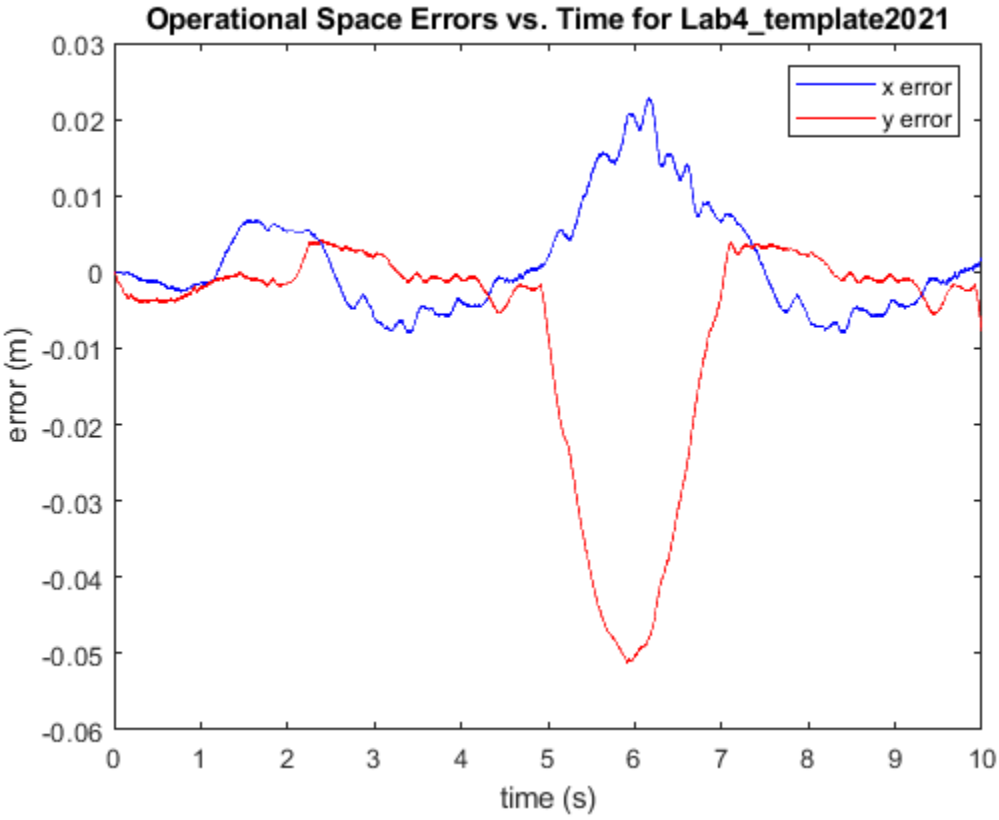
1.2. Code

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

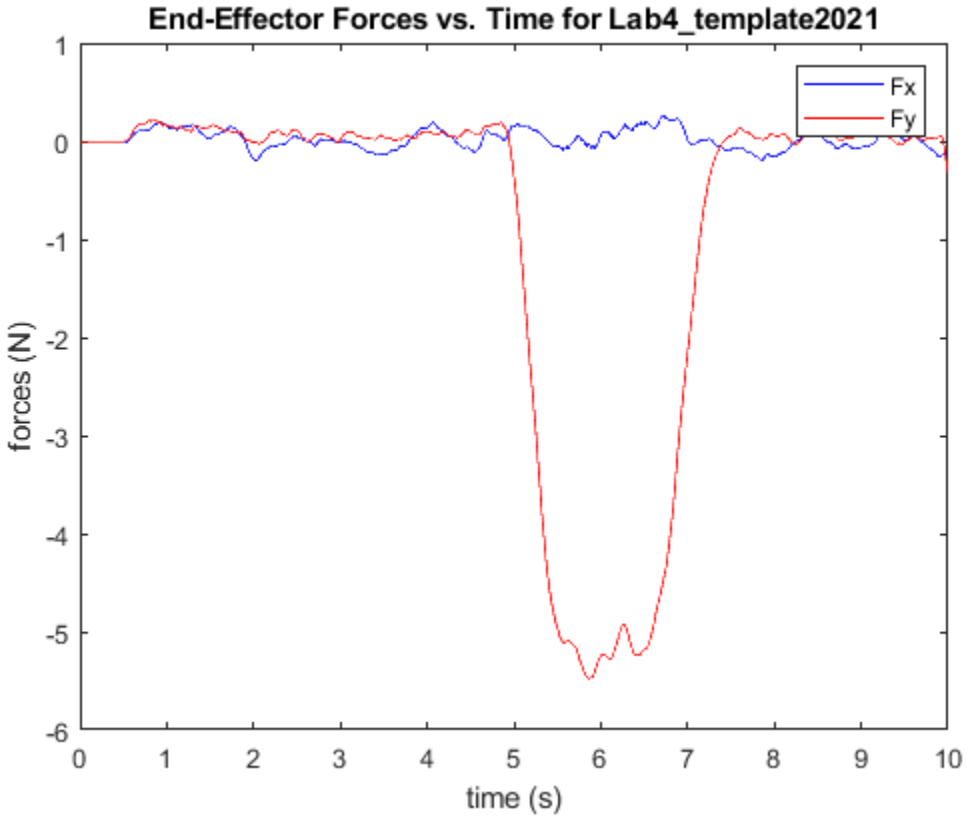
1.3. X-Y Trajectory



1.4. Operational Space Errors



1.5. Contact Forces

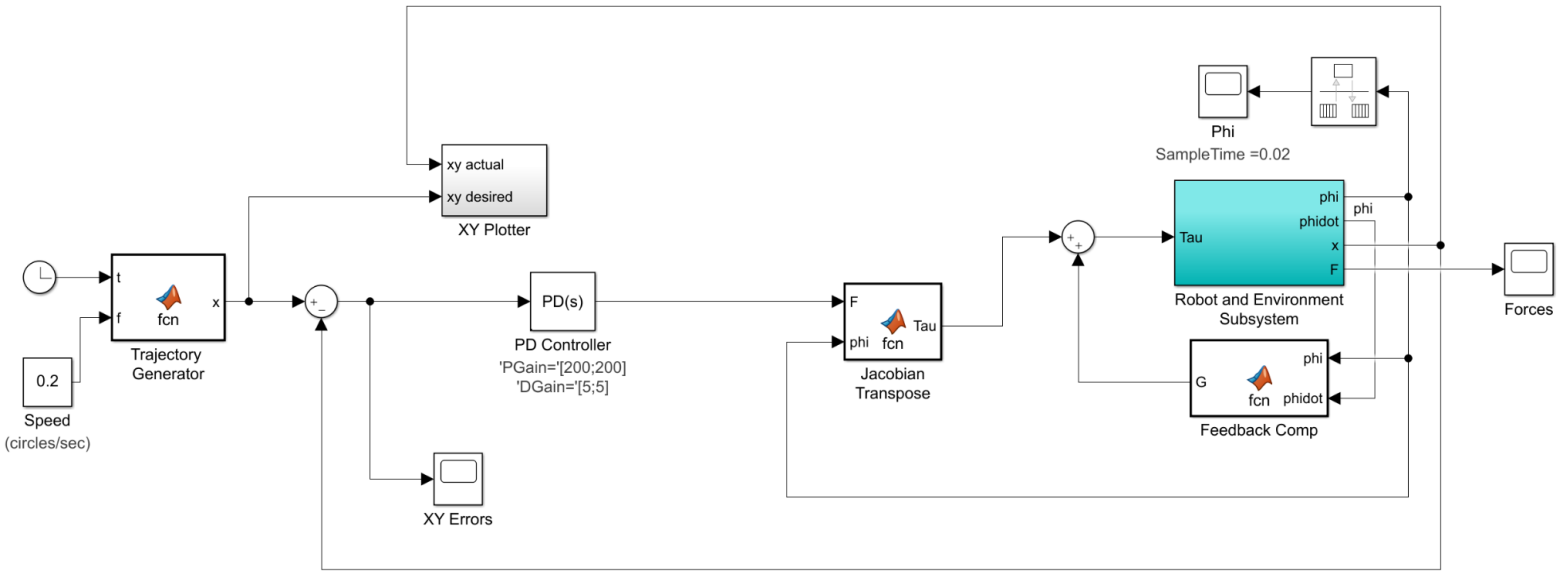


1.6. Analysis

For the template model, which is Jacobian transpose control in operational space, the magnitude of the maximum force in the y-direction was 5.475 N. Tracking is good because of the reasonably stiff PD gain values but contact forces get fairly high since there is no indirect force feedback to augment the trajectory as the other controller tested in this lab.

2. Stiffness/Damping Control

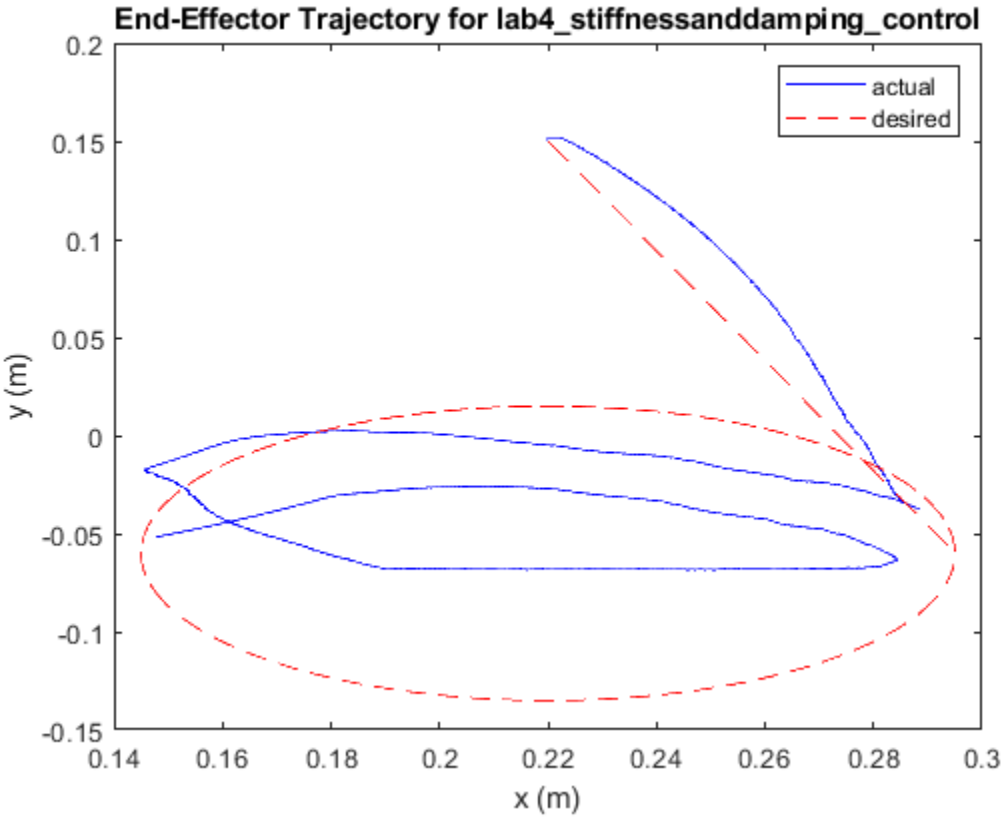
2.1. Model



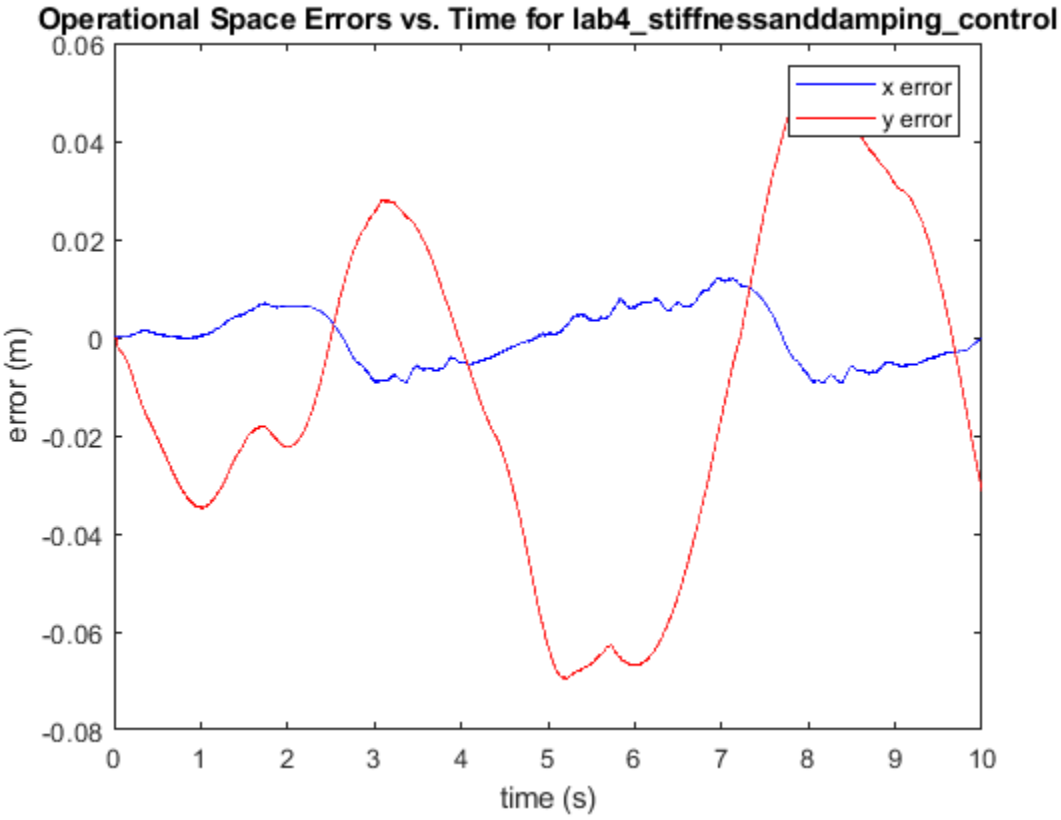
2.2. Code

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

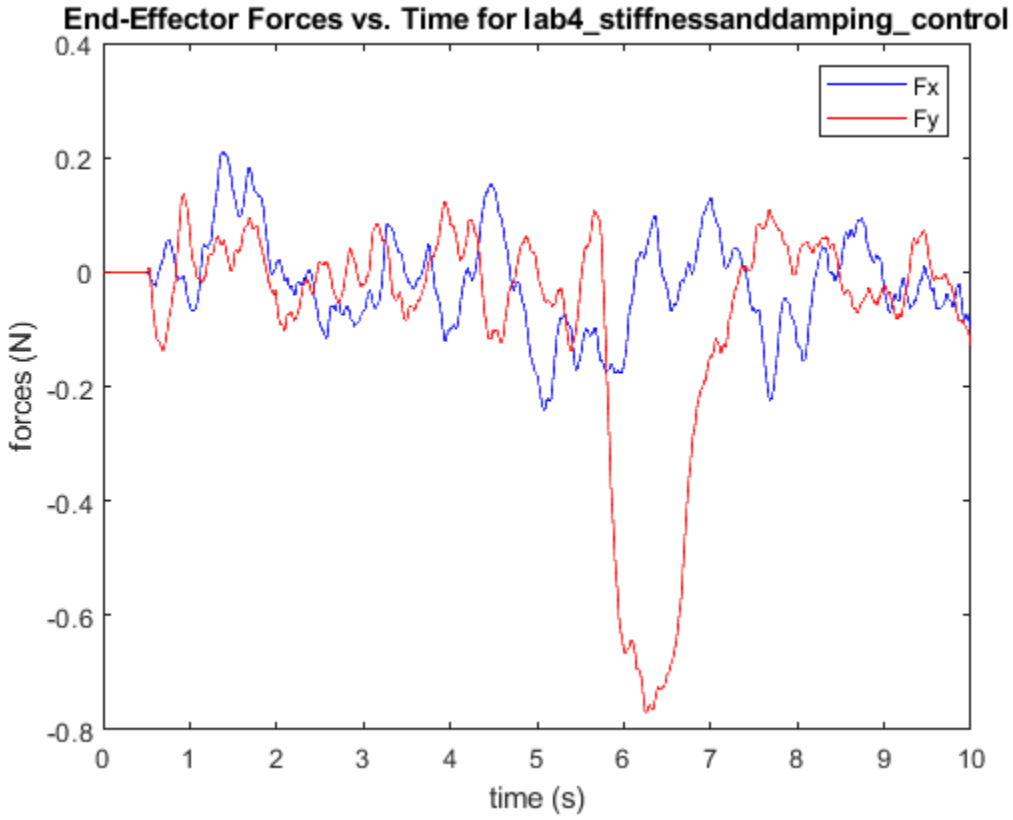
2.3. X-Y Trajectory



2.4. Operational Space Errors



2.5. Contact Forces



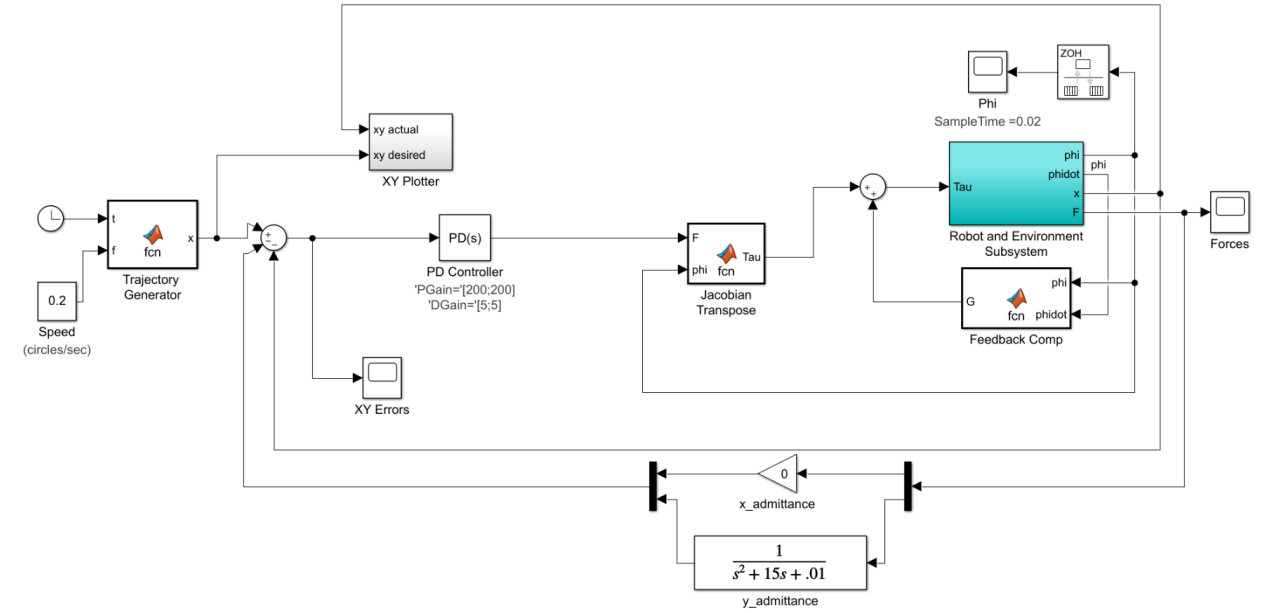
2.6. Analysis

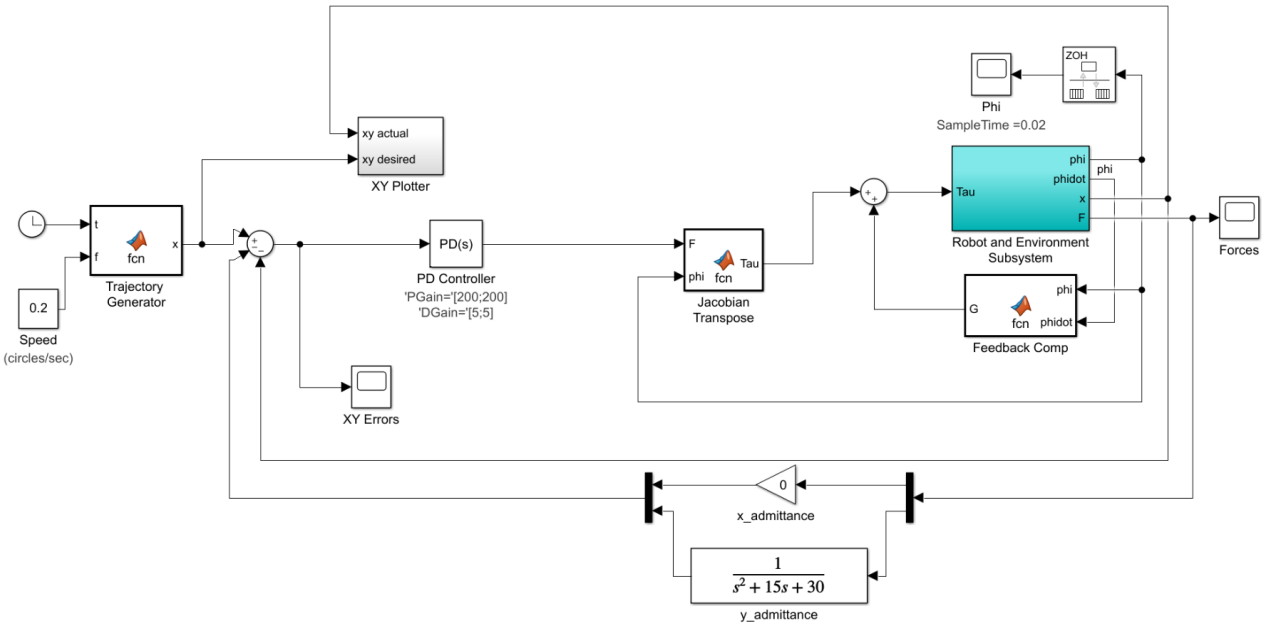
For stiffness and damping control it was possible to reduce the maximum force in the y-direction to 0.772 N. As a result the robot was able to be robust to external forces in the y-direction. The consequence of this control law when compared to the template baseline was that trajectory tracking suffered since soft PD gains had to be used.

3. Compliance Control (Implemented as Admittance Control)

Note: Two sets of admittance transfer functions ($1/(M_e s^2+B_e s+K_e)$) are shown below. The top model (M_e=1, B_e=15, K_e=0.01) almost achieves the 1N contact force target although suffers when tracking the trajectory in the y-direction since the trajectory is augmented heavily by the force feedback. The bottom model (M_e=1, B_e=15, K_e=30) still achieves reasonable contact forces yet maintains a good trajectory.

3.1. Model

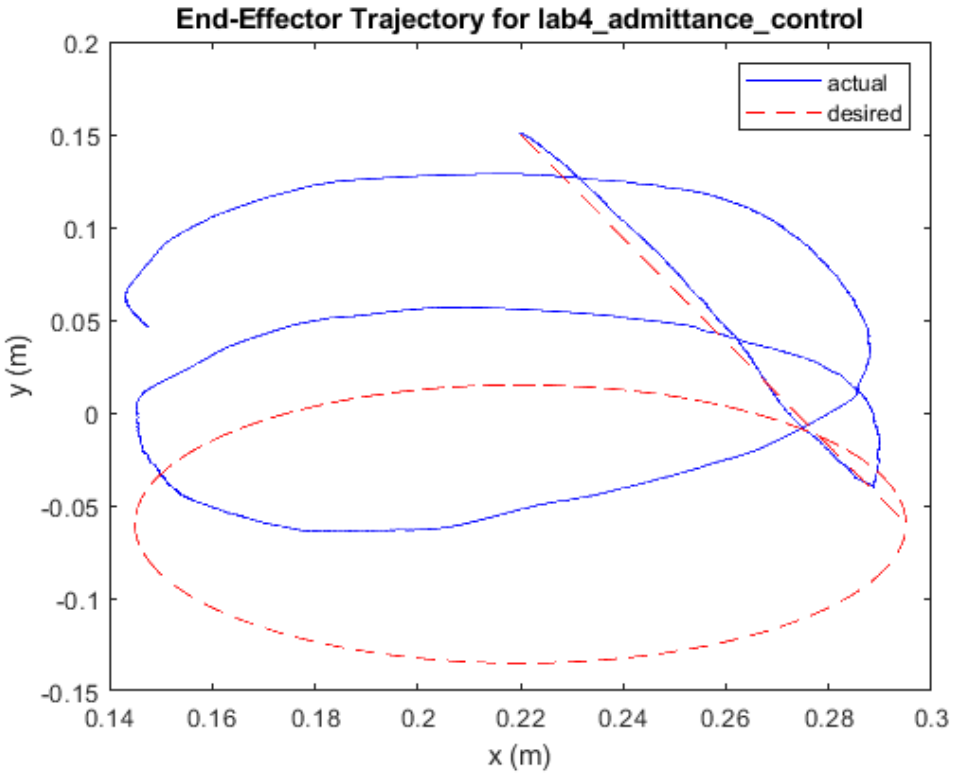


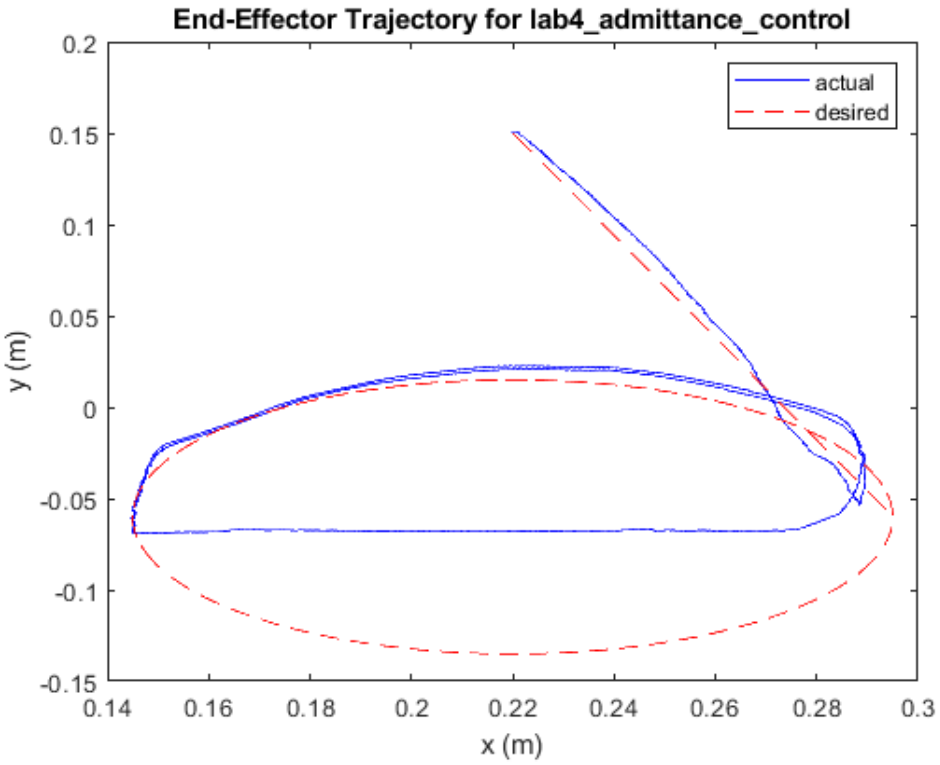


3.2. Code

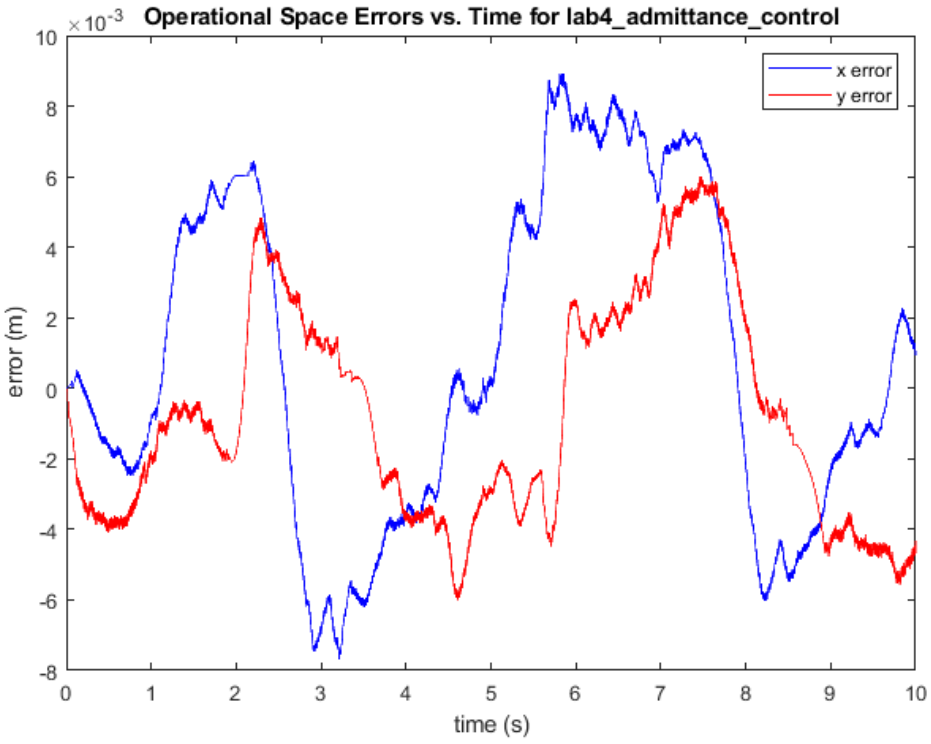
No additional MATLAB function blocks were needed for this controller that were not already described by the template or above.

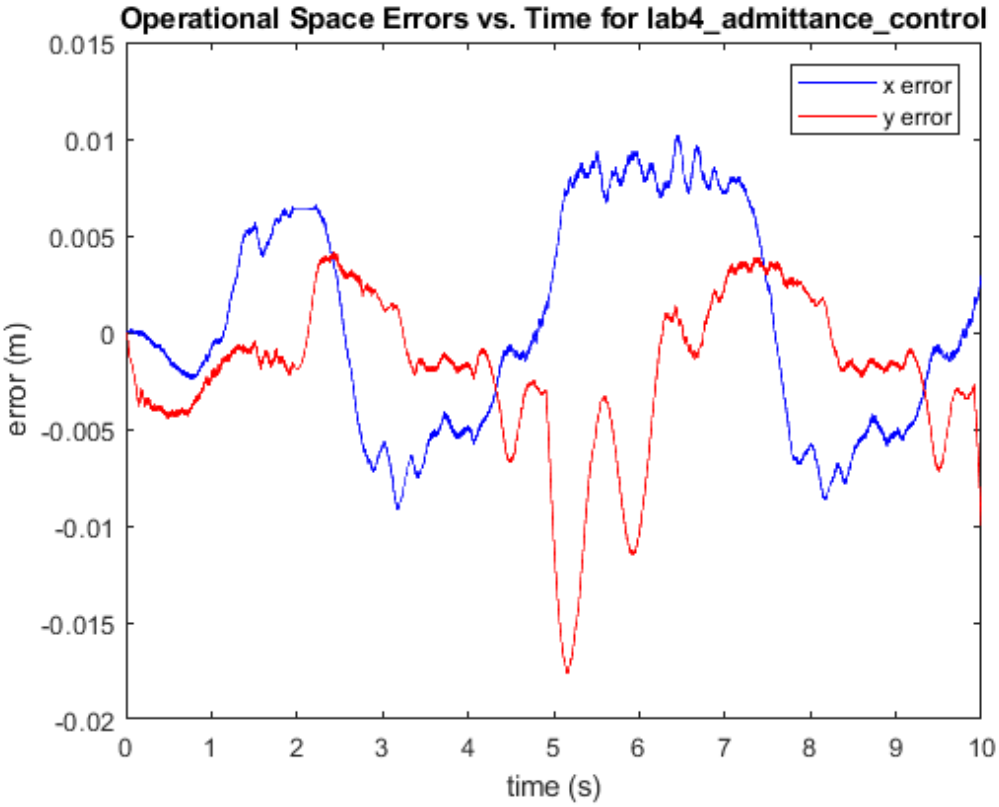
3.3. X-Y Trajectory



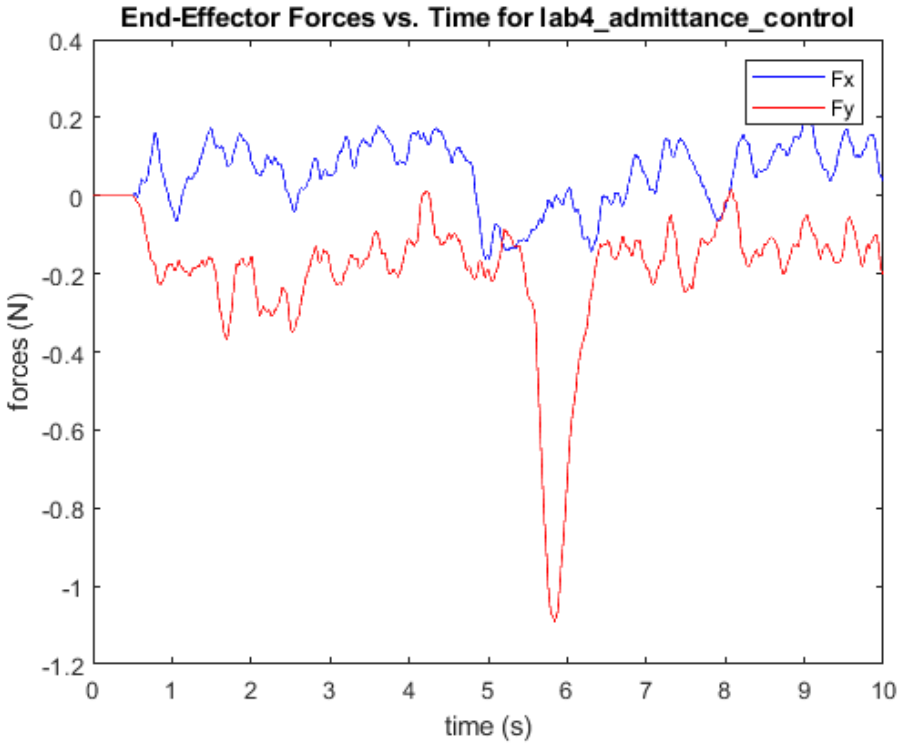


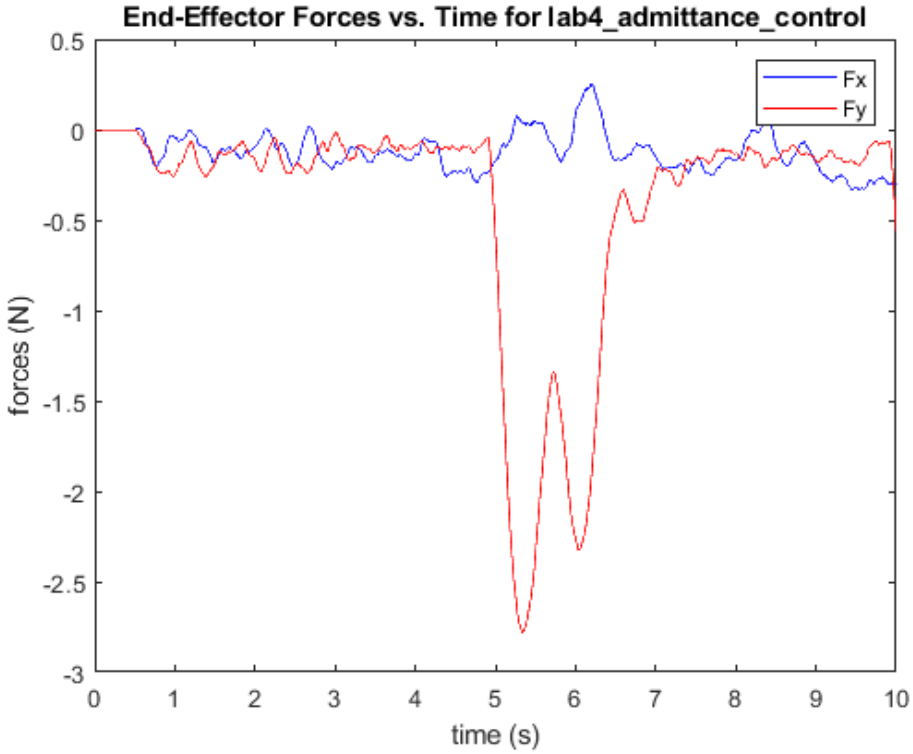
3.4. Operational Space Errors





3.5. Contact Forces





3.6. Analysis

For this lab compliance control was implemented as admittance control to give more control over the mass and damping parameters of the spring-mass damper that the end-effector behaves with admittance control. As mentioned previously, there were two sets of admittance transfer function values that were chosen since they each performed differently (these were found after trying many different parameter values). Performance for the top controller resulted in a maximum contact force of 1.090 N with poor tracking in the y-direction. Performance for the bottom controller resulted in a maximum contact force of 2.779 N with tracking performance that was just as good as the template baseline. These two admittance controllers show the range of performance characteristics you can get by altering the admittance transfer function, although it is necessary to be careful to not make the controller go unstable. The admittance controller may be good for a specific trajectory that is not susceptible to disturbances but I would be concerned about possible instability with any sort of novel circumstances.

(Note: Compliance control alone with a compliance of [0; 0.03] was also implemented but was not shown here since it gives performance comparable results to the bottom admittance controller but is not as tunable.)
(Also note: M_e values of 1 were chosen since any other values of M_e would be equivalent to dividing out M_e and multiplying the transfer function by $1/M_e$ which is just scaling the transfer function output.)

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

```
%% ME EN 6230 Lab 4 Ryan Dalby
set(groot, 'DefaultTextInterpreter', 'none') % Prevents underscore from
becoming subscript

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

% Plot End-Effector X-Y 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");

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

% Plot End-Effector Forces vs Time
figure;
plot(time, forces(:,1), 'b-');
hold on;
plot(time, forces(:,2), 'r-');
title(strcat("End-Effector Forces vs. Time for ", model_title));
xlabel("time (s)");
ylabel("forces (N)");
legend("Fx", "Fy");
```