Jesse Weisberg
ESE 446: Robotics: Dynamics and Control
Final Project: Part 2
Terry

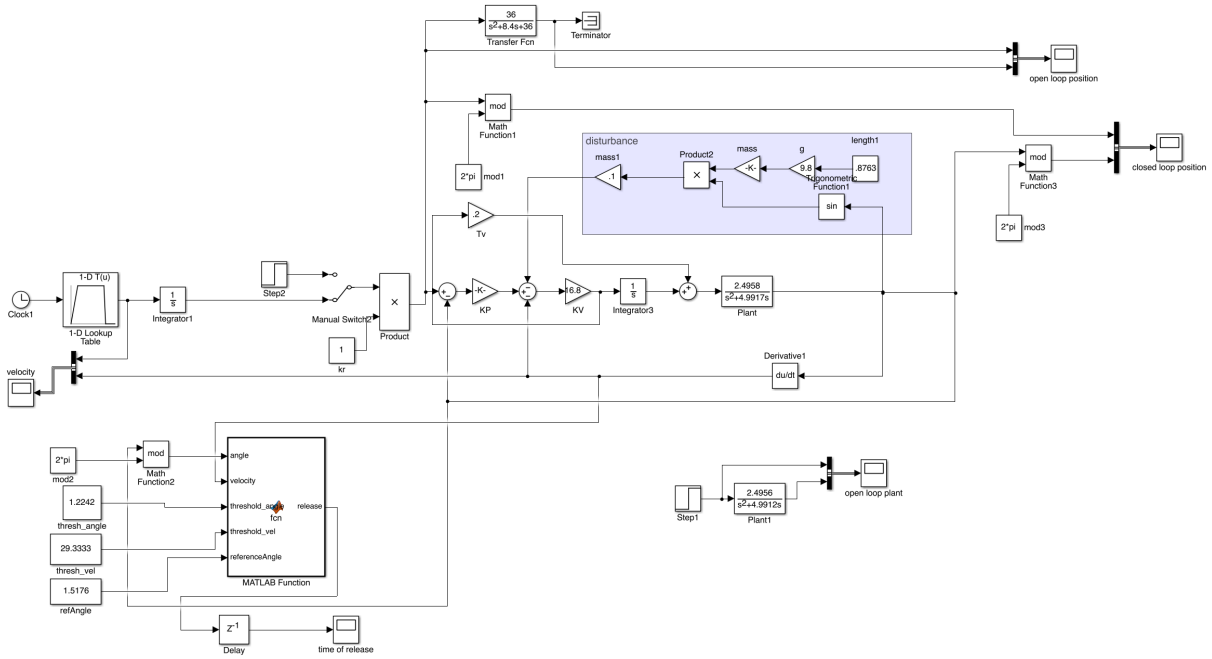
Strikeout-Inator 2000: Critical Design Review

Review of System Requirements Review

The problem involves developing a prototype of a robotic manipulator that will test new baseballs to be used by major league pitchers. Our robotic manipulator will throw a ball from the pitchers mound to home plate, subject to the requirement that the ball is thrown 40-60mph and within the strike box.

Although the first part of this project we worked on a 3-link anthropomorphic arm, the current model and control were built for a single-link planar manipulator. Using the given parameters for the motor and the link and the target speed of the ball, a controller was built to simulate the manipulator throwing a ball into the strike zone at a speed of 60mph. This simulation has been successfully achieved and demonstrated in the report below.

Control Architecture



The overall control system built in Simulink is shown above.

Plant Manipulator

The first step to build this system was to derive the transfer function from the dynamics of a single link manipulator. This was done by taking the Laplace transform of the following equation:

$$\frac{(I + I_m k_r^2)}{k_r^2}\ddot{\theta}_m + \frac{F_v}{k_r^2}\dot{\theta}_m + \frac{mglsin(\theta)}{k_r} + \frac{K_t K_v}{R_a}\dot{\theta}_m = \frac{K_t}{R_a}E_a$$

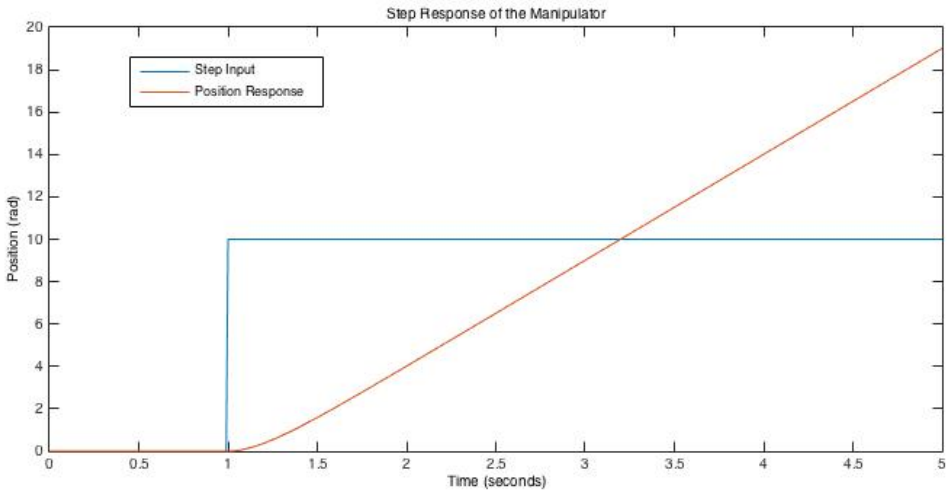
The $mglsin(\theta)$ term was treated as a disturbance since we are only modeling the linear dynamics of the system with the Laplace transform. The modeled disturbance can be seen above in the shaded purple region. The resulting transfer function followed the form:

$$\frac{\theta_m(s)}{V(s)} = \frac{\frac{k_t k_r^2}{R_a(I + I_m k_r^2)}}{s^2 + \frac{F_v}{I + I_m k_r^2}s + \frac{k_R k_t k_v}{R_a(I + I_m k_r^2)}}$$

After plugging in constants, the resulting transfer function became:

$$\frac{\theta_m(s)}{V(s)} = \frac{2.4956}{s^2 + 4.9912s}$$

To check if this is correct, a step response was run to see if the plant manipulator reacted as expected. As can be seen below, the plant manipulator reacts as expected, since the manipulator should continue to rotate as a constant voltage is applied.



Position Controller and Closed Loop System

With this transfer function modeling the linear dynamics of the manipulator, it is possible to create negative feedback loops for angle of rotation of the arm and angular velocity of the arm that control for the position of the arm. These negative feedback loops, along with

their respective gains, allow the controller to optimally control for the position. The gains were derived as follows:

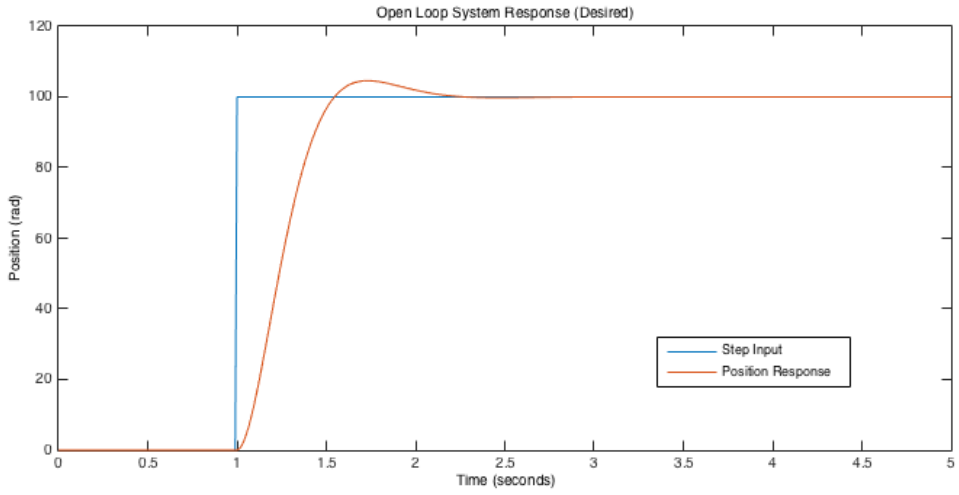
$$K_v = 2\zeta\omega_n k_v = 16.8$$

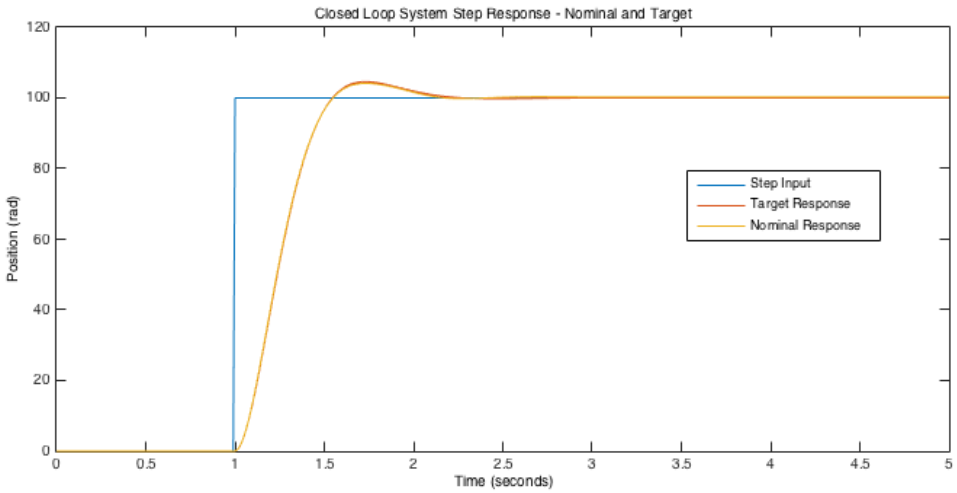
$$K_p = \frac{\omega_n^2 k_v}{k_r K_v} = 42.9$$

Now that the control architecture is set up for the closed loop system, it is possible to determine the transfer function of the closed loop system. This was evaluated, with given parameters of $\zeta = .7, \omega_n = 6,$ as:

$$H(s) = \frac{w_n^2}{s^2 + 2\zeta\omega_n s + w_n^2} = \frac{36}{s^2 + 8.4s + 36}$$

Now it is possible to compare the expected step response from the transfer function of the closed loop system with the actual step response of the closed loop system to check if the closed loop system was set up correctly. The step responses shown below are identical, which proves that the closed loop system was indeed set up correctly.





Model Input: Reference Manipulator Motion

Now that the closed loop system is set up correctly to control for position, an input that resembles the trajectory of the ball can be fed into the control system. This allows us to see how the model controls for the angular velocity of the ball and angle of rotation for the arm with respect to a desired range of motion for the arm. Since the end goal here is to throw a strike, the desired motion of the manipulator was determined from the results of simple projectile motion equations that would ensure the ball would land in the strike zone given a certain initial velocity and height of release from the mound to the home plate.

The projectile motion equations used were as follows:

First, the time of the ball trajectory was calculated, where v is the velocity vector of the ball:

$$\boldsymbol{v_{ix} = vcos(\theta) \quad \rightarrow \quad x = vcos(\theta)t \quad \rightarrow \quad t = \frac{x}{vcos(\theta)}}$$

$$\boldsymbol{v_{iy} = vsin(\theta), \;\; y = v_{iy}t - \frac{1}{2}gt^2}$$

Combining these equations, we can solve for values of θ, v at which the ball must be released:

$$\boldsymbol{h_s - h_l = -xtan(\theta) - \frac{1}{2}g\left(\frac{x}{vcos(\theta)}\right)^2,}$$

$h_s = height\ of\ desired\ strike\ (vertical\ midpoint\ of\ strike\ box)$
$h_l = height\ of\ release\ point$

Next, the simple equation below was used to determine the desired angular velocity of the manipulator from the calculated linear velocity.

$$\boldsymbol{\omega} = \frac{\boldsymbol{v}}{\boldsymbol{l}}, \qquad l = length\ of\ the\ link$$

From these equations, which are executed in a MATLAB script that can be found in the appendix, the desired θ, ω to achieve a strike at a certain speed can be determined.

Gripper Dynamics

The final part of the architecture of the system was simulating the gripper dynamics, which essentially consisted of the 10ms that it takes for the gripper to actually release the ball. The idea behind simulating this was figuring out how much the arm rotates in 10ms at the desired release velocity, and then subtracting this angular displacement of the 10ms delay from the desired release theta calculated from projectile motion equations (this is the "threshold angle"). The logic to implement this was as follows:

```
if (angle >= thresholdAngle && angle<=desiredAngle && velocity >= desiredVelocity)
        flag up
else
        keep flag down
```

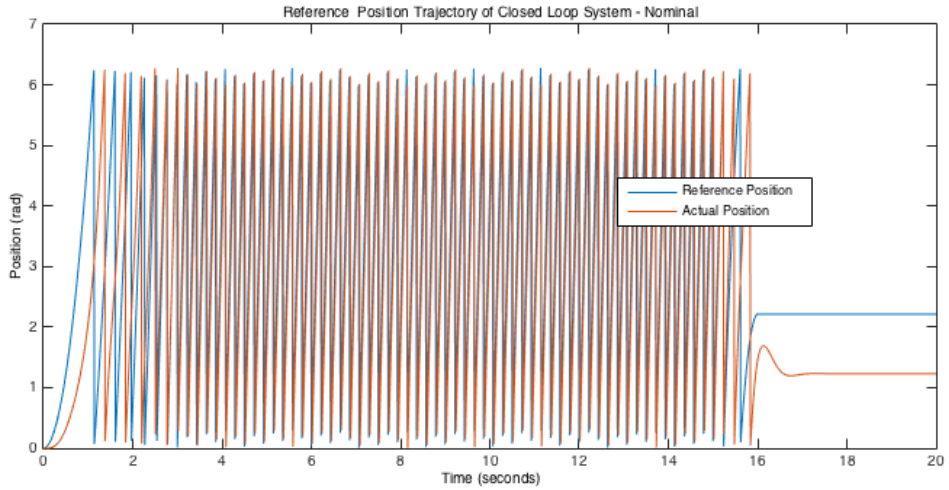
Here, once the angle falls in within the threshold angle and the desired angle and the velocity is great enough to throw a strike, a signal will flag up (and this signal will be delayed by 10ms in Simulink using a discrete delay block) at the time when the gripper should initiate release of the ball. Then, 10ms later, the gripper should actually release the ball at the desired velocity and desired angle.

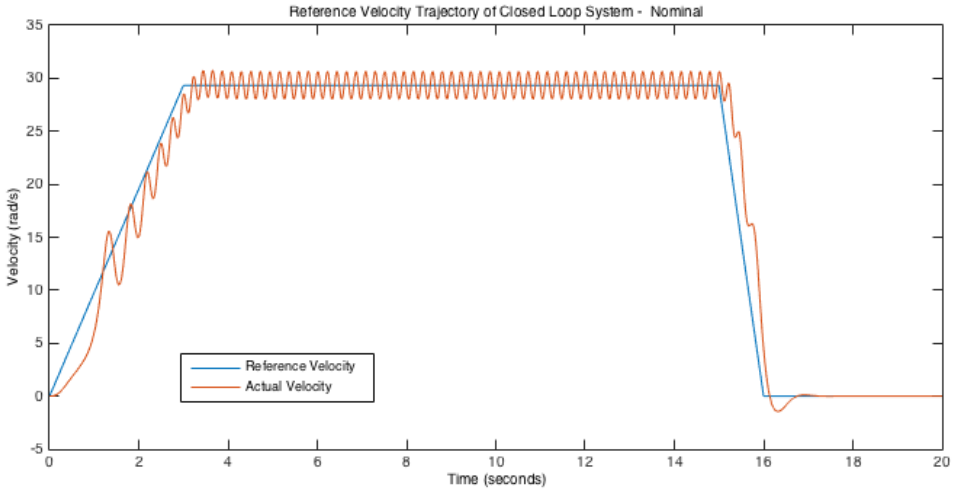
We can then see the velocity and angle of the ball at the time at which the signal flagged up (time of ball release), so we can check if the desired velocity and desired angle were reached at the time of release.

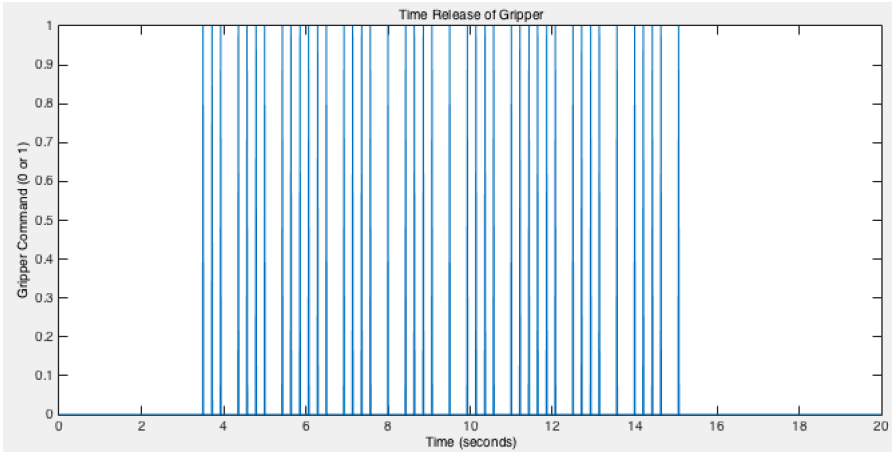
Nominal Trajectory

With all parts of the system simulation implemented, it is now possible to check the position control response to an input angular velocity trajectory of the manipulator. The angular velocity trajectory was created using a 1-D Lookup table and passed through an integrator block before feeding into the closed loop position control system, so that essentially both velocity and position could be controlled.

The results for the position and velocity control responses for a nominal trajectory are shown below:

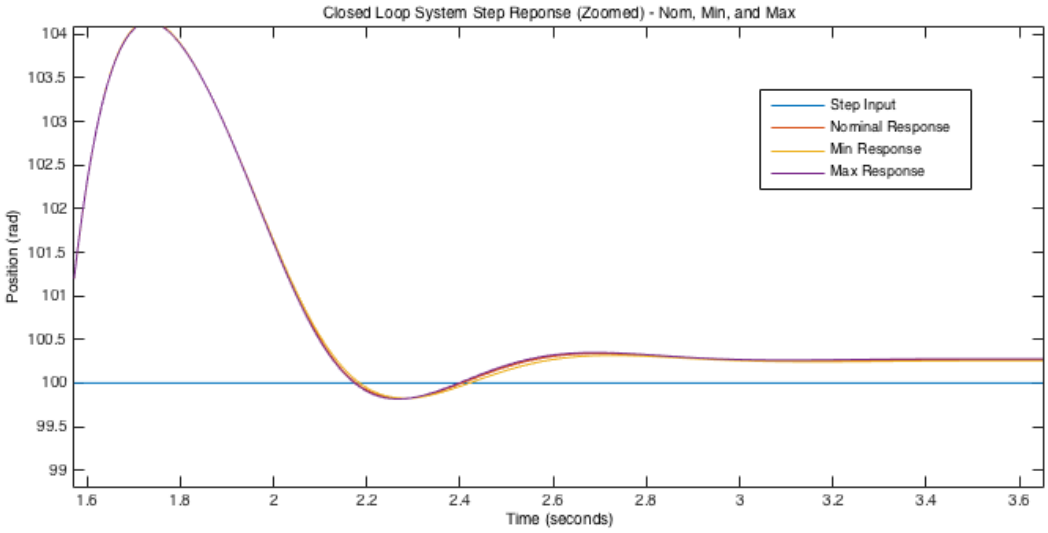

Reference Position Trajectory of Closed Loop System - Nominal
Position (rad)
Time (seconds)
Reference Position
Actual Position


Reference Velocity Trajectory of Closed Loop System - Nominal
Velocity (rad/s)
Time (seconds)
Reference Velocity
Actual Velocity

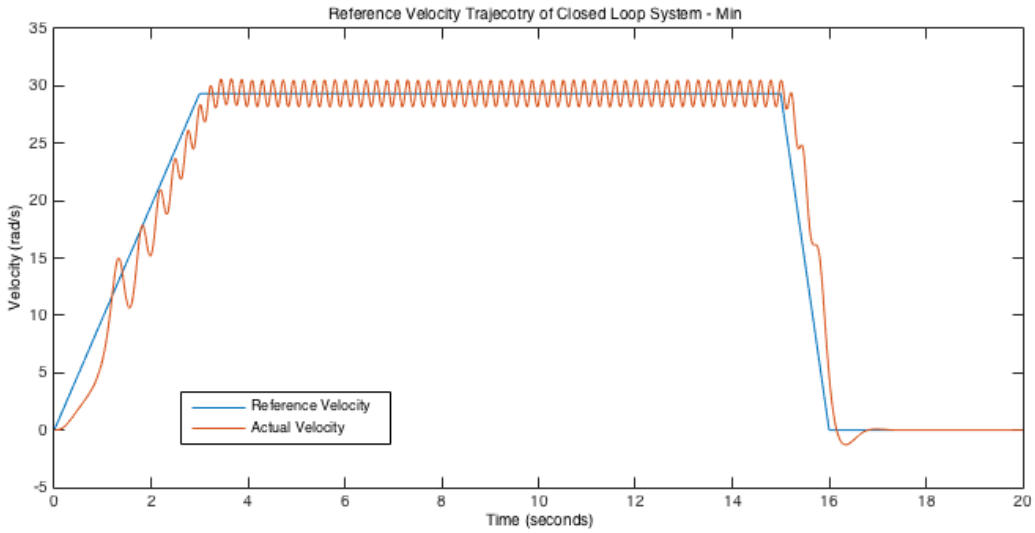

Time Release of Gripper
Gripper Command (0 or 1)
Time (seconds)

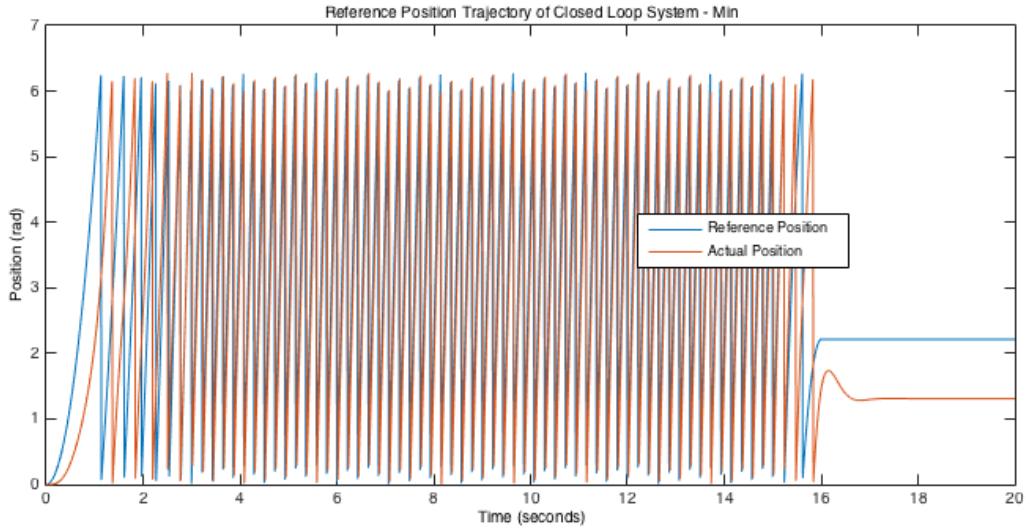
Uncertainty Analysis

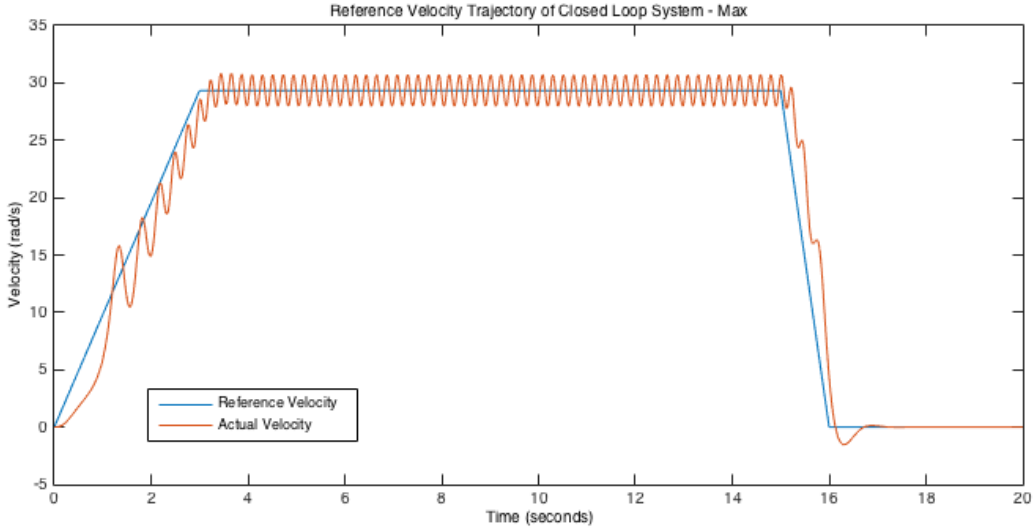
Before beginning the uncertainty analysis for trajectory responses, it is important to note the slight difference in step response for nominal, min, and max parameters.

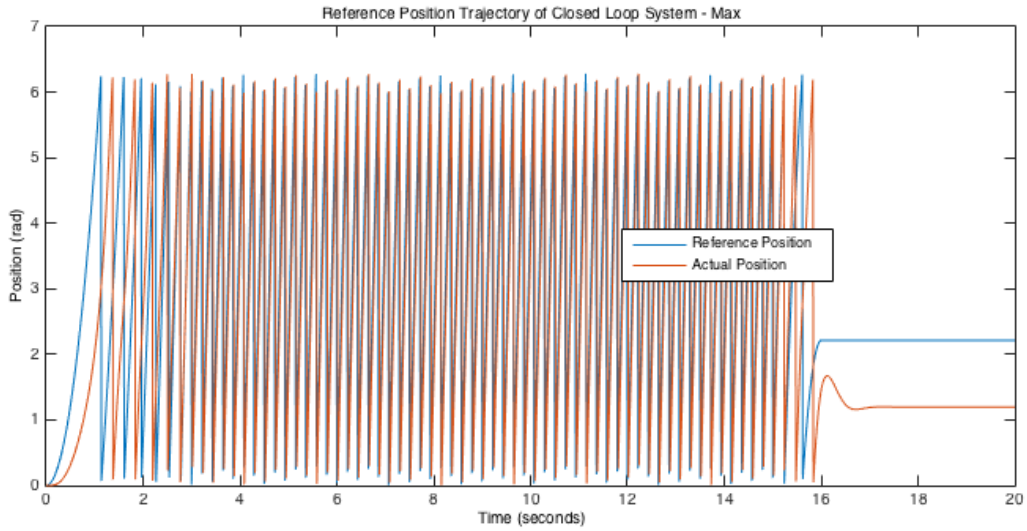


The position and velocity trajectory responses were found for min/max parameters for the model and are shown below:




Reference Position Trajectory of Closed Loop System - Min
Position (rad)
Time (seconds)
Reference Position
Actual Position


Reference Velocity Trajectory of Closed Loop System - Max
Velocity (rad/s)
Time (seconds)
Reference Velocity
Actual Velocity


Reference Position Trajectory of Closed Loop System - Max
Position (rad)
Time (seconds)
Reference Position
Actual Position

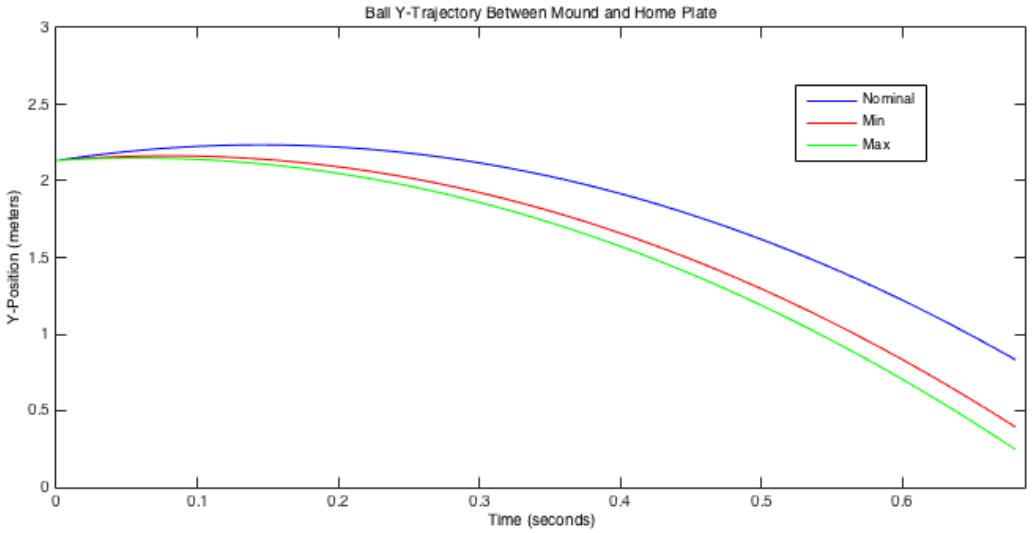
The scope showing the time release of the gripper is not shown for the min/max simulations, but a table below shows all values for time of release (t), angle of release ($\theta_{release}$), and angular velocity at release ($\omega_{release}$) in each scenario.

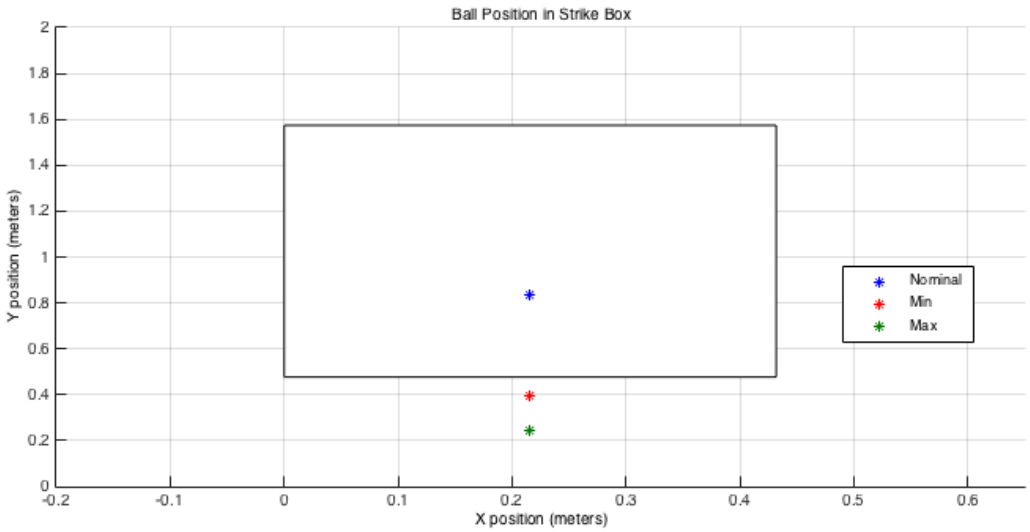
	Nominal	**Min**	**Max**
t (s)	3.5	3.5	3.5
$\boldsymbol{\theta_{release}}$ ***(rad)***	1.542	1.5178	1.55
$\boldsymbol{\omega_{release}}$ ***(rad/s)***	29.366	29.33	29.35

With these values now as inputs to the projectile motion equations described earlier, the resulting vertical height above home plate can be determined. The results of this process are shown below, where a strike was thrown under nominal conditions, but not under min or max conditions.

Strike Box Results

This image shows the y-position of the trajectory of the ball from point at which it is released from the manipulator to the time at which it crosses home plate. The final height of the ball as it crosses home plate can be seen even clearer in the strike box image, which is below the trajectory plot.





In conclusion, our manipulator can successfully throw a 60mph ball into the strike zone under nominal parameters. It fails to throw a strike under min/max parameters, as the angle of release and velocity of release are altered just enough to get the ball under the strike zone as it passes home plate. This highlights the sensitivity of our system.

Appendix: MATLAB Code

Gripper Release Function

```
function release = fcn(angle, velocity, threshold_angle,threshold_vel,
referenceAngle)
%#codegen

if (angle >= threshold_angle && angle<=referenceAngle && velocity >=
threshold_vel)
release=1;
else
release=0;
end
```

referenceAngle.m

Determines desired angle and desired angular velocity of arm given desired speed of pitch across home plate

```
function [omega, refAngle] =Projectile(v)

syms theta
x=18.4404;
hs=.79248;
hl=.9144*cos(theta)+1.2192;
g=9.8;
v=v*.44704;
linkLength=.9144;


refAngle= simplify(vpasolve(hs-hl==-x*tan(theta)-
.5*g*(x/(v*cos(theta)))^2)+pi/2);
omega=v/linkLength;
threshAngle= refAngle-omega*.01;


end
```

pitchTrajectory.m

Plots the y-position of the trajectory of the ball between the release point from the manipulator to home plate

```
theta = 1.5178-pi/2; %theta of motor at time of release (found from scope of
simulation)
w= 29.333; %angular velocity of motor at time of release (found from scope of
simulation)
hs=1.2192; %height of the 'stand' upon which robot sits
l=.9144; %length of link
g=9.8; %acceleration due to gravity
x=18.4404+l*sin(theta); %distance between end effector and home plate
tTotal=x/(l*w*cos(theta)); %total time it takes ball to reach home plate
y=zeros(1,length(0:.01:tTotal));
length(y)
k=1;
for t=0:.01:tTotal
    y(k)=-.5*g*t^2-l*w*sin(theta)*t+(hs+l*cos(theta));
    k=k+1;
end

plot(0:0.01:tTotal,y)
```

strikeBox.m

Plots the y-position of the ball as it crosses home plate in reference to the strike box

```
close all; clc;

rectangle('position',[0 0.4768 0.4318 1.09728],'facecolor','w');
   %text(0.5,0.5,'A Box','HorizontalAlignment','center');
   axis([-.2 .65 0 2]);
   hold on;
   plot(.2159,.8327,'b*')
   plot(.2159,.3932,'b*')
   plot(.2159,.2471,'b*')
   grid on;
%0.8327††††0.3932††††0.2471
```