# Ball on Plate

**Group 7**

Ryan Brewer

Andrew Renteria

Alex Robie

Alex Roth

Josh Verdon

MECA 482 - 7534

Control Systems Design

December 17, 2021

Department of Mechanical and Mechatronic Engineering and Advanced Manufacturing

California State University, Chico

Chico, California 95929

## 1. Introduction

The purpose of this system is to balance a ball on a plate using two stepper motors attached in line to the bottom side of the plate. The ball would be placed on the plate, the plate would move in the desired axis to center the ball on the plate. The biggest challenge for this project is the plate positioning as well as the ball balance math. These must work in kind to produce a position on the plate. The plate is linked via X and Y axis.

## 2. Modeling

The mathematical model for the Ball and Plate system is shown below in Figure 1. The modeling utilized for the X-axis was replicated for the Y-axis due to symmetry in the model. The nonlinear equation of motion for the ball given below was calculated from further examination of the mathematical representation.

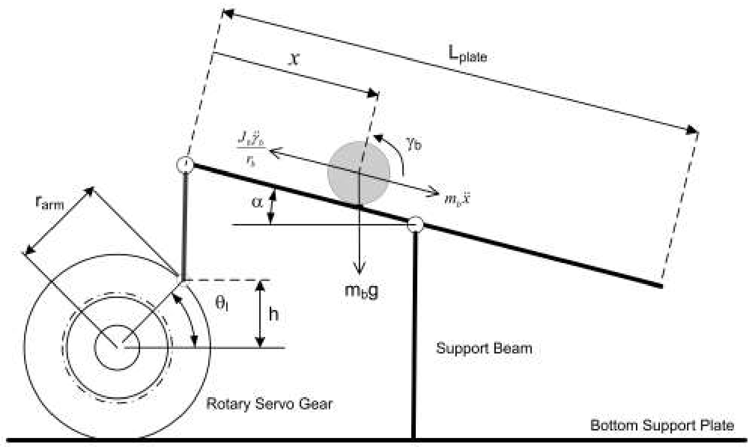


**Figure 1.** Mathematical Model of Ball on Plate

$$\ddot{x}(t) = \frac{m_b sin\alpha(t) r_b^2}{m_b r_b^2 + J_b} \qquad \text{Eq. (1)}$$

The arm of the motor and its relationship to the angle of the plate can be utilized with small angle approximation to form a near linearization equation.

$$\ddot{x}(t) = \frac{2 m_b g r_{arm} r_b^2}{L_{plate}(m_b r_b^2 + J_b)} \theta_l(t) \qquad \text{Eq. (2)}$$

From this point, the Laplace of the linear equation can be used to determine the transfer function.

$$\frac{x(s)}{\theta(s)} = \frac{2 m_b g r_{arm} r_b^2}{L_{plate}(m_b r_b^2 + J_b) s^2} \qquad \text{Eq. (3)}$$

Using the measured ball position, the servo shaft angle is computed to find the desired ball position. The block diagram controls the servo position with the use of proportional and derivative gain.

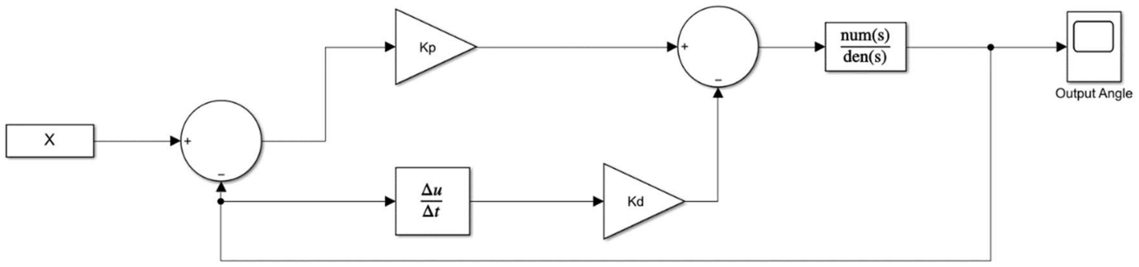


**Figure 2.** Control System Block Diagram

With the specifications of less than 5% overshoot and a settling time of less than three seconds, the equations for the damping ratio and natural frequency can be determined.

$$\zeta = \frac{-\ln\left(\frac{\%OS}{100}\right)}{\left(\pi^2+\ln^2\left(\frac{\%OS}{100}\right)\right)^{1/2}} \qquad \text{Eq. (4)}$$

$$\omega_n = \frac{4}{\zeta T_s} \qquad \text{Eq. (5)}$$

Using the values obtained from Eq. (4) and Eq. (5), the proportional and derivative gain can be calculated.

$$k_p = \omega_n^2 \frac{L_{plate}(m_b r_b^2 + J_b)}{m_b g r_{arm} r_b^2} \qquad \text{Eq. (6)}$$

$$k_d = \zeta\omega_n \frac{L_{plate}(m_b r_b^2 + J_b)}{m_b g r_{arm} r_b^2} \qquad \text{Eq. (7)}$$

**Figure 3** shown below represents the requirements of a working Ball and Plate system.

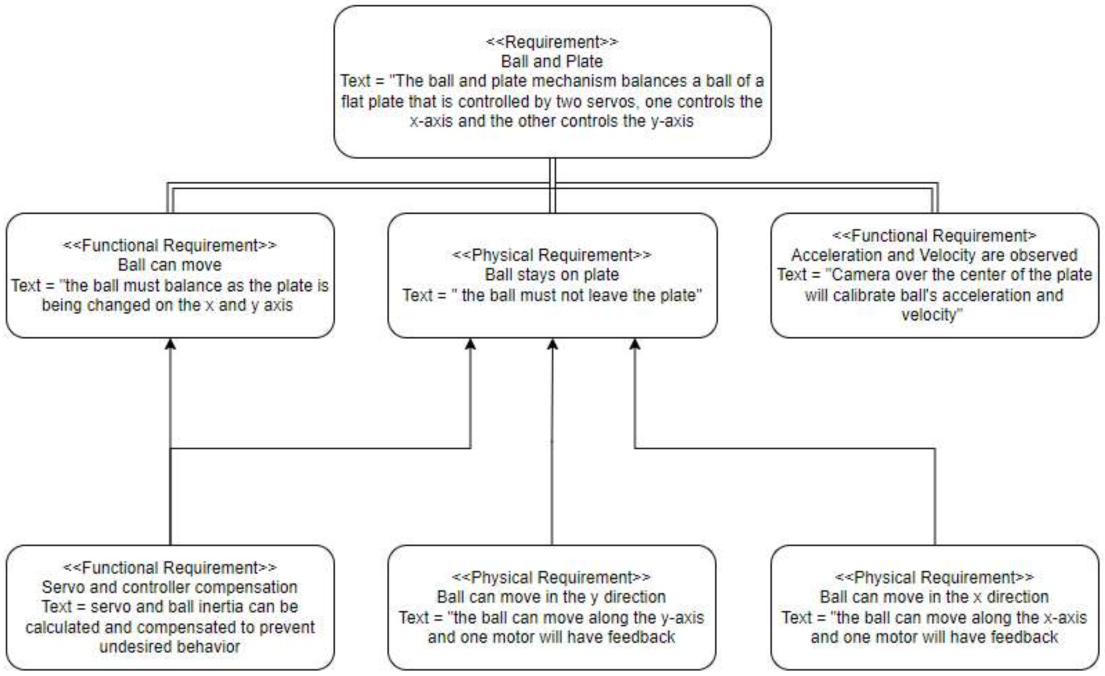
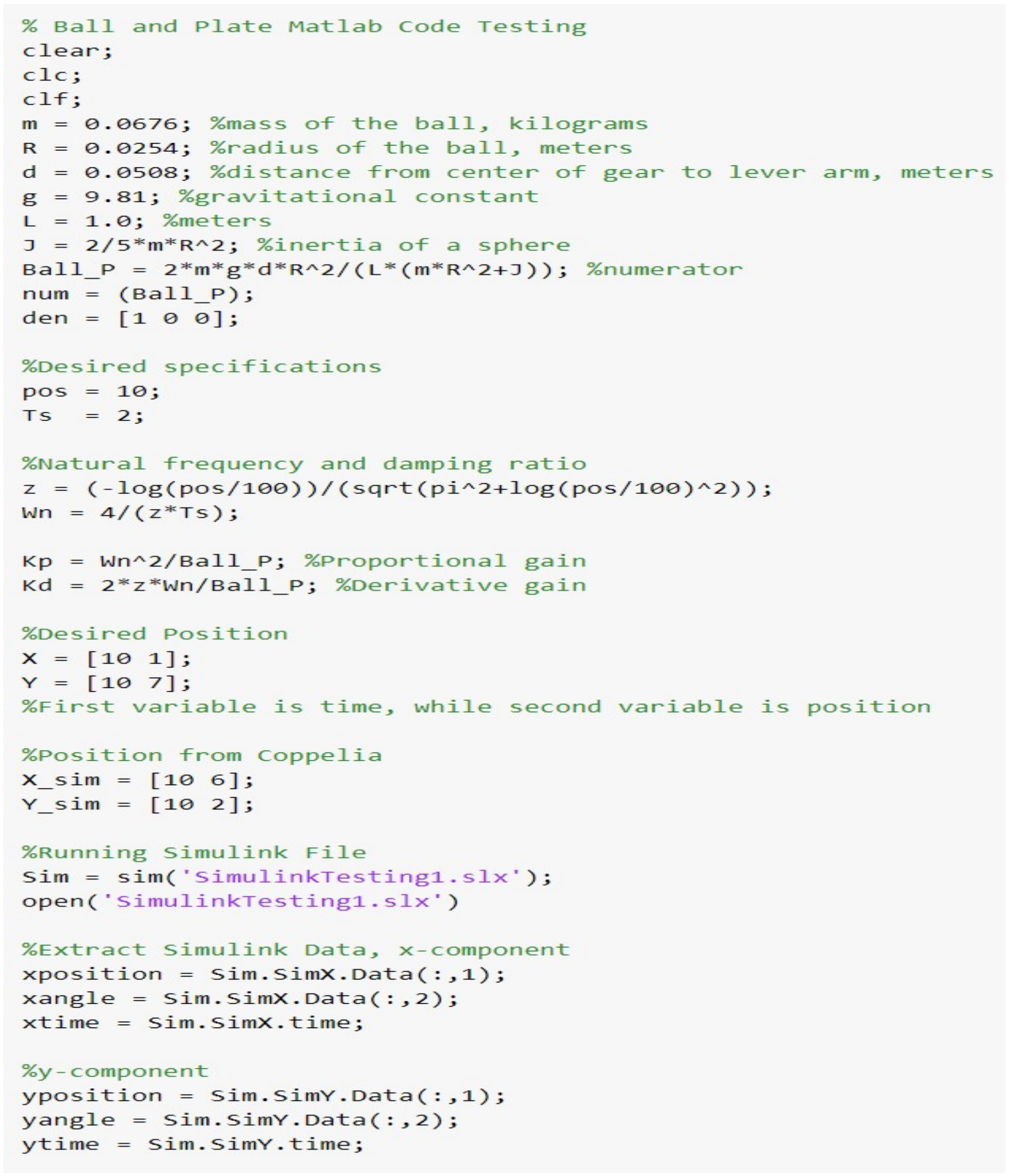

<<Requirement>>
Ball and Plate
Text = "The ball and plate mechanism balances a ball of a flat plate that is controlled by two servos, one controls the x-axis and the other controls the y-axis
<<Functional Requirement>>
Ball can move
Text = "the ball must balance as the plate is being changed on the x and y axis
<<Physical Requirement>>
Ball stays on plate
Text = " the ball must not leave the plate"
<<Functional Requirement>
Acceleration and Velocity are observed
Text = "Camera over the center of the plate will calibrate ball's acceleration and velocity"
<<Functional Requirement>>
Servo and controller compensation
Text = servo and ball inertia can be calculated and compensated to prevent undesired behavior
<<Physical Requirement>>
Ball can move in the y direction
Text = "the ball can move along the y-axis and one motor will have feedback
<<Physical Requirement>>
Ball can move in the x direction
Text = "the ball can move along the x-axis and one motor will have feedback

**Figure 3.** Working System Requirements

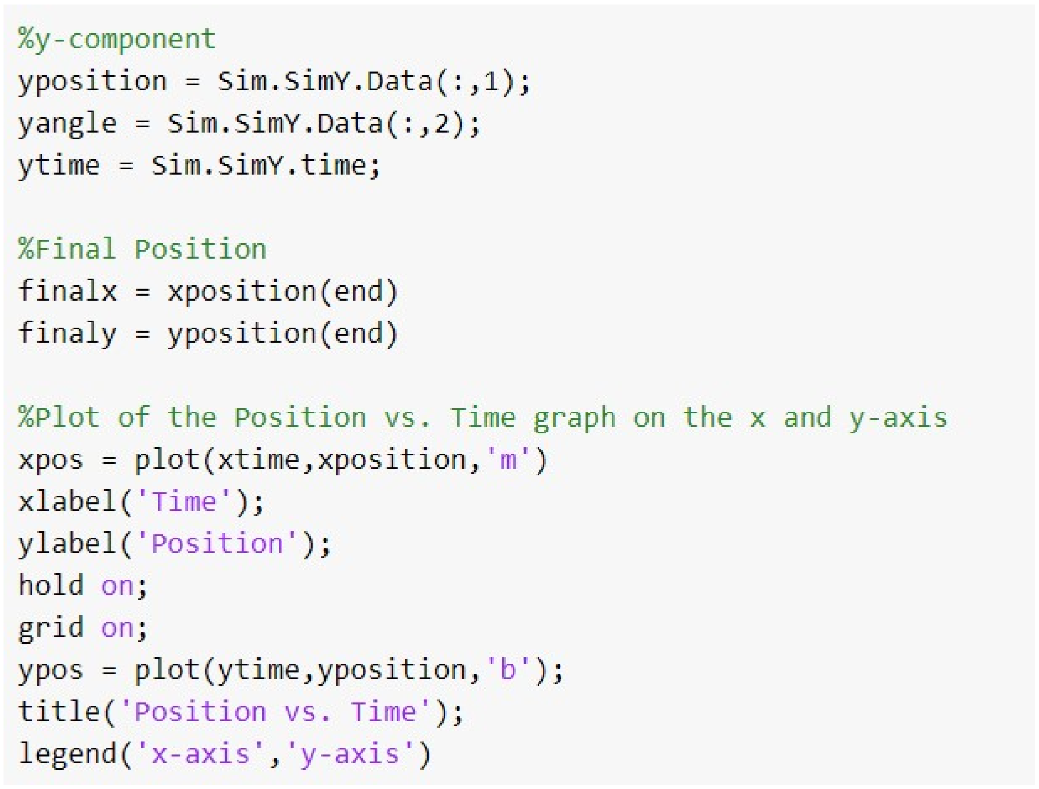
## 3. Controller Design and Simulation

**Figure 4** and **Figure 5** given below contains the MATLAB code implemented to test the mathematical model. The interpretation of the code is as follows: Clear all possible variables, generate a link to Coppelia, create and set up the physical parameters of the system, run the code if there is a valid connection to Coppelia, and display a "Fail Connection Error" if connection to Coppelia is not secured. If connected to Coppelia, the code will continue to run in the following order: Initialization of Simulink, set up handles, gather required Coppelia data, update Simulink values, run the Simulink, output the data to Coppelia to adjust motor angle, and repeat the process so long as Coppelia is running.



```
% Ball and Plate Matlab Code Testing
clear;
clc;
clf;
m = 0.0676; %mass of the ball, kilograms
R = 0.0254; %radius of the ball, meters
d = 0.0508; %distance from center of gear to lever arm, meters
g = 9.81; %gravitational constant
L = 1.0; %meters
J = 2/5*m*R^2; %inertia of a sphere
Ball_P = 2*m*g*d*R^2/(L*(m*R^2+J)); %numerator
num = (Ball_P);
den = [1 0 0];

%Desired specifications
pos = 10;
Ts  = 2;

%Natural frequency and damping ratio
z = (-log(pos/100))/(sqrt(pi^2+log(pos/100)^2));
Wn = 4/(z*Ts);

Kp = Wn^2/Ball_P; %Proportional gain
Kd = 2*z*Wn/Ball_P; %Derivative gain

%Desired Position
X = [10 1];
Y = [10 7];
%First variable is time, while second variable is position

%Position from Coppelia
X_sim = [10 6];
Y_sim = [10 2];

%Running Simulink File
Sim = sim('SimulinkTesting1.slx');
open('SimulinkTesting1.slx')

%Extract Simulink Data, x-component
xposition = Sim.SimX.Data(:,1);
xangle = Sim.SimX.Data(:,2);
xtime = Sim.SimX.time;

%y-component
yposition = Sim.SimY.Data(:,1);
yangle = Sim.SimY.Data(:,2);
ytime = Sim.SimY.time;
```

**Figure 4.** MATLAB Code



```
%y-component
yposition = Sim.SimY.Data(:,1);
yangle = Sim.SimY.Data(:,2);
ytime = Sim.SimY.time;

%Final Position
finalx = xposition(end)
finaly = yposition(end)

%Plot of the Position vs. Time graph on the x and y-axis
xpos = plot(xtime,xposition,'m')
xlabel('Time');
ylabel('Position');
hold on;
grid on;
ypos = plot(ytime,yposition,'b');
title('Position vs. Time');
legend('x-axis','y-axis')
```

**Figure 5.** MATLAB Code

The output data from MATLAB depicting position vs. time in respect to the ball of the system is given by the graph below. Note the x- and y- positions of the ball are listed independently on the graph and the y-position rapidly increases in relation to the x-position.

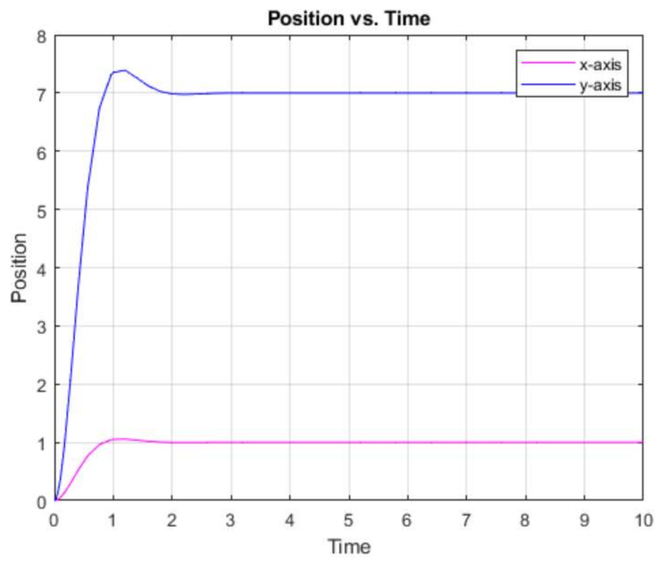


**Figure 6.** Ball Position vs Time

The final Simulink result file implemented for the project is given below. The first step is to enter a desired position for the ball. Once the balls position is obtained, the value is input into the left end. The data obtained is from the Coppelia and MATLAB codes elaborated on in previous sections. The next step is to calculate the motor angle, which results directly from the Simulink simulation. The final step in the Simulink is the reverse of a previous step; the calculated motor angle value is transferred back into the Coppelia and MATLAB code, alerting the motor of the angle required.

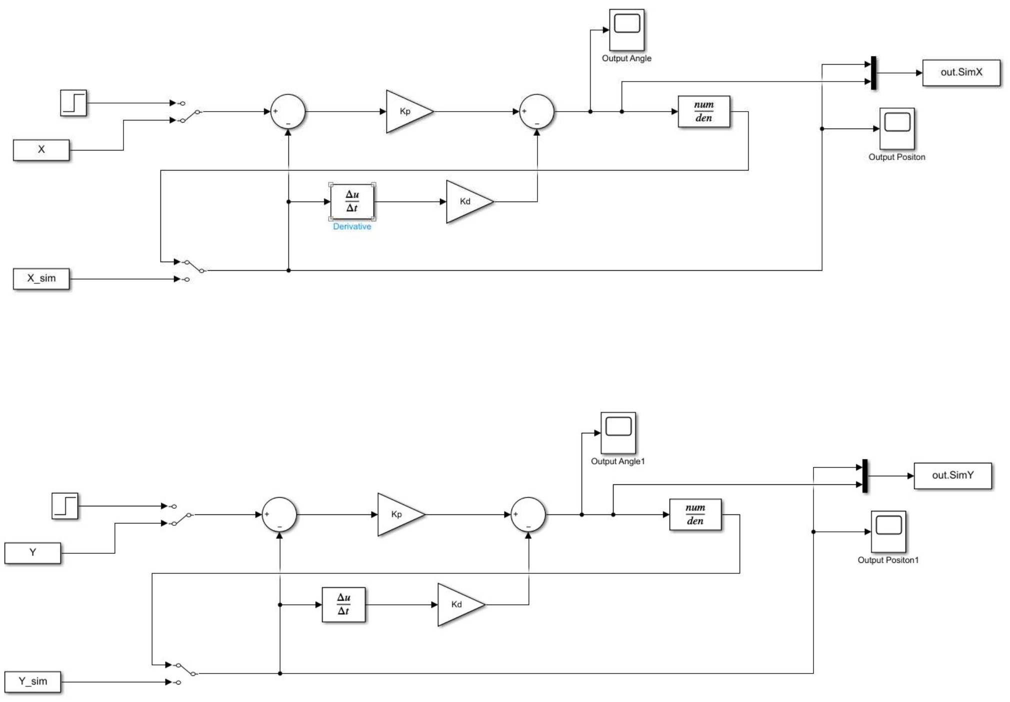


**Figure 7.** Simulink

In order to physically model the ball on plate system, CoppeliaSim was utilized. The model used multiple joint connections as well as dynamic objects. The plate was held in place using a series of rods and joints. More specifically, a spherical joint served as the direct connection to the plate, allowing the plate to swivel and rotate in all directions. Connected to the spherical joint are two support rods with linearly sliding prismatic joints. The joint serves as a connection for the two support rods and allows the plate vertical motion. The combination of unique joints and rods allows the system three degrees of freedom. The motion of the support rods is provided by two motors. The final modeled object is a vision sensor positioned above the plate. This location provides a birds-eye view of the plate which allows for sensing ball position and movement.

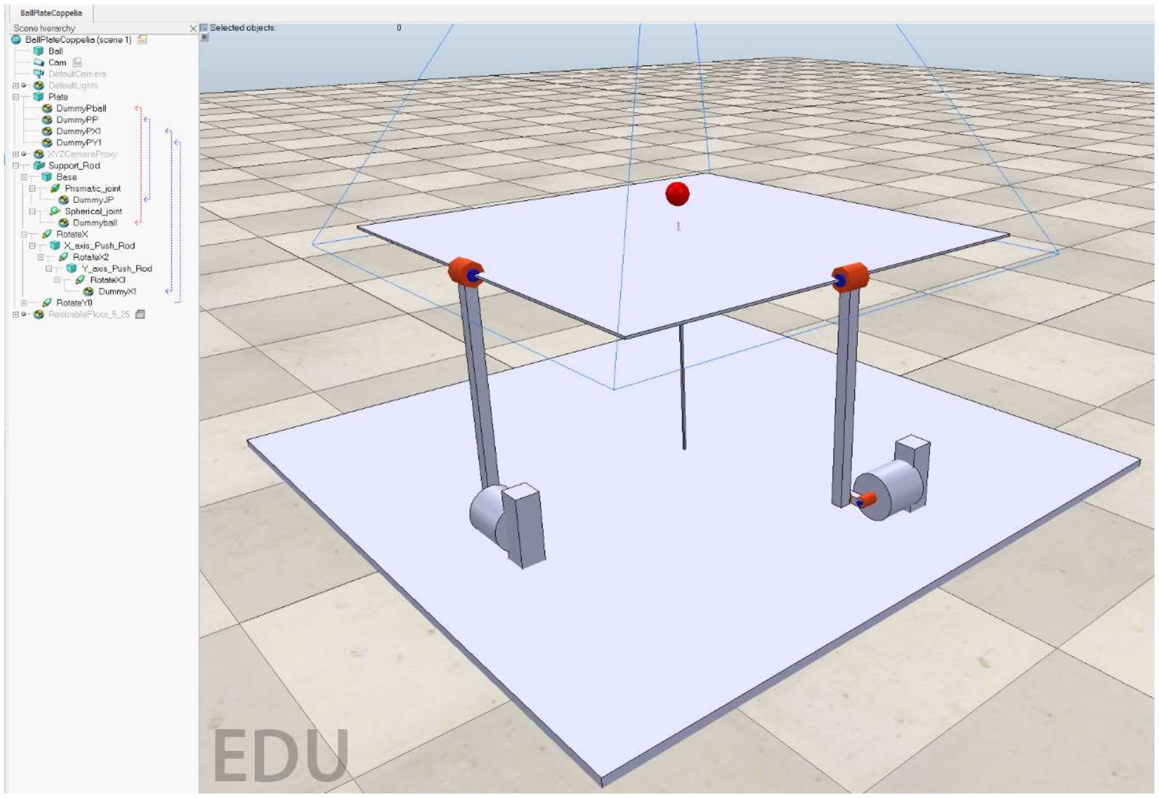


**Figure 8.** CoppeliaSim Model

## 4. Conclusion

The ball and plate system allows further exploration of elementary concepts in the field of control systems engineering. Specifically, the concepts of modeling, controller design, and simulation were implemented to model the system. Through mathematical calculations, MATLAB coding, and its relationship to Simulink and Coppelia, an accurate system was designed. The biggest issue resulted from attempting to connect the various software. The RemoteApi server connection failed to connect to Coppelia, requiring manual testing to verify the design. The full feedback loop between MATLAB, Simulink, and Coppelia was incomplete due to the aforementioned error.

## Appendix A

```matlab
% Ball and Plate Matlab Code
clear;
clc;
clf;
sim=remApi('remoteApi'); % using file (remoteApiProto.m)
    sim.simxFinish(-1); % close all opened connections
    clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5);

%System properties
m = 0.0676; %mass of the ball, kilograms
R = 0.0254; %radius of the ball, meters
g = 9.81; %gravitational constant
L = 1.0; %meters
d = 0.0508; %distance from center of gear to lever arm, meters
J = 2/5*m*R^2; %inertia of a sphere

%Desired specifications
pos = 10;
Ts  = 6;

%Natural frequency and damping ratio
z = (-log(pos/100))/(sqrt(pi^2+log(pos/100)^2));
Wn = 4/(z*Ts);

%Transfer function of ball position and gear angle
Ball_P = 2*m*g*d*R^2/(L*(m*R^2+J)); %numerator
num = [Ball_P];
den = [1 0 0];
Kp = Wn^2/Ball_P; %Proportional gain
Kd = 2*z*Wn/Ball_P; %Derivative gain

if (clientID>-1)
        disp('Connected to remote API server');
        set_param('Sysmodel','SimulationCommand','start');
        h=[0,0,0];
    %allocating joint information
    [r,h(1)]=sim.simxGetObjectHandle(clientID,'Motorx',sim.simx_opmode_blocking);
    [r,h(2)]=sim.simxGetObjectHandle(clientID,'MotorY',sim.simx_opmode_blocking);
    [r,h(3)]=sim.simxGetObjectHandle(clientID,'ball',sim.simx_opmode_blocking);

    while true
        [r,pball]=sim.simxGetObjectPosition(clientID,h(3),-1,sim.simx_opmode_blocking);
           xcoord=pball(1);
           ycoord=pball(2);
```

## Appendix A

```
            %Coppelia to simulink
            set_param('Sysmodel/xpos','Value',num2str(xcoord));
            pause(0.005)
            set_param('Sysmodel/ypos','Value',num2str(ycoord));
            pause(0.005)
            set_param('Sysmodel','SimulationCommand','start');

            %Calculated output for desired position
            A=out.SimX.Data(:,1);
            xaxis=A(1);
            B=out.SimY.Data(:,1);
            yaxis=B(1);

            %Set positions
            sim.simxSetJointTargetPosition(clientID,h(1),xaxis,sim.simx_opmode_streaming);
            sim.simxSetJointTargetPosition(clientID,h(2),yaxis,sim.simx_opmode_streaming);


    end
         else
        disp('Failed connecting to remote API server');
end

sim.delete();
disp('Program ended');
```

## References

Bayar, G. “Theoretical and Experimental Investigation of Backlash Effects on a 2-DOF Robotic Balancing Table.” Mechanika (Kaunas, Lithuania : 1995), vol. 23, no. 1, Kauno Technologijos Universitetas, 2017, https://doi.org/10.5755/j01.mech.23.1.14008.

*2 DOF Ball Balancer Workbook*, https://www.made-for-science.com/en/quanser/?df=made-for-science-quanser-2-dof-ball-balancer-coursewarestud-matlab.pdf.