Control Methods for the “Papi Rubber” Game

Peng Mun Siew & Joe Viavattine

ME8281 Final Project Report

Perry Li

5/13/2016

Table of Contents:

1. Introduction and System Definition:

Many dynamic, physical systems that we desire to control can be modeled as simple linear time-invariant state space systems. These systems often allow for the implementation of several control techniques. This report will address some of these convenient and effective open and closed-loop control methods for a an interesting state space system, a model of the "Papi Rubber" game. First, two open loop methods, least norm control and input shaping, will be derived and discussed, followed by the least squares estimation for state estimation and finally a closed loop method of full state feedback will be applied.

In this game, the player moves the papi with his or her finger to swing the spikey ball attached with a spring into moving enemies. A diagram of the game is shown in figure 1. The ball was modeled with a mass of 1 kg, the spring with a spring constant of 1 N/m, and the ball with a viscous friction coefficient of 2 N/(m/s). The resulting matrices A and B describing the system are listed in equations 1 through 3.

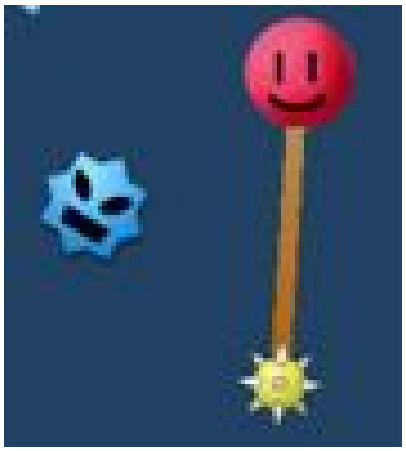


Figure 1: "Papi Rubber" game elements

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{k}{m} & 0 & -\frac{k}{m} & 0 & -\frac{b}{m} & 0 \\ 0 & \frac{k}{m} & 0 & -\frac{k}{m} & 0 & -\frac{b}{m} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & -2 \end{bmatrix} \quad (1)$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (2)$$

$$\frac{d}{dt}\begin{bmatrix} x_p \\ y_p \\ x_b \\ y_b \\ \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & -2 \end{bmatrix}\begin{bmatrix} x_p \\ y_p \\ x_b \\ y_b \\ \dot{x}_b \\ \dot{y}_b \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} u_x \\ u_y \end{bmatrix} \quad (3)$$

2. Least Norm Control:

System Modeling, Problem Definition, Design, and Analysis:

Least norm control is an elegant control method that uses the reachability map of a system to calculate the optimal control sequence for a given cost function (equation 4) to transfer between two sets of states. However, this method is rarely applied because most systems are either too large or include additional complexities which can't be modeled. In addition, the non-recursive formulation is not robust to disturbances. The development of a least norm controller is straightforward and given by equations 8 and 9, where 9 is the reachability gramian from the initial to final times. Both of these equations use the state transition matrix which was derived symbolically in MATLAB and a cost function R(t). Three different R(t) functions were used in the simulations (equations 5-7): a constant, a quadratic function penalizing input at the beginning and end of the control period, and a quadratic function penalizing input in the middle of the control period. The resulting papi and ball states and trajectories for each cost function are shown for final times of 5 and 10 seconds in the next section. All of the final states were set to [10;8;10;10;0;0] to compare the influence of cost functions and total time; however, any

combination of states could be achieved at any time because the system is controllable and the gramian is full rank.

$$J = \int_{t_o}^{Tf} R(t)u^T(t)u(t)dt \qquad (4)$$

$$R_1(t) = 1 \qquad (5)$$

$$R_2(t) = 1 + t(T_f - t) \qquad (6)$$

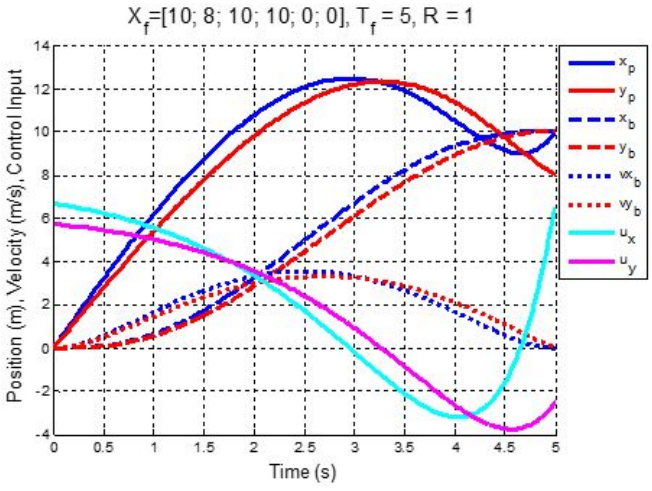
$$R_3(t) = 1 + \left(\frac{T_f}{2} - t\right)^2 \qquad (7)$$

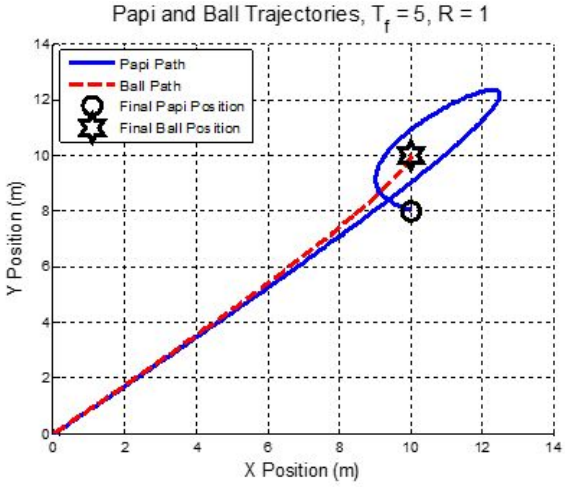
$$u_{opt} = R(t)^{-1}B^T\varphi(T_f, t)W^{-1}_{r[t_o,T_f]}\left(X_f - \varphi(T_f, t_0)X_0\right) \qquad (8)$$

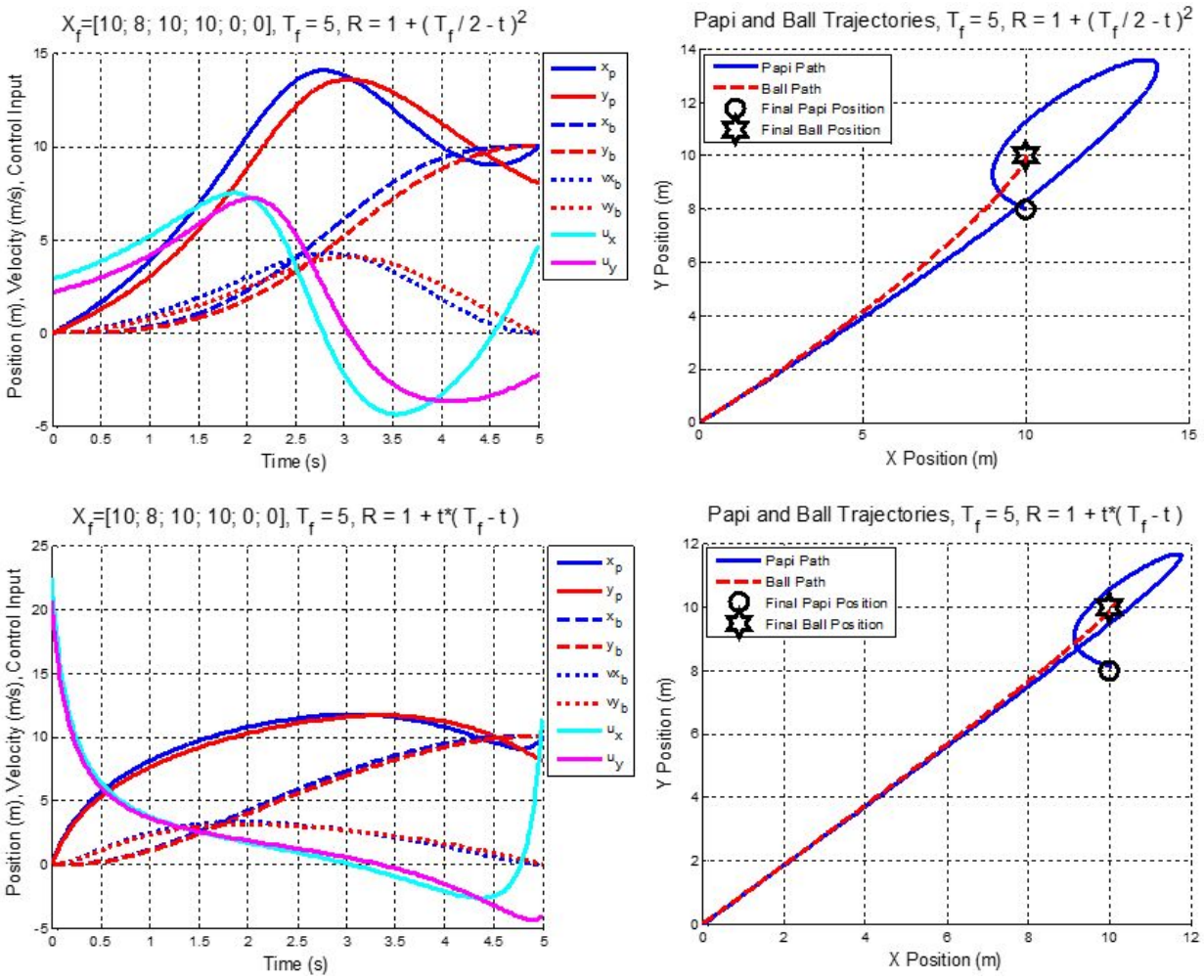
$$W_{r[t_o,T_f]} = \int_{t_o}^{T_f} \varphi(T_f, t)BR(t)^{-1}B^T\,\varphi^T(T_f, t)dt \qquad (9)$$

$$\varphi(t,0) = \begin{bmatrix} 1 & 0 & a & 0 & b & o \\ 0 & 1 & 0 & a & 0 & b \\ a & 0 & c & 0 & d & 0 \\ 0 & a & 0 & c & 0 & d \\ b & 0 & d & 0 & f & 0 \\ 0 & b & 0 & d & 0 & f \end{bmatrix} \quad where \begin{Bmatrix} a = 1 - te^{-t} \\ b = te^{-t} \\ c = (e^{-t} + te^{-t} - 1)^2 \\ d = -te^{-t}(e^{-t} + te^{-t} - 1) \\ f = t^2e^{-2t} \end{Bmatrix} \qquad (10)$$

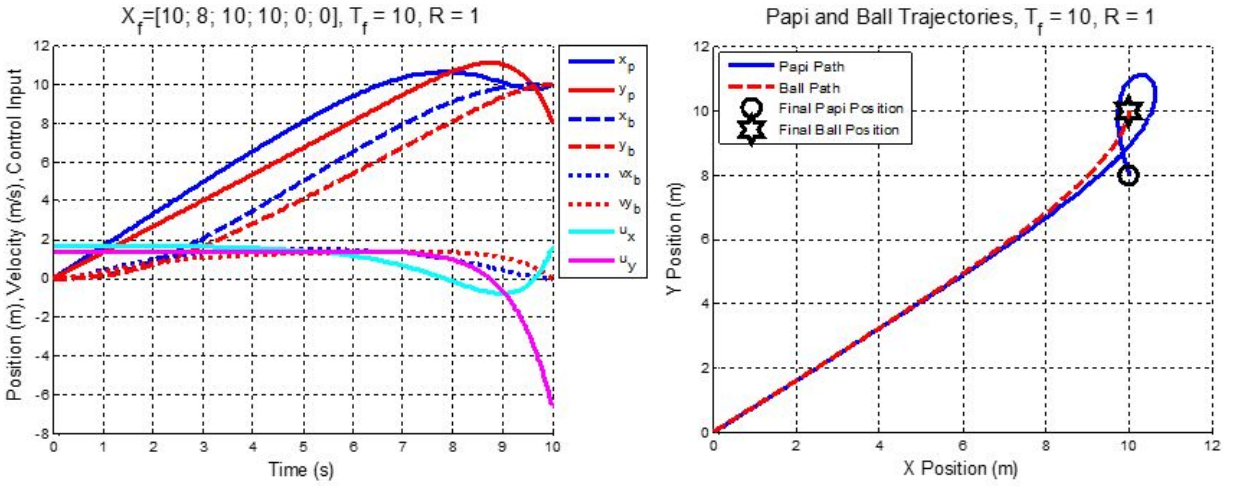
Simulations:

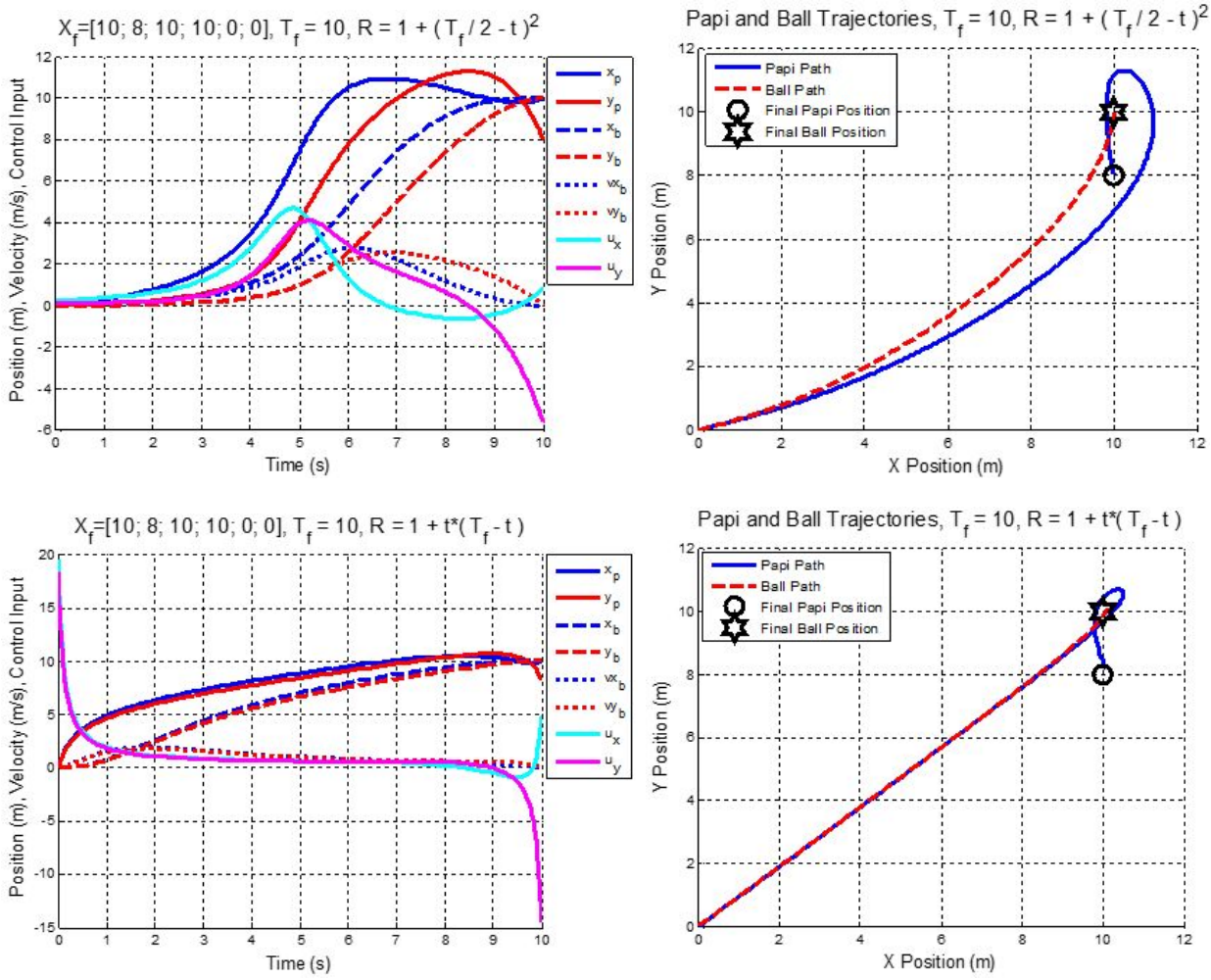






Figures 7-12: Papi and ball states and trajectories and control input for initial state Xi = [0;0;0;0;0;0], final state Xf = [10;8;10;10;0;0], and Tf = 5 seconds for each cost function R(t)





Figures 7-12: Papi and ball states and trajectories and control input for initial state Xi = [0;0;0;0;0;0], final state Xf = [10;8;10;10;0;0], and Tf = 10 seconds for each cost function R(t)

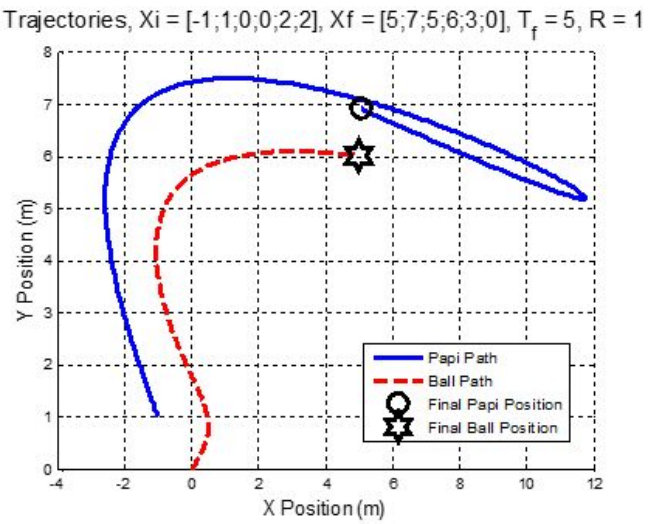


Figure 13: Papi and ball trajectories for non-zero initial state initial state Xi = [-1;1;0;0;2;2], final state Xf = [5;7;5;6;3;0], and Tf = 5 seconds for R(t) = 1

Discussion of Results:

As expected, the most input occurs during the cheaper time periods of the quadratic cost functions. These patterns become more pronounced with the larger final times because of the larger range in cost function values. The input of the two simulations with a cost function of 1 are also shaped similarly upon close inspection. In both cases it starts positive to accelerate the ball and decreases just as the papi approaches its peak position to reach the final states while pulling the overdamped ball. This extra input at the end of the control period is characteristic of the other simulations as well, especially those with the larger final times. Finally, there is a significant difference between the shape of the responses with cost functions expensive during the ends of the control period. In the 10 second case, a large impulse is applied in the middle to begin accelerating the ball, while this input is applied immediately for the 5 second case. Figure 13 shows the trajectory of the ball and papi with nonzero initial states.

To summarize these observations, final time is another important state that must be optimized with the other states to reduce cost. Least norm minimizes the cost between two sets of states, but it doesn't determine unknown final states to minimize the cost to achieve a select few important final states, such as the position of the ball. If this program were being developed

further to maximize performance by reducing cost and time to hit an enemy or multiple enemies, algorithms would have to be designed to determine optimal final times, papi positions, and ball velocities considering for a given final ball position considering the initial states and future enemy trajectories.

3. Input Shaping:

System Modeling and Problem Definition:

Input shapers are an open loop control method used to eliminate the oscillation of flexible modes in multi-degree of freedom systems after a short period of time. An input shaper is made from two impulses of equal time periods. The second impulse is designed to start at time 2 defined by equation 15, where time 1 is the time of the beginning of the first impulse, and the amplitude of the second impulse is defined by equation 16.

In order to apply an input shaper to this system, the stiffness of the spring had to be increased to create an underdamped system. A value of 37 N/m was chosen.

$$m\ddot{x}_b = k(x_p - x_b) - b\dot{x}_b \qquad (11)$$

$$\frac{X_b(s)}{X_p(s)} = \frac{k}{m_b s^2 + bs + k} = \frac{1}{s^2 + 2s + 1} = \frac{1}{(s+1)^2} \rightarrow Overdamped \qquad (12)$$

$$Increasing\ k\ to\ 37: \frac{X_b(s)}{X_p(s)} = \frac{1}{(s + (1 + 6i))(s + (1 - 6i))} \rightarrow Underdamped \qquad (13)$$

$$Poles: \quad s = -\sigma \pm \omega_d j \qquad (14)$$

$$t_2 = \frac{\pi}{\omega_d} + t_1 = \frac{\pi}{6} + t_1 \qquad (15)$$

$$A_2 = A_1 e^{-\sigma t_2} = A_1 e^{-\frac{\pi}{6}} \qquad (16)$$

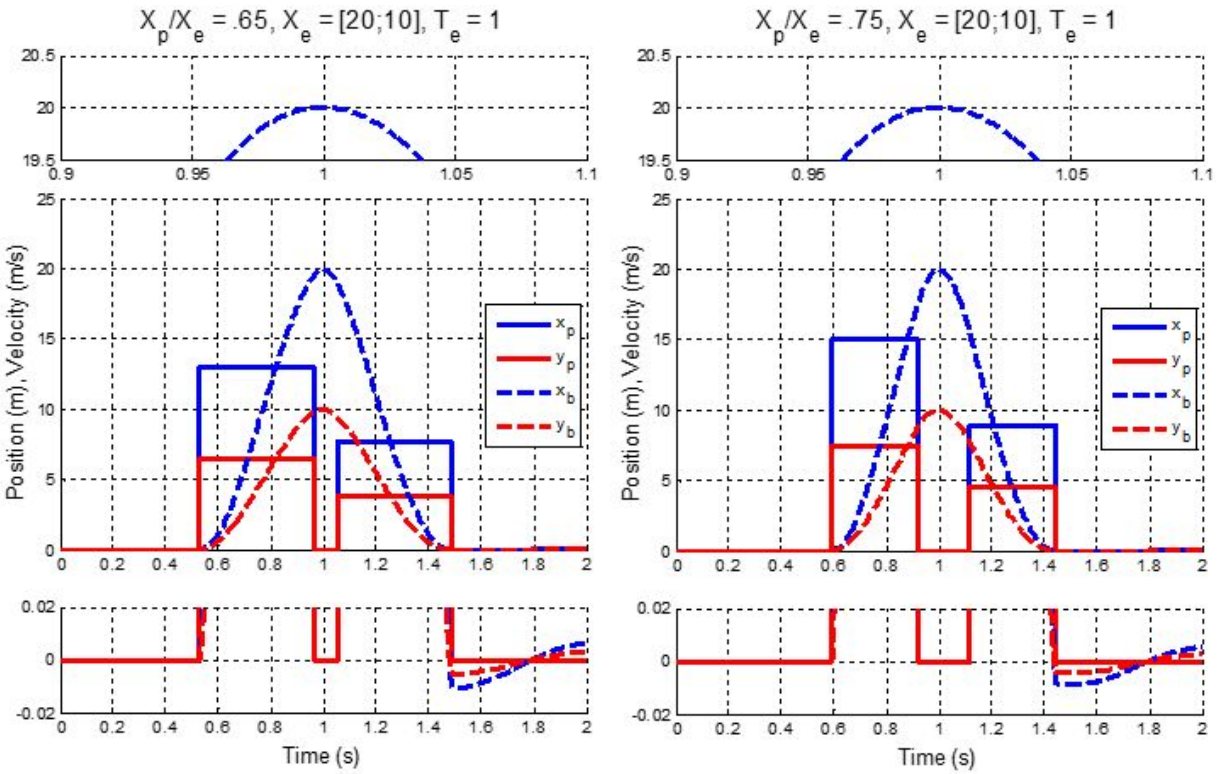
Design and Analysis:

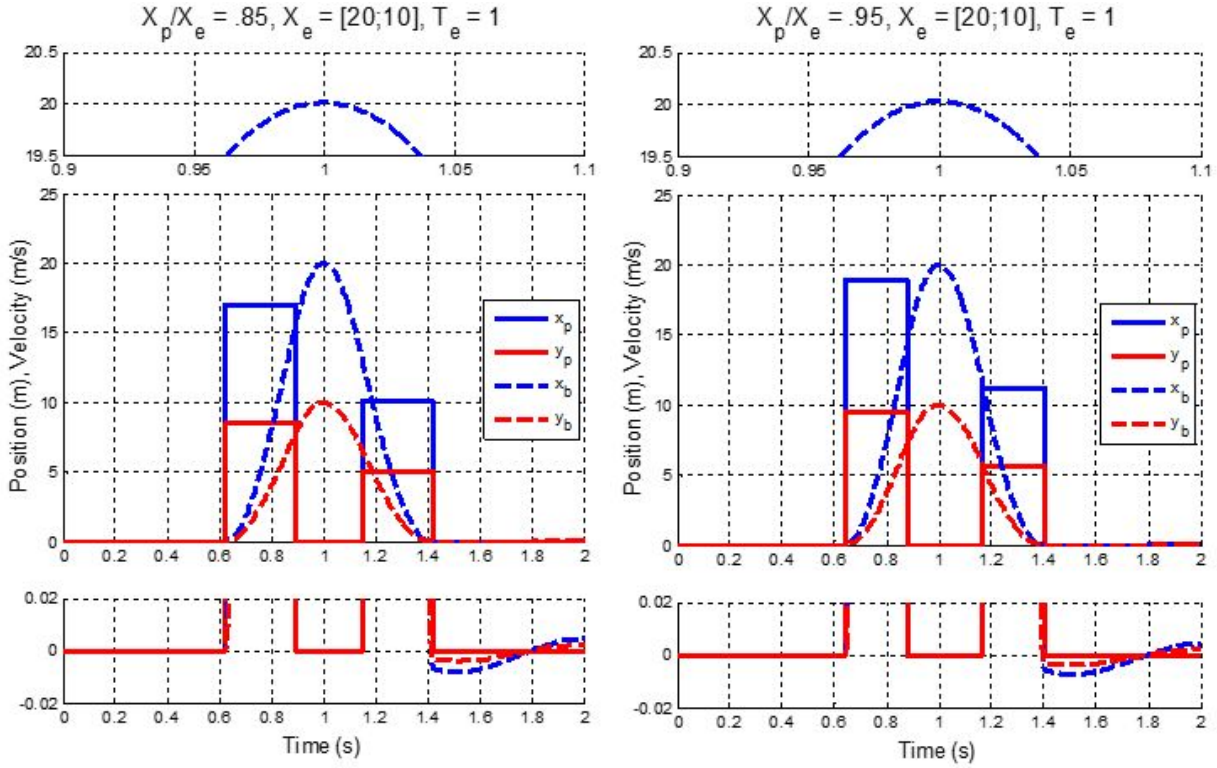
Similar to the input shaper, any final states could be achieved by the ball at any final time, so the impulses were designed to drive the ball to maximum displacements of [20;10] and [10;5] in a

time of 1 second. These displacements would correspond to the positions of the enemies at their interception. The first pulse magnitude (or papi displacement) was set to a fraction or multiple of the maximum ball displacement by applying a single input pulse equal to the quotient of the desired papi displacement and simulation time step. To end the frequency shaper impulse, a single equal and opposite input pulse was applied after the required impulse period. The impulse period was determined by looping through different impulse times until the desired ball displacements were achieved. Finally, the time of the first impulse was calculated as the difference of 1 second and the ball peak time.

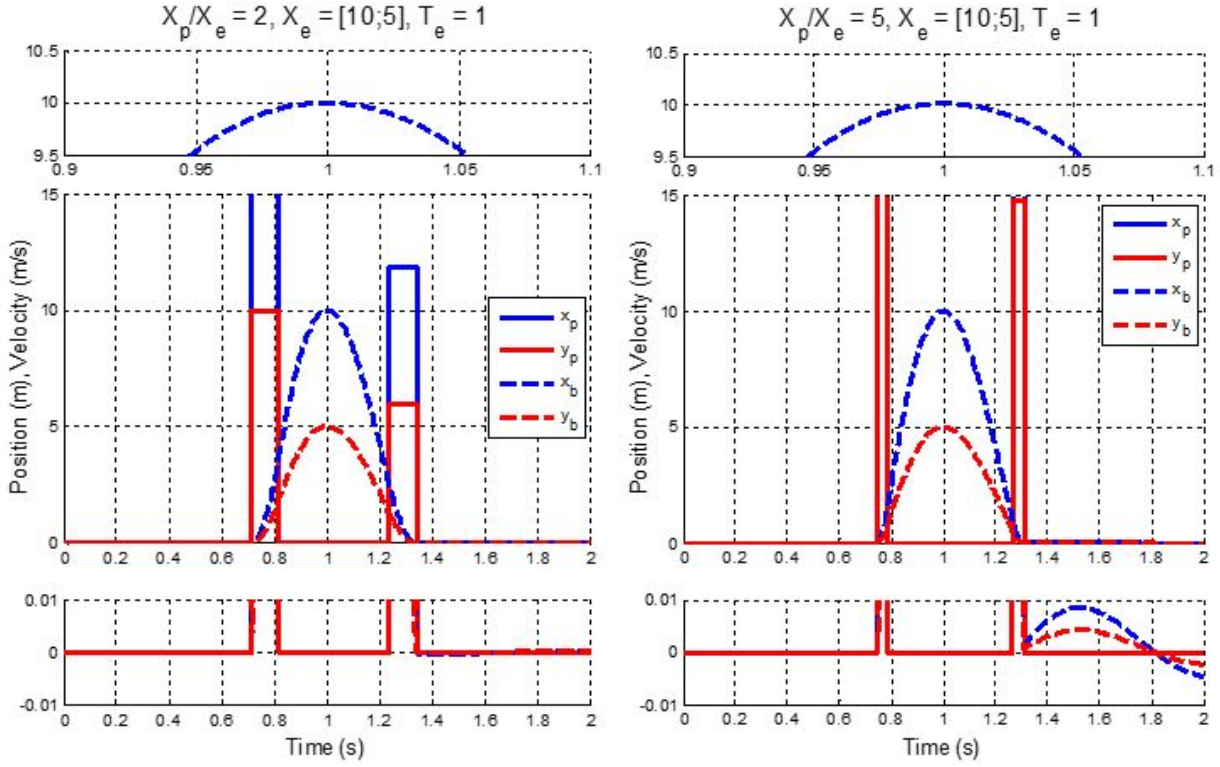
Simulations were performed for papi displacements of .65, .75, .85, and .95 times the maximum ball displacement of [20;10] and 2 and 5 times the maximum ball displacement of [10;5]. Since, the x and y modes both have the same poles, a single shaper with scaled impulses was used. Each figure below shows a zoomed in view of the residual oscillations caused by rounding index errors and the x position of the ball at 1 second in addition to the full response.

Simulations:





Figures 14-17: Input shaping responses for step inputs of papi displacements of .65, .75, .85, and .95 times the enemy displacement required to hit the enemy at 1 second and stop the ball after half a cycle



Figures 18-19: Input shaping responses for step inputs of papi displacements of 2 and 5 times the enemy displacement required to hit the enemy at 1 second and stop the ball after half a cycle

Discussion of Results:

Impulses of lower amplitude required larger periods to impart the same change in momentum on the ball and drive it to the desired max position. Therefor these impulses also started earlier and ended later to drive the ball to the max position at the set time. The amplitude of the remaining oscillations after the second impulses were all below .01 m. Any errors were likely caused by rounding in the indexes and selection of impulse periods leading to different absolute impulses.

From these results, it can be concluded that there are multiple impulses that can be chosen from. Two constraints are the maximum impulse period and the maximum absolute input or input rate of change. A constraint on the input would require the more complicated solution of a combination of ramp, step, and negative ramp inputs. The periods of the two impulse functions are limited in that they can not overlap (the first impulse must finish before the second impulse starts).

4. Least Squares Estimation:

System Modeling and Problem Definition

In real life application, all the states of the system are not readily available, and this can be due to various reasons, ranging from the states being impractical or impossible to be observed or budget constraints restricting the amount of sensors.

However, it is advantageous to have the full states of the system at any one time, as it provides a better understanding on the performance and characteristics of the system as well as providing a mean for full state feedback. Full state feedback is desired in controller designs as it allows the designer to place the system poles at any arbitrary locations. This gives the designer a greater control on the performance and characteristics of the resultant system. Full states feedback are also widely used in other modern controller designs such as Linear Quadratic controllers. By obtaining the full states of the system at every time steps, it gives the designer more controller choices and better control on the performance of the system.

Least square estimation can be used to obtain the full state of the system from the measured states. Least square estimation employs a technique of minimizing the squared discrepancies between measured data to estimates the full state of the system.

In this part's model, the X and Y location of Papi will be measured, and by using these two variables, the entire state of the system will be estimated. The A, B, C and D matrix of the system are as shown below.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & -2 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Design and Analysis

Prior to designing the least square estimator, the observability of the system need to be checked. If the system is not observable, modifications, such as increasing the number of sensor and changing the measured state, need to be carried out on the system before a least square estimator can be used. The Papi system is observable, hence no modification is required.

A modified observability Grammian with a forgetting factor of 0.4 is used. A forgetting factor is used is place more emphasis to recent data (measurements) and discard old information as time passes. The modified observability Grammian is calculated using the following equation:

$$W_o = \int_0^t \Phi_{(\tau,t)}^T C_{(\tau)}^T C_{(\tau)} \Phi_{(\tau,t)} \, exp^{-0.4(t-\tau)} \, d\tau$$

The P matrix of the system is simply the inverse of the modified observability Grammian evaluated at each time step. The estimated state of the system at each time step can then be calculated using the following equation:

$$\hat{x} = \mathrm{P}_{(t)} \int_{t_0}^{t_1} \Phi_{(\tau,0)}^{T} C_{(\tau)}^{T} y_{(\tau)} \, dt$$

The system is also modelled to be subjected to both random process noise and random measurement noise to mimic real life scenarios. A random process noise with magnitude of 5 and a random measurement noise with a magnitude of 10 are introduced to the system. The system is then simulated using Simulink by introducing a pulse input for both inputs. The Simulink diagram for the system is as shown in Figure 20 below.

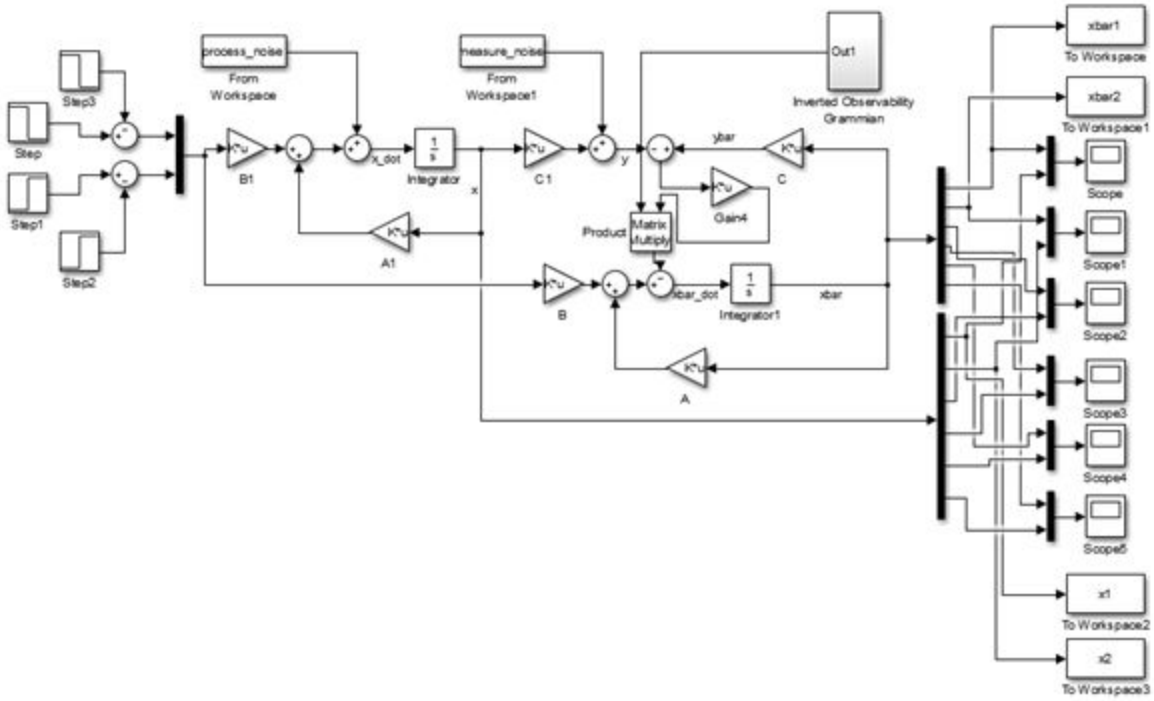


Figure 20: Simulink Model for Least Square Estimation

Results and Discussions

As seen from figure 21 to figure 26 below, the least square estimator is able to correctly estimate the states of the system after it has gotten enough data. During the initial stages ($t < 1s$), there is a large oscillation and discrepancies in the estimated state, however as time passes, the estimator "learns" and is able to estimate the full state of the system more accurately. Even though the

system is subjected to both process and measurement noise, the estimator is still able to give a good estimation of the full states of the robot.

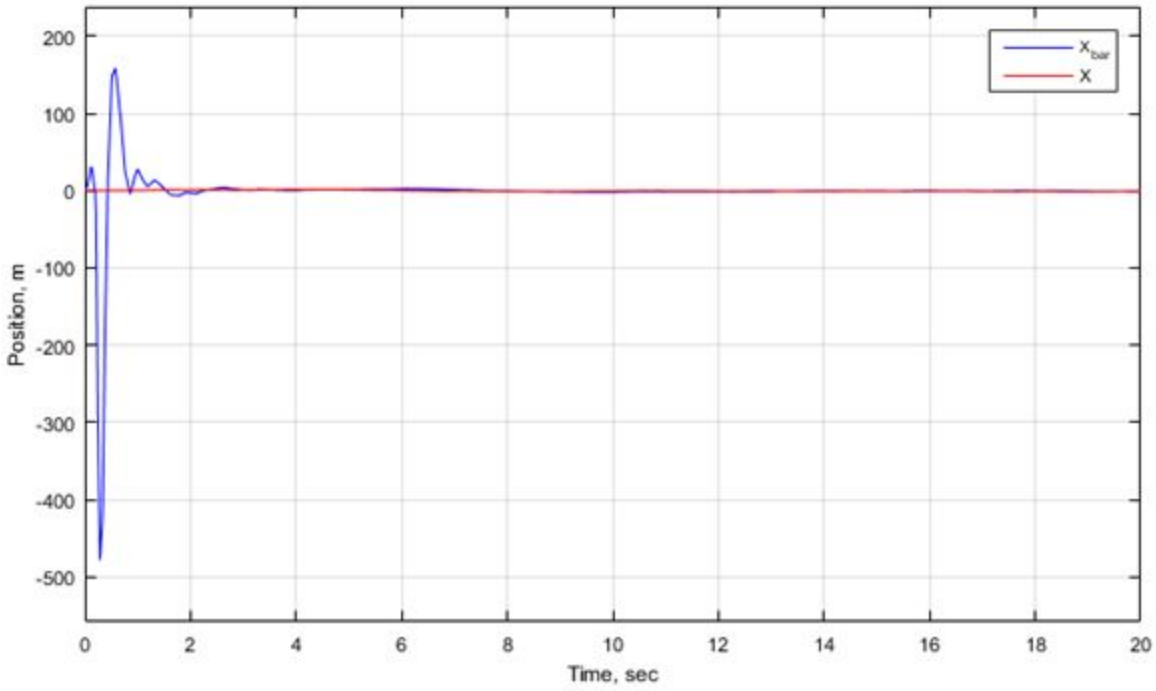


Figure 21: Estimated X position and actual X position

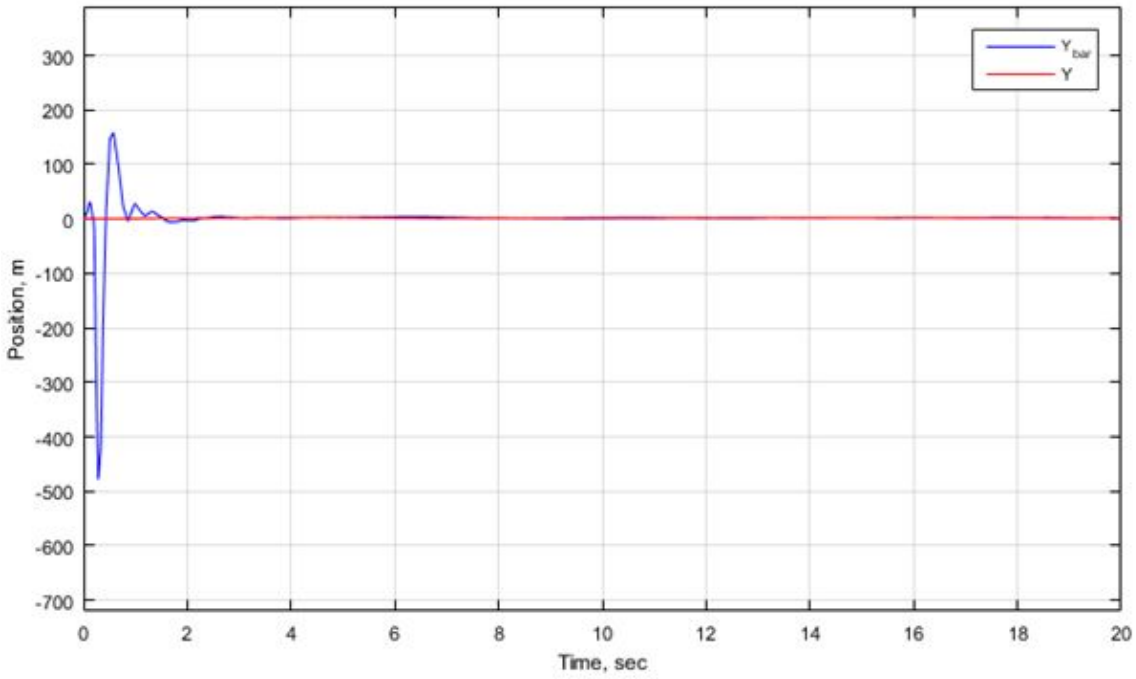


Figure 22: Estimated Y position and actual Y position

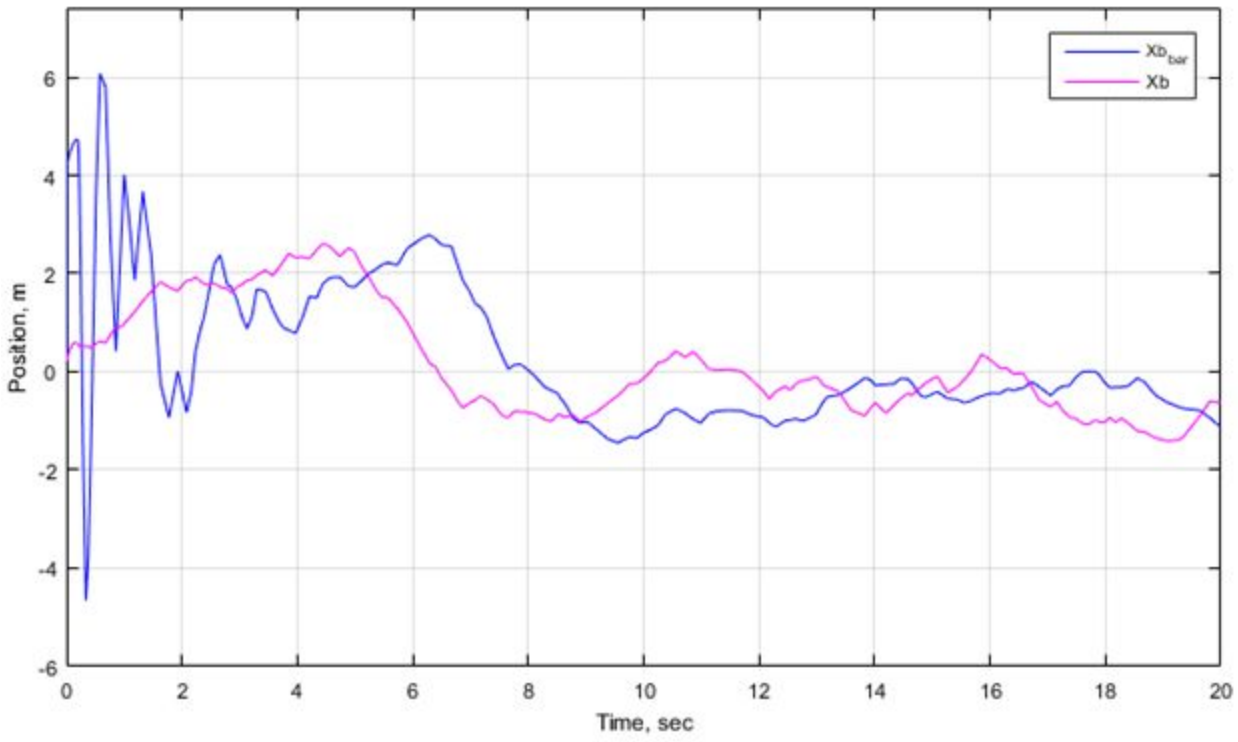


Figure 23: Estimated Xb position and actual Xb position

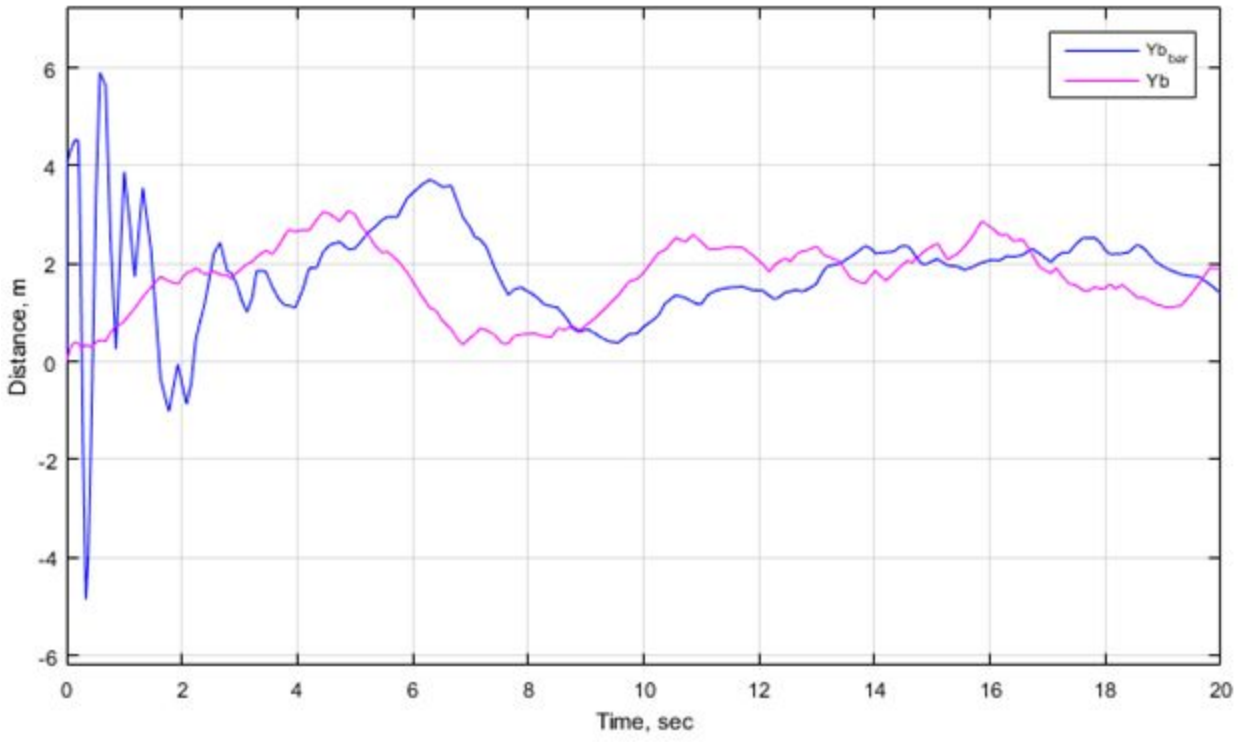


Figure 24: Estimated Yb position and actual Yb position

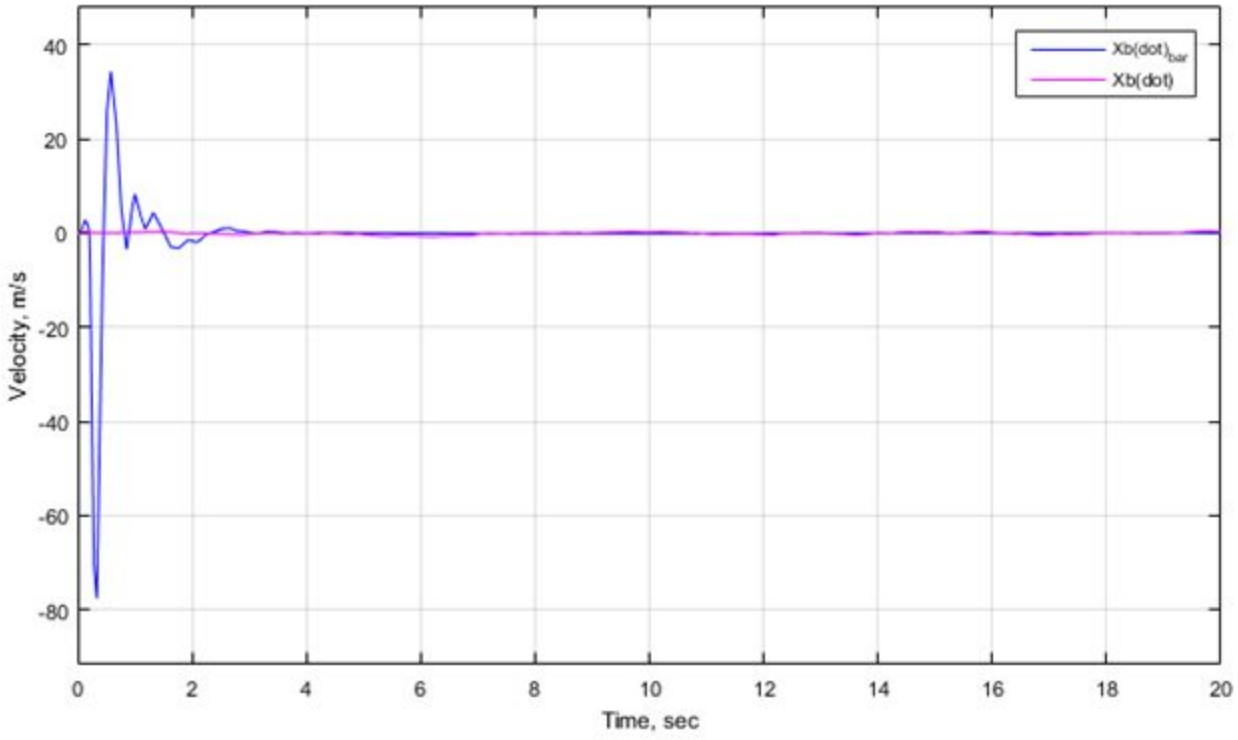


Figure 25: Estimated Xb velocity and actual Xb velocity

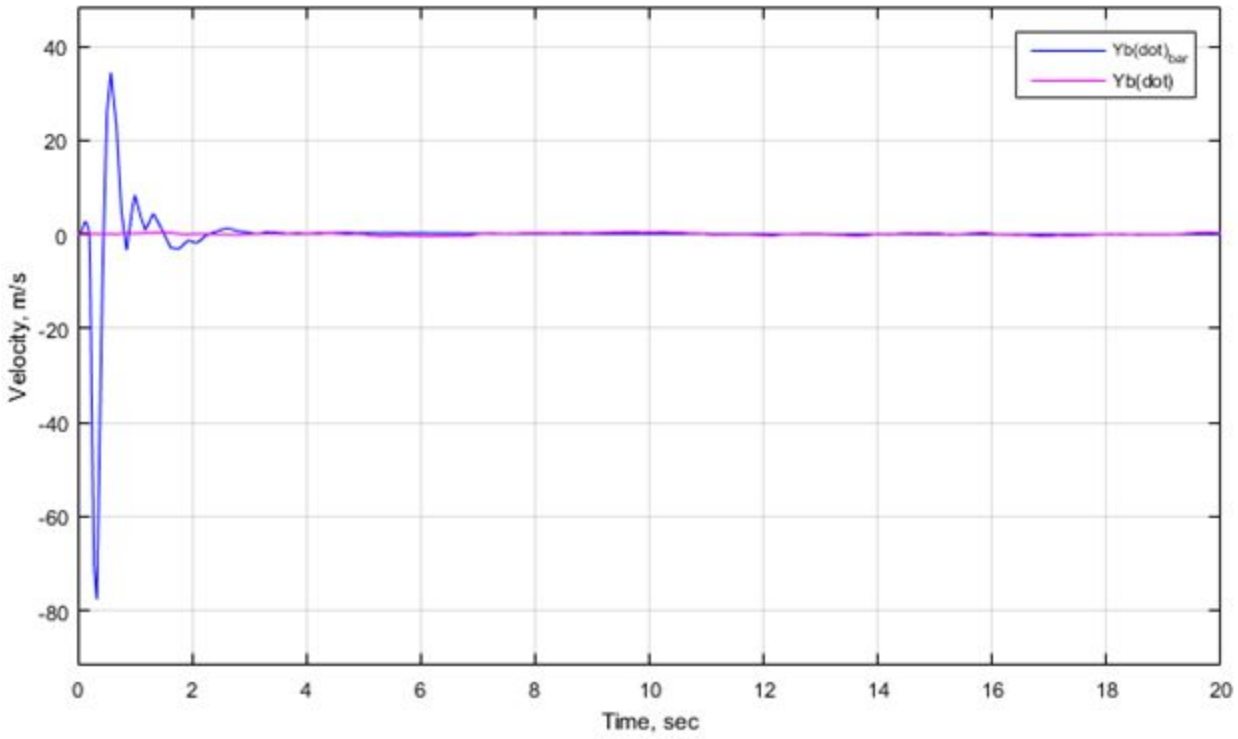


Figure 26: Estimated Yb velocity and actual Yb velocity

5. Pole Placement Approach to Full State Feedback and Observer Design:

System Modeling and Problem Definition

The locations of pole of a system are closely related to the performance and transient response of the system. The pole locations not only affect the system's stability and convergence rate, it also affects the disturbance rejection capabilities and the system's sensitivity to noise. By correctly designing and placing the pole of the system, the designer can precisely control the performance of the system.

One of the method to achieve this is via pole placement. By using the pole placement approach, the designer is able to precisely place the location of the pole at any arbitrary point.
As the full states of the system are not readily available, a Luenberger observer is designed using the pole placement method to estimate the full state of the system. The Luenberger observer uses an innovation term, known as the output prediction error as a feedback to estimate the full state of the system.

Towards the end of this section, an integral action is also introduced to help drive the system state to the required state. As long as there is a discrepancy between the required state and the current state, the integral action will acts to compensate for this discrepancy.

The same plant model as used in the previous section will be used. However, two random noises with magnitude of 2mm are added to the input of the system to simulate input sensor noises.

Design and Analysis

The observer poles need to be placed slower than the expected measurement noise such that it acts as a noise filter. At the same time, the slowest observer pole need to be faster than the closed loop pole for the estimated states to converge before the controller acts.

After some trial and error, the controller poles are decided to be placed at around -5, while the observer poles are placed at twice the magnitude, at around -10.

Using the Matlab place command, the controller poles and observer poles are placed respectively around -5 and -10. The system is then simulated with random noises while paying attention to its performance to drive from an initial state to zero state. The Simulink model of the system is as shown in the figure 27 below.

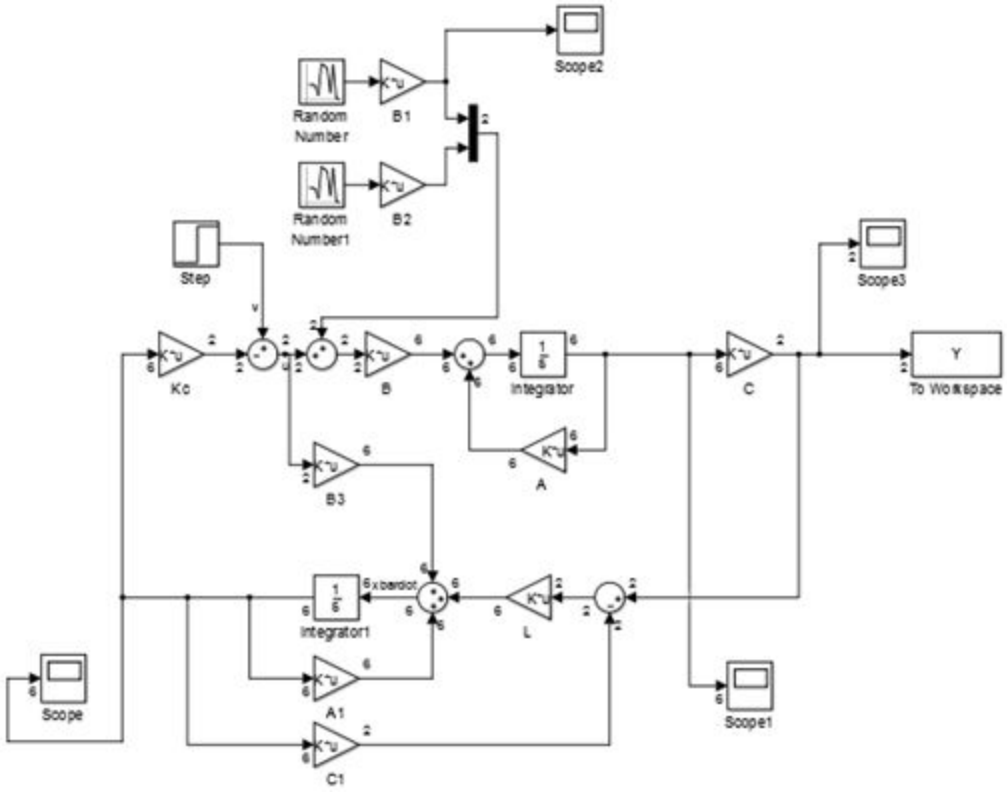


Figure 27: Simulink Model for State Feedback with Luenberger Observer

An integral action is introduced by augmenting the system with additional integral state for both the X and Y position of the spike ball. The augmented state of the system with integral action are as shown in figure 28 below.

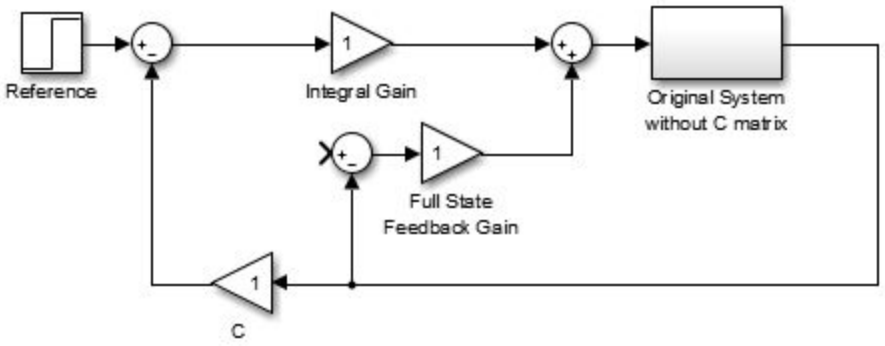


Figure 28: Block Diagram for State Feedback with Integral Action

$$\begin{bmatrix} \dot{x} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \xi \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u$$

Using the Matlab place command, the controller poles and observer poles are placed respectively around -5 and -10. The controller poles for both the full state feedback and integral action are placed around -5. The system is then simulated with random noises while paying attention to its performance to drive from an initial state to the required state. The Simulink model of the system is as shown in the figure 29 below.

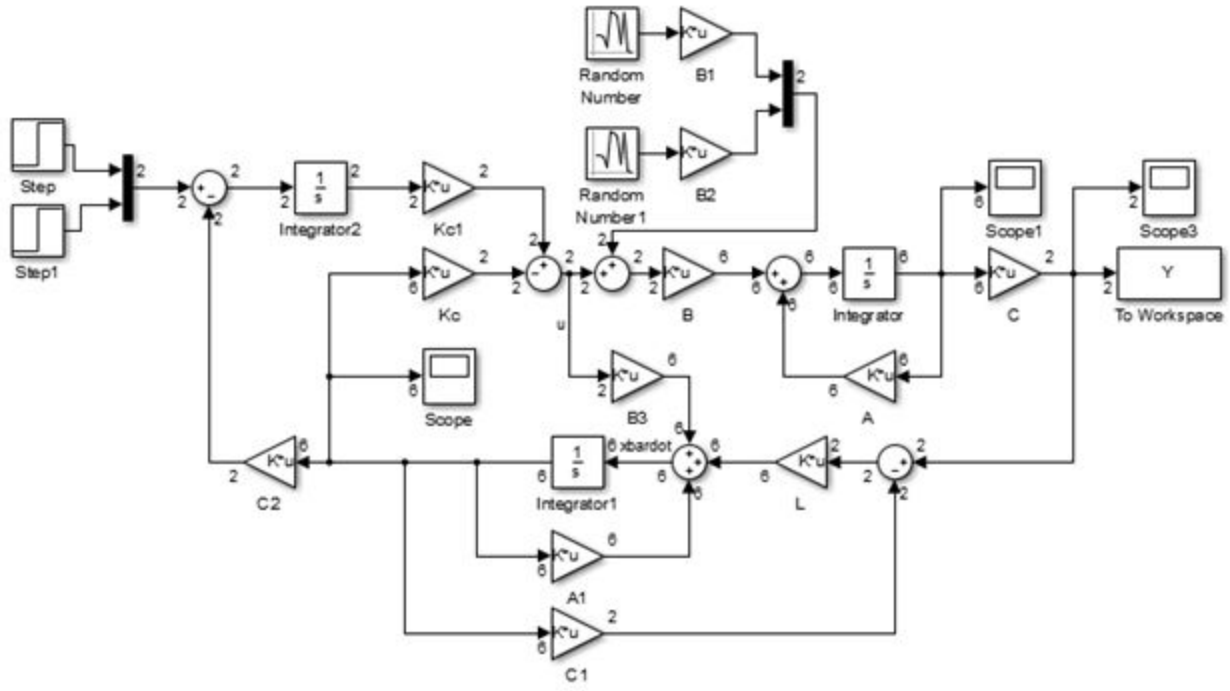


Figure 29: Simulink Model for State Feedback with Integral Action

Results and Discussions

By using the full state feedback and Luenberger Observer, the final state of the system was able to be successfully driven to zero as shown in the following figures. The system is able to reject the random noises and maintain its states even under external disturbances.

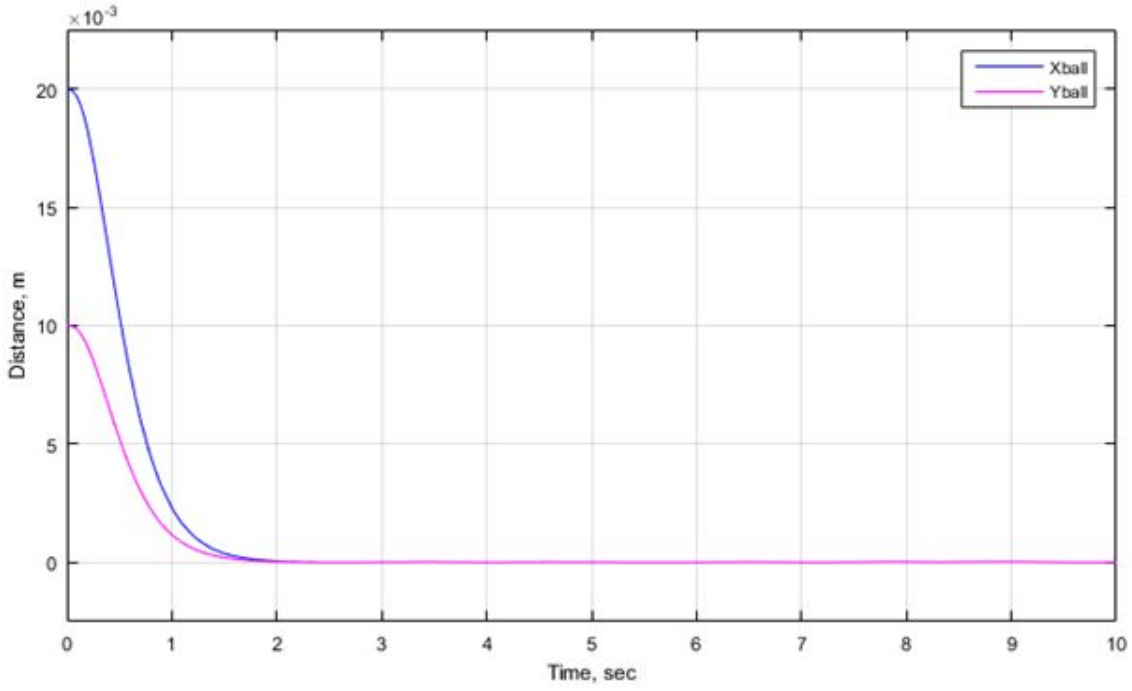


Figure 30: Performance of the system with state feedback and Luenberger Observer

The State Feedback with Integral Action is then used to drive the position of the ball from (0.02,0.01) to (0,0) before driving it again to (0.05,0). The system was able to perform as intended and reached its desired location within less than a second. This is shown in the figure below.

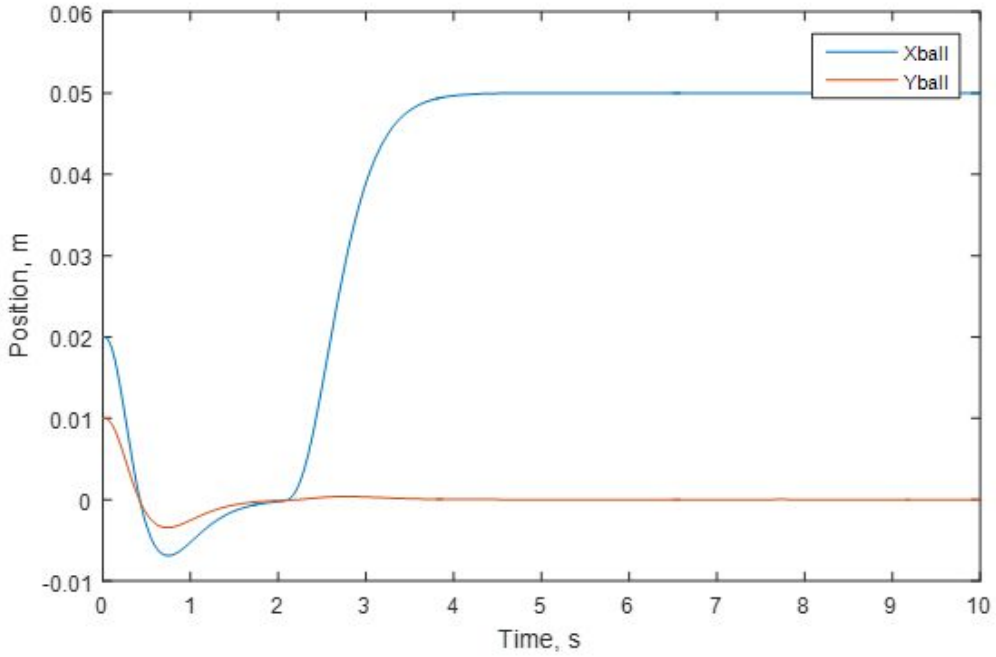


Figure 31: Performance of the system with state feedback with Integral Action

6. Conclusion

In this report, a few method of controller design was analyzed and successfully implemented for the Papi Rubber game. The method covered in this report includes least norm control, where the minimal energy is used to control the Papi ball from one state to another, input shaping control, where any oscillations due to the control inputs are cancelled off within a cycle, least square estimation, where the least square algorithm is used to estimate the states of the system at any step, and finally the pole placement approach to state feedback and observer design, where an observer is first design to estimate the full state of the system before feeding it back as in input to control the system. All these controller are able to function as intended.

7. Future Work

Some proposed future work includes better modelling for the trajectory of the enemies, such as having multiple enemies with different travel profile, generating a real time simulation of the path of the enemies and the Papi, experimenting with 2 loop control algorithm, where the outer loop is used to estimate the optimal trajectory of the Papi to hit multiple enemies as they approach Papi before utilizing the controller in this report as an inner loop.