Golf Swing Robot: A Control System

Maxwell Urbani, Kaylee Cornelius, Ricky Williams

***Abstract*—The goal of this project is to develop a control system capable of swinging a club to hit a golf ball . The robot is modeled as a double pendulum with independent control for each joint. The problem will include controlling a motor at the shoulder joint and at the wrist joint with the goal of hitting the golf ball when it is positioned on the ground in front of the robot.**

Fig. 1. This picture is from an example of a golf swing robot with both wrist and arm (two motors) control. [1]

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

For this system we will focus on the backswing and downswing. The backswing can be described as the movement of the club from a position directly behind the ball upward to the desired position (height varies depending on how far you want the ball to go). The downswing can be described as the movement from the top of the swing back down to where impact with the ball is made. In this system, we will not concern ourselves with the follow through, otherwise known as the part of swing that occurs after impact with the ball.

The problem we seek to address is having the ability to hit a ball on the path that the robot can maneuver the golf club in. We aim to control the movement on both motors located at the shoulder and wrist joints to achieve this goal. We will make sure that the robot will be able to strike the golf ball with the end of the club no matter the starting position.

If we have time to take this system a step further in the future, we will seek to accurately create enough torque to achieve the desired impact speed, therefore controlling how far the ball will travel. We would aim to control the backswing so that the desired hitting speed is achieved on impact with the ball during the downswing. We will consider conservation of energy so that the control will result in the specified goal of kinetic energy to be reached. Then, the achieved swing speed can be used to calculate the approximate distance that the ball will travel. In addition, we may go further and address the way in which kinetic energy will be appropriately removed from the system so the robot is able to return to a state of rest to address the next ball.

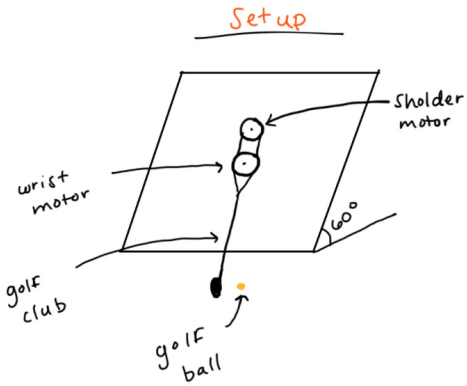


Fig. 2. Swing element: Setup

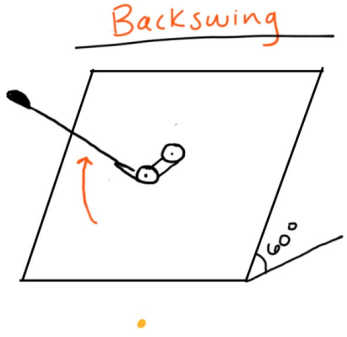


Fig. 3. Swing element: Backswing

Trade Offs: This system greatly simplifies the intricacies of a golf swing. The effectiveness of a golf swing relies not only on one's ability to swing and hit the ball (modeled by our system), but it also depends on other dynamics like weight shifting, hip and shoulder rotation, club selection, and wrist movement. All of these elements are not reflected in our system which greatly reduces the material and control needs. While we lose the exact replication of a human swing by a robot, we can use a double pendulum motion of a robot arm

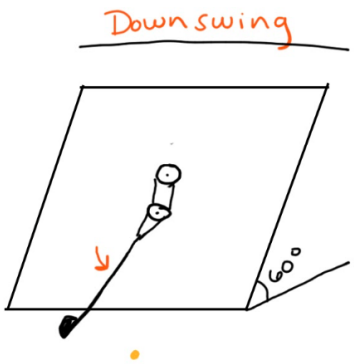


Fig. 4. Swing element: Downswing

to receive results similar to that of a human's golf swing with a significantly less complex system.

II. System

A. System Model

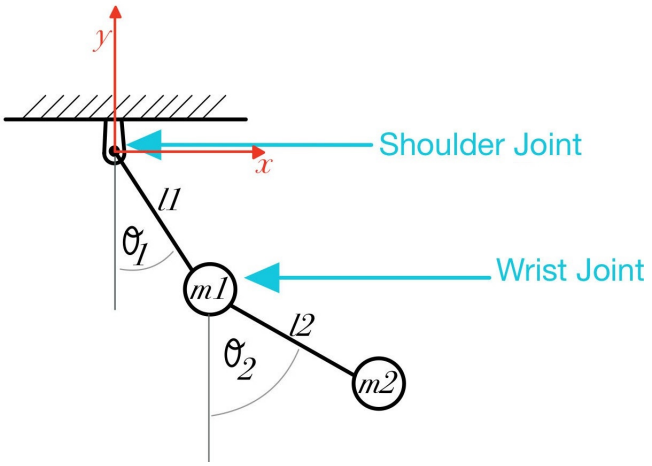


Fig. 5. System Model

We chose a double pendulum model to model our system. This is an appropriate model because we will be using two joints to control how the robot moves so that it can hit any ball placed within its 2-D range of reach. We want to control the motors in order to move the golf club to any position that we want. Therefore, our goal will be to control the torque at each of the joints so that we may ultimately control the position of the club head so that it strikes the ball.

Two assumptions that we are making with this system include that the follow through part of the swing does not affect the system, and that there is no damping or friction effecting the system. We are considering improving the model by modeling the stop of the swing, or the removal of kinetic energy from the system, in future iterations. The implications of no damping will be illuminated in the trajectory simulation.

We will start by identifying the control input, state, and output.

The control input can be described as the torque applied via the motors on each of the two joints.

$$u = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{1}$$

The state of the system consists of the angles of each pendulum from the vertical axes, and their respective angular velocities. Mathematically speaking,

$$x = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta_1} \\ \dot{\theta_2} \end{bmatrix} \tag{2}$$

Because we care about both the position and the velocity of the club, the output of the system will be described as the full state x above.

B. System Dynamics

We can use the Lagrange method to derive the dynamics.

$$L = K - U \tag{3}$$

$$K = \frac{1}{2} m_1 {v_1}^2 + \frac{1}{2} m_2 {v_2}^2 \tag{4}$$

$$= \frac{1}{2}(m_1 + m_2){l_1}^2 {\dot{\theta_1}}^2 + \frac{1}{2} m_2 {l_1}^2 {\dot{\theta_2}}^2 + m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2}\cos(\theta_2 - \theta_1) \tag{5}$$

$$U = m_1 g y_1 + m_2 g y_2 \tag{6}$$

$$= -(m_1 + m_2) g l_1 \cos(\theta_1) - m_2 g l_2 \cos(\theta_2) \tag{7}$$

Plugging in equations (5) and (6) into equation (3) we get

$$L = \frac{1}{2}(m_1 + m_2){l_1}^2 {\dot{\theta_1}}^2 + \frac{1}{2} m_2 {l_1}^2 {\dot{\theta_2}}^2 + m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2}\cos(\theta_2 - \theta_1) + (m_1 + m_2) g l_1 \cos(\theta_1) + m_2 g l_2 \cos(\theta_2) \tag{8}$$

After taking the appropriate Lagrange derivatives, we arrive at the following dynamics equations.

$$\tau_1 = (m_1 + m_2) l_1 \ddot{\theta_1} + m_2 l_2 \ddot{\theta_2}\cos(\theta_1 - \theta_2) + m_2 l_2 {\dot{\theta_2}}^2 \sin(\theta_1 - \theta_2) + g(m_1 + m_2)\sin\theta_1 \tag{9}$$

$$\tau_2 = m_2 l_2 \ddot{\theta_2} + m_2 l_1 \ddot{\theta_1}\cos(\theta_1 - \theta_2) - m_2 l_1 {\dot{\theta_1}}^2 \sin(\theta_1 - \theta_2) + m_2 g \sin\theta_2 \tag{10}$$

If we decouple $\ddot{\theta_1}$ and $\ddot{\theta_2}$ in the dynamics equations, we can derive the following state space representation of the dynamics.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{11}$$

Where

$$\alpha_1 = \frac{-1}{l_1(m_1+m_2) - m_2 l_1 \cos^2(\theta_1-\theta_2)} (m_2 l_1 \cos(\theta_1-\theta_2)\sin(\theta_1-\theta_2)\dot{\theta_1}^2 + m_2 l_2 \sin(\theta_1-\theta_2)\dot{\theta_2}^2 + g(m_1+m_2)\sin\theta_1 - g m_2 \cos(\theta_1-\theta_2)\sin\theta_2) - \tau_1 + \tau_2 \cos(\theta_1-\theta_2) \tag{12}$$

and

$$\alpha_2 = \frac{1}{(m_1+m_2)l_1 - m_2 l_1 \cos^2(\theta_1-\theta_1)} (g(m_1+m_2)\cos(\theta_1-\theta_2)\sin\theta_1 - g(m_1+m_2)\sin\theta_2 + (m_1+m_2)l_1\dot{\theta_1}^2 \sin(\theta_1-\theta_2) + m_2 l_2 \dot{\theta_2}^2 \cos(\theta_1-\theta_2)\sin(\theta_1-\theta_2)) + 2\tau_2 - \tau_1\cos(\theta_1-\theta_2) \tag{13}$$

C. Trajectory Simulation and Model Analysis

Using Matlab's ODE45 function, we are able to simulate the evolution of state of this double pendulum, and plot the dynamics in order to help visualize the system model for analyzing. With a mathematical treatment for deriving the equilibrium states appears in the next section, for the purposes of this section we can deduce that there are 4 equilibrium positions, through direct observation. By the definition of equilibrium point, all equilibrium points have zero velocity. The first equilibrium point is the Down-Down position (both pendulums hanging down). There is also the point of inverted balance, UP-UP, and the mixed positions, Down-Up and Up-Down. To begin to understand the behavior of our model, we can simulate the trajectories around these points, and also at other non-equilibrium initial conditions.
First we simulate the trajectory around the Down-Down equilibrium point

As we can see in figure 6, starting at this equilibrium point, where both pendulums are in the down position, the system does not change.

Next is the Up-Up equilibrium point.

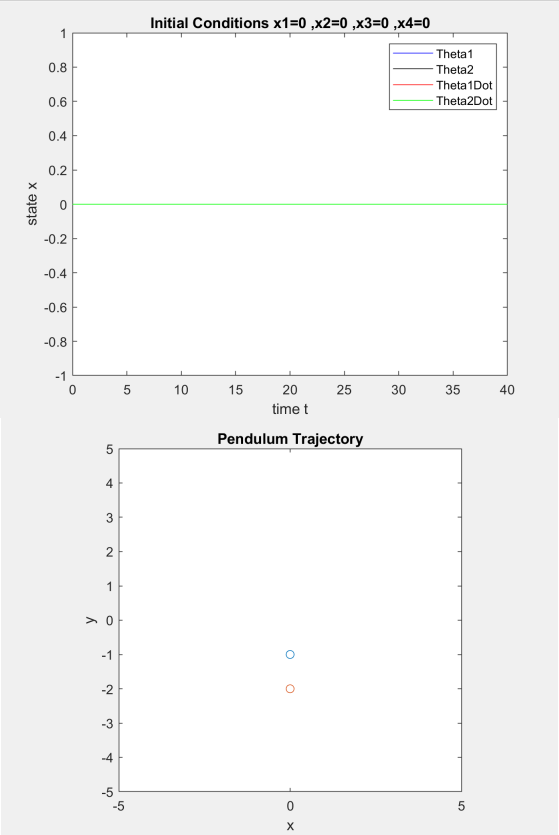
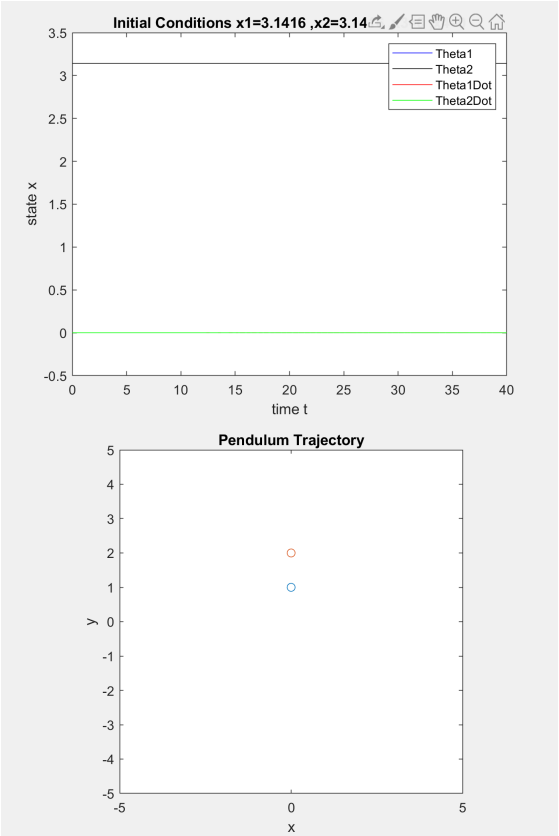


Fig. 6. Down-Down eq point



Up-Up eq point

The behavior of the system beginning at this initial condition is exactly the same as the first initial condition.
Next we will look at the system evolution starting in the Down-Up position with nonzero initial angular velocities. For a double pendulum, we expect this to display chaotic behavior.

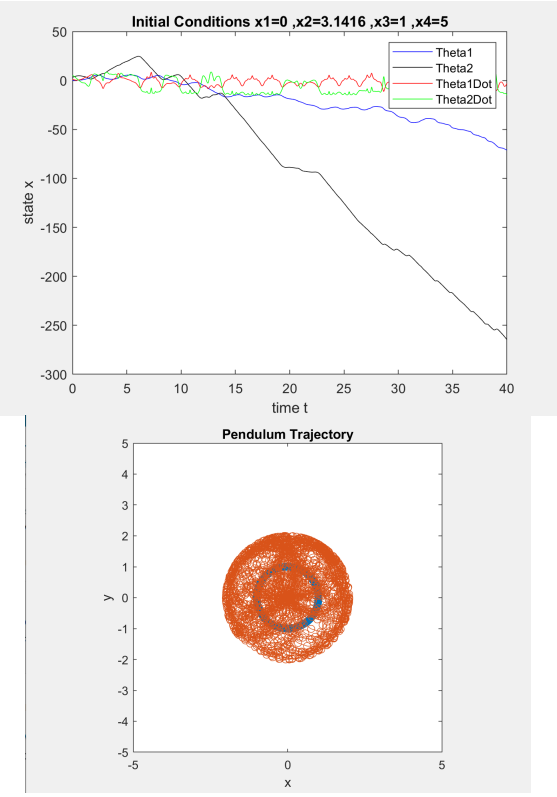
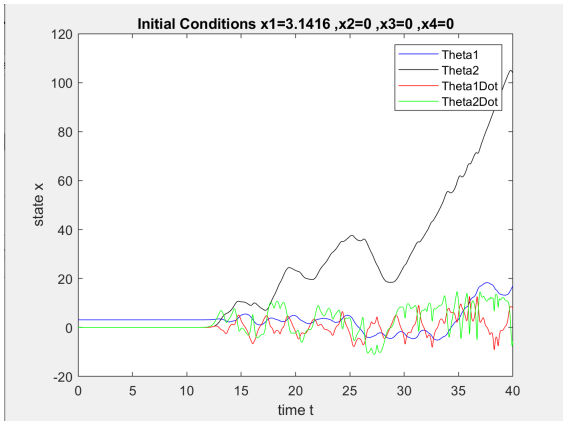
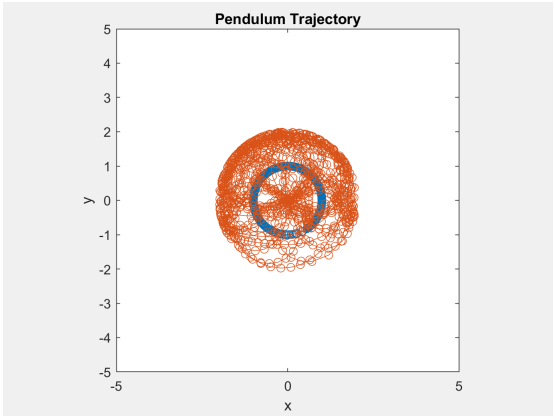


Fig. 7. Mixed Up-Down

Figure 7 shows some curious behavior of our model observed at either of the mixed Down-Up or Up-Down equilibrium positions. With our understanding of an equilibrium position, we expect the state evolution to look like the first two examples, but this is not what we see.





Instead we observe behavior that looks more like the third example, except it takes considerable time for the system to fall out of the equilibrium position. What this looks like is a system that starts very near to the equilibrium point. Since our system does not introduce any disturbances, our hypothesis is that there are some rounding errors with the function pi in matlab, causing this initial condition to be very close to the equilibrium position, but not exactly at it. Curiously, we noted that starting at pi, pi (or UP-UP), we do not observe this behavior. We believe that somewhere in the dynamics, the rounding error must be canceled out in the UP UP condition. The place where this likely occurs is where the angles are subtracted within a sine term. In this place, subtracting Matlab's imperfect estimation of pi by itself would cancel to zero, but subtracting the imperfect pi by zero would not equal a perfect representation of pi.

From these Simulations we understand that there are no asymptotically stable equilibrium points (due to the lack of any damping term) and that there is some error compounding around the mixed equilibrium positions. While this system model leads to misleading trajectory simulation under no control, the model should closely reflect that of the real system under control.

D. Equilibrium and Linearization

Since we are not yet applying control, we set $\tau_1 = \tau_2 = 0$ and from there derived equations for $\ddot{\theta_1}$ and $\ddot{\theta_2}$, our system dynamics. Additionally, at equilibrium, we clearly require $\dot{\theta_1} = 0$ and $\dot{\theta_2} = 0$.

Prior to calculating our linearized matrices, we plugged in values for our model parameters: $m_1 = m_2 = 1$, $l_1 = l_2 = 1$, and $g = 9.8$. For our equilibrium points $(\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}) = (0,0,0,0), (\pi,0,0,0), (0,\pi,0,0), (\pi,\pi,0,0)$ the linearized matrices are:

$$A(0,0,0,0) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.3421 & 0.1710 & 0 & 0 \\ 0.3421 & -0.3241 & 0 & 0 \end{bmatrix} \tag{14}$$

$$A(\pi,0,0,0) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0.3421 & -0.1710 & 0 & 0 \\ 0.3421 & -0.3421 & 0 & 0 \end{bmatrix} \quad (15)$$

$$A(0,\pi,0,0) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.3421 & 0.1710 & 0 & 0 \\ -0.3421 & 0.3241 & 0 & 0 \end{bmatrix} \quad (16)$$

$$A(\pi,\pi,0,0) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0.3421 & -0.1710 & 0 & 0 \\ -0.3421 & 0.3421 & 0 & 0 \end{bmatrix} \quad (17)$$

respectively. Furthermore, we have

$$eig(A(0,0,0,0)) = \begin{bmatrix} 0.7642i \\ -0.7642i \\ 0.3165i \\ -0.3165i \end{bmatrix} \quad (18)$$

meaning the equilibrium point (0, 0, 0, 0) is stable in the sense of Lyapunov, but not asymptotically. Due to lack of energy damping, a perturbation near the Down-Down position will cause the pendulum to oscillate but not settle into the position. Likewise

$$eig(A(\pi,0,0,0)) = \begin{bmatrix} 0.4918 \\ 0.4918i \\ -0.4918i \\ -0.4918 \end{bmatrix} \quad (19)$$

and

$$eig(A(0,\pi,0,0)) = \begin{bmatrix} 0.4918 \\ 0.4918i \\ -0.4918i \\ -0.4918 \end{bmatrix} \quad (20)$$

$$eig(A(\pi,\pi,0,0)) = \begin{bmatrix} 0.7642 \\ 0.3165 \\ -0.3165 \\ -0.7642 \end{bmatrix} \quad (21)$$

Due to a lack of energy damping terms in our system model, none of the system's equilibrium points are asymptotically stable.

E. Viability of the linearized model

Since the ball will be placed somewhere close to the Down-Down position, our control law will be developed around this point. It is important to check that this linearization approximately models the system. First we compare the original nonlinear and the linearized trajectories with initial conditions very close to the Down-Down eq position.

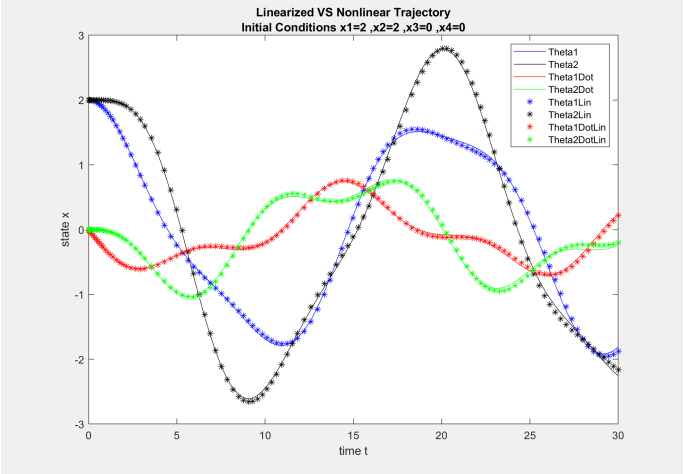


Fig. 8. Linearized trajectroy (stars) plotted on top of noninear trajectory. Initial Condition: x= [1;2;0;0]

From Figure 8 we can see that for small angles (1 and 2 degrees) around the Down-Down position, the model tracks very well. Then, we open up the angles to 10 degrees.

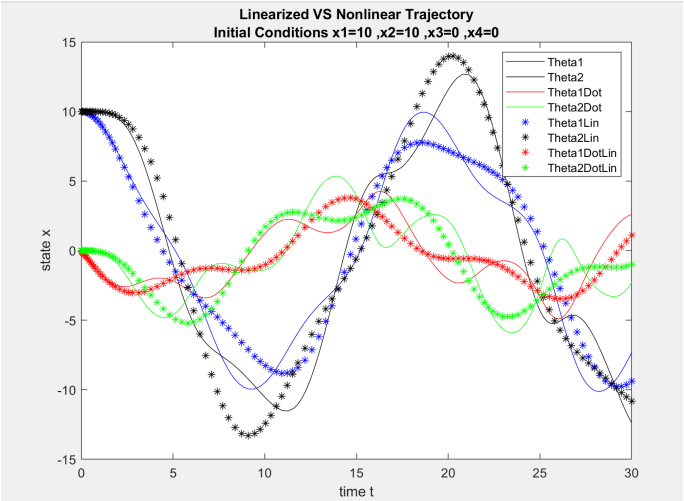


Fig. 9. Linearized trajectroy (stars) plotted on top of noninear trajectory. Initial Condition: x= [10;10;0;0]

Figure 9 shows that for both angles being initialized to positive 10 degrees, the linearized model also tracks the original nonlinear model reasonably well. Further simulation shows that we reach the limits of our approximation when one angle is initialized to positive 10 and the other to negative 10.

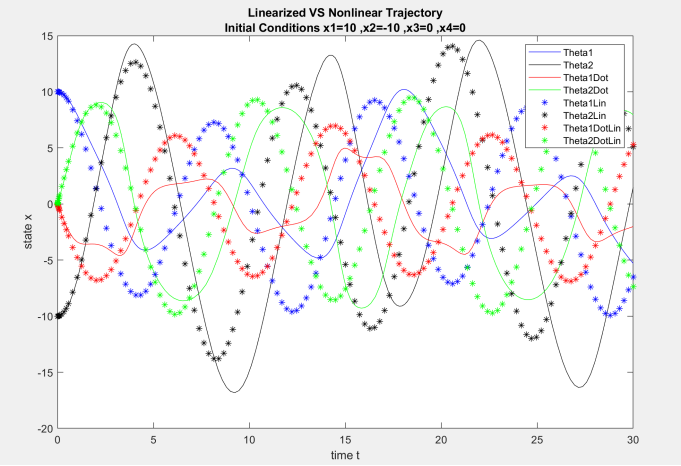


Fig. 10. Linearized trajectroy (stars) plotted on top of noninear trajectory. Initial Condition: x= [10;10;0;0]

In this test, where the angles are intitalized to positive 10 and negative 10 respectively, we can see that the shape of the linearized behavior is approximately correct, but as time goes on it lags further and further behind the actual trajectory. For control laws that stabilize the system quickly, within 5 seconds or so, this linear model is still quite accurate. From this analysis we conclude that our linearization is adequate for controlling around $\pm 10 degrees$.

III. Control

A. Control Objective

Our control objective ultimately is to hit a ball that is placed somewhere near the Down-Down position, with some velocity. To begin working towards this goal, we first develop a control law that can reshape the dynamics to have an asymptotically stable equilibrium at any position in the reachable space near enough to the Down-Down position such that the linearization is still valid ("near enough" is about $\pm 10 degrees$.)

In order to arbitrarily set the poles of our system, we are using full state feedback control.

The B matrix used for full state feedback control was

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{22}$$

The C matrix used was

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \tag{23}$$

B. Controllability

The control matrix is

$$W_r = \tag{24}$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & -0.3421 & 0.1710 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0.3421 & -0.3421 \\ 1 & 0 & 0 & 0 & -0.3421 & 0.1710 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.3421 & -0.3421 & 0 & 0 \end{bmatrix} \tag{25}$$

This matrix is full row rank. Therefore, our system is controllable.

C. Full State Feedback Control

We designed three different controllers and tested them on two sets of initial values to insure the states were driven to the correct values.

To calculate each controller, we began by specifying the desired eigenvalues for the matrix A. Next, we used the place() function in MATLAB to design the state-feedback control, K. Then, we designed the reference gain control, k_r, using the formula below.

$$kr1(1,1) = -.5 * (Ahat(3,1) + Ahat(3,2))/.5365;$$
$$kr1(2,1) = .5 * (Ahat(3,1) + Ahat(3,2))/2.91;$$

where *Ahat* is

$$Ahat = A - (B * K)$$

We then used the state-feedback control and the reference gain controller to generate two controllers, one for θ_1 and another for θ_2 using a control equation of the following form.

$$u = -Kx + k_r r$$

In the above equation, x is a vector in R^4 defining the states and r is the desired output value. In this case, we aim to drive the the system from it's initial position to the following output by setting r equal to 1.

$$x = \begin{bmatrix} 1^o \\ 1^o \\ 0 \\ 0 \end{bmatrix}$$

Controller 1

We calculated the first controller using the following eigenvalues.

$$\lambda = \begin{bmatrix} -2 & -10 & -1+i & -1-i \end{bmatrix}$$

The resulting controllers were

$$u_1 = \begin{bmatrix} -1.6579 & -0.1710 & -2 & -1.3481 * 10^{-15} \end{bmatrix} x + 1.8639 * r$$

$$u_2 = \begin{bmatrix} -1.6579 & -0.1710 & -2 & -1.3481 * 10^{-15} \end{bmatrix} x + -0.3436 r$$

For the first test of our control, we set θ_1 to 15^o and θ_2 to 15^o so that

$$x(0) = \begin{bmatrix} 15^o \\ 15^o \\ 0 \\ 0 \end{bmatrix}$$

This controller succeeded in driving θ_1 to 1^oor θ_2 to 1^o, which was the reference value provided.

To view the transient dynamics see III-C. Analyzing the transient dynamics, we can see that there is some wild behavior with the angular velocities in the beginning, and the system is brought back to steady state at around 5 seconds.

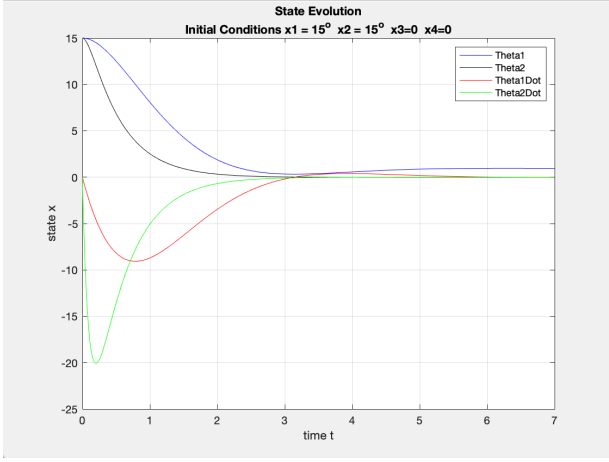


Fig. 11.

For the second test of our control, we set θ_2 equal to 30^o so that

$$x(0) = \begin{bmatrix} 15^o \\ 30^o \\ 0 \\ 0 \end{bmatrix}$$

This controller succeeded in driving θ_1 to 1^o or θ_2 to 1^o again.

To view the transient dynamics see Figure 12. Analyzing the transient dynamics, we can see that there is some extreme initial behavior with the angular velocities, and the system is brought back to steady state a little after 5 seconds.

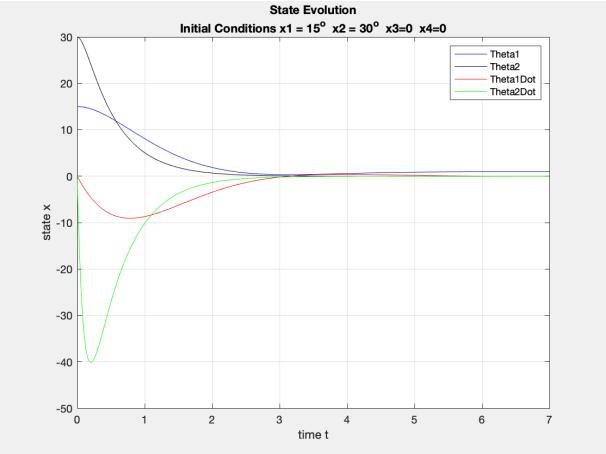


Fig. 12.

Controller 2

We calculated the second controller using the following eigenvalues.

$$\lambda = \begin{bmatrix} -5 & -1 & -2+7i & -2-7i \end{bmatrix}$$

The system was stable, but the k_r values needed to be adjusted to drive θ_1 and θ_2 to 1^o. For this controller, we calculated k_r using

$$kr1(1,1) = (Ahat(3,1) + Ahat(3,2))/(-2)$$

$$kr1(2,1) = (Ahat(3,1) + Ahat(3,2))/(2.9)$$

The resulting controllers were

$$u_1 = \begin{bmatrix} -24.4582 & -5.8238 & -6.0830 & 9-6.9118 \end{bmatrix} x + 15.2265 * r$$

$$u_2 = \begin{bmatrix} 27.1984 & -4.0659 & 4.1906 & -3.9170 \end{bmatrix} x + -10.5010 * r$$

For the first test of our control, we set θ_1 and θ_2 to 15^o so that

$$x(0) = \begin{bmatrix} 15^o \\ 15^o \\ 0 \\ 0 \end{bmatrix}$$

This controller succeeded. To view the transient dynamics see 13. Analyzing the transient dynamics, we can see that there is some overshoot and oscillation. The system behavior is more erratic than with the first controller. The system is brought to steady state at around 5 and a half seconds.

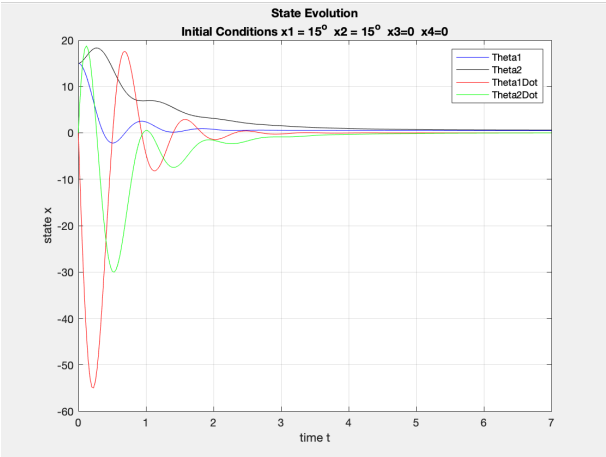


Fig. 13.

For the second test of our control, we set θ_1 equal to 15^o and θ_2 equal to 30^o so that

$$x(0) = \begin{bmatrix} 15^o \\ 30^o \\ 0 \\ 0 \end{bmatrix}$$

This controller succeeded again. To view the transient dynamics see Figure 14. Analyzing the transient dynamics, we can see that there is significant overshoot, and the system is brought to steady state at around 5.5 seconds. This trend of the graph is similar to the first initial condition.

Controller 3

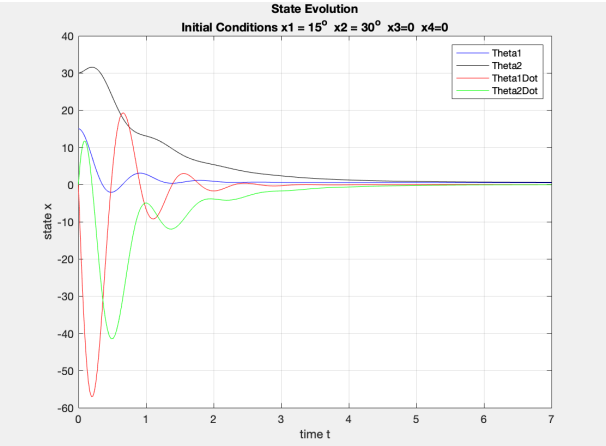


Fig. 14.

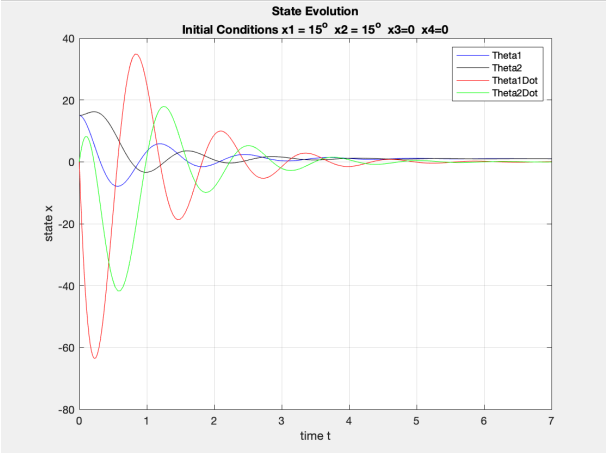


Fig. 15.

We calculated the third controller using the following eigenvalues.

$$\lambda = \begin{bmatrix} -9 & -4 & -1+5i & -1-5i \end{bmatrix}$$

Again, the k_r values needed to be adjusted to drive θ_1 and θ_2 to 1^o. For this controller, we calculated k_r using

$$kr1(1,1) = (Ahat(3,1) + Ahat(3,2)) * -1$$

$$kr1(2,1) = (Ahat(3,1) + Ahat(3,2))/(1.785 * 2)$$

The resulting controllers were

$$u_1 = \begin{bmatrix} -12.9501 & -31.3984 & -8.1296 & -7.7596 \end{bmatrix} x + 44.5195 * r$$

$$u_2 = \begin{bmatrix} 24.2108 & -12.3932 & 0.7886 & -6.8704 \end{bmatrix} x + -12.4704 * r$$

For the first test of our control, we set θ_1 and θ_2 to 15^o so that

$$x(0) = \begin{bmatrix} 15^o \\ 15^o \\ 0 \\ 0 \end{bmatrix}$$

This controller succeeded. To view the transient dynamics see Figure 15. Analyzing the transient dynamics, we can see that increased overshot and oscillation with this controller. The system is brought to steady state around 6 and a half seconds.

For the second test of our control, we set θ_1 equal to 15^o and θ_2 equal to 30^o so that

$$x(0) = \begin{bmatrix} 15^o \\ 30^o \\ 0 \\ 0 \end{bmatrix}$$

This controller succeed with these initial conditions as well.

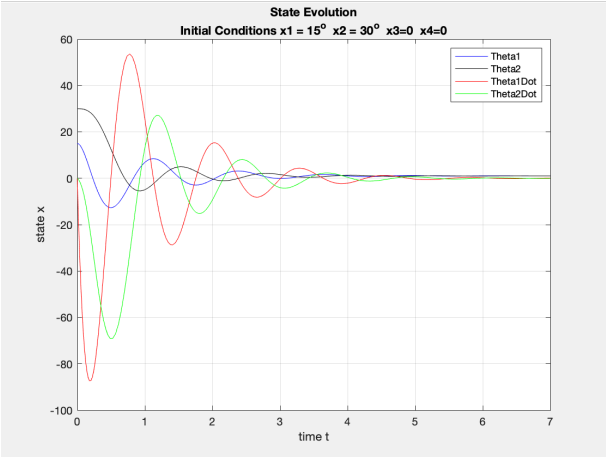


Fig. 16.

To view the transient dynamics see Figure 16. Analyzing the transient dynamics, we can see that the system has similar behavior to the first set of initial conditions, but the angular velocities oscillate at higher values before converging a little after 6 seconds.

D. LQR Control

Full state feedback control is powerful in that we can place the eigenvalues arbitrarily. But how do we choose our eigenvalues to appropriately shape both the transient and asymptotic dynamics of the system? LQR control frames Full-State Feedback as an optimization problem, allowing us to develop a cost function (also referred to as the objective function), whose minimization decides where the eigenvalues are placed. The objective function we used is of the form: $\int_{t_0}^{t_f} (x^T Q x + u^T R u)\, dt$. This objective function makes tuning the controllers intuitive through the Q and R matrices. The diagonal entries of the Q matrix can be thought of as penalizing deviation from good reference tracking, while the R value penalizes actuation effort. In some cases, it may be appropriate to loosen up the constraint on reference tracking if it means we save on actuation effort, such as rocket fuel. In other cases one might want to spend more energy for better

reference tracking. The reference value r tells the system what the final θ_1 and θ_2 positions should be. For our first test we try the following setup:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 + -0 & 0 & 1 & \end{bmatrix}$$

$$R = .001$$

$$r = 2$$

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

This yields:

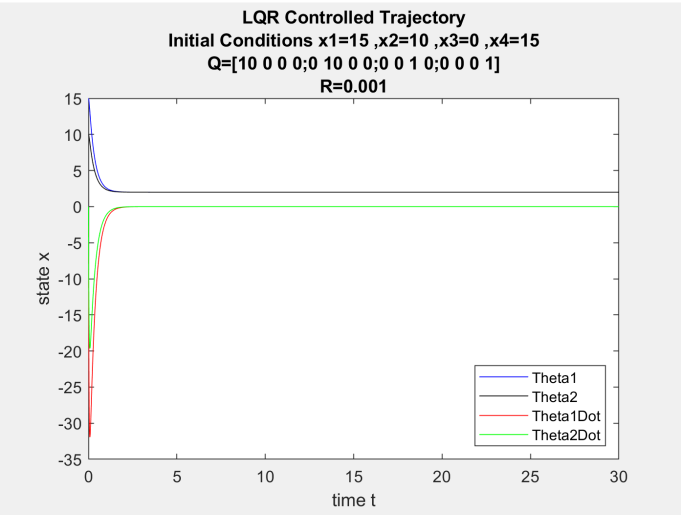


Fig. 17. LQR Control. r =2, R=.001

Figure 17 shows both angles converging very quickly to the reference value with no overshoot. Already we see great improvements in transient response. One thing to note is the spike in $\dot{\theta_1}$ (in red). This is reflective of high actuation effort, which makes sense because the R value was set to a very small value. For a physical implementation of our golf swing Robot, we might want to avoid motor saturation, in which case we can increase the value of R at the expense of the state settling time. Changing R to be 1 we get the following behavior:

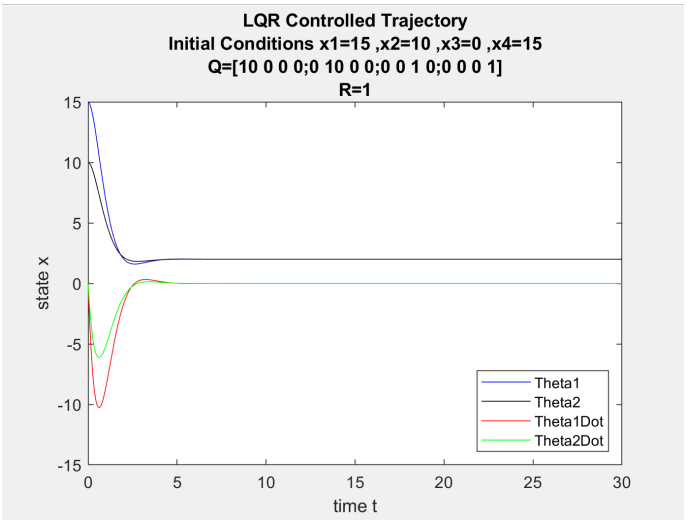


Fig. 18. LQR Control. r =2, R=1

With this new "Cost Function" we get a small amount of overshoot, but now the peak in velocity is roughly a third of what it was before. The settling time is only about 5 seconds, so this is a good compromise.

Now lets say we wanted θ_2 and $\dot{\theta_2}$ to converge the fastest for some reason. This would be reflected by increasing the weights of the corresponding Q entries. We try,

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$$

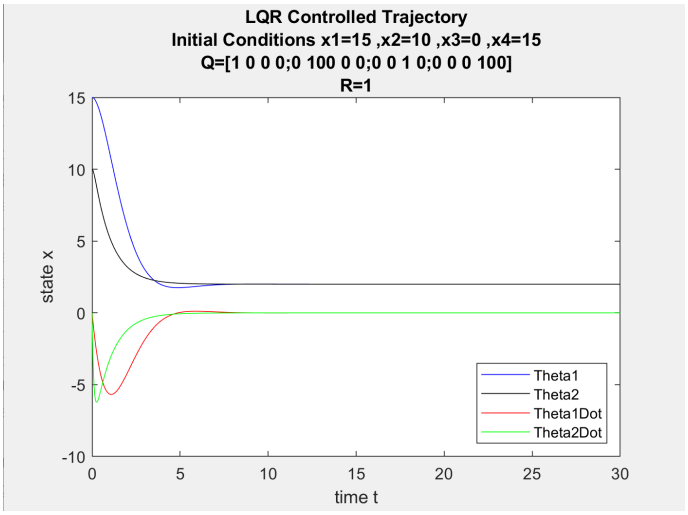


Fig. 19. LQR Control. r =2, R=1

As seen in 19, the corresponding trajectory shows θ_2 and $\dot{\theta_2}$ converging before the others. Finaly we try changing the reference value r to 5 to show the flexibility in setting the state output with this controller.

$$r = 5$$

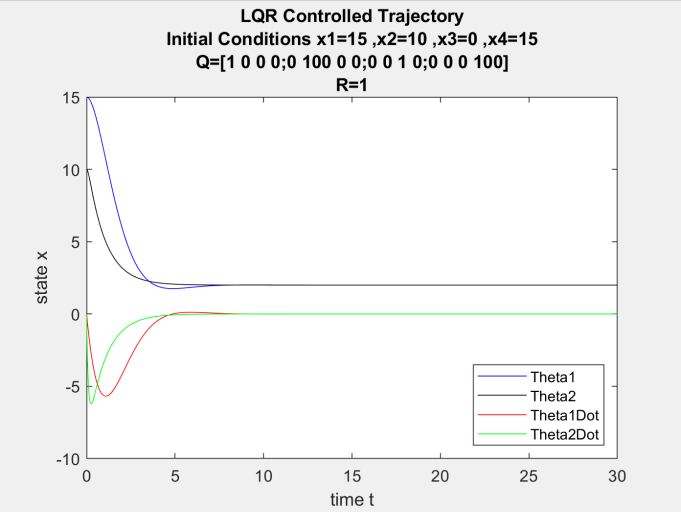


Fig. 20. LQR Control. r = 5, R=1

While we have not yet accomplished our goal of hitting the ball at a position close to the Down-Down eq point with some specified velocity, we have successfully implemented multiple controllers that can drive the equilibrium position to an arbitrary location in the reachable position around the center of our linearization. Through experimentation with LQR control, we arrive at the conclusion that LQR is a very powerful and possibly the most intuitive implementation of Full-State Feedback control that allows for nuanced shaping of the transient dynamics. LQR is also makes for attractive control in it's ability to converge a system to a desired trajectory, as opposed to simply a desired point, which would be an elegant implementation of a two-link golfing robot. We will continue to look into modeling and controls that can allow us to specify a velocity at the ball's position.

IV. Closed-loop LTI with Estimated State

We now design a Luenberger estimator for the system and combine it with our controller for output feedback control. Once again, for our output, we decided our C matrix should be:

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

A. Observability

Our observability matrix is

$$W_o = \begin{bmatrix} 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.3241 & -0.3241 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.3241 & -0.3241 \end{bmatrix}$$

This matrix is full rank, so our system is observable.

Similarly to the controller design, we implemented three separate estimators, all defined by the equation

$$\hat{x} = A\hat{x} + Bu + L(y - C\hat{x})$$

Our closed loop dynamics were defined by

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} + \begin{bmatrix} Bk_r \\ 0 \end{bmatrix} r$$

Our goal was to select the observer gain matrix L such that the eigenvalues of $A - LC$ stabilized the system to our desired output of 0 while minimizing state error. All closed-loop systems did achieve our goal of setting $\theta_1 = \theta_2 = 0$, although in different ways.

B. First Estimator

Our first estimator was defined by the following eigenvalues:

$$\begin{bmatrix} -1 & -2 & -10 & -20 \end{bmatrix}$$

which produced the following observer gain matrix:

$$L_1 = \begin{bmatrix} 979.6761 \\ 23.0000 \\ 428.0104 \\ 162.3158 \end{bmatrix}$$

For our initial conditions, we targeted setting $\theta_1 = \theta_2 = 15$. However, since we cannot know the true initial state, we added constant noise to θ_1 and θ_2 to represent state error. To be more specific, for our initial state, we used

$$\begin{bmatrix} x(0) \\ e(0) \end{bmatrix} = \begin{bmatrix} 17 \\ 10 \\ 0 \\ 0 \\ 2 \\ -5 \\ 0 \\ 0 \end{bmatrix}$$

Our simulation of this closed loop system is given below in Figures 17 and 18. Like simulations without the estimator, we have large overshoot in $\dot{\theta}_1$ and some oscillatory behavior. The system reaches steady state after about 8 seconds. Figure 12 shows the evolution of the state error over time. We have large overshoot in the error of θ_1 and undershoot in the error of both θ_1 and $\dot{\theta}_1$. The estimator only becomes accurate (the error goes to zero) after 15 seconds.

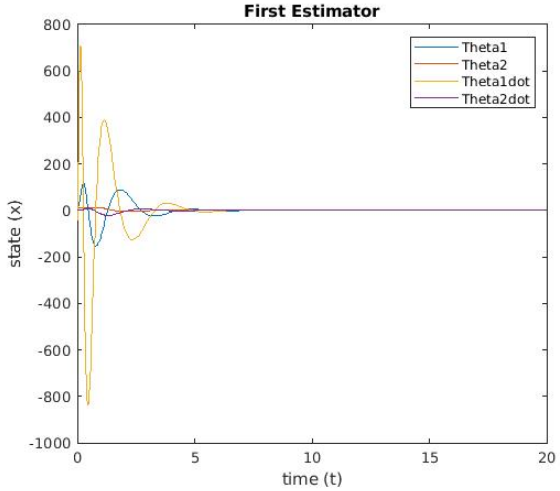


Fig. 21.

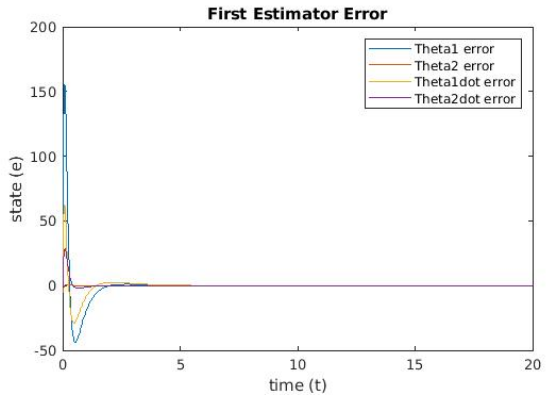


Fig. 22.

C. Second Estimator

For the second observer, we used the following eigenvalues:

$$\begin{bmatrix} -10 & -1 & -1+i & -1-i \end{bmatrix}$$

As a result, our observer gain matrix was given by

$$L_2 = \begin{bmatrix} 109.7767 \\ 13.0000 \\ 24.9782 \\ 33.3158 \end{bmatrix}$$

We used the same initial conditions as above. The simulation of the system is given in Figures 19 and 20. The system acts largely the same compared to the above, but the overshoot for $\dot{\theta}_1$ is significantly smaller. All error terms are also significantly smaller, as seen in Figure 14. The estimator only takes around 6 seconds to become fully accurate.

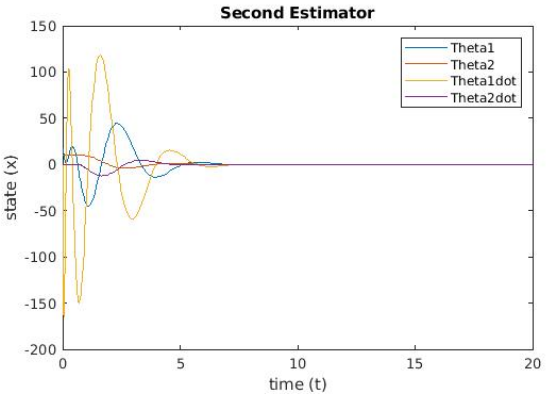


Fig. 23.

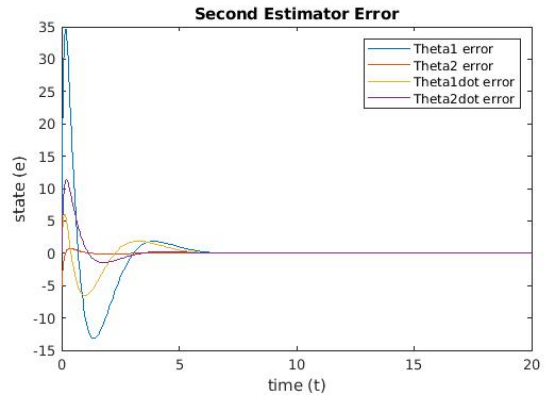


Fig. 24.

D. Third Estimator

For the third and final observer, we used eigenvalues

$$\begin{bmatrix} -1+0.5i & -1-0.5i & -1+2i & -1-2i \end{bmatrix}$$

Our third observer gain matrix was

The simulation results are given in Figures 21 and 22. We saw undershoot and overshoot between that of the first and second estimators for $\dot{\theta}_1$. Error in θ_1 had the lowest overshoot and undershoot between that of the first and second estimators. The third estimator's error takes the longest to settle, taking around 18 seconds to reach equilibrium.

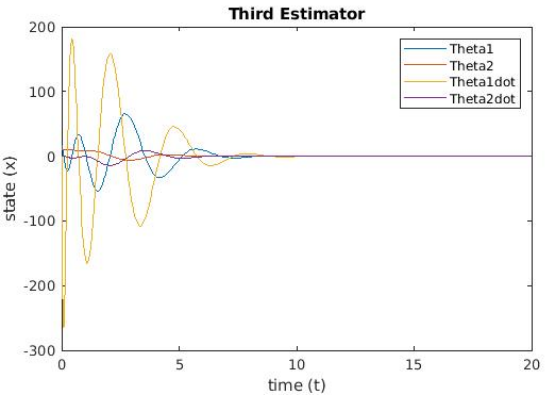


Fig. 25.

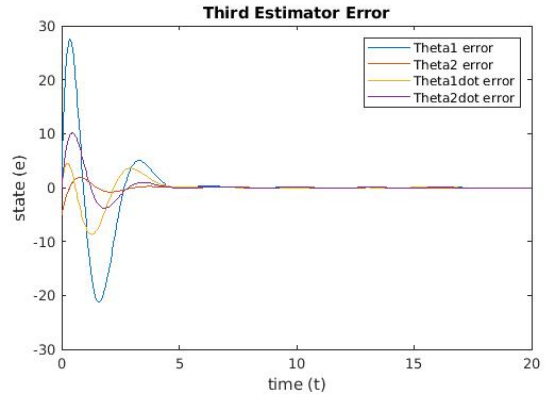


Fig. 26.

Overall, the second estimator is likely the best choice for our system due to its smaller range of values and quick error settling time.

V. Control in the Frequency Domain

For the final portion of this paper, we will study control techniques and analysis in the Frequency Domain. As with the scope of this course, for frequency domain studies we narrow our system model to be single control input and single output. The single control input will be the torque of the motor that forms the shoulder joint. The output will be the position of the end of the golf club.

See (Figure V) for a block diagram of the closed-loop system. In the diagram, r is the reference signal, C(s) is the controller transfer function, d is the disturbance, P(s) is the system plant transfer function, n is the noise, and y is the output.

The disturbance in our system is taken to be the the wind that would affect the robot while hitting golf balls outside.

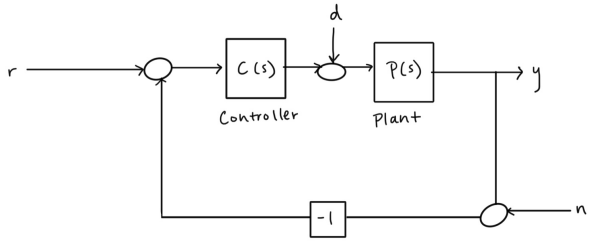


Fig. 27.

A. System Transfer Function

Our system plant is given by

$$P(s) = C(sI - A)^{-1}B = \frac{34210000}{100000000s^4 + 66620000s^2 + 5237551}$$

This transfer function has four poles, all complex: $s = 0.7584i$, $s = -0.7584i$, $s = 0.3018i$, and $s = -0.3018i$. As a result, we see two peaks in the magnitude of its bode plot (Figure 28). The peaks are approximately 160 dB at 0.3018 rad/s and 150 dB at 0.7584 rad/s. The phase plot rises and falls rapidly from -360 degrees to -180 degrees and back at the same points.

B. PID Control

We now try several PID controllers and evaluate their performance.

First, using values $k_p = 0.1$, $k_i = 0.001$, $k_d = 0.1$, from the step response of the closed-loop transfer function, we can see that this controller makes the system immediately unstable (Figure 29). The Bode plot of the open-loop transfer function (Figure 30) gives additional insight: while the phase margin is reasonable (21.8 degrees), the gain margin is extremely small (-74.6 dB).

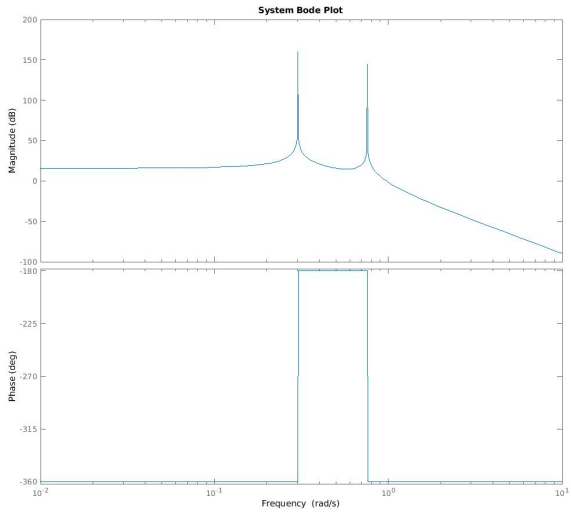


Fig. 28. Bode plot of $P(s)$

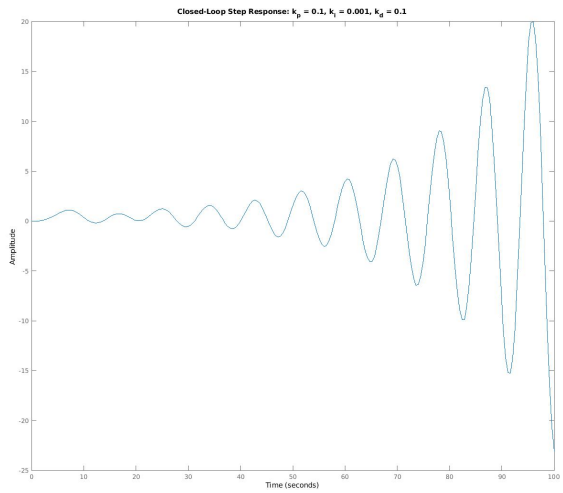


Fig. 29. Closed-loop step response (kp = 0.1, ki = 0.001, kd = 0.1)

In order to boost the gain margin, for our next PID controller, we use a higher k_p value. Specifically, we use values $k_p = 100$, $k_i = 0.001$, and $k_d = 0.1$. However, it is clear from the step response of the resulting closed-loop transfer function that the closed-loop system is still unstable. The bode plot of the open-loop function further shows that the gain margin is still miniscule and the phase margin is now negative.

Finally, we try values $k_p = 10$, $k_i = 1$, and $k_d = 100$. Our closed-loop step response still oscillates and blows up, even within the first 10 seconds. Despite the higher k_d, our phase margin is again negative, and the gain margin is negligible.

After trying various other values, is is clear that our system requires a more complicated setup than a PID controller.

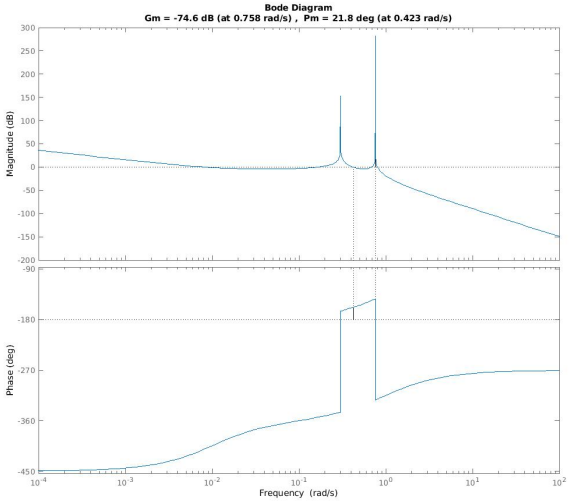


Fig. 30. Open-loop frequency response (kp = 0.1, ki = 0.001, kd = 0.1)

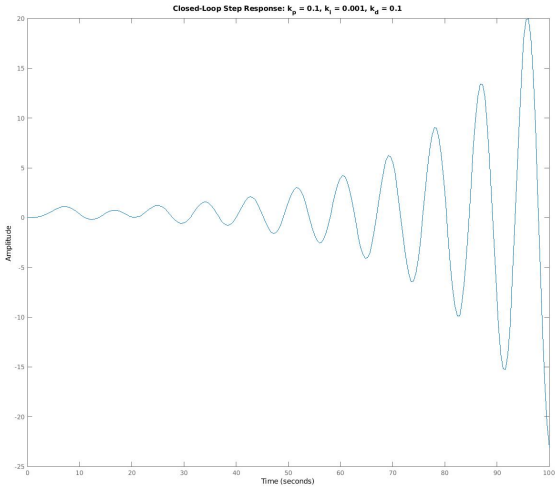


Fig. 31. Closed-loop step response (kp = 10, ki = 0.001, kd = 0.1)

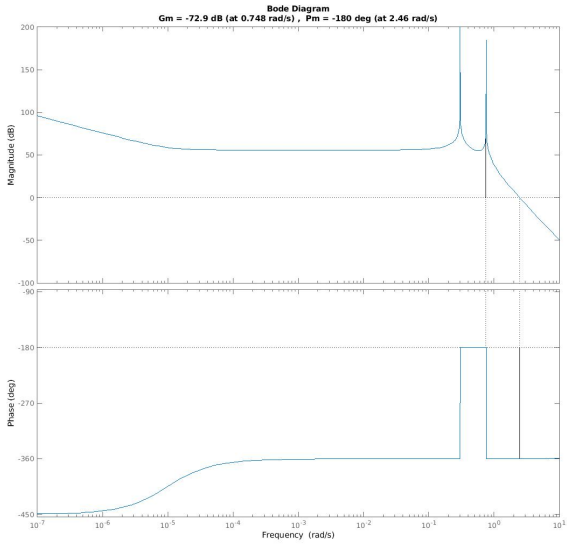


Fig. 32. Open-loop frequency response (kp = 10, ki = 0.001, kd = 0.1)

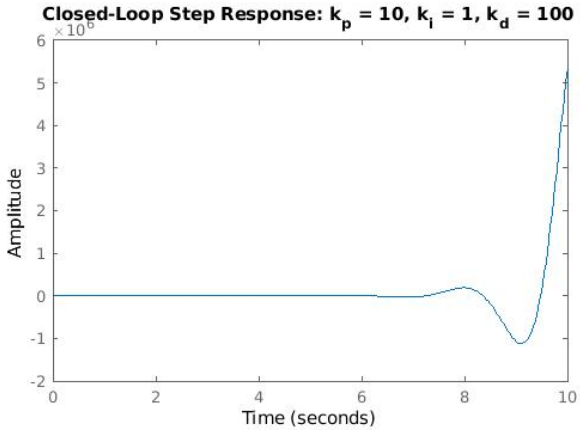


Fig. 33. Closed-loop step response (kp = 10, ki = 1, kd = 100)

C. Feedback Control

We aimed to design a feedback control such that the system would track the given reference input, r, with the following specifications:

- Steady-state error of less than 2%
- Tracking error of less than 10% from 0 to 1 rad/s
- Phase margin of at least 30^o

First, we will need to translate the above specifications into constraints on our system.

To meet the steady state error requirement, we will need to consider how the system preforms as time goes to infinity. To meet this requirement, we want to study the frequency domain error at s = 0. We will use the Final Value Theorem, which relates frequency domain expressions to the behavior in the time domain as time goes to infinity, to find the threshold for the zero frequency gain that will meet the steady-state requirement.

Using the transfer function for the tracking error and noting that it must be less than 2% when s = 0, we can solve the equation for the necessary zero frequency gain.

$$\left|\frac{1}{1+L(0)}\right| < 0.02$$

$$\frac{1}{\left|(1+L(0)\right|} < 0.02$$

Assuming L(s) ¿¿ 1, we can say the following:

$$\frac{1}{\left|(L(0)\right|} < 0.02$$

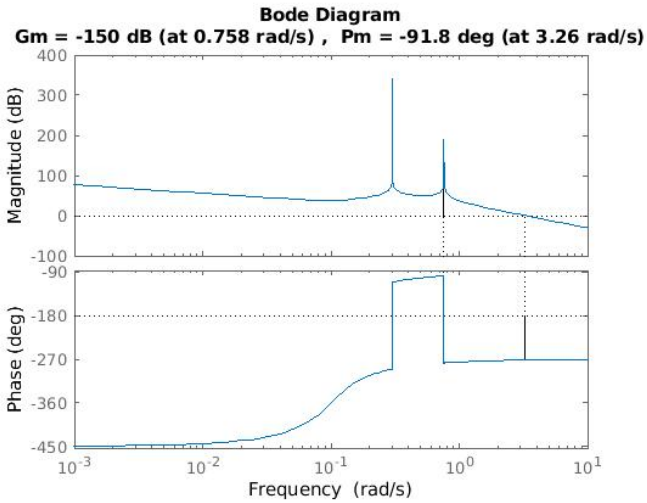


Fig. 34. Open-loop frequency response (kp = 10, ki = 1, kd = 100)

$$50 < L(0)$$

We can now convert 50 into dB.

$$20log(50) \approx 34dB$$

$$34dB < L(0)$$

This tells us that in order for our controller to meet the steady-state specification, the zero frequency gain on the open loop magnitude Bode plot must be greater than 34 dB.

To meet the tracking error requirement, we'll do similar calculations, but between the frequencies of 0 Hz and 1 Hz. We want to find the value that the Bode magnitude plot for the open loop system should not drop below for frequencies up to 1 Hz.

$$\left|\frac{1}{1+L(s)}\right| < 0.1$$

$$\frac{1}{\left|(1+L(s)\right|} < 0.1$$

$$\frac{1}{\left|(L(s)\right|} < 0.1$$

$$10 < L(s)$$

$$20log(10) \approx 20dB$$

$$20dB < L(s)$$

This tells us that in order for our controller to meet the tracking error specification, the gain for frequencies up to 1 Hz on the open loop magnitude Bode plot must be greater than 20 dB.

Lastly, for the phase margin requirement to be met, we will need the phase at the gain crossover frequency, or the point on the Bode magnitude plot that the gain changes from positive to negative, to be between -150^o and -180^o. This will be shown on the Bode phase plot.

These specifications are marked on (Figure **??**).

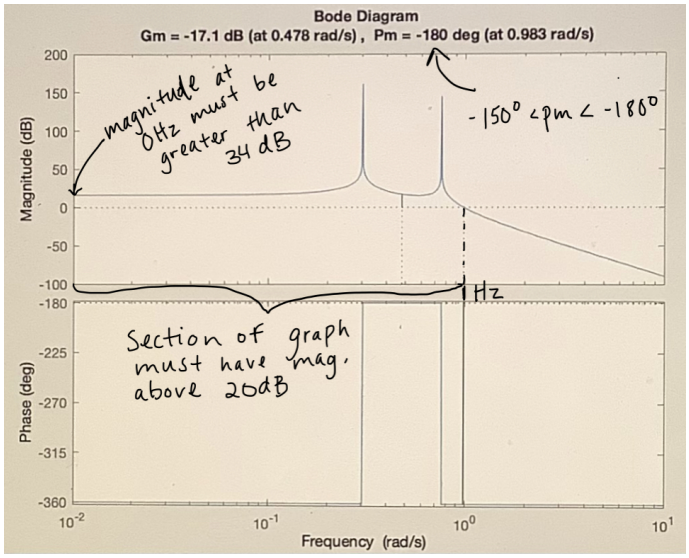


Fig. 35.

A controller for the system that satisfies that the three specifications prior to listed are met is

$$C(s) = 9710 * \frac{(1+2.1s)(1+1.1s)(1+0.99s)(1+0.99s)(1+0.026s)}{(1+0.11s+(0.062s)^2)}$$

By analyzing the open-loop Bode plot where L(s) = P(s)*C(s) in Figure 36) you can verify that all of the specifications are met.

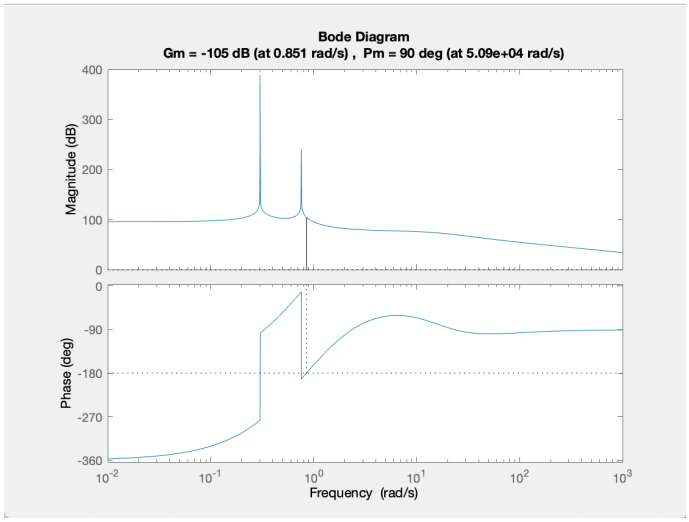


Fig. 36.

As you can see in the step response in Figure 37), this controller stabilizes the system because the steady-state error is zero. The reference signal is equal to 1 in this case.

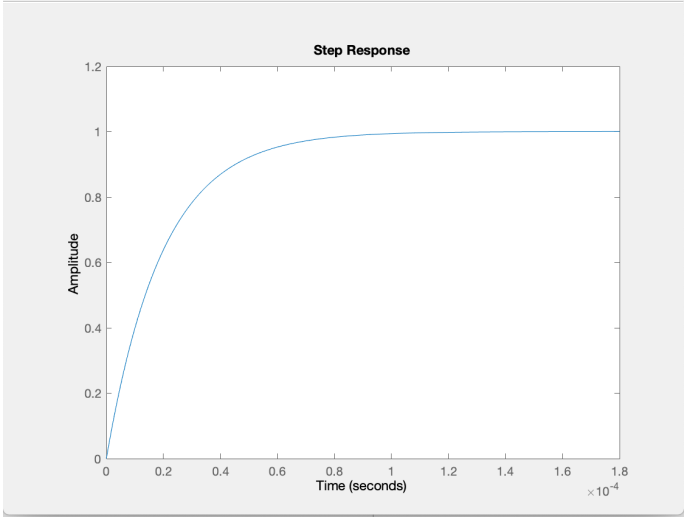


Fig. 37.

For the step response with this controller, you get the resulting values:

$$steady - state\ error : 0 \tag{26}$$
$$risetime : 4.315\ x\ 10^{-5}s \tag{27}$$
$$overshoot : 1.002 \tag{28}$$
$$settlingtime : 7.66\ x\ 10^{-5}s \tag{29}$$
$$\tag{30}$$

Figure 38) displays the Bode plot for the closed-loop system $\frac{L}{1+L(s)}$.

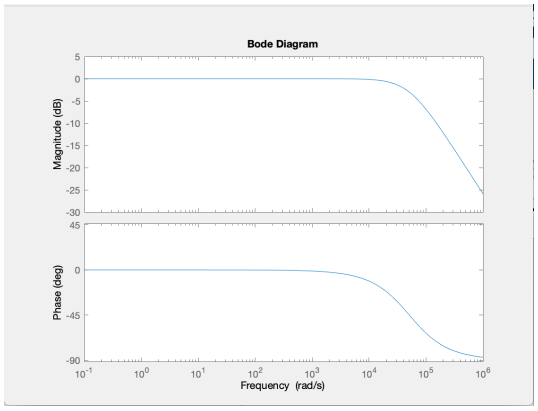


Fig. 38.

D. Frequency Input Simulation

Based on the Bode Plot of the controller C(s), we would expect lower frequencies to track better than high frequencies. To test this we ran two input-output simulations using a sine wave input and the Matlab function lsim. The first was a low frequency input with a period of 3 seconds.

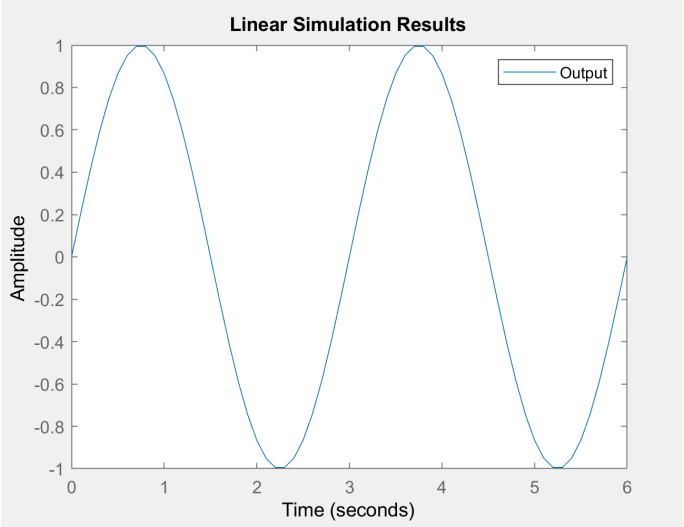


Fig. 39. period = 3 sec

As we can see, the Output in blue is directly on top of the input signal, which is plotted in gray, though by lsim default does not appear in the legend. Next we test a higher frequency input with a period of 0.3 seconds.

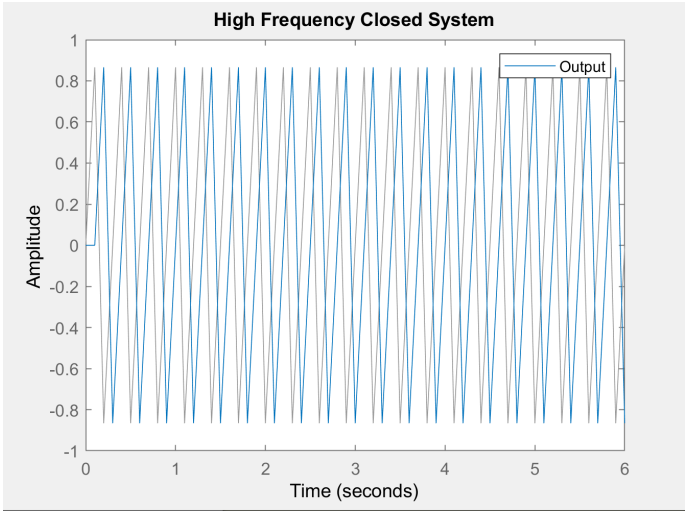


Fig. 40. period = 0.3 sec

From this we can see that there is now a slight phase shift between input and output. The tracking is not as good. If we decrease the period further, the tracking would worsen.

E. Disturbance and Measurement noise Analysis

Thus far we have discussed the tracking performance of our controller C(s). In this section we will see how disturbances and also measurement noise influence the output of the closed loop system. First, let us consider Disturbance. A reasonable disturbance for our system might be the effect of wind on

the golf swing. From the block diagram, one can derive the following transfer function from Disturbance to Output:

$$Gdtoy = \frac{P}{(1+L)}$$

This function has the following Bode Plot.

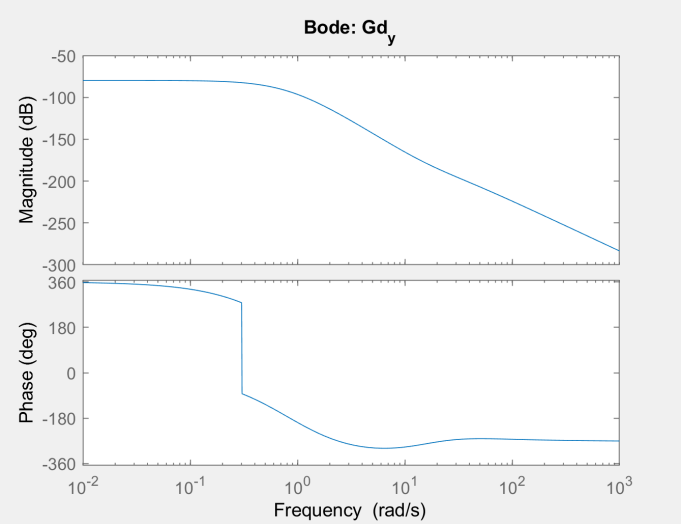


Fig. 41. period = 0.3 sec

From the Bode plot we can see that for low frequencies, disturbances are greatly attenuated, and this attenuation only increases further. This predicts that for any reasonable disturbance input, it should have virtually no influence on the output. We simulate the the disturbance to output dynamics with a sine wave with period 0.3 sec.

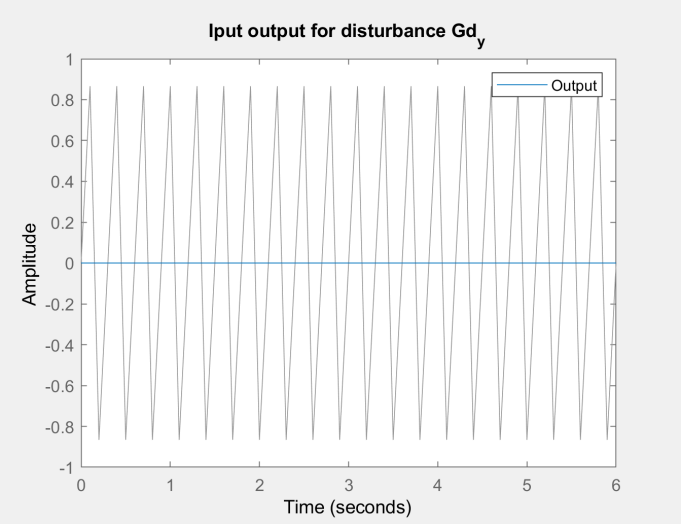


Fig. 42. period = 0.3 sec

From the simulation we can see a nearly complete attenuation of disturbance signals to the output.

Next we study the transfer of measurement noise to the output of the system. From the block diagram we derive the following transfer function.

$$Gntoy = \frac{-L}{(1+L)}$$

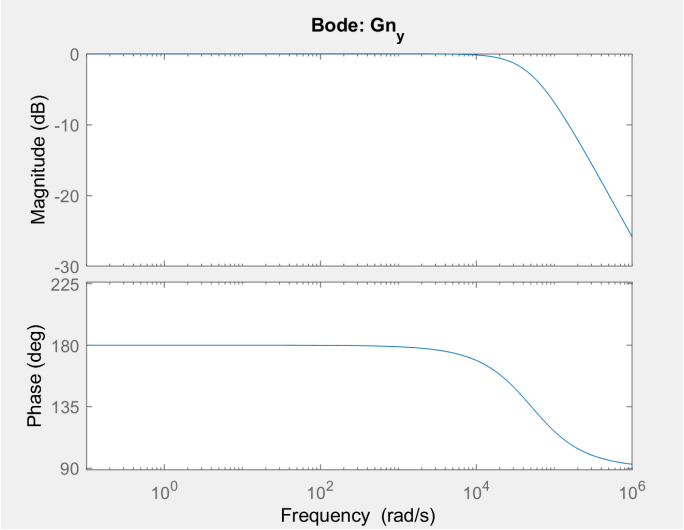


Fig. 43. period = 0.3 sec

This Bode plot suggests an almost unity transfer of measurement noise to output at low frequencies.

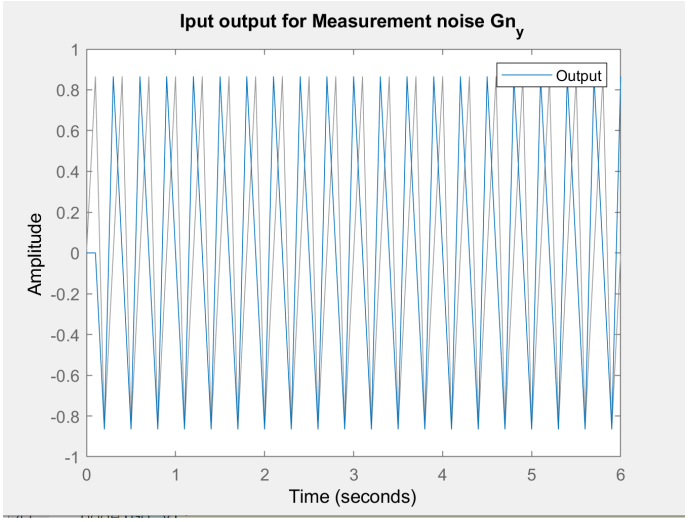


Fig. 44. period = 0.3 sec

From the simulation we can see that this is indeed the case. This seems problematic at first, until we look at the bode plot a little closer. From the bode plot we can observe that at high frequencies, above about 100000 rad/s, the measurement noise is greatly attenuated. This means that as long as our measurement noise is at a sufficiently high frequency, and the input signal is sufficiently low, it will not have an effect on the output of our system.

VI. Conclusion

For this research project we set out to control a golfing robot to be able to hit a ball placed on the ground with a specified speed. To this end we modeled the robot as a two link, planar, RR-manipulator (RR meaning each joint is rotational). Another interpretation of the model is that it is a double pendulum with actuation at each joint. From the model we found the system has four inherent equilibrium points, which are the combination of vertically up and down orientations of each joint. We linearized the model about each equilibrium position, but since our robot is to hit a golf ball on the ground, we focused the study on the down down equilibrium point. Simulating the dynamics with the actuators off of both the linear and nonlinear models, we found that the linearization describes the dynamics reasonably well around ± 10 degrees. We then found that the model was controllable, and started our quest to control the system with the state space model.

The control techniques we explored all worked to set the steady state of the system. With the state being the angular positions of each link, steady state by definition means that the angular velocity will converge to zero. This means that our full state feedback and LQR control methods would be successful at driving the golf club to whatever position we want, but without the capability of hitting it with a certain velocity. We explored the idea that there might be different ways to define the state to achieve both a position and velocity, ultimately we concluded that we will be unable to achieve both with steady state control methods. With this new understanding, we elaborated on steady state controllers, specifically full state feedback and LQR, to stabilize the two link pendulum at any desired position, and also set up a trajectory with which future research on trajectory tracking can use to achieve the goal of hitting a ball at a specific place with a specific velocity. With our full state feedback and LQR methods, we were able to stabilize the pendulum at any position in its reachable space. We found that LQR control is a form of full state feedback that provides an intuitive method of tuning the controller by framing eigenvalue placement as an optimization problem.

We concluded our study by exploring controls in the frequency domain. To remain within the scope of ES155, we narrowed the system from MIMO to SISO. Using PID control, we were able to achieve a steady-state error of less than 2%, tracking error of less than 10% from 0 to 1 rad/s, and phase margin of at least 30^o. One major advantage of frequency domain analysis and transfer functions is that it allows us to study how disturbance and measurement noise transfer to the output. By studying transfer functions, we were able to learn that for this system, the system tracks best at low input reference frequencies, disturbances are rejected well across all frequencies, and measurement noise passes through with almost unity gain but is attenuated at high frequencies. Thus, if our input frequency is relatively low, but the measurement noise frequency is relatively high, the system will track well and reject both disturbances and measurement noise.

VII. Further Research

In order to achieve the goal of hitting the ball at a targeted position with a targeted velocity, we would need to implement some form of trajectory tracking. In trajectory tracking, rather than stabilizing at a point or state, the system stabilizes to a trajectory. In order to implement this, a trajectory first needs to be planned.

A. Trajectory Planning

The trajectory created used Matlab's nonlinear program solver fmincon. Fmincon takes in linear and nonlinear constraints, an objective function, and an initial guess of the trajectory and returns a trajectory that optimizes the objective function while satisfying the constraints. To achieve the goal of hitting a ball at a location with a specified speed, we specify the initial and final positions as linear constraints. Because fmincon only returns the evolution of state, but we are interested in the control as well, we put the motor control u in the state vector so that the output of fmin con provides both the state and control trajectory. To reflect how a human might swing a ball, the desired initial and final positions are set to be:

$$x_{init} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta_1} \\ \dot{\theta_2} \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 180 \\ 270 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$x_{fin} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 20 \\ 0 \\ 0 \end{bmatrix}$$

The trajectory is calculated as an evolution of state with 80 points. The choice of the initial and final positions was to reflect that of a golfer. The initial velocity is zero to simulate the golf club starting from rest, and the final velocities is somewhat arbitrary, but could be chosen to say, hit a golf ball to a specific hole.

The Objective function used is the sum of the actuator effort squared. This objective function is convenient because it has a global minimum, provides relatively smooth results, and minimizes torques, making it useful for physical implementations. The nonlinear constraints passed to fmincon constrain the solution to satisfying the dynamics.

For an initial guess, we used a linear path from X_{init} to X_{fin}.The trajectory of the pendulum and the corresponding actuator effort are as follows:

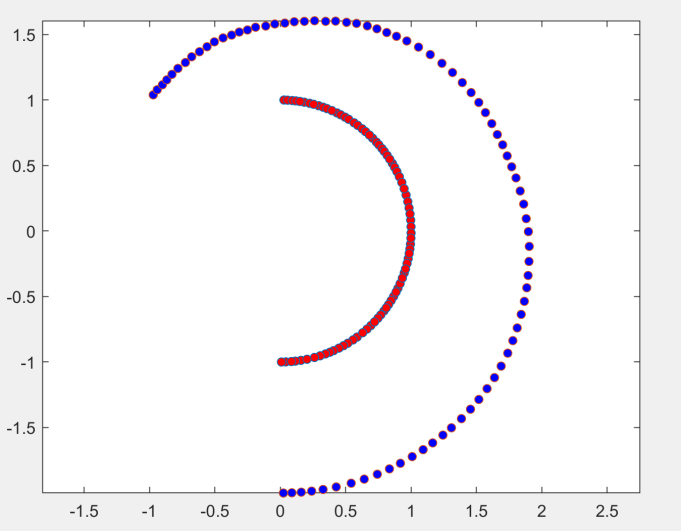


Fig. 45. Double Pendulum Trajectory

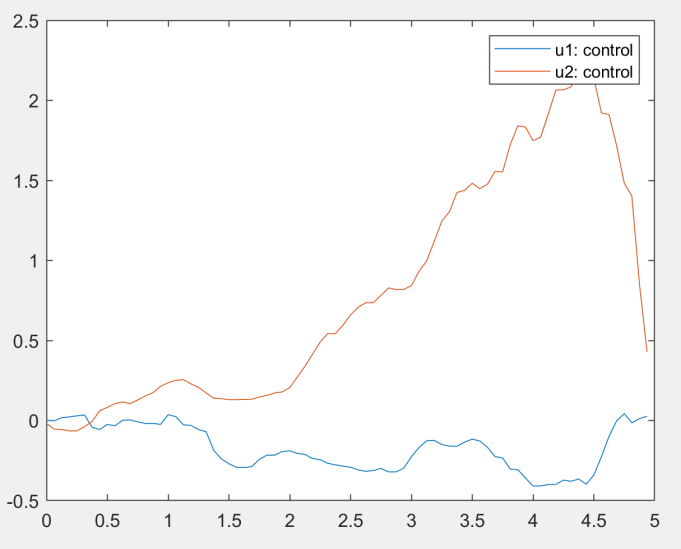


Fig. 46. Actuator Effort trajectory

This trajectory satisfies our constraints, meaning it satisfies the dynamics and passes through the position of the ball with the specified velocity. Using discrete methods, we can linearize about this trajectory and stabilize it. While this was not fully accomplished within this paper, it is a good jumping off point for further studies.

B. Contributions

Kaylee:
Abstract/Introduction, Control Objective, Controllability, Full State Feedback Control, Closed-Loop System Diagram, Feedback Control to meet specifications

Ricky:
Dynamics, Linearization, Observability, Luenberger Estimator, System Transfer Function, PID Control

Maxwell:
System Model, Dynamics, Trajectory Simulation and Analysis, Viability of Linearization, LQR Controller, Disturbance and Measurement analysis, Conclusion, Further Research and Trajectory Planning.

REFERENCES

[1] Xu Chunquan et al. *Motion Control of a Golf Swing Robot*. Report 227-299. Journal of Intelligent Robotic Systems, 2009.