Inverted Pendulum Controllers

Garrett Wilson and Nathan Zimmerly

ENGR 352

May 25, 2016

Contents

1 PID Controller

Can add if you want Garrett. I have it all written up in a seperate document.

2 LQR State Space Controller

2.1 4th Order

2.1.1 State-Space Modeling

Below is the free-body diagram of the inverted pendulum and cart

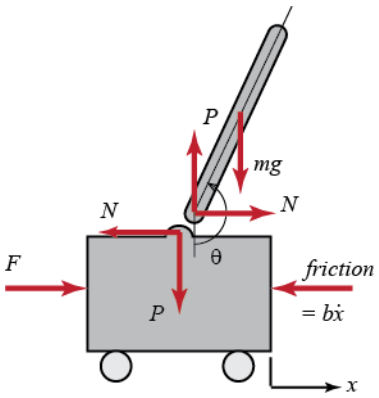

Figure 1: Source: http://ctms.engin.umich.edu/

Summation of carts forces in the horizontal direction

$$F = M\ddot{x} + b\dot{x} + N \tag{1}$$

Summation of the pendulum forces in the horizontal direction

$$N = m\ddot{x} + ml\ddot{\phi}cos(\phi) - ml\dot{\phi}^2 sin(\phi) \tag{2}$$

Substituting 2 into 1

$$F = M\ddot{x} + b\dot{x} + m\ddot{x} + ml\ddot{\phi}cos(\phi) - ml\dot{\phi}^2 sin(\phi) \tag{3}$$

Next, the sum of forces perpendicular to the pendulum

$$Psin(\phi) + Ncos(\phi) - mgsin(\phi) = ml\ddot{\phi} + m\ddot{(x)}cos(\phi) \tag{4}$$

Summing tthe moments about the centroid of the pendulum

$$-Psin(\phi) - Ncos(\phi) = I\ddot{\phi}/l \tag{5}$$

Combining 4 and 5 leads to

$$(I + ml^2)\ddot{\phi} + mglsin(\phi) = -ml\ddot{x}cos(\phi) \tag{6}$$

Small angle approximations. θ replaces ϕ so that the angle $\theta = 0\,^\circ$ is straight up.

$$cos(\phi) = cos(\pi + \theta) \approx -1 \tag{7}$$

$$sin(\phi) = sin(\pi + \theta) \approx -\theta \tag{8}$$

$$\dot{\phi} = \dot{\theta} \approx 0 \tag{9}$$

Plugging 7 and 8 into 3 and 6 leads to

$$F = (M+m)\ddot{x} + b\dot{x} - ml\ddot{\theta} \tag{10}$$

$$(I + ml^2)\ddot{\theta} - mgl\theta = ml\ddot{x} \tag{11}$$

Motor Equations

$$Li + Ri = V - k_m\dot{\theta} \tag{12}$$

$$\tau = k_m * i \tag{13}$$

To arrive at state space equations add 12 and 13 into 10 and 11. Note: L = 0.

Plugging 13 into 12 leads to

$$F = \frac{Rk_m}{r(V - k_m\dot{\theta})} \tag{14}$$

Finally, plugging 14 into 10 and 11 and solving for $\ddot{x}$ and $\ddot{\theta}$ leads to the state space equations below

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{(I+ml^2)/(ml)(-(b+K^2/(Rr^2))}{(M+m)(I+ml^2)/(ml)-ml} & \frac{(I+ml^2)/(ml)(M+m)g}{[(M+m)(I+ml^2)/(ml)-ml]-g} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-(b+K^2/(Rr^2))}{(M+m)(I+ml^2)/(ml)-ml} & \frac{(M+m)g}{(M+m)(I+ml^2)/(ml)-ml} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{(I+m*l^2)/(m*l)*(K/(R*r)}{(M+m)(I+ml^2)/(ml)-ml} \\ 0 \\ \frac{K/(R*r)}{(M+m)(I+ml^2)/(ml)-ml} \end{bmatrix} v \tag{15}$$

$$\begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{16}$$

2.1.2 Poles, Stability, and Observability

Now that the state-space equations are found, the poles of the model are equal to the eigen values of the A matrix. The original Poles are shown below:

1. 0.00000 + 0.00000i
2. -9.9916 + 0.00000i
3. -4.9981 + 0.00000i
4. 5.1345 + 0.00000i

Because one of the poles is positive, the system is unstable. We will fix this with a controlled matrix A in the next section.

If the rank of the controllability matrix shown below equals the number of states the system (n) is made up of, then the system is controllable. In our case the rank of R equals 4 so our system is controllable. This means that our system can reach any state.

$$R = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \tag{17}$$

If the rank of the observability matrix shown below equals the number of states the system is made up of, then the system is observable. In our case the rank of O eqauls 4 so our system

is observable. This means that in any possible situation, the current state of our system can be determined using only the outputs.

$$O = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{18}$$

2.1.3 LQR Controller

A linear-quadratic regulator (LQR) weights factors depending on their importance using the nxn Q matrix, while minimizing another factor using the variable R. In our case, the weighted factors were the position of the cart (x) and the angle of the pendulum (θ). Also, the factor that we minimized is the velocity of the cart. A good starting point for Q is C'C evenly weighting the factors. However, we care much more about the pendulum staying vertical than we do the position staying at 0 m. Because of this, after some tests we weighted x at 1000 and θ at 100,000. leaving the Q matrix below. Also, a good place to start the limiting factor is 1, but we change R = 0.1.

$$Q = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 100,000 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

In LQR controllers, the state-feedback control gains vector k is used to move the poles in the system. It can be calculated by minimizing the cost function J. We used Octave's LQR function to calculate our k vector. Once calculated, the corrected A matrix can be calculated using the equation below.

$$A_c = [A - Bk] \tag{20}$$

2.1.4 Precompensator

At this point, our pendulum seemed to stay up, but the cart would not stay at the center. Because of this, we had to add a precompensator to account for steady-state error. Because the cart's position had steady-state error, we chose the C matrix below.

$$C_n = [1000] \tag{21}$$

By replacing the state space C matrix with C_n, $\bar{N}$ can be calculated. We used Octave's rscale function to calulate $\bar{N}$. $\bar{N}$ is a scale factor that eliminates the stead-state error of a system to a step input. It should be noted that $\bar{N}$ only works for single input systems, which is why we could use it to only eliminate the steady state error in the carts position. Once $\bar{N}$ is calculated, the state space equation uses the corrected A matrix, B $\bar{N}$, C, and D for the A, B, C, and D matrices respectively.

2.1.5 Adding an Observer-Based Estimator

Previous work assumes full-state feedback, which means that x, $\dot{x}$, θ, and $\dot{\theta}$ are all being measure. However, only x and θ are being measured so we must add observer based control. Below are the previous block diagram, and the updated block diagram with the observer-based estimator.

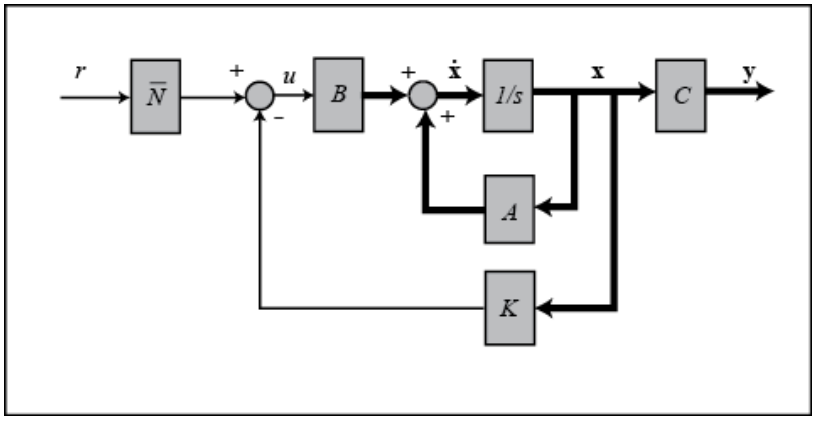

Figure 2: Original

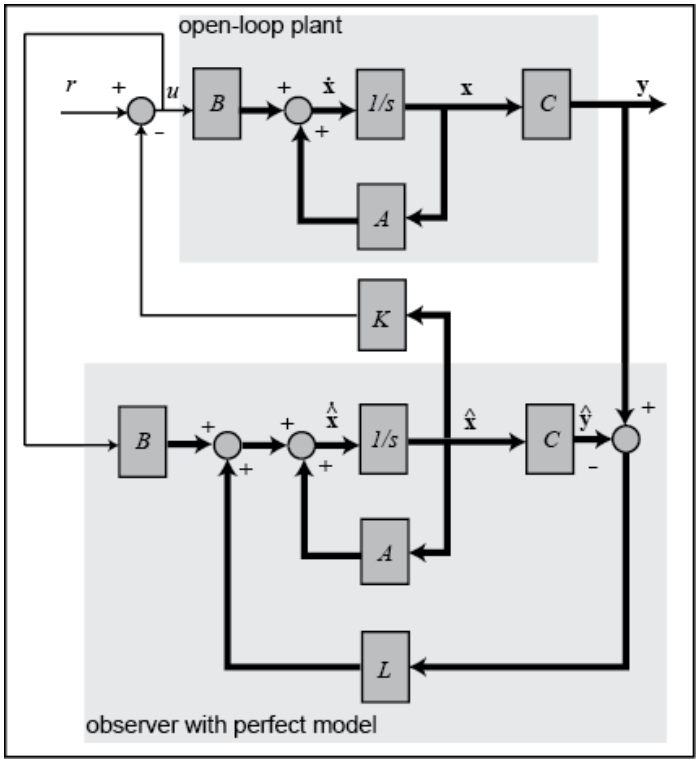

Figure 3: With Precompensator

The speed of convergence depends on the poles of the estimator. Since we are going to use the estimation before the actual event accurs, we want the estimation to converge must faster. Because of this we want the poles of our estimator to be at least four times faster than the slowest controller pole. If the estimator poles are too fast it can cause error in the measurement, so usually the poles of the estimator should not be more than 10 times larger than the smallest controller pole. Also, the estimator poles should be close together. The current poles of our C matrix are listed below. Placing the estimator poles P at [-30, -31, -32, -33] was found to work after several trials. Note: we did not place all the estimator poles in the same position because the Octave place function used to find L does not support that.

1. -50.8860 +50.1382i
2. -50.8860 -50.1382i
3. -0.7136 + 0.6863i
4. -0.7136 - 0.6863i

Based on the estimator poles, we placed the Octave place function which computes L. Based on the Original A, B, C, and D matrices for state space, the final state space equations are shown below:

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A - Bk & Bk \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} + \begin{bmatrix} B\bar{N} \\ 0 \end{bmatrix} r \tag{22}$$

$$y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} X \\ e \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} r \tag{23}$$

2.2 6th Order

3 Future Work

Future work for this pendulum includes using a UFIR or kalman filter to eliminate error from noise. Currently, the Pendulum works (with the wrong radius value), but it does have some noise cause it to rock back and forth. A kalman filter should help quite efficiently, however error matrices have to be found through tests which is difficult. There is another way to calculate the error matrices through iteration and there is an octave package called ALS which is supposed to help find them. The UFIR method is more complex, making more work for the computer running the program. However, it works significantly than the original kalman filter, unless the error matrices are exact in which case the kalman filter is slightly better. We have spent time trying to get both methods to work without success, however we're fairly certain that if we had more time we could get at least one of the methods to work.