Ashton Billings, Alan Lu, Hamza Lutfi

ECE 486: Final Lab Report

Fall 2024

Tuesday Section B

TA: Junjie Gao

December 20, 2024

1 Introduction

In this lab, we studied and learned to control a reaction wheel pendulum (RWP). These systems typically include a free-spinning and un-actuated pendulum with an actuator-controlled wheel, or rotor, attached to its end, as we had in this lab. Our system also included two encoders, one for the pendulum and one for the rotor. The idea behind this system is that by using Newton's Third Law of equal and opposite reactions, by actuating the rotor to spin one way, this will induce an equal and opposite reaction in the pendulum, turning it the other way. Our rotor actuator is a DC motor, while our two encoders measure the relative offset of the pendulum and rotor, respectively, from a starting or reference point. This gives our system one actuator and two sensors to control it. The question is now, what are we trying to control it to do? In this lab, our main task is to design a controller to stabilize the RWP system at an equilibrium position, mainly its unstable equilibrium position, even if it encounters disturbances that attempt to move it away from this position. The two static equilibrium positions we care about for this lab are the upright position, and the downward hanging position. These positions are said to be in static equilibrium as they are 'static' because their velocity at these points is zero and are said to be in 'equilibrium' as the net forces and/or torques of the system are zero or balanced. The upright position is inherently unstable, having a large potential energy, as any slight disturbance at this position will cause it to fall and stabilize at the downright stable equilibrium position.

To access the data from our encoders as well as drive the actuator, we use a real-time Windows Target in simulink and the C6X ENC chip onboard the PC to interface with the RWP system. This system contains more than just the pendulum, rotor, two sensors, and actuator, however, as it contains other electronic hardware. The main purpose of this hardware is to convert our actuator input from the PC to the proper inputs to drive the motor, and to convert the 'ticks' or pulses coming from the encoders to a usable quantity to then send to the PC.

In our model of the system, we think of the motor as a current transducer, in which our input u from our pc is converted to a current i we drive the DC motor with in which the current provided to the motor is proportional to its torque τ. As such, the hardware needs to be able to turn this input u into a current to then drive the motor. In our case, our input is limited too the range of $\pm 10u$ in which these maximum values correspond to the maximum current our system can provide our motor.

As for the sensors, these sensors are optical encoders, that 'tick' as they rotate. There are 'relative' and 'absolute' encoders, here we use relative encoders. relative encoders work by outputting 'ticks' or logical highs on the output line periodically as the encoder turns. The motor encoder ticks 4000 times per revolution, while the pendulum encoder ticks 5000 times per revolution. This ticking output line isn't very useful to our simulink system as for this we need a quantity, the amount of ticks, rather than a continuous stream of ticks, as this amount corresponds to the encoder's angle. As such our hardware needs to first set a zero point, as in deciding when to begin counting, then determine its position by counting these ticks as the encoder spins. The one important caveat is that we need to decrease this counted value if our encoder spins in the reverse direction. Thus, our hardware first converts these pulses into a position value before sending it to the PC to be used.

2 Mathematical Model

2.1 Deriving the System

Now that we have a good understanding of the interface between our RWP system and our PC, as well as a good understanding of how to interpret what we are sending and receiving, we can now attempt to control it. Before we can do this, however, we need to know what we are controlling, as in, create a mathematical model of the system.

With the idea behind our system being that by turning the rotor one way, we can induce the pendulum to turn the other way, reasonable variables to define are the angle of the pendulum and the angle of the rotor. These angles need to be set relative to some reference, so here we set them relative to the downward hanging position. The main important caveat of this definition is that while the pendulum angle θ_p is directly related to the output of the pendulum encoder system φ_p, the rotor angle θ_r is actually the sum of the output of the two encoders φ_p and φ_r as if the pendulum angle deviates from the downward position, so will the wheel. As such, the following variables of our system are defined below:

$$\begin{aligned} \theta_p &= \varphi_p, \\ \theta_r &= \varphi_p + \varphi_r. \end{aligned} \tag{1}$$

In this lab, we create the equations of motion of our system through the Lagrangian method. This method works by defining a set of generalized coordinates, computing the kinetic energy K and potential energy V in terms of these coordinates, finding the Lagrangian of the system which is just $L = K - V$, then solving the equation below:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} = \tau_k, \quad k = 1, \ldots, n. \tag{2}$$

In our case, our generalized coordinates are θ_p and θ_r, with the physically meaning behind τ_k in the equation above being the torque applied in the direction of these generalized coordinates. To find K and V and then solve the Lagrangian, we first need to establish the following variables:

m_p	mass of the pendulum and motor housing/stator
m_r	mass of the rotor
m	combined mass of rotor and pendulum
J_p	moment of inertia of the pendulum about its center of mass
J_r	moment of inertia of the rotor about its center of mass
ℓ_p	distance from pivot to the center of mass of the pendulum
ℓ_r	distance from pivot to the center of mass of the rotor
ℓ	distance from pivot to the center of mass of pendulum and rotor
k	torque constant of the motor
i	input current to motor

As well as establish the following relationships:

$$J = J_p + m_p\ell_p^2 + m_r\ell_r^2, \tag{3}$$
$$m = m_p + m_r, \tag{4}$$
$$m\ell = m_p\ell_p + m_r\ell_r. \tag{5}$$

We start off by finding the total Kinetic Energy K:

$$K = K_{\text{pendulum}} + K_{\text{rotor}}$$

where:

$$K_{\text{pendulum}} = \frac{1}{2}J_p\dot{\theta}_p^2 + \frac{1}{2}m_pl_p^2\dot{\theta}_p^2$$

$$K_{\text{rotor}} = \frac{1}{2}J_r\dot{\theta}_r^2 + \frac{1}{2}m_rl_r^2\dot{\theta}_p^2$$

Therefore:

$$K = \dot{\theta}_p^2\left(\frac{1}{2}J_p + \frac{1}{2}m_pl_p^2 + \frac{1}{2}m_rl_r^2\right) + \frac{1}{2}J_r\dot{\theta}_r^2$$

Next, we find the change in potential energy of our system V by multiplying mg by the change in vertical height of the rotor. This yields the following equation:

$$V = mgl(1 - \cos\theta_p)$$

We can now find Lagrange's equations by finding L and doing the derivative calculations below and substitute them into (2):

$$\frac{d}{dt}\left(\frac{dL}{d\dot{\theta}_p}\right) = (J_p + m_pl_p^2 + m_rl_r^2)\ddot{\theta}_p$$

$$\frac{dL}{d\theta_p} = -mgl\sin\theta_p$$

$$\frac{d}{dt}\left(\frac{dL}{d\dot{\theta}_r}\right) = J_r\ddot{\theta}_r$$

$$\frac{dL}{d\theta_r} = 0$$

We can now finally derive our equations of motion by plugging in the parameters from (3), (4), and (5) into our Lagrangian Equation. This will yield the following final result:

$$\begin{cases} \ddot{\theta}_p + \omega_{np}^2\sin\theta_p = -\frac{k}{J}i, \\ \ddot{\theta}_r = \frac{k}{J_r}i. \end{cases} \tag{6}$$

As one final step, since we will be working with input u, by noting the relationship $ki = k_u u$, and defining new variables $a = \omega_{np}^2$, $b_p = \frac{k_u}{J}$, and $b_r = \frac{k_u}{J_r}$, we can represent our system as:

$$\begin{cases} \ddot{\theta}_p + a\sin\theta_p = -b_p u, \\ \ddot{\theta}_r = b_r u. \end{cases} \tag{7}$$

2.2 Linearizing the System

We can see from these equations of motion that our system is not linear. Although non-linear control methods exist, these are outside of our scope, and instead we opted to linearize our system around the equilibrium point $\theta_p = \pi$. To do this, we define new delta-angle variables $\delta\theta_p$ and $\delta\theta_r$ in which $\delta\theta_p = \theta_p - \pi$ and $\delta\theta_r = \theta_r$. If our system is near this equilibrium point, this linearized system will track the original non-linear system very closely, but will deviate greatly once we are far from it. Below is how we derived the linearized model:

We started off by finding the equilibrium point by evaluating our equations of motion at $\theta_p = \pi$ in order to find $\bar{u}$:

$$a\sin\pi = -b_p\bar{u} \implies \bar{u} = 0$$

It is worth noting that our second equation is already linear and linearizing it will not be necessary. We will only evaluate the partial derivatives of the first equation with respect to θ_p and u at the equilibrium point:

$$\frac{\partial f}{\partial\theta_p} = -acos\theta_p\bigg|_{\bar{\theta}_p=\pi,\bar{u}=0} = a$$

$$\frac{\partial f}{\partial u} = -b_p$$

this will give us the model below:

$$\delta\ddot{\theta}_p - a\delta\theta_p = -b_p\delta u$$

since $\delta\ddot{\theta}_p = \ddot{\theta}_p$ and $\delta u = u$ because $\bar{u} = 0$, our final model will look like the one below:

$$\begin{cases} \ddot{\theta}_p - a\delta\theta_p = -b_p u, \\ \ddot{\theta}_r = b_r u. \end{cases} \tag{8}$$

We then converted this linearized model to state space form, as this will be a convenient representation once we attempt to control it. To do this we first need to define what our states are. Since we care about controlling our pendulum position through changing our rotor's position, our state variables are θ_p, $\dot{\theta}_p$, θ_r, and $\dot{\theta}_r$. Since we are using a linearized model of this system, we actually care about the linearized state variables $\delta\theta_p$, $\delta\dot{\theta}_p$, $\delta\theta_r$, and $\delta\dot{\theta}_r$, noting that $\delta\dot{\theta}_p = \dot{\theta}_p$ and $\delta\dot{\theta}_r = \dot{\theta}_r$. From this, we can derive the state space form below:

$$\begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \\ \dot{\theta}_r \\ \ddot{\theta}_r \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ a & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & b & 0 \end{bmatrix} \begin{bmatrix} \delta\theta_p \\ \dot{\theta}_p \\ \delta\theta_r \\ \dot{\theta}_r \end{bmatrix} + \begin{bmatrix} 0 \\ -b_p \\ 0 \\ b_r \end{bmatrix} u \tag{9}$$

where:

$$x = \begin{bmatrix} \delta\theta_p \\ \dot{\theta}_p \\ \delta\theta_r \\ \dot{\theta}_r \end{bmatrix} \dot{x} = \begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \\ \dot{\theta}_r \\ \ddot{\theta}_r \end{bmatrix} \tag{10}$$

3 Full State Feedback Control: Friction Compensation

3.1 The PD Controller

Now that we have a good mathematical model of our system linearized around our unstable equilibrium position $\theta_p = \pi$, we can develop a PD controller to control the system around this position. Given that we are attempting to stabilize $\delta\theta_p$, we first start with a two state feedback controller. By two state we mean only focusing on controlling state space variables $\delta\theta_p$ and $\delta\dot{\theta}_p$ ignoring how our controller may affect our other state space variables $\delta\theta_r$ and $\delta\dot{\theta}_r$. This reduces our system too:

$$\begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ a & 0 \end{bmatrix} \begin{bmatrix} \delta\theta_p \\ \dot{\theta}_p \end{bmatrix} + \begin{bmatrix} 0 \\ -b_p \end{bmatrix} u \tag{11}$$

With this, we can use full state feedback control methods in which $u = -K\boldsymbol{x}$ to determine a controller matrix K, and hence a PD controller, such that our system meets the given requirements that $\omega_{np} > \omega_{np}^{'}$ in which $\omega_{np} = \sqrt{\frac{mgl}{J}}$ and $\omega_{np}^{'} = \sqrt{\frac{mgl}{J+J_r}}$ and that the damping coefficient of the system is $\zeta < \frac{1}{\sqrt{2}}$. Using the place() command in matlab, we found the following controller matrix K:

$$K = \begin{bmatrix} 210 \\ 22.3 \end{bmatrix} \tag{12}$$

Through simulation, we find that with a constant disturbance, our rotor will undergo a constant acceleration causing it to get arbitrarily large due to not feeding back rotor velocity information into our controller. Although in actuality our rotors speed will not get arbitrarily large as the force of friction will eventually balance this force, this result leads to an impractical controller. As such, to control the rotor's velocity, we designed a three-state feedback controller in which $\delta\dot{\theta}_r$ is our third state, giving us the new system and controller matrix K below:

$$\begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \\ \ddot{\theta}_r \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ a & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\theta_p \\ \dot{\theta}_p \\ \dot{\theta}_r \end{bmatrix} + \begin{bmatrix} 0 \\ -b_p \\ b_r \end{bmatrix} u \tag{13}$$

$$K = \begin{bmatrix} 210 \\ 22.3 \\ 0.036 \end{bmatrix} \tag{14}$$

In both of our controller systems, we had friction compensation along with our PD controller. This greatly helped our control process as the state space models of our system did not include the effects of friction, so we could expect erroneous results such as steady state error without its inclusion. First off, we noted that the pendulum's friction is negligible, while the rotor's friction is not. Therefore in our case, we only worried about compensating for the rotor's friction. Next, we used a PI controller to compensate for friction. We did this by developing a PI controller in conjunction with equation (7.b) such that such a controller would result in a constant value with zero steady-state error for state variable $\omega_p = \dot{\theta}_p$. Then, we noted that in order for our PI controller to have the rotor spin at a constant steady state velocity, the controller's control effort u must be equal and opposite to the friction control effort. Thus, by driving the motor at different velocities, and noting the steady-state control effort u and hence the friction control effort, we could linear fit these results and determine its slope b, the viscous friction coefficient, and its y-intercept c, the coefficient of static friction. Doing this for both the forward and reverse direction, we found the following friction coefficients of our rotor shown below:

+ Friction Coeff		- Friction Coeff	
b_+	0.0121	b_-	0.0118
c_+	0.6355	c_-	-0.7024

Table 1: Friction Coefficients

With all of this, we result in the final simulink system shown below:

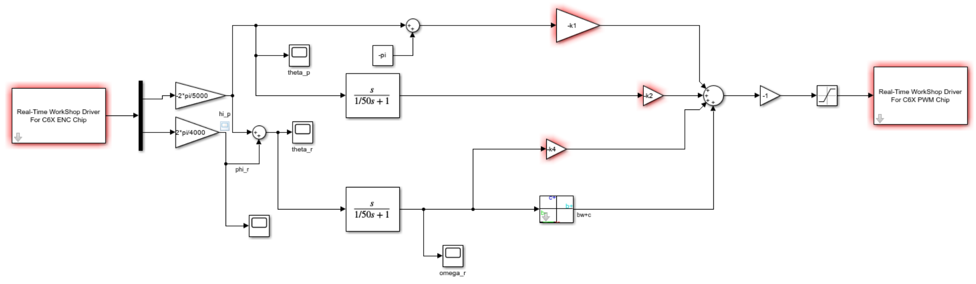

Figure 1: PD Controller System

3.2 Mathematical Proof of Stability

We aim to show that the linearized, frictionless closed-loop system in the inverted position is stable. Specifically, we need to prove that $\theta_p(t)$ goes to π as $t \to \infty$ under the given 3-state feedback controller.

The linearized system can be expressed as:

$$\dot{x} = Ax + Bu,$$

where $x = \begin{bmatrix} \delta\theta_p \\ \dot{\theta}_p \\ \delta\theta_r \\ \dot{\theta}_r \end{bmatrix}$ represents the state vector, and

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ a & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -b_{\mathrm{p}} \\ 0 \\ b_{\mathrm{r}} \end{bmatrix}.$$

3.2.1 Step 1: Equilibrium Points

To find the equilibrium points, substitute $\dot{x} = 0$ into the linearized system:

$$Ax - BK_{\mathrm{p}}x = 0.$$

Solve for x with the given controller u to determine the equilibrium points. By setting $\dot{x} = 0$, the only equilibrium point occurs at:

$$x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

This corresponds to the state where $\delta\theta_p = 0$, $\dot{\theta}_p = 0$, and $\dot{\theta}_r = 0$, which $\delta\theta_p = \theta_p - \pi$ Therefore, the equilibrium position is at $\theta_p = \pi$, which is what we expect.

3.2.2 Step 2: Reduced State Analysis

The hint indicates that θ_r does not influence other states. Therefore, we reduce the system to a smaller 3×3 system by considering the states $\Delta\theta_p$, $\dot{\theta}_p$, and $\dot{\theta}_r$. The reduced system matrices are:

$$A_{\text{reduced}} = \begin{bmatrix} 0 & 1 & 0 \\ a & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_{\text{reduced}} = \begin{bmatrix} 0 \\ -b_{\mathrm{p}} \\ b_{\mathrm{r}} \end{bmatrix}, \quad x_{\text{reduced}} = \begin{bmatrix} \delta\theta_p \\ \dot{\theta}_p \\ \dot{\theta}_r \end{bmatrix}.$$

3.2.3 Step 3: Stability of the Equilibrium Point

To prove stability, we check the eigenvalues of $A_{\text{reduced}}x - B_{\text{reduced}}K_{\mathrm{p}}x_{\text{reduced}}$. The equilibrium point $x_{\text{reduced}} = 0$ is stable if all eigenvalues of this smaller system have negative real parts.

Compute the eigenvalues by solving the characteristic equation:

$$\det(sI - A_{\text{reduced}}x + B_{\text{reduced}}K_{\mathrm{p}}x_{\text{reduced}}) = 0.$$

They follow our designed poles for the system, which is -6+j6, -6-j6 and -6 for each entry respectively. Therefore, since the smaller system is stable and equilibrium position is at π, the whole three-state feedback system is stable.

3.3 Simulation Specification Results

	Two-State		Three-State	
Max IC Deviation	0.135	1.2	0.125	1.18
Max Pulse	7		8.2	
Max Disturbance	5.33		7.1	

Table 2: Robustness for Friction Compensation Controller

3.4 Experimental Results

To initiate the controller, we must first bring the pendulum within the equilibrium position we linearized around π. It needs to be near this point as our controller was designed around the linearized model of the system which is only accurate near this equilibrium point. As the pendulum nears this point, our controller begins to spin the rotor in such a way that it attempts to spin the pendulum in the opposite way. This meant that if our pendulum is at a location to the right of the upward position, our rotor will spin clockwise as to induce a torque in the pendulum to counter-clockwise, bringing it to the left. If the pendulum is located to the left, the rotor will spin counter-clockwise now to induce a torque in the pendulum clockwise, moving it to the right. Once it's at or very close to its equilibrium point, the motor shuts off, as it no longer needs to attempt to bring it to this position. We found that in order to get our expected results in which the controller holds the pendulum up, the initial position of the pendulum must be very near to this equilibrium position, as we'd expect from our linearized model. As expected from our robustness results, once we used the three-state controller, this maximum deviation from the equilibrium point and hence control robustness of the system increased.

4 Full State Feedback Control: Decoupled Observer

4.1 Need for an Observer and The Decoupled Observer

The need for an observer can be answered by answering the question: what are we trying to observe, and why can't we directly observe them? For full state feedback, the answer is we want to observe our states, which are $\delta\theta_p$, $\dot{\theta}_p$, $\delta\theta_r$, and $\dot{\theta}_r$, and since our encoders only give us φ_p and φ_r, we can only directly observe states $\delta\theta_p$ and $\delta\theta_r$. Thus, if using the observer method, an observer is necessary to determine states $\dot{\theta}_p$ and $\dot{\theta}_r$. In our case we use a Luenberger observer, which is modeled as the following system:

$$\begin{cases} \dot{\hat{x}} = \boldsymbol{A}\hat{x} + \boldsymbol{B}u + \boldsymbol{L}(y - \hat{y}), \\ \hat{y} = \boldsymbol{C}\hat{x} + \boldsymbol{D}u. \end{cases} \tag{15}$$

From this, we know that we need to place the eigenvalues of $\boldsymbol{A} - \boldsymbol{LC}$ far away relative to the eigenvalues of controller matrix $\boldsymbol{K}$. Using the matlab command place(), we found:

$$\boldsymbol{L} = \begin{bmatrix} 194.448 & 6.827 \\ 9515 & 662.7 \\ -3.726 & 199.55 \\ -360.8 & 9997 \end{bmatrix} \tag{16}$$

By analysis of equation (9), we note that matrix A is in whats called block diagonal form. This allows us to 'decouple' the 4-state observer design into the following two 2-state observer design:

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{1,2} \\ \dot{\boldsymbol{x}}_{3,4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{1,2} \\ \boldsymbol{x}_{3,4} \end{bmatrix} + \begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{Q} \end{bmatrix} \boldsymbol{u} \tag{17}$$

$$\dot{\boldsymbol{x}}_{1,2} = \boldsymbol{M}\boldsymbol{x}_{1,2} + \boldsymbol{P}\boldsymbol{u}, \quad \dot{\boldsymbol{x}}_{3,4} = \boldsymbol{N}\boldsymbol{x}_{3,4} + \boldsymbol{Q}\boldsymbol{u} \tag{18}$$

$$\begin{aligned} \begin{bmatrix} \dot{\boldsymbol{x}}_{1,2} \\ \dot{\boldsymbol{x}}_{3,4} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ a & 0 \end{bmatrix} \boldsymbol{x}_{1,2} + \begin{bmatrix} 0 \\ -b_p \end{bmatrix} \boldsymbol{u}, \quad \boldsymbol{C}_{1,2} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \\ \begin{bmatrix} \dot{\boldsymbol{x}}_{3,4} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}_{3,4} + \begin{bmatrix} 0 \\ b_r \end{bmatrix} \boldsymbol{u}, \quad \boldsymbol{C}_{3,4} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \end{aligned} \tag{19}$$

What this decoupling also tells us is that the dynamics of states $\delta\theta_p$, $\dot{\theta}_p$ and $\delta\theta_r$, $\dot{\theta}_r$ are independent of each other. Note this does not mean that the current state of $\boldsymbol{x}_{\mathbf{1,2}}$ wont affect the states of $\boldsymbol{x}_{\mathbf{3,4}}$, as $\boldsymbol{x}_{\mathbf{1,2}}$ affects control input u which in result affects $\boldsymbol{x}_{\mathbf{3,4}}$, and vice-versa. In designing our observer system, we wish set the eigenvalues of $\boldsymbol{A} - \boldsymbol{LC}$ to be fast relative to the controller poles. Given $\boldsymbol{A} - \boldsymbol{LC}$ below:

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \boldsymbol{L} = \begin{bmatrix} 194.448 & 6.827 \\ 9515 & 662.7 \\ -3.726 & 199.55 \\ -360.8 & 9997 \end{bmatrix}, \quad (\boldsymbol{A} - \boldsymbol{LC}) = \begin{bmatrix} -l_{11} & 1 & -l_{12} & 0 \\ a - l_{21} & 0 & -l_{22} & 0 \\ -l_{31} & 0 & -l_{32} & 1 \\ -l_{41} & 0 & -l_{42} & 0 \end{bmatrix} \tag{20}$$

We note from the previous decoupling discussion that states $\boldsymbol{x}_{\mathbf{3,4}}$ as $\boldsymbol{x}_{\mathbf{1,2}}$ are independent of each other, implying that ℓ_{11}, ℓ_{11}, ℓ_{11}, and ℓ_{11} are unnecessary and worse yet introduce unnecessary noise due to model errors. This is the main pro of using a two 2-state observer design in this case over a 4-state observer design. Hence, below is our new Luenberger observer:

$$\boldsymbol{L} = \begin{bmatrix} 200 & 0 \\ 10140 & 0 \\ 0 & 194 \\ 0 & 9400 \end{bmatrix} \tag{21}$$

From all this, we get the following simulink model:

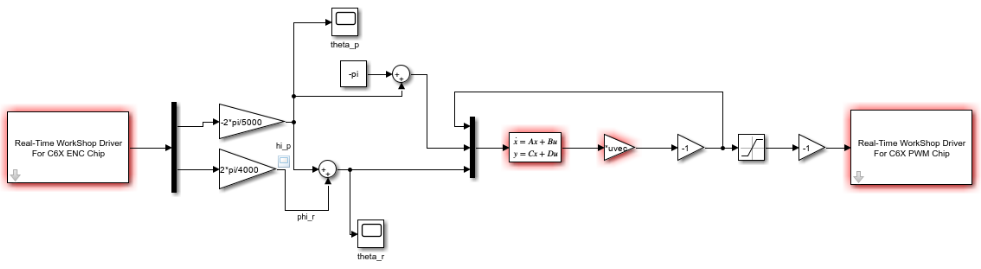

Figure 2: Decoupled Observer Controller

4.2 Proof of Observer State to Real State Convergence

To prove that the observer states converge to the real states over time, we begin with the error definition:

$$e = x - \hat{x}$$

Step 1: Differentiate the Error Equation

Differentiating e with respect to time:

$$\dot{e} = \dot{x} - \dot{\hat{x}}$$

Step 2: Substitute Equations for $\dot{x}$ and $\dot{\hat{x}}$

The dynamics of the actual states are given by:

$$\dot{x} = Ax + Bu$$

The observer dynamics are given by:

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

where $y = Cx$ and $\hat{y} = C\hat{x}$. Substituting these into the error dynamics:

$$\dot{e} = Ax + Bu - (A\hat{x} + Bu + L(Cx - C\hat{x}))$$

Simplify the terms:

$$\dot{e} = Ax - A\hat{x} - L(Cx - C\hat{x})$$

Factorize $x - \hat{x}$:

$$\dot{e} = A(x - \hat{x}) - LC(x - \hat{x})$$

$$\dot{e} = (A - LC)e$$

Step 3: Stability Analysis

The error dynamics are governed by:

$$\dot{e} = (A - LC)e$$

To prove stability, we examine the eigenvalues of the matrix $A - LC$. If all eigenvalues have negative real parts, the system is stable, and $e \to 0$ as $t \to \infty$.

Step 3.1: Verify Equilibrium

At equilibrium ($\dot{e} = 0$):

$$(A - LC)e = 0$$

The only solution is $e = 0$ since $A - LC$ is non-singular and all eigenvalues are designed to stay in the left-half plane, which implies that the equilibrium is unique.

Step 3.2: Pole Placement

The observer gain L is chosen such that the poles of $A - LC$ are placed in the left-half plane (stable region). This ensures that the error e decays to zero over time.

4.3 Simulation Specification Results

	Observer	
Max IC Deviations	0.12	1.1
Max Pulse	5.5	
Max Disturbance	5	

Table 3: Robustness for Observer Controller

4.4 Experimental Results

For both the 4-state observer design and two 2-state observer design, our controller was able to stabilize the system around the pendulum equilibrium point $\theta_p = \pi$. We found that the 4-state observer design was more prone to destabilization which we expected due to introduced noise due to the unnecessary values of the Luenberger matrix. Once we removed these erroneous values, our controller fared far better in stabilizing our system.

5 Conclusion

In this lab, we were tasked with understanding, mathematically modeling, and finally controlling a reaction wheel pendulum system. To mathematically model our system, we used the Lagrangian method to derive the equations of motion of our system. With this, we first designed two full state feedback controllers, one with a 2-state design and another with a 3-state design, noting that we had to use friction compensation in these designs as our linearized model is based on our system being frictionless. We then explored using two different observer designs, one using a 4-state observer design, and another using a two 2-state observer design.

Our 2-state design only regulated the angle and angular speed of the pendulum, which although under perfect conditions worked in simulation, once we introduced disturbances, this caused one of our uncontrolled states the rotor angular speed to increase without bound, leading to an impractical controller. Thus to regulate the rotor's angular velocity we redesigned the controller to be a 3-state design in which this new third state was the rotor's angular velocity.

We then explored another approach to control, observer design. We used an observer system to estimate the states of our system to then be used in feedback. Using the normal approach of calculating the Luenberger matrix through matlab function place(), we found that although this original 4-state observer design worked, it was very unstable. Upon mathematical analysis of the system, we found that we can 'decouple' our system, showing us that infact the states related to the pendulum and states related to the rotor were dynamically independent. This means that the Luenberger terms relating these independent states were introducing unessesary noise into our system, reducing our robustness. Thus by removing these terms and instead opting for a two 2-state observer design, we were able to develop a far more robust observer controller.

Given that our first approach used our 'actual' states for state feedback, while our observer approach used 'estimates' of our states for state feedback, we expected our observer design to be worse than our first design. Despite this, however, we instead found that our observer design fared better than our first design. Perhaps this can be explained as our first design not being designed well enough. Another explanation is that to determine our states $\dot{\theta}_p$ and $\dot{\theta}_r$, we had to use approximation methods to determine them from the other states θ_p and θ_p. Thus the notion of our first design using our 'actual' state values while our observer design using 'estimates' may not be entirely accurate. Nontheless, all four designs were able to amply control the RWP system within the ranges determined by the linear approximation.

6 Extra Credit

6.1 Part 6: Up and Down Stabilizing Control

To finish the up and down stabilizing control of the RWP, we simply used a switch to detect the pendulum angle and switch to the appropriate equilibrium control. The switch component in simulink has three inputs and one output. If the second input is greater than zero, then the first input is the output; otherwise, the third input is the output. The first input is set to connect stabilizing control at $\theta_p = \pi$, and the third input is set to connect stabilizing control at $\theta_p = 0$. By setting the second input to $\theta_p - \pi/2$, when θ_p is zero, that is, at the lower equilibrium point, the third input will activated. Similarly, when θ_p is π, the first input will activated. Therefore, the control will automatically switch to appropriate equilibrium points.

Below is our simulink model we used for this part:

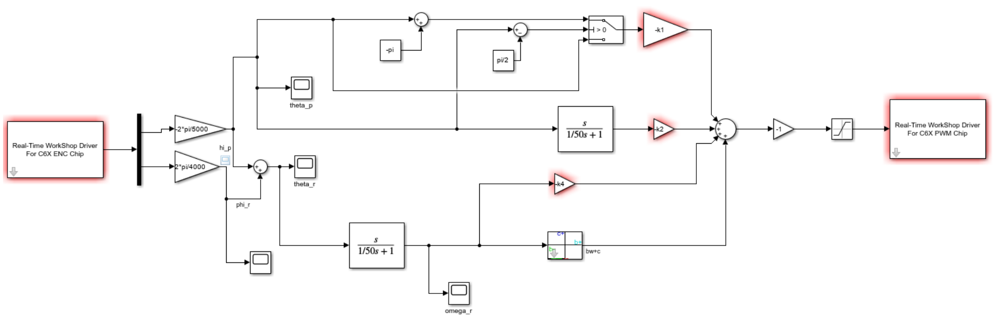
Real-Time WorkShop Driver For C6X ENC Chip
-2*pi/5000
2*pi/4000
hi_p
phi_r
theta_p
theta_r
-pi
pi/2
-k1
k2
k4
-1
omega_r
bw+c
Real-Time WorkShop Driver For C6X PWM Chip

Figure 3: Up and Down Stabilizing Controller

6.2 Part 7: Swing-up Control

The swing-up control is mainly divided into two steps: swing up the pendulum and stabilize the pendulum at $\theta_p = \pi$. To swing up the pendulum, we are using switches to change the sign of angular velocity when θ_p change its sign. By doing this, the pendulum will swing back and forth, and rise up slowly for each swing. However, only using this control will not make the pendulum stabilize when θ_p is around π. Also, since each time the direction that the pendulum is approaching the top is quite random(either CCW or CW), an additional control at equilibrium point $\theta_p = -\pi$ needs to be activated. This is also achieved by adding more switches into the system. By combining these sections with switches and adjusting gain parameters, we finally arrive at a Swing-up control, not robust, but succeeded in completing its work.

Below is our simulink model we used for this part:

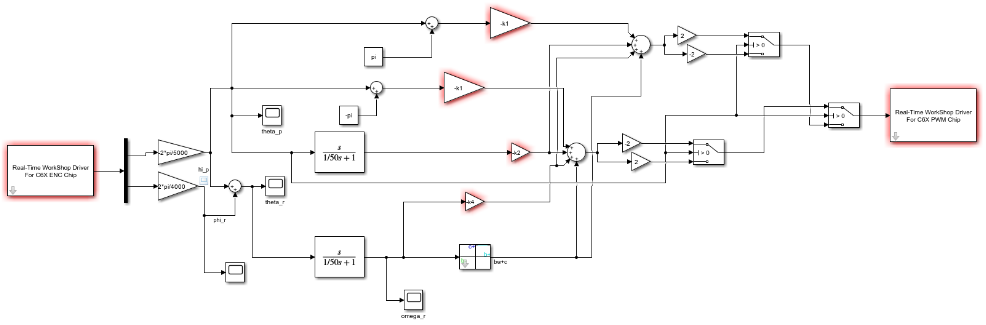
Real-Time WorkShop Driver For C6X ENC Chip
-2*pi/5000
2*pi/4000
hi_p
phi_r
theta_p
theta_r
pi
-pi
-k1
k2
k4
omega_r
bw+c
Real-Time WorkShop Driver For C6X PWM Chip

Figure 4: Swing-up Controller