UNIVERSITY OF ILLINOIS AT
URBANA-CHAMPAIGN

ECE 486 : Final Project Report

FALL 2017

Vishrut Dixit **Kush Gupta**

Teaching Assistant: Khaled Al-Shehri
Day of Laboratory Section: Wednesday

January 24, 2018

Contents

1 Introduction

The system we are working with for this project is the Reaction Wheel Pendulum; this simple pendulum uses a motor attached to a reaction wheel to produce a *reaction* torque on the motor and thus the pendulum. Our goal was to design a controller capable of balancing the pendulum in the inverted position (pointing upwards). In this report, we detail the steps that we took to develop a series of controllers that each built upon the previous in order to design an optimal controller for stabilizing the reaction wheel pendulum.

1.1 Sensors & Actuators

The inputs to our controller come from 2 *relative* encoders that measure the position of the pendulum and rotor. The output from the controller is used to control the motor, which is the only actuator in the RWP system. This is important to note, as our controller needed to track the states of 4 variables in the system and influence each based on the actions of only the single actuator that we had access too. This presented a novel challenge in that we needed do design a system that defined the relationships between each of the components of the reaction wheel pendulum before even beginning to develop a control methodology for it.

1.2 Equilibrium Positions & Implementation

There are two equilibrium positions associated with the system at $\theta_p = 0$ and $\theta_p = \pi$. Naturally, the pendulum will be stable equilibrium in its resting position at $\theta_p = 0$; similarly, when gravity alone exerts a downward force on the pendulum at exactly $\theta_p = \pi$, the system will be in unstable equilibrium. While this equilibrium is unstable, throughout the course of this project, we developed methods to be able to keep the system at or around that position using only the effects of the reaction wheel as a control input to the plant. We used SIMULINK, a block diagram package built into MATLAB to design and test these controllers, as it allowed us to develop and design controllers in a way that closely modeled the way that we have learned to analyze them throughout this course.

2 Mathematical Model

2.1 Derivation of differential equations from Lagrangian

The RWP system has two degrees of freedom; We define a set of generalized coordinates using the angles θ_p of the pendulum and θ_r of the rotor. To define the mathematical model, we declare the following variables:

- m_p - mass of the pendulum and motor housing/stator
- m_r - mass of the rotor
- m - combined mass of the 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

We compute the *kinetic energy* K and *potential energy* V in terms of the two degrees of freedom: θ_p and θ_r. Typically, potential energy is only a function of the generalized coordinates, but kinetic energy is a function of the generalized coordinates and their derivatives.

Let's start by expressing the kinetic and potential energy of the pendulum (KE_p and PE_p) using the generalized coordinates:

$$PE_p = mgh = mg(1 - cos(\theta_p))$$
$$KE_p = \frac{1}{2}J\omega_p^2 = \frac{1}{2}J\left(\frac{\partial\theta_p}{\partial t}\right)^2$$

Similarly, we can express the kinetic and potential energy of the rotor (KE_r and PE_r) as follows (note that the rotor potential energy is *always* 0 because the rotor height is less than the pendulum height):

$$PE_r = 0$$
$$KE_r = \frac{1}{2}J_r\omega_r^2 = \frac{1}{2}J_r\left(\frac{\partial\theta_r}{\partial t}\right)^2$$

Once the kinetic and potential energies are determined, we can define the *Lagrangian* as follows:

$$L(q_1,\ldots,q_n,\dot{q}_1,\ldots,\dot{q}_n)$$

So, the Lagrangian equations of the RWP system are:

$$L_q(q_p,\dot{q}_p) = KE_p - PE_p = \frac{1}{2}J_p\left(\dot{q}_p^2\right) - mgl_p(1-cos(_p))$$
$$L_r(q_r,\dot{q}_r) = KE_r - PE_r = \frac{1}{2}J_r\left(\dot{q}_r^2\right) - 0$$

Now, we can define τ_k is the torque in the q_k direction; with this, we can use the general equation for motion:

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

The new Lagrange equations for the system are as follows:

For the pendulum:

$$J\ddot{q}_p + mgl_p sin(q_p) = \frac{-tau_k}{J} = \frac{-ki}{J}$$
$$\ddot{q}_p + \omega_{np}^2 sin(q_p) = \frac{-k}{J}i$$

For the rotor:

$$\frac{J_r\ddot{q}_r}{J_r} = \frac{\tau_k}{J_r} = \frac{ki}{J_r}$$
$$\ddot{q}_r = \frac{k}{J_r}i$$

We can now account for friction. Because the mass on the pendulum is large enough that the friction on the pendulum link can be ignored; so, we only have to account for the friction on the rotor link. We can use the fact that the rotor is attached directly to the motor to model friction; the motor current i is generated as a pulse width modulation system. Due to current feedback, the current is proportional to the control command u from the computer. Furthermore, the control variable used in the computer is scaled such that 10 units correspond to maximum current. Using this, we can say:

$$ki = k_u u, \qquad |\, u \,| \leq 10$$

Assuming friction is a function of the rotor speed $F(\omega_r)$, we can model friction in command units as follows:

$$\ddot{\theta}_P + \omega_p^2 sin(\theta_p) = \frac{-k_u}{J}\left(u + F(\dot{\theta}_r)\right)$$
$$\ddot{\theta}_r = \frac{k_u}{J}\left(u + F(\dot{\theta}_r)\right)$$

Or equivalently,

$$a = \omega_{np}^2 = \frac{mgl}{J}, \quad b_p = \frac{k_u}{J}, \quad b_r = \frac{k_u}{J_r}$$

$$\ddot{\theta}_p + asin(\theta_p) = -b_p\left(u + F(\dot{\theta}_r)\right)$$
$$\ddot{\theta}_r = b_r\left(u + F(\dot{\theta}_r)\right)$$

2.2 Linearization into State Space Form

In the previous section, we derived the equations representing this system using the Lagrangian method, which are as follows:

$$\ddot{\theta}_p + \omega_{np}^2 \sin(\theta_p) = -b_p u$$
$$\ddot{\theta}_r = b_r u$$

Because this system is non-linear, it is necessary to linearize it around the point that we wish to stabilize it, namely π. This is done by defining delta-states that represent these new linearized coordinates:

$$\delta\theta_p = \theta_p - \pi$$
$$\delta\theta_r = \theta_r$$

Now, we can apply these new variables to our original equations, linearizing any components around 0 in the new states. This is done to $\sin(\theta_p)$.

$$\ddot{\theta}_p + \omega_{np}^2 \sin(\theta_p) = -b_p u \qquad \approx \qquad \ddot{\theta}_p + \omega_{np}^2 \delta\theta_p = -b_p u$$

Using the new set of linearized equations, we can write the full state-space model as follows:

$$\begin{bmatrix} \delta\dot{\theta}_p \\ \ddot{\theta}_p \\ \delta\dot{\theta}_r \\ \ddot{\theta}_r \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \omega_{np}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\theta_p \\ \dot{\theta}_p \\ \delta\theta_r \\ \dot{\theta}_r \end{bmatrix} + \begin{bmatrix} 0 \\ -b_p \\ 0 \\ b_r \end{bmatrix} u$$

3 Full State Feedback Control with Friction Compensation

3.1 Development of the PD Control with Friction Compensation

The first controller that we implemented was a two-state feedback model that only took into account the pendulum angle and velocity. We simplified the system developed earlier into a 2-dimensional one and used MATLAB to find and place poles at a position that would help us achieve the specs necessary to maintain the pendulum at the inverted position. In simulation, we found that this was generally feasible, however because there is no bound on the rotational velocity of the motor, we expected that it might be difficult to replicate this success on the physical model.

As such, we iterated upon the controller by adding in a third state, $\dot{\theta}_r$. Using this, we now had the ability to control more of the system parameters, and maintain a bound on the rotor speed so that it wouldn't increase indefinitely as a response to an erroneous control effort. In order to accomplish this we placed a new set of poles that included one corresponding to the velocity eigenvalue in the LHP, giving the controller the ability to maintain its behavior while controlling the system. This took the form of a three-state feedback control state-space model, which was similar to the one used above, however it included another equation relating the velocity of the rotor to the control input. When we did this, we found that the rotor would behave less erratically and as such the control was much smoother. The reaction wheel pendulum was as such able to stable much more easily at the inverted position and handle more disturbance.

At this point, we noticed that the reaction wheel would sometimes overcompensate for the arm movement, a problem that we sought to fix using friction compensation. In theory, friction was causing the controller to think that its output was not enough to cause the expected action, and as such it was overcompensating, causing the behavior that we were seeing. Since we have computed and evaluated the effects of friction compensation on a system before, we believed that by adding it, the controller may be able to function more accurately. We started with the values of the two friction coefficients that we obtained for the other motor used in the lab and implemented the compensation in the three-state model. We found that these values were effective at canceling out the effects that we were seeing before, however they were also causing the model to act somewhat erroneously. As such, we adjusted the value of the b coefficient slightly, reducing its effects on the system. Once finishing the tweaks to the compensator, our controller was able to keep the pendulum inverted and almost completely stationary, a result that we did not see until we had added the friction compensation to the model.

3.2 Mathematical Proof

3.2.1 Finding the Full State Equilibria

We will now show that the linearized model for this system with the full state feedback control in place is stable in the inverted position.

The system is modeled as

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

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ a & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -b_p \\ 0 \\ b_r \end{bmatrix}$$

We know $u = -Kx$, where

$$K = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix}$$

Substituting this into the original equation yields

$$\dot{x} = Ax - BKx$$

In order to find the equilibrium points, we will set $\dot{x} = 0$

$$Ax - BKx = 0$$
$$(A - BK)x = 0$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ a + b_p k_1 & b_p k_2 & b_p k_3 & b_p k_4 \\ 0 & 0 & 0 & 1 \\ -b_r k_1 & -b_r k_2 & -b_r k_3 & -b_r k_4 \end{bmatrix} x = 0$$

From this, $\boxed{x_2 = 0}$ and $\boxed{x_4 = 0}$, which implies the following

$$\begin{cases} ak_1x_1 + b_pk_1x_1 + b_pk_2x_2 + b_pk_3x_3 + b_pk_4x_4 = 0 \\ b_rk_1x_1 + b_rk_2x_2 + b_rk_3x_3 + b_rk_4x_4 = 0 \end{cases}$$

$$\begin{cases} \frac{a+b_p}{b_p}k_1x_1 + k_3x_3 = 0 \\ k_1x_1 + k_3x_3 = 0 \end{cases}$$

$$\frac{a+b_p}{b_p}k_1x_1 - k_1x_1 = 0$$

$$k_1(\frac{a+b_p}{b_p} - 1)x_1 = 0$$

Remember, the gain values we calculated for use in this system are as follows:

$$K = \begin{bmatrix} -245.3499 & -23.1081 & 0 & -0.0448 \end{bmatrix}$$

And the values for a and b_p we calculated are as follows:

$$a = 77.64, b = 0.9278$$

Recall the following equations defining the equilibria of the closed-loop system,

$$\begin{cases} k_1(\frac{a+b_p}{b_p} - 1)x_1 = 0 \\ k_1x_1 + k_3x_3 = 0 \end{cases}$$

$$-20531x_1 = 0$$

$$\boxed{x_1 = 0}$$

Additionally, $k_3 = 0$

$$\therefore \quad \boxed{\mathrm{x}_3 \in \mathbb{R}}$$

Therefore, the equilibrium points of the closed loop linearized system are:

$$x = \begin{bmatrix} 0 \\ 0 \\ \{x_3 \mid x_3 \in \mathbb{R}\} \\ 0 \end{bmatrix}$$

3.2.2 Showing Three State Stability

To check that the equilibria points found in section 2.2.1 are stable, we create a smaller 3x3 A matrix and a 3x1 B matrix as follows:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ a & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -b_p \\ b_r \end{bmatrix}$$

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

Solving this using a similar method to that used above will yield the trivial solution of

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

Therefore, $\delta\theta_p$, $\dot{\theta}_p$, and $\dot{\theta}_r$ all converge to 0 and the full state x is a stable equilibrium.

3.3 Robustness Comparisons

In order to measure the robustness of this model, we connected the components in a simulated test using SIMULINK and a modeled reaction wheel pendulum. The results of this comparison were as follows.

Table 1: Robustness Comparison of Simulated Two and Three State Controllers

	Two-State Feedback		Three-State Feedback	
Max IC Deviation	0.13 $\delta\theta_p$	1.21 $\delta\dot{\theta}_p$	0.11 $\delta\theta_p$	1.0 $\delta\dot{\theta}_p$
Max Pulse	7.95		7.83	
Max Disturbance	5.43		7.62	

3.4 System Behavior

Our finalized three-state controller with friction compensation was quite adept at controlling the pendulum and maintaining its position at the unstable equilibrium at the top of its arc. With all of the optimizations that we added, the compensation made by the reaction wheel became more finely tuned and less jerky, meaning that the pendulum arm could be held nearly stable in the inverted position without any issue. Additionally, it was able to withstand disturbances in the form of poking and prodding, as well as being set at a position that was not initially at the equilibrium. There was still some degree of motion in the pendulum arm that was introduced by the controller, likely because of an inaccurate derivative transfer function.

In order to implement the model in SIMULINK, we used a transfer function to approximate the derivatives of both $\dot{\theta}_p$ and $\dot{\theta}_r$, which may not have necessarily been accurate at all points. This error could potentially cause the slight issues that we were seeing in the output. In the next section we explored ways to circumvent this problem by using an observer.

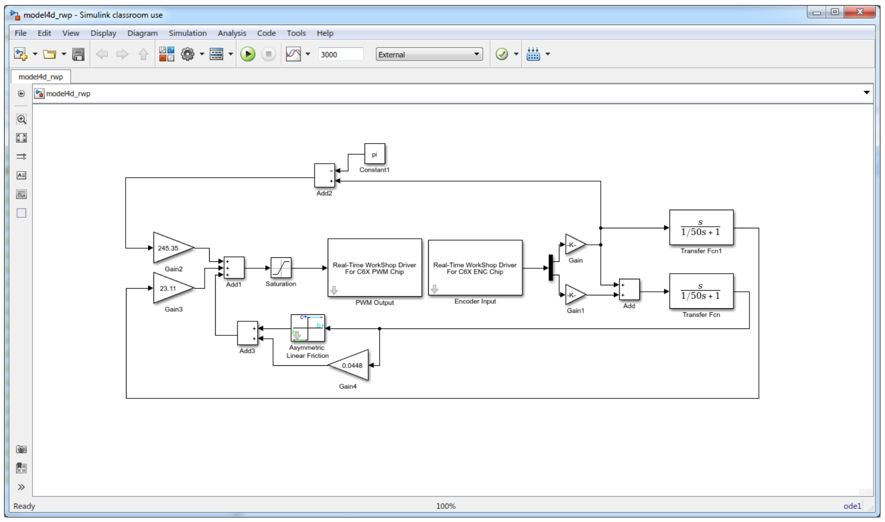
model4d_rwp - Simulink classroom use
File Edit View Display Diagram Simulation Analysis Code Tools Help
3000
External
model4d_rwp
pi
Constant1
Add2
245.35
Gain2
23.11
Gain3
Add1
Saturation
Real-Time WorkShop Driver For C6X PWM Chip
PWM Output
Real-Time WorkShop Driver For C6X ENC Chip
Encoder Input
-K-
Gain
-K-
Gain1
Add
$\frac{s}{1/50s+1}$
Transfer Fcn1
$\frac{s}{1/50s+1}$
Transfer Fcn
Add3
Asymmetric Linear Friction
0.0448
Gain4
Ready
100%
ode1

Figure 1: Three-State Model feedback controller SIMULINK diagram

4 Full State Feedback Control with Decoupled Observer

4.1 Theoretical Background

In order to improve upon our three-state model, we employed an observer that would allow us to estimate, or in other words 'observe,' each of the system states without direct observation or calculation. While we had direct access to some of these states, such as the relative positions of each of the motors, we were calculating a derivative in our earlier model that in turn might have been introducing variances into our system. In this case, we created an observer that could analyze each of these states independently and give us an estimate for the actual velocities along both axes.

In the specific case of the reaction wheel pendulum, it was possible to use a decoupled observer that would estimate each of the states separately. Mathematically, this is achieved by noticing that splitting the A matrix into 4 sections of 2x2 matrices yields both components along the block diagonal and 0 matrices along the opposite one. As such, the system is clearly split into two, where each one is dependent only one one state. This makes sense, as the derivatives of each of the states should not be affected by the current value of the other state. Thus, we can employ this decoupled observer model when building a controller for the reaction wheel pendulum. With a decoupled observer in place, we were able to achieve a more accurate and refined controller even without being able to directly measure the derivatives of either of the positional states present in the system.

4.2 Derivation

We will show that the error between the observed states and the actual states is zero

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

From the Observer states we know that:

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

$$\dot{\hat{x}} = (A - LC - BK)\hat{x} + Ly$$

In state-space matrix form, this can be represented as:

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A & -BK \\ LC & A - LC - BK \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}$$

We can perform the coordinate transfer that follows

$$\begin{bmatrix} x \\ \hat{x} \end{bmatrix} \mapsto \begin{bmatrix} x \\ e \end{bmatrix} = \begin{bmatrix} x \\ x - \hat{x} \end{bmatrix} = T \begin{bmatrix} x \\ \hat{x} \end{bmatrix} \qquad T = \begin{bmatrix} I & 0 \\ I & -I \end{bmatrix}$$

The error can be represented using these new coordinates,

$$\dot{x} = Ax + BK\hat{x}$$

$$= (A - BK)x + BK(x - \hat{x})$$

$$= (A - BK)x + BKe$$

Since T is invertible, we can rearrange the system as such

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

Recall our experimental parameters:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 77.64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

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

$$L = \begin{bmatrix} 102.8 & -1.4 \\ 2715.8 & -70 \\ -4.4 & 102.2 \\ -220.1 & 2612.3 \end{bmatrix}$$

Solving the equation above with these experimental values using Matlab's

$$linsolve(A - LC, \dot{e})$$

yielded the following stable equilibrium value

$$\boxed{\text{e} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}$$

Thus, the error of the system with the observer tends to zero.

4.3 Robustness

In order to measure the robustness of this model, we connected the components in a simulated test using SIMULINK and a modeled reaction wheel pendulum. The results of this comparison were as follows.

Table 2: Robustness Comparison of Observer Model

	Observer	
Max IC Deviation	0.03 $\delta\theta_p$	0.35 $\delta\dot{\theta}_p$
Max Pulse	5.15	
Max Disturbance	7.90	

4.4 System Behavior

When tested on the reaction wheel pendulum, the observer model was quite effective at maintaining the position of the pendulum arm at the inverted equilibrium. We found that, unlike the transfer function used in the three-state controller, the observer was far more accurate and did not introduce any extraneous motion into the system. While our simulated tests showed that the controller was not as effective against consistent pushing or deviations in initial conditions, we did not experience these problems in the final model.

Additionally, in comparison with the three-state controller, the observer model had a higher robustness in regard to pulse disturbance, which was a characteristic that was carried through to the final model as well. Because of how effective it was, we chose to use this controller to implement our swing-up controller because its motion was mimicked by the pulse disturbance of the arm being pushed. The observer model displayed firsthand the importance of having accurate estimates of each of the states in the system, and outlined how it may be possible to have a reasonably accurate estimate even if a state is not directly measured in a system.

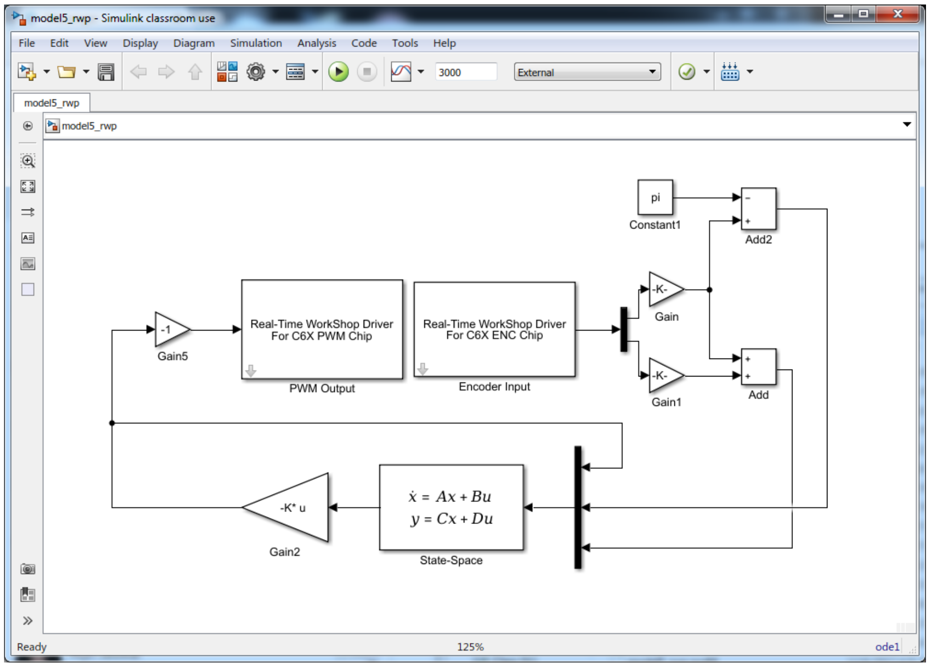
model5_rwp - Simulink classroom use
File Edit View Display Diagram Simulation Analysis Code Tools Help
3000
External
model5_rwp
pi
Constant1
Add2
-K-
Gain
-1
Gain5
Real-Time WorkShop Driver For C6X PWM Chip
PWM Output
Real-Time WorkShop Driver For C6X ENC Chip
Encoder Input
-K-
Gain1
Add
-K* u
Gain2
ẋ = Ax + Bu
y = Cx + Du
State-Space
Ready
125%
ode1

Figure 2: Observer model feedback controller SIMULINK diagram

5 Conclusions

5.1 Results in Simulation

We performed a robustness comparison between the two and three state feedback models as well as the observer model for controlling the simulated reaction wheel pendulum system. Once each controller was stable and optimized for this configuration, we tested each one for their ability to perform even under disturbance meant to model external factors. These came in three forms, a pulse continuously applied to the pendulum arm, a disturbance applied repeatedly on the pendulum arm, and finally a deviation from the equilibrium in the initial conditions.

Our findings were that while the simulated two and three state feedback were able to withstand a substantial degree of deviation in each of the tests, the observer model outperformed both in the disturbance test, which is most important for the functionality of the reaction wheel pendulum, mainly because it models the pendulum arm falling to one or the other side, forces the model to compensate for the movement. Even though the observer model did not perform as well on the other two tests, we found that implementing it in practice was quite different from the simulation.

5.2 Results on the Reaction Wheel Pendulum

When implementing each of the controllers on the reaction wheel pendulum, we found that while each of the controllers were able to stabilize and maintain the arm at the inverted equilibrium position, the observer model was better at maintaining the position of the pendulum. Unlike in the simulation, the three state feedback controller was not as robust to disturbance and even though it generally was able to balance, it did not always return to a stable position. Additionally, it would shift back and forth between a position to either side of the equilibria, meaning it was easier to set off of that position.

On the other hand, the observer model, when implemented on the reaction wheel pendulum, was much more effective at keeping the arm in the upright position without extra movement and was more resistant to being knocked over by disturbances. We were able to more easily determine the poles for this system and use it to optimize the controller even further, allowing us to find the most optimal poles for the system. With this strategy, we created the best controller possible so that in moving onto the extra credit portion of the lab, we would be able to focus on the other aspects of the controller.

6 Extra Credit

6.1 Extra Credit: Up and Down Stabilizing Control

For the first extra credit section of the report, we needed to first be able to modify the controller to be able to stabilize the pendulum arm at both of the equilibrium positions, rather than just at the inverted position. To accomplish this, we utilized a switching controller in order to change the gains and setpoint being used to stabilize the arm. This meant that we had two separate controllers in the SIMULINK model, one meant to stabilize the controller at the inverted position, and another with a completely separate set of computed gains that stabilized the arm at the downwards position. These gains performed best when we removed the $\dot{\theta}_r$ gain, which let the controller come to a complete stop at the base of the arm's swing.

The switch block in the diagram was set according to $\cos(\theta_p)$, and by specifying an angle at which the pendulum arm was considered to be nearer to the upright position, we allowed the model to switch from one controller to the other. In this way, the controller could be made to balance at the inverted equilibrium, but upon being pushed it would fall down and, under the control effort of the 'down' controller, come to rest at the stable equilibrium.

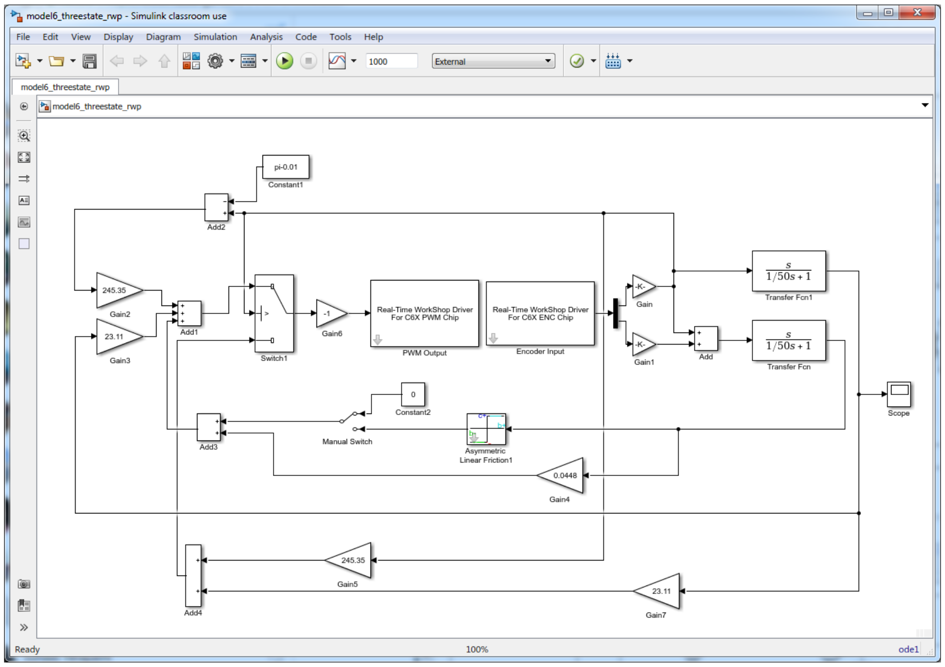

Figure 3: Switching Controller with Up/Down Stabilization SIMULINK diagram.

6.2 Extra Credit: Swing-Up Control

Based on the information that we had gathered so far through the development of this system, we believed that it would be possible to create a simple controller that would allow the pendulum to right itself after falling down through a repeated 'swing-up' motion that eventually resulted in the original controller taking over and stabilizing the arm at the top of its arc. We knew that an optimal solution to this problem would be to inject an amount of energy proportional to the arm's velocity into the system each time the arm crossed the base position of 0 degrees. With this repeated motion, the arm would theoretically move higher and higher on each swing until eventually the original controller we designed would be able to take over and stabilize it.

Before we had actually come up with a way to inject this energy into the system, we were actually attempting to solve a different problem–that of the encoder values incrementing indefinitely, preventing the controller from recognizing the equilibrium if it had gone around the wrong direction or gone past the inverted state. Essentially, the positions $0, 360, -360, 720...$ should all be considered the same from the perspective of the controller, as each is actually mapped to the same physical location of directly upwards in our linearized coordinate system. We solved this problem by introducing a *modulus* operator that would keep the original θ_p between 0 and 360, and then our linearized $\delta\theta_p$ would range between -180 and 180. This solved the problem that we were attempting to fix, as now the controller would be able to balance the arm upright no matter how many times it had been rotated or in which direction. As a side note, this method was incompatible with the observer model that we were originally using for the extra credit section, because the observer relies on the previous states of the variables and the control effort, and a discontinuity which would appear as the arm wrapped around could not be handled as the arm would no longer balance in any position. However, the three-state model with estimated derivatives at each position using a transfer function was much more effective and we continued this section using a model built from that controller.

The actual swing-up control that we used in our final model was actually a byproduct of using this modulus operator; the discontinuity was present specifically at the base equilibrium position, and it tricked the controller into thinking that there was a massive unexpected derivative jump as the arm crossed that threshold. Based on the actual speed of the arm, the response of the controller would be correspondingly high, which we noticed was exactly what we needed to implement to achieve swing-up control. As such, we modified the system to adjust an overall gain applied to the system that allowed us to exploit this behavior and swing the controller higher and higher until it reached a position reasonably close to the top at which point a set of switch blocks would transfer the system over to the original controller, and allowing the arm to stabilize on the top of its arc. While the method that we used to achieve swing-up control was not traditional or even expected, we found that it was a perfect way to use an unintended effect of a system component to achieve a useful result.

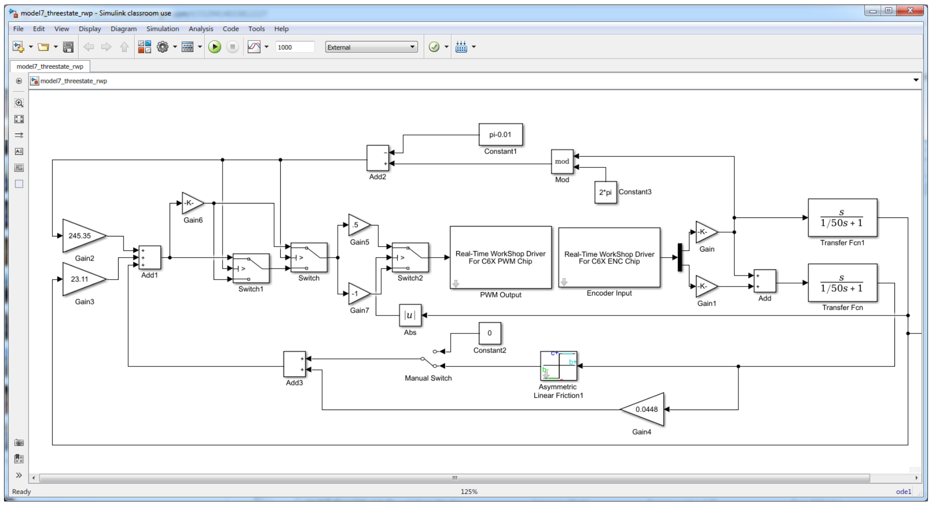

Figure 4: Swing-up Control Model with Original Three-State Controller SIMULINK diagram.