Analysis of the Cubli as a Cyber-Physical System

Joseph Adamson
Department of ECE
Baskin School of Engineering, UCSC
Santa Cruz, California
jmadamso@ucsc.edu

Conrad Esch
Department of ECE
Baskin School of Engineering, UCSC
Santa Cruz, California
cesch@ucsc.edu

Dongshuo Li
Department of ECE
Baskin School of Engineering, UCSC
Santa Cruz, California
dli40@ucsc.edu

***Abstract*—The Cubli is a cube that can self balance using reaction wheels which are controlled by an LQR controller. Treating the cubli as cyber-physical system allowed an effective analysis of the effects of sampling on the stability of the system. Simulations are presented which demonstrate that the cubli can effectively balance and can traverse a given distance. Simulations were performed using the Hybrid Equations Toolbox [2], and analyzed for stability and other properties.**

***Index Terms*—cubli, inverted pendulum, control, cyber-physical, reaction wheel**

I. Introduction [Li]

This paper's goal is to simulate a 15cm*15cm square that can achieve both the action of jumping and the action of balancing on a corner. In the paper "The Cubli: A Cube that can Jump Up and Balance" written by Mohanarajah Gajamohan, Michael Merz, Igor Thommen and Raffaello D' Andrea, their cube has three momentum wheels that can rotate at high angular velocities and then brake suddenly, causing the cubli to jump up. While the cube jumps up, the momentum wheels can then be controlled to make it balance. Our paper builds upon this work and simulates a one dimensional prototype consisting of one face of the cube containing one momentum wheel with a motor mounted in the center of the square. The simulation includes making the prototype balance and making the prototype jump for a certain number of steps and then hold itself upright.

The overall goal of this project is to build upon this work, while using the physical models presented therein as a starting point for our own formulation of the cubli as a cyber-physical system. Our simulations will focus on stabilization and controlling the prototype to 'walk' along a path by flipping over corner to corner.

II. Design Mechanics [Li]

The Cubli design applies the theory of the inverted pendulum. This model is represented by a square which pivots on its corner. A DC is mounted on the center of the square and holds a wheel for the use of inertia control. The system dynamics show the ability to control the tilt angle of the pendulum body θ_b by spinning a momentum wheel (also know as a reaction wheel) with a controlled angular velocity θ_w. The system takes advantage of conservation of angular momentum by braking the reaction wheel in order to impart an angular velocity to the pendulum body.

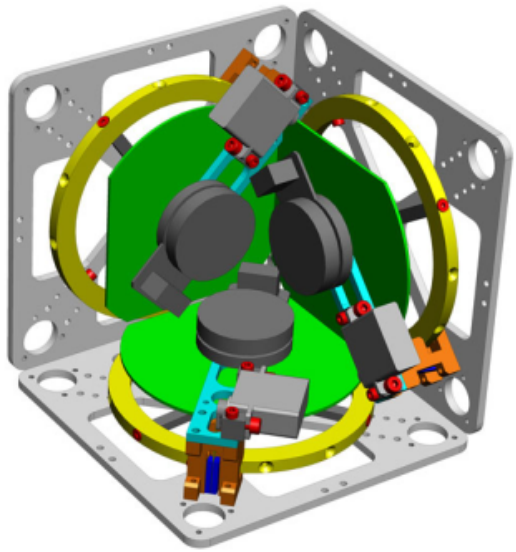

Fig. 1. An image of the inside of the Cubli showing the three reaction wheels as well as the brakes and motors. Our model represents only one side and reaction wheel from this Cubli. Referenced from [1]

D'Andrea et al. provided the following dynamical model in [1], in addition to the table below, which summarizes the parameters of their prototype. These parameters were used in our own simulations, described in section VI. The authors note that a derivation of the physical model is beyond the scope of this paper.

$$\ddot{\theta_b} = \frac{(m_b l_b + m_w l) g \sin\theta_b - T_m - C_b \dot{\theta_b} + C_w \dot{\theta_w}}{l_b + m_w l^2} \quad (1)$$

$$\ddot{\theta_w} = \frac{(I_b + I_w + m_w l^2)(T_m - C_w \dot{\theta_w})}{I_w (I_b + m_w l^2)} - \quad (2)$$

$$\frac{(m_b l_b + m_w l) g \sin\theta_b - C_b \dot{\theta_b}}{(I_b + m_w l^2)} \quad (3)$$

l	0.085 m
l_b	$0.075m$ m
m_b	0.419 kg
m_w	0.204 kg
I_b	$3.34 \times 10^{-3} \cdot kgm^2$
I_w	$0.57 \times 10^{-3} \cdot kgm^2$
C_b	$1.02 \times 10^{-3} \cdot kgm^2 \cdot s^{-1}$
C_w	$0.05 \times 10^{-3} \cdot kgm^2 \cdot s^{-1}$

[1]

A. Reaction Wheel Sensors and Control

The prototype presented in [1] uses a magnetic hall sensor-based encoder to determine the angular velocity of the torque wheel, and accelerometers to measure the body angle. Much work was done by their group to filter and fuse these measurements, with the end result being a stable estimate of the tilt angle. For our purposes, which will be simulation-based, we will assume a direct angle reading is available.

B. Braking Mechanism

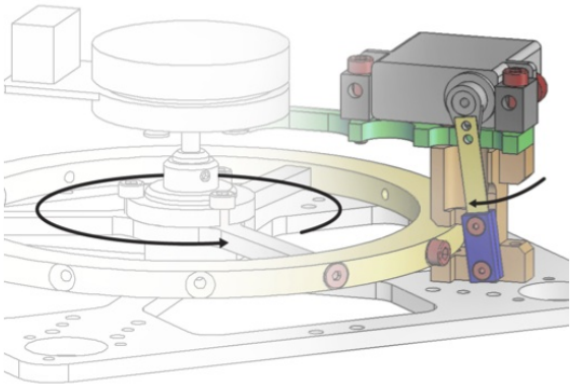

Fig. 2. CAD Drawing of braking mechanism, taken from [1]

D'Andrea et.al. swung their device up to the balancing point by incorporating a braking mechanism in their prototype, which allows a quick transfer of momentum from the torque wheel to the pendulum body. An RC servo is used to stop the wheel, which in turn quickly accelerates the pendulum body upwards. This mechanism can be seen above.

Equation 4 describes this conservation of momentum relationship:

$$I_w\omega_w = (I_w + I_b + m_w l^2)\omega_b \tag{4}$$

This action easily lends itself to modelling as a CPS, because we can consider this braking to be a system jump, with the updated value of the wheel velocity being 0, and the new body velocity calculated according to equation 4. In our simulation, we assumed the angular wheel speed(ω_w) was 10 which was enough to jump up the pendulum body. After the collision, the body is swinging up; once it reaches the equilibrium position, a balancing controller will hold the system upright on its point.

III. LQR Balancing Controller [Adamson]

Using similar methods as presented in [1], we designed the balancing controller in state space using full-state feedback, with gains determined using the Linear Quadratic Regulator (LQR) method.

State space formulation describes the system as vectors of differential equations of the form:

$$\dot{x} = Ax + Bu \tag{5}$$

where x contains all the variables (and their related derivatives) in the system and u describes the inputs available to the system. Together, they form a powerful method for describing the continuous evolution of these systems, provided they are LTI. Given that A and B are both nonchanging linear functions, state space design is a natural choice.

Full state feedback is a method of control used in state-space controller design using the control law:

$$u = -kx \tag{6}$$

where k is a row vector of the same dimension as x. This control law is well studied in control theory, and will stabilize system states to zero provided certain conditions hold; for further detail refer to section VI-C.

The gain vector K can be calculated optimally using the LQR method, which minimizes the quadratic cost function shown in 7, where Q and R are weighting matrices for state values and control effort respectively. We note that the solution to this optimization is beyond the scope of this paper.

$$\int_0^\infty x^T Q x + u^T R u \, dt \tag{7}$$

For this system, Q is a 3x3 matrix, and R is a scalar. When a component has a higher weighting, its contribution to the cost function is increased and thus the minimization process results in more aggressive control of that element. The equation is made quadratic (and thus convex) by the presence of the vector transpose multiplication, which squares and sums the components as weighted by Q and R.

Since we have not defined specific design constraints (rise time, overshoot, etc) it was sufficient to use identity matrices for the weighting. In the future, these weights can be modified to suit the design requirements. Possible properties to explore include using a higher value of R, which would cause the controller to exert less control effort (though at the cost of speed in the system.) Furthermore, the controller could be tuned to allow less deviation of the body angle if the application requires it; this would be achieved by using an identity matrix with the top left element replaced by some large number (though at the cost of control effort.)

IV. Cyber-physical Model [Adamson, Esch, Li]

The full cyber-physical model forms a sample-and-hold system, which may be seen in figure IV. A definition of each

block is given in this section. The general flow of the system is as follows:

1) The physical system, which evolves continuously according to eqn. 1, is *sampled* periodically by an Analog-to-Digital converter (ADC)
2) The cyber system uses the sampled states as its window into the physical world and performs a discrete calculation of eqn. 6. This generates a control value.
3) The resultant control value is *held* by a Digital-to-Analog converter (DAC) which periodically outputs the signal directly to the physics

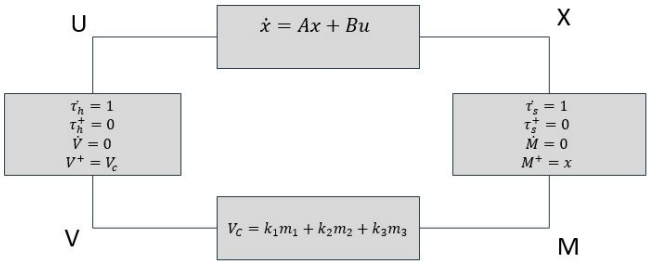

Fig. 3. Full CPS Block Diagram demonstrating the interconnections. U and X are continuous quantites, while V and M are the discretely sampled versions

A. Physical Model

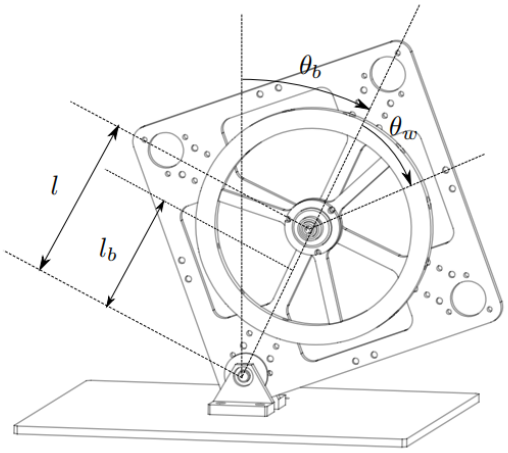

Fig. 4. Graphic illustrating the relationship between parameters and overview of cubli prototype [1]

The formulation of the physical model given in [1] was used as a starting point for the modelling of this system. The system state contains the three parameters:

θ_b : Angle of the pendulum body
$\dot{\theta_b}$: Angular velocity of the pendulum body
$\dot{\theta_w}$: Angular velocity of the torque wheel

We will define the physical state vector:
$x := [\theta_b, \dot{\theta_b}, \dot{\theta_w}]^T$

The body angle is considered to be 0 when the body is balanced upright, with clockwise movement being positive.

1) Linearization of Continuous System: Following one standard approach to nonlinear control, a linearized model was used, which locally approximates the dynamics given in eqn. (1) using a generalized tangent plane, with a gradient determined by taking the jacobian of the function evaluated at the point we want to linearize about. We are using the linearized model given by D'Andrea et. al. which is linearized about the point (θ_b, $\dot{\theta_b}$, $\dot{\theta_w}$) = (0,0,0). The resultant matrices for the state space model in eqn. 7 captures the physical dynamics as follows:

$$A := \begin{bmatrix} 0 & 1 & 0 \\ \frac{m_b l_b + m_w l) g}{I_b + m_w l^2} & -\frac{C_b}{I_b + m_w l^2} & \frac{C_w}{I_b + m_w l^2} \\ -\frac{m_b l_b + m_w l) g}{I_b + m_w l^2} & \frac{C_b}{I_b + m_w l^2} & -\frac{C_w (I_b + I_w + m_w l^2)}{I_w (I_b + m_w l^2)} \end{bmatrix}$$

$$B := \begin{bmatrix} 0 \\ -\frac{K_m}{I_b + m_w l^2} \\ .\frac{K_m (I_b + I_w + m_w l^2)}{I_w (I_b + m_w l^2)} \end{bmatrix}$$

Again, this model was provided from [1], though verifiable by linearizing eqns. 1. We note that the system has only one input u, which makes sense given that B is a 3x1 column vector; the input, which is taken to be motor current, spins the torque wheel, which causes the body to spin in the opposite direction, as indicated by the sign change from the second to third elements.

Using the MATLAB utility ctrb() with this system reveals that the controllability matrix is full-rank. Control theory learned during other coursework tells us that this condition indicates the linearized system will be controllable via the methods discussed in section III.

2) Flow Conditions: The system is allowed to flow as long as the interface timers, which are defined in the following subsection, have not expired,

B. Cyber Model

1) Interface : The interface system consists of both the ADC and DAC portions, which function as input/output for the cyber controller. These methods are taken directly from coursework.

Two timer variables are used to keep track of the sampled input and held output sample times, respectively:

$\tau_s :=$ ADC timer
$\tau_h :=$ DAC timer

Three memory variables are used to store the sampled system state, and one memory variable is used to store the

output value held by the DAC:

m_1, m_2, m_3 := Sampled system state.
v := DAC output

The timer variable dynamics are given by $\dot{\tau} = 1$ for both, and the memory variable dynamics are such that they do not change during flows (derivatives are zero) and allowed to flow as long as they are below the sample time parameters T_s^* and T_H^*. As such, these variables track time.

2) Jump Conditions: When the timer variables reach their limiting values, the jump map resets them to zero:

$\tau^+ = 0$

If the ADC timer is expired, then the memory variables are updated with a discrete snapshot of the continuous state x:
$\mathbf{m}^+ = \mathbf{x}$

If the DAC timer is expired, then the output variable is updated with the new control law calculation for that time step. Note that the gains are applied to the sampled state m, not the continous state x:
$v^+ = -K * \mathbf{m}^T$

This last line is critical; it is the portion of the jump map which sets v^+, which the DAC applies to the physical model as the input u. This is where the full-state feedback controller is applied to the system! The controller multiplies the gains K by the sampled states M, which according to the system dimensionality, produces a scalar control value.

The gain vector K was calculated using the LQR() function in MATLAB, which automatically performs the minimizations described in section III given the matrices A,B,Q, and R.

C. Jump Model [Esch]

As stated above our this system is modelled as an inverted pendulum. When θ_b arrives at either $\frac{\pi}{4}$ or $\frac{-\pi}{4}$ the frame of the Cubli is resting entirely on its side and cannot fall any further. The simulation parameters were chosen to handle this situation in a similar fashion to a bouncing ball during the moment of impact. When our square is flat against the ground, the velocity is set to 0 and an acceleration is added to counteract gravity. This prevents it from falling any further in the simulation. One of the key elements of the Cubli is the ability for it to hop itself upright or entirely flip itself over. This is accomplished by spinning up its reaction wheel in the direction it would like to flip. The brakes are then applied, transferring the momentum to the square itself. In the model, it was assumed that the square automatically jumps when it encounters the ground and that there is no delay between wheel spinnup and the next jump. This was much simpler to model than than the alternative and

1) Hopping Model: Wheel velocity was chosen to be 10 due to this easily being enough velocity to flip the Cubli and not too much to prevent the controller from keeping the cube upright if it was turned on. The defining system for this jump is shown below.

$$x_j^+ = \begin{bmatrix} -x_1 + 0.001 \\ 10 \\ 0 \end{bmatrix} \quad when \quad x_1 = (\frac{\pi}{4} \vee x = -\frac{\pi}{4})$$

In the above model θ_b is reversed to correspond to flipping around a new corner. It is also offset slight to overcome the issue in our simulation in which it would never leave the jump state due to the square always being at an angle of $\frac{\pi}{4}$. The velocity of the wheel, $\dot{\theta_b}$ is set to 10 to account for the rotational inertia of the wheel suddenly being halted and being transferred to the cube. Choosing values much lower than this does not impart enough momentum to flip the cube around. As a result of the braking, the rotational speed of the wheel, $\dot{\theta}_w$ is set to 0.

D. Full Cyber-Physical Model

We define the hybrid state, which comprises of all variables presented in this section:

$$z = [x^T \tau_s m_1 m_2 m_3 \tau_h v]^T \tag{8}$$

All analysis presented herein is performed with respect to the evolution of the hybrid state z.

Consolidating the various components of the flow and jump behaviors yields the complete cyber-physical model, which may be found in Appendix A.

V. Simulation Setup [Adamson, Esch, Li]

A. Assumptions

In order to simulate the cubli rolling along a path, our simulation makes a few assumptions for simplification:

- Once the cube is ready to make another jump, the torque wheel can spin up to speed immediately. In reality, the system will need to spend some time in a resting position while the wheel spins up.
- The computation of the control law takes much less time than the sampling period
- The linearized model is valid in the ranges of angle we are analyzing

B. Hybrid Equations Toolbox

The Hybrid Equations toolbox makes it relatively easy to simulate a cyber-physical system given F,G,C, and D. Following the examples that come with the toolbox, the simulation was set up using all relevant information. Challenges during simulation included restricting the cube to $[-\pi/4, \pi/4]$ and correctly extracting the path data from the results. Since the linearized model relies upon working with values close to zero, we needed some way to keep the values in this range. The solution, as outlined in section IV-C, was to 'wrap

around' the states by letting the body angle jump from $\pi/4$ to $-\pi/4$ as soon as one full turn was completed (assuming a rightward/counterclockwise direction of travel.)

VI. Simulation Results [Adamson, Esch, Li]

A. Balancing on Corner

The first simulation shows the balancing controller, which makes the one-dimensional model balance on its tip. The controller, which is the cyber portion of the system, immediately drives the body angle to 0. The simulation is performed with an initial body angle of $-\frac{\pi}{5}$. The states are expected to converge to (0,0,0), with the control effort spiking as the wheel tries to stabilize the system. The wheel speed should increase as the control effort increases, with the body speed reacting oppositely to the torque wheel.

Our simulations confirmed this, with results shown below. The sampling rate for both interfaces was set to 30ms for these simulations. Interestingly, it can be seen that the system starts by falling down at about .025 seconds before the controller picks it back up and kicks it upright. The resultant spike in wheel velocity, caused by the spike in control effort, shows the system stabilization process. The simulation results can be seen below in figure 5

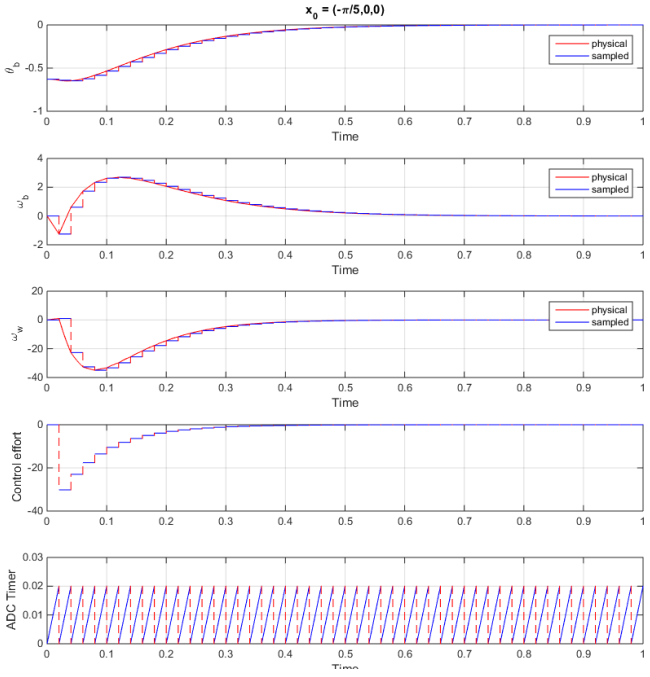

Fig. 5. Balancing controller with nonzero initial conditions, demonstrating stability. ADC timer provided for sampling reference

B. Jumping

The goal of the second simulation is to make the one dimensional prototype take four "steps" and then balance on the corner. While the prototype tries to finish the steps, the body angle increase from $\frac{-\pi}{4} to \frac{\pi}{4}$. When the body angles reaches $\frac{\pi}{4}$, it jumps to $-\frac{\pi}{4}$ because the pivot point changes. During the steps, the balancing controller is not active until the steps are finished; once the square has reached its destination, the balancing regulator is activated to keep it upright on the corner. The body speed behaves cyclically, as the square swings upwards (losing speed as potential energy) and falls back down the other side (releasing the energy). Interestingly, the torque wheel speed can also be observed fluctuating as it reacts passively to the spinning of the body.

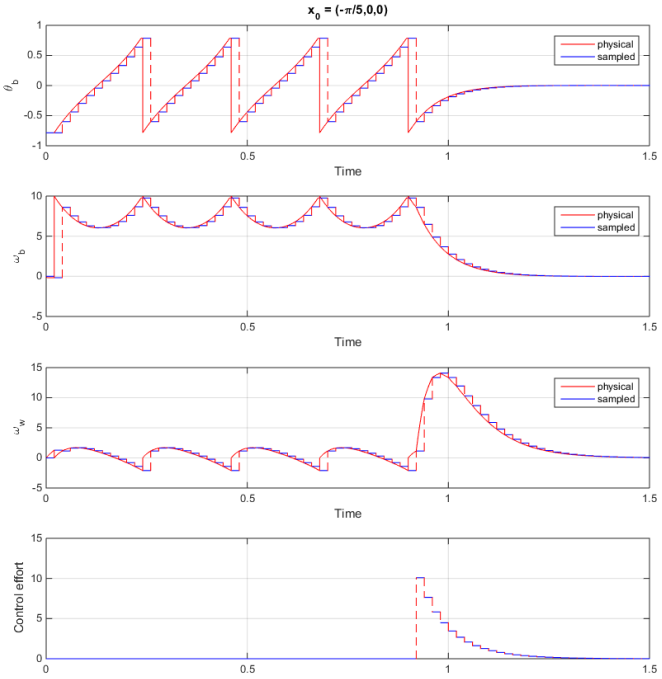

Fig. 6. System performing 4 jumps and ending balanced on point

C. Stability Analysis

We can see from the plots that the balancing controller is able to stabilize the system about the physical state $x = [0, 0, 0]$ in finite time.

To understand why, we can rearrange and substitute between eqns. (5) and (6) to reach the closed-loop dynamical system:

$$\dot{x} = (A - B * K) * x \tag{9}$$

For our simulated system, the closed loop matrix is:

$$\begin{bmatrix} 0 & 1 & 0 \\ -106.2 & 20.7 & 0 \\ 1842 & 1935 & 20.7 \end{bmatrix}$$

with eigenvalues: $[-21.6074, -10.1850, -9.6108]$

The presence of purely negative eigenvalues means that the closed loop system will converge to zero in finite time, thus rendering the set (0,0,0) stable. Furthermore, points around this region are shown by our simulations to converge to (0,0,0) and thus demonstrate attractivity.

By contrast, the eigenvalues of the open-loop system A are $[-20.3832, -10.0650, 9.8566]$, which contains a positive value and thus indicates instability.

The digital implementation of this controller does not affect stability, as the sampling rates used appear to be fast enough that the unstable pole doesn't have enough time to go unstable without the cyber controller noticing and responding with corrective control effort.

VII. Conclusion[Li]

During the completion of this project we were able to implement a cyber physical control scheme to stabilize a one-dimensional prototype of the Cubli. Starting from the work presented in [1], we were able to use the complex physical model in a Sample-and-Hold scheme to stabilize it.

The system has features related to both continuous systems and discrete systems and was therefore a good candidate to apply CPS techniques.

Using these techniques, we derived a full system modeland simulated the control algorithms with the aid of the Hybrid Equations toolbox. Once we determined the full model, entering the sets C and D, and the maps F and G allowed us to produce the results we have presented.

References

[1] Gajamohan, M., Merz, M., Thommen, I., Dandrea, R. (2012). The Cubli: A cube that can jump up and balance. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. doi:10.1109/iros.2012.6385896

[2] Sanfelice, R., Copp. D, Nanez, P. (2014), Hybrid Equations (HyEQ) Toolbox V2.02

APPENDIX

A. Total Cyberphysical Model

Physical State:

$$x = [\theta_b, \dot{\theta}_b, \dot{\theta}_w]^T$$

Hybrid State:

$$z = [x^T, \tau_s, m_1, m_2, m_3, \tau_h, v]^T$$

Flow Map F:

$$\dot{z} = [F(x,u)^T, 1, 0, 0, 0, 1, 0]^T \qquad z \epsilon C$$

Flow Set C:

$$\{z : (\tau_s < T_s^*) \wedge (\tau_h < T_h^*) \wedge (-\frac{\pi}{4} < x_1 < \frac{\pi}{4})\}$$

Jump Map G:

$$z^+ = [x^T, 0, x_1, x_2, x_3, \tau_h, v]^T \qquad \tau_s = T_S^*$$
$$z^+ = [x^T, \tau_s, m_1, m_2, m_3, 0, -k * m]^T \qquad \tau_h = T_h^*$$
$$z^+ = [-\theta_b, 10, 0, \tau_s, m_1, m_2, m_3, \tau_h, v]^T \qquad \theta_b = \pm\frac{\pi}{4}$$

Jump Set D:

$$\{z : (\tau_s = T_s^*) \vee (\tau_h = T_h^*) \vee (x_1 = \pm\frac{\pi}{4})\}$$

Where:

θ_b = Angle of the pendulum body

$\dot{\theta_b}$ = Angular velocity of the pendulum body

$\dot{\theta_w}$ = Angular velocity of the torque wheel

$F(x,u) = \dot{x} = Ax + Bu$

T_s^* = ADC Sample Period = 20ms

T_h^* = DAC Sample Period = 20ms

m_i = ADC Memory Variables

v = DAC Output

$$A := \begin{bmatrix} 0 & 1 & 0 \\ \frac{m_b l_b + m_w l) g}{I_b + m_w l^2} & -\frac{C_b}{I_b + m_w l^2} & \frac{C_w}{I_b + m_w l^2} \\ -\frac{m_b l_b + m_w l) g}{I_b + m_w l^2} & \frac{C_b}{I_b + m_w l^2} & -\frac{C_w (I_b + I_w + m_w l^2)}{I_w (I_b + m_w l^2)} \end{bmatrix}$$

$$B := \begin{bmatrix} 0 \\ -\frac{K_m}{I_b + m_w l^2} \\ \cdot\frac{K_m (I_b + I_w + m_w l^2)}{I_w (I_b + m_w l^2)} \end{bmatrix}$$