RBE 502 Final - Cart-Pole Controller Evaluation

Paul Crann

I. Introduction

The inverted pendulum on cart (or cart-pole) is a standard dynamical system used to study control systems. The goal of the system is to control an input force on the cart in a way that both moves the cart to a desired position, and keeps the pendulum in the upright position. This system is challenging to control due to it being under-actuated and naturally unstable in the upright position. The analysis of this system has real world applications, such as in Segway like balancing robots.

My intent in this project is to simulate multiple control laws for a variation of the pendulum on cart system in the presence of process and sensor noise.

The control algorithms selected will be two Linear Quadratic Regulators (LQRs), and a two stage PD controller. The algorithms will be examined with a Kalman Filter.

II. Related literature

Due to the cart-pole system being of sorts a baseline for designing control systems, it appears very frequently in the literature. Two examples will be highlighted here. The first being 'Control Design and Implementation of an Inverted Pendulum on a Cart' [1], where a PID controller is implemented on a real-world cart-pole system to stabilize about the up words position. However, this work does not include consideration for controlling the position of the base of the cart, only the pendulum angle. A second piece of literature that uses the cart-pole system is 'Swing-up Control of a Single Inverted Pendulum on a Cart With Input and Output Constraints' [2], where a trajectory was created to swing the pendulum up from its downwards position. A feed forward controller was implemented to allow the system to follow the designated trajectory. However, this work only implemented this system in a perfect simulation assuming no sensor or porcess noise.

III. Simulation Techniques

A. System Overview

The system used contains 3 masses and is shown below in figure 1. It consists of a base (A) with mass m_a, a pole (B) of length $2 * l_b$ with mass m_b and moment of inertia I_b, and a point mass (C) of length $2 * l_b$ of mass m_c and moment of inertia I_c.

1) Model Parameters: Throughout this work, the system parameters used will be defined by table I.

2) Assumptions: As with any simulation, assumptions must be made. In this work, the following were chosen to allow realistic results with reasonable complexity.

- Input Force: The model will ignore motor dynamics, and reduce the motor torque/wheel radius relationship to a force F or U applied to the base of the system. This input force will be limited to a minimum and maximum force as defined by F_{range} in table I.
- Friction: Only kinetic friction will be modeled, and will be included for both degrees of freedom. From table I, f_a defines the translational friction related to $\dot{x}$, and f_b defines the rotational friction related to $\dot{\theta}$.
- Noise: The robot will be capable of measuring zero mean Gaussian noise readings of x and θ. This noise will be defined by the noise covariance matrix $\boldsymbol{Q}$. Process noise will also be modeled by zero mean Gaussian noise, as defined by $\boldsymbol{R}$.

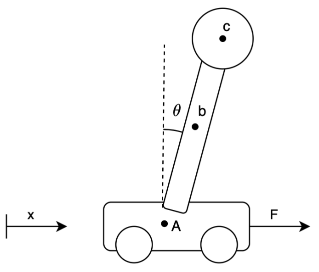

Fig. 1. System

Parameter	Value	Units
m_a	1.0	kg
m_b	0.2	kg
m_c	0.5	kg
l_b	0.2	m
l_c	0.4	m
I_b	0.0107	km^2
I_c	0.08	km^2
f_a	0.01	$N/\frac{m}{s}$
f_b	0.01	$N/\frac{rad}{s}$
g	9.81	m/s^2
F_{range}	[-25, 25]	N

TABLE I
System Parameters

$$\boldsymbol{Q} = \begin{bmatrix} (1mm)^2 & 0 \\ 0 & (0.1deg))^2 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & (1\frac{mm}{s})^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (0.1\frac{deg}{s}))^2 \end{bmatrix}$$

B. Equations of Motion

To obtain the dynamical model of the system, the Lagrange method was used. By modeling the translational kinetic energy of all three masses, and the rotational, kinetic, and potential energy of masses B and C, the Lagrangian can be formulated and used in the Euler-Lagrange equation as shown in Appendix

A. The generalized forces used in this equation are derived from the input force F, the friction force acting on the cart $f_a\dot{x}$, and the friction force on the pole $f_b\dot{\theta}$

The state variables chosen are as follows: $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$

It should also be noted that throughout the paper, constants M, a, and b are used to clean up equations. They are defined in appendix A.

1) Non-Linear Model: The Euler-Lagrange equations yield two nonlinear differential equations describing the linear accelerations along the x axis and angular acceleration about θ, which correspond with equations (2) and (4) below. Equations (1) and (3) are given by definition.

$$\dot{x}_1 = x_2 \tag{1}$$

$$\dot{x}_2 = \frac{-absin(x_3)x_4^2 - bu + bf_a x_2}{a^2cos(x_3)^2 - Mb} + \frac{bcos(x_3)(agsin(x_3) - f_b x_4 + ax_2x_4sin(x_3))}{a^2cos(x_3)^2 - Mb} \tag{2}$$

$$\dot{x}_3 = x_4 \tag{3}$$

$$\dot{x}_4 = \frac{-aMgsin(x_3) + Mf_b x_4 - aMx_2x_4sin(x_3)}{a^2cos(x_3)^2 - Mb} + \frac{acos(x_3)(asin(x_3)x_4^2 + u - f_a x_2)}{a^2cos(x_3)^2 - Mb} \tag{4}$$

These four equations yield the nonlinear state space functions to be used in the simulation.

2) Linearized Model: For some control methods, as well as testing observability and controllability, a linearized model comes in handy. Linearizing about the upwards stationary equilibrium where $\theta = 0$ and $\dot{\theta} = 0$ is chosen. Converting to the state space, we develop the symbolic equations found in Appendix B. Substituting our chosen parameter values from table 1 we obtain the following state space system:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.0071 & -1.6613 & 0.0071 \\ 0 & 0 & 0 & 1 \\ 0 & 0.0084 & 13.9349 & -0.0592 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.7063 \\ 0 \\ 0.8364 \end{bmatrix} * u \tag{5}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + 0 \tag{6}$$

C. Simulator

The simulation used in this work is a custom matlab function. This function steps through time at a given step size (dt). At each time step, the truth position will be updated according to the non-linear dynamics and process noise, a measurement will be taken according to the current state and sensor noise, the state will be estimated, and a control input for the next step will be computed according to the current controller.

IV. Controller Techniques

A. Controllability and Observability

Before designing specific control laws, I check the observability and controllability of the system to ensure that our system parameters and sensor architecture allows for theoretical control.

1) Observability: The Observability of the system can be checked while linearized about the upwards position by ensuring that the rank of the observability matrix is equal to the number of state variables (n=4).

$$O = [C; CA; CA^2; CA^3]$$

$$O = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -0.0071 & -1.6613 & 0.0071 \\ 0 & 0.0084 & 13.9349 & -0.0592 \\ 0 & 0.0001 & 0.1102 & -1.6617 \\ 0 & -0.0006 & -0.8395 & 13.9385 \end{bmatrix}$$

$$rank(O) = 4 = n$$

Therfore, the system is observable.

2) Controllability: Likewise to observability, the controllability can be checked by ensuring that the rank of the controllability matrix is equal to n.

$$C = [B, AB, A^2B, A^3B]$$

$$C = \begin{bmatrix} 0 & -0.7063 & 0.0109 & -1.3900 \\ -0.7063 & 0.0109 & -1.3900 & 0.1843 \\ 0 & 0.8364 & -0.0555 & 11.6590 \\ 0.8364 & -0.0555 & 11.6590 & -1.4753 \end{bmatrix}$$

$$rank(C) = 4 = n$$

Therefore, the system is controllable.

B. Kalman Filter

In the interest of keeping this simulation realistic, my system assumes both process and sensor noise. Due to the sensor noise, calculating control inputs based on raw sensor measurements will lead to sub optimal inputs and an unstable closed loop system. The solution to this problem, as it often is in the real-world, is using a Kalman Filter. This technique will allow our system to leverage its knowledge of the systems dynamics, sensor measurements, and covariance of both to find a near optimal solution to the state estimation problem. The solution will not be exactly optimal due to the linearization that must take place in order to update the KF. This filter is adapted from the ones described in [3].

The Kalman Filter algorithm incorporated here consist of the following steps:

1) Predict the current state using the past estimation, control input, and dynamical system knowledge
2) Update the covariance matrix P
3) Compute the Kalman gain
4) Compute the new estimate using the prediction, Kalman gain, and measurement

This new state estimate will be more accurate than the sensor measurements themselves, and thus a better input for our control system calculations.

C. *Linear Quadratic Regulator*

A commonly used method of optimal control for linear systems is the Linear Quadratic Regulator (LQR). This method produces the optimal state feedback matrix, K, in order to minimize a cost function. My system is approximately linear near the region of interest, so a linear controller can be designed based on the linearized system and then deployed in the nonlinear simulation.

By changing the cost function, we can tailor the response to different goals. In this work, two different LQR's will be created. The first with the goal of moving to the set point as quick as possible, or minimizing the error of x. The second will move to the set point with minimal deviation of θ, or minimizing the error of x_3.

The LQR gain K is calculated using MATLABS built in lqr() function [4]. This function will compute K in order to minimize the cost function J (7). For these two controllers, we will differ Q while keeping R = 1 and N = 0.

$$J = \int_0^\infty (x^T Q x + u^T R u + 2x^T N u)dt \tag{7}$$

1) *Minimizing x error:*

$$Q1 = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R = 1, N = 0 \tag{8}$$

2) *Minimizing θ error:*

$$Q2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R = 1, N = 0 \tag{9}$$

D. *2 Stage PD Controller*

A 2 stage PD controller will also be implemented on this system. Due to the nature of the dynamical system (single input multiple output), a singular PD controller trying to drive all system states to the desired values at the same time will not produce an attractive controller. In summary, the system can only move towards the desired x location while remaining stable by first moving in the opposite direction. My solution to this is a high level PD controller providing a reference angle to the lower level PD controller.

The first stage PD controller will computed a desired $\theta_{desired}$ angle based on errors in x and $\dot{x}$ and then saturate this value between [-3, 3] deg. The second stage will use PD control to drive θ to $\theta_{desired}$. Both stages are described below.

Stage 1:

$$\theta_{desired} = P_1 * (x_{desired} - x) - D_1 * \dot{x} \tag{10}$$

State 2:

$$u = -P_2 * (\theta_{desired} - \theta) + D_2 * \dot{\theta} \tag{11}$$

The values chosen after tuning for this controller are as follow: P1 = 10 D1 = 30 P2 = 65 D2 = 45

V. Results and Discussion

The performance of each controller will be evaluated through a Monte Carlo simulation based on the following criteria: The percentage of runs that reach the desired position (Percent Successful), The average time to settle withing 5% of the desired position (5% Settling Time), the maximum x overshoot, maximum $\dot{x}$, maximum θ, and maximum $\dot{\theta}$ values.

The Monte Carlo analysis consists of 100 runs for each controller, with a starting state of $x_0 = [0; 0; 0; 0]$ and final state of $x_f = [5; 0; 0; 0]$. In other words, the systems starts in an upright and stationary position, and must move 5m to the right, where it will then again become stationary.

First, a simulation of each controller will be shown below. The highlighted data consists of x, $\dot{x}$, θ, $\dot{\theta}$, and u vs time plots.

A. *LQR1 Example Sim*

Figure 2 shows the x and $\dot{x}$ vs time response. As evident, the system moves to position $x = 5$ within 2 seconds, where it then settles with minimal deviation from its set point. The systems linear velocity ($\dot{x}$) peaks at a maximum of 5.5 m/s.

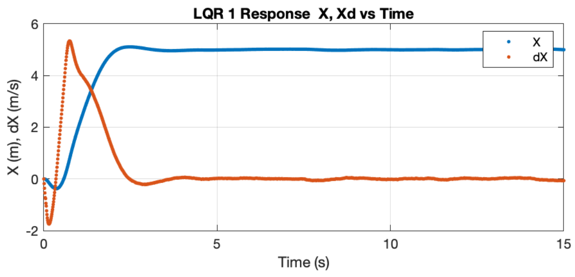

Fig. 2. LQR 1 X, Xd Response

Figure 3 shows the θ and $\dot{\theta}$ vs time response. As desired, both states converge on 0 deg or deg/s. The systems angle (θ) peaks at around 40 deg, while the angular velocity($\dot{\theta}$) peaks at 140 deg/s.

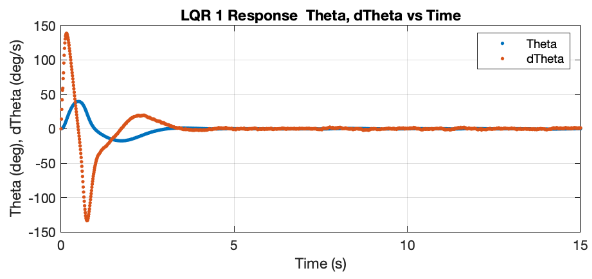

Fig. 3. LQR 1 Theta, Thetad Response

Figure 4 shows the input force U vs time. With this controller, the input force saturates at +- 25 N during the most aggressive part of the maneuver.

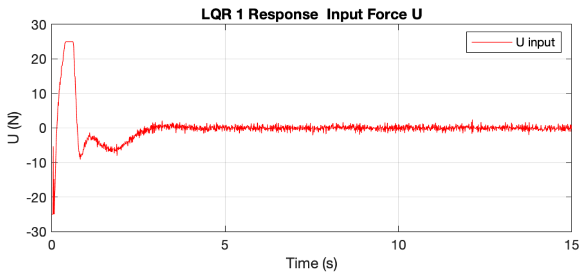

Fig. 4. LQR 1 U Response

B. LQR2 Example Sim

Figure 5 shows the x and $\dot{x}$ vs time response. As evident, the system moves to near position $x = 5$ within 5 seconds, where it then settles with minimal deviation from its set point. The systems linear velocity ($\dot{x}$) peaks at a maximum of 1.5 m/s.

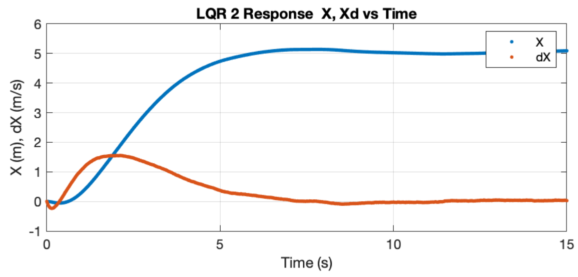

Fig. 5. LQR 2 X, Xd Response

Figure 6 shows the θ and $\dot{\theta}$ vs time response. As desired, both states converge on 0 deg or deg/s. The systems angle (θ) peaks at around 7 deg, while the angular velocity($\dot{\theta}$) peaks at 20 deg/s.

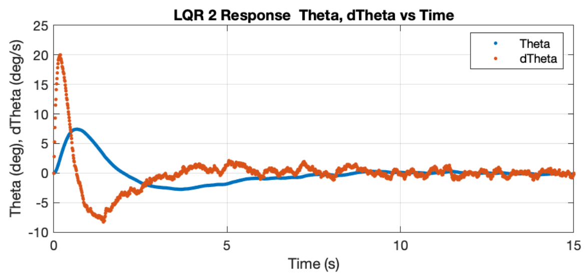

Fig. 6. LQR 2 Theta, ThetaD Response

Figure 7 shows the input force U vs time. With this controller, the maximum input force demanded is -5 N, far from its saturation limits.

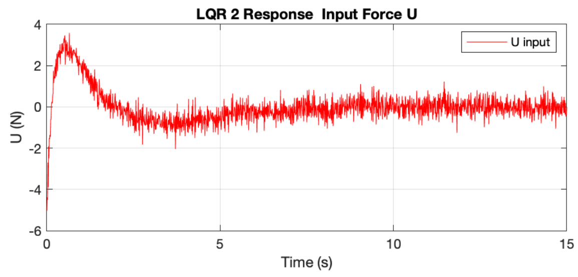

Fig. 7. LQR 2 U Response

C. 2 Stage PD Example Sim

Figure 8 shows the x and $\dot{x}$ vs time response. As evident, the system moves to near position $x = 5$ within 6 seconds, but takes much longer to converge at exactly 5m. The systems linear velocity ($\dot{x}$) peaks at a maximum of 1.5 m/s, and oscillates in its steady state.

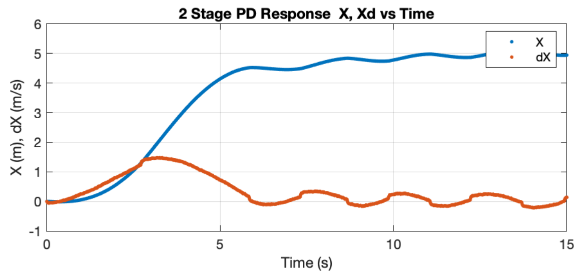

Fig. 8. 2 Stage PD X, Xd Response

Figure 9 shows the θ and $\dot{\theta}$ vs time response. Undesirably, neither states converge on 0 deg or deg/s. The systems angle (θ) peaks at around 4 deg, while the angular velocity($\dot{\theta}$) peaks at 7 deg/s.

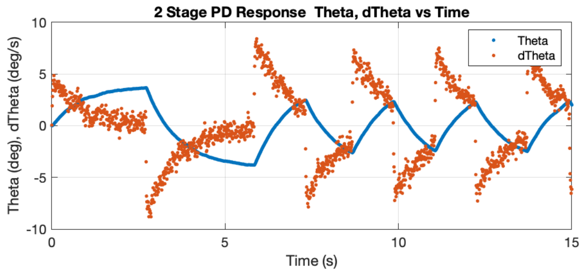

Fig. 9. 2 Stage PD Theta, Thetad Response

Figure 10 shows the input force U vs time. With this controller, the input force follows a different pattern, where its peak does not occur in the initial portion of the simulation. Additionally, it avoids saturation by a wide margin.

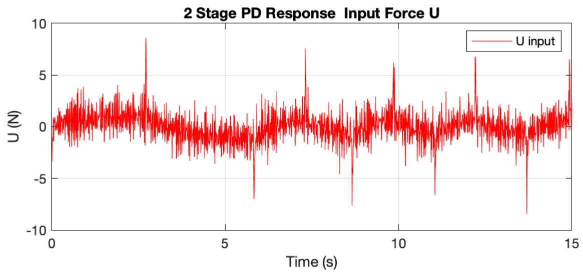

Fig. 10. 2 Stage PD U Response

D. Kalman Filter Effectiveness

As evident by table II below, the addition of the Kalman Filter drastically reduces the RMS error values for states $\dot{x}$ and $\dot{\theta}$, making them suitable for controller usage.

state	Raw	KF
x (m)	0.001	0.010
$\dot{x}$ (m/s)	0.134	0.017
θ (deg)	0.098	0.582
$\dot{\theta}$ (deg/s)	14.18	1.260

TABLE II
RMS STATE ERRORS

E. Monte-Carlo Results

Controller type	LQR 1	LQR 2	2 Stage PD
% Success	100	100	100
5% Settling Time (s)	1.89	5.17	8.11
x Overshoot (mm)	94.1	84.1	66.3
Max $\dot{x}$ (m/s)	5.36	1.55	1.55
Max θ (deg)	39.62	7.37	3.83
Max $\dot{\theta}$ (deg/s)	141	19.6	9.54

TABLE III
MONTE-CARLO RESULTS

Using the evidence from table III, similarities and differences between the three controllers can be concluded. First and foremost, all three controllers produced a reliably stable closed loop system with 100% accuracy in these trials.

Comparing the two LQR controllers, LQR 1 produces a much quicker settling time (1.89 s vs 5.17 s) at the expense of a larger x overshoot (94.1 mm vs 84.1 mm), maximum $\dot{x}$ (5.36 m/s vs 1.44 m/s), most notably maximum θ (39.62 deg vs 7.37 deg), and maximum $\dot{\theta}$ (141 deg/s vs 19.6 deg/s). This comes at no surprise, as the first LQR's values (defined by equation 8) placed a much larger emphasis on minimizing the x distance error at the expense of the other state variables. While the second LQR's values (defined by equatiion 9) prioritized a minimized θ deviation from 0, thus reducing the speed at which the system could move to the desired x position.

The two stage PD controller had 100% of runs settle within 5% of its goal in the allotted time (15 s), with the average in 8.11s. This is much larger than LQR 1 and LQR 2. This controller shows an advantage over the other two when it comes to maximum θ and $\dot{\theta}$ values (3.83 deg, 9.54 deg/s), which are much smaller than even LQR 2. This is due to the controller calculating a desired θ value, and then saturating it withing a given range. Therefore, and unlike the LQR's, the control input will not continually grow based on a large x error.

VI. CONCLUSION

In this work, three controllers were designed and implemented on a cart pole variation in a custom simulation. The results highlighted the strengths of each controller. LQR 1 provided a very fast response time. LQR 2 provided a medium response time with much lower θ and $\dot{\theta}$ errors. And the 2 Stage PD Controller provided a response with minimal jerk and very small θ and $\dot{\theta}$ errors. This analysis also highlighted the weakness of each controller. LQR 1 had the highest overshoot, a saturated input force, and large θ and $\dot{\theta}$ errors. LQR 2 had some overshoot along with a shorter settling time. And the 2 Stage PD Controller had a very slow settling time as well as small oscillations in its steady state.

There are many directions that future work could build off of this project. To name a few, these controllers could be tested in different tasks than set point control (ie. trajectory following, disturbance rejection), the 2 stage PD controller could be further tuned to reduce the steady state oscillations that occur, and the system could become one of higher fidelity by reducing some of the assumptions made (namely the wheel/motor simplifications).

This project taught me a deeper understanding of implementing control systems for under actuated robots, designing LQR's, and how to implement a Kalman Filter for state estimation.

VII. REFERENCES

[1] M. Owais, A. Ul-Haque, H. A. Rahim, S. Aftab and A. A. Jalal, "Control Design and Implementation of an Inverted Pendulum on a Cart," 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Kuala Lumpur, Malaysia, 2019, pp. 1-6, doi: 10.1109/ICETAS48360.2019.9117388.

[2] M. Tum, G. Gyeong, J. H. Park and Y. S. Lee, "Swing-up control of a single inverted pendulum on a cart with input and output constraints," 2014 11th International Conference on Informatics in Control, Automation and

Robotics (ICINCO), Vienna, Austria, 2014, pp. 475-482, doi: 10.5220/0005018604750482.

[3] J. Bongard, ”Probabilistic Robotics. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. (2005, MIT Press.) 647 pages,” in Artificial Life, vol. 14, no. 2, pp. 227-229, April 2008, doi: 10.1162/artl.2008.14.2.227.

[4] “Linear-Quadratic Regulator (LQR) design - MATLAB lqr,” www.mathworks.com. https://www.mathworks.com/help/control/ref/lti.lqr.html

Appendix A
Lagrange Summary

$$M = m_a + m_b + m_c$$

$$a = m_b l_b + m_c l_c$$

$$b = m_b l_b^2 + m c l_c^2 + I_b + I_c$$

$$L = \frac{1}{2} M \dot{x}^2 + \frac{1}{2} b \dot{\theta}^2 + a \dot{x} \dot{\theta} cos(\theta) - agscos(\theta)$$

$$\frac{\partial}{dt}(\frac{\partial L}{\partial \dot{x}}) - \frac{\partial L}{\partial x} = F - f_a \dot{x}$$

$$\frac{\partial}{dt}(\frac{\partial L}{\partial \dot{\theta}}) - \frac{\partial L}{\partial \theta} = -f_b \dot{\theta}$$

Appendix B
Linear State-Space Model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-f_a b}{a^2 - Mb} & \frac{-abg}{a^2 - Mb} & \frac{bf_b}{a^2 - Mb} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{af_a}{a^2 - Mb} & \frac{aMg}{a^2 - Mb} & \frac{-f_b M}{a^2 - Mb} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{b}{a^2 - Mb} \\ 0 \\ \frac{-a}{a^2 - Mb} \end{bmatrix} u$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + 0$$