Daniel Foreman and Praveen Ravishankar 10/09/2020
ME 6404 Lab 4 Report

Introduction

Inverted pendulums are a class of control problems that apply to many real-world devices such as segways and self-balancing robots. The inverted pendulum problem involves balancing a pendulum attached to a moving cart in its unstable, vertical equilibrium as seen in Figure 1 of Appendix A. Typically, the problem is twofold: one to get the pendulum into its unstable equilibrium from its stable downward equilibrium, and two to maintain the pendulum in its unstable equilibrium. A common approach to solve these dual problems is to use linear control law when the pendulum is near vertical, and a nonlinear control-scheme to elevate the pendulum when the pendulum is too far from its vertical equilibrium for the underlying assumptions of linear control law to hold.

The objectives of this lab were to derive and implement the dynamics of an inverted pendulum in software simulation, as well as develop and execute control laws to stabilize the pendulum's motion. Specifically, the motion of the inverted pendulum was independently controlled and stabilized in two different domains: the force domain and the velocity domain.

Methodology

As seen in Figure 1 of Appendix A, the inverted-pendulum problem offers two degrees of freedom in its system: the horizontal displacement of the cart carrying the pendulum (x), and the counterclockwise angle of the pendulum from its upright position (θ). Additionally, in this case, the pendulum is subject to an external control force along the horizontal direction (F) as well as an angular damping force for the point mass m_p ($b\frac{d\theta}{dt}$). A controller can alternatively provide prescribed force, acceleration, or velocities for the cart to follow which, if implemented correctly, can raise or maintain the pendulum upright above the cart. Using rigid-body assumptions and Newton's second law of motion, equations of motion for the pendulum-cart system were derived based on the assumption that the control input would be a force. These equations are exhibited in Equations 1 of Appendix C, with their derivation shown in Derivation 1 of Appendix D. Via simple algebra, these equations were rewritten to solve for the translational and angular accelerations as shown in Equations 2 of Appendix C. Using Equations 2, the state-space representation of the inverted pendulum was formed using the system's angular and translational positions and velocities as the state variables, which is shown in Equations 3. Furthermore, as shown in Derivation 2 of Appendix D, using the small-angle approximation, the linearized state-space representation of the system was then created, as shown in Equations 4. Similarly, by rearranging Equation (2) in Equations 1 of Appendix C, an alternate state-space representation of the system was formed using the translational acceleration as the input, which is shown in Equations 5. Via differentiation, which is exhibited in Derivation 3, an additional state-space representation of the inverted pendulum system was created using the translational velocity as the system's input, which is showcased in Equations 6. All these aforementioned equations are vital in programming the dynamics of the inverted pendulum system and creating the corresponding control laws either in the force domain and/or the velocity domain.

To test system dynamics, a pre-made MATLAB simulation was utilized, which was able to track the pendulum's angular displacement, angular velocity, and the cart's horizontal displacement, as well as graphically simulate the "physical" representation of the system's state on a cart-pendulum CAD model. The non-linear equations of motion for both the force and velocity modes were used in the simulation to generate physics-adherent pendulum motion for the virtual pendulum. The resulting motion was tested by manually moving the cart via keypresses and observing the resulting pendulum motion.

Results and Discussion

To actuate the pendulum into its vertical equilibrium when the pendulum is far from upright, a feedback linearization controller was derived for the force-based controller. As shown in Derivation 4 in Appendix D, by operating on only the angular states of the inverted pendulum system, this feedback linearization created a specific type of control input (represented as "u" in the derivation) that transformed the nonlinear system into an alternate linear system involving only the angular states. However, as indicated in

this derivation, the resulting 2 x 2 state matrix ("A") of the linear system contains two constants, k_o and k_1, whose values must be determined. Using the fact that the eigenvalues of the state matrix of a linear system must be on the left half of the complex plane in order for the system to reliably achieve stability, the expressions for the two eigenvalues of the state matrix in terms of k_o and k_1 yielded two constraints on the values of these constants, which is seen in the derivation as well. Although an endless number of different combinations of the values of k_o and k_1 would theoretically cause the two eigenvalues of the state matrix to have negative, real parts, different choices for these two constants would yield different control performances in flipping the pendulum to its vertical, upright position. This performance is not only measured by the amount of time required to flip the pendulum to its upright position (given that the pendulum is initially pointing downwards), but also is characterized by the horizontal distance translated by the cart and its final horizontal velocity; to elaborate, smaller amounts of time in flipping the pendulum, smaller horizontal distances traveled, and smaller horizontal velocities achieved correspond to better controller performances, and vice-versa. After experimentation with different values of these two constants, two values of k_o and k_1 that greatly minimized the three previously-mentioned quantities were 10 and $\frac{20}{3}$ respectively, which corresponded to eigenvalues of -2.279 and -4.387. Figure 2 in Appendix A is a plot of these two eigenvalues on the complex plane. Because the two eigenvalues have negative real parts and therefore are in the left-half of the complex plane, it is implied that the system should theoretically achieve angular stability. Because the simulations using these two eigenvalues resulted in reliable angular stabilization of the inverted pendulum system each time they were run, the controlled dynamics of the angular position and velocity states of the system match the behavior predicted by the location of these two eigenvalues. However, during the experimentation process of finding the optimal values of k_o and k_1, it was discovered that some tested negative eigenvalues (which predict that the system theoretically achieves stability) actually caused the angular states of the system to become unstable. This is due to the additional stability requirements that digitized systems, such as the simulated pendulum, require over continuous systems, which the controller assumes the system is in its calculations. The added delay and complications with sampling holds can often destabilize a digitized-continuous system in comparison to its purely continuous system counterpart.

When the pendulum is near upright, linear control offers a more robust control solution than the previously mentioned feedback linearization controller. The linearized systems used, given by Equations 4 and Equations 6, were obtained using the small angle approximation under the assumption of force-based or velocity-based control, respectively. The use of such linearization allows for better control characterization due to the well-defined properties of linear systems. To bring the cart-pendulum system to its desired target state (pendulum upright and cart having no displacement), an LQR controller was used to specify feedback gains (K) for the linearized controller, identifying optimal gains that would minimize control effort and error-weighted time spent in erroneous states in accordance to their relative weighting in the Q and R matrices. To elaborate, as the Q matrix was ensured to have the same dimensions as the state matrix for both the force-domain and velocity-domain cases, the Q matrix was formed to be a diagonal matrix whose diagonal entries represent the weights (penalties) applied to each of the state variables; as a result, the dimensions of the Q matrix were 4 x 4 for the force-domain case and 3 x 3 for the velocity-domain case. Table 1 showcases the values of the entries of the Q and R matrices for the force-based LQR controller, while Table 2 showcases the same information for the velocity-based LQR controller. As seen in this table, for both domains, because the system's angular position and velocity were desired to not exceed $\pi/10$ rad and π rad/s respectively, the penalties for these parameters in the Q matrix were chosen to be the reciprocals of these limits, which were $10/\pi$ and $1/\pi$, respectively. The other diagonal entries of the Q matrix, which correspond to the translational position and velocities of the system (Q_{33} and Q_{44}) for the force-domain case and the translational position in the velocity-domain case (Q_{33}), and the R-value were picked after experimentation to yield a controller that could rapidly and stably return the system to its target state after disturbances. Hence, using MATLAB's lqr() function, these choices in the values of Q's diagonal entries and R resulted in the calculated feedback gains for the control matrix (K) in both the force and velocity modes, which are additionally shown in Tables 1 and 2, respectively.

As feedback linearization in the force domain was effective in flipping the pendulum to its unstable vertical equilibrium at the cost of causing the system's translational movements to diverge, and as LQR control reliably allowed the system to achieve a negligible final displacement and velocity while maintaining

the unstable vertical equilibrium of the pendulum at the cost of being functional for only small angles, the two types of controllers were combined only in the force domain in an attempt to flip the pendulum to its upright position after initially being downward and to return the pendulum to its initial starting position with little to no velocity, all while being resistant to any external disturbances applied to the system via the keypresses in the simulation. Specifically, if the angle was detected to be greater than $\pi/10$ rad, feedback linearization was used to decrease the angle to a near-upright position; after the angle was reduced to be below $\pi/10$ rad, feedback linearization was disabled and LQR control in the force domain was enabled in order to further stabilize the upright angle of the inverted pendulum while returning the pendulum to a starting position of zero. As seen in the group's accompanying video of the simulation results, the implementation of this combined controller yielded satisfactory results in reliably transforming the inverted pendulum to its desired zero states over a wide range of initial conditions, thus emphasizing the robustness of this controller.

Conclusion

Feedback linearization using force-based control was successful in raising the pendulum when the pendulum initially started below the cart. While this technique was robust to the heavy nonlinearity of the sine/cosines in the equations of motion, it would not have worked to cancel out nonlinearity had the system's parameters not been exactly known, such as in a real-world application. Additionally, while the technique raised the pendulum above the cart, it caused the cart's uncontrolled horizontal displacement to diverge, which would not be desirable in a real-world application. It is clear that while feedback linearization is a powerful tool, it alone is not sufficient for addressing the inverted pendulum problem.

LQR controllers in both the force and velocity domains were able to successfully maintain the pendulum's position above the cart while simultaneously bringing the cart back to its starting position. Specifically, while the pendulum was near upright, the LQR controller was able to reliably restore the pendulum and cart back to their desired positions, even in the presence of disruptive force/velocity inputs. Notably, if the disruption was large enough to push the pendulum far from its vertical position, the control would devolve and become unstable. In this simulation, the LQR controllers were designed to be functional given the inverted pendulum's angle remains below $\pi/10$ radians; however, further testing revealed that these LQR controllers remain functional for larger angles up to $\pi/4$ radians. Unlike feedback linearization, LQR controllers are dependent on the small angle approximation for their linearization requirement and hence work very poorly at large θ values. However, LQR controllers nevertheless excel in their ease of use and tuning. The Q and R matrices allow a user to tune for performance characteristics, rather than estimate such interactions from pole placements. Increasing Q values penalizes errors in system states more, while increasing the R value penalizes the required control effort. Abstracting the control problem in such a way makes designing linear controllers easier, at the cost of ignoring other countervailing concerns such as max overshoot or resulting oscillation frequencies of the system.

Developing controllers in alternate domains (force, velocity) offer multiple ways of approaching the same problem. In certain cases, it may be more convenient to develop a controller in one domain over the other. In the case of the inverted pendulum problem, modeling the problem in terms of an acceleration-based control as seen in Figure 5 produces a far simpler state matrix than modeling the problem with force-based control as seen in Figure 3. Additionally, it may be beneficial to model a system with control inputs in a particular domain, as certain domains may be far more controllable or measurable based on the hardware being controlled, requiring the use of alternate control domain formulations. However, the use of developing controllers in different domains may increase the complexity/difficulty in the controller design, causing the process to become more error-prone.

The MATLAB simulation offered an easy way to rapidly prototype different control laws, which gave control engineers the ability to see real-time system states and even a graphic of the system being controlled. However, it is not a perfect simulation, as seen when certain control laws that are expected to have worked in theory failed in simulation; as stated previously, this is due to the fact that both the control law and the physics of the system are controlled in the discrete time-domain, while on an actual inverted pendulum system, only the controller would function in a discrete time-domain while the inverted pendulum plant would act continuously.

Appendix A - Figures

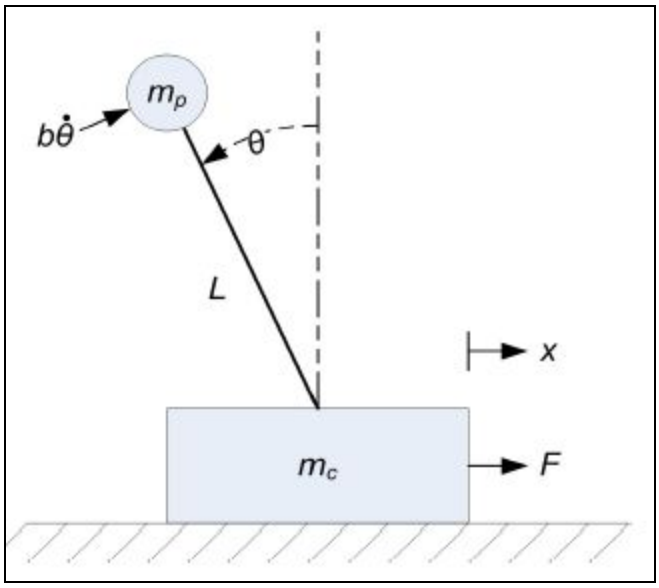


Figure 1: Diagram of the Inverted Pendulum System

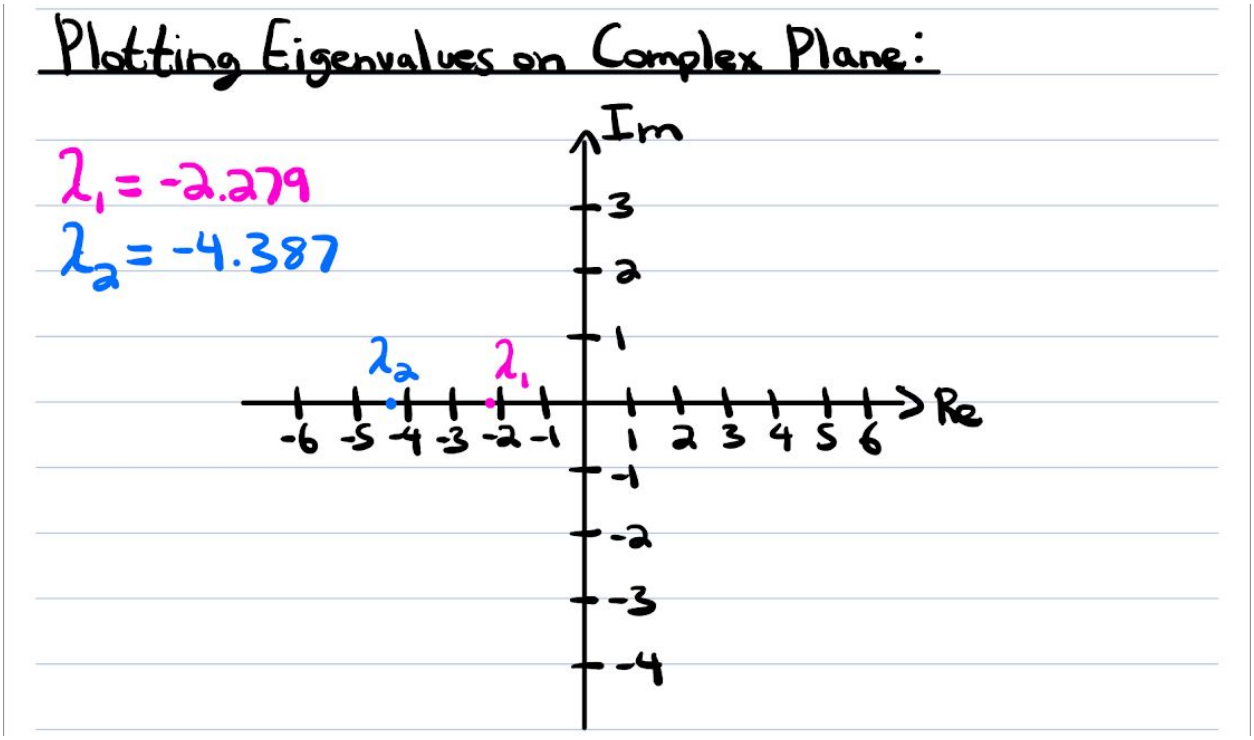


Figure 2: Plot of the Eigenvalues of the Feedback Linearization State Matrix on the Complex Plane

Appendix B - Tables

Table 1: Entries of the Q and R Matrices and Gains of the LQR Controller of the Linearized System in the Force Domain

$\mathbf{Q_{11}}$	$10/\pi$
$\mathbf{Q_{22}}$	$1/\pi$
$\mathbf{Q_{33}}$	100/3
$\mathbf{Q_{44}}$	1
R	0.5
$\mathbf{K_1}$	139.6402
$\mathbf{K_2}$	40.7009
$\mathbf{K_3}$	-8.1650
$\mathbf{K_4}$	-13.0049

Table 2: Entries of the Q and R Matrices and Gains of the LQR Controller of the Linearized System in the Velocity Domain

$\mathbf{Q_{11}}$	$10/\pi$
$\mathbf{Q_{22}}$	$1/\pi$
$\mathbf{Q_{33}}$	100
R	15
$\mathbf{K_1}$	36.0602
$\mathbf{K_2}$	11.3308
$\mathbf{K_3}$	-2.5820

Appendix C - Equations

Non-Linear EoMs:

$$(m_p + m_c)\ddot{x} - b\dot{\theta}\cos\theta - m_p\ddot{\theta}L\cos\theta + m_p\dot{\theta}^2 L\sin\theta = F \quad (1)$$

$$\ddot{x}\cos\theta - \ddot{\theta}L + g\sin\theta - \frac{1}{m_p}b\dot{\theta} = 0 \quad (2)$$

Equations 1: Non-Linear Equations of Motion (EoMs) of the Inverted Pendulum System Assuming Force Input

$$\ddot{x} = \frac{F + m_p g s\theta c\theta - m_p\dot{\theta}^2 L s\theta}{M - m_p c^2\theta} \quad (3)$$

$$\ddot{\theta} = \frac{Fc\theta + (c^2\theta - \frac{M}{m_p})b\dot{\theta} - m_p L\dot{\theta}^2 s\theta c\theta + M g s\theta}{ML - m_p L c^2\theta} \quad (4)$$

Equations 2: Non-Linear EoMs of the Inverted Pendulum System Assuming Force Input (Rewritten to Solve for Translational and Angular Acceleration)

$$3.\ q_1 = \theta;\ q_2 = \dot{\theta};\ q_3 = x;\ q_4 = \dot{x}$$

$$\dot{q}_2 = \frac{b q_2 (c^2 q_1 - M/m_p) - m_p L q_2^2 s_{q_1} c_{q_1} + M g s_{q_1}}{L(M - m_p c^2 q_1)} + \frac{c_{q_1}}{L(M - m_p c^2 q_1)} F$$

$$\dot{q}_4 = \frac{m_p g s_{q_1} c_{q_1} - m_p L q_2^2 s_{q_1}}{M - m_p c^2 q_1} + \frac{F}{M - m_p c^2 q_1}$$

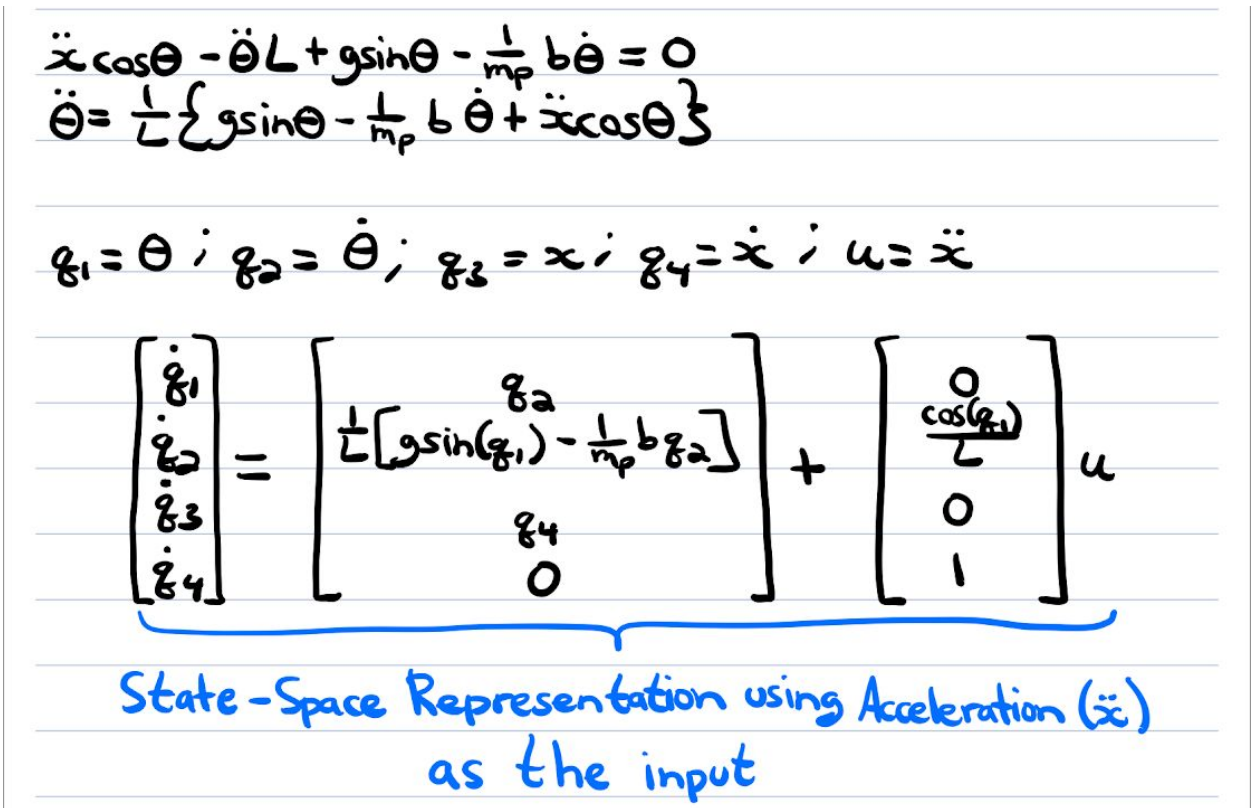
State - Space Equations:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} q_2 \\ \frac{b q_2 (c^2 q_1 - M/m_p) - m_p L q_2^2 s_{q_1} c_{q_1} + M g s_{q_1}}{L(M - m_p c^2 q_1)} \\ q_4 \\ \frac{m_p g s_{q_1} c_{q_1} - m_p L q_2^2 s_{q_1}}{M - m_p c^2 q_1} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{c_{q_1}}{L(M - m_p c^2 q_1)} \\ 0 \\ \frac{1}{M - m_p c^2 q_1} \end{bmatrix} F$$

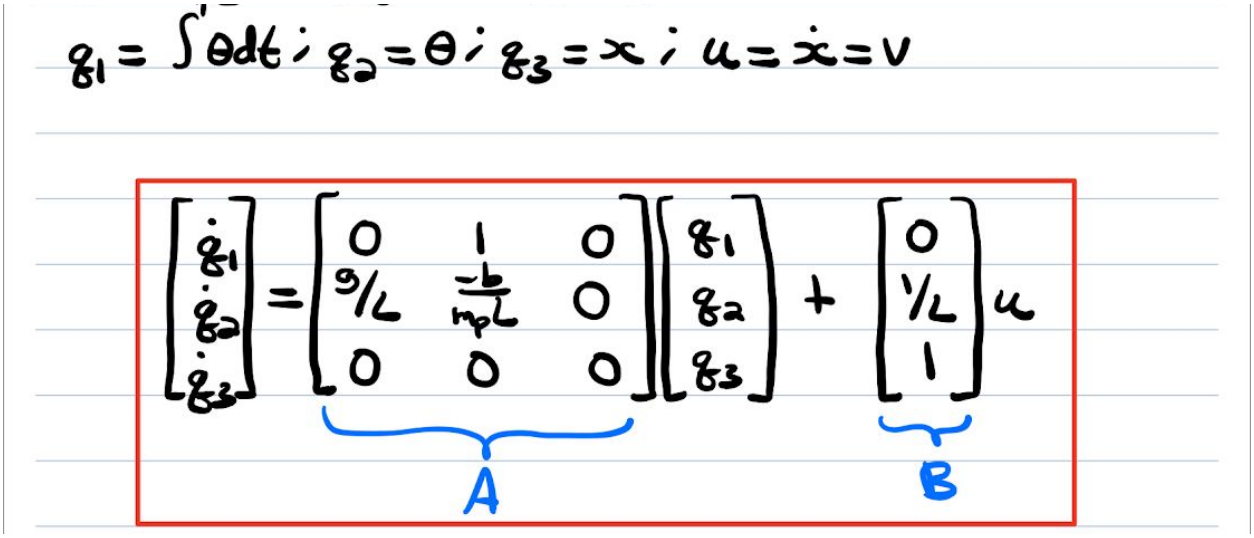
Equations 3: Non-Linear State-Space Representation of the Inverted Pendulum System using Force as the Input

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M g}{L m_c} & \frac{-b}{m_p L} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{m_p g}{m_c} & 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1/L m_c \\ 0 \\ 1/m_c \end{bmatrix}}_{B} F$$

Equations 4: Linearized State-Space Representation of the Inverted Pendulum System using Force as the Input



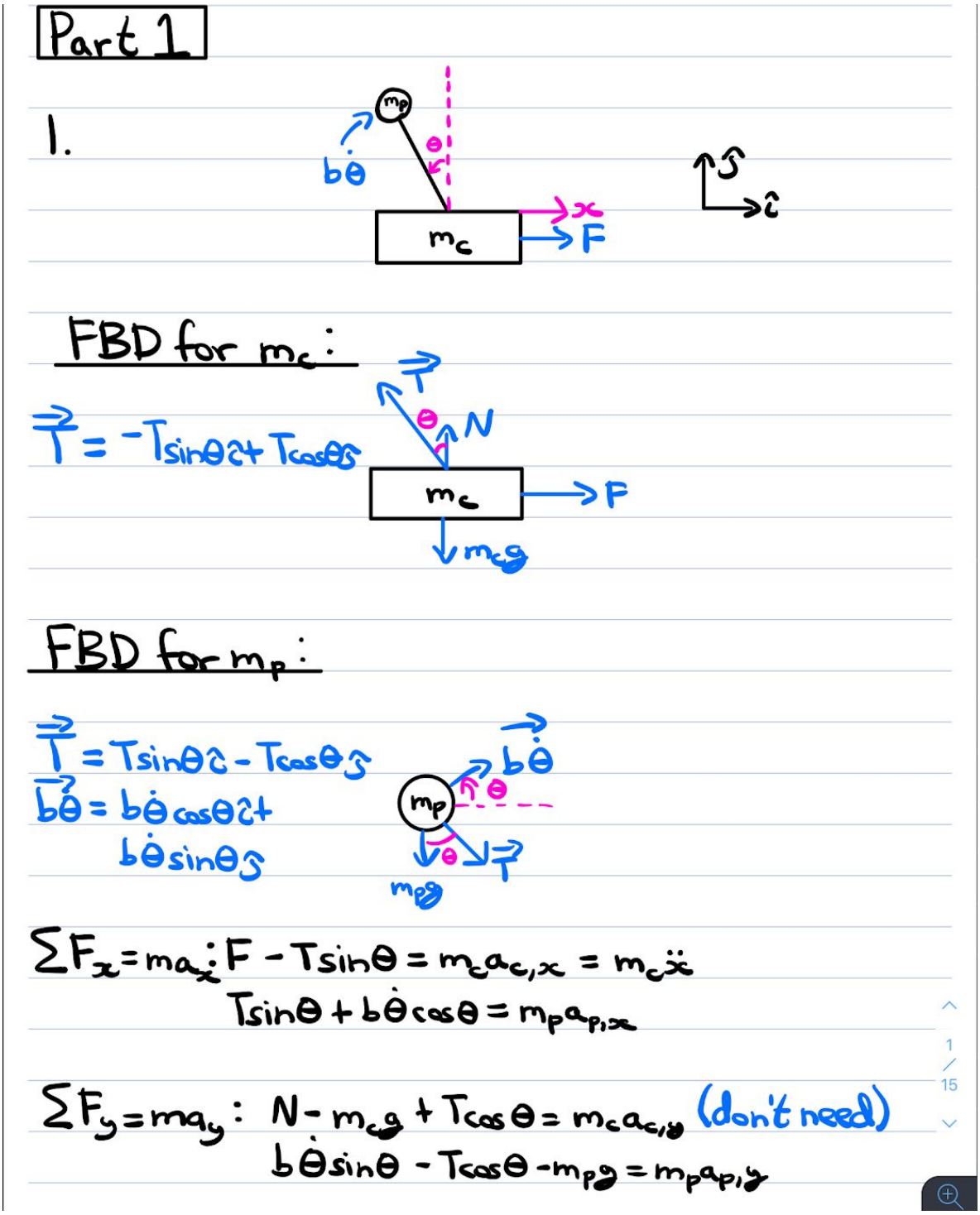
Equations 5: Non-Linear State-Space Representation of the Inverted Pendulum System using Translational Acceleration as the Input



Equations 6: Linearized State-Space Representation of the Inverted Pendulum System using Translational Velocity as the Input

Appendix D - Derivations

Derivation 1 - Derivation of the Non-Linear EoMs of the Inverted Pendulum System:



$$\vec{a}_p = \vec{a}_c + (\vec{a}_p)_c + \vec{\alpha}_c \times \vec{r}_{cp} + \vec{\omega}_c \times (\vec{\omega}_c \times \vec{r}_{cp}) + 2\vec{\omega}_c \times (\vec{v}_p)_c$$

relative, tangential, centripetal, coriolis

$$\vec{a}_p = \ddot{x}\hat{\imath} + [\ddot{\theta} L \hat{e}_t - \dot{\theta}^2 L \hat{e}_r]$$

$$\hat{e}_t = -\cos\theta\,\hat{\imath} - \sin\theta\,\hat{\jmath} \;;\; \hat{e}_r = -\sin\theta\,\hat{\imath} + \cos\theta\,\hat{\jmath}$$

$$a_{p,x} = \ddot{x} - \ddot{\theta} L \cos\theta + \dot{\theta}^2 L \sin\theta$$
$$a_{p,y} = -\ddot{\theta} L \sin\theta - \dot{\theta}^2 L \cos\theta$$

$$F - T\sin\theta = m_c \ddot{x} \quad \dots (1)$$
$$T\sin\theta + b\dot{\theta}\cos\theta = m_p(\ddot{x} - \ddot{\theta} L\cos\theta + \dot{\theta}^2 L\sin\theta) \quad \dots (2)$$
$$b\dot{\theta}\sin\theta - T\cos\theta = m_p(-\ddot{\theta} L\sin\theta - \dot{\theta}^2 L\cos\theta) \quad \dots (3)$$
$$-m_p g$$

(1) → (2)

$$F - m_c\ddot{x} + b\dot{\theta}\cos\theta = m_p(\ddot{x} - \ddot{\theta} L\cos\theta + \dot{\theta}^2 L\sin\theta)$$

$\cos\theta(2) + \sin\theta(3)$:

$$b\dot{\theta} = m_p\cos\theta(\ddot{x} - \ddot{\theta} L\cos\theta + \dot{\theta}^2 L\sin\theta) + m_p\sin\theta(-\ddot{\theta} L\sin\theta - \dot{\theta}^2 L\cos\theta)$$
$$-m_p g\sin\theta$$

$$(m_p + m_c)\ddot{x} = F + b\dot{\theta}\cos\theta + m_p\ddot{\theta} L\cos\theta - m_p\dot{\theta}^2 L\sin\theta$$

$$\frac{b\dot{\theta}}{m_p} - g\sin\theta = \ddot{x}\cos\theta - \ddot{\theta} L$$

Non-Linear EoMs:

$$(m_p + m_c)\ddot{x} - b\dot{\theta}\cos\theta - m_p\ddot{\theta} L\cos\theta + m_p\dot{\theta}^2 L\sin\theta = F \quad (1)$$
$$\ddot{x}\cos\theta - \ddot{\theta} L + g\sin\theta - \frac{1}{m_p} b\dot{\theta} = 0 \quad (2)$$

Derivation 2 - Derivation of the Linearized State-Space Representation of the Inverted Pendulum System using Force as Input:

4. To linearize the system, the small-angle assumption is made ($s\theta \approx \theta$, $c\theta \approx 1$):

$$\dot{q}_2 = \frac{b q_2 (1 - M/m_p) - m_p L q_2^2 q_1 + M g q_1}{L(M - m_p)} + \frac{1}{L(M-m_p)} F$$

$$\dot{q}_4 = \frac{m_p g q_1 - m_p L q_2^2 q_1}{M - m_p} + \frac{1}{M - m_p} F$$

Additionally, terms such as $(q_2^2 q_1)$ must be linearized about $(\theta, \dot{\theta}) = (0,0)$:

$$\tilde{f}(x,y) \approx f(a,b) + f_x(0,0)(x) + f_y(0,0)(y)$$

$$q_2^2 q_1 = 0 + \underbrace{2 q_1 q_2 |_{q_1, q_2 = 0}}_{0} + \underbrace{q_2^2 |_{q_1, q_2 = 0}}_{0} = 0$$

$$\dot{q}_2 = \frac{M g q_1}{L(M-m_p)} + \underbrace{\frac{b(1 - M/m_p)}{L(M-m_p)} q_2}_{\frac{-b}{L m_p} q_2} + \frac{1}{L(M-m_p)} F$$

$$\dot{q}_4 = \frac{m_p g}{M - m_p} q_1 + \frac{1}{M - m_p} F \; ; \; M - m_p = m_c$$

Hence, the linearized state-space representation of this system is:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{Mg}{L m_c} & \frac{-b}{m_p L} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{m_p g}{m_c} & 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1/L m_c \\ 0 \\ 1/m_c \end{bmatrix}}_{B} F$$

Derivation 3 - Derivation of the Linearized State-Space Representation of the Inverted Pendulum System using Translational Velocity as Input:

$$\ddot{x}\cos\theta - \ddot{\theta}L + g\sin\theta - \frac{1}{mp}b\dot{\theta} = 0$$

Linearizing about $(\theta, \dot{\theta}) = (0,0)$:

$$\ddot{x} - \ddot{\theta}L + g\theta - \frac{b}{mp}\dot{\theta} = 0$$

$$\ddot{\theta}L + \frac{b}{mp}\dot{\theta} - g\theta = \ddot{x} \Rightarrow \ddot{\theta} + \frac{b}{mpL}\dot{\theta} - \frac{g}{L}\theta = \frac{\ddot{x}}{L}$$

Differentiating

$$\dot{\theta} + \frac{b}{mpL}\theta - \frac{g}{L}\int\theta dt = \frac{\dot{x}}{L} = \frac{v}{L}$$

$$q_1 = \int\theta dt \; ; \; q_2 = \theta \; ; \; q_3 = x \; ; \; u = \dot{x} = v$$

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ g/L & -\frac{b}{mpL} & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1/L \\ 1 \end{bmatrix}}_{B} u$$

Derivation 4 - Derivation of the Control Law and the State-Space Representation of the Inverted Pendulum System by using Feedback Linearization:

$$\ddot{x}\cos\theta - \ddot{\theta}L + g\sin\theta - \frac{1}{mp}b\dot{\theta} = 0$$

$$\ddot{\theta} = \frac{1}{L}\left\{\ddot{x}c\theta + gs\theta - \frac{1}{mp}b\dot{\theta}\right\}$$

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\\ \dot{q}_4\end{bmatrix} = \begin{bmatrix} q_2 \\ \frac{1}{L}\{Ac_{q_1} + gs_{q_1} - \frac{1}{mp}bq_2\} \\ q_4 \\ A\end{bmatrix}$$

Feedback Linearization:

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix} = \begin{bmatrix} q_2 \\ \frac{1}{L}\left(gs_{q_1} - \frac{1}{mp}bq_2\right)\end{bmatrix} + \begin{bmatrix} 0 \\ \left(\frac{c_{q_1}}{L}\right)A\end{bmatrix}$$

$$\dot{\bar{q}} = \begin{bmatrix} q_2 \\ f(\bar{q}) + b(\bar{q})u\end{bmatrix}$$

Let $u = \frac{1}{b}(v - f)$

$$v = -k_0 q_1 - k_1 q_2$$

$$u = \frac{L}{c_{q_1}}\left(v - \frac{g}{L}s_{q_1} + \frac{1}{mpL}bq_2\right) \rightarrow \text{control input!}$$

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k_0 & -k_1\end{bmatrix}}_{A}\begin{bmatrix} q_1 \\ q_2\end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 \\ -k_0 & -k_1 \end{bmatrix}$$

$$A - \lambda I = \begin{bmatrix} -\lambda & 1 \\ -k_0 & -k_1 - \lambda \end{bmatrix}$$

$$|A - \lambda I| = \lambda(k_1 + \lambda) + k_0 = 0$$

$$\Rightarrow \lambda^2 + k_1\lambda + k_0 = 0$$

$$\lambda = \frac{-k_1 \pm \sqrt{k_1^2 - 4k_0}}{2}$$

λ must be in the left half of the complex plane so the system is stable:

$$-k_1 + \sqrt{k_1^2 - 4k_0} < 0$$

$$-k_1 - \sqrt{k_1^2 - 4k_0} < 0$$

After experimentation, $k_0 = 10$ and $k_1 = 20/3$ were found to produce the best results:

$$\lambda_1 = \frac{-k_1 + \sqrt{k_1^2 - 4k_0}}{2} \qquad \lambda_2 = \frac{-k_1 + \sqrt{k_1^2 - 4k_0}}{2}$$

$$\lambda_1 = -2.279 \qquad \lambda_2 = -4.387$$

Appendix E - MATLAB Code

source.m

```
function source
clear all
clear classes
clc

z1 = 0;

%CONSTANTS
L  = 1.0;             %Length
mc = 3;               %Mass of Cart
mp = 1;               %Mass of Payload
b  = 0.1;             %Damping factor
g  = 9.81;            %Gravity
min_cycle = 0.05;     %Minimum cycle time
Mode = 'Acceleration';  %Velocity or Force or Acceleration
th_in = pi;           %Initial Angle

%PREPARE THE FIGURE WINDOW & GET PENDULUM OBJECT
figure(1); clf
Pend = PENDULUM('InitialStates',[th_in 0 0 0], 'WorkspaceLength', 8,'Damping',b,...
   'MassCart', mc, 'MassPendulum', mp, 'PendulumLength',L,'Mode',Mode);

%PREPARE THE STREAMING FIGURE WINDOWS
figure(2); clf

subplot(3,1,1);
Stream_Theta = STREAM_AXIS_DATA(gca);
ylabel('\theta')

subplot(3,1,2);
Stream_Pos = STREAM_AXIS_DATA(gca);
ylabel('Position')

subplot(3,1,3);
Stream_Vel = STREAM_AXIS_DATA(gca);
ylabel('Velocity')


%BEGINNING OF SIMULATION LOOP
quit   = 0;
U_fbl  = 0;
del    = min_cycle;

while quit == 0

   %KEEP TRACK OF TIME
   now = GETTIME;
```

```
%CHECK IF THE ESC, F1 KEY IS BEING PRESSED
tmp = keyinfo;
tmp = tmp(1);
if  tmp == 27
   quit = 1;
end
clear tmp

%GET VR FROM THE KEYBOARD
%37 = left  arrow  %39 = right arrow
U_manual = GET_MANUAL_COMMAND(37, 39);

% The States
Q  = Pend.Q;
q1 = Q(1);
q2 = Q(2);
q3 = Q(3);
q4 = Q(4);


if(abs(q1) < pi/10)
   Mode = 'Force';
else
   Mode = 'Acceleration';
end

%Feedback Linearization
switch Mode
   case 'Force'
      U_fbl = 0;

   case 'Velocity'
      %U_fbl = ??

   case 'Acceleration'
      k0 = 10;
      k1 = 20/3;
      v = (-k0*q1) - (k1*q2);
      U_fbl = (L/cos(q1)) * (v - ((g/L) * sin(q1)) + ((1/(mp * L))*(b*q2)));
      U_lqr = 0;
end

%Full State Feedback (from LQR)
switch Mode
   case 'Force'
      K = [139.6402   40.7009   -8.1650  -13.0049];
      X = Q;
      U_lqr = -K*X;
   case 'Velocity'
      K = [36.0602   11.3308   -2.5820];
      z1 = z1*0.99 + Q(1)*del;
      z2 = Q(1);
```

```
        z3 = Q(3);
        Z = [z1 z2 z3]';

        U_lqr = -K * Z;
        %U_lqr = ??

        %hint: this mode is sensitive to numerical error that manifests itself in the form of 'integral wind-up'.
See if
        %you can understand why this is the case. To make this mode work reliably in this simulation
environment, you need to
        %mitigate the integral wind-up.

        %K      = [??  ??  ??];
        %z1     = z1*0.99 + Q(1)*del;            %0.99 mitigates integral wind-up in the z1 state caused by
numerical errors.
        %z2     = Q(1);
        %z3     = Q(3);
        %Z      = [z1 z2 z3]';
        %U_lqr   = -K * Z;

      case 'Acceleration'
           %U_lqr = ??
   end


   %Actuate the Pendulum
   Pend.U = U_manual + U_fbl + U_lqr;

   %Trace states
   Stream_Theta.stream(now, Q(1));
   Stream_Pos.stream(now, Q(3));
   Stream_Vel.stream(now, Q(4));

   %update screen
   drawnow

   %HANDLE ANY IDLE TIME
   finish = GETTIME;
   while finish <= now + min_cycle;
      finish = GETTIME;
   end
   del = finish - now;
end


%-------------------------------------------------
function Uout = GET_MANUAL_COMMAND(xm, xp)
umax   = 1;
U      = 0;

tmp = keyinfo;
if sum(abs(tmp)) ~= 0
```

```
    if find(tmp == xp)
        U = U + umax;
    end
    if find(tmp == xm)
        U = U - umax;
    end
end

Uout = U;

%-------------------------------------------------
function out = GETTIME
%time = GETTIME returns the current cpu clock time in milliseconds

tmp = clock;
out = 3600*tmp(4) + 60*tmp(5)+tmp(6);
```

dequations.m

```
function qdot = dequations(T,q,u,L,mp,mc,b,Mode)

%Define some parameters that we can use in our differential equations:
mt = mp+mc;
g = 9.81;
t = q(1);
w = q(2);
x = q(3);
v = q(4);
st = sin(t);
ct = cos(t);

switch Mode

   case 'Force'
      F = u;
      qdot(1,1) = w;
      qdot(2,1) = ((F.*ct) + (((ct.^2) - (mt./mp))*b*w) - (mp*L*(w^2)*st*ct) + (mt*g*st))./((mt*L) -
(mp*L*(ct^2)));
      qdot(3,1) = v;
      qdot(4,1) = (F + (mp*g*st*ct) - (mp*(w^2)*L*st))./(mt - (mp*(ct^2)));

   case 'Velocity'
      A = u;
      qdot(1, 1) = w;
      qdot(2, 1) = ((g*st) - ((b*w)/mp) + (ct * A))/L;
      qdot(3, 1) = v;
      qdot(4, 1) = A;

   case 'Acceleration' %these equations are actually for acceleration control, but the mode
      %for the pendulum class says velocity, and we just differentiate.
      A = u;
      qdot(1,1) = w;
      qdot(2,1) = (A*ct + g*st - ((1/mp)*b*w))/L;
      qdot(3,1) = v;
      qdot(4,1) = A;
end
```

LQRStuff.m (for deriving the gains of the control matrix K)

```
clear
clc

L  = 1.0;            %Length
mc = 3;              %Mass of Cart
mp = 1;              %Mass of Payload
b  = 0.1;            %Damping factor
g  = 9.81;           %Gravity
M = mc + mp;

% Force Domain LQR:
a21 = (M * g)/(L * mc);
a22 = (-b)/(mp * L);
a41 = (mp * g)/(mc);

A = [0 1 0 0; a21 a22 0 0; 0 0 0 1; a41 0 0 0];
B = [0 (1/(L*mc)) 0 (1/mc)]';
Q = [(10/pi) 0 0 0; 0 (1/pi) 0 0; 0 0 (100/3) 0; 0 0 0 1];
R = 0.5;

[K_forc, S, e] = lqr(A, B, Q, R);

% Velocity Domain LQR:
A = [0 1 0; (g/L) (-b/(mp * L)) 0; 0 0 0];
B = [0; (1/L); 1];
Q = [(10/pi) 0 0; 0 (1/pi) 0; 0 0 (300/3)];
R = 15;

[K_vel, S, e] = lqr(A, B, Q, R);
```