Linear Quadratic Regulator Control of an Inverted Pendulum

[redacted]

[redacted]

[redacted]

Research Question:

In a classic inverted pendulum problem, how to identify an optimal ***K*** gain matrix for LQR control at an equilibrium position that will reduce the cost and time to equilibrium for a set of weight matrices ***Q*** and ***R***?

[redacted]

[redacted]

Table of Contents

Nomenclature

Symbol	Definition	Unit
g	Gravitational Constant	(m s^{-2})
θ	Angle Between Pendulum and Vertical Axis	(deg)
M	Mass of Cart	(kg)
m	Mass of Pendulum	(kg)
F	Force	(N)
l	Length of Pendulum	(m)
KE	Kinetic Energy	(kg m^2 s^{-2})
PE	Potential Energy	(J)
v	Velocity	(m s^{-2})
μ	Friction Coefficient	-

Abbreviations

LQR	Linear Quadratic Regulator
IP	Inverted Pendulum
PWM	Pulse-width Modulation
DOF	Degree of Freedom
KE	Kinetic Energy
PE	Potential Energy
H.O.T	Higher Order Terms

1 Introduction

1.1 Background

Throughout the years, there have been massive changes in technological advancements. Ever since the first inventions of machines, technology has continued to thrive. Though, what used to focus on machines' mechanical aspects, these days technology is more on control; a machine that can adapt to its environment. One such example is the Space Shuttle's Main Engine, its power level is controlled via a closed-loop controller. Which means that the output is dependent on the control action of the system. In this case, the main combustion chamber pressure is measured as an indication of thrust level, and in response to that value, a valve is opened or closed to increase or decrease the engine power level. Another example is in robotics, in 4th grade, I used to tinker around with Lego EV3, a computer-controlled hardware. One of the most exciting components that I used was the gyro sensor Fig(1). A sensor that can identify the robot's pitch, yaw, and roll. This robot's general application was to create a loop-based program that will use the robot's state to balance itself upright.

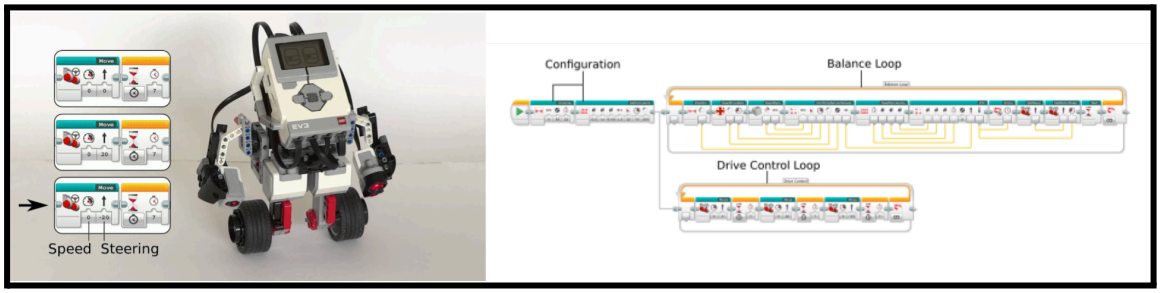

Figure 1: Lego EV3 Gyrobot, closed-loop balancing

This paper will explore control theory through the classic IP (inverted pendulum), a highly unstable system. The system consists of a pole mounted onto a cart that is moving horizontally on a track. It possesses two fixed states: pendulum downwards $(\theta = 0)$ and

pendulum upwards $(\theta = \pi)$. Thus, to achieve this state of equilibria, a force is required to move the cart. To analyze this system, we must first understand its dynamic model. In this case, it has 2DOF (degrees of freedom). Most dynamic model systems are non-linear, and this system is an underactuated mechanical system because it has fewer control inputs than the DOF. In this paper, the LQR method will be employed to obtain optimal control of the IP.

1.2 Objectives

1. Mathematically model the dynamics of the IP by employing Euler-Lagrange's Equation. Deriving the equations of motions.
2. Obtain a Linear state-space representation through Taylor series expansion and Jacobian Matrix
3. Observing system stability and controllability with real-world parameters
4. Design a simple control law by using gain matrix $\boldsymbol{K}$
5. Design a state-feedback control system that will balance the pendulum at equilibrium using LQR
6. Evaluating different $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices
7. Experimenting using Arduino & discuss results

2 Theory

2.1 Equations of Motion

Based on the system dynamics we can identify that it has 2DOF, the cart movement and the rotational motion of the pendulum. The system consists of an inverted pole of mass, *m*, hinged by an angle, θ, of length, *l*, from the vertical axis on a cart with mass, *M*, which is free to move on the horizontal axis, *x*, a force, *F*, moves the cart and is opposed by surface friction, *fric*.

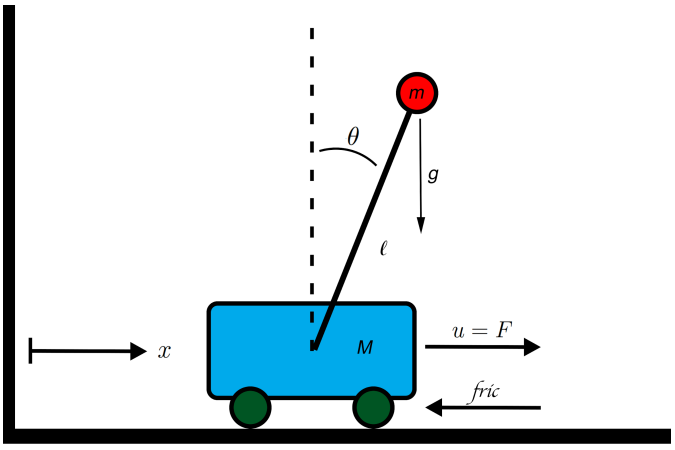

Figure 2: Diagram of the inverted pendulum

Kinematics of the system:

Equations of motion can be identified using the Euler-Lagrange's equation [5] describing how the system behaves over time. With the non-relativistic Lagrangian that describes the state of a dynamic system in terms of position and their time derivatives to be:

$$L = KE - PE \tag{1}$$

Where KE is kinetic energy and PE is potential energy. The only PE is from the pendulum suspended by length, *l*:

$$PE = mgh = mgl\cos\theta \tag{2}$$

Translational KE of the system is the sum of KE in the system:

$$KE = KE_M + KE_m \tag{3}$$

KE of cart is:

$$KE_M = \tfrac{1}{2} M \dot{x}^2 \tag{4}$$

KE of pendulum is:

$$KE_M = \tfrac{1}{2} m \left(\dot{x_m}^2 + \dot{y_m}^2 \right) \tag{5}$$

x and *y* components of the pendulum are:

$$x_m = x + l \sin\theta \tag{6}$$

$$y_m = l \cos\theta \tag{7}$$

For a composite function *x* its time derivative of θ, with chain rule:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\mathrm{d}x}{\mathrm{d}\theta} \cdot \frac{\mathrm{d}\theta}{\mathrm{d}t} \tag{8}$$

For velocities of each component, we take their time derivatives, using overdot to denote its first derivative with respect to time:

$$\dot{x_m} = \tfrac{\mathrm{d}x_m}{\mathrm{d}t} = \tfrac{\mathrm{d}}{\mathrm{d}t}(x + l\sin\theta) = \dot{x} + l\sin\theta\dot{\theta} \tag{9}$$

$$\dot{y_m} = \tfrac{\mathrm{d}y_m}{\mathrm{d}t} = \tfrac{\mathrm{d}}{\mathrm{d}t}(l\cos\theta) = -l\sin\theta\dot{\theta} \tag{10}$$

Inputting to Eq(4) we have:

$$KE_m = \frac{1}{2} m((\dot{x} + l\cos\theta\dot{\theta})^2 + (-l\sin\theta\dot{\theta})^2) \tag{11}$$

$$= \frac{1}{2} m(\dot{x}^2 + 2l\cos\theta\dot{\theta}\dot{x} + l^2\dot{\theta}^2(\cos^2\theta + \sin^2\theta)) \tag{12}$$

$$= \frac{1}{2} m\dot{x}^2 + ml\cos\theta\dot{\theta}\dot{x} + \frac{1}{2} ml^2\dot{\theta}^2 \tag{13}$$

Total KE of the system is Eq(3):

$$KE = \frac{1}{2}(M+m)\dot{x}^2 + ml\cos\theta\dot{\theta}\dot{x} + \frac{1}{2} ml^2\dot{\theta}^2 \tag{14}$$

Therefore, Lagrangian is given to be Eq(1):

$$L = \frac{1}{2}(M+m)\dot{x}^2 + ml\cos\theta\dot{\theta}\dot{x} + \frac{1}{2}ml^2\dot{\theta}^2 - mgl\cos\theta \tag{15}$$

The system has 2DOF, thus two equations of motion, for coordinates x and θ. As x is affected by external controlled force F, we must take friction, *fric,* into account, denoted by $\mu\dot{x}$, where μ is friction coefficient. Additionally, as rotational friction is minimal, it is neglected. Based on Euler-Lagrange [6]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F - \mu\dot{x} \tag{16}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0 \tag{17}$$

The first equation of motion

Partial L with respect to partial $\dot{x}$, power rule:

$$\frac{\partial L}{\partial \dot{x}} = \frac{1}{2}(M+m)2\dot{x}^1 + ml\dot{\theta}\cos\theta + 0 - 0 \tag{18}$$

$$= (M+m)\dot{x} + ml\dot{\theta}\cos\theta \tag{19}$$

Time derivative of Eq(19), chain and product rule, with x and θ functions of time:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} = (M+m)\ddot{x} + ml(-\dot{\theta}\sin\theta\dot{\theta} + \cos\theta\ddot{\theta}) \tag{20}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} = (M+m)\ddot{x} - ml\dot{\theta}^2\sin\theta + ml\ddot{\theta}\cos\theta \tag{21}$$

As x does not appear in the lagrangian, partial L with respect to partial x:

$$\frac{\partial L}{\partial x} = 0 \tag{22}$$

Substituting to Eq(16) the **first equation** of motion is:

$$(M+m)\ddot{x} - ml\dot{\theta}^2\sin\theta + ml\ddot{\theta}\cos\theta = F - \mu\dot{x} \tag{23}$$

The second equation of motion

Partial derivative L with respect to partial $\dot{\theta}$, power rule:

$$\begin{aligned}\frac{\partial L}{\partial \dot{\theta}} &= 0 + ml\dot{x}\cos\theta + \frac{1}{2}ml^2 2\dot{\theta}^1 - 0 \\ &= ml\dot{x}\cos\theta + ml^2\dot{\theta}\end{aligned} \tag{25}$$

Time derivative of Eq(25), chain and product rule, with x and θ functions of time:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} = ml^2\ddot{\theta} + ml(-\dot{x}\sin\theta\dot{\theta} + \cos\theta\ddot{x}) \tag{26}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} = ml^2\ddot{\theta} - ml\dot{x}\dot{\theta}\sin\theta + ml\ddot{x}\cos\theta \tag{27}$$

Partial L with respect to partial θ:

$$\frac{\partial L}{\partial \theta} = -ml\dot{\theta}\dot{x}\sin\theta - (-mgl\sin\theta) \tag{28}$$

Substituting to Eq(17) the **second equation** of motion is:

$$ml^2\ddot{\theta} - ml\dot{x}\dot{\theta}\sin\theta + ml\ddot{x}\cos\theta - (-ml\dot{\theta}\dot{x}\sin\theta - (-mgl\sin\theta)) = 0 \tag{29}$$

$$ml^2\ddot{\theta} + ml\ddot{x}\cos\theta - mgl\sin\theta = 0 \tag{30}$$

Simplify by dividing *ml*:

$$l\ddot{\theta} + \ddot{x}\cos\theta - g\sin\theta = 0 \tag{31}$$

Thus, the equations of motion representing this system are given in Eq(23) and Eq(31).

2.2 State Space Representation

A Dynamical system can be represented by differential equations [14] because of its unique property that how the system transforms at any point in time is a function of its current state. In the IP system, we can see that angular acceleration is a function of its angle. When the system is simulated without any external disturbance, acceleration changes velocity, which changes angle, and changes acceleration.

With this 2DOF system, we have angle and position. So what are its state variables? As the masses are moving, the velocity is a variable as it oscillates, and we need velocity to predict its current position and angle. Thus it's a state variable. Similarly, we need its position to predict acceleration and velocity at a point in time. Acceleration is not a state variable as it is already a byproduct of the position. Thus, the state of a dynamical system can be represented to be the states y, with variables, position, velocity, angle, and angular velocity [16]:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{34}$$

Based on Eq(23) and Eq(31), we have angular and horizontal acceleration, but both are second order equations, thus, to represent it into a system of first order, we take the time derivative of y.

$$\dot{y} = \begin{bmatrix} \dot{y_1} \\ \dot{y_2} \\ \dot{y_3} \\ \dot{y_4} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} \tag{35}$$

First row denotes velocity:

$$\dot{y_1} = y_2 = \dot{x} \tag{36}$$

Second row denotes acceleration, replacing $\ddot{\theta}$ in Eq(23) to terminate any second order variables. Isolating $\ddot{\theta}$ from Eq(31):

$$\ddot{\theta} = \frac{g\sin\theta - \ddot{x}\cos\theta}{l} \tag{37}$$

Substitute to Eq(23), isolate $\ddot{x}$, simplify:

$$F - \mu\dot{x} = (M+m)\ddot{x} + ml\cos\theta\left(\frac{g\sin\theta - \ddot{x}\cos\theta}{l}\right) - ml\dot{\theta}^2\sin\theta \tag{38}$$

$$Fl - \mu\dot{x}l - mlg\cos\theta\sin\theta + ml^2\dot{\theta}^2\sin\theta = \ddot{x}((M+m)l - ml\cos^2\theta) \tag{39}$$

$$\dot{y_2} = \ddot{x} = \frac{Fl - \mu\dot{x}l - mlg\cos\theta\sin\theta + ml^2\dot{\theta}^2\sin\theta}{(M+m)l - ml\cos^2\theta} \tag{40}$$

Third row denotes angular velocity:

$$\dot{y_3} = y_4 = \dot{\theta} \tag{41}$$

Fourth row denotes angular acceleration, replacing $\ddot{x}$ in Eq(31), and with $\ddot{x}$ from Eq(23):

$$\ddot{x} = \frac{F - \mu\dot{x} - ml\ddot{\theta}\cos\theta + ml\dot{\theta}^2\sin\theta}{(M+m)} \tag{42}$$

Substitute to Eq(31), isolate $\ddot{\theta}$, and simplify:

$$0 = l\ddot{\theta} + \frac{F\cos\theta - \mu\dot{x}\cos\theta - ml\ddot{\theta}\cos^2\theta + ml\dot{\theta}^2\sin\theta\cos\theta}{(M+m)} - g\sin\theta \tag{43}$$

$$0 = l\ddot{\theta}(M+m) + F\cos\theta - \mu\dot{x}\cos\theta - ml\ddot{\theta}\cos^2\theta + ml\dot{\theta}^2\sin\theta\cos\theta - g\sin\theta(M+m) \tag{44}$$

$$g\sin\theta(M+m) + \cos\theta\left(\mu\dot{x} + F - ml\dot{\theta}^2\sin\theta\right) = \ddot{\theta}\left(l(M+m) - ml\cos^2\theta\right) \tag{45}$$

$$\dot{y_4} = \ddot{\theta} = \frac{g\sin\theta(M+m) + \cos\theta\left(\mu\dot{x} + F - ml\dot{\theta}^2\sin\theta\right)}{l(M+m) - ml\cos^2\theta} \tag{46}$$

Replacing equations into Eq(35) our state-space equation:

$$\dot{y} = \begin{bmatrix} \dot{x} \\ \frac{Fl - \mu\dot{x}l - mlg\cos\theta\sin\theta + ml^2\dot{\theta}^2\sin\theta}{(M+m)l - ml\cos^2\theta} \\ \dot{\theta} \\ \frac{g\sin\theta(M+m) + \cos\theta\left(\mu\dot{x} + F - ml\dot{\theta}^2\sin\theta\right)}{l(M+m) - ml\cos^2\theta} \end{bmatrix} \tag{47}$$

The following **MATLAB** ODE45, is used to simulate the full nonlinear system:

```
%Modified MATLAB code based on Steve Brunton https://https://www.eigensteve.com/

clear all, close all, clc

m = 0.25;
M = 0.635;
l = 0.5;
g = -9.81;
mu = 0.5;

tspan = 0:.03:50;
y0 = [0; -.01; -pi; 0]; %Initial state
[t,y] = ode45(@(t,y)Equation(y,m,M,l,g,mu,0),tspan,y0); %Integrate the system of differential equation

function dy = Equation(y,m,M,l,g,b,u) %State-space equations

Sinx = sin(y(3));
Cosx = cos(y(3));A
a = ((M+m)*l-m*l*Cosx^2);

dy(1,1) = y(2);
dy(2,1) = (u*l-mu*y(2)*l-m*g*l*Cosx*Sinx+m*l^2*y(4)^2*Sinx)/a;
dy(3,1) = y(4);
dy(4,1) = (g*Sinx*(M+m)+Cosx*mu*y(2)-Cosx*u-m*l*y(4)^2*Sinx*Cosx)/a;

end
```

Figure 3: MATLAB Inverted Pendulum Simulation

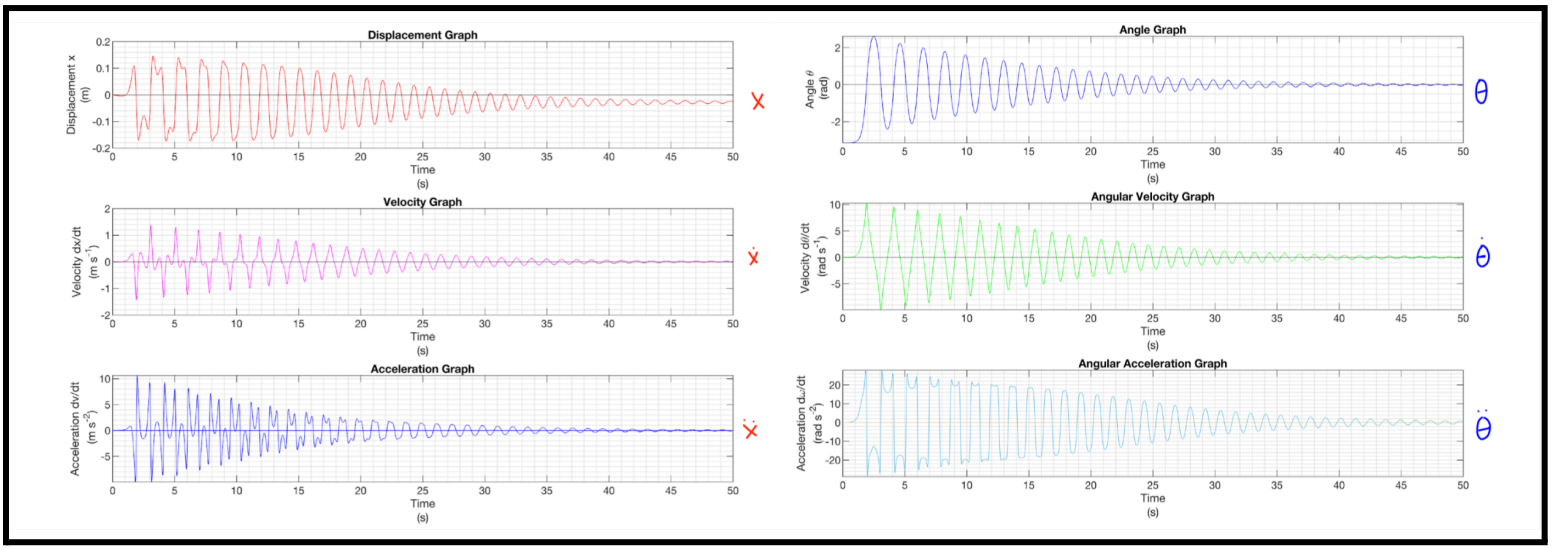

Figure 4:State outputs of simulation

Notice that based on Fig(4), all the values will reach a state of equilibrium where it will reach $y = 0$ after a certain time. Only its displacement will not reach 0. Intuitively we can tell that the equilibrium of the system will not be dependent on x as it does not matter where it is positioned, as long as the velocities, accelerations and angle are 0, its at equilibrium.

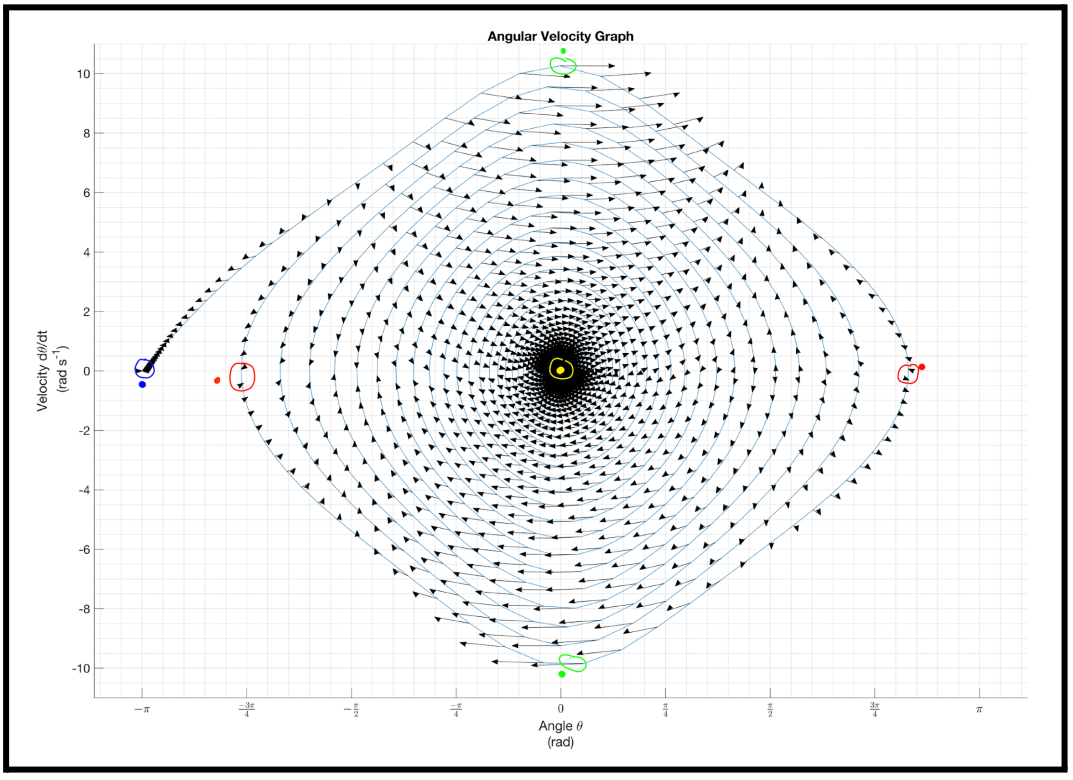

Figure 5: No-control IP Phase Portrait, MATLAB simulation

From the phase portrait, we see that a normal pendulum is a stable system. All points will spiral towards the equilibrium point. I would suggest that the origin point (dot) is a stable equilibrium point and would have negative eigenvalues as all trajectories move towards the origin as $t = \infty$ and towards eigenvectors with more negative eigenvalues. Furthermore, if the system is perturbed slightly it will slowly return to equilibrium point. Unlike our IP system, with equilibrium point at π, slight perturbations will not make it return to equilibrium, thus why IP is unstable [8].

2.3 Linearizing The Non-Linear Model

We have established that our system is non-linear. Though, we can linearize this system upon a fixed point. Thus, our nonlinear model $\dot{y}$, is a function of its current state, *y,* and an external force input, *F,* denoted by *u*:

$$\dot{y} = f(y, u) \tag{48}$$

From the portrait we know that there are two fixed points for angle, up and down, or 0 and π respectively. With both its velocities at 0, and x being a free variable. Thus its fixed points are:

$$\text{fixed point} = y = \begin{bmatrix} 0 \\ 0 \\ \pi \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{49}$$

To linearize, we can use Taylor's series [17], expanded to infinity through its equilibrium position. Meaning that we can get a tangent line from its equilibrium position, and within a small boundary we can linearly approximate it. Our approximation is the small angles approximation in our nonlinear scenario, so when the pendulum is perturbed slightly, either

right or left, we can still use its linear dynamics. Thus, with external force, and a valid control, the pendulum will stay close to its equilibrium point. Taylor series is represented by:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + ... + H.O.T = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \quad (50)$$

As we have two variables, it will be a partial derivative, with y_0 and u_o being its equilibrium positions, and y and u a close point near it, thus we have:

$$\dot{y} = f(y_0, u_0) + \frac{\partial f}{\partial y}(y_0, u_0)(y - y_0) + \frac{\partial f}{\partial u}(y_0, u_0)(u - u_0) + ... + H.O.T \quad (51)$$

Neglecting H.O.T, and approximating $(y - y_0)$, the displacement of points near its equilibrium, we can assume change is so small, because we are taking points that are close to y_0:

$$(y - y_0) \cong \Delta y \quad (u - u_0) \cong \Delta u \quad (52)$$

So points near its equilibrium will be assumed to be linear:

$$\Delta\dot{y} = \frac{\partial f}{\partial y}(y_0, u_0)\Delta y + \frac{\partial f}{\partial u}(y_0, u_0)\Delta u \quad (53)$$

As this is a multivariable partial derivative, we will have multiple functions. With Jacobian matrix [10]:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad (54)$$

Substitute Eq(47) the state-space in terms of $f_n(y, u)$, we have:

$$\dot{y} = \begin{bmatrix} y_2 \\ \frac{ul - \mu y_2 l - mlg\cos y_3 \sin y_3 + ml^2 y_4^2 \sin y_3}{(M+m)l - ml\cos^2 y_3} \\ y_4 \\ \frac{g\sin y_3 (M+m) + \cos y_3 (\mu y_2 + u - mly_4^2 \sin y_3)}{l(M+m) - ml\cos^2 y_3} \end{bmatrix} = \begin{bmatrix} f_1(y,u) \\ f_2(y,u) \\ f_3(y,u) \\ f_4(y,u) \end{bmatrix} \quad (54)$$

From Taylor expansion Eq(51) we have the linear system, where the partial derivatives is a linear combination [2]:

$$\dot{y} = Ay + Bu \tag{55}$$

Where ***A*** and ***B*** are fixed matrices that are evaluated at its first equilibrium point Eq(49):

$$A = \frac{\partial f}{\partial y}|_{(y_0,u_0)} \tag{56}$$

$$B = \frac{\partial f}{\partial u}|_{(y_0,u_0)} \tag{57}$$

Thus, linearizing the nonlinear system around the fixed point, using the Jacobian matrix, $|y_0u_0$ means evaluated at y_0, states at fixed point and no external force, u_0. Matrix ***A*** [11] with reference Eq(54):

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial y_1}\Big|_{y_0,u_0} & \frac{\partial f_1}{\partial y_2}\Big|_{y_0,u_0} & \frac{\partial f_1}{\partial y_3}\Big|_{y_0,u_0} & \frac{\partial f_1}{\partial y_4}\Big|_{y_0,u_0} \\ \frac{\partial f_2}{\partial y_1}\Big|_{y_0,u_0} & \frac{\partial f_2}{\partial y_2}\Big|_{y_0,u_0} & \frac{\partial f_2}{\partial y_3}\Big|_{y_0,u_0} & \frac{\partial f_2}{\partial y_4}\Big|_{y_0,u_0} \\ \frac{\partial f_3}{\partial y_1}\Big|_{y_0,u_0} & \frac{\partial f_3}{\partial y_2}\Big|_{y_0,u_0} & \frac{\partial f_3}{\partial y_3}\Big|_{y_0,u_0} & \frac{\partial f_3}{\partial y_4}\Big|_{y_0,u_0} \\ \frac{\partial f_4}{\partial y_1}\Big|_{y_0,u_0} & \frac{\partial f_4}{\partial y_2}\Big|_{y_0,u_0} & \frac{\partial f_4}{\partial y_3}\Big|_{y_0,u_0} & \frac{\partial f_4}{\partial y_4}\Big|_{y_0,u_0} \end{bmatrix} \tag{58}$$

Utilizing derivative rules, we have ***A*** matrix, Appendix(1):

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{\mu}{M} & -\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\mu}{Ml} & \frac{(M+m)g}{Ml} & 0 \end{bmatrix} \tag{59}$$

Evaluating for Matrix ***B*** with reference to Eq(54), with only one variable:

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial u}\Big|_{y_0,u_0} \\ \frac{\partial f_2}{\partial u}\Big|_{y_0,u_0} \\ \frac{\partial f_3}{\partial u}\Big|_{y_0,u_0} \\ \frac{\partial f_4}{\partial u}\Big|_{y_0,u_0} \end{bmatrix} \tag{60}$$

Our input matrix yields, Appendix(1):

$$B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{Ml} \end{bmatrix} \tag{61}$$

Similarly, using our second fixed point, with derivative rule, matrix ***A*** and ***B***:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{\mu}{M} & \frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{\mu}{Ml} & -\frac{(M+m)g}{Ml} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml} \end{bmatrix} \tag{62}$$

From both matrices notice that their differences are only in operator signs. Simplifying using variable *c*, for pendulum down $(c = -1)$and pendulum up $(c = 1)$, our linearized state space equation, accounting for input force [4].

$$\dot{y} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{\mu}{M} & c\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -c\frac{\mu}{Ml} & -c\frac{(M+m)g}{Ml} & 0 \end{bmatrix} y + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ c\frac{1}{Ml} \end{bmatrix} u \tag{64}$$

Inputting ***A*** and ***B*** matrix into **MATLAB**, with experimental parameters:

Name	Variable	Value
Pendulum Mass	m	0.25kg
Cart Mass	M	0.635kg
Length of Pendulum	*l*	0.5m
Gravity Constant	g	9.8m/s
Cart Friction	μ	0.5Ns/m

*Figure 6: Matrix **A** & **B***

```
clear all, close all, clc

m = 0.25;
M = 0.635;
l = 0.5;
g = -9.81;
mu = 0.5;

c = 1; % -1 for pendulum down

A =[0 1 0 0;
    0 -mu/M c*m*g/M 0;
    0 0 0 1;
    0 -c*mu/M*l -c*(M+m)*g/M*l 0]; % A Matrix

B = [0; 1/M ; 0; c*1/M*l]; B Matrix
```

*Figure 7: Matrix **A** & **B***

And, values [19]:

```
>> A
A =

         0    1.0000         0         0
         0   -0.7874   -3.8622         0
         0         0         0    1.0000
         0   -0.3937    6.8361         0

>> B
B =

         0
    1.5748
         0
    0.7874
```

Figure 8:A & B Output Values

3 LQR Controller

3.1 Stability

Before diving into controller design, we must first determine the stability of the system. Though we have assumed based on the phase portrait that IP is unstable, defining it mathematically would be whether the system will blow up or not as time goes to infinity.

$$\dot{y} = Ay + Bu \tag{65}$$

Stability is determined by the real eigenvalues of matrix $\boldsymbol{A}$. Which is similar to finding the poles of a transfer function (function that models the system's output for all input), the frequencies for which the denominator and numerator of the transfer function is zero [18]. We know that $\boldsymbol{A}$ is a 4×4 matrix, and its eigenvalues can be computed by [8]:

$$Ax = \lambda x \tag{66}$$

Where, λ is the eigenvalues of $\boldsymbol{A}$, and x is its eigenvectors. As this is a 4×4 matrix, evaluating its eigenvalues will be tedious, however, using **MATLAB** we can simplify it by using the built-in 'eig()' function, thus [19]:

```
>> lambda = eig(A)

lambda =

         0
   -2.4313
   -1.0529
    2.6968
```

Figure 9: Eigenvalues of Current Matrix $\boldsymbol{A}$

Consider Eq(55), with the absence of control we can get the exponential stability through [9]:

$$\dot{y} = Ay \tag{67}$$

$$\dot{y} - Ay = 0 \tag{68}$$

Multiply through e^{-At}:

$$e^{-At}\dot{y} - Ae^{-At}y = 0 \tag{69}$$

Product rule:

$$\frac{d}{dt}\left(e^{-At}y\right) = uv' + vu' = e^{-At}\dot{y} - Ae^{-At}y = 0 \tag{70}$$

and,

$$e^{-At}y = c \tag{71}$$

a constant vector, whence,

$$y(t) = e^{At}c \tag{72}$$

c found by taking $t = 0$:

$$y(0) = e^{A\cdot 0}c = Ic = c \tag{73}$$

Thus,

$$y(t) = e^{At}y(0) \tag{74}$$

Matrix exponential is given by [4]:

$$e^{At} = I + At + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \ldots \tag{75}$$

The solution to Eq(74) is purely reliant on its eigenvalues and eigenvectors of the matrix $\boldsymbol{A}$. We know that if $\boldsymbol{A}$ contains real number entries and distinct eigenvalues, then $\boldsymbol{A}$ is diagonalizable and can be written in the form of:

$$A = PDP^{-1} \tag{76}$$

Where, $\boldsymbol{D}$ is a diagonal matrix of eigenvalues, $\boldsymbol{A}$ and $\boldsymbol{P}$ is a matrix whose columns are corresponding linearly independent eigenvectors of $\boldsymbol{A}$. We can replace Eq(74) with Eq(75), and identity of Eq(76):

$$e^{At} = e^{PDP^{-1}t} = PP^{-1} + PDP^{-1} + \frac{PD^2P^{-1}t^2}{2!} + \dots \tag{77}$$

$$= P\left[I + Dt + \frac{D^2t^2}{2!} + \dots\right]P^{-1} \tag{78}$$

With reference to Eq(75) we can simplify Eq(78):

$$= Pe^{Dt}P^{-1} \tag{79}$$

Inputting to Eq(74):

$$y(t) = Pe^{Dt}P^{-1}y(0) \tag{80}$$

For diagonal matrix ***D***, matrix exponential is:

$$e^{Dt} = \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix} \tag{81}$$

Thus, a system is unstable if one of the matrix exponentials blows up, based on Eq(80), as the output is dependent on it. Every eigenvalue will have a real and imaginary part, in the form of:

$$\lambda = a + ib \tag{82}$$

In Euler form:

$$e^{\lambda t} = e^{at}[\cos(bt) + i\sin(bt)] \tag{83}$$

Notice that the cosine and sine component will never have an amplitude > 1, thus only Euler's number and its exponent will tell us if the system decays or grows as time approaches infinity:

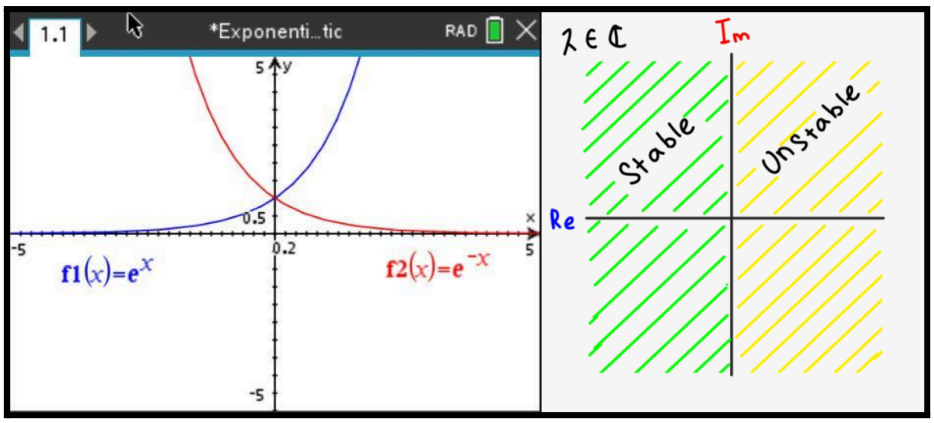

Figure 10: Exponential Graph Characteristic

From Fig(10), and Eq(80), as Euler's number is raised to the diagonal matrix of eigenvalues, if the system has $Re(\lambda) < 0$, it is stable, and solutions will all decay to $y = 0$ as $t = \infty$. As $y(t)$ is the states of our system and over time it should exponentially decay to 0, energy dissipates. Moreover, if there are any eigenvalues in the unstable region, which diverge the fixed point along the unstable eigenvector direction, we would want to use our input to drive it into the stable region. Thus, based Fig(9) we can tell that the IP system is unstable as we have one positive eigenvalue.

3.2 Controllability

So, we already have our state space, and determine that our system is unstable, though before any control design we must first confirm that matrices ***A*** and ***B*** are controllable. If the system is controllable it means we can place the eigenvalues anywhere. Take the form Eq(65):

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} u \tag{84}$$

Eq(84) is uncontrollable as x_1, is unchangeable by u, as matrix ***B*** row 1 is 0. If we modify to:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \end{bmatrix} u \tag{85}$$

The system is now controllable as both x_1 and x_2 is changeable by u as both rows in matrix $\boldsymbol{B}$ will be affected by u. So, controllability can be verified using the controllability matrix, ς, of $\boldsymbol{A}$ and $\boldsymbol{B}$ which is [4]:

$$\varsigma = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{86}$$

System is controllable if and only if rank(ς) = n, where dimension of matrix $\boldsymbol{A}$ is $n \times n$ and dimension matrix $\boldsymbol{B}$ is $n \times q$. Replace Eq(84) to Eq(86), and since $n = 2$, we can stop at second term:

$$\varsigma = \begin{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix} & \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix} \end{bmatrix} \tag{87}$$

$$\varsigma = \begin{bmatrix} 0 & 0 \\ 3 & 3 \end{bmatrix} \tag{88}$$

System is uncontrollable as the upper row is untouched, and constant at 0. Though Eq(85):

$$\varsigma = \begin{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} & \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \end{bmatrix} \tag{89}$$

$$\varsigma = \begin{bmatrix} 1 & 3 \\ 3 & 3 \end{bmatrix} \tag{90}$$

Is controllable, as the upper row is touched. Similarly, in MATLAB we can use [19]:

```
>> c = ctrb(A,B)

>> rank(c)
```

Figure 11: MATLAB Controllability and Rank

'ctrb()' to get ς, and 'rank()' to verify how many variables are touched. In our system based on Fig(8).

```
>> A
A =
         0    1.0000         0         0
         0   -0.7874   -3.8622         0
         0         0         0    1.0000
         0   -0.3937    6.8361         0

>> B
B =
         0
    1.5748
         0
    0.7874

>> ctrb(A,B)
ans =

         0    1.5748   -1.2400   -2.0647
    1.5748   -1.2400   -2.0647    4.0203
         0    0.7874   -0.6200    5.8709
    0.7874   -0.6200    5.8709   -3.4255

>> rank(ctrb(A,B))
ans =

     4
```

Figure 12: Controllability and Rank of System

As our rank is 4, our IP system is **controllable**.

3.3 Closed Loop Pole-Placement

We have established our state-space equation Eq(47), linearize it about a fixed point Eq(64), identify its ***A*** and ***B*** matrix, reason that this system is unstable Fig(9) and justifying that our system is controllable, meaning we spanned all our state-space with the controllability subspace [4], thus, now we know that by measuring the full-state feedback, the position and velocity of cart and pendulum, we can design a closed-loop system to place our eigenvalues anywhere [1].

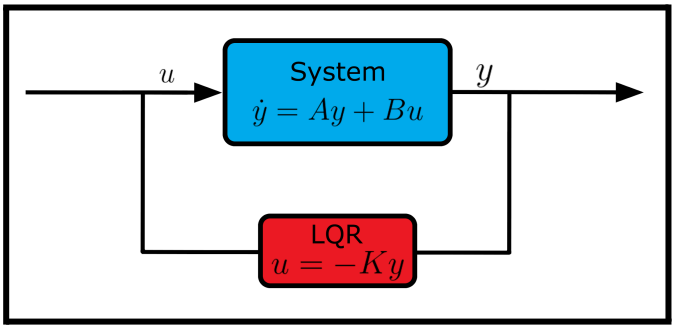

Figure 13: Control System Diagram

A theorem that we can use to drive the eigenvalues to our desired values is called pole-placement. If we refer to, Fig(13) of how our system works, our input *u* can be defined to be [2]:

$$u = -Ky \tag{91}$$

This is our **control law**. Of which, we are measuring our full state y and feeding it back, multiplied by a gain matrix -***K***. If we replace, *u,* in our system equation Eq(65) we have:

$$\dot{y} = Ay - BKy \tag{92}$$

Factoring y leaves us with:

$$\dot{y} = (A - BK)y \tag{93}$$

Where now it becomes a close-loop matrix ***A***, and we can now move the eigenvalues anywhere by a gain matrix ***K***, where it is a 4×1 matrix as we are multiplying it by matrix ***B*** that is of size 4×1. Consider the following examples:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{94}$$

Based on [8]:

$$det(A - \lambda I) = 0 \tag{95}$$

Eigenvalues of matrix ***A*** is:

$$\lambda_1 = 5.3723, \quad \lambda_2 = -0.3723 \tag{96}$$

If this is our system, we can see that it's unstable as λ_1 is positive. To pole-place with Eq(93) we have the new matrix, $\mathring{A}$,of [3]:

$$\mathring{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \tag{97}$$

$$\mathring{A} = \begin{bmatrix} 1 - K_1 & 2 - K_2 \\ 3 & 4 \end{bmatrix} \tag{98}$$

With Eq(95) we have its determinant equation:

$$det(\begin{bmatrix} 1 - K_1 - \lambda & 2 - K_2 \\ 3 & 4 - \lambda \end{bmatrix}) = 0 \tag{99}$$

$$(1 - K_1 - \lambda)(4 - \lambda) - (2 - K_2)(3) = 0 \tag{100}$$

$$\lambda^2 + (K_1 - 5)\lambda + (-4K_1 + 3K_2 - 2) = 0 \tag{101}$$

Thus, stable eigenvalues of:

$$\lambda_1 = -2, \ \lambda_2 = -1 \tag{102}$$

From the desired eigenvalues we have quadratic equation:

$$(\lambda + 2)(\lambda + 1) = 0 \tag{103}$$

$$\lambda^2 + 3\lambda + 2 = 0 \tag{104}$$

To get the desired eigenvalues, we set the coefficients of Eq(101) to Eq(104), which yields our gain matrix ***K***, values:

$$K_1 = 8, \ K_2 = 12 \tag{105}$$

This can be verified in **MATLAB,** recalling 'eigs()' function, we can now verify that our eigenvalues are indeed, -2 and -1 [19]:

```
>> A=[1 2; 3 4]
A =

   1   2
   3   4

>> B=[1;0]
B =

   1
   0

>> K=[8 12]
K =

   8   12

>> eigs(A-B*K)
ans =

  -2.0000
  -1.0000
```

Figure 14: Eigenvalues of Closed-loop A

In **MATLAB** we have a function that can skip all these process, thus, going back to our IP system, using our ***A*** and ***B*** matrix, if want stable eigenvalues of:

$$\lambda_1 = -0.5,\ \lambda_2 = -1,\ \lambda_3 = -1.5,\ \lambda_4 = -2, \tag{106}$$

We can use 'place()' function, where 'eigs' is a matrix of our desired eigenvalues:

```
>> K = place(A,B,eigs)
```

Figure 15: MATLAB Eigenvalue Placement

With matrix ***A*** and ***B*** from our system in Fig(8), we can verify, and get matrix ***K*** of:

```
>> eig=[-0.5;-1;-1.5;-2] %Wanted eigenvalues
eig =

   -0.5000
   -1.0000
   -1.5000
   -2.0000

>> K=place(A,B,eig) %K matrix for desired eigenvalues
K =

   -0.1086   -0.9527   20.0116    7.2554

>> eigs(A-B*K)%Verifying eigenvalues
ans =

   -2.0000
   -1.5000
   -1.0000
   -0.5000
```

Figure 16: Eigenvalue Placement Verification

Thus, If we apply this to our modified script Fig(3), with initial parameters of $[-3 \quad 0 \quad 4 \quad 0]$, and we want it to stabilize at $[1 \quad 0 \quad \pi \quad 0]$, and with the given eigenvalues in Eq(106):

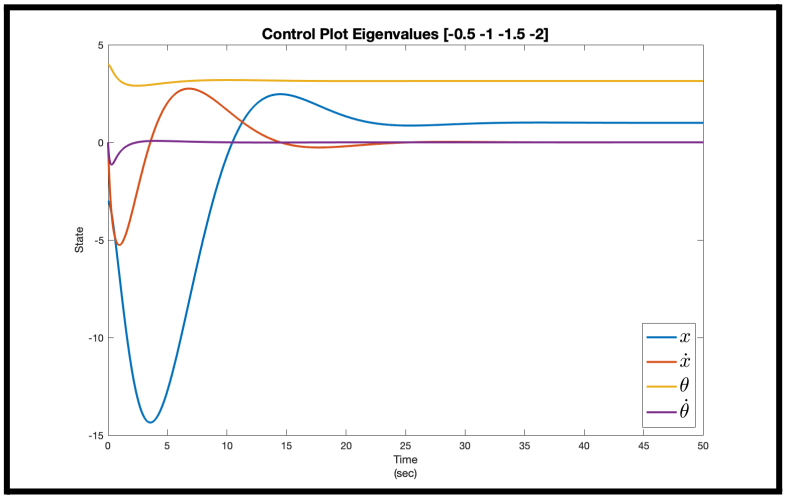

Figure 17: Pole-Placement State Output for Eigenvalues $[-0.5 \quad -1 \quad -1.5 \quad -2]$

From Fig(17), position x is initialized at -3 and θ at 4 radians, we can see that by feeding the system with its full-state and gain by specified stable eigenvalues of ***K***, the system will equilibrate at our desired position of $x = 1$ and θ at upright position π. With its

corresponding velocities at 0, meaning it reached a fixed point. Focusing on state θ, we can vary its eigenvalues.

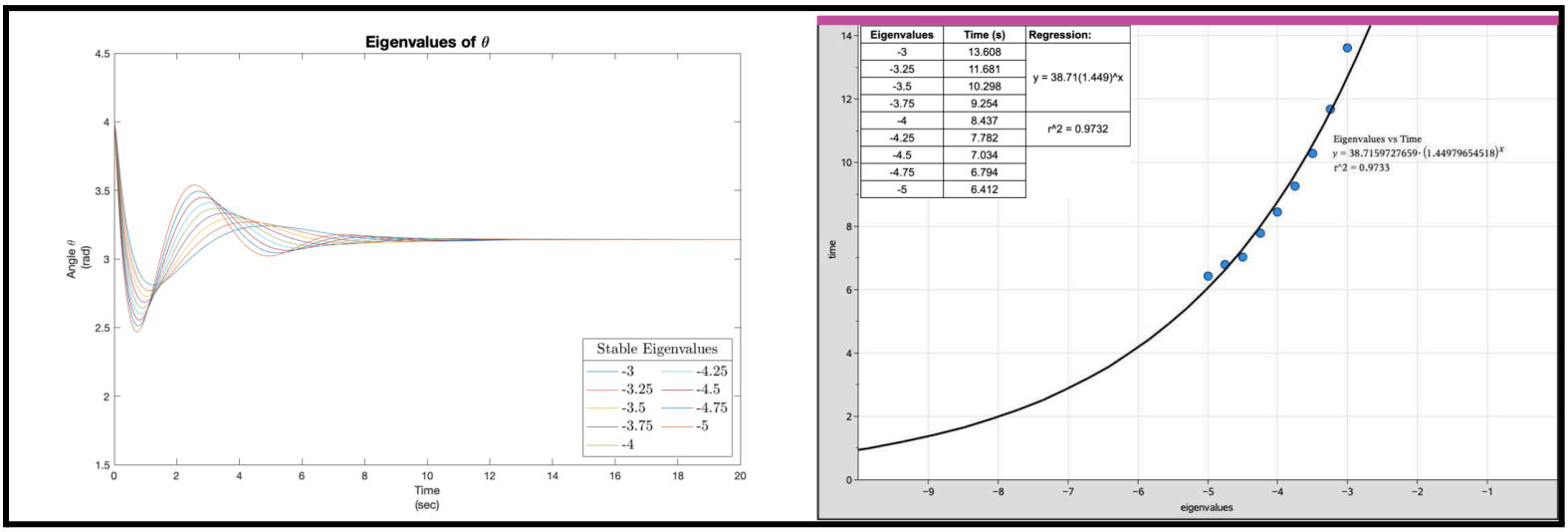

Figure 18: Exponential Model for Eigenvalue Relation and Time

Based on, Fig(18), more negative eigenvalues increases in stability, for θ, the time would decrease exponentially, with a strong correlation $(r^2 = 0.973)$ we can verify that the exponential model is valid. Thus, theoretically if we were to substitute a more stable eigenvalue we would be able to reach the fixed state much more efficiently. This similar trend can be seen with the other states Fig(19).

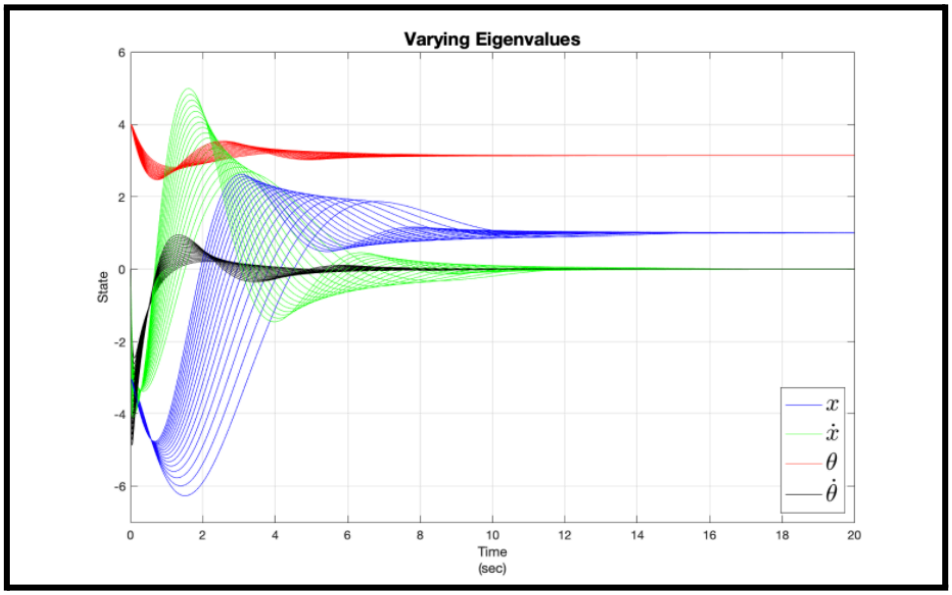

Figure 19: Varying Eigenvalue Data

3.4 Optimal LQR Control

However, we can't immediately assume that more stable eigenvalues would be most optimal for the system. Thus, what are the best eigenvalues? Here, we can use the concept of LQR control, where we can make a cost function that assists us in finding an optimal $\boldsymbol{K}$ by choosing characteristics from the close-loop $\boldsymbol{A}$ that is important [13]. As in experimental setups, we will always have finite resources, i.e. how much it costs if the system converges too slowly or cost for actuation u, motor speed. Thus, increasing eigenvalues stability infinitely is not ideal. We can use an example, in terms of time and DC cost.

Time (s)	DC Motor Cost ($)
20	5
15	10
10	15
5	20

Figure 20: Cost Time and Motor Cost

If we want our system to reach convergence quickly we would pick the $20 motor, though if we are not concerned with time but DC cost we would pick the $5 motor, and optimal solution can be found by:

$$J = Q \cdot time + R \cdot DCcost \tag{107}$$

Where, $\boldsymbol{Q}$ is a $n \times n$ positive semidefinite matrix and $\boldsymbol{R}$ is a positive definite value that describes the importance of that variable. Thus, we can picture this as:

Q	Time (s)	R	DC Motor Cost ($)	J (Cost)
3	20	5	5	85
	15		10	95
	10		12	90
	5		20	115

Figure 21: Cost Function Output Example

For the given constants above, our most optimal solution would be DC with \$5 as ***J*** is least. So, rather than predicting which eigenvalues would be best, we can create a cost function of our importance to determine the most optimal solution to minimize cost. For our system we can have a similar equation [7]:

$$J = \int_0^\infty (y^T Q y + u^T R u) dt \tag{108}$$

So, system performance is judged by our initial state, y, and for good performance we want to see how fast our system restores to our desired state, this is determine by the area under the graph, thus we have $\int_0^\infty$, and at times we can see that there are negative values in y, thus, transposed to ensure its positive, y^{T}. ***Q*** is the penalty matrix, with size of states, $n \times n$, and ***R***, a constant as we only have 1 actuator [4]. Thus if we have diagonal matrix ***Q***:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{109}$$

First diagonal variable and second which is, x and $\dot{x}$, we penalize with 1, which means that we are not concerned about cart position and velocity. But, third and fourth diagonal which is θ and $\dot{\theta}$, we penalize it higher as we want it to quickly go into upright position. To solve for ***K***:

$$K = R^{-1} B^T S \tag{110}$$

Our solution is ***S*** by solving the Algebraic Riccati Equation, and ***A*** and ***B*** are our state matrices:

$$A^T S + SA - SBR^{-1} B^T S + Q = 0 \tag{111}$$

While, there are usually multiple outputs for ***S***, only one would have stable eigenvalues, meaning negative. Though, a similar process is available in **MATLAB** using 'lqr()' function [19]:

```
>>K = lqr(A,B,Q,R)
```

Figure 22: Optimal K Solution by Algebraic Riccati

Which will give the optimal ***K*** matrix for our cost function. So, if we use ***Q*** from Eq(109) and ***R*** to be 0.1, or actuator is cheap:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}, \quad R = 0.1 \tag{112}$$

We can use the following **MATLAB** code, with ***A*** and ***B*** from Fig(8):

```
Q = [1 0 0 0;
     0 1 0 0;
     0 0 100 0;
     0 0 0 10];

R = 0.1;

K = lqr(A,B,Q,R);

tspan = 0:0.001:20;
if(c==-1)
    y0=[0; 0; 0; 0];
    [t,y] = ode45(@(t,y)Equation(y,m,M,l,g,b,-K*(y-[4; 0; 0; 0])),tspan,y0);
elseif(c==1)
    y0=[-3; 0; 4; 0];
    [t,y] = ode45(@(t,y)Equation(y,m,M,l,g,b,-K*(y-[1; 0; pi; 0])),tspan,y0);
else
end
```

Figure 23: Simulate Parameters Given Initial and Desired Location

The following code and parameters will yield:

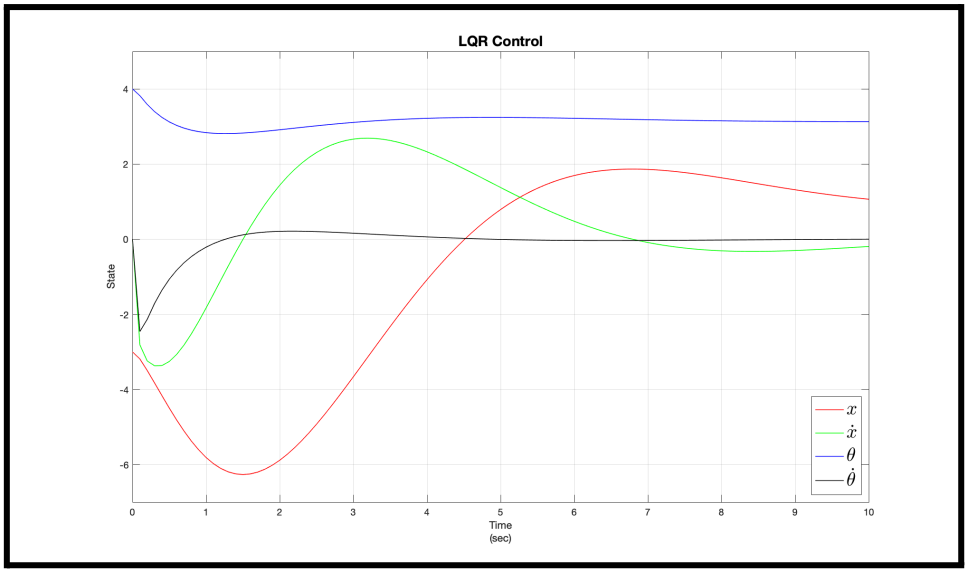

Figure 24: Optimal LQR Control

And, the optimal eigenvalues and the optimal ***K*** gain matrix is found to be:

```
>> K =

  -3.1623   -6.8440   91.8453   31.6148

>> eigs(A-B*K)

ans =

  -9.4102 + 0.0000i
  -3.1791 + 0.0000i
  -1.1568 + 0.3481i
  -1.1568 - 0.3481i
```

*Figure 25: Eigenvalues and **K** gain matrix of Optimal Given Parameters*

Based on simulation results comparing Fig(17) we can see a significant drop in θ and $\dot{\theta}$, this would be due to our ***Q*** specification. We can see that there is a high negative real part for the first eigenvalue, most likely its $\dot{\theta}$. The system can be animated:

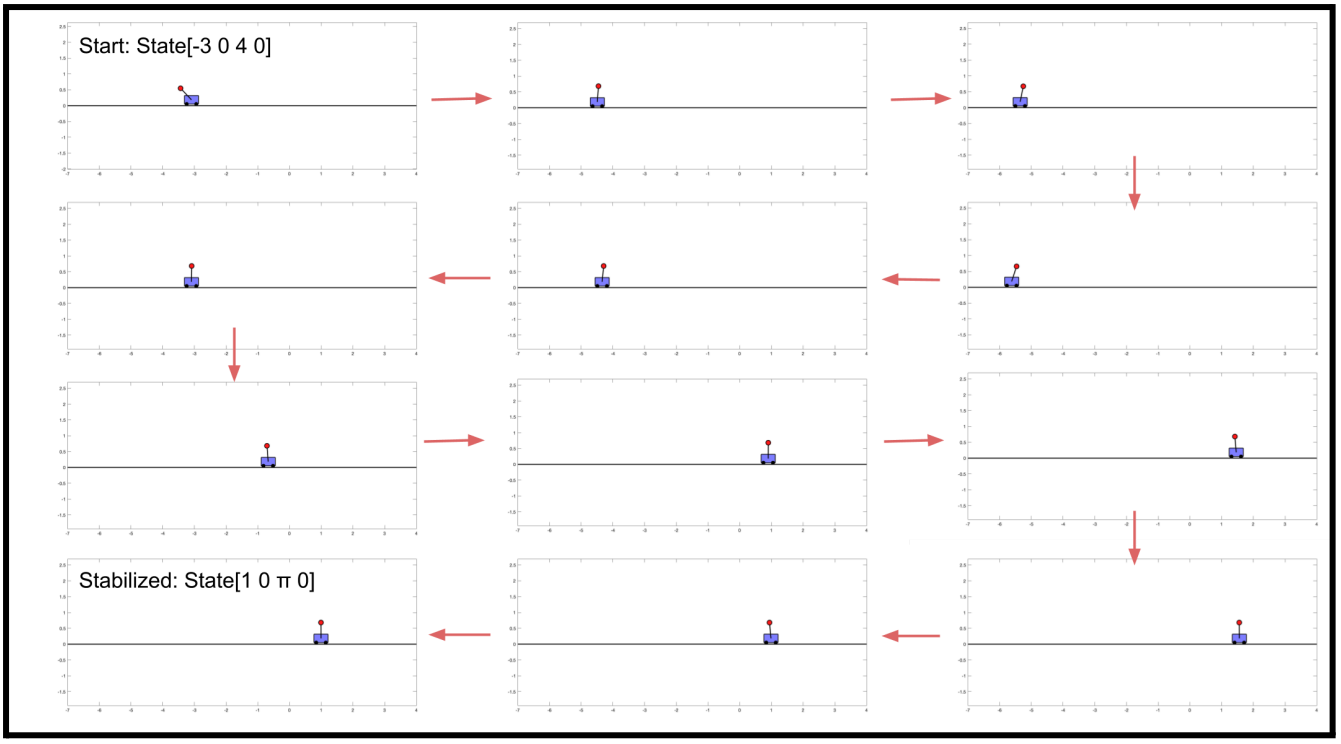

Figure 26: IP System Visualization

As we want to stabilize the pendulum at position $x = 1$, with an initial angle 4 radians. The system would move in a manner that is opposing the pendulum direction until it equilibrates at position 1.

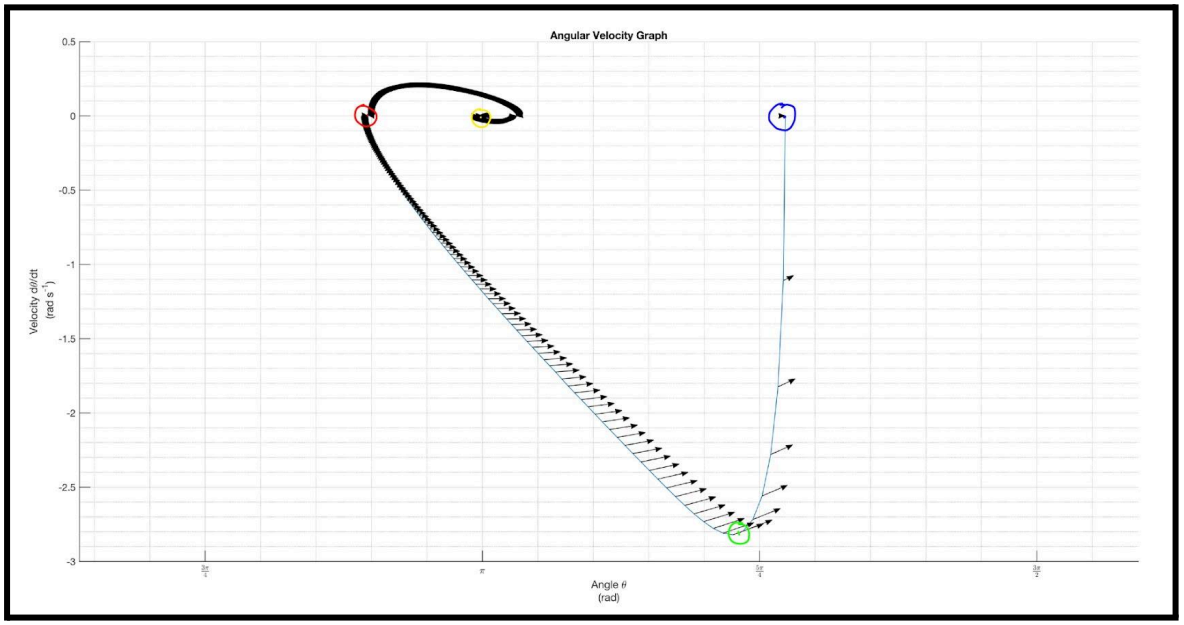

Figure 27: IP Controlled System

Similar to Fig(5), we can see that the controlled IP is a stable system as it equilibrates to the desired equilibrium point. This also proves that the eigenvalues all have negative real parts. Though in contrast to Fig(5) the phase is not symmetrical, expected, as our desired equilibrium point must satisfy $\theta = \pi$. We can also observe that as the system reaches equilibrium the angular velocity will reach 0, satisfying the other equilibrium state. Though this configuration of $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices are only optimal for my system and parameters.

Consider:

$\boldsymbol{Q}$	$\boldsymbol{R}$	Time (t) to reach equilibrium [0 0 π 0] (seconds)			
		x	$\dot{x}$	θ	$\dot{\theta}$
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	1	16.6	17.8	14.3	9.8
$\begin{bmatrix}10&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	1	15.7	17.6	15.8	10.3
$\begin{bmatrix}100&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	1	17.8	13.9	12.4	10.4
$\begin{bmatrix}1&0&0&0\\0&10&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	1	14.9	15	16.2	5.2
$\begin{bmatrix}1&0&0&0\\0&100&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	1	43.2	15.2	19.6	4.6
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&10&0\\0&0&0&1\end{bmatrix}$	1	20.6	17.9	14.4	10.1

$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&100&0\\0&0&0&1\end{bmatrix}$	1	21.4	18.6	14.1	5.5
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&10\end{bmatrix}$	1	16.8	18.0	14.5	5.4
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&100\end{bmatrix}$	1	23.1	19.7	15.8	6
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	10	25.2	19.6	20.3	10.1
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	0.1	15.3	14.1	11.5	8.1

*Figure 28: Variation of **Q** and **R** matrices*

Based on Fig(28), we can observe that the penalizing matrix ***Q*** and ***R*** affects the state variables and control signals. The larger each element is, the more that state is penalized. From data collected, we observe that as the weight is increased for a state variable, the time decreases. Still, if it's raised too much, it increases the time, this is most likely due to the increased aggressiveness of the control design trying to reach the desired point; thus, it will overshoot. The most stabilized parameter is ***R*** with a constant decrease in time as we decrease the parameter. Therefore, it would be best to just alter ***R***, and leave ***Q*** as identity. Though all this will be relative to the IP model's design, as not all models have large control signals. Though we must also consider that these time values are taken not when $y = 0$, as they are still slightly oscillating to infinity, these are estimates.

4 Experimental Result & Discussion

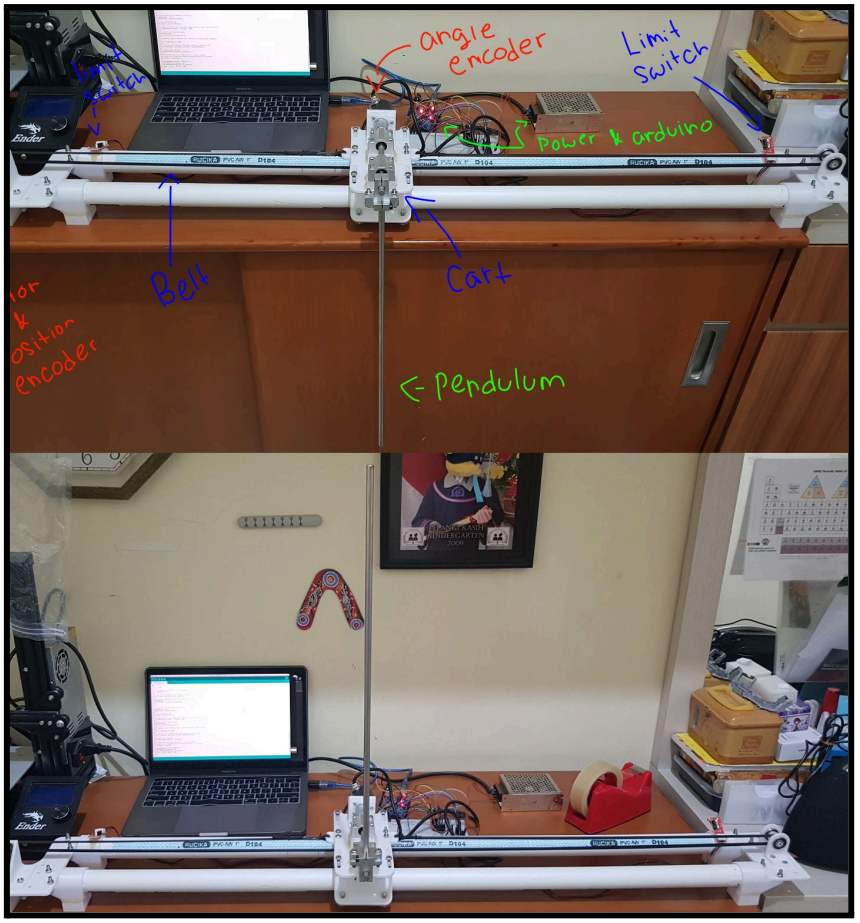

Figure 29:Experimental IP Setup

Our control requires input as a force to stabilize the pendulum, as the Arduino control signal is in PWM signal, the voltage. We must convert voltage to given force. As it is beyond the scope of this paper, DC motor equation and parameters are estimated through a code by *zjor*, alongside the IP code implementation, though with my own gain parameters [20]. The experiment with approximated initial condition of $[0 \quad 0 \quad 4 \quad 0]$ cart and angle of LQR Using constant matrix $\boldsymbol{Q}$, $\boldsymbol{R}$, and gain matrix $\boldsymbol{K}$ found in Fig(25). We can see that compared to the simulation data, the position, x, Fig(30) oscillates about the mean position in order to maintain the pendulum at π, this could've been due to external disturbances in the real-world model. Similarly the angle, θ, Fig(31) is not stationary as it is slightly perturbed about the

equilibrium position; this is most likely due to the friction that was unaccounted for or low frequency noise in the instrument. Though they both have a stabilization phase when the system is experiencing high displacement. Angle is also observed to reach equilibrium phase first, showing the utilization of $\boldsymbol{Q}$, θ weighs the most. (**Note:** As using the serial plotter in Arduino is not reliable, data is exported to excel.) As the means of control, voltage applied to the DC motor was also plotted Fig(32).

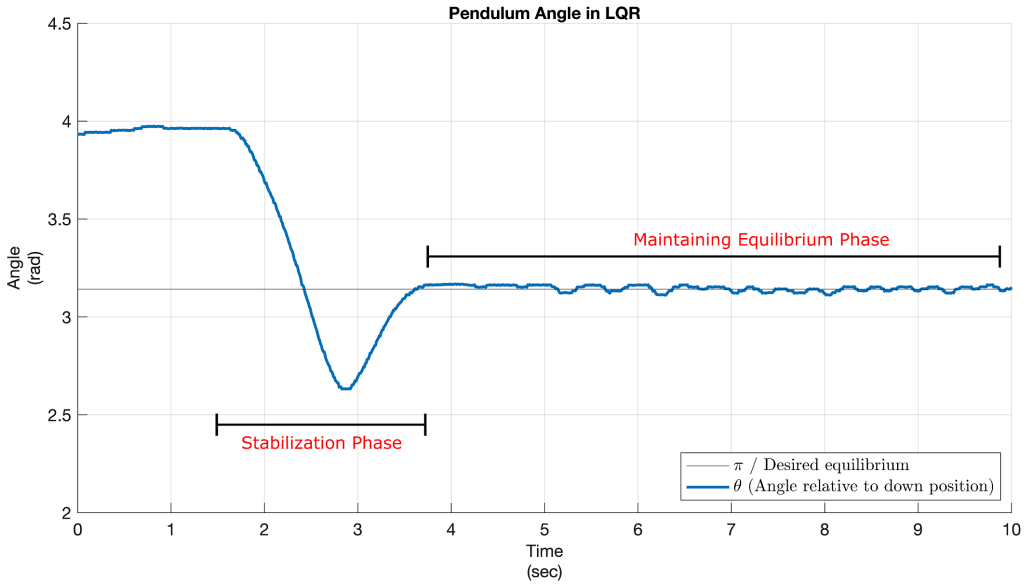

Figure 30:Experimental IP angle data recording through arduino serial monitor and plotted through MATLAB. With equilibrium position at $\theta = \pi$

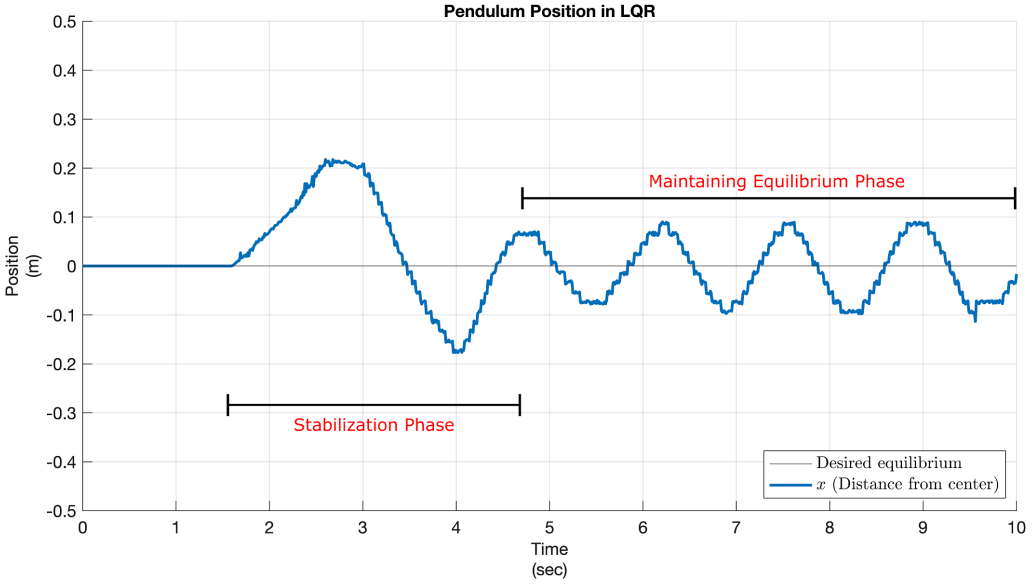

Figure 31:Experimental IP position data recording through arduino serial monitor and plotted through MATLAB. With equilibrium position at $x = 0$

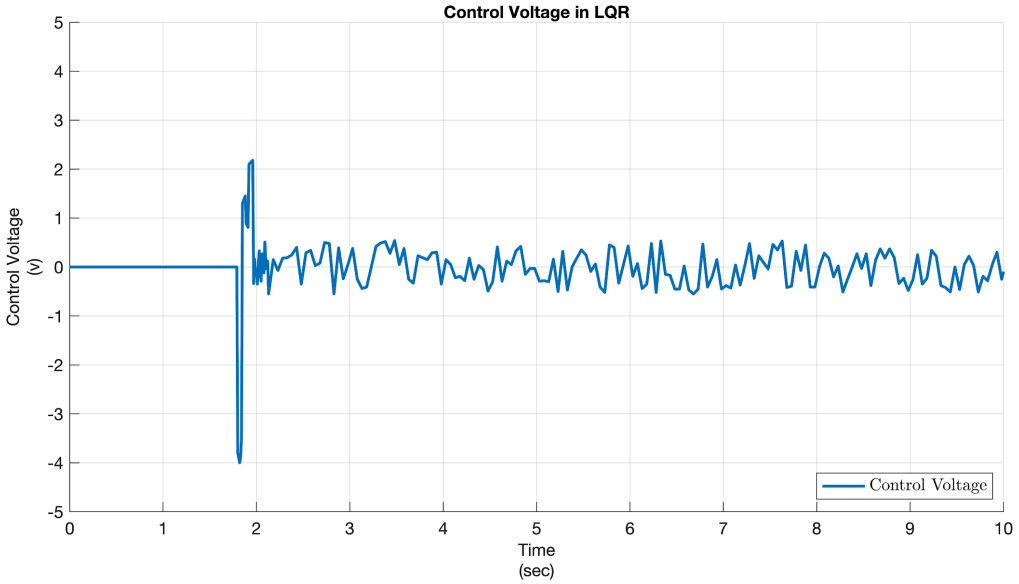

Figure 32:Experimental IP voltage data recording through arduino serial monitor and plotted through MATLAB.

5 Conclusion

This paper demonstrates the effective use of MATLAB simulation through controller design and implementation into a basic IP physical model. Going back to my research question, I have found multiple conditions must be met for an LQR to be effective: eigenvalues must be negative, $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices must be controllable with a rank of 4, and the system must be initially unstable. Though not all negative eigenvalues will yield optimal results; some could be too aggressive or too non-reactive. It will entirely depend on the choice of weighting $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices, as my choice of $\boldsymbol{K}$ is only optimal for my system. Different IP may have other weighing costs. Though from research I suggest only $\boldsymbol{R}$ should be increased for faster performance and keep $\boldsymbol{Q}$ at identity as it may be unstable if it is increased too high.

Though with how good LQR is at controlling systems, it also has limitations. Primarily, having a system that has unmodeled dynamics makes it difficult to create a controller. Secondly, the full state is not always available in experimental systems, in most cases, we need to create a full state estimation or use an observer (LGQ). Lastly, the parameters used to generate the controller are not directly specified, determining them is purely trial and error, finding which fits the system best.

Though this system could be improved by integrating a PID controller, full state estimation, or a Kalman filter to reduce noise and improve performance, due to the time constraints and focus of this paper, I am unable to put more research into it.

6 Bibliography

[1] Kumar, Chandan, et al. “Optimal Controller Design for Inverted Pendulum System Based on LQR Method.” 2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), 2012, www.researchgate.net/publication/261317362_Optimal_controller_design_for_inverted_pendulum_system_based_on_LQR_method.

[2] °Aström Karl Johan, and Richard M. Murray. Feedback Systems: an Introduction for Scientists and Engineers. Princeton University Press, 2008.

[3] Zak, Stanislaw H. Systems and Control. Oxford University Press, 2003.

[4] Brunton, Steven L., and J. Nathan Kutz. Data Driven Science & Engineering. Machine Learning, Dynamical Systems, and Control. Brunton & Kutz., 2017.

[5] Brizard, Alain J. An Introduction to Lagrangian Mechanics. World Scientific, 2015.

[6] Malham, Simon J.A. An Introduction to Lagrangian and Hamiltonian Mechanics. 23 Aug. 2016, www.macs.hw.ac.uk/~simonm/mechanics.pdf.

[7] Kumar, E. Vinodh, and Jovitha Jerome. “Robust LQR Controller Design for Stabilizing and Trajectory Tracking of Inverted Pendulum.” Procedia Engineering, vol. 64, 2013, pp. 169–178., doi:10.1016/j.proeng.2013.09.088.

[8] Strang, W. Gilbert. “21. Eigenvalues and Eigenvectors.” YouTube, YouTube, 24 Sept. 2019, www.youtube.com/watch?v=cdZnhQjJu4I&ab_channel=MITOpenCourseWare.

[9] Strang, W. Gilbert, and Cleve Moler. “The Matrix Exponential.” YouTube, YouTube, 6 May 2016, www.youtube.com/watch?v=LwSk9M5lJx4&ab_channel=MITOpenCourseWare.

[10] Lecture 5: Jacobians - Rice University. www.stat.rice.edu/~dobelman/notes_papers/math/Jacobian.pdf.

[11] “Jacobian Matrix.” Jacobian Matrix - an Overview | ScienceDirect Topics, www.sciencedirect.com/topics/computer-science/jacobian-matrix.

[12] Lecture 1 Linear Quadratic Regulator: Discrete-Time FInite ... stanford.edu/class/ee363/lectures/dlqr.pdf.

[13] Murray, R. M. “Control and Dynamical Systems: Lecture 2 – LQR Control.” Lecture 2 – LQR Control. Lecture 2 – LQR Control, 11 Jan. 2006, CALIFORNIA INSTITUTE OF TECHNOLOGY, CALIFORNIA INSTITUTE OF TECHNOLOGY.

[14] Teschl, Gerald. Ordinary Differential Equations and Dynamical Systems. 1991, www.mat.univie.ac.at/~gerald/ftp/book-ode/ode.pdf.

[15] “Inverted Pendulum: System Modeling.” *Control Tutorials for MATLAB and Simulink - Inverted Pendulum: System Modeling*, ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum§ion=SystemModeling.

[16] *Chapter 3 State Variable Models - Engineering*. www.site.uottawa.ca/~rhabash/ELG4152L305.pdf.

[17] Approximating Functions by Taylor Polynomials. www.math.smith.edu/~rhaas/m114-00/chp4taylor.pdf.

[18] Understanding Poles and Zeros 1 System Poles and Zeros. web.mit.edu/2.14/www/Handouts/PoleZero.pdf.

[19] “Linear Algebra.” Linear Algebra - MATLAB & Simulink, www.mathworks.com/help/matlab/linear-algebra.html?s_tid=CRUX_lftnav.

[20] Zjor. “Zjor/Inverted-Pendulum.” *GitHub*, github.com/zjor/inverted-pendulum.

7 Appendix 1 (Jacobian Matrix)

From Eq(54) & Eq(58):

$$\dot{y} = \begin{bmatrix} y_2 \\ \frac{ul-\mu y_2 l-mlg\cos y_3 \sin y_3+ml^2 y_4^2 \sin y_3}{(M+m)l-ml\cos^2 y_3} \\ y_4 \\ \frac{g\sin y_3(M+m)+\cos y_3\left(\mu y_2+u-mly_4^2 \sin y_3\right)}{(M+m)l-ml\cos^2 y_3} \end{bmatrix} = \begin{bmatrix} f_1(y,u) \\ f_2(y,u) \\ f_3(y,u) \\ f_4(y,u) \end{bmatrix}$$

With the Jacobian matrices in the form of ***A*** and ***B***:

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial y_1}|_{y_0,u_0} & \frac{\partial f_1}{\partial y_2}|_{y_0,u_0} & \frac{\partial f_1}{\partial y_3}|_{y_0,u_0} & \frac{\partial f_1}{\partial y_4}|_{y_0,u_0} \\ \frac{\partial f_2}{\partial y_1}|_{y_0,u_0} & \frac{\partial f_2}{\partial y_2}|_{y_0,u_0} & \frac{\partial f_2}{\partial y_3}|_{y_0,u_0} & \frac{\partial f_2}{\partial y_4}|_{y_0,u_0} \\ \frac{\partial f_3}{\partial y_1}|_{y_0,u_0} & \frac{\partial f_3}{\partial y_2}|_{y_0,u_0} & \frac{\partial f_3}{\partial y_3}|_{y_0,u_0} & \frac{\partial f_3}{\partial y_4}|_{y_0,u_0} \\ \frac{\partial f_4}{\partial y_1}|_{y_0,u_0} & \frac{\partial f_4}{\partial y_2}|_{y_0,u_0} & \frac{\partial f_4}{\partial y_3}|_{y_0,u_0} & \frac{\partial f_4}{\partial y_4}|_{y_0,u_0} \end{bmatrix}, \; B = \begin{bmatrix} \frac{\partial f_1}{\partial u}|_{y_0,u_0} \\ \frac{\partial f_2}{\partial u}|_{y_0,u_0} \\ \frac{\partial f_3}{\partial u}|_{y_0,u_0} \\ \frac{\partial f_4}{\partial u}|_{y_0,u_0} \end{bmatrix}$$

The derivation of entries in matrix ***A*** are as follows (matrix location denoted by $a_{i,j}$):

A$_{1,1}$: not in equation

$$\frac{\partial f_1}{\partial y_1}|_{y_0,u_0} = y_2 = 0$$

A$_{1,2}$: exponent rule

$$\frac{\partial f_1}{\partial y_2}|_{y_0,u_0} = y_2^{1-1} = 1$$

A$_{1,3}$: not in equation

$$\frac{\partial f_1}{\partial y_3}|_{y_0,u_0} = y_2 = 0$$

A$_{1,4}$: not in equation

$$\frac{\partial f_1}{\partial y_4}|_{y_0,u_0} = y_2 = 0$$

A$_{2,1}$: not in equation

$$\frac{\partial f_2}{\partial y_1} = \frac{ul-\mu y_2 l-m\lg\cos y_3 \sin y_3+ml^2 y_4^2 \sin y_3}{(M+m)l-ml\cos^2 y_3} = 0$$

$\mathbf{A}_{2,2}$: quotient rule

$$\begin{aligned}
\frac{\partial f_2}{\partial y_2} &= \frac{ul - \mu y_2 l - m\lg\cos y_3 \sin y_3 + ml^2 y_4^2 \sin y_3}{(M+m)l - ml\cos^2 y_3} \\
(\text{quotient rule}) &= \frac{vu' - uv'}{v^2} \\
&= \frac{\big((M+m)l - ml\cos^2 y_3\big)(-\mu l) - 0}{\big((M+m)l - ml\cos^2 y_3\big)^2} \\
&= \frac{-\mu l}{(M+m)l - ml\cos^2 y_3} \\
(\text{evaluated at fixed point}) &= \frac{-\mu l}{(M+m)l - ml\cos^2(0)} \\
&= \frac{-\mu l}{(M+m)l - ml(1)} \\
(\text{cancel } l \,\&\, ml) &= \frac{-\mu l}{Ml + ml - ml(1)} \\
&= \frac{-\mu}{M}
\end{aligned}$$

$\mathbf{A}_{2,3}$: quotient rule & product rule

$$\begin{aligned}
\frac{\partial f_2}{\partial y_3} &= \frac{ul - \mu y_2 l - m\lg\cos y_3 \sin y_3 + ml^2 y_4^2 \sin y_3}{(M+m)l - ml\cos^2 y_3} \\
(\text{quotient rule}) &= \frac{vu' - uv'}{v^2} \\
(\text{product rule}) &= \frac{\big((M+m)l - ml\cos^2 y_3\big)\big(-m\lg\cos 2y_3 + ml^2 y_4^2 \cos y_3\big) - \big(ul - \mu y_2 l - m\lg\cos y_3 \sin y_3 + ml^2 y_4^2 \sin y_3\big)\big((M+m)l - ml(-\sin 2y_3)\big)}{\big((M+m)l - ml\cos^2 y_3\big)^2} \\
(\text{evaluated at fixed point}) &= \frac{-m\lg(1) + 0 - 0}{(M+m)l - ml\cos^2 y_3} \\
&= \frac{-m\lg}{Ml + ml - ml} \\
&= \frac{-m\lg}{Ml} \\
&= \frac{-mg}{M}
\end{aligned}$$

$\mathbf{A}_{2,4}$: exponent rule & quotient rule

$$\begin{aligned}
\frac{\partial f_2}{\partial y_4} &= \frac{ul - \mu y_2 l - m\lg\cos y_3 \sin y_3 + ml^2 y_4^2 \sin y_3}{(M+m)l - ml\cos^2 y_3} \\
(\text{quotient rule}) &= \frac{vu' - uv'}{v^2} \\
(\text{exponent rule}) &= \frac{\big((M+m)l - ml\cos^2 y_3\big)\big(ml^2 y_4^{2-1}\sin y_3\big) - (0)}{(M+m)l - ml\cos^2 y_3} \\
(\text{evaluated at fixed point}) &= \big(\big(ml^2(0)(0)\big)\big) \\
&= 0
\end{aligned}$$

$\mathbf{A}_{3,1}$: not in equation

$$\frac{\partial f_3}{\partial y_1}\Big|_{y_0,u_0} = y_4 = 0$$

$\mathbf{A}_{3,2}$: not in equation

$$\frac{\partial f_3}{\partial y_2}\Big|_{y_0,u_0} = y_4 = 0$$

$\mathbf{A}_{3,3}$: not in equation

$$\frac{\partial f_3}{\partial y_3}\Big|_{y_0,u_0} = y_4 = 0$$

$\mathbf{A}_{3,4}$: exponent rule

$$\frac{\partial f_3}{\partial y_4}\Big|_{y_0,u_0} = y_4^{1-1} = 1$$

$\mathbf{A}_{4,1}$: not in equation

$$\frac{\partial f_4}{\partial y_1} = \frac{g\sin y_3(M+m)+\cos y_3\left(\mu y_2+u-mly_4^2\sin y_3\right)}{(M+m)l-ml\cos^2 y_3} = 0$$

$\mathbf{A}_{4,2}$: exponent rule & quotient rule

$$\begin{aligned}
\frac{\partial f_4}{\partial y_2} &= \frac{g\sin y_3(M+m)+\cos y_3\left(\mu y_2+u-mly_4^2\sin y_3\right)}{(M+m)l-ml\cos^2 y_3} \\
&= \frac{vu'-uv'}{v^2} \\
&= \frac{\left((M+m)l-ml\cos^2 y_3\right)\left(\cos y_3\left(\mu+u-mly_4^2\sin y_3\right)\right)-0}{\left((M+m)l-ml\cos^2 y_3\right)^2} \\
&= \frac{\left(\cos y_3\left(\mu+u-mly_4^2\sin y_3\right)\right)-0}{(M+m)l-ml\cos^2 y_3} \\
&= \frac{1(\mu)}{Ml+ml-ml} \\
&= \frac{\mu}{Ml}
\end{aligned}$$

$\mathbf{A}_{4,3}$: quotient rule & product rule

$$\begin{aligned}
\frac{\partial f_4}{\partial y_3} &= \frac{g\sin y_3(M+m)+\cos y_3\left(\mu y_2+u-mly_4^2\sin y_3\right)}{(M+m)l-ml\cos^2 y_3} \\
&= \frac{vu'-uv'}{v^2} \\
\text{(quotient rule \& evaluate at fixed point)} &= \frac{\left((M+m)l-ml\cos^2 y_3\right)\left(g\cos y_3(M+m)+0\right)-\left(g\sin y_3(M+m)+\cos y_3\left(\mu y_2+u-mly_4^2\sin y_3\right)\right)\left((M+m)l-ml(-\sin 2y_3)\right)}{\left((M+m)l-ml\cos^2 y_3\right)^2} \\
&= \frac{(M+m)g}{Ml}
\end{aligned}$$

$\mathbf{A}_{4,4}$: exponent rule

$$\begin{aligned}\frac{\partial f_4}{\partial y_4} &= \frac{g \sin y_3 (M+m) + \cos y_3 \left(\mu y_2 + u - m l y_4^2 \sin y_3\right)}{(M+m)l - ml\cos^2 y_3} \\ &= \frac{vu' - uv'}{v^2} \\ &= \frac{0 - 0}{v^2} \\ &= 0\end{aligned}$$

$\mathbf{B}_{1,1}$: not in equation

$$\frac{\partial f_1}{\partial u} = y_2 = 0$$

$\mathbf{B}_{2,1}$: quotient rule & exponent rule

$$\begin{aligned}\frac{\partial f_2}{\partial u} &= \frac{ul - \mu y_2 l - mlg\cos y_3 \sin y_3 + ml^2 y_4^2 \sin y_3}{(M+m)l - ml\cos^2 y_3} \\ &= \frac{vu' - uv'}{v^2} \\ &= \frac{\left((M+m)l - ml\cos^2 y_3\right)\left(u^{1-1}l\right) - 0}{\left((M+m)l - ml\cos^2 y_3\right)^2} \\ &= \frac{l}{(M+m)l - ml\cos^2 y_3} \\ &= \frac{l}{Ml} \\ &= \frac{1}{M}\end{aligned}$$

$\mathbf{B}_{3,1}$: not in equation

$$\frac{\partial f_3}{\partial u} = y_4 = 0$$

$\mathbf{B}_{4,1}$: quotient rule & exponent rule

$$
\begin{aligned}
\frac{\partial f_4}{\partial u} &= \frac{g \sin y_3 (M+m) + \cos y_3 \left(\mu y_2 + u - m l y_4^2 \sin y_3 \right)}{(M+m)l - ml \cos^2 y_3} \\
&= \frac{vu' - uv'}{v^2} \\
&= \frac{\left((M+m)l - ml \cos^2 y_3 \right) \left(u^{1-1} \cos y_3 \right) - 0}{\left((M+m)l - ml \cos^2 y_3 \right)^2} \\
&= \frac{\cos y_3}{(M+m)l - ml \cos^2 y_3} \\
&= \frac{1}{(M+m)l - ml} \\
&= \frac{1}{Ml}
\end{aligned}
$$