ENPM 667 : Control of Robotic Systems

Project -1 Report

Authors

Hitesh Kyatham - 120275364

Gautam Sreenarayanan Nair - 119543092

Submitted on: 11 - 27 - 2024

Contents

List of Figures **iii**

1 Introduction **2**

1.1 Purpose 2

1.2 Background 2

1.3 Preliminaries 2

1.3.1 Simple Swing-up Strategy 9

1.4 Energy Control 10

2 Discussions and Results **14**

2.1 Single-swing double-switch (SSDS) behavior 15

2.1.1 Simulation of SSDS behavior 17

2.2 Single-swing triple-switch (SSTS) behavior 18

2.2.1 Simulation of SSTS behavior 23

2.3 Multi-swing behavior 24

2.3.1 Simulation of Multi-swing behavior 27

2.4 The general case 28

2.4.1 Simulation of five swing behavior 30

2.5 Minimum time strategies 31

2.6 Generalizations 34

2.6.1 General dynamical system 34

2.6.2 Two pendulums on a cart with simulation 38

3 Conclusion **42**

References **43**

Appendix **44**

4 Appendix **44**

4.1 Control Law 44

4.2 Lyapunov function 44

4.3 Pontryagins Maximum Principle 45

4.4 Bang-Bang control 45

4.4.1 Why Does PMP Lead to Bang-Bang Control? 45

4.5 MATLAB Simulation Code 47

4.6 Code for Single Swing Double Switch (SSDS) 47

4.7 Code for Single Swing Triple Switch (SSTS) 51

4.8 Code for Multi-Swing Behavior 54

4.9 Code for Five Swing Behavior 57
4.10 Code for Minimum time strategies 60
4.11 Code for Two pendulums on a Cart 63
4.12 Code for the Contour plot of Energy of the pendulum 68
4.13 Code for numerical solution for relation between k and n value 69

List of Figures

1 Pendulum basic setup . . . 3
2 Pendulum basic diagram . . . 5
3 Energy Plot . . . 7
4 Geometric Illustration for Simple Swing up Strategy . . . 9
5 (a) Goal Position with Energy $E_0 = 0$. (b) Position with Energy $E = -2mgl$ . . . 15
6 Single swing double switch Simulation Output . . . 17
7 Single Swing Triple Switch Simulation Results. . . . 19
8 Triangle formed from the input acceleration and the acceleration due to gravity 20
9 Diagram to explain the single swing, triple switch behavior. . . . 24
10 Multi Swing Multi Switch Diagram . . . 25
11 Double Swing Multi Switch Simulation Results . . . 28
12 The value of n for each value of k varying from 1 to 10 . . . 30
13 Simulation Results for Five Swing Behavior . . . 31
14 Minimum Time strategy applied to n=2.1 . . . 33
15 Minimum Time strategy applied to n=5 . . . 33
16 Two Pendulums on a Cart . . . 39
17 Two pendulums on a Cart Simulation Output . . . 41

Abstract

This report presents a comprehensive replication and in-depth explanation of the results from the paper titled "Swinging up a Pendulum by Energy Control" by Karl J. Åström and Katsuhisa Furuta, two prominent figures in control theory. The study revisits energy-based strategies for swinging up a pendulum, a classic problem in nonlinear control. The dynamics of the inverted pendulum are rigorously analyzed, and the system's energy is mathematically defined to formulate control strategies. The critical influence of the ratio of maximum pivot acceleration to gravitational acceleration on swing-up behavior is discussed and mathematically validated. Various swing-up behaviors, categorized by the number of swings and control input switches, are investigated. The results are implemented in MATLAB and compared with those presented in the original paper. The report further explores minimum-time strategies and generalizations of energy control, extending the solution to general dynamic systems and to the control of two pendulums on a cart. This work provides a detailed examination of energy-based control, reinforcing its importance as a reliable control strategy.

1 Introduction

1.1 Purpose

This report is a submission for Project-1 of the course ENPM 667: Control of Robotic Systems at the University of Maryland. This project requires that a paper be chosen, fully studied, and converted to a report format with simpler explanations. The paper selected for this project is "Swinging up a pendulum by energy control" [1]. This report is written to be self-contained and includes essential background information on several topics discussed in detail in the Appendix Section.

1.2 Background

The inverted pendulum is a classic problem utilized in dynamics and control theories often due to their non-linearity, associated instability, and ease of experimental testability. Initial studies in the 1960s and 70s focussed on demonstrating how linear control techniques could stabilize unstable systems using an inverted pendulum. Later on, the use of an inverted pendulum evolved into becoming a testbed for exploring control strategies like nonlinear control, task-oriented control, hybrid control, and chaotic systems control. In addition to utilizing inverted pendulum models for control theory education and research, it has real-world significance for systems like self-balancing robots that require stabilization [2]. It can also be used to model control problems associated with the initial stages of a rocket or missile launch when there is no aerodynamic stability as the airspeed is too small for maintaining vertical orientation [3].

The authors in this paper [1] have done an investigation on energy control strategies for swinging up the pendulum without considering the position of the pivot and its velocity. The key finding is that the global behavior of the swing-up can be fully characterized by the ratio n which is a value that is equal to the maximum acceleration of the pivot to the acceleration due to gravity. This study also sheds light on the robustness of the minimum-time swing-up with respect to energy overshoot. In addition to these, the authors have outlined potential applications to systems involving multiple pendulums, showcasing the adaptability of energy control methods across different experimental setups.

1.3 Preliminaries

Let us discuss the setup and derive the preliminary equations that the authors talk about in the paper. Refer the Figure 1 given. A single pendulum at two different positions are given in the figure. The goal is to swing up the pendulum to upright position. The reference position is the upright position of the pendulum and it is considered as zero degree. The pivot point is the point about which the pendulum pivots. If we consider the motion of pendulum as a circle, the

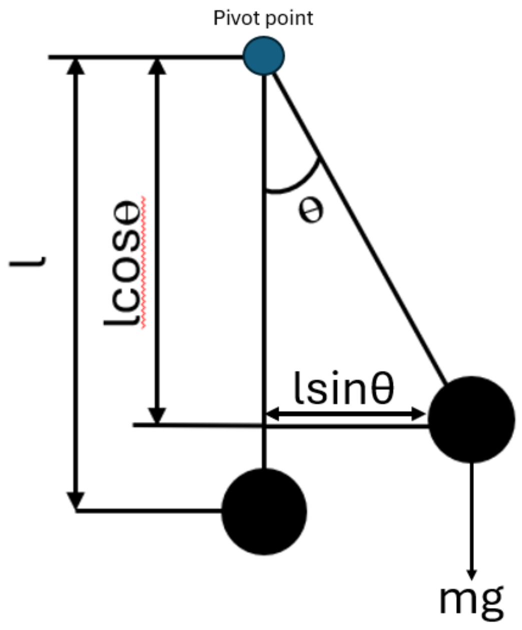

Figure 1: Pendulum basic setup

pivot point would be the center of the circle. There are two state variables in this setup. The angle θ, and the angular velocity $\dot{\theta}$. Assumptions considered in this setup are:

1. No friction or aerodynamic forces are acting on the system.
2. The pendulum is a rigid body that does not change its shape or structure.
3. The velocity of the pivot is not bounded and could take any value.

Definition of Variables
The basic definition of the variables used is given below.

m :	Mass of the pendulum (kg)
l :	Length of distance (m) from the pivot point to the center of mass of the pendulum
J :	Moment of inertia (kg·m^2) with respect to the pivot point
θ :	Angle (radians) between the vertical and the pendulum, positive in the clockwise direction
g :	Acceleration due to gravity (m/s^2)
u :	Acceleration of the pivot (m/s^2), positive in the direction of the positive x-axis

The equation for the motion of the pendulum as given in the paper is:

$$J\ddot{\theta} - mgl\sin(\theta) + ul\cos(\theta) = 0$$

Let's derive this equation.
The torque on the pendulum due to gravity is given by:

$$\tau = \text{Force} \times \text{distance from current position to downward position}$$

$$\tau = \text{Force} \times \text{l} \times \sin(\theta)$$

Refer to Figure 1 for reference of $\text{l} \times \sin(\theta)$.

Thus, we have:

$$\tau = mgl\sin(\theta) \tag{1}$$

If we assume the pendulum as a point mass, we can take the moment of inertia as:

$$J = ml^2$$

From Newton's second law, we know that:

$$\tau = J\ddot{\theta} \tag{2}$$

Equating (1) and (2), we get:

$$J\ddot{\theta} = mgl\sin(\theta) \tag{3}$$

Adding the force that contributes to the horizontal acceleration of the pivot,

$$F_u = mu$$

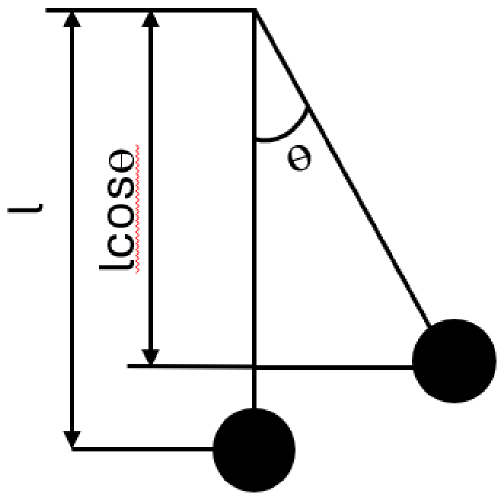

Figure 2: Pendulum basic diagram

As τ = Force $\times$ length, and the length here is $l\cos(\theta)$. Refer Figure 2

Therefore, $\quad \tau = -mul\cos(\theta)$

Equating the forces acting on the system, we get:

$$J\ddot{\theta} = mgl\sin(\theta) - mul\cos(\theta)$$

Therefore, we have:

$$J\ddot{\theta} - mgl\sin(\theta) + mul\cos(\theta) = 0$$

(This is the same as equation (1) in the paper).
The equation for the energy of the uncontrolled pendulum(u=0) as given in the paper is:

$$E = \frac{1}{2}J\dot{\theta}^2 + mgl(\cos\theta - 1)$$

Let's derive this equation. The total energy E of a system is given by sum of potential energy and kinetic energy.
The kinetic energy(K.E) of any system is given by the sum of kinetic energy due to angular

motion and kinetic energy due to linear motion.

$$K.E = \frac{1}{2}J\dot{\theta}^2 + \frac{1}{2}mu^2$$

Since the system is uncontrolled(u=0), the input signal to the pendulum is zero. Thus the kinetic energy is

$$K.E = \frac{1}{2}J\dot{\theta}^2$$

Considering the upright position as the home position with zero potential energy, the total potential energy(P.E) of the system is given by:

$$P.E = mgh$$

where h is the height of the pendulum from the home position.
As shown in the Figure 2, the pendulum position from the home position is $l - l\cos\theta$

$$P.E = -mg(l - l\cos\theta)$$

Since the energy is decreasing a negative sign is added to the above equation.

$$P.E = mgl(\cos\theta - 1)$$

Thus the total energy E of the pendulum is given by:

$$E = K.E + P.E$$

$$E = \frac{1}{2}J\dot{\theta}^2 + mgl(\cos\theta - 1)$$

(This is the same as equation (2) in the paper).
The contour plot of the energy of the system as given in the above equation is plotted in Figure 3 as we were curious to visualize the same. It is to be noted that the reference point here is the upright position, and at that position, the Energy is zero. The angles that correspond to the upright position are 0, π, $-\pi$, and so on. The MATLAB code utilized for the same is available in the Appendix Section.
Let us assume the maximum acceleration of the pivot to be **u** and the ratio $\frac{u}{g}$ to be **n**

$$u_max = max|u| = ng$$

(This is the same as equation (3) in the paper).
In the paper, normalized variables are introduced to transform the aforementioned equations into their dimensionless forms.

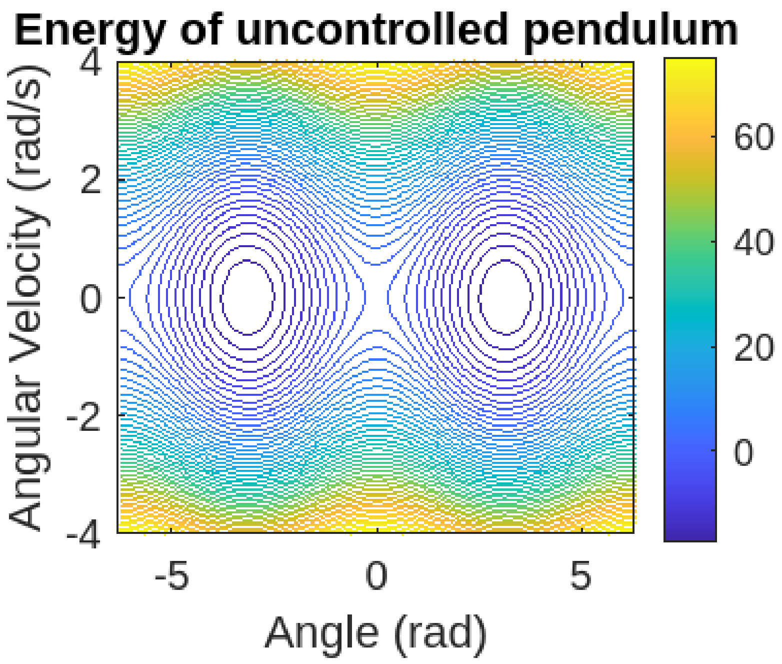

Figure 3: Energy Plot

The normalized variables are $\omega_0 = \sqrt{\frac{mgl}{J}}$, $\tau = \sqrt{\frac{mgl}{J}}t = \omega_0 t$ and $\nu = \frac{u}{g}$.

$$\tau = \omega_0 t$$

Squaring on both sides,

$$\tau^2 = (\omega_0 t)^2$$
$$\tau^2 = \omega_0^2 t^2$$

Applying derivative on both sides,

$$2\tau\dot{\tau} = \omega_0^2 2t\dot{t}$$
$$\tau\dot{\tau} = \omega_0^2 t\dot{t}$$

Applying derivative on both sides again,

$$\ddot{\tau} = \omega_0^2 \ddot{t}$$

Normalizing the earlier equations, starting from equation (1):

$$J\ddot{\theta} - mgl\sin(\theta) + mul\cos(\theta) = 0$$

Dividing the above equation with mgl on both sides:

$$\frac{J\ddot{\theta}}{mgl} - \frac{mgl\sin(\theta)}{mgl} + \frac{mul\cos(\theta)}{mgl} = \frac{0}{mgl}$$

$$\frac{J\ddot{\theta}}{mgl} - \sin(\theta) + \frac{u\cos(\theta)}{g} = 0$$

$$\frac{J}{mgl}\frac{d^2\theta}{dt^2} - \sin(\theta) + \frac{u\cos(\theta)}{g} = 0$$

$$\frac{1}{\omega_0^2}\frac{d^2\theta}{dt^2} - \sin(\theta) + \frac{u\cos(\theta)}{g} = 0$$

From the earlier normalized variables $\omega_0^2 = \frac{mgl}{J}$ and $\omega_0^2 dt^2 = d\tau^2$. Thus replacing $\omega_0^2 dt^2$ with $d\tau^2$ and $\frac{u}{g}$ with ν we get

$$\frac{d^2\theta}{d\tau^2} - \sin(\theta) + \nu\cos(\theta) = 0$$

Similarly, normalizing the energy of the uncontrolled system.

$$E = \frac{1}{2}J\dot{\theta}^2 + mgl(\cos\theta - 1)$$

Dividing the above equation with mgl on both sides,

$$\frac{E}{mgl} = \frac{1}{2}J\frac{d^2\theta}{mgldt^2} + \frac{mgl(\cos\theta - 1)}{mgl}$$

$$\frac{E}{mgl} = \frac{1}{2}\frac{d^2\theta}{\omega_0^2 dt^2} + (\cos\theta - 1)$$

Replacing $\omega_0^2 dt^2$ with $d\tau^2$ we get

$$\frac{E}{mgl} = \frac{1}{2}\frac{d^2\theta}{d\tau^2} + (\cos\theta - 1)$$

Thus the normalized energy E_n is given by

$$E_n = \frac{E}{mgl} = \frac{1}{2}\frac{d^2\theta}{d\tau^2} + (\cos\theta - 1)$$

(This is the same as equation (4) in the paper).

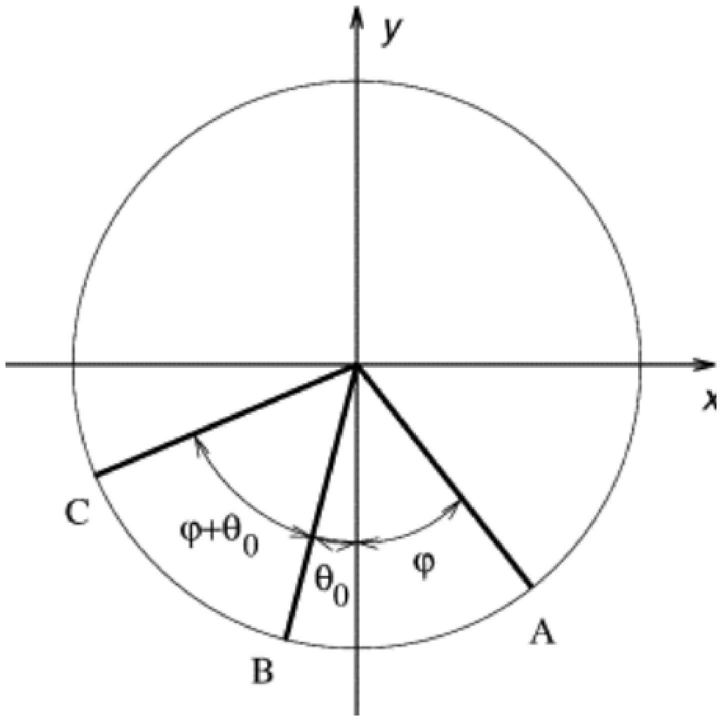

Figure 4: Geometric Illustration for Simple Swing up Strategy

1.3.1 Simple Swing-up Strategy

The system equations are formulated so that the energy in the system is zero when the pendulum is in the upright position. Consequently, we can bring the pendulum to the upright position by reducing its energy to zero. As shown in Figure 2, the potential energy (with no kinetic energy) of the system at the downward position is $-2mgl$. This value arises because the center of mass of the pendulum is located a distance of $2l$ below the upright position, resulting in a potential energy of $-2mgl$. The energy is negative because the upright position is defined as the reference point with zero energy. The same is mathematically proven in the Discussions and Results section of this document.

The authors discuss the simple strategy for swinging up the pendulum. Here they consider that the pendulum starts at point A as shown in figure 4. The pivot is accelerated with a maximum acceleration of ng to the right. To an observer moving along with the pivot, the pendulum seems to experience a force (the authors call it felt gravity) pulling it into a new direction tilting away from the vertical. The reason for this experienced force is an inertial force acting in the opposite direction of acceleration. The angle of this tilt θ is $\arctan(n)$.

As

$$\tan(\theta) = \frac{\text{horizontal force}}{\text{vertical force}} = \frac{ng}{g} = n.$$

Therefore,

$$\theta = \arctan(n).$$

The magnitude of this experienced force is $g\sqrt{1+n^2}$.
This is because the experienced force (Resultant force) is given by:

$$\text{Resultant force} = \sqrt{(\text{horizontal force})^2 + (\text{vertical force})^2}.$$

Substituting the values:

$$\text{Resultant force} = \sqrt{(ng)^2 + g^2} = g\sqrt{n^2+1}.$$

This resultant force causes the pendulum to swing symmetrically around OB. The velocity becomes zero when it reaches the point C, after which it changes the direction of motion. The pendulum has the highest potential energy at this point with no kinetic energy left in it.

The angle at C is $\phi + 2\theta_0$. This is because the motion is symmetric about OB, which is the momentary point about which the observer at the pivot experiences gravity (or resultant force). This method is a simple way to pump energy into the system by moving the pivot. Further strategies discussed in this paper are based on this simple approach.

1.4 Energy Control

To perform energy control it is necessary to understand how the energy is influenced by the acceleration of the pivot. This can be derived from the energy E from equation (2).

$$E = \frac{1}{2}J\dot{\theta}^2 + mgl(\cos\theta - 1)$$

Applying derivative with respect to time on both sides.

$$\frac{dE}{dt} = \frac{d}{dt}(\frac{1}{2}J\dot{\theta}^2 + mgl(\cos\theta - 1))$$

$$\frac{dE}{dt} = \frac{d}{dt}\frac{1}{2}J\dot{\theta}^2 + \frac{d}{dt}mgl(\cos\theta - 1)$$

$$\frac{dE}{dt} = \frac{1}{2}J\frac{d}{dt}\dot{\theta}^2 + mgl(\frac{d}{dt}\cos\theta - \frac{d}{dt}1)$$

$$\frac{dE}{dt} = \frac{1}{2}J \cdot 2\dot{\theta}\ddot{\theta} + mgl(-\sin\theta\dot{\theta} - 0)$$

$$\frac{dE}{dt} = J\dot{\theta}\ddot{\theta} - mgl\dot{\theta}\sin\theta$$

UNIVERSITY OF MARYLAND

Taking $\dot{\theta}$ common we get,

$$\frac{dE}{dt} = \dot{\theta}(J\ddot{\theta} - mgl\sin\theta)$$

From equation (1) we have

$$J\ddot{\theta} - mgl\sin(\theta) + mul\cos(\theta) = 0$$

$$J\ddot{\theta} - mgl\sin(\theta) = -mul\cos(\theta)$$

Substituting this equation in the above derivative of energy equation we get,

$$\frac{dE}{dt} = J\dot{\theta}\ddot{\theta} - mgl\dot{\theta}\sin\theta = -mul\dot{\theta}\cos(\theta)$$

(This is the same as equation (5) in the paper).

The Lyapunov function chosen in the paper is $V = \frac{(E-E_0)^2}{2}$ where E is the energy of the system and E_0 is the target energy.

Let us validate if the above Lyapunov function satisfies the two required conditions,

Positive Definiteness:

- $V(x) > 0 \quad \forall \quad x \neq x_e$ (x_e is equilibrium point)

Since $(E - E_0)^2$ is a squared term, it should always be greater than zero $\forall$ $E \neq E_0$. Implies $V > 0$ $\forall$ $E \neq E_0$.

- $V(x) = 0 \quad at \quad x = x_e$

Since our target is the equilibrium point, $E = E_0$.

$$V = \frac{(E - E_0)^2}{2}$$

$$V = \frac{(E_0 - E_0)^2}{2}$$

$$V = 0$$

Negative derivative Definiteness:

Applying the derivative with respect to time for the earlier Lyapunov function V.

$$\frac{dV}{dt} = \frac{d}{dt}\frac{(E - E_0)^2}{2}$$

$$\frac{dV}{dt} = \frac{2(E - E_0)}{2}\frac{d(E - E_0)}{dt}$$

$$\frac{dV}{dt} = (E - E_0)\frac{dE}{dt}$$

Substituting the value of $\frac{dE}{dt}$ from equation (5),

$$\frac{dV}{dt} = (E - E_0) \cdot -mul\dot{\theta}\cos(\theta)$$

$$\frac{dV}{dt} = -mul(E - E_0)\dot{\theta}\cos(\theta)$$

Substituting the value of u in the control law from equation (6)

$$\frac{dV}{dt} = -m(k(E - E_0)\dot{\theta}\cos\theta)l(E - E_0)\dot{\theta}\cos(\theta)$$

$$\frac{dV}{dt} = -mukl((E - E_0)\dot{\theta}\cos(\theta))^2$$

The Lyapunov function decreases as long as $\dot{\theta} \neq 0$ (no angular velocity) and $\cos\theta \neq 0$. $\cos\theta$ would be equal to zero only at $\theta = \pm\frac{\pi}{2}$. At this condition, the pendulum is horizontal to the ground and the gravity will change the pendulum position, hence even if the control input is zero at this point, the change due to gravity will induce control input at nearby positions of θ.

The control law chosen in the paper is $u = k(E - E_0)\dot{\theta}\cos\theta$ where k is the gain parameter and u is the control input parameter. ***(This is the same as equation (6) in the paper).***

The goal of control law is to reduce the energy difference between actual(E) and the target(E_0). The equation (6) drives E to E_0. The authors proposed another control law to change the energy as fast as possible. This is done by changing the magnitude of the control signal as large as possible.

The proposed control law is:

$$u = ngsign((E - E_0)\dot{\theta}\cos\theta)$$

(This is the same as equation (7) in the paper).

where sign here is a function defined as:

$$sign(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$$

In the equation (7) the term ng controls the magnitude and the sign function controls the control input. Problem with the equation (7) is rapid oscillations of control signal around the desired, value because of the discontinuity of the sign function, also known as the chattering issue.

To avoid this chattering issue authors proposed another control law given as:

$$u = sat_{ng}(k(E - E_0)sign(\dot{\theta}\cos\theta)$$

(This is the same as equation (8) in the paper).

where sat_{ng} is a function defined as:

$$sat_{L,U}(x) = \begin{cases} L, & \text{if } x < L \\ x, & \text{if } L \leq x \leq U \\ U, & \text{if } x > U \end{cases}$$

2 Discussions and Results

In this section, we will discuss the energy control strategies the authors have opted for to bring the pendulum to an upright position. The strategy is then generalized in the end, and simulation results of two pendulums on a cart are also discussed.

The primary control strategy revolves around the energy of the system, with the goal of bringing the energy to the desired level, which is $E_0 = 0$. This represents the energy defined at the upright position. The minimum energy of the system at the downward position is -2mgl.

The goal position is assumed to be with zero energy and thus the home position defined earlier would be at $2l$ distance from the goal position, as shown in the figure 5. Since the distance from the desired position is $2l$ the potential energy of the pendulum is given by

$$P.E = mgh$$

Here h is $-2l$ as the position is in negative direction of Y-axis.

$$P.E = -2mgl$$

Since the pendulum is at rest, the velocity is zero and thus the kinetic energy is also zero. Total energy is defined as the sum of potential energy and kinetic energy.

$$E = P.E + K.E$$

$$E = P.E + 0$$

$$E = P.E$$

$$E = -2mgl$$

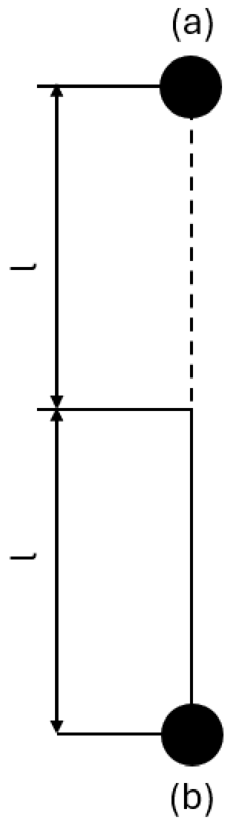

Figure 5: (a) Goal Position with Energy $E_0 = 0$. (b) Position with Energy $E = -2mgl$

The challenge with this strategy is that the equilibrium is an unstable saddle, meaning that even slight disturbances can lead to a loss of equilibrium. The authors have noted that, due to this instability, it is necessary to employ other strategies to capture the system as it approaches equilibrium and stabilize it at that point. While the authors have pointed to suitable hybrid strategies, they have not discussed them in detail as it is outside the scope of the paper. The energy control strategies are discussed below, classified based on two criteria: the number of swings required before reaching the upright position and the control signal u used during the swing-up. The classifications and results obtained from our own analysis are discussed and contrasted with the findings in the paper below.

2.1 Single-swing double-switch (SSDS) behavior

Single-swing double-switch strategy means that there is a single monotonic increase or decrease of pendulum angle and the control signal is switched twice. From zero to maximum and then

UNIVERSITY OF MARYLAND

from maximum to zero. Thus the name Single-Swing Double-Switch (SSDS). In this strategy the maximum control signal gives maximum acceleration to the pendulum and then it is switched off to zero before the pendulum reaches to the horizontal position. The same is visible in the simulation output given in Figure 6.

The energy supplied to a mass when it is moved from a to b by a force F is given by the work done principle as:

$$W_{ab} = \int_a^b F dx$$

(This is the same as equation (9) in the paper).

For the pendulum to reach the upright position in a single swing and double switch the necessary energy has to be delivered to the pendulum before the pendulum reaches horizontal position. Since the maximum input u is ng and the distance moved by pendulum by the time the pendulum reaches horizontal position is l. From earlier work done principle, the energy supplied to the pendulum is:

$$W = mngl$$

As discussed earlier the energy required to swing up the pendulum is $2mgl$. This energy is supplied by the input, which implies the energy is equal to work done.

$$W = 2mgl$$

$$mngl = 2mgl$$

$$n = 2$$

Thus the maximum acceleration should be at least 2g for the pendulum to swing up in a single swing. If the acceleration is larger than 2g, the acceleration will be switched off when the pendulum angle has changed by θ^*. When the pendulum angle has changed by θ^*, the pendulum has moved a distance of $l \sin\theta^*$ horizontally. Thus the energy supplied to the pendulum will be given by:

$$W = mngl \sin\theta^*$$

Equating the above equation to the energy required to swing up a pendulum:

$$mngl \sin\theta^* = 2mgl$$

$$n \sin\theta^* = 2$$

$$\sin\theta^* = \frac{2}{n}$$

UNIVERSITY OF MARYLAND

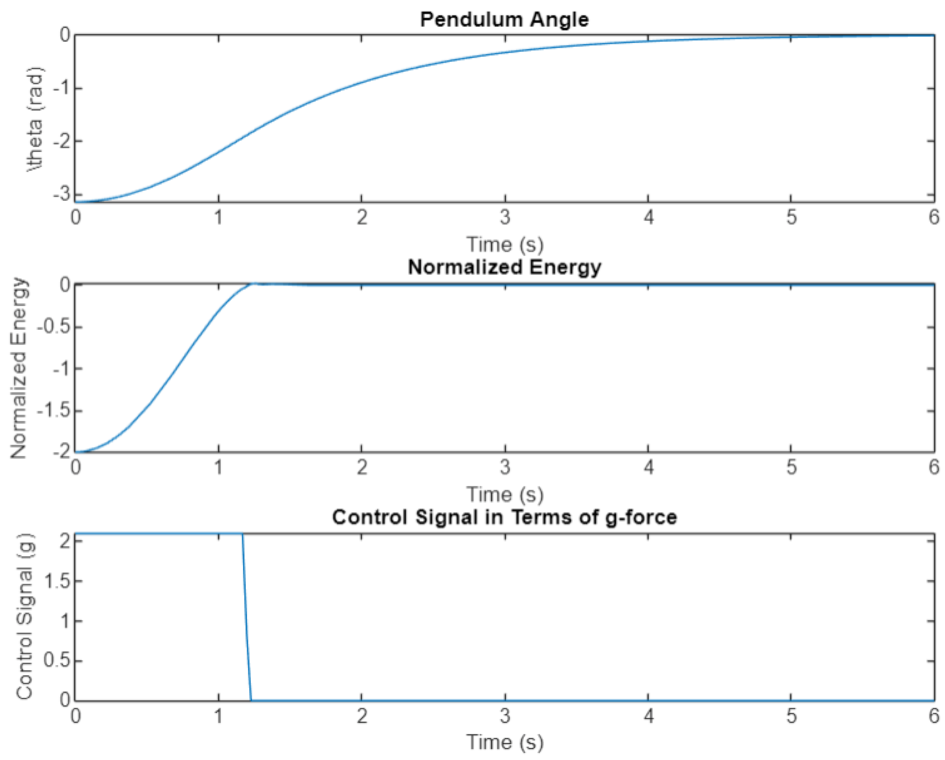

Figure 6: Single swing double switch Simulation Output

2.1.1 Simulation of SSDS behavior

The Single swing double switch behavior has been simulated in MATLAB. The code used to simulate the behavior can be found in the *MATLAB Code for SSDS* section. The results from the paper for SSDS have been replicated and shown in the figure 6. The starting point of the simulation is the downward position ($\theta = -\pi$). This simulation, like all other simulations in this document, is performed using the normalized model. The general parameter values used are $\omega_0 = 1$, $n = 2.1$, and $k = 100$. The higher value of k makes the behavior resemble a pure switching strategy, as the energy supplied transitions from zero to its maximum value and back to zero. Additionally, since $\omega_0 = 1$, the value of l (the length of the pendulum) used in the simulation is 9.81, based on the relationship $\omega_0 = \sqrt{g/l}$. The choice of some parameters chosen is not given in the paper. Hence, after multiple trial and error, we were able to fine tune the parameters to achieve results similar to the paper.

For the purpose of better understanding, the pendulum motion is visualized as an animation. The visualizations utilized are purely for reference and are not part of the discussions mentioned in the paper. The visualization of animation done purely for reference is available here.

Scan or click the QR code to view the simulation.

2.2 Single-swing triple-switch (SSTS) behavior

In the single swing and triple switch behavior, the triple switch means that the control signal is switching signs three times. Authors mention about $n > 1$, as the pendulum must reach horizontal position in one swing, but how much larger than 1 does the value of n needs to be for the pendulum to get to upright position in one swing.

To find the n value, we consider the situation illustrated in figure 9. It is assumed that the observer is at point O and when observed from point O. It is assumed that the pivot is accelerated by ng along the positive x-axis. Thus the acceleration of the pendulum at B is given by $w = g\sqrt{1+n^2}$ and the direction is along OB.

The potential energy $P.E_{OB}$ of the pendulum at position B is given by $mw(a)$, where a is the distance qr as shown in figure 9.

The input signal changes direction as soon as the pendulum crosses the point D.

Thus the acceleration of the pendulum in position E is given by $w = g\sqrt{1+n^2}$, but by this time the input signal changes direction and the overall resultant acceleration has a direction along OC.

The potential energy $P.E_{OE}$ of the pendulum at position B is given by $mw(b)$, where b is the distance rB as shown in figure 9.

When the pendulum moves from A to D it loses the potential energy mwa, which is converted to kinetic energy. The pendulum will swing towards the uproght position if its kinetic energy is so large that it reaches the point E. The condition for this is $a \geq b$

The value of a and b can be obtained from figure 9. It can be said that the $qB = a + b$ and from the triangle ODr.

$$\sin\theta_0 = \frac{Or}{OD}$$

$$OD = l - b$$

$$\sin\theta_0 = \frac{l-b}{l}$$

$$l\sin\theta_0 = l - b$$

$$b = l - l\sin\theta_0$$

$$b = l(1-\sin\theta_0)$$

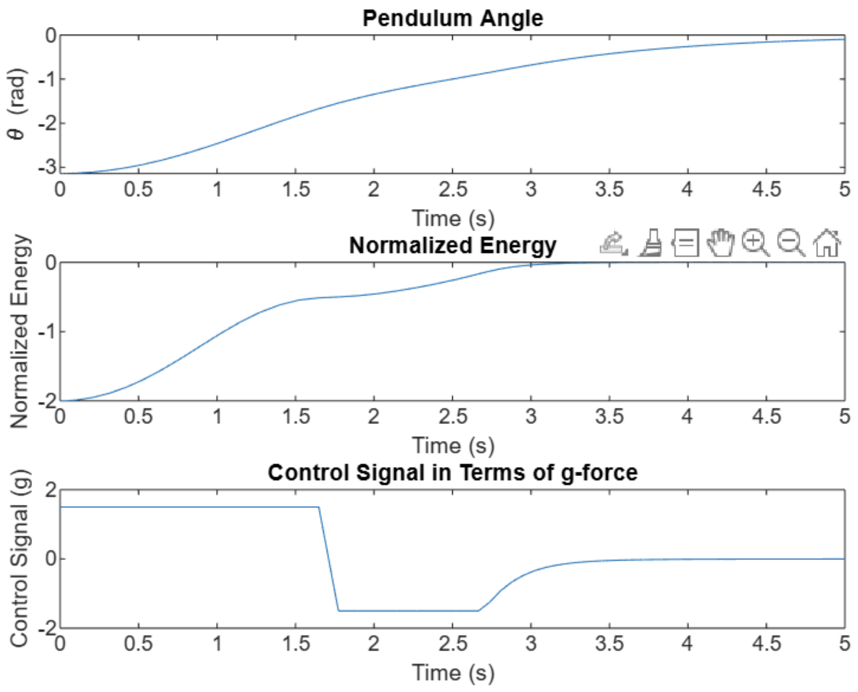

Figure 7: Single Swing Triple Switch Simulation Results.

From the triangle OBp, the length Op is given as $l - (a + b)$ and

$$\cos\theta_0 = \frac{Op}{OB}$$

$$\cos\theta_0 = \frac{l - (a + b)}{l}$$

$$l\cos\theta_0 = l - (a + b)$$

$$(a + b) = l - l\cos\theta_0$$

$$a = l - l\cos\theta_0 - b$$

Substituting the value of b,

$$a = l - l\cos\theta_0 - (l - l\sin\theta_0)$$

$$a = l - l\cos\theta_0 - l + l\sin\theta_0$$

$$a = l\sin\theta_0 - l\cos\theta_0$$

$$a = l(\sin\theta_0 - \cos\theta_0)$$

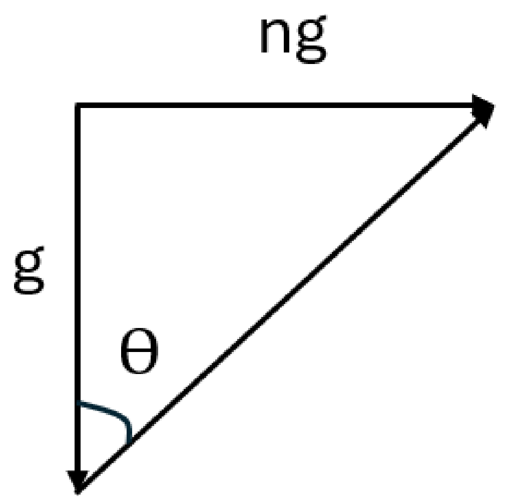

Figure 8: Triangle formed from the input acceleration and the acceleration due to gravity

Now substituting this in the earlier condition,

$$a \geq b$$

$$l(\sin\theta_0 - \cos\theta_0) \geq l(1 - \sin\theta_0)$$
$$(\sin\theta_0 - \cos\theta_0) \geq (1 - \sin\theta_0)$$
$$2\sin\theta_0 \geq (1 + \cos\theta_0)$$

(This is the same as equation (11) in the paper).

From the figure 8,

$$\tan\theta_0 = \frac{ng}{g}$$
$$n = \tan\theta_0$$

Substituting the value of n and using equality in equation 11.

$$2\sin\theta_0 = (1 + \cos\theta_0)$$

Dividing the above equation with $\cos\theta_0$ on both sides.

$$\frac{2\sin\theta_0}{\cos\theta_0} = \frac{(1 + \cos\theta_0)}{\cos\theta_0}$$

$$2\tan\theta_0 = \frac{(1+\cos\theta_0)}{\cos\theta_0}$$

$$2\tan\theta_0 = \frac{1}{\cos\theta_0} + 1$$

$$2\tan\theta_0 = \frac{\sqrt{1}}{\cos\theta_0} + 1$$

$$2\tan\theta_0 = \frac{\sqrt{\sin{\theta_0}^2 + \cos{\theta_0}^2}}{\cos\theta_0} + 1$$

$$2\tan\theta_0 = \sqrt{\frac{\sin{\theta_0}^2 + \cos{\theta_0}^2}{\cos{\theta_0}^2}} + 1$$

$$2\tan\theta_0 = \sqrt{\frac{\sin{\theta_0}^2}{\cos{\theta_0}^2} + \frac{\cos{\theta_0}^2}{\cos{\theta_0}^2}} + 1$$

$$2\tan\theta_0 = \sqrt{\frac{\sin{\theta_0}^2}{\cos{\theta_0}^2} + 1} + 1$$

$$2\tan\theta_0 = \sqrt{1 + \tan{\theta_0}^2} + 1$$

$$2n = \sqrt{1+n^2} + 1$$

$$(2n-1) = \sqrt{1+n^2}$$

Squaring on both sides

$$(2n-1)^2 = 1 + n^2$$

$$4n^2 + 1 - 4n = 1 + n^2$$

$$3n^2 - 4n = 0$$

$$3n^2 = 4n$$

$$3n = 4$$

$$n = \frac{4}{3}$$

Thus to have a single swing triple switch behavior the acceleration of the pivot must thus be at least $\frac{4g}{3}$. If $n = \frac{4}{3}$ the pivot accelerates to the right until the pendulum reaches the horizontal. The pivot is then accelerated in the opposite direction until the desired energy is obtained.

In the case where n is greater than $\frac{4}{3}$, the acceleration of the pivot can be set to zero before the pendulum reaches the point E. Let θ^* be the angle of the pendulum when the acceleration of the pivot is set to zero. Let us assume F be the position where the acceleration of the pivot is set to zero and θ^* as mentioned earlier be the angle between AOF.

Now since the angle AOD is 90^o, the angle DOF or sOF will be given by

$$\theta^* = 90^o + \angle DOF$$

$$\angle DOF = \theta^* - 90^o$$

Now to find the horizontal distance traveled by the center of mass of the pendulum when moving from A to F will be sum of horizontal distance traveled by the center of mass when moving from A to D and horizontal distance traveled by the center of mass when moving from D to F. Horizontal distance traveled by the center of mass when moving from A to D is l and horizontal distance traveled by the center of mass when moving from D to F is given by Ds which can be solved using the $\triangle sOF$

From $\triangle sOF$,

$$\cos \angle sOF = \frac{sO}{OF}$$

Since $\angle DOF = \angle sOF$

$$\cos \angle DOF = \frac{sO}{OF}$$

From earlier $\angle DOF = \theta^* - 90^o$

$$\cos \theta^* - 90^o = \frac{sO}{OF}$$

Since $cos{-\theta} = cos\theta$

$$\cos (90^o - \theta^*) = \frac{sO}{OF}$$

$$\cos (90^o - \theta^*) = \frac{sO}{l}$$

Since $cos(90 - \theta) = sin\theta$

$$\sin \theta^* = \frac{sO}{l}$$

$$sO = l \sin \theta^*$$

UNIVERSITY OF MARYLAND

Since $DO = l$ and $DO = sO + Ds$

$$Ds = l - sO$$

$$Ds = l - l\sin\theta^*$$

So the total distance traveled by the center of the mass of the pendulum is

$$l + Ds$$

$$l + l - l\sin\theta^*$$
$$2l - l\sin\theta^*$$
$$l(2 - \sin\theta^*)$$

Force in the horizontal direction is mu where u is the input signal. Since $u = ng$, the force in the horizontal direction will be mng. Thus work done by the pendulum will be $Force *$ distance traveled, which is $mngl(2-\sin\theta^*)$, which is the energy supplied to the pendulum. This energy is driving the pendulum to the upright position and thus, this has be equal to $2mgl$.

$$2mgl = mngl(2 - \sin\theta^*)$$

$$2gl = ngl(2 - \sin\theta^*)$$
$$2l = nl(2 - \sin\theta^*)$$
$$2 = n(2 - \sin\theta^*)$$
$$\frac{2}{n} = 2 - \sin\theta^*$$
$$\sin\theta^* = 2 - \frac{2}{n}$$
$$\sin\theta^* = 2(1 - \frac{1}{n})$$

2.2.1 Simulation of SSTS behavior

The Single swing triple switch behavior has been simulated in MATLAB. The code used to simulate the behavior can be found in the *MATLAB Code for SSTS* section. The results from the paper for SSTS have been replicated and shown in the figure 7.

UNIVERSITY OF MARYLAND

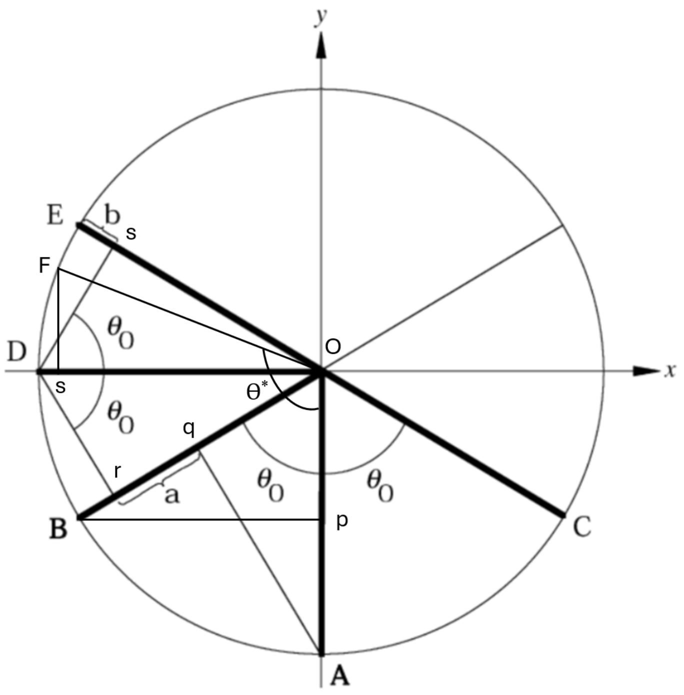

Figure 9: Diagram to explain the single swing, triple switch behavior.

For the purpose of better understanding, the pendulum motion is visualized as an animation. The visualizations utilized are purely for reference and are not part of the discussions mentioned in the paper. The visualization of animation done purely for reference is available here.

Scan or click the QR code to view the simulation.

2.3 Multi-swing behavior

In the SSTS and SSDS sections we found that n value needs to be greater than or equal to $\frac{4g}{3}$ for the pendulum to get to the inverted position in one swing. So if the n value if less than $\frac{4g}{3}$ then the pendulum needs more than one swing to get to the upright position. Let us look into getting the pendulum into the upright position in 2 swings. As defined earlier, the observer is still at pivot O and from his point of view the net acceleration of the pendulum at any point is given by $w = g\sqrt{1+n^2}$.

The Multi Swing behavior is further explained using the figure 10, where the pendulum starts at rest at A and the pivot first accelerates to the right. The pendulum first swings from A to D and then the acceleration of the pivot is reversed, which allows the pendulum to reach

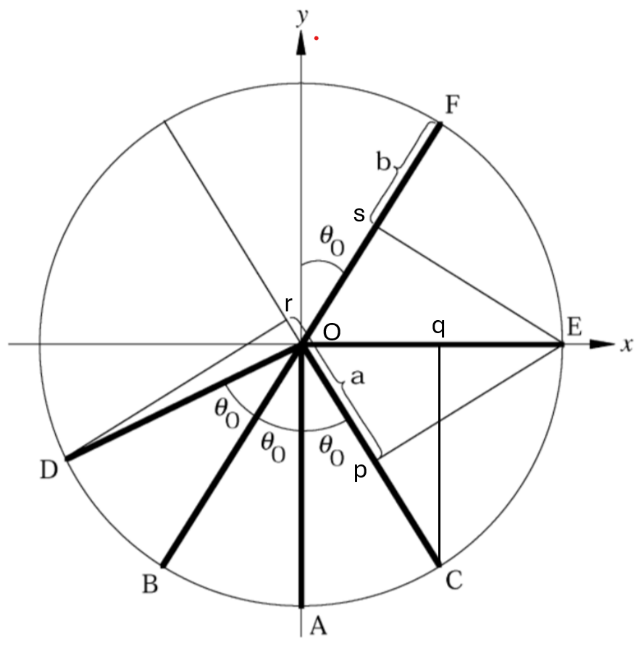

Figure 10: Multi Swing Multi Switch Diagram

E and then the acceleration of the pivot is again reversed. It is necessary that the pendulum reach point F without any additional reversal of the acceleration. This is possible if its energy at E is sufficiently large to bring it up to point F. Thus we can say that the work done by the pendulum while moving from D to E is mwa. Which is the change in energy of the pendulum. This change in energy is then converted into kinetic energy, this energy must be sufficiently large to move the pendulum from E to F. The energy required to move the pendulum from E to F is mwb.

$$mwa \geq mwb$$

$$a \geq b$$

The value of a and b can be obtained from the figure 10 as shown below:

We know that $OF = OE = l$ and $OF = Os + sF$ and $sF = b$

From the $\triangle OEs$ we know that $\angle sOE = 90 - \theta_0$ and

$$cos90 - \theta_0 = \frac{Os}{OE}$$

$$sin\theta_0 = \frac{Os}{l}$$

$$Os = lsin\theta_0$$

UNIVERSITY OF MARYLAND

Since $OF = Os + sF$

$$l = Os + b$$

$$b = l - Os$$

Substituting Os value from earlier

$$b = l - lsin\theta_0$$

$$b = l(1 - sin\theta_0)$$

To find a value we need to find pO and Or as $a = pO + Or$, The value of pO can be found from the $\triangle EOp$:

We know that the $\angle AOE = 90^o$ and $\angle AOC or AOp = \theta_0$

$$\angle AOp + \angle pOE = 90^o$$

$$\angle pOE = 90^o - \angle AOp$$

$$\angle pOE = 90^o - \theta_0$$

From the right angled $\triangle OpE$

$$\cos pOE = \frac{pO}{OE}$$

$$\cos 90^o - \theta_0 = \frac{pO}{l}$$

$$pO = l\cos 90^o - \theta_0$$

Since $\cos 90^o - \theta = \sin\theta$

$$pO = l\sin\theta_0$$

We know that the COr is a straight line thus the $\angle COr = 180^o$ Since $\angle COr = \angle COD + \angle DOr$ and $\angle COD = 3\theta_0$

$$\angle DOr = \angle COr - \angle COD$$

$$\angle DOr = 180^o - \angle COD$$

$$\angle DOr = 180^o - 3\theta_0$$

From the right angled $\triangle DrO$

$$\cos DOr = \frac{Or}{OD}$$

$$\cos 180^o - 3\theta_0 = \frac{Or}{OD}$$

$$\cos 180^o - 3\theta_0 = \frac{Or}{l}$$

$$Or = l\cos 180^o - 3\theta_0$$

Since $\cos 180 - \theta = -\cos\theta$

$$Or = -l\cos 3\theta_0$$

Now to find the value of a from earlier $a = pO + Or$ substituting the values of pO and Or

$$a = l\sin\theta_0 + (-l\cos 3\theta_0)$$

$$a = l\sin\theta_0 - l\cos 3\theta_0$$

$$a = l(\sin\theta_0 - \cos 3\theta_0)$$

Substituting the a and b values in the condition $a \geq b$ we get

$$l(\sin\theta_0 - \cos 3\theta_0) \geq l(1 - sin\theta_0)$$

$$(\sin\theta_0 - \cos 3\theta_0) \geq (1 - sin\theta_0)$$

$$\sin\theta_0 + \sin\theta_0 \geq 1 + \cos 3\theta_0$$

$$2\sin\theta_0 \geq 1 + \cos 3\theta_0$$

(This is the same as equation (12) in the paper).

2.3.1 Simulation of Multi-swing behavior

The Double swing Multi switch(DSMS) behavior has been simulated in MATLAB. The code used to simulate the behavior can be found in the *MATLAB Code for Multi-Swing* section. The results from the paper for DSMS have been replicated and shown in the figure 11.

For the purpose of better understanding, the pendulum motion is visualized as an animation. The visualizations utilized are purely for reference and are not part of the discussions mentioned in the paper. The visualization of animation done purely for reference is available here.

Scan or click the QR code to view the simulation.

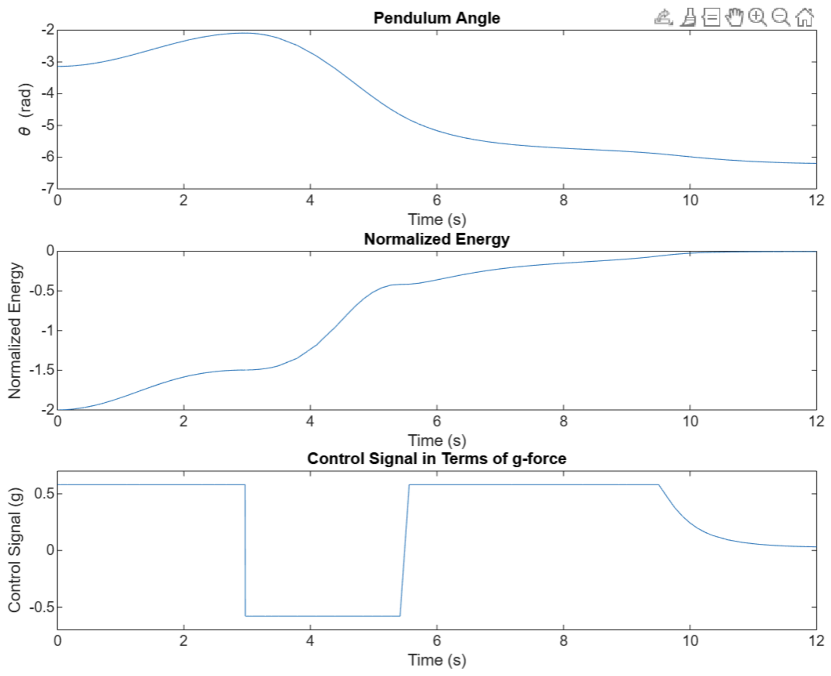

Figure 11: Double Swing Multi Switch Simulation Results

2.4 The general case

From equations (11) and (12) it can be observed that for single swing the condition becomes $2\sin\theta_0 \geq 1 + \cos\theta_0$ and for two swings the condition becomes $2\sin\theta_0 \geq 1 + \cos 3\theta_0$. This can be further generalized for k swings as shown below:

$$2\sin\theta_0 \geq 1 + \cos{(2k-1)}\theta_0$$

where k is the number of swings. ***(This is the same as equation (13) in the paper).***

we know that the value of n is given by $n = \tan\theta_0$, so for various values of k, value of n can be obtained by solving the equation

$$2\sin\theta_0 \geq 1 + \cos{(2k-1)}\theta_0$$

We already solved this for $k = 1$ and found that value of n should be at least $n = \frac{4}{3} = 1.333$ For $k = 2$, we can solve the below equation as:

$$2\sin\theta_0 = 1 + \cos 3\theta_0$$

From the expansion $\cos 3\theta = 4\cos^3\theta - 3\cos\theta$

$$2\sin\theta_0 = 1 + 4\cos^3\theta_0 - 3\cos\theta_0$$

Dividing the above equation with $\cos\theta_0$ on both sides

$$\frac{2\sin\theta_0}{\cos\theta_0} = \frac{(1 + 4\cos^3\theta_0 - 3\cos\theta_0)}{\cos\theta_0}$$

$$\tan\theta_0 = \frac{1}{\cos\theta_0} + \frac{4\cos^3\theta_0}{\cos\theta_0} - \frac{3\cos\theta_0}{\cos\theta_0}$$

$$\tan\theta_0 = \frac{1}{\cos\theta_0} + 4\cos^2\theta_0 - 3$$

From identity $\cos^2\theta + \sin^2\theta = 1$ we can obtain $\cos^2\theta = \frac{1}{1+\tan^2\theta}$

$$\tan\theta_0 = \frac{1}{\cos\theta_0} + 4\frac{1}{1+\tan^2\theta_0} - 3$$

$$\tan\theta_0 = \frac{1}{\cos\theta_0} + \frac{4}{1+\tan^2\theta_0} - 3$$

From the identity $\frac{1}{\cos\theta} = \sec\theta$ and $\sec^2\theta = 1 + \tan^2\theta$

$$\tan\theta_0 = \sqrt{1+\tan^2\theta_0} + \frac{4}{1+\tan^2\theta_0} - 3$$

Solving the above equation for $\tan\theta_0$ we get $\tan\theta_0 = 0.577$. Thus the value of n for $k = 2$ is $n = 0.577$

Similarly the generalized condition mentioned in the equation (13) can be solved using numerical methods for further k values and corresponding n values can be obtained. Which are shown in both the table below and also in the figure 12. The MATLAB code for that finds the numerical relation between the number of swings and n value is available in the Appendix Section.

n	1.333	0.577	0.388	0.296	0.241	0.204	0.177	0.156	0.140	0.128
k	1	2	3	4	5	6	7	8	9	10

UNIVERSITY OF MARYLAND

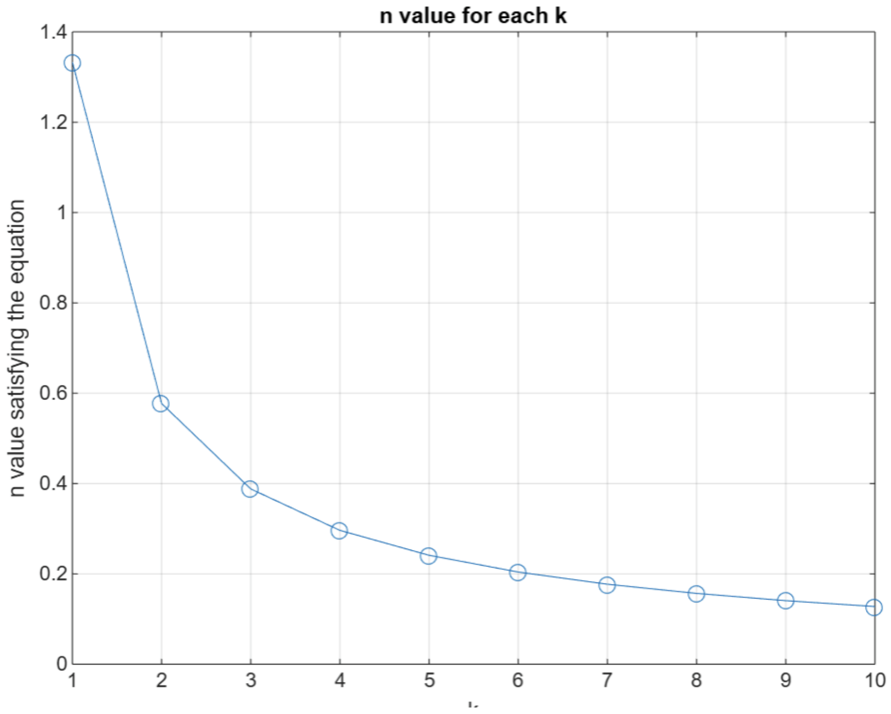

Figure 12: The value of n for each value of k varying from 1 to 10

2.4.1 Simulation of five swing behavior

The Five swings Multi switch(FSMS) behavior has been simulated in MATLAB. The code used to simulate the behavior can be found in the *MATLAB Code for Five Swing Behavior* section. The results from the paper for FSMS have been replicated and shown in the figure 13.

For the purpose of better understanding, the pendulum motion is visualized as an animation. The visualizations utilized are purely for reference and are not part of the discussions mentioned in the paper. The visualization of animation done purely for reference is available here.

Scan or click the QR code to view the simulation.

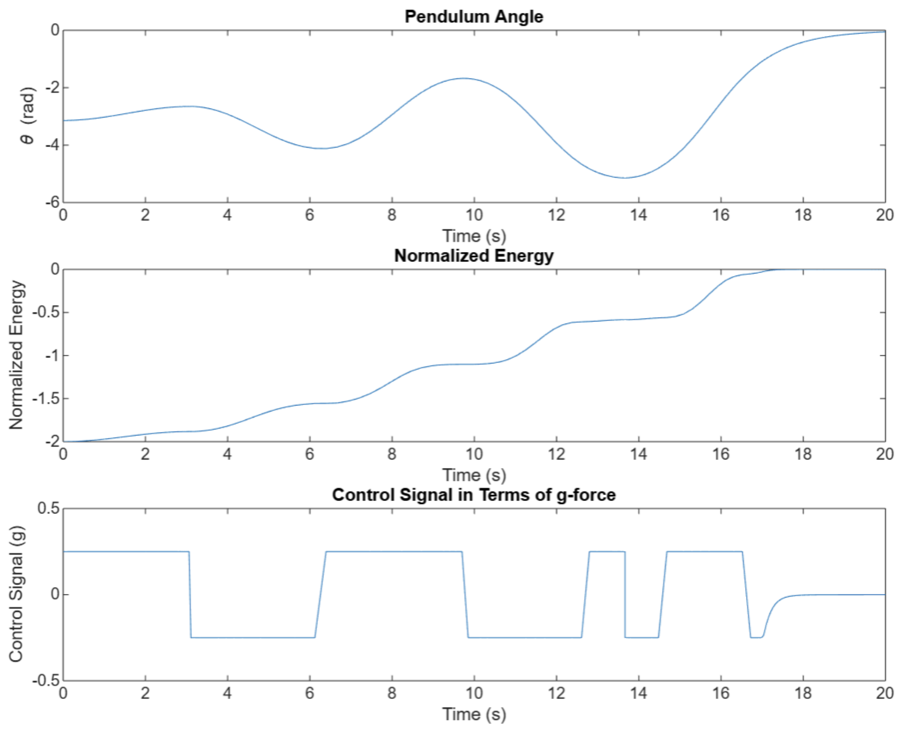

Figure 13: Simulation Results for Five Swing Behavior

2.5 Minimum time strategies

The authors talk about Pontryagins maximum principle , where they mentioned that the minimum time strategies for swinging up a pendulum are of bang-bang type. This means that the energy is injected to the pendulum at a maximum rate and then removed at a maximum rate in such a way that the pendulum retains energy corresponding to the equilibrium energy at upright position. An explanation for this given in the Appendix. The authors mention that for small values of n, the control signal for minimum time strategy will be same as energy control initially, however the final part of the control signal are different because the strategy will set the control signal to zero when the desired energy has been obtained, they described this as a strategy in which there is no overshoot in the energy.

The example mentioned in the paper considers $n > 2$ where a single swing strategy can be used. It has been already shown in SSDS that the acceleration needs to be switched off at θ^* where the value of θ^* is given from the equation $\sin\theta^* = \frac{2}{n}$. To minimize the time the authors suggest to continue this acceleration till the pendulum reaches horizontal position and then reversing the acceleration till the desired energy is obtained.

From equation (9) we already know that the energy supplied to the pendulum is $nmgl(2 - \sin\theta^*)$. Since are reversing the direction of acceleration after reaching the horizontal position till the pendulum travels $l\sin\theta^*$, the reduction in energy is given by $-nmgl\sin\theta^*$. We know that the energy supplied to the pendulum to reach upright position is $2mgl$. Thus from above we can say:

$$nmgl(2 - \sin\theta^*) - nmgl\sin\theta^* = 2mgl$$

$$nmgl(2 - \sin\theta^* - \sin\theta^*) = 2mgl$$

$$nmgl(2 - 2\sin\theta^*) = 2mgl$$

$$2nmgl(1 - \sin\theta^*) = 2mgl$$

$$n(1 - \sin\theta^*) = 1$$

$$\sin\theta^* = (1 - \frac{1}{n})$$

$$\theta^* = \arcsin{(1 - \frac{1}{n})}$$

The maximum energy is:

$$E_{max} = nmgl\sin\theta^*$$

$$E_{max} = nmgl\sin\arcsin{(1 - \frac{1}{n})}$$

$$E_{max} = nmgl(1 - \frac{1}{n})$$

$$E_{max} = (n-1)mgl$$

(This is the same as equation (14) in the paper).

For $n = 2$ the maximum energy is mgl. The "energy overshoot" is 50% for $n = 2$ and it increases rapidly with n. The results given in the paper are replicated here in Figure 14 for $n = 2.1$ and in Figure 15 for $n = 5$. The code used for simulation of minimum time strategies can be found in the *MATLAB Code for Minimum Time Strategies* section.

UNIVERSITY OF MARYLAND

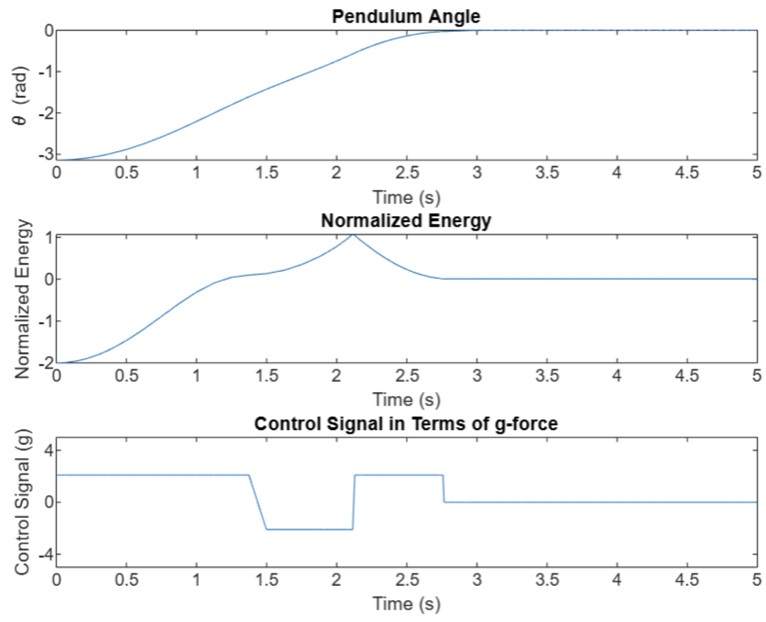

Figure 14: Minimum Time strategy applied to n=2.1

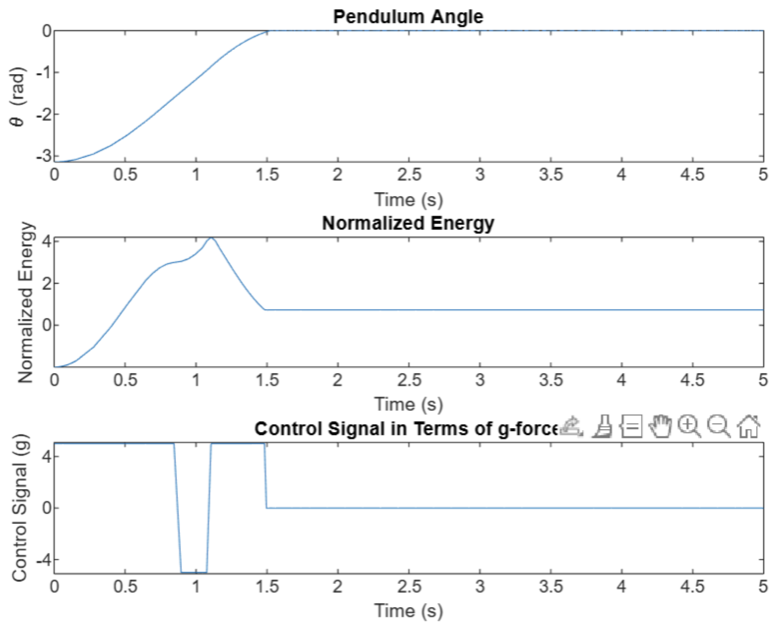

Figure 15: Minimum Time strategy applied to n=5

2.6 Generalizations

The focus of the authors thus far has been on addressing control challenges for a single pendulum system. While the time-varying nature of the system presented an intriguing and worthwhile problem, a single pendulum remains relatively simple in its dynamics. The system can be characterized as a first-order system, with its gain determined by the pendulum's angle and the rate of change of this angle. A notable limitation of such a system is the potential for the gain to vanish, which, while significant, occurs only at isolated time instants due to the time-varying nature of the gain produced by the pendulum's motion.

This limitation presents an opportunity for extending the approach to more complex and challenging configurations. Specifically, the underlying principles of the single pendulum system can be generalized to control systems involving rotating pendulums or multiple interconnected pendulums. These extended systems inherently exhibit higher-order dynamics and more intricate interactions, making their control significantly more complex. The transition from a single pendulum to multiple pendulum systems introduces additional degrees of freedom and nonlinearities, which demand more advanced control strategies.

By building upon the insights gained from the single pendulum problem, the authors aims to address the control of these more sophisticated configurations, exploring how time-varying gains can be effectively managed in systems with multiple rotating elements. This extension not only broadens the scope of the problem but also enhances its relevance to real-world applications where multi-pendulum systems frequently arise, such as robotics, aerospace, and structural engineering.

2.6.1 General dynamical system

To generalize consider a mechanical system, which can be described by the equation

$$M(q,\dot{q})\ddot{q} + C(q,\dot{q})\dot{q} + \frac{\partial U(q)}{\partial q} = T$$

(This is the same as equation (15) in the paper).

Where q is a vector of generalized coordinates, $M(q,\dot{q})$ is the inertia matrix, $C(q,\dot{q})$ is the damping matrix, $U(q)$ is the potential energy and T is the external control torques

The total energy is given by the sum of kinetic energy(K.E) and potential energy(P.E).

$$E = K.E + P.E$$

$$K.E = \frac{1}{2}(inertia)(\text{angular velocity})^2$$

This equation is valid only when the terms are scalar values and doesn't represent any form of matrix. Since we are trying to generalize the system is assumed to me multi dimensional and thus is represented using a matrix.

$$K.E = \frac{1}{2}(\text{angular velocity})^{`}(inertia)(\text{angular velocity})$$

Where the symbol ` represents the transpose of the angular velocity matrix. Thus the kinetic energy is given by:

$$K.E = \frac{1}{2}(\dot{q})^{`}M(q,\dot{q})\dot{q}$$

Thus the total enery E is given by:

$$E = \frac{1}{2}(\dot{q})^{`}M(q,\dot{q})\dot{q} + U(q)$$

(This is the same as equation (16) in the paper).

The time derivative of E is given by:

$$\frac{dE}{dt} = \frac{d}{dt}(\frac{1}{2}(\dot{q})^{`}M(q,\dot{q})\dot{q} + U(q))$$

$$\frac{dE}{dt} = \frac{d}{dt}\frac{1}{2}(\dot{q})^{`}M(q,\dot{q})\dot{q} + \frac{d}{dt}(U(q))$$

Using the chain rule for derivatives,

$$\frac{dE}{dt} = \frac{d}{dt}\frac{1}{2}(\dot{q})^{`}M(q,\dot{q})\dot{q} + \frac{d}{dt}(U(q))$$

From the Matrix calculus, $y = x^{`}Ax$ then $\frac{dy}{dt} = x^{`}\dot{A}x + 2x^{`}A\dot{x}$

$$\frac{dE}{dt} = \frac{1}{2}(\dot{q})^{`}\frac{d}{dt}M(q,\dot{q})\dot{q} + (\dot{q}^{`}M(q,\dot{q}))\frac{d\dot{q}}{dt} + \frac{\partial U(q)}{\partial q}\frac{dq}{dt}$$

$$\frac{dE}{dt} = \frac{1}{2}(\dot{q})^{`}\frac{d}{dt}M(q,\dot{q})\dot{q} + \dot{q}^{`}M(q,\dot{q})\ddot{q} + \frac{\partial U(q)}{\partial q}\dot{q}$$

$$\frac{dE}{dt} = \frac{1}{2}(\dot{q})^{`}\dot{M}(q,\dot{q})\dot{q} + \dot{q}^{`}M(q,\dot{q})\ddot{q} + \frac{\partial U(q)}{\partial q}\dot{q}$$

$$\frac{dE}{dt} = \frac{1}{2}(\dot{q})^{`}\dot{M}(q,\dot{q})\dot{q} + \dot{q}^{`}M(q,\dot{q})\ddot{q} + \frac{\partial U(q)}{\partial q}\dot{q}$$

From the equation (15) we already know that

$$(M(q,\dot{q})\ddot{q}) + (\frac{\partial U(q)}{\partial q}) = T - C(q,\dot{q})\dot{q}$$

Substituting this in above equation for time derivative of Energy, we get:

$$\frac{dE}{dt} = \frac{1}{2}(\dot{q})^{`}\dot{M}(q,\dot{q})\dot{q} + \dot{q}^{`}M(q,\dot{q})\ddot{q} + \frac{\partial U(q)}{\partial q}\dot{q}$$

$$\frac{dE}{dt} = \frac{1}{2}(\dot{q})^{`}\dot{M}(q,\dot{q})\dot{q} + \dot{q}^{`}(T - C(q,\dot{q})\dot{q} - \frac{\partial U(q)}{\partial q}) + \frac{\partial U(q)}{\partial q}\dot{q}$$

Rearranging the terms we get:

$$\frac{dE}{dt} = \frac{1}{2}(\dot{q}^{`}\dot{M}(q,\dot{q})\dot{q}) - \dot{q}^{`}C(q,\dot{q})\dot{q} + \dot{q}^{`}(T) - \dot{q}^{`}\frac{\partial U(q)}{\partial q} + \frac{\partial U(q)}{\partial q}\dot{q}$$

Since the potential energy is a scalar and $\dot{q}^{`}\frac{\partial U(q)}{\partial q} = \frac{\partial U(q)}{\partial q}\dot{q}$, similarly $\dot{q}^{`}(T) = T^{`}\dot{q}$

$$\frac{dE}{dt} = \frac{1}{2}\dot{q}^{`}(\dot{M}(q,\dot{q})\dot{q} - 2C(q,\dot{q})\dot{q}) + \dot{q}^{`}(T) - \frac{\partial U(q)}{\partial q}\dot{q} + \frac{\partial U(q)}{\partial q}\dot{q}$$

$$\frac{dE}{dt} = \frac{1}{2}\dot{q}^{`}(\dot{M}(q,\dot{q})\dot{q} - 2C(q,\dot{q})\dot{q}) + T^{`}\dot{q}$$

(This is the same as equation (17) in the paper).

Now in the given inertia matrix $M(q,\dot{q})$, the $(k,j)^{th}$ component of $\dot{M}(q,\dot{q})$ is given by the chain rule as

$$\dot{m}_{kj} = \sum_{i=1}^{n}\frac{\partial m_{kj}}{\partial q_i}\dot{q}_i$$

We also know that for a given damping matrix $C(q,\dot{q})$, the $(k,j)^{th}$ is defined as

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i = \sum_{i=1}^{n}\frac{1}{2}(\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k})\dot{q}_i$$

Now the $(k,j)^{th}$ component of $\dot{M}(q,\dot{q}) - 2C(q,\dot{q})$ is given by:

$$n_{kj} = \dot{d}_{kj} - 2c_{kj}$$

$$n_{kj} = \sum_{i=1}^{n}\frac{\partial m_{kj}}{\partial q_i}\dot{q}_i - 2(\sum_{i=1}^{n}\frac{1}{2}(\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k})\dot{q}_i)$$

$$n_{kj} = \sum_{i=1}^{n}[\frac{\partial m_{kj}}{\partial q_i}\dot{q}_i - (\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k})\dot{q}_i]$$

$$n_{kj} = \sum_{i=1}^{n}[\frac{\partial m_{kj}}{\partial q_i} - (\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k})]\dot{q}_i$$

$$n_{kj} = \sum_{i=1}^{n}[\frac{\partial m_{kj}}{\partial q_i} - \frac{\partial m_{kj}}{\partial q_i} - \frac{\partial m_{ki}}{\partial q_j} + \frac{\partial m_{ij}}{\partial q_k}]\dot{q}_i$$

$$n_{kj} = \sum_{i=1}^{n}[\frac{\partial m_{ij}}{\partial q_k} - \frac{\partial m_{ki}}{\partial q_j}]\dot{q}_i$$

Now using the above equation finding the $(j,k)^{th}$ component:

$$n_{jk} = \sum_{i=1}^{n}[\frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{ji}}{\partial q_k}]\dot{q}_i$$

From above we can see that

$$n_{jk} = -n_{kj}$$

If we substitute $k = j$ in the equation for the compenent we get:

$$n_{kj} = \sum_{i=1}^{n}[\frac{\partial m_{ij}}{\partial q_k} - \frac{\partial m_{ki}}{\partial q_j}]\dot{q}_i$$

$$n_{jj} = 0$$

Thus the matrix $\dot{M}(q,\dot{q}) - 2C(q,\dot{q})$ is skew symmetric.
For a skew symmetric matrix A and any vector x:

$$(x^{`}Ax)^{`} = x^{`}Ax$$

$$x^{`}A^{`}x = x^{`}Ax$$

$$x^{`}(-A)x = x^{`}Ax$$

$$-x^{`}(A)x = x^{`}Ax$$

$$2x^{`}Ax = 0$$

$$x^{`}Ax = 0$$

Thus the matrix $\dot{q}^{`}(\dot{M}(q,\dot{q}) - 2C(q,\dot{q}))\dot{q} = 0$ as $\dot{M}(q,\dot{q}) - 2C(q,\dot{q})$ is skew symmetric, then the value of $\frac{dE}{dt}$ will be:

$$\frac{dE}{dt} = T^{`}\dot{q}$$

UNIVERSITY OF MARYLAND

2.6.2 Two pendulums on a cart with simulation

The authors have demonstrated how useful the energy control method is even for two pendulums on a cart. Figure 16 represents how a physical model might look in the real world.

Referencing the earlier derivations derived in Preliminaries, the rate of energy change for each pendulum can be written as follows:

$$\frac{dE_1}{dt} = -m_1 u l_1 \dot{\theta}_1 \cos(\theta_1)$$

$$\frac{dE_2}{dt} = -m_2 u l_2 \dot{\theta}_2 \cos(\theta_2)$$

Where,

E_1	:	Energy of pendulum 1 (J)
E_2	:	Energy of pendulum 2 (J)
m_1	:	Mass of pendulum 1 (kg)
m_2	:	Mass of pendulum 2 (kg)
l_1	:	Length of pendulum 1 (m)
l_2	:	Length of pendulum 2 (m)
$\dot{\theta}_1$	:	Angular velocity of pendulum 1 (rad/s)
$\dot{\theta}_2$	:	Angular velocity of pendulum 2 (rad/s)
θ_1	:	Angle of pendulum 1 (radians)
θ_2	:	Angle of pendulum 2 (radians)

The authors have chosen the Lyapunov function as

$$V = \frac{E_1^2 + E_2^2}{2}.$$

This function can drive both pendulums' energies E_1 and E_2 to zero.To demonstrate that the chosen function can be used as a Lyapunov function we need to verify if it meets the basic conditions of a Lyapunov function. The conditions as described in the Appendix are;

1. Positive Definiteness

The Lyapunov function V must be positive definite. For our Lyapunov function, since the square of energies are always non-negative, i.e $E_1^2 \geq 0$ and $E_2^2 \geq 0$.

The Lyapunov function,

$$V = 0 \text{ if and only if } E_1 = 0 \text{ and } E_2 = 0.$$

UNIVERSITY OF MARYLAND

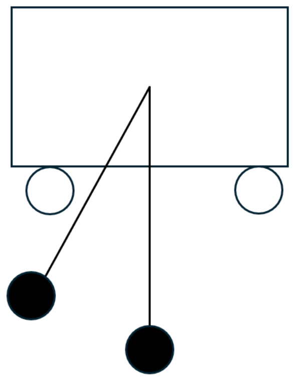

Figure 16: Two Pendulums on a Cart

Thus we have proved that, V is a positive definite which meets the first condition of choosing a Lyapunov function.

2. Negative derivative Definiteness

The derivative of the Lyapunov function V with respect to time, $\frac{dV}{dt}$, should be negative definite.
Let's compute the derivative of V with respect to time (t), we get,

$$\frac{dV}{dt} = \frac{\partial V}{\partial E_1}\frac{dE_1}{dt} + \frac{\partial V}{\partial E_2}\frac{dE_2}{dt}.$$

$\boxed{\text{By chain rule.}}$

Thus, we have:

$$\frac{dV}{dt} = E_1\frac{dE_1}{dt} + E_2\frac{dE_2}{dt}.$$

$$\boxed{\text{As, } \quad \frac{\partial V}{\partial E_1} = E_1, \quad \frac{\partial V}{\partial E_2} = E_2.}$$

Substituting $\frac{dE_1}{dt}$ and $\frac{dE_2}{dt}$ which we defined earlier into the above equation. we get,

$$\frac{dV}{dt} = E_1\left(-m_1 l_1 \dot{\theta}_1 \cos(\theta_1)\right) + E_2\left(-m_2 l_2 \dot{\theta}_2 \cos(\theta_2)\right).$$

This can be rearranged to:

$$\frac{dV}{dt} = -m_1 l_1 E_1 \dot{\theta}_1 \cos(\theta_1) - m_2 l_2 E_2 \dot{\theta}_2 \cos(\theta_2).$$

The above equation can be simplified using the expression,

$$G = m_1 l_1 E_1 \dot{\theta}_1 \cos(\theta_1) + m_2 l_2 E_2 \dot{\theta}_2 \cos(\theta_2),$$

Thus, the derivative of the Lyapunov function becomes:

$$\frac{dV}{dt} = -Gu.$$

The value of $\frac{dV}{dt}$ will stay negative in all cases except the following:
1. When the energy values of the pendulums are zero ($E_1 = 0$ and $E_2 = 0$), then

$$\frac{dV}{dt} = 0.$$

This is the condition for equilibrium that we have defined earlier at which both the pendulums are in an upright position.

2. If both pendulums are identical (i.e, same mass, length), then there is a possibility that their positions are in opposite directions,

$$\theta_1 = -\theta_2,$$

. Then such a case can lead to $G = 0$, which makes $\frac{dV}{dt}$ also be zero.

Control Law and simulation

The authors have proposed a control law

$$u = \text{Sat}_{\text{ng}}(kG)$$

This control law can drive the pendulums to upright positions (energy = 0). The only condition when this would not work is when the system is not controllable which would only arise when G becomes zero as discussed above.Utilizing the second order differential equation we have and the proposed control strategy, a simulation was ran over time period $t = 0$ to $t = 10$ (same as given in Fig.8 of the paper). The control input u is also constrained between $-n \cdot g$ to $n \cdot g$ to avoid instability. The authors specified the following parameters: $n = 1.5$, $k = 20$, $\omega_{0_1} = 1$, and $\omega_{0_2} = 2$. From the values of ω_0, we can calculate $l_1 = 9.81$ and $l_2 = \frac{9.81}{4}$, as $\omega_0 = \sqrt{\frac{g}{l}}$. The simulation results are available in 17. It can be observed that the simulation results are matching with the results that the authors published in the paper. As the mass values and initial velocities of the pendulums were not given, it took multiple attempts of trial and error spanning weeks to finally get results that matched the published results. The MATLAB code used with all parameter values and its explanation (as comments) is available

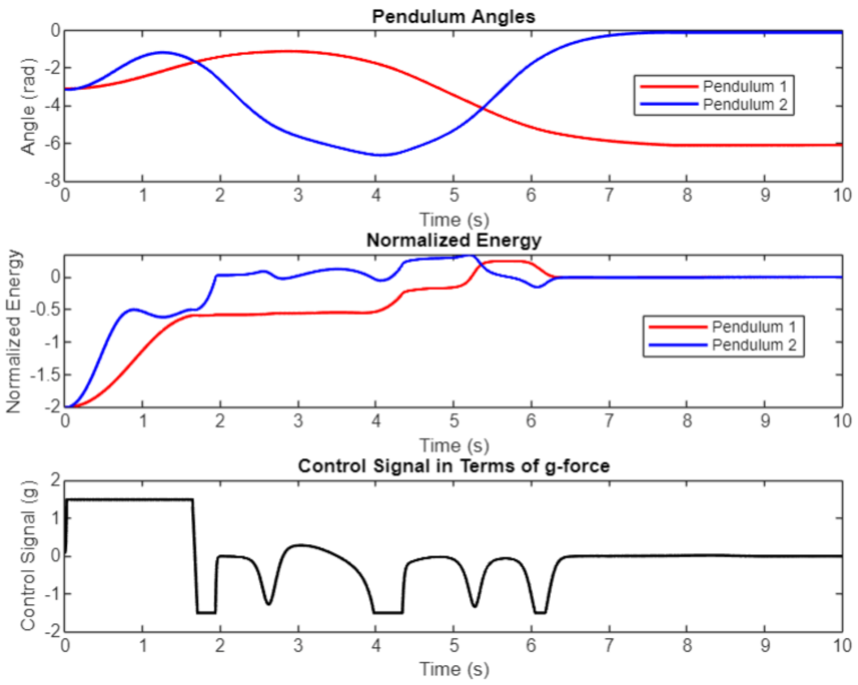

Figure 17: Two pendulums on a Cart Simulation Output

in the MATLAB code for Two Pendulums on a Cart section.

For the purpose of better understanding, the pendulum motion is visualized as an animation. The visualizations utilized are purely for reference and are not part of the discussions mentioned in the paper. The visualization of animation done purely for reference is available here.

Scan or click the QR code to view the simulation.

3 Conclusion

The authors demonstrated that the energy control is a practical and effective method to swing up a pendulum, with system performance dependent on the acceleration of the pivot relative to acceleration due to gravity, denoted as n (where $n = \frac{u}{g}$). The results of these demonstrations were replicated in this report, and additional derivations were performed on the equations presented by the authors. The key findings of this paper are ;

Single-Swing behavior is only possible when $n > \frac{4g}{3}$. The pendulum can be swung up in single swing with just two signal switches If $n \geq 2$. In this case the control signal applies maximum acceleration until the required energy is achieved and then switches off. For cases where $\frac{4g}{3} \leq n < 2$, single swing is possible but requires three control signal switches to achieve the required energy.

Multi-Swing behavior is required when $n < \frac{4g}{3}$. The pendulum requires multiple swings to accumulate enough energy to reach the upright position. In short, the number of swings increases as as n decreases.

Energy Control strategies apply enough energy into the system such that it can move from current state to the required state. In case of inverted pendulum, the energy is applied by supplying acceleration depending on the energy state of the system. The main advantage of this strategy is that there is minimal energy overshoot, making this approach robust and less sensitive approach especially in case of minimum time strategies. It has also been shown that the energy control strategy can be applied to other systems too like two pendulums on a cart.

References

[1] K. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[2] O. Boubaker, "The inverted pendulum: A fundamental benchmark in control theory and robotics," in *International Conference on Education and e-Learning Innovations*, pp. 1–6, 2012.

[3] K. H. Lundberg and T. W. Barton, "History of inverted-pendulum systems," *IFAC Proceedings Volumes*, vol. 42, no. 24, pp. 131–135, 2010. 8th IFAC Symposium on Advances in Control Education.

[4] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ: Princeton University Press, 2008.

[5] W. A. Malik, "Advanced control methods," 2024. Lecture notes for ENPM667 - Control of Robotic Systems.

[6] L. C. Evans, *An Introduction to Mathematical Optimal Control Theory*. Berkeley, CA: University of California, Berkeley, 2024. Spring 2024 version.

[7] J. LaSalle, "The 'bang-bang' principle," in *IFAC Proceedings Volumes*, vol. 1, pp. 503–507, 1960. 1st International IFAC Congress on Automatic and Remote Control, Moscow, USSR.

[8] I. The MathWorks, "Matlab," 2024. Version R2024a.

4 Appendix

4.1 Control Law

Control Law is generally a mathematical function or an algorithm that gives the control input to be given to a system based on the current and desired state of the system. In the book "Feedback Systems: An Introduction for Scientists and Engineers" [4], the authors define the control law as an algorithm that computes control action as a function of the sensor values. The goal of control law is to achieve the desired system state.

4.2 Lyapunov function

A Lyapunov function is a mathematical tool used in the analysis of the stability of dynamical systems. It is named after the Russian mathematician Aleksandr Lyapunov, who introduced the concept in the context of studying the stability of differential equations.

The explanation for Lyapunov Stability given in this document is referenced from [4], [5]. As discussed above Lyapunov function is a mathematical tool generally used for analyzing stability. It is an energy-like function that helps analyze a system's behavior over time. In a sense, it assesses whether trajectories will converge to an equilibrium point or diverge from it. By taking the derivative of the Lyapunov function with respect to time, we can understand if a system is stable, unstable, or asymptotically stable. Asymptotic stability is a condition where the system returns to equilibrium even after small disturbances in the system. That is as time approaches infinity, the system will always approach equilibrium.

If we consider a system defined by:

$$\dot{x} = f(x),$$

A Lyapunov function $V : R^n \rightarrow R$ will have the following conditions for stability.

Positive Definiteness:

$$V(x) > 0 \quad \text{for all } x \neq 0 \quad \text{and} \quad V(0) = 0.$$

This means that the Lyapunov function is zero at the equilibrium point and positive elsewhere.

Negative Definite Derivative for Asymptotic Stability:

$$\dot{V}(x) < 0 \quad \text{for all } x \neq 0.$$

This stricter condition means the "energy" strictly decreases, ensuring that trajectories converge to the equilibrium point over time.

4.3 Pontryagins Maximum Principle

Pontryagins Maximum principle is referenced only once in the paper were authors stated that the minimum time strategies follows from this principle in the *Minimum time strategies* subsection. The Pontryagin's Maximum Principle is a concept in optimal control theory that can provide an optimal way to control the system. The principle requires that the control inputs maximize a specially defined Hamiltonian function, which encapsulates the dynamics of the system and the cost structure of the control problem.

The Hamiltonian $H(x, p)$ for an optimal control problem is typically defined as [6]:

$$H(x, p) = p \cdot f(x, u) - L(x, u)$$

where:

- x is the state vector,
- p is the costate vector (also called adjoint variables),
- $f(x, u)$ represents the system dynamics,
- $L(x, u)$ is the running cost function,
- u is the control input.

4.4 Bang-Bang control

Bang-bang control is a feed back based control strategy where the control input switches between its maximum and minimum values or between on/off (hence the term "bang-bang"). This approach has it's basis on a principle that to move from one state to another in the shortest possible time, it is necessary to utilize the maximum available control input at all times. This system also reduces the complexity in control logic [7].

4.4.1 Why Does PMP Lead to Bang-Bang Control?

Linearity in u: When $H(x, u, \lambda, t)$ is linear in u, the maximuzation condition in PMP causes the optimal control vector u^* to take values at the boundaries of its admissible range. For example, if $H = au$ where a is some coefficient, the optimal value of u will be:

$$u^* = \begin{cases} U_{max}, & \text{if } a > 0, \\ U_{min}, & \text{if } a < 0 \end{cases}$$

For many physical systems, controlling the system as aggressively as possible (by using the extreme values of u) minimizes the time or energy cost, making bang-bang control optimal. The control law often depends on a "switching function," derived from the Hamiltonian and costate equations. The sign of this function determines whether u should be at u_{min} or u_{max}

4.5 MATLAB Simulation Code

The results in the paper are replicated using MATLAB [8]. The results of the same were already discussed above. For the purpose of better understanding, the pendulum motion is visualized as an animation. The visualizations utilized are purely for reference and are not part of the discussions mentioned in the paper. Matlab's "fanimator" function to visualize the pendulum's motion and it's final upright positions for various cases mentioned in the paper.The visualizations of each case are given in their respective sections.

4.6 Code for Single Swing Double Switch (SSDS)

```
% Parameters
% Parameters
n = 2.1; % ratio of maximum acceleration to g
k = 100; % control gain
m = 0.01; % mass (kg)
l = 9.81; % length (m) adjusted for omega0 = 1
g = 9.81; % gravity (m/s^2)
J = m * l^2; % moment of inertia

% Initial conditions
theta0 = -pi; % initial angle (rad), downward position
dtheta0 = 0.0001; % initial angular velocity (rad/s)

% Desired energy level at the upright position
E0 = 0;


% Time span
simTime = 6;
tspan = [0 simTime];


% State-space model function
function dxdt = pendulumStateSpace(~, x, J, m, g, l, k, n, E0)
    theta = x(1);
    dtheta = x(2);
    % Defnining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u = min(max(u_unsat, -n*g), n*g); % Saturate control signal


    dxdt = [dtheta;
            (m * g * l * sin(theta) - m * l * u * cos(theta)) / J];
end


% Solving ODE using ode45 with state-space model
[t, x] = ode45(@(t, x) pendulumStateSpace(t, x, J, m, g, l, k, n, E0), tspan,
↪  [theta0, dtheta0]);


% Calculating initial energy
E_initial = 0.5 * J * dtheta0^2 + m * g * l * (cos(theta0) - 1);
fprintf('Initial Energy E: %.2f J\n', E_initial);


% Calculating control signal over time and print initial value
u_control_g_force = zeros(length(t), 1); % Control signal in terms of g-force
for i = 1:length(t)
    % Initializing theta variable
    theta = x(i, 1);
    % Initializing dtheta
    dtheta = x(i, 2);
    % Defining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    E = E / m*g*l ;
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u_control_g_force(i) = min(max(u_unsat / g, -n), n); % Normalize by g to express
    ↪  in terms of g-force
    % Printing the first value of control input
    if i == 1
        fprintf('Initial Control Signal: %.2f g\n', u_control_g_force(i));
    end


    % Stop applying force once the desired energy is reached
    if E >= E0 && i > 1
        u_control_g_force(i:end) = 0;
        break;
    end
end


% Plot results
figure;
subplot(3,1,1);
plot(t, x(:,1));
title('Pendulum Angle');
xlabel('Time (s)');
ylabel('\theta (rad)');


subplot(3,1,2);
E_norm = (0.5 * J * x(:,2).^2 + m * g * l .* (cos(x(:,1)) - 1)) / (m*g*l);
plot(t, E_norm);
title('Normalized Energy');
xlabel('Time (s)');
ylabel('Normalized Energy');


subplot(3,1,3);
plot(t(1:length(u_control_g_force)), u_control_g_force);
title('Control Signal in Terms of g-force');
xlabel('Time (s)');
ylabel('Control Signal (g)');


%% Simulating the results
figure;
% Assuming theta representing the pendulum angles (in radians)
theta = x(:,1);  % Actual vector of theta values
time = linspace(0, simTime, length(theta));  % Create a time vector for sim time
  ↪  seconds


% Getting the length of theta
len_theta = length(theta);


% Define x and y positions as a function of theta
x_pos = @(t) sin(interp1(time, theta, t));   % Interpolate theta for continuous time
  ↪  t
```

```
y_pos = @(t) cos(interp1(time, theta, t));  % Interpolate theta for continuous time t


% Set up animation loop
fanimator(@(t) plot(x_pos(t), y_pos(t), 'ko', 'MarkerFaceColor', 'k'),
↪  'AnimationRange', [0 simTime], 'FrameRate', len_theta/simTime);
hold on;


% Plot the line connecting the origin to the pendulum bob
fanimator(@(t) plot([0 x_pos(t)], [0 y_pos(t)], 'k-'), 'AnimationRange', [0 simTime],
↪  'FrameRate', len_theta/simTime);


% Display the timer as text (t maps to time directly)
fanimator(@(t) text(-0.3, 0.3, "Timer: " + num2str(t, 2) + " s"), 'AnimationRange',
↪  [0 simTime], 'FrameRate', len_theta/simTime);


% Display the animation
hold off;
playAnimation;
```

4.7 Code for Single Swing Triple Switch (SSTS)

```
% Parameters
n = 1.5; % ratio of maximum acceleration to g
k = 100; % control gain
m = 0.01; % mass (kg)
l = 9.81; % length (m) adjusted for omega0 = 1
g = 9.81; % gravity (m/s^2)
J = m * l^2; % moment of inertia

% Initial conditions
theta0 = -pi; % initial angle (rad), downward position
dtheta0 = 0.001; % initial angular velocity (rad/s)

% Desired energy level at the upright position
E0 = 0;

% Time span
simTime = 6;
tspan = [0 simTime];

% State-space model function
function dxdt = pendulumStateSpace(~, x, J, m, g, l, k, n, E0)
    theta = x(1);
    dtheta = x(2);
    % Defnining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u = min(max(u_unsat, -n*g), n*g); % Saturate control signal
    % State equation
    dxdt = [dtheta;
            (m * g * l * sin(theta) - m * l * u * cos(theta)) / J];
end

% Solving ODE using ode45 with state-space model
[t, x] = ode45(@(t, x) pendulumStateSpace(t, x, J, m, g, l, k, n, E0), tspan,
↪  [theta0, dtheta0]);

% Calculating initial energy
E_initial = 0.5 * J * dtheta0^2 + m * g * l * (cos(theta0) - 1);
```

```
fprintf('Initial Energy E: %.2f J\n', E_initial);

% Calculating control signal over time
u_control_g_force = zeros(length(t), 1); % Control signal in terms of g-force
for i = 1:length(t)
    % Initializing theta variable
    theta = x(i, 1);
    % Initializing dtheta
    dtheta = x(i, 2);
    % Defnining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u_control_g_force(i) = min(max(u_unsat / g, -n), n); % Normalize by g to express
     ↪  in terms of g-force
    % Printing the first value of control input
    if i == 1
        fprintf('Initial Control Signal: %.2f g\n', u_control_g_force(i));
    end

    % Stoping the applied force once the desired energy is reached
    if E >= E0 && i > 1
        u_control_g_force(i:end) = 0;
        break;
    end
end

% Plot results
figure;
subplot(3,1,1);
plot(t, x(:,1));
title('Pendulum Angle');
xlabel('Time (s)');
ylabel('\theta (rad)');

subplot(3,1,2);
E_norm = (0.5 * J * x(:,2).^2 + m * g * l .* (cos(x(:,1)) - 1)) / (m*g*l);
plot(t, E_norm);
title('Normalized Energy');
xlabel('Time (s)');
ylabel('Normalized Energy');

```

```matlab
subplot(3,1,3);
plot(t(1:length(u_control_g_force)), u_control_g_force);
title('Control Signal in Terms of g-force');
xlabel('Time (s)');
ylabel('Control Signal (g)');
ylim([-2, 2]);


%% Simulation
figure;
% Assuming theta representing the pendulum angle (in radians)
theta = x(:,1);  % Actual vector of theta values
time = linspace(0, simTime, length(theta));  % Create a time vector for sim time
↪  seconds

% Getting the length of theta
len_theta = length(theta);

% Define x and y positions as a function of theta
x_pos = @(t) sin(interp1(time, theta, t));   % Interpolate theta for continuous time
↪  t
y_pos = @(t) cos(interp1(time, theta, t));  % Interpolate theta for continuous time t

% Set up animation loop
fanimator(@(t) plot(x_pos(t), y_pos(t), 'ko', 'MarkerFaceColor', 'k'),
↪  'AnimationRange', [0 simTime], 'FrameRate', len_theta/simTime);
hold on;

% Plotting the line connecting the origin to the pendulum bob
fanimator(@(t) plot([0 x_pos(t)], [0 y_pos(t)], 'k-'), 'AnimationRange', [0 simTime],
↪  'FrameRate', len_theta/simTime);

% Display the timer as text (t maps to time directly)
fanimator(@(t) text(-0.3, 0.3, "Timer: " + num2str(t, 2) + " s"), 'AnimationRange',
↪  [0 simTime], 'FrameRate', len_theta/simTime);

% Display the animation
hold off;
playAnimation;
```

4.8 Code for Multi-Swing Behavior

```
% Parameters
n = 0.58; % ratio of maximum acceleration to g
k = 100; % control gain
m = 0.01; % mass (kg)
l = 9.81; % length (m) adjusted for omega0 = 1
g = 9.81; % gravity (m/s^2)
J = m * l^2; % moment of inertia

% Initial conditions
theta0 = -pi; % initial angle (rad), downward position
dtheta0 = 0.001; % initial angular velocity (rad/s)

% Desired energy level at the upright position
E0 = 0;

% Time span
simTime = 12;
tspan = [0 simTime];

% State-space model function
function dxdt = pendulumStateSpace(~, x, J, m, g, l, k, n, E0)
    theta = x(1);
    dtheta = x(2);
    % Defnining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u = min(max(u_unsat, -n*g), n*g); % Saturate control signal

    dxdt = [dtheta;
            (m * g * l * sin(theta) - m * l * u * cos(theta)) / J];
end

% Solving ODE using ode45 with state-space model
[t, x] = ode45(@(t, x) pendulumStateSpace(t, x, J, m, g, l, k, n, E0), tspan,
↪  [theta0, dtheta0]);

% Calculating initial energy and print it
E_initial = 0.5 * J * dtheta0^2 + m * g * l * (cos(theta0) - 1);
```

```
fprintf('Initial Energy E: %.2f J\n', E_initial);

% Calculating control signal over time and print initial value
u_control_g_force = zeros(length(t), 1); % Control signal in terms of g-force
for i = 1:length(t)
    % Initializing theta variable
    theta = x(i, 1);
    % Initializing dtheta
    dtheta = x(i, 2);
    % Defnining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u_control_g_force(i) = min(max(u_unsat / g, -n), n); % Normalizing by g to
    ↪  express in terms of g-force
    % Printing the first value of control input
    if i == 1
        fprintf('Initial Control Signal: %.2f g\n', u_control_g_force(i));
    end

    % Stoping the applied force once the desired energy is reached
    if E >= E0 && i > 1
        u_control_g_force(i:end) = 0;
        break;
    end
end

% Plot results
figure;
subplot(3,1,1);
plot(t, x(:,1));
title('Pendulum Angle');
xlabel('Time (s)');
ylabel('\theta (rad)');

subplot(3,1,2);
E_norm = (0.5 * J * x(:,2).^2 + m * g * l .* (cos(x(:,1)) - 1)) / (m*g*l);
plot(t, E_norm);
title('Normalized Energy');
xlabel('Time (s)');
ylabel('Normalized Energy');

```

UNIVERSITY OF MARYLAND

```
subplot(3,1,3);
plot(t(1:length(u_control_g_force)), u_control_g_force);
title('Control Signal in Terms of g-force');
xlabel('Time (s)');
ylabel('Control Signal (g)');
ylim([-0.7, 0.7]);

figure;
% Assuming theta representing the pendulum angles (in radians)
theta = x(:,1);  % Actual vector of theta values
time = linspace(0, simTime, length(theta));  % Create a time vector for sim time
↪  seconds

% Getting the length of theta
len_theta = length(theta);

% Define x and y positions as a function of theta
x_pos = @(t) sin(interp1(time, theta, t));   % Interpolate theta for continuous time
↪  t
y_pos = @(t) cos(interp1(time, theta, t));  % Interpolate theta for continuous time t

% Setting up animation loop
fanimator(@(t) plot(x_pos(t), y_pos(t), 'ko', 'MarkerFaceColor', 'k'),
↪  'AnimationRange', [0 simTime], 'FrameRate', len_theta/simTime);
hold on;

% Plotting the line connecting the origin to the pendulum bob
fanimator(@(t) plot([0 x_pos(t)], [0 y_pos(t)], 'k-'), 'AnimationRange', [0 simTime],
↪  'FrameRate', len_theta/simTime);

% Display the timer as text (t maps to time directly)
fanimator(@(t) text(-0.3, 0.3, "Timer: " + num2str(t, 2) + " s"), 'AnimationRange',
↪  [0 simTime], 'FrameRate', len_theta/simTime);

% Display the animation
hold off;
playAnimation;

```

4.9 Code for Five Swing Behavior

```
% Parameters
n = 0.25; % ratio of maximum acceleration to g
k = 100; % control gain
m = 0.01; % mass (kg)
l = 9.81; % length (m) adjusted for omega0 = 1
g = 9.81; % gravity (m/s^2)
J = m * l^2; % moment of inertia

% Initial conditions
theta0 = -pi; % initial angle (rad), downward position
dtheta0 = 0.001; % initial angular velocity (rad/s)

% Desired energy level at the upright position
E0 = 0;

simTime = 20;
% Time span
tspan = [0 simTime];

% State-space model function
function dxdt = pendulumStateSpace(~, x, J, m, g, l, k, n, E0)
    theta = x(1);
    dtheta = x(2);
    % Defnining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u = min(max(u_unsat, -n*g), n*g); % Saturate control signal

    dxdt = [dtheta;
            (m * g * l * sin(theta) - m * l * u * cos(theta)) / J];
end

% Solving ODE using ode45 with state-space model
[t, x] = ode45(@(t, x) pendulumStateSpace(t, x, J, m, g, l, k, n, E0), tspan,
↪  [theta0, dtheta0]);

% Calculating initial energy and print it
E_initial = 0.5 * J * dtheta0^2 + m * g * l * (cos(theta0) - 1);
```

```
fprintf('Initial Energy E: %.2f J\n', E_initial);

% Calculate control signal over time and print initial value
u_control_g_force = zeros(length(t), 1); % Control signal in terms of g-force
for i = 1:length(t)
    % Initializing theta variable
    theta = x(i, 1);
    % Initializing dtheta
    dtheta = x(i, 2);
    % Defnining Energy as per the equation 2
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    % Control equation as per the equation 8
    u_unsat = k * (E - E0) * sign(dtheta * cos(theta));
    % Applying the sat fuction in equation 8
    u_control_g_force(i) = min(max(u_unsat / g, -n), n); % Normalizing by g to
    ↪  express in terms of g-force
    % Printing the first value of control input
    if i == 1
        fprintf('Initial Control Signal: %.2f g\n', u_control_g_force(i));
    end

    % Stopping the applied force once the desired energy is reached
    if E >= E0 && i > 1
        u_control_g_force(i:end) = 0;
        break;
    end
end

% Plotting the results
figure;
subplot(3,1,1);
plot(t, x(:,1));
title('Pendulum Angle');
xlabel('Time (s)');
ylabel('\theta (rad)');

subplot(3,1,2);
E_norm = (0.5 * J * x(:,2).^2 + m * g * l .* (cos(x(:,1)) - 1)) / (m*g*l);
plot(t, E_norm);
title('Normalized Energy');
xlabel('Time (s)');
ylabel('Normalized Energy');

```

```
subplot(3,1,3);
plot(t(1:length(u_control_g_force)), u_control_g_force);
title('Control Signal in Terms of g-force');
xlabel('Time (s)');
ylabel('Control Signal (g)');
ylim([-0.5, 0.5]);

figure;
theta = x(:,1);  % Actual vector of theta values
time = linspace(0, simTime, length(theta));  % Creating a time vector for sim time in
↪  seconds

% Getting the length of theta
len_theta = length(theta);

% Defining x and y positions as a function of theta
x_pos = @(t) sin(interp1(time, theta, t));   % Interpolating theta for continuous
↪  time t
y_pos = @(t) cos(interp1(time, theta, t));  % Interpolating theta for continuous time
↪  t

% Setting up animation loop
fanimator(@(t) plot(x_pos(t), y_pos(t), 'ko', 'MarkerFaceColor', 'k'),
↪  'AnimationRange', [0 simTime], 'FrameRate', len_theta/simTime);
hold on;

% Plotting the line connecting the origin to the pendulum bob
fanimator(@(t) plot([0 x_pos(t)], [0 y_pos(t)], 'k-'), 'AnimationRange', [0 simTime],
↪  'FrameRate', len_theta/simTime);

% Displaying the timer as text (t maps to time directly)
fanimator(@(t) text(-0.3, 0.3, "Timer: " + num2str(t, 2) + " s"), 'AnimationRange',
↪  [0 simTime], 'FrameRate', len_theta/simTime);

% Display the animation
hold off;
playAnimation;

```

UNIVERSITY OF MARYLAND

4.10 Code for Minimum time strategies

```
% Parameters
% change n to 2.1 to view the other plot
n = 5; % ratio of maximum acceleration to g
k = 100; % control gain
m = 0.01; % mass (kg)
l = 9.81; % length (m) adjusted for omega0 = 1
g = 9.81; % gravity (m/s^2)
J = m * l^2; % moment of inertia

% Initial conditions
theta0 = -pi; % initial angle (rad), downward position
dtheta0 = 0.0001; % initial angular velocity (rad/s)

% Desired energy level at the upright position
E0 = 0;

% Time span
simTime = 5;
tspan = [0 simTime];


% State-space model function
function dxdt = pendulumStateSpace(~, x, J, m, g, l, n, ~, ~)
    theta = x(1);
    dtheta = x(2);
    E = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1);
    theta_star = -asin(1 - 1/n);
    % Bang-bang control logic
    if theta <= -pi/2
        u = n * g; % Apply max positive input until horizontal position
    elseif theta > -pi/2 && theta <= theta_star
        u = -n * g; % Switch to max negative input until \theta^*
    elseif theta > theta_star && theta < -0.03
        u = n * g; % Switch back to max positive input until upright
    else
        u = 0;
    end

    dxdt = [dtheta;
            (m * g * l * sin(theta) - m * l * u * cos(theta)) / J];
```

```
end

% Solving ODE using ode45 with state-space model
[t, x] = ode45(@(t, x) pendulumStateSpace(t, x, J, m, g, l, n, E0), tspan, [theta0,
↪  dtheta0]);

% Calculating initial energy
E_initial = 0.5 * J * dtheta0^2 + m * g * l * (cos(theta0) - 1);
E_initial  = E_initial/m*g*l;
fprintf('Initial Energy E: %.2f J\n', E_initial);

% Calculating control signal over time
u_control_g_force = zeros(length(t), 1); % Control signal in terms of g-force
for i = 1:length(t)
    % Initializing theta variable
    theta = x(i, 1);
    % Initializing dtheta
    dtheta = x(i, 2);
    % Defnining Energy as per the equation 2
    theta_star = -asin(1 - 1/n);

    E(i) = 0.5 * J * dtheta^2 + m * g * l * (cos(theta) - 1)/ (m*g*l);
    % Bang-bang control logic
    if theta <= -pi/2
        u = n * g; % Apply max positive input until horizontal position
    elseif theta > -pi/2 && theta <= theta_star
        u = -n * g; % Switch to max negative input until \theta^*
    elseif theta > theta_star && theta < -0.03
        u = n * g; % Switch back to max positive input until upright
    else
        u = 0; % Stop input once upright position is reached
    end

    u_control_g_force(i) = u/g;

end
theta = x(:,1);
theta(theta>0) = 0;
% Plotting results
figure;
subplot(3,1,1);
plot(t, theta);
title('Pendulum Angle');
```

```
xlabel('Time (s)');
ylabel('\theta (rad)');

subplot(3,1,2);
E_norm = (0.5 * J * x(:,2).^2 + m * g * l .* (cos(x(:,1)) - 1)) / (m*g*l);
plot(t, E_norm);
title('Normalized Energy');
xlabel('Time (s)');
ylabel('Normalized Energy');

subplot(3,1,3);
plot(t(1:length(u_control_g_force)), u_control_g_force);
title('Control Signal in Terms of g-force');
xlabel('Time (s)');
ylabel('Control Signal (g)');
ylim([-5.1,5.1])
yticks([-4, 0, 4]); % Set y-axis ticks to -4, 0, and 4

```

4.11 Code for Two pendulums on a Cart

```
% Parameters for Pendulum 1 and Pendulum 2
m1 = 0.1; % mass of Pendulum 1 (kg)
l1 = 9.81; % length of Pendulum 1 (m)
m2 = 0.335; % mass of Pendulum 2 (kg)
l2 = 9.81 / 4; % length of Pendulum 2 (m)
g = 9.81; % gravity (m/s^2)
n = 1.5; % ratio of maximum acceleration to g
k = 20; % control gain, fixed at 20

% Initial conditions for both pendulums
theta1_0 = -pi + 0.05; % initial angle of Pendulum 1 (rad)
% Adding a small value +0.05 for numerical stability
dtheta1_0 = 0.001; % initial angular velocity of Pendulum 1 (rad/s)
theta2_0 = -pi; % initial angle of Pendulum 2 (rad)
dtheta2_0 = 0.001; % initial angular velocity of Pendulum 2 (rad/s)

% Time span for simulation
simTime = 10;
tspan = [0 simTime];


% State-space model for two pendulums with control input 'u'.
function dxdt = pendulumsStateSpace(~, x, m1, l1, m2, l2, g, k, n)
    theta1 = x(1);
    dtheta1 = x(2);
    theta2 = x(3);
    dtheta2 = x(4);

    % Energy calculation of Pendulum 1 and Pendulum 2
    E1 = 0.5 * m1 * l1^2 * dtheta1^2 + m1 * g * l1 * (cos(theta1) - 1);
    E2 = 0.5 * m2 * l2^2 * dtheta2^2 + m2 * g * l2 * (cos(theta2) - 1);

    % G value from the Lyapunov derivative
    G = m1 * l1 * E1 * dtheta1 * cos(theta1) + m2 * l2 * E2 * dtheta2 * cos(theta2);

    % Control input and saturating the same
    u_unsat = k * G;
    u = min(max(u_unsat, -n * g), n * g); % Saturate control input to prevent
     ↪ instability

    % Calculating the angular velocities of both the pendulums
    dxdt = [
        dtheta1;
        (m1 * g * l1 * sin(theta1) - m1 * l1 * u * cos(theta1)) / (m1 * l1^2); %
        ↪  Pendulum 1 dynamics
        dtheta2;
        (m2 * g * l2 * sin(theta2) - m2 * l2 * u * cos(theta2)) / (m2 * l2^2); %
        ↪  Pendulum 2 dynamics
    ];
end

% Solving the ODE for the pendulum system using ODE89
options = odeset('RelTol',1e-8, 'AbsTol',1e-8);
[t, x] = ode89(@(t, x) pendulumsStateSpace(t, x, m1, l1, m2, l2, g, k, n), tspan,
↪  [theta1_0, dtheta1_0, theta2_0, dtheta2_0], options);

u_vals = zeros(size(t));
last_theta1 = theta1_0; % Initialize with initial angle
last_dtheta1 = dtheta1_0; % Initialize with initial velocity
last_theta2 = theta2_0; % Initialize with initial angle
last_dtheta2 = dtheta2_0; % Initialize with initial velocity

% Initializing the pendulum data structure to store angles
pendulum_angles = struct('theta1', zeros(size(t)), 'theta2', zeros(size(t)));
freeze_flag = false; % Flag to track when to freeze angles
pendulum_1_flag = false;  % Upright position flag for pendulum 1
pendulum_2_flag = false;  % Upright position flag for pendulum 2
not_upright_flag=true;    % Flag for not upright check
threshold = 0.15; % Threshold for freezing angles

for i = 1:length(t)
    % Extract states at the current time step
    theta1 = x(i,1);
    dtheta1 = x(i,2);
    theta2 = x(i,3);
    dtheta2 = x(i,4);

    % Storing the angles in the pendulum_angles data structure
    pendulum_angles.theta1(i) = theta1;
    pendulum_angles.theta2(i) = theta2;

    % Check if theta1 and theta2 are near 0 or a multiple of 2*pi
    if abs(mod(theta1, 2*pi)) < threshold || abs(abs(mod(theta1, 2*pi)) - 2*pi) <
    ↪  threshold
        pendulum_1_flag = true;
    end

    if abs(mod(theta2, 2*pi)) < threshold || abs(abs(mod(theta2, 2*pi)) - 2*pi) <
    ↪  threshold
        pendulum_2_flag = true;
    end

% As pointed out in the report, the control strategy discussed takes the
% pendulum to the upright position. This position is, however, not stable
% and requires a separate control strategy to catch and hold the pendulums
% near that position. However, this is out of the scope of the paper discussed,
% and we are freezing the angles at that position in the plot to showcase the same.



    % If both angles are near multiples of 2*pi, freeze them
    if pendulum_1_flag && pendulum_2_flag
        freeze_flag = true; % Set the flag to freeze angles
    end

    if freeze_flag && not_upright_flag
        pendulum_1_final = pendulum_angles.theta1(i);
        pendulum_2_final = pendulum_angles.theta2(i);
        not_upright_flag = false;
    end

    % If freeze_flag is true, stop updating angles in pendulum_angles
    if freeze_flag
        pendulum_angles.theta1(i) = pendulum_1_final; % Freeze theta1
        pendulum_angles.theta2(i) = pendulum_2_final; % Freeze theta2
    end

    % Compute energies, G value, and control input for plotting
    E1 = 0.5 * m1 * l1^2 * dtheta1^2 + m1 * g * l1 * (cos(theta1) - 1);
    E2 = 0.5 * m2 * l2^2 * dtheta2^2 + m2 * g * l2 * (cos(theta2) - 1);
    G = m1 * l1 * E1 * dtheta1 * cos(theta1) + m2 * l2 * E2 * dtheta2 * cos(theta2);
    u_unsat = k * G;
    u_vals(i) = min(max(u_unsat, -n * g), n * g); % Saturate control input to prevent
    ↪  instability
end
```

```

% Plot results
figure;

% Plotting Pendulum 1 and Pendulum 2 Angles using the pendulum_angles structure
subplot(3,1,1);
plot(t, pendulum_angles.theta1, 'r', 'LineWidth', 1.5); % Pendulum 1 angle
hold on;
plot(t, pendulum_angles.theta2, 'b', 'LineWidth', 1.5); % Pendulum 2 angle
title('Pendulum Angles');
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('Pendulum 1', 'Pendulum 2');

% Plotting the  Normalized Energy for Pendulum 1 and Pendulum 2
subplot(3,1,2);
E1_norm = (0.5 * m1 * l1^2 * x(:,2).^2 + m1 * g * l1 .* (cos(x(:,1)) - 1)) / (m1 * g
↪  * l1);
E2_norm = (0.5 * m2 * l2^2 * x(:,4).^2 + m2 * g * l2 .* (cos(x(:,3)) - 1)) / (m2 * g
↪  * l2);
plot(t, E1_norm, 'r', 'LineWidth', 1.5); % Normalized Energy for Pendulum 1
hold on;
plot(t, E2_norm, 'b', 'LineWidth', 1.5); % Normalized Energy for Pendulum 2
title('Normalized Energy');
xlabel('Time (s)');
ylabel('Normalized Energy');
legend('Pendulum 1', 'Pendulum 2');

% Plotting the Control Signal in terms of g-force
subplot(3,1,3);
plot(t, u_vals / g, 'k', 'LineWidth', 1.5); % Control signal in terms of g-force
title('Control Signal in Terms of g-force');
xlabel('Time (s)');
ylabel('Control Signal (g)');

%% Simulating the results
figure;
% Assuming theta representing the pendulum angles (in radians)
theta1 = pendulum_angles.theta1;  % First pendulum's angles
theta2 = pendulum_angles.theta2;  % Second pendulum's angles
time = linspace(0, simTime, length(theta1));  % Create a time vector for sim time
↪  seconds

```

```
% Getting the length of theta
len_theta = length(theta1);

% Define positions of the pendulums as functions of time
x1 = @(t) sin(interp1(time, theta1, t));  % x-coordinates of first pendulum
y1 = @(t) cos(interp1(time, theta1, t)); % y-coordinates of first pendulum
x2 = @(t) sin(interp1(time, theta2, t));  % x-coordinates of second pendulum
y2 = @(t) cos(interp1(time, theta2, t));  % y-coordinates of second pendulum

% Animation setup
hold on;
axis equal;
xlim([-2, 2]);
ylim([-2, 2]);

% Animate pendulums
fanimator(@(t) plot([0, x1(t)], [0, y1(t)], 'k-', 'LineWidth', 2));  % First pendulum
fanimator(@(t) plot([0, x2(t)], [0, y2(t)], 'r-', 'LineWidth', 2));  % Second
 ↪  pendulum
fanimator(@(t) plot(x1(t), y1(t), 'ko', 'MarkerFaceColor', 'k'));  % First pendulum
fanimator(@(t) plot(x2(t), y2(t), 'ro', 'MarkerFaceColor', 'r'));  % Second pendulum

% timer display
fanimator(@(t) text(-1.5, 1.5, sprintf('Time: %.2f s', t), 'FontSize', 12));

% Display animation
hold off;
playAnimation;
```

4.12 Code for the Contour plot of Energy of the pendulum

```
% Parameters
n = 2.1; % normalized maximum acceleration
k = 100; % control gain
m = 0.1; % mass (kg)
l = 9.81; % length (m) adjusted for omega0 = 1
g = 9.81; % gravity (m/s^2)
J = m * l^2; % moment of inertia

% A grid with range of values for angle and angular velocity is defined
% here.
theta_vals = linspace(-2*pi, 2*pi, 100); % Angle from -2pi to 2pi
dtheta_vals = linspace(-4, 4, 100); % Angular velocity range from -4 to 4 rad/s

% A mesh grid is created for contour plot
[Theta, Dtheta] = meshgrid(theta_vals, dtheta_vals);

% Energy is calculated here as same as given in the paper.
Energy = 0.5 * J * Dtheta.^2 + m * g * l * (cos(Theta)-1);


% Contour plot creation
figure;
contour(Theta, Dtheta, Energy, 50); % 50 contour levels are used
title('Energy of uncontrolled pendulum');
xlabel('Angle (rad)');
ylabel('Angular Velocity (rad/s)');
colorbar; % color bar for enegy values
```

UNIVERSITY OF MARYLAND

4.13 Code for numerical solution for relation between k and n value

```matlab

% Defining the range for x and the values of k
x_vals = linspace(0, pi, 1000);
k_values = 1:10;                    % Range of k values

% Initializing array to store tan(x) values for the first root for each k
first_tan_value = zeros(1, length(k_values));

for idx = 1:length(k_values)
    k = k_values(idx);

    % Defining the equation as a function handle
    equation = @(x) 2 * sin(x) - (1 + cos((2 * k - 1) * x));

    % Finding the first root by detecting sign change and using fzero
    found_root = false;
    for i = 1:length(x_vals) - 1
        if sign(equation(x_vals(i))) ~= sign(equation(x_vals(i + 1)))
            % Refining the root using fzero starting at midpoint
            root = fzero(equation, (x_vals(i) + x_vals(i + 1)) / 2);
            first_tan_value(idx) = tan(root);  % Store tan(x) for first root
            found_root = true;
            break;  % Exiting loop after finding the first root
        end
    end

    % Displaying the result for current value of k
    if found_root
        fprintf('Value of tan(x) for k = %d: %f\n', k, first_tan_value(idx));
    else
        fprintf('No root found for k = %d within the range.\n', k);
    end
end

figure;
plot(k_values, first_tan_value, 'o-', 'MarkerSize', 8);
xlabel('k');
ylabel('n value satisfying the equation');
title('n v/s k');
grid on;
```

Thank You

This marks the conclusion of this report.