Nonlinear Dynamics and Control of a Cube Robot

1st Anthony Rezkalla
Mechatronics
German University in Cairo
Cairo, Egypt
anthony.rezkalla@student.guc.edu.eg

2nd Joseph Maher
Mechatronics
German University in Cairo
Cairo, Egypt
joseph.estawro@student.guc.edu.eg

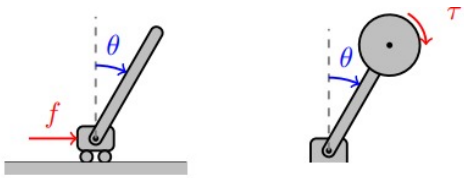
3rd Paula Magdy
Mechatronics
German University in Cairo
Cairo, Egypt
paula.kousa@student.guc.edu.eg

4th Salma Essam
Mechatronics
German University in Cairo
Cairo, Egypt
Salma.abass@student.guc.edu.eg

***Abstract*—This paper presents the design and implementation of a self-balancing cube that achieves stability while resting on a single Edge. The objective of the project is to enhance the balancing capabilities of the cube by exerting additional effort to maintain equilibrium on a single Axis. The cube incorporates sensors, actuators, and a sophisticated control system to achieve and maintain balance. The mechanical design focuses on structural integrity and compactness, while the control algorithms are developed to handle the nonlinear dynamics and real-time requirements. The cube's sensors provide accurate measurements of tilt and angular velocity, enabling the control system to generate precise control signals. The actuator system, strategically positioned within the cube, adjusts the cube's center of mass to counteract external disturbances and maintain stability.**

I. Overview of the System

The inverted pendulum system has long served as a prominent example of applying feedback control to stabilize inherently unstable open-loop systems. Initially introduced by Stephenson in 1908, the first solution to this challenge was presented by Roberge in 1960 [1]. It continues to be widely utilized for testing, demonstrating, and evaluating new control theories and concepts. Two main types of inverted pendulums exist: the cart-pole configuration, featuring a controlled cart with linear motion, and the reaction wheel configuration, where a controlled rotating wheel exchanges angular momentum with the pendulum.



(a) Cart-pole (b) Reaction wheel

Fig. 1: Inverted pendulum types

One of the most noteworthy examples of a reaction wheel-based inverted pendulum is the Cubli. Originally conceived and named in 2012 by Gajamohan and Muehlebach from the Institute for Dynamic Systems and Control at ETH Zurich, the Cubli comprises a cube with three reaction wheels mounted on perpendicular faces [2]. When positioned on its edge, it functions as a reaction wheel-based 1D inverted pendulum, while on its vertex, it operates as a reaction wheel-based 3D inverted pendulum.

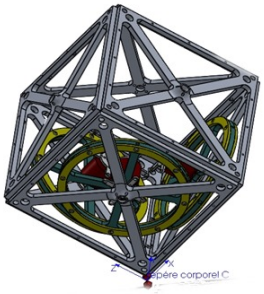


Fig. 2: Cubli positioned on its vertex

This paper aims to develop a model for the 1D configuration and subsequently design and implement a nonlinear controller for the Cubli. The Cubli's structure includes a housing, typically made of plastic or wood, which houses the reaction wheels via motors. Opting for a lightweight housing is preferred to achieve higher recovery angles. In terms of electronics, everything, except the power source, is integrated within the Cubli. A simplified diagram is depicted in Figure 3 . The main controller for the Cubli is a Raspberry Pi 4B board. IMUs (MPU6050, InvenSense) are positioned on each face of the Cubli to facilitate tilt estimation. Each IMU incorporates a rate gyro and an accelerometer, both connected to the main controller. Furthermore, a 12v 250 rpm DC motor is utilized to drive the reaction wheel.The gears of the geared motor have been removed to increase its RPM up to 800.

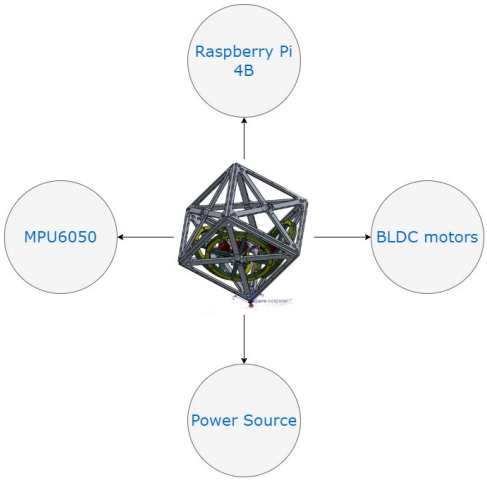


Fig. 3: Overview

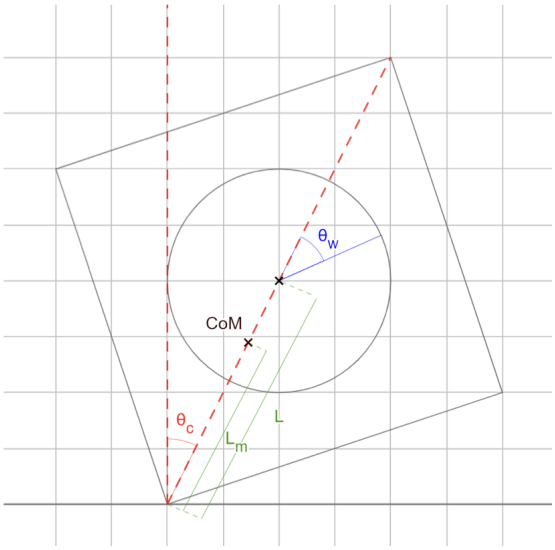


Fig. 4: Visualization

II. System Model

A. Mathematical System Model

The equation of motion is a mathematical formula that describes the motion of an object including its position, velocity and acceleration in relation to the force acting on the object. Equations of motion are vital to consider in the design of robots, in simulation and animation of robot motion, and in the design of control algorithms.

To be able to formulate the equation of motion of the Cubli, the Euler-Lagrange equation is introduced which describes the evolution of a mechanical system that is used to decrease the number of degrees of freedom of a system depending of the number of constraints made to the motion of that system. A 2D mathematical representation for the self-balancing cube must be done in order to interpret the Equation of Motion of that system. It is also necessary to examine both its initial form and its state space representation.

To start the computational process to obtain the equations of motion of the system, the moment of inertia of the system is considered to be additive moment of inertia and the generalized coordinates are θ_c and θ_w which are the defined as the tilt angle of the cube body and the rotational displacement of the reaction wheel respectively.

1) Symbol Definitions:

m_w : Mass of the Wheel
m_c : Mass of Cube
L : Length of Cube
L_m : Length to the Center of Mass
g : Gravity
$K_2\phi_u$: Motor Constant
f_c : Friction Cube
f_w : Friction Wheel
I_c : Inertia Cube
I_w : Inertia Wheel
θ_c : Angular Position of Cube
$\dot{\theta}_w$: Angular Velocity of Wheel
$\dot{\theta}_c$: Angular Velocity of Cube
$\ddot{\theta}_w$: Angular Acceleration of Wheel
$\ddot{\theta}_c$: Angular Acceleration of Cube

2) Raw Form Representation:

$$\ddot{\theta}_c = \frac{(m_w L + m_c L_m) g \sin(\theta_c) - K_2\phi_u + f_w\dot{\theta}_w - f_c\dot{\theta}_c}{I_c + m_w L^2} \tag{1}$$

$$\ddot{\theta}_w = A + B \tag{2}$$

$$A = \frac{(I_c + I_w + m_w L^2)(K_2\phi_u - f_w\dot{\theta}_w)}{I_w(I_c + m_w L^2)} \tag{3}$$

$$B = \frac{fc\dot{\theta}_c I_w - (m_c L_m + m_w L) g \sin(\theta_c) I_w}{I_w(I_c + m_w L^2)} \tag{4}$$

3) State Space Representation:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{(Lm_w+L_m mc)g}{I_c+m_w L^2} & \frac{-f_c}{I_c+m_w L^2} & \frac{-f_w}{I_c+m_w L^2} \\ \frac{-(Lm_w+L_m mc)g}{I_c+m_w L^2} & \frac{f_c}{I_c+m_w L^2} & \frac{-(I_c+I_w+m_w L^2)c_w}{I_w(I_c+m_w L^2)} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{-K_2\phi}{I_c+mwL^2} \\ \frac{(Ic+Iw+mwL^2)K_2\phi}{Iw(Ic+mwL^2)} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

B. Computational System Model (MATLAB Simulink)

The system model of the project is introduced including the design and fabrication process of the systems as well as the problems occurred within each design and their proposed solutions. The practical implementation of the setups was performed. The Cubli and Self- Balancing Stick projects were analyzed and a model for each setup was developed on Soliworks software.

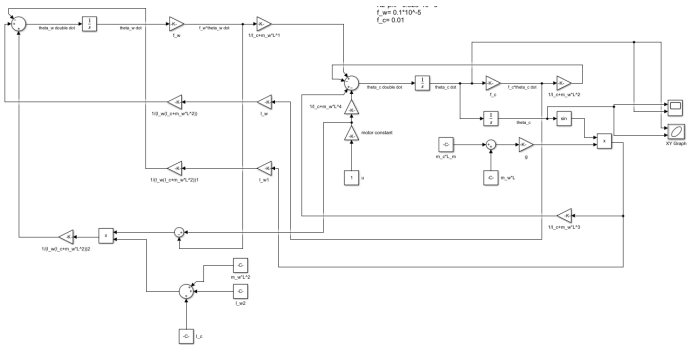


Fig. 5: Simulink Open Loop Model

Figure 5 depicts our Simulink implementation of the open-loop system model. This model was developed by representing the raw form of the system based on the equations 1 and 2.

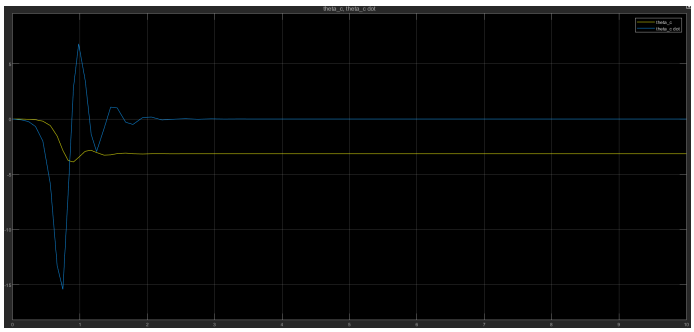


Fig. 6: Scope Visualization

In Figure **??**, the inclusion of a scope block is presented, serving the purpose of monitoring the angular position of the cube throughout time. Within this visual representation, the blue line represents $\dot{\theta}_c$, while the yellow line represents θ_c. This visual depiction facilitates the analysis of critical parameters such as settling time, rise time, peak time, and overshoot. Notably, an observation from the figure reveals that both of these signals converge towards a stable state over time. The value of θ_c is $-\pi$ which makes sense as the cube is up by angle θ_c and falls on the other side so it makes angle of $-\pi$. At the same instant the cube falls the $\dot{\theta}_c$ falls to zero.

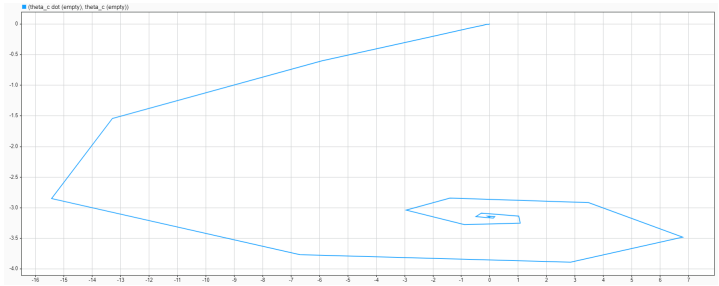


Fig. 7: XY Graph

Figure 7 presents our XY representation, where the X-axis is associated with θ_c, and the Y-axis signifies $\dot{\theta}_c$. A thorough analysis of the graph unveils the system's behavior: it initiates from the origin, subsequently follows a dynamic trajectory. For instance, if the system experiences a significant input, it may execute an upward jump maneuver. However, in the final phase, the cube falls and attains an equilibrium point where x = 0 and y = $-\pi$.

These observations shed light on the system's asymptotically unstable nature, indicating its inability to return to a stable state after experiencing perturbations.

C. Closed-loop control

Input state feedback linearization is a control technique used to make a nonlinear dynamical system behave as if it were a linear system through appropriate feedback control. In the context of a system like Cubli, which is a self-balancing cube,

input state feedback linearization helps simplify the control design by transforming the system dynamics into a linear form. In the context of Cubli, the closed-loop control system would use input-state feedback linearization to create a control law that transforms the nonlinear dynamics of the self-balancing cube into an equivalent linear system.

To perform input state feedback linearization, we need to define the following states. :

$$x1 = \theta_c \tag{5}$$

$$x2 = \dot{\theta_c} \tag{6}$$

$$x3 = \dot{\theta_w} \tag{7}$$

Thus by differentiating :

$$\dot{x1} = x2 \tag{8}$$

$$\dot{x2} = \frac{(m_w L + m_c L_m) g \sin(x1) - K_2 \phi_u + f_w x3 f_c x2}{I_c + m_w L^2} \tag{9}$$

$$\begin{aligned} \dot{x3} = &\frac{(I_c + I_w + m_w L^2)(K_2 \phi_u - f_w x3_w)}{I_w (I_c + m_w L^2)} + \\ &\frac{fcx2_c I_w - (m_c L_m + m_w L) g \sin(x2) I_w}{I_w (I_c + m_w L^2)} \end{aligned} \tag{10}$$

To determine the controllability and involutivity of this system, we utilized MATLAB. The results showed that the system is controllable and involutive. As a result, we can now proceed to find the state and input transformations. We started by obtaining the appropriate state transformation using the solution of the following PDEs:

$$\begin{aligned} 0.55\frac{\partial z1}{\partial x3} - 3\frac{\partial z1}{\partial x2} &= 0 \\ 3\frac{\partial z1}{\partial x1} - 6.4\frac{\partial z1}{\partial x2} + 7.4\frac{\partial z1}{\partial x3} &= 0 \\ 6.4\frac{\partial z1}{\partial x1} + (500\cos x1 + 26)\frac{\partial z1}{\partial x3} - (500\cos x1 + 14)\frac{\partial z1}{\partial x2} &\neq 0 \end{aligned} \tag{11}$$

These equations are solved using Maple software as shown below :

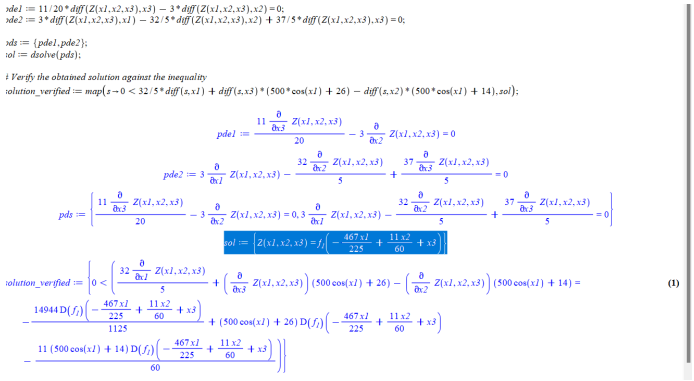


Fig. 8: Solving PDEs

Thus we can define

$$\begin{aligned} Z1 = \frac{-467}{225}x1 + \frac{11}{60}x2 + x3 \\ Z2 = \dot{Z1} \\ Z3 = \dot{Z2} \end{aligned} \tag{12}$$

Based on the information provided earlier, we can formulate our linear system in the following manner:

$$\begin{aligned} \dot{Z1} &= Z2 \\ \dot{Z2} &= Z3 \\ \dot{Z1} &= v \end{aligned} \tag{13}$$

and for tracking, we define the virtual control input v as

$$v = Z_{d1}^{(3)} - K1\tilde{\ddot{Z}1} - K2\tilde{\dot{Z}1} - K3\tilde{Z1} \tag{14}$$

Moreover, we can define the control input u as :

$$u = \alpha(x) + \beta(x)v \tag{15}$$

where $\alpha = \frac{L_f Z1}{L_g L_f^2 Z1}$ and $\beta = \frac{1}{L_g L_f^2 Z1}$

The objective of this control law is to facilitate the placement of a specific set of poles, determining the closed-loop system's behavior to ensure favorable dynamic responses. These responses encompass various critical performance metrics such as the rise Time (tr) ,settling Time (st), overshoot (pM) and peak Time (pt).For this particular scenario, a settling time of around 3 seconds and an acceptable undershoot were chosen as benchmarks. Consequently, the desired eigenvalues were estimated to be approximately -4.001, -4, and -2. The intention behind closely matching eigenvalues is to enhance the system's robustness. When eigenvalues are nearly identical, it tends to bolster the system's stability and resilience against disturbances or uncertainties. The gain values in this scenario are: $K_1 = 53.26$, $K_2 = 39.55$, and $K_3 = -16$.. The response of the system can be seen in the following:

Fig. 9: Angular Position of Cube

The system that settles near the origin exhibiting desirable behavior and indicating stability and successful control.

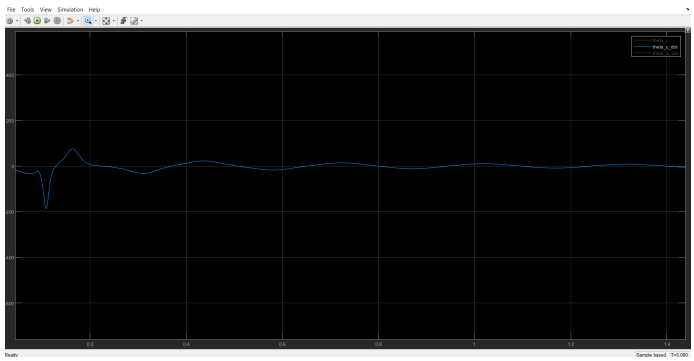


Fig. 10: Angular Velocity of Cube

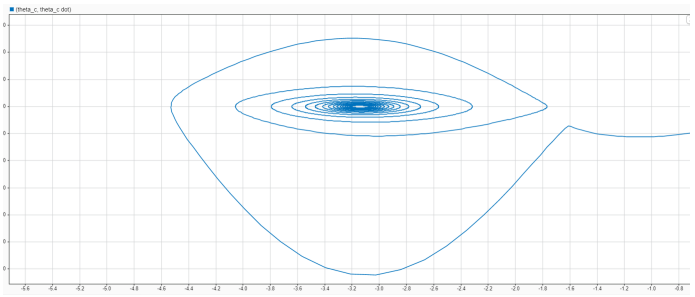


Fig. 12: θ_c vs $\dot{\theta}_c$

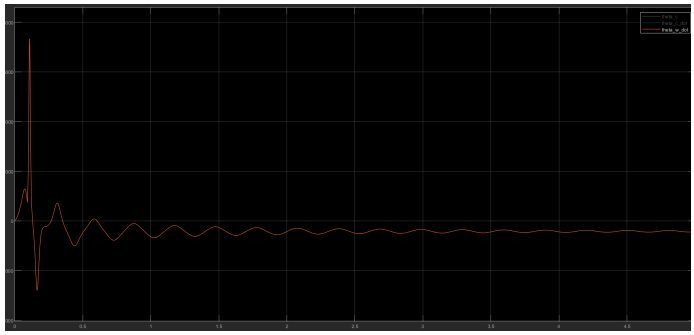


Fig. 11: Angular Velocity of Wheel

D. Hardware

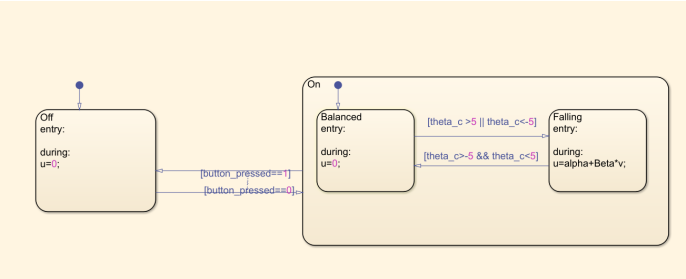
The balancing cube project presents a unique integration of hardware components and software algorithms to achieve stability and control. Crafted through 3D printing with PLA material and an infill density of 50%, the cube offers a sturdy yet lightweight structure. The propulsion system relies on a customized 12V 250 RPM DC motor, modified to enhance its rotational speed by removing gears, complemented by an H-bridge for motor control. Power is supplied through a 12V adapter, ensuring consistent and reliable operation.

The core sensing element of the cube centers around an IMU MPU 6050, responsible for accurately determining the cube's angle inclination and angular velocity. To minimize unwanted vibrations during motor operation, the sensor is strategically mounted on foam, enhancing stability and ensuring reliable sensor readings. The project utilizes two distinct code implementations: one crafted in C code within Visual Studio Code and the other generated through Simulink's C code generation feature. Both codes serve to validate and ensure consistent performance, enabling cross-verification and refinement of the control algorithms across different development environments. This holistic integration of hardware and software elements showcases a comprehensive approach to achieve a stable and controllable balancing cube system.

E. System Discrete States

1) **Balancing State:** This is the primary operational state where the system actively works to maintain the inverted pendulum in an upright position. In this state, the control algorithm continuously assesses sensor data (such as angle and velocity) to make real-time adjustments to the pendulum's position and ensure it remains balanced. The control system applies corrective actions, like motor torques or forces, to counteract any deviations from the upright position, aiming to keep the pendulum stable.
2) **Correcting˙Tilt State**:When the pendulum deviates beyond a certain threshold from the balanced position, it enters the Correcting Tilt state. Here, the control system identifies that the pendulum has tilted significantly from the upright position and takes corrective measures to bring it back within an acceptable range. This could involve applying specific control inputs or strategies tailored to correcting large deviations or disturbances, aiming to transition back to the Balancing state.
3) **Off State:** This state represents the system being turned off. In the Off state, the control algorithms are inactive, and the system ceases its balancing operations. This state might occur during system shutdown, maintenance, or when it's intentionally powered down. It's the state where no active control actions are applied to the pendulum, and the system is essentially inactive.

Each of these states plays a crucial role in the operation



and control of the inverted pendulum system. The transitions between these states are triggered predefined thresholds or user commands, and the control system's response aims to maintain stability and functionality throughout these states.

REFERENCES

[1] Kent H. Lundberg and James K. Roberge, *Classical dual-inverted-pendulum control*, in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 5, pp. 4399–4404, 2003.

[2] Mohanarajah Gajamohan, Michael Merz, Igor Thommen, and Raffaello D'Andrea, *The cubli: A cube that can jump up and balance*, in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3722–3727, 2012.

[3] Johannes Mayr, Franz Spanlang, and Hubert Gattringer, *Mechatronic design of a self-balancing three-dimensional inertia wheel pendulum*, *Mechatronics*, vol. 30, pp. 1–10, 2015, publisher=Elsevier.

[4] GIDLÖF, TIM and GRUNEAU, CARL *Balancing Cube* year 2020

[5] Chen, Zhigang and Ruan, Xiaogang and Li, Yuan, *Dynamic modeling of a cubical robot balancing on its corner*, *MATEC Web of Conferences* volume=139, pages=00067, year=2017, organization=EDP Sciences