2025

Dynamics and Control of Reaction Wheel Inverted Pendulum

THEORY AND PRACTICE

VLADIMIR DANILOV

Table of Content

Introduction

Certain physical systems exhibit unstable behavior in their desired operating mode when left uncontrolled. A classic example of such systems includes aerial vehicles in which the center of pressure is located ahead of the center of mass. These vehicles are characterized by high maneuverability, but at the cost of being statically unstable. To maintain stability, an active control system is required.

A particularly challenging task in control theory arises when designing control systems for objects where the number of control inputs is less than the number of degrees of freedom. Such systems are said to be *underactuated*. Examples include aerial vehicles, most automobiles, and inverted pendulums [1].

An inverted pendulum is a pendulum whose center of mass lies above its pivot point, typically positioned at the end of a rigid rod. Unlike a conventional pendulum, which naturally settles into a stable downward position, the inverted pendulum is inherently unstable and must be actively balanced by a control system to remain upright. Several types of inverted pendulums exist, and some of the most notable variations are described below.

Classic example is the cart-pendulum system (Fig. 1). In this configuration, the pendulum is mounted on a cart that can move along a horizontal axis. The objective is to stabilize the pendulum by moving the cart appropriately. It is well known that one can balance a vertical stick on the palm of their hand by moving the hand horizontally – especially if the stick is quite long and has most of its mass concentrated at the top. There are also multi-link variations of this system involving two, three, or more interconnected rods (Fig. 2).

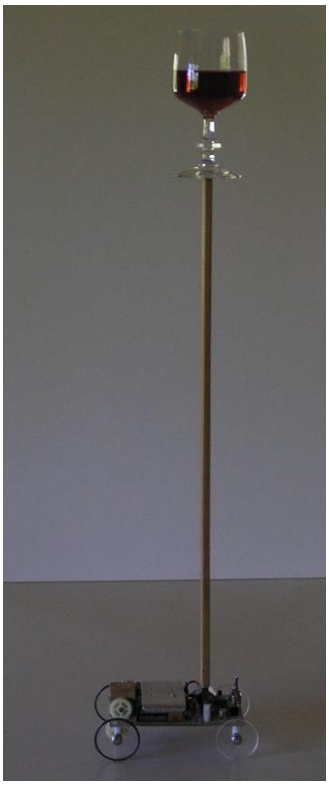

Figure 1. Pendulum on a Cart.

Another common model is the Furuta pendulum, where the pendulum is stabilized through the rotational motion of a rotating base (Fig. 3).

There also exist pendulums with a fixed base, in which stabilization is achieved via a reaction wheel mounted at the end of the rod and driven by an electric motor (Fig. 4). The motion of such a pendulum is restricted to a single vertical plane. However, the same principle can be extended to achieve balance in multiple planes – for example, to stabilize a moving stick or even a cube (Figs. 5 and 6). In fact, such a cube can not only maintain balance but also move through space.

Below is a list of video links demonstrating the operation of the described pendulum systems:

- Linear inverted pendulum
- Linear inverted pendulum with three links
- Furuta pendulum
- Flywheel-based inverted pendulum
- Balancing stick
- Balancing cube

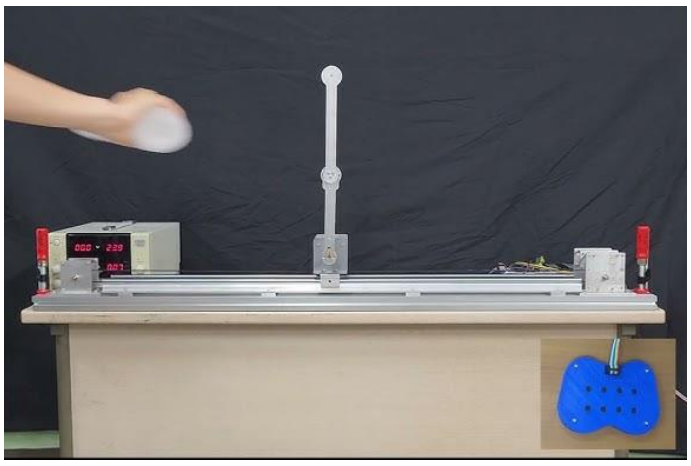

Figure 2. Two-link inverted pendulum

Figure 3. Furuta Pendulum.

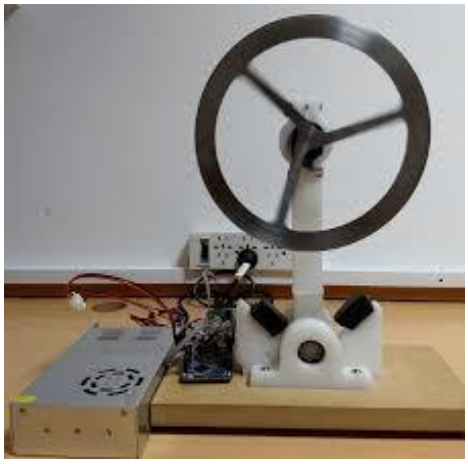

Figure 4. Reaction Wheel Inverted Pendulum.

Figure 5. Balancing Stick.

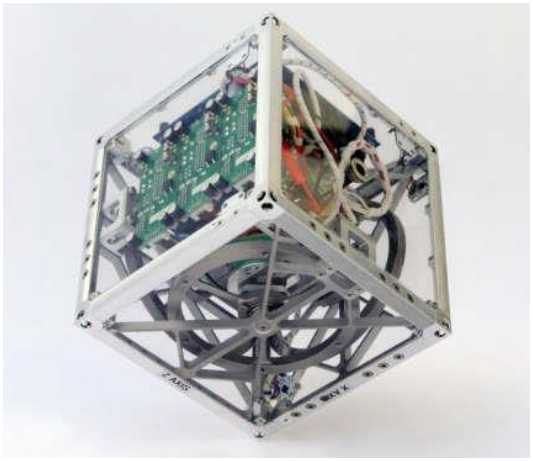

Figure 6. Balancing Cube.

Although the inverted pendulum is widely used as a benchmark system in control theory education and research, its underlying principles are fundamental to many real-world applications. For instance, the orientation of artificial Earth satellites is achieved using reaction wheels installed within the satellite's body. Inverted pendulum dynamics are also employed in personal transportation devices such as the Segway, electric unicycles, and hoverboards.

Even the process of two-legged walking can be modeled using the inverted pendulum concept [2]. In the case of bipedal locomotion, one can imagine the foot of the supporting leg acting as the

pivot point of the pendulum, while the torso represents the center of mass at the end of the rod (Fig. 7).

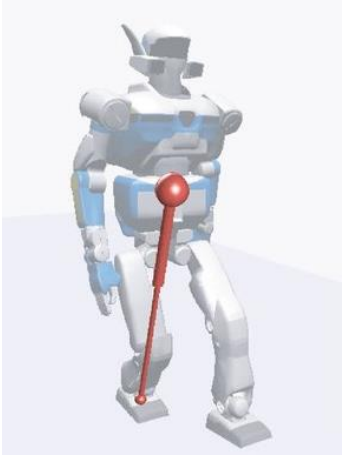

Figure 7. Three-dimensional linear inverted pendulum model used in the locomotion control system of a bipedal robot.

The objective of this lab exercise is to develop control algorithms for a reaction wheel inverted pendulum. The desired modes of motion to be implemented include swinging up the pendulum and transferring it to the upright, inherently unstable equilibrium position, as well as stabilizing it in that position. The control laws required to achieve these behaviors are synthesized in the form of feedback.

Before proceeding with this laboratory assignment, please ensure the following requirements are met:

- **Theoretical knowledge**
  You are expected to have a basic understanding of theoretical mechanics and control theory. If you encounter difficulties in understanding the material, refer to the recommended textbooks [3] and [5].
- **Practical skills**
  Familiarity with the Arduino IDE is required to complete the hardware-related part of the assignment.
- **Software installation**
  All necessary software must be installed and configured in accordance with the instructions provided in: *Tutorials/Eng_Installation_Guide.docx*

1. Structure of the Reaction Wheel Inverted Pendulum

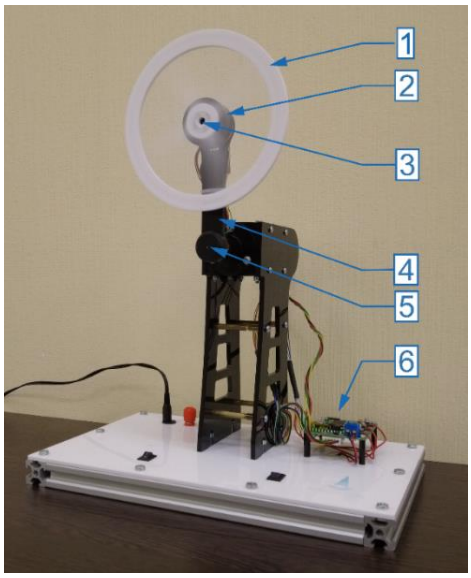

Рисунок 8. Main components of the inverted pendulum. 1 – reaction wheel, 2 – BLDC-motor, 3 – reaction wheel rotation axis, 4 – rod, 5 – rod rotation axis, 6 – control board.

Figure 8 shows a single-link pendulum with a rod (4) and a reaction wheel (1). The pendulum is capable of rotational motion within a vertical plane. Its axis of rotation (5) is fixed to an immobile base and is not actuated by any motor.

The rotation axis of the reaction wheel (3) is mounted on the rod and is parallel to the rod's own axis of rotation. The wheel is driven by a brushless DC motor (2). Both the motor and the wheel are mounted directly on the rod. The motor stator is rigidly fixed to the rod, while the rotor shaft is rigidly coupled to the wheel's axis.

The control board (6) includes a microcontroller, a motor driver, and a three-phase power amplifier. The control system operates based on measurements of two angular positions: the angle of the rod relative to the fixed base, and the angle of the wheel relative to the rod. These angles are measured by magnetic rotary encoders connected to the control board.

2. Theoretical Background

2.1. Dynamics of the Pendulum with a Reaction Wheel

Figure 9 presents a schematic of the pendulum system. The rod *OB* is connected to a fixed base at point *O* via a revolute joint. The axis of this joint is perpendicular to the plane of the pendulum's motion. A symmetric reaction wheel, balanced about its own axis of rotation, is mounted at the end of the rod at point *B*. The wheel can rotate in either direction about a horizontal axis passing through point *B*, which is perpendicular to the pendulum's plane of motion and parallel to the joint axis at point *O*. The reaction wheel's axis of rotation coincides with the rotor axis of the motor.

All relevant parameters and notations are summarized in Table 1. The system has two degrees of freedom. The only control input available is the torque generated by the electric motor.

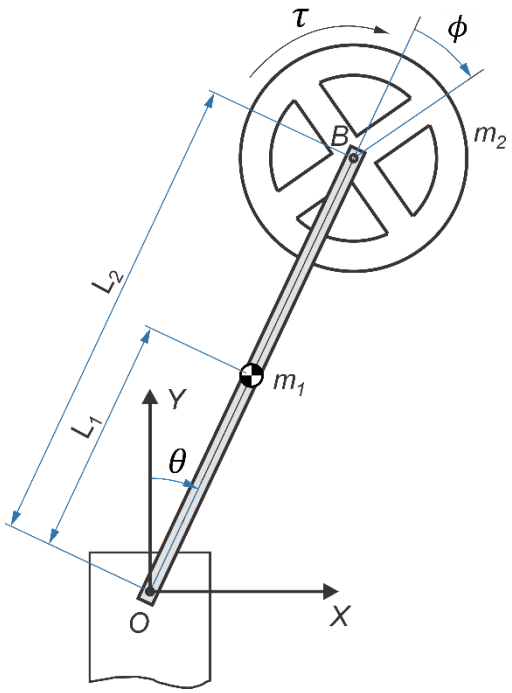

Рисунок 9. Schematics of the Inverted Pendulum.

Table 1. Pendulum Parameters.

Параметр	Ед. изм.	Описание
L_1	м	Distance from point *O* to the center of mass of the rod
L_2	м	Distance from point *O* to the reaction wheel
m_1	кг	Mass of the rod
m_2	кг	Mass of the reaction wheel
θ	рад	Angular displacement of the rod
ϕ	рад	Angular displacement of the flywheel
I_1	кг/м^2	Inertia of the rod
I_2	кг/м^2	Inertia of the flywheel
g	м/с^2	Gravity acceleration
τ	Н м	Torque produced by the electric motor
μ_p	-	Viscous friction coefficient at hinge *O*
μ_m	-	Viscous friction coefficient between the motor stator and rotor

The motor is actuated using Field Oriented Control (FOC) algorithm. In practice, we apply a desired voltage input – specifically, the so-called quadrature voltage in FOC terminology – thereby emulating the behavior of a brushed DC motor. As a result, the control input for the pendulum system will also be the voltage. To accurately relate the desired torque to this control input, the pendulum model must be extended to include a model of a brushed DC motor.

The dynamic model will be constructed using the Lagrange method. This method is one of the classical approaches for deriving the equations of motion of mechanical systems by means of the Lagrange equations of the second kind. Unlike Newton's second law, the Lagrange method operates with energy quantities: kinetic and potential energy.

The essence of the method is as follows: for a system with n generalized coordinates $\eta_1, \eta_2, \dots, \eta_n$, the so-called Lagrangian $L = T - U$ is defined, where T is the kinetic energy and U is the potential energy of the system. Then, for each coordinate η_i, the Lagrange equation of the second kind is written as:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\eta}_i}\right) - \frac{\partial L}{\partial \eta_i} = Q_i,$$

where Q_i denotes the generalized force.

A detailed study of the principles of deriving dynamic equations can be found in [3]. We recall the general algorithm for model construction using the Lagrangian method:

1. Select generalized coordinates — a minimal set of independent variables that completely describe the configuration of the system.
2. Determine the generalized forces.
3. Calculate the kinetic energy T as the sum of the energies of all moving masses, including rotational motion.
4. Calculate the potential energy U, which usually depends on the position within a gravitational field.
5. Construct the Lagrangian $L = T - U$.
6. Apply the Lagrange equations of the second kind to obtain a system of differential equations, accounting for generalized forces Q_i in the presence of external control inputs or friction.

The Lagrangian method is a powerful tool for modeling dynamics, especially in problems where energy relations and constraints between bodies play a significant role.

After deriving the system of nonlinear differential equations, it is necessary to linearize it, then represent it in state-space form, and finally convert it to a discrete form. This is motivated by the fact that analysis and control synthesis are simpler when the system dynamics are described in this form.

Let us proceed to build the dynamic model of the pendulum according to the algorithm outlined above. First, we choose the generalized coordinates θ and ϕ, which fully describe the system's configuration.

Three forces act on the system: the viscous friction force $\mu_p\dot{\theta}$ at the fixed hinge point O, the viscous friction force $\mu_m\dot{\phi}$ between the motor stator and rotor, and the torque τ generated by the electric motor. Thus, the vector of generalized forces is expressed as

$$\boldsymbol{Q} = \begin{bmatrix} -\mu_p\dot{\theta} & \tau - \mu_m\dot{\phi} \end{bmatrix}^T.$$

Next, we calculate the total kinetic energy of the pendulum by summing the kinetic energies of each link according to the equation:

$$T = \sum_{i=1}^{2} T_i,$$

where $T_i = \frac{1}{2} m_i \dot{x}_i^2 + \frac{1}{2} m_i \dot{y}_i^2 + \frac{1}{2} I_i \dot{\alpha}_i^2$ - is the general expression for kinetic energy, m_i is the mass of the i-th link, I_i is the moment of inertia of the i-th link, x_i and y_i are the coordinates of the center of mass of the i-th link along the X and Y axes, respectively, and α_i is the angular displacement of the i-th link.

In the case of our pendulum:

$$x_1 = L_1 \cos\theta\,; \quad y_1 = L_1 \sin\theta\,; \quad \alpha_1 = \theta; \tag{1}$$

$$x_2 = L_2 \cos\theta\,; \quad y_2 = L_2 \sin\theta\,; \quad \alpha_2 = \phi. \tag{2}$$

Their time derivatives take the following form:

$$\dot{x}_1 = -L_1 \dot{\theta} \sin\theta\,; \quad \dot{y}_1 = L_1 \dot{\theta} \cos\theta\,; \quad \dot{\alpha}_1 = \dot{\theta}; \tag{3}$$

$$\dot{x}_2 = -L_2 \dot{\theta} \sin\theta\,; \quad \dot{y}_2 = L_2 \dot{\theta} \cos\theta\,; \quad \dot{\alpha}_2 = \dot{\theta} + \dot{\phi}. \tag{4}$$

The expression $\dot{\alpha}_2 = \dot{\theta} + \dot{\phi}$ follows from the Velocity Transformation Law, which states:

Velocity Transformation Law

The velocity of a point in one frame is equal to the velocity of the frame itself plus the velocity of the point relative to the frame:

$$\mathbf{v}_a = \mathbf{v}_e + \mathbf{v}_r,$$

where $\mathbf{v}_a$ is the vector of absolute velocity, $\mathbf{v}_e$ is the vector of frame velocity, and $\mathbf{v}_r$ is the vector of point velocity relative to the frame.

In our case, the reaction wheel rotates relative to the rod. Therefore, its angular velocity is given by $\dot{\theta} + \dot{\phi}$. A more detailed discussion of complex point motion in space can be found in [3].

Let us now construct the kinetic energy equation of the pendulum by substituting the previously obtained expressions (3–4) into the general formula:

$$T = \frac{1}{2}\Big(m_1 L_1^2 \dot{\theta}^2 \sin^2\theta + m_1 L_1^2 \dot{\theta}^2 \cos^2\theta + I_1 \dot{\theta}^2 + m_2 L_2^2 \dot{\theta}^2 \sin^2\theta + m_2 L_2^2 \dot{\theta}^2 \cos^2\theta + I_2\big(\dot{\theta} + \dot{\phi}\big)^2\Big).$$

Simplifying the expression yields:

$$T = \frac{1}{2}(m_1 L_1^2 + m_2 L_2^2 + I_1 + I_2)\dot{\theta}^2 + I_2 \dot{\theta}\dot{\phi} + \frac{1}{2} I_2 \dot{\phi}^2. \tag{5}$$

Similarly, the potential energy of the pendulum is equal to the sum of the potential energies of the rod and the flywheel:

$$U = m_1 g y_1 + m_2 g y_2.$$

Substituting expressions (1–2), we obtain:

$$U = (m_1 L_1 + m_2 L_2) g \cos\theta\,. \tag{6}$$

For simplicity in further calculations, we introduce the following variable substitutions: let $a = m_1 L_1^2 + m_2 L_2^2 + I_1$ and $b = (m_1 L_1 + m_2 L_2) g$. Then equations (5) and (6) take the form:

$$T = \frac{1}{2}(a + I_2)\dot{\theta}^2 + I_2 \dot{\theta}\dot{\phi} + \frac{1}{2} I_2 \dot{\phi}^2$$

$$U = b \cos\theta.$$

Now we construct the Lagrangian:

$$L = T - U = \frac{1}{2}(a + I_2)\dot{\theta}^2 + I_2 \dot{\theta}\dot{\phi} + \frac{1}{2} I_2 \dot{\phi}^2 - b \cos\theta\,. \tag{7}$$

We now write the Lagrange equations of the second kind, taking into account the generalized coordinates θ, ϕ and the vector of generalized forces $\boldsymbol{Q}$:

$$\begin{cases} \dfrac{d}{dt}\left(\dfrac{\partial L}{\partial \dot{\theta}}\right) - \dfrac{\partial L}{\partial \theta} = -\mu_p \dot{\theta} \\ \dfrac{d}{dt}\left(\dfrac{\partial L}{\partial \dot{\phi}}\right) - \dfrac{\partial L}{\partial \phi} = \tau - \mu_m \dot{\phi} \end{cases} \tag{8}$$

We compute the necessary derivatives of the Lagrangian (7):

$$\frac{\partial L}{\partial \theta} = b \sin\theta\,; \quad \frac{\partial L}{\partial \dot{\theta}} = (a + I_2)\dot{\theta} + I_2 \dot{\phi}; \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = (a + I_2)\ddot{\theta} + I_2 \ddot{\phi};$$

$$\frac{\partial L}{\partial \phi} = 0; \qquad \frac{\partial L}{\partial \dot{\phi}} = I_2 \dot{\theta} + I_2 \dot{\phi}; \qquad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) = I_2 \ddot{\theta} + I_2 \ddot{\phi}.$$

Substituting these expressions into system (8), we obtain:

$$\begin{cases} (a + I_2)\ddot{\theta} + I_2 \ddot{\phi} - b \sin\theta = -\mu_p \dot{\theta} \\ I_2 \ddot{\theta} + I_2 \ddot{\phi} = \tau - \mu_m \dot{\phi}. \end{cases}$$

We rearrange the equations so that $\ddot{\theta}$ and $\ddot{\phi}$ appear on the left-hand side:

$$\begin{cases} \ddot{\theta} = -\dfrac{\mu_p}{a}\dot{\theta} + \dfrac{\mu_m}{a}\dot{\phi} + \dfrac{b}{a}\sin\theta - \dfrac{1}{a}\tau \\ \ddot{\phi} = -\dfrac{\mu_m}{I_2}\dot{\phi} + \dfrac{\mu_p}{a}\dot{\theta} - \dfrac{\mu_m}{a}\dot{\phi} - \dfrac{b}{a}\sin\theta + \left(\dfrac{1}{I_2} + \dfrac{1}{a}\right)\tau. \end{cases} \tag{9}$$

The derived system of differential equations describes the dynamics of the mechanical part of the inverted pendulum. As mentioned at the beginning of this section, we will convert the torque generated by the motor into an equivalent voltage input. In this way, the control system will treat the motor as a brushed DC motor and will use the applied voltage as the control input.

The relationship between the input voltage and the output shaft torque of the motor—neglecting gear reduction and friction—is described by the following equations:

$$V = L_m \frac{di}{dt} + R_m i + K_e \dot{\phi},$$

$$\tau = K_t i.$$

All parameters and variables are listed in Table 2. For more details on how these equations are derived, refer to textbook [4].

Table 2. Electric Motor Parameters.

Параметр или переменная	Ед. измерения	Описание
V	В	Motor voltage
i	А	Motor current
L_m	Гн	Motor inductance
R_m	Ом	Coil resistance
K_e	рад/с/В	Back-EMF constant
K_t	Н м/А	Torque constant

The parameters L_m, R_m, K_e, K_t are typically specified in the motor datasheet or can be determined experimentally. In most cases, the inductance of the windings is significantly smaller than their resistance. Therefore, the inductance term L_m can be neglected. Under this assumption, the relationship between the torque at the shaft and the input voltage simplifies to:

$$\tau = \frac{K_t\left(V - K_e\dot{\phi}\right)}{R_m}.$$

Substituting this expression into system (9), we obtain:

$$\begin{cases} \ddot{\theta} = \dfrac{b}{a}\sin\theta - \dfrac{\mu_p}{a}\dot{\theta} + \dfrac{R_m\mu_m + K_tK_e}{aR_m}\dot{\phi} - \dfrac{K_t}{aR_m}V \\ \ddot{\phi} = -\dfrac{b}{a}\sin\theta + \dfrac{\mu_p}{a}\dot{\theta} - \dfrac{a+I_2}{aI_2}\left(\mu_m + \dfrac{K_tK_e}{R_m}\right)\dot{\phi} - \dfrac{K_t(a+I_2)}{aI_2R_m}V. \end{cases} \tag{10}$$

This yields a complete mathematical model that describes the behavior of the inverted pendulum system. To perform control synthesis and stability analysis, we must linearize the system and express it in state-space form.

We linearize system (10) around the equilibrium point $\theta = 0$. Near this point, $\sin\theta \approx \theta$, and the system simplifies to:

$$\begin{cases} \ddot{\theta} = \dfrac{b}{a}\theta - \dfrac{\mu_p}{a}\dot{\theta} + \dfrac{R_m\mu_m + K_tK_e}{aR_m}\dot{\phi} - \dfrac{K_t}{aR_m}V \\ \ddot{\phi} = -\dfrac{b}{a}\theta + \dfrac{\mu_p}{a}\dot{\theta} - \dfrac{a+I_2}{aI_2}\left(\mu_m + \dfrac{K_tK_e}{R_m}\right)\dot{\phi} - \dfrac{K_t(a+I_2)}{aI_2R_m}V. \end{cases}$$

We now express the system in state-space form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}. \tag{11}$$

where the state vector is defined as $\mathbf{x} = [\theta \quad \dot{\theta} \quad \dot{\phi}]^{\mathrm{T}}$, and the control input is $\mathbf{u} = [V]$. The system becomes:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \dfrac{b}{a} & -\dfrac{\mu_p}{a} & \dfrac{R_m\mu_m + K_tK_e}{aR_m} \\ -\dfrac{b}{a} & \dfrac{\mu_p}{a} & -\dfrac{a+I_2}{aI_2}\left(\mu_m + \dfrac{K_tK_e}{R_m}\right) \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ -\dfrac{K_t}{aR_m} \\ -\dfrac{K_t(a+I_2)}{aI_2R_m} \end{bmatrix} V \tag{12}$$

We now convert equation (12) to discrete-time form. There are multiple methods for discretizing continuous systems. Here we use the Euler method. This method is straightforward but less accurate for large sampling intervals. To ensure sufficient accuracy, the time step Δt should be approximately 1 ms.

We approximate the time derivative as:

$$\dot{\mathbf{x}} = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta t},$$

where Δt is the sampling time and $\mathbf{x}_k$ is the state at time step k.
Substituting into equation (11):

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta t} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k.$$

Multiplying both sides by Δt and rearranging:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{A}\mathbf{x}_k\Delta t + \mathbf{B}\mathbf{u}_k\Delta t.$$

This can be written as:

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k \tag{13}$$

where the discrete-time matrices $\mathbf{A}_d$ and $\mathbf{B}_d$are defined as:

$$\mathbf{A}_d = \mathbf{I}_n + \mathbf{A}\Delta t, \tag{14}$$

$$\mathbf{B}_d = \mathbf{B}\Delta t, \tag{15}$$

and $\mathbf{I}_n$ is the identity matrix of size $n \times n$. Substituting into (12), the final discrete-time system is:

$$\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \\ \dot{\phi}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 \\ \frac{b}{a}\Delta t & 1 - \frac{\mu_p}{a}\Delta t & \frac{R_m\mu_m + K_tK_e}{aR_m}\Delta t \\ -\frac{b}{a}\Delta t & \frac{\mu_p}{a}\Delta t & 1 - \frac{a + I_2}{aI_2}\left(\mu_m + \frac{K_tK_e}{R_m}\right)\Delta t \end{bmatrix} \begin{bmatrix} \theta_k \\ \dot{\theta}_k \\ \dot{\phi}_k \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{K_t}{aR_m}\Delta t \\ -\frac{K_t(a + I_2)}{aI_2R_m}\Delta t \end{bmatrix} V_k. \tag{16}$$

2.2. Pendulum Stabilization in the Upright Unstable Equilibrium Position

Let us consider the problem of stabilizing the pendulum in its upright, inherently unstable equilibrium position, defined by $\theta = 0$, $\dot{\theta} = 0$, under the assumption that at the start of the stabilization process, the system is already within a neighborhood of this desired state. This problem can be addressed using a full-state feedback controller.
The block diagram of such a control system is shown in Figure 10, where **K** denotes the state feedback gain matrix. Note that the entire state vector is used for feedback, and the control input is formed according to the following law:

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k. \tag{17}$$

Thus, the design task reduces to choosing the feedback gain matrix **K** such that the closed-loop system ensures stable maintenance of the pendulum in the upright vertical position.

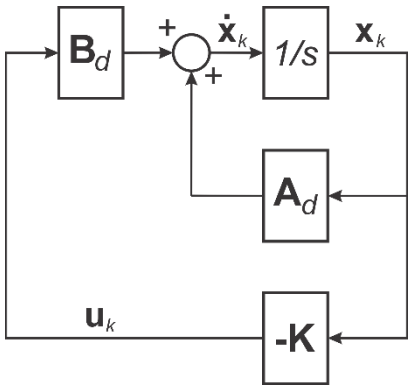

Figure 10. Schematic diagram of the control system with full-state feedback controller.

There are many approaches to selecting state feedback gains, or in other words, methods for control *synthesis*. In this work, we focus on a solution based on Linear Quadratic Regulation (LQR). Throughout the remainder of the text, the controller designed using this approach will be referred to as LQR, short for Linear Quadratic Regulator.

LQR is a type of optimal controller that minimizes a quadratic cost functional of the form:

$$J = \int_0^\infty (\mathbf{x}^\mathrm{T}\mathbf{Q}\mathbf{x} + \mathbf{u}^\mathrm{T}\mathbf{R}\mathbf{u})\,\Delta t,$$

where $\mathbf{Q}$, $\mathbf{R} \in \mathbb{R}^{n\times n}$ are user-defined, symmetric positive definite **weighting matrices**. The elements of matrix $\mathbf{Q}$ determine how strongly deviations of the state vector from zero are penalized: larger values lead to faster convergence to the desired state. Conversely, the elements of matrix $\mathbf{R}$ penalize control effort: larger values result in smoother control actions. If the elements of $\mathbf{Q}$ are too small, the system may tolerate large steady-state errors; if the elements of $\mathbf{R}$ are too small, the controller may behave too aggressively. Thus, the cost functional represents a trade-off between deviation from the desired state and the energy expended on control.

Weight matrices $\mathbf{Q}$ and $\mathbf{R}$ are typically selected based on the following principles:

- Begin with diagonal identity matrices:

$$\mathbf{Q} = \mathrm{diag}(q_1, q_2, \dots, q_n), \quad \mathbf{R} = \mathrm{diag}(r_1, r_2, \dots, r_n).$$

- Larger values of q_i increase the penalty for deviations of the corresponding state variable x_i .
- Larger values of r_i increase the penalty on the control signal u_i, resulting in smoother control behavior.

In our case, we use the discrete-time system from equation (16). The cost functional for a discrete system is given by:

$$J = \sum_{k=0}^{\infty} \left(\mathbf{x}_k^\mathrm{T}\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^\mathrm{T}\mathbf{R}\mathbf{u}_k\right)\Delta t.$$

The negative feedback law defined in equation (17) must minimize this cost functional. According to the results presented in [5], the optimal gain matrix $\mathbf{K}$ is computed as:

$$\mathbf{K} = \left(\mathbf{R} + \mathbf{B}_d^\mathrm{T}\mathbf{P}\mathbf{B}_d\right)^{-1}\mathbf{B}_d^\mathrm{T}\mathbf{P}\mathbf{A}_d, \tag{18}$$

where $\mathbf{P}$ is a symmetric positive definite matrix that solves the **discrete algebraic Riccati equation (DARE)**:

$$\mathbf{P} = \mathbf{A}_d^\mathrm{T}\mathbf{P}\mathbf{A}_d - \mathbf{A}_d^\mathrm{T}\mathbf{P}\mathbf{B}_d\left(\mathbf{R} + \mathbf{B}_d^\mathrm{T}\mathbf{P}\mathbf{B}_d\right)^{-1}\mathbf{B}_d^\mathrm{T}\mathbf{P}\mathbf{A}_d + \mathbf{Q}.$$

This equation has a unique solution **P** if the system is **controllable** and the weighting matrices **Q** and **R** are positive definite.

A system is said to be controllable if, for any initial state $\mathbf{x}_0$ and any desired final state $\mathbf{x}_f$, there exists an admissible control signal **u** that transfers the system from $\mathbf{x}_0$ to $\mathbf{x}_f$ in a finite time interval $[t_0, t_f]$.

Controllability Criterion for Linear Time-Invariant Systems

A linear time-invariant (LTI) system is controllable if and only if the controllability matrix

$$\mathbf{C} = [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \quad \ldots \quad \mathbf{A}^{n-1}\mathbf{B}]$$

has full rank, that is, its rank is equal to n, the dimension of the state vector.

Recall that the rank of a matrix is defined as the number of linearly independent rows, the number of linearly independent columns, or equivalently, the order of the largest non-zero minor.

After synthesizing the control law, it is essential to verify that the closed-loop system is stable. To do this, we apply the stability criterion for discrete-time systems.

Stability Criterion for Discrete-Time Systems

A discrete-time system is stable if and only if all its eigenvalues lie strictly inside the unit circle in the complex plane, i.e., within a circle of radius one centered at the origin.

Let us examine the meaning of this criterion in more detail. We consider a closed-loop system of the form:

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k \tag{19}$$

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k. \tag{20}$$

Substituting equation (20) into equation (19) and factoring out $\mathbf{x}_k$, we obtain:

$$\mathbf{x}_{k+1} = (\mathbf{A}_d - \mathbf{B}_d\mathbf{K})\mathbf{x}_k.$$

According to the stability criterion, all eigenvalues λ_i of the matrix $(\mathbf{A}_d - \mathbf{B}_d\mathbf{K})$ must satisfy $|\lambda_i| < 1$. If at least one eigenvalue satisfies $|\lambda_i| > 1$, the system is unstable. A case where one eigenvalue satisfies $|\lambda_j| = 1$ while all others satisfy $|\lambda_i| < 1$corresponds to the boundary of stability for a discrete-time system.

To solve the algebraic Riccati equation, iterative methods such as Newton's method are commonly used. In practice, the equation is most often solved using tools available in MATLAB or Control Systems Library in Python. The same tools are then used to compute the matrix **K** and to analyze the controllability and stability of the system. Once computed, the gain matrix **K** is applied in the control of the hardware platform. In the case of our inverted pendulum, **K** is a vector, as the system has only one control input.

The stabilization algorithm described in this section can be summarized as follows:

7. Obtain the sensor readings for θ and ϕ.
8. Compute the angular velocities $\dot{\theta}$ and $\dot{\phi}$ by subtracting the previous sensor readings from the current ones and dividing by the time interval between measurements. For example, the angular velocity of the rod is computed as:

$$\dot{\theta}_k = \frac{\theta_k - \theta_{k-1}}{\Delta t}.$$

9. Compute the desired motor voltage:

$$V = -\mathbf{K}\mathbf{x}_k = -\big(k_1\theta + k_2\dot{\theta} + k_3\dot{\phi},\big),$$

where k_i is i-th element of the vector **K**.

10. Send the computed value V to the motor control system.

2.3. **Swing-Up Control**

Initially, the pendulum rests in the lower equilibrium position. To transition it to the upper equilibrium position, it must first be swung up and then stabilized upon reaching the top [6].

During the swing-up phase, energy must be injected into the system so that the pendulum acquires enough total energy to reach the upright position. The total energy of the pendulum EEE, excluding the contribution from the flywheel's rotation relative to the pendulum, is described by the following expression:

$$E = \frac{1}{2}(m_1L_1^2 + m_2L_2^2 + I_1 + I_2)\dot{\theta}^2 + (m_1L_1 + m_2L_2)g\cos\theta.$$

The desired energy E^* of the pendulum when at rest in the upper equilibrium position is given by:

$$E^* = (m_1L_1 + m_2L_2)g.$$

The control law that drives the pendulum to swing up until its energy reaches E^* can be written as:

$$u = k(E^* - E)\ \text{sign}\left(\dot{\theta}\cos\theta\right), \tag{21}$$

where $k > 0$ is a feedback gain. The term $sign\left(\dot{\theta}\cos\theta\right)$ ensures that energy is pumped into the system in the direction of the equilibrium position. This method is known as *Energy Shaping Control*.

The control law (21) increases the total energy of the system until it reaches the desired value, while the sign of the angular velocity and $\cos\theta$ determine the direction of rotation.

Tracking of the desired energy E^* according to the control law (21) is terminated once the system enters a certain region of attraction. At this point, the stabilizing control law (17) is activated, which guides the pendulum to the upper equilibrium position and maintains its stability there.

It is also worth noting that this approach can be used to transition the pendulum to the lower equilibrium position with braking. In this case, the desired energy value E^* should be set with a negative sign:

$$E^* = -(m_1L_1 + m_2L_2)g.$$

3. Practical Section

When working with real-world control systems, the typical workflow begins with a mathematical description of the controlled object, followed by controller design, synthesis, and analysis within a simulation environment. Once a control law and feedback gain values have been determined that meet the desired performance specifications, they are implemented on the hardware platform.

In many cases, the controlled systems include devices that can malfunction or harm a person if incorrect gain parameters are used. Such systems include aerial vehicles, automobiles, and legged robots. For this reason, simulation is always performed prior to deployment on the physical system. The practical part of our work will follow this same principle.

Before proceeding with the following tasks, make sure to install and configure the necessary software as described in the file *Tutorials/Ru_Installation_Guide.docx*.

3.1. Mathematical Modeling

- Open the file *Python/model_student.ipynb*. It contains an example model of a DC motor. The dynamics of the motor are described by the following system of equations:

$$\begin{cases} \ddot{\theta} = -\frac{b}{J}\dot{\theta} + \frac{K}{J}i \\ \frac{di}{dt} = -\frac{K}{L}\dot{\theta} - \frac{R}{L}i - \frac{1}{L}V, \end{cases} \tag{22}$$

  where θ is the angular position of the motor's output shaft, i is the current through the stator windings, V is the applied voltage, b is the friction coefficient, J is the rotor inertia, K is the back-EMF constant, L is the inductance of the windings, and R is the winding resistance.

  In matrix form, this system can be written as:

$$\frac{d}{dt}\begin{bmatrix}\theta \\ \dot{\theta} \\ i\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K}{J} \\ 0 & -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta} \\ i\end{bmatrix} + \begin{bmatrix}0 \\ 0 \\ \frac{1}{L}\end{bmatrix}V. \tag{23}$$

- The block **Pendulum Physical Parameters** contains all the necessary physical parameters of the control object under investigation. These parameters are typically obtained from the datasheet of the component devices or determined experimentally. For example, the resistance of the motor windings can either be taken from the motor's datasheet or measured directly using an ohmmeter. In this particular case, the block provides the parameters of a DC motor. You should now enter all the required parameters of the pendulum into this block, using Table 3 as a reference. Assign variable names that match those listed in the "Notation" column of the table.

Table 3. Pendulum Physical Parameters.

Parameter	Desig-nation	Units	Value	Description
L_1	`L1`	m	0.06	Distance from point O to the center of mass of the rod
L_2	`L2`	m	0.088	Distance from point O to the reaction wheel
R	`R`	m	0.0655	Reaction wheel radius
m_1	`m1`	kg	0.1975	Rod mass
m_2	`m2`	kg	0.2545	Reaction wheel mass
I_1	`I1`	kg/m^2	`m1*L1**2/12`	Rod inertia
I_2	`I2`	kg/m^2	`m2*R**2/2`	Reaction wheel inertia
g	`g`	m/s^2	9.81	Gravity acceleration
μ_p	`b_p`	-	0.007	Viscous friction coefficient at joint O
μ_m	`b_m`	-	0.0	Viscous friction coefficient between the stator and rotor of the motor
L_m	`Lm`	H	0.01	Inductance of the motor winding
R_m	`Rm`	Ohm	22	Resistance of the motor winding
K_e	`Ke`	rad/s/V	0.29	Speed constant
K_t	`Kt`	M m/A	0.27	Torque constant
V_{max}	`Vmax`	V	12	Maximum motor voltage

- The block **Nonlinear model** describes the system of equations (22). Modify this block so that it represents the nonlinear model of the inverted pendulum as given by equation (10). Use the following code template:

```python
# constants
a = ...
b = ...

# function that describes nonlinear dynamics
def nonlinear_dynamics(theta, d_theta, d_phi, voltage):
    # Input:
    #   theta - pendulum angle [rad]
    #   d_theta - pendulum angular velocity [rad/s]
    #   d_phi - wheel angular velocity [rad/s]
    #   voltage - motor input voltage [volt]

    _dd_theta = ...
    _dd_phi = ...

    _theta = theta + d_theta * dt
    _d_theta = d_theta + _dd_theta * dt
    _d_phi = d_phi + _dd_phi * dt


    return np.array([_theta, _d_theta, _d_phi])
```

To verify the correctness of your code, use the block **Nonlinear Model Simulation**, which performs the simulation and plots the transient response. If the model has been implemented correctly, the resulting plot should resemble that shown in Fig. 11.

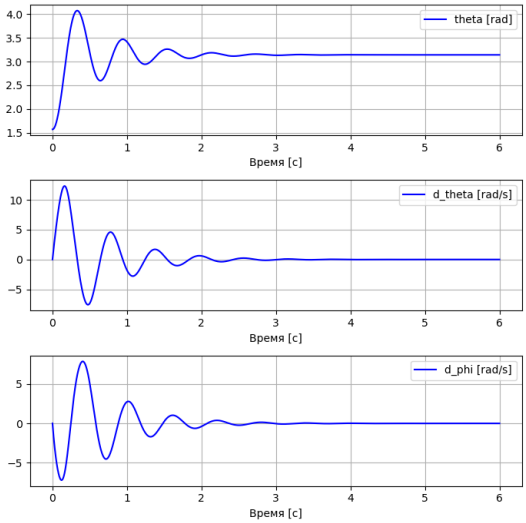

Figure 11. Nonlinear model simulation results.

Provide an explanation for the observed behavior of the system. Try modifying the parameters `x0` and `u` to gain a deeper understanding of the system's dynamics.

- The State **Space Model** block represents the linearized model of the motor in state-space form, as described by equation (23). Modify this block so that it implements the state-space model of the pendulum given in equation (12). Use the template provided below:

```python
# constants
a = ...
b = ...

# elements of matrix A and vector B
a21 = ...
a22 = ...
a23 = ...
a31 = ...
a32 = ...
a33 = ...

b2 = ...
b3 = ...

# State transition matrix A and control vector B
A = np.array([  [0,   1,   0],
                [a21, a22, a23],
                [a31, a32, a33]])

B = np.array([  [0],
                [b2],
                [b3]])

# Output matrices and vectors
C = np.array([  [1, 0, 0],     # pend angle
                [0, 1, 0],     # pend angular vel
                [0, 0, 1]])    # wheel speed

D = np.array([[0],[0],[0]])
```

To verify the correctness of your implementation, use the **State Space Model Simulation** block, which performs the simulation and plots the transient response. If the model has been constructed correctly, the resulting plot should match the one shown in Fig. 12.

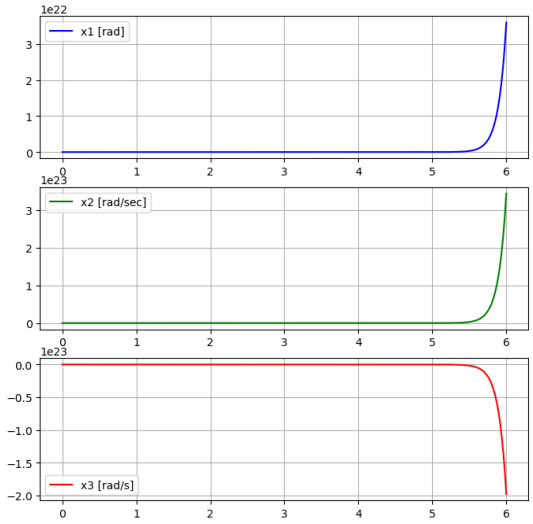

Рисунок 12. Результаты моделирования линейной модели.

Provide an explanation for why the state of the linear model diverges to infinity, in contrast to the nonlinear model.

- The block **Convert to Discrete Time Domain** transforms the continuous-time model into a discrete-time one using functions from the *Control Systems Library*. Based on equations (14) and (15), implement your own version of this conversion manually.

- In the block **Controllability Analysis**, perform an analysis of the system's controllability. Proceed to the following steps only if the system is controllable. If the system turns out to be uncontrollable, it is likely that a mistake was made in the previous steps.
- The block **Stabilization in Upright Position via LQR Control** solves LQR problem. The feedback gain matrix **K** is computed using the `dlqr()` function from the *Control Systems Library*.

  Alternatively, the *scipy* library provides the function `solve_discrete_are(Ad, Bd, Q, R)`, which solves the discrete algebraic Riccati equation. It takes the matrices $\mathbf{A}_d, \mathbf{B}_d, \mathbf{Q},$ and **R** as follows and returns the solution matrix **P**. This function has already been imported in Block 1 of the project. Implement your own computation of the matrix **K** using this function along with equation (18).
- In the block **Stability Analysis**, analyze the stability of the closed-loop system using the discrete-time system stability criterion described on page 13. To compute the eigenvalues of a matrix, use the following NumPy function:

  ```
  eigvals = np.linalg.eigvals(M).
  ```

  Here, instead of `M`, pass the matrix $(\mathbf{A}_d - \mathbf{B}_d\mathbf{K})$ as input. The function will return all eigenvalues of the closed-loop system.
- The blocks **Closed-Loop System Simulation** simulate the behavior of the closed-loop system using both linear and nonlinear models. Compare the results obtained from these simulations. Experiment with different weight matrices **Q** and **R**, and observe how they affect the transient response. Choose the set of weights that you consider the best.
- The block **Swing-Up Control** implements swing-up control of the pendulum followed by a transition to the stabilization mode. The feedback coefficient k and the switching threshold ϵ are initially assigned arbitrary values. You need to tune these parameters so that the pendulum can swing up and then switch into the stabilization mode. Choose the value of k based on the *Energy* plot generated in the block. Aim to minimize the error between E and E^*. Select ϵ such that it prevents premature switching to the stabilization mode before the pendulum enters the region of attraction.

  In our case, the maximum possible error between E and E^* is approximately 0.7 J. Therefore, if you set $\epsilon = 0.4$, for instance, the controller may switch to stabilization mode before sufficient energy has been accumulated.

3.2. Hardware Testing

As a result of completing the steps described above, you should have obtained the following parameters: the feedback gain matrix **K** for stabilization, the swing-up control gain k, and the mode-switching threshold ϵ. These values are required to control the physical pendulum platform.

To apply them, proceed with the following steps:

- Open the file *Firmware/wheel_pendulum_practicum/wheel_pendulum_practicum.ino* using the Arduino IDE. This file contains the implementation for sensor communication, motor control, and the control modes for swing-up, braking, and stabilization.

  The declaration of the feedback coefficients **K** and k, as well as the switching threshold ϵ, is located in lines 17–21 of the code.

The function `control()` is executed every 10 ms. This function implements the logic for stabilization, swing-up, and braking. Study this function carefully to understand under which conditions each control mode is activated.

- Replace the placeholder constants in lines 17–21 with the values you obtained during your work.
- Connect the programmer to the pendulum's control board, and then to the computer via a USB port.
- Compile and upload the firmware by clicking the **Upload** button in the Arduino IDE.

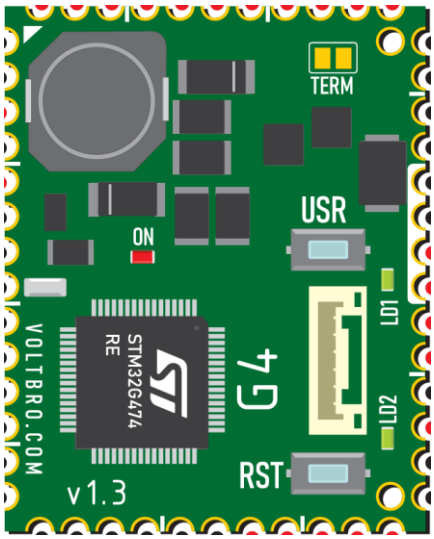

Figure 13. MCU Module VB32G4

- After uploading the firmware, press the **RST** button on the control board (see Fig. 13). The motor will begin to rotate. During this time, the system initializes all necessary motor control parameters. Wait until the motor comes to a complete stop.
- Press the **USR** button. The pendulum control should now be activated and the pendulum should start swinging up.

If the motor does not start within one second after pressing the **USR** button, perform the following actions:

- Gently swing the pendulum by hand.
- Disconnect the programmer, then power off and power on the control board again.

If the pendulum control is activated, the system successfully performs the swing-up maneuver, and the pendulum transitions into the stabilization mode, this indicates that the feedback gains have been correctly configured.

In practice, it is common for gains that perform well in simulation to produce suboptimal results on the real system. This discrepancy arises due to differences between the mathematical model and the physical system. Most often, such differences result from inaccuracies in the estimated physical parameters of the system or from unmodeled effects in the equations of motion (e.g., friction, elasticity, motor speed limitations, etc.).

In such cases, it is recommended to first try adjusting the gain values. If this does not yield satisfactory performance, the model itself may need to be refined. Therefore, if you encounter

difficulties with stabilization, try tuning the gains by modifying the weighting matrices **Q** and **R**. Likewise, if you experience issues with swing-up behavior or control mode switching, try adjusting the coefficients k and ϵ. If these measures do not resolve the issue, verify once again that your mathematical model has been correctly implemented in the *model_student.ipynb* file.

Additionally, observe how the pendulum behaves in the stabilization mode when using different values for the weighting matrices **Q** and **R**. This will help you better understand the sensitivity of the system to control parameters.

Review Questions

1. Why is the inverted pendulum with a reactive wheel considered an underactuated system?
2. How is the feedback gain matrix computed in LQR problem?
3. What is the physical meaning of the matrices **Q** and **R** in the linear-quadratic cost functional?
4. What criterion is used to verify the controllability of a system?
5. What criterion is used to assess the stability of a discrete-time system?
6. Why is the nonlinear model of the pendulum linearized?
7. What role does the reactive wheel play in controlling the system?
8. What happens if the entries of the matrix **R** are set too low in LQR?
9. What physical principles underlie the energy shaping control method?

References

1. Формальский А.М. Управление движением неустойчивых объектов. – Физматлит, 2012
2. Budanov, V., Danilov, V., Kapytov, D., Klimov K. MORS: BLDC-Based Small Quadruped Robot. Journal of Computer System Sciense Int. 63, 2024. https://doi.org/10.1134/S1064230724700710
3. Голубев Ю. Ф. Основы теоретической механики: Учебник. 3-е издание, переработанное и дополненное. – Издательство Московского университета, 2019.
4. Дементьев Ю. Н., Чернышев А.Ю., Чернышев И.А. Электрический привод. – Томский политехнический университет, 2010.
5. Ким Д.П. Теория автоматического управления. Том 2. Многомерные, нелинейные, оптимальные и адаптивные системы. – Физматлит, 2007.
6. Александров В. В., Болотин Ю. В. Спецпрактикум по теоретической и прикладной механике. – Издательство Московского университета, 2019.