Swing-up and Balance Controller for an Inverted Pendulum on a Cart

(Energy Shaping + LQR hybrid with automatic fallback)

December 31, 2025

Abstract

This document summarizes the dynamical model, controller design, and the hybrid control architecture used in the provided implementation. It contains the mathematical derivation of the energy-shaping swing-up law, the linearization used for LQR balance design, the finite-state machine (FSM) for mode switching, implementation notes mapping variables to code, pseudocode for the algorithms, and practical tuning and safety guidelines.

Contents

1 System description and notation

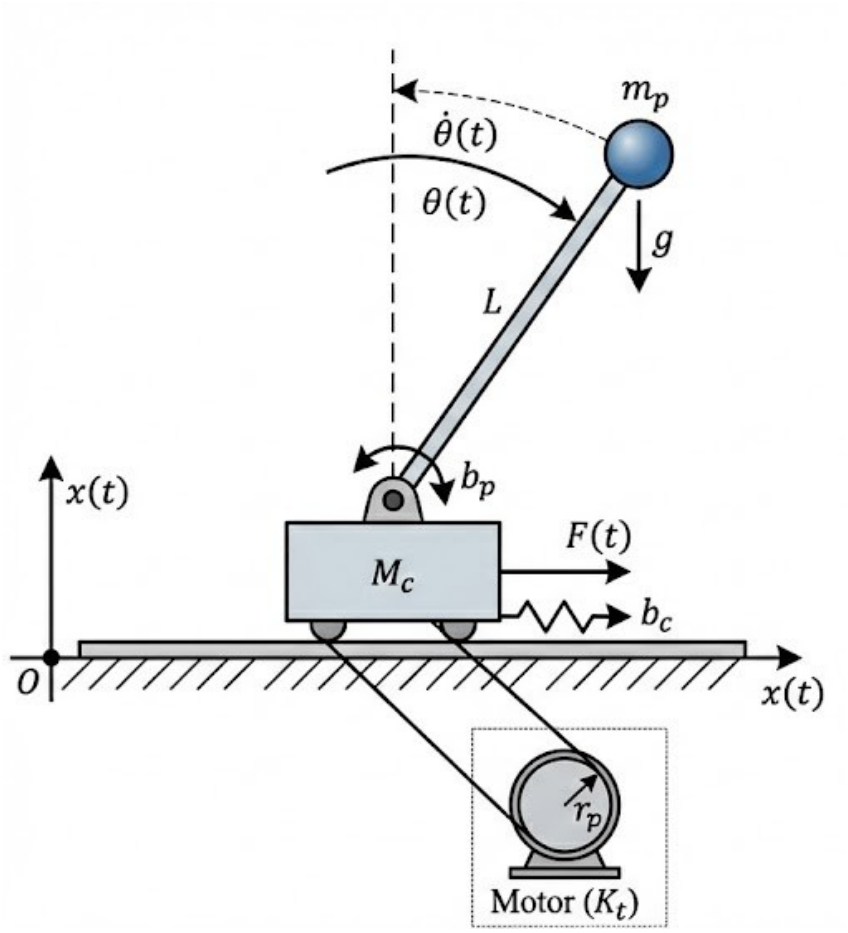

Figure 1: Schematic of the pendulum-on-cart (pulley) system and variable mapping.

The following variables are used:

- $x(t)$: cart horizontal position (m).
- $\dot{x}(t)$: cart velocity (m/s).
- $\theta(t)$: pendulum angle in radians, with $\theta = 0$ when upright. (Note: in code angle is often in degrees; mapping is described below.)
- $\dot{\theta}(t)$: pendulum angular velocity (rad/s).
- L: distance from pivot to pendulum centre-of-mass (m).
- m (or m_p): pendulum mass (kg).
- g: gravitational acceleration (m/s^2).
- $F(t)$: horizontal force applied to the cart (N). The actuator provides torque τ; through the pulley radius r_p this maps to $F = \tau / r_p$.
- Motor torque constant: $\tau = K_t I$. The code uses $I = (F r_p)/K_t$.

2 Dynamical model (simplified)

The standard underactuated pendulum-on-cart model is used. Taking the pendulum as a point mass m at distance L from pivot (small-angle linearization later), the nonlinear equations (including cart mass M_c for generality) are:

$$(M_c + m)\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta = F(t) - b_c\dot{x}, \tag{1}$$

$$mL^2\ddot{\theta} + mL\ddot{x}\cos\theta + mgL\sin\theta = -b_p\dot{\theta}, \tag{2}$$

where b_c and b_p represent viscous friction terms (cart friction and pendulum damping). For the energy-shaping design the pendulum's energy is the primary focus.

2.1 Pendulum energy

Define potential and kinetic energy:

$$PE(\theta) = mgL\cos\theta, \tag{3}$$

$$KE(\dot{\theta}) = \tfrac{1}{2}mL^2\dot{\theta}^2. \tag{4}$$

Total pendulum energy:

$$E(\theta, \dot{\theta}) = mgL\cos\theta + \tfrac{1}{2}mL^2\dot{\theta}^2.$$

Upright target energy:

$$E_{\text{target}} = mgL.$$

Energy error:

$$E_{\text{err}} = E_{\text{target}} - E.$$

3 Energy shaping swing-up controller

3.1 Control law

A continuous energy-shaping input is used to modify pendulum energy toward E_{target}. The energy injection term is:

$$u_{\text{energy}} = K_{\text{energy}}\ E_{\text{err}}\ \dot{\theta}\ \cos\theta.$$

Damping and cart terms are combined with this to form the actuator command (in units of force before conversion to current):

$$u_{\text{pump}} = u_{\text{energy}} - K_{\text{damp}}\dot{\theta} - K_{\text{center}}x - K_{x\_dot}\dot{x}.$$

A virtual wall force $F_{\text{wall}}(x, \dot{x})$ (spring-damper) is computed when the cart approaches limits. The final motor current command is:

$$I_{\text{cmd}} = \text{clip}\Big(u_{\text{pump}} + \frac{F_{\text{wall}}r_p}{K_t},\ -I_{\max},\ I_{\max}\Big),$$

where r_p is the pulley radius and K_t is the motor torque constant.

3.2 Rationale

- The factor $\dot{\theta}$ aligns the energy injection with pendulum motion sign — energy is provided in-phase with motion.
- The $\cos\theta$ factor enforces correct phase dependence around top and bottom.
- Multiplication by E_{err} scales injection so it diminishes near the upright energy.
- The damping term $-K_{\text{damp}}\dot{\theta}$ limits overshoot and reduces kinetic energy near capture.

4 Virtual wall (safety) model

To protect against hitting physical limits, a virtual wall at $x = \pm X_{\text{wall}}$ is used with spring-damper action:

$$F_{\text{wall}}(x, \dot{x}) = \begin{cases} -K_{\text{wall}}(x - X_{\text{wall}}) - D_{\text{wall}}\dot{x}, & x > X_{\text{wall}}, \\ -K_{\text{wall}}(x + X_{\text{wall}}) - D_{\text{wall}}\dot{x}, & x < -X_{\text{wall}}, \\ 0, & \text{otherwise.} \end{cases}$$

Converted to current:

$$I_{\text{wall}} = \frac{F_{\text{wall}} r_p}{K_t}.$$

Outward pumping is suppressed while the wall is active.

5 Linearization and LQR design

5.1 Linearization about the upright

Linearizing the full equations about $\theta = 0$ results in a linear time-invariant model in the state vector $z = [x,\ \dot{x},\ \theta,\ \dot{\theta}]^\top$:

$$\dot{z} = Az + Bu,$$

with A, B dependent on the physical parameters M_c, m, L, b_c, b_p, g. For controller design the conceptual LTI model is used; numerical values require M_c and friction coefficients.

5.2 LQR control law

For the linearized system $\dot{z} = Az + Bu$, select weighting matrices $Q \succeq 0$ and $R \succ 0$ and solve the continuous algebraic Riccati equation for P. The optimal full-state feedback gain is:

$$K_{\text{LQR}} = R^{-1}B^\top P,$$

and the control law (force) is:

$$F(t) = -K_{\text{LQR}} z(t).$$

Mapping the computed force to motor current:

$$I(t) = \frac{F(t)\, r_p}{K_t}.$$

In the implementation the hard-coded LQR gain vector is:

$$K = \begin{bmatrix} K_1 & K_2 & K_3 & K_4 \end{bmatrix} = \begin{bmatrix} -10.0 & -11.7749 & 50.7317 & 6.3 \end{bmatrix},$$

so the discrete implementation uses:

$$I(t) = \mathrm{clip}\Big(\frac{-(Kz(t))\, r_p}{K_t},\ -I_{\max},\ I_{\max}\Big).$$

Interpretation of the gain vector:

- K_3 acts as angle stiffness around the upright,
- K_4 damps angular rate and is critical for capturing high-speed arrival at the top,
- K_1, K_2 regulate cart position and velocity.

6 Hybrid control logic (FSM)

A finite-state machine with two states is used:

$$\text{SWING\_UP} \xrightarrow{\text{if } |\theta|<\theta_{\text{enter}},\ |\dot{\theta}|<\omega_{\text{enter}}} \text{LQR}$$

$$\text{LQR} \xrightarrow{\text{if } |\theta|>\theta_{\text{fall}} \text{ for } N \text{ frames}} \text{SWING\_UP}$$

Values used in the implementation:

- Enter LQR: $\theta_{\text{enter}} = 15^\circ$, $\omega_{\text{enter}} = 1.0$ (rad/s).
- Fall back to swing-up: $\theta_{\text{fall}} = 35$–$45^\circ$, debounced for $N = 5$ frames at 200 Hz.

7 Pseudocode

Algorithm 1 MAIN_LOOP (FSM)

```
Initialize hardware: drv, enc
while true do
    start_rot ← RUN_SWINGUP()
    if start_rot == None then
        break                                   ▷ sensor failure or exit
    end if
    RUN_LQR(start_rot)
end while
drv.setDeviceToIdle()
```

Algorithm 2 RUN_SWINGUP

Read `start_rot` from motor (timeout on failure)
Wait small delay (e.g. 1.5 s)
Initialize filters and previous states
loop
    Read sensors: $(s_1, \texttt{raw\_angle}) \leftarrow \texttt{enc.getAbsoluteAngle()}$
        $(s_2, \texttt{raw\_rot}) \leftarrow \texttt{drv.getMotorPosition()}$
    if $s_1 \neq 0$ **or** $s_2 \neq 0$ **then**
        `drv.setTorqueControl(0.0,0)` ▷ fail-safe zero current
        continue
    end if
    $\theta_{\text{deg}} \leftarrow \text{MAP\_ANGLE}(\texttt{raw\_angle})$
    $\theta \leftarrow \text{rad}(\theta_{\text{deg}})$
    $x \leftarrow (\texttt{raw\_rot} - \texttt{start\_rot}) \cdot 2\pi r_p$
    if prev is None **then**
        store prev values and **continue**
    end if
    Compute $\dot{\theta}$ (with unwrap) and filter; compute $\dot{x}$
    $PE \leftarrow mgL\cos\theta, \quad KE \leftarrow \frac{1}{2}mL^2\dot{\theta}^2$
    $E \leftarrow PE + KE, \quad E_{\text{err}} \leftarrow E_{\text{target}} - E$
    $u_{\text{energy}} \leftarrow K_{\text{ENERGY}}\, E_{\text{err}}\, \dot{\theta}\cos\theta$
    $u_{\text{damp}} \leftarrow -K_{\text{DAMP}}\dot{\theta}$
    $u_{\text{center}} \leftarrow -K_{\text{CENTER}}x - K_{XDOT}\dot{x}$
    $u_{\text{pump}} \leftarrow u_{\text{energy}} + u_{\text{damp}} + u_{\text{center}}$
    Compute wall force $F_{\text{wall}}(x, \dot{x})$ and suppress outward pump if active
    $I \leftarrow \text{clip}\big(u_{\text{pump}} + F_{\text{wall}}r_p/K_t,\ -I_{\max}, I_{\max}\big)$
    `drv.setTorqueControl`(I, 3000)
    if $|\theta_{\text{deg}}| < \theta_{\text{enter}}$ **and** $|\dot{\theta}| < \omega_{\text{enter}}$ **then**
        `drv.setTorqueControl`(0.0,0); **return** `start_rot`
    end if
end loop

Algorithm 3 RUN_LQR(`start_rot`)

Initialize state $z = [0, 0, 0, 0]$, velocity filters, `fall_count`$\leftarrow 0$
loop
  Read sensors: $(s_1, \texttt{raw\_angle}), (s_2, \texttt{raw\_rot})$
  if $s_1 \neq 0$ **or** $s_2 \neq 0$ **then**
    continue
  end if
  $\theta_{\text{deg}} \leftarrow \texttt{raw\_angle} - \texttt{UPRIGHT\_OFFSET}$ (wrap to $[-180, 180]$)
  $\theta \leftarrow \text{rad}(\theta_{\text{deg}})$
  $x \leftarrow (\texttt{raw\_rot} - \texttt{start\_rot}) \cdot 2\pi r_p$
  Compute raw derivatives and EMA-filtered $\dot{x}, \dot{\theta}$
  $z \leftarrow [x, \dot{x}_{\text{filt}}, \theta, \dot{\theta}_{\text{filt}}]$
  if $|\theta_{\text{deg}}| >$ `FALL_ANGLE` **then**
    `fall_count` += 1
  else
    `fall_count` $\leftarrow 0$
  end if
  if `fall_count` $\geq$ `FALL_COUNT_MAX` **then**
    `drv.setTorqueControl`(0.0,0); **return** ▷ switch to swing-up
  end if
  $F \leftarrow -K \cdot z$ ▷ LQR force
  $I \leftarrow \text{clip}\big((F\, r_p)/K_t,\ -I_{\max}, I_{\max}\big)$
  `drv.setTorqueControl`(I,3000)
end loop

Algorithm 4 WATCHDOG_THREAD

Shared variables: `desired_current`, `last_update_time`
loop(run in separate thread at high rate)
  if time.now() - `last_update_time` > `timeout_s` **then**
    `drv.setTorqueControl`(0.0,0) ▷ fail-safe
  else
    `drv.setTorqueControl`(`desired_current`,3000)
  end if
  sleep(small_interval)
end loop

8 Parameter table (mapping to python variables)

9 Implementation notes and safety

- **Actuator mapping**: The code computes current commands. To convert a desired linear force F to current, use $I = (Fr_p)/K_t$.
- **Watchdog**: Because some motor drivers hold the last command if software stalls, a hardware or software watchdog that sets current to zero if the control loop stalls is strongly recommended.
- **Hysteresis**: Use asymmetric thresholds (enter at small angle, exit at larger fall angle) to avoid chattering.
- **Tuning suggestions**:

Symbol	Meaning	Python variable / unit
M	pendulum mass	`M` (kg)
L	pendulum length to CoM	`L` (m)
g	gravity	`G` (m/s^2)
r_p	pulley radius	`PULLEY_RADIUS` (m)
K_t	motor torque constant	`KT` (Nm/A)
$I_{\max}$	current limit	`MAX_CURRENT` (A)
K_{energy}	energy gain	`K_ENERGY`
K_{damp}	pendulum damping gain	`K_DAMP`
K_{center}	cart centering	`K_CENTER`
K_{x_dot}	cart damping	`K_XDOT`
$K_{\text{wall}}, D_{\text{wall}}$	virtual wall gains	`K_WALL`, `D_WALL`

Table 1: Key parameters and variable mapping.

– If LQR cannot catch because $\dot{\theta}$ is too large: increase angular speed feedback gain $K[3]$.

– If pendulum overshoots the upright violently: increase K_{damp}.

– If cart drifts: increase K_{x_dot} or introduce a small K_{center} during swing-up.