Pendulum Documentation

brian.a.dizio

May 2025

Introduction

The inverted pendulum is a classic control problem in dynamics and control theory that simulates the behavior of a pendulum balanced in an unstable equilibrium position. This document outlines the physics, mathematical model, numerical integration techniques, and key parameters used in our implementation, with special attention to tuning parameters for optimal gameplay.

1 Mathematical Model

1.1 Governing Equation of Motion

The dynamics of our inverted pendulum are described by the following second-order nonlinear differential equation:

$$\ddot{\theta} = k_a \sin(\theta) - k_s\theta - k_b\dot{\theta} + k_j u(t)$$

Where:

- θ is the angular displacement from the vertical (positive is clockwise)
- $\dot{\theta}$ is the angular velocity
- $\ddot{\theta}$ is the angular acceleration
- $u(t)$ is the control input (external force)

System coefficients:

$$k_a = \frac{mLg}{mL^2 + I_z}, \quad k_s = \frac{k_{sp}}{mL^2 + I_z}, k_b = \frac{b}{mL^2 + I_z}, \quad k_j = \frac{K_{joy}}{mL^2 + I_z}$$

1.2 Physical Interpretation

1. **Gravitational term** ($k_a \sin(\theta)$): Causes the pendulum to fall away from equilibrium.
2. **Spring term** ($-k_s\theta$): Restores pendulum toward center.

3. **Damping term** ($-k_b\dot{\theta}$): Viscous damping slows motion.
4. **Control input** ($k_j u(t)$): Torque applied by player input.

2 Numerical Integration

2.1 State-Space Formulation

$$\dot{\theta} = \omega, \quad \dot{\omega} = k_a \sin(\theta) - k_s\theta - k_b\omega + k_j u(t)$$

2.2 Runge-Kutta 4th Order Method (RK4)

$$k_1 = hf(t_n, y_n) k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) k_3 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) k_4 = hf(t_n+h, y_n+k_3) y_{n+1} = y_n + \frac{1}{6}(k_1$$

Alternative methods include:

- Euler method (1st order)
- Improved Euler method (2nd order)

3 Key Parameters and Their Effects

3.1 Sensitivity Analysis

Parameter	Symbol	Effect When Increased
Mass	m	Increases inertia, slower, more stable
Length	L	Increases inertia, harder to stabilize
Spring constant	k_{sp}	More centering force, can cause oscillation
Damping	b	Reduces oscillations, increases control
Force scaling	k_j	More responsive control

3.2 Gameplay Recommendations

- Lower k_j to reduce excessive torque
- Slightly increase b for better damping
- Try nonlinear control mappings

4 Visualization

4.1 Pendulum Rendering

$$x_{bob} = x_{pivot} + L \cdot \sin(\theta), \quad y_{bob} = y_{pivot} - L \cdot \cos(\theta)$$

4.2 Phase Space Interpretation

- Stable points: attractors
- Unstable points: repellers
- Cycles: oscillations
- Irregularity: chaos

5 Implementation Notes

5.1 Boundary Conditions

[language=Python] if abs(position) ¿ fallBoundary: position = fallBoundary if position ¿ 0 else -fallBoundary velocity = 0

5.2 Time Step

Fixed timestep: 1/60 seconds (60 Hz).

6 Further Development

- Adaptive difficulty tuning
- Add perturbation forces
- Impulse response analysis
- Test multiple control schemes