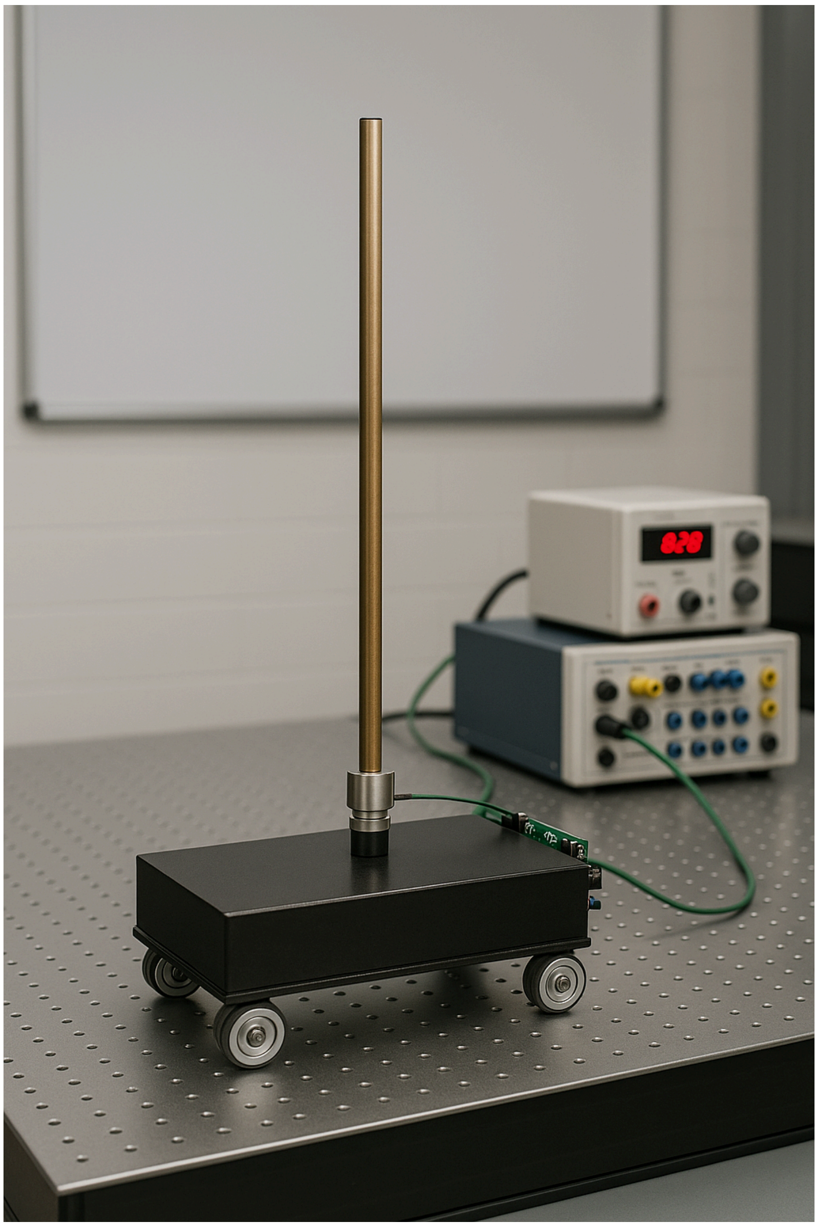
Modeling and Simulation of an Inverted Pendulum using Python



Henrik Schoofs (31501)
Advanced Modelling and Simulation

1. Introduction

This document presents the modeling, linearization, control design, and simulation of an inverted pendulum on a cart. The system is a classical benchmark problem in control theory and robotics due to its nonlinear dynamics and inherent instability. Controlling the inverted pendulum serves as a compelling example of applying advanced mathematical tools to real-time control problems.

The project involves using Python and scientific computing libraries to derive the equations of motion, linearize them, design a stabilizing controller using optimal control theory (LQR), and simulate the dynamic behavior of the closed-loop system.

2. System Description

The inverted pendulum system is a two-degree-of-freedom underactuated mechanical system with one actuator (horizontal force applied to the cart) and two state variables of interest (the cart position and the pendulum angle).

The components of the system are:

- A cart of mass moving on a horizontal track
- A pendulum of mass and length suspended from a pivot on the cart
- A control input , representing the horizontal force applied to the cart

Objectives

The control objective is to stabilize the pendulum in the upright position and optionally to regulate the cart position. In the absence of control, the pendulum will fall due to gravity, and the system is unstable.

The model assumes no friction in the pivot and that the cart moves without resistance. The pendulum is treated as a rigid rod with mass concentrated at its end.

3. Nonlinear Modeling with Lagrangian Mechanics

x : horizontal position of the cart
θ : angle of the pendulum from the vertical

3.1 Kinetic Energy (T)

The total kinetic energy includes contributions from both the translational motion of the cart and the combined translational and rotational motion of the pendulum mass.

- Position of the pendulum mass center: $x_p = x + l\sin\theta, \quad y_p = -l\cos\theta$
- Velocities: $\dot{x}_p = \dot{x} + l\dot{\theta}\cos\theta, \quad \dot{y}_p = l\dot{\theta}\sin\theta$
- Velocity squared: $v_p^2 = (\dot{x} + l\dot{\theta}\cos\theta)^2 + (l\dot{\theta}\sin\theta)^2 = \dot{x}^2 + 2l\dot{x}\dot{\theta}\cos\theta + l^2\dot{\theta}^2$

Thus, the total kinetic energy is:

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{x}^2 + 2l\dot{x}\dot{\theta}\cos\theta + l^2\dot{\theta}^2)$$

$$= \frac{1}{2}(M+m)\dot{x}^2 + ml\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2$$

3.2 Lagrangian

The Lagrangian is given by:

$$L = T - V = \frac{1}{2}(M+m)\dot{x}^2 + ml\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos\theta$$

3.3 Euler-Lagrange Equations

Using the Euler-Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i$$

We derive the two second-order differential equations for x and theta:

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = u$$

$$ml^2\ddot{\theta} + ml\ddot{x}\cos\theta + mgl\sin\theta = 0$$

These nonlinear equations form the basis for linearization and controller design.

4. Linearization

To facilitate control design, the system is linearized about the upright position (theta = 0). This allows the application of linear control theory such as LQR.

4.1 Assumptions

- Small-angle approximation: $\sin\theta \approx \theta, \cos\theta \approx 1$
- Linearized about $x = 0, \dot{x} = 0, \theta = 0, \dot{\theta} = 0$

4.2 State-Space Representation

Define the state vector:

$$x = \begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} \end{bmatrix}^T$$

Then the system becomes:

$$\dot{x} = Ax + Bu$$

Where A and B are 4x4 and 4x1 matrices determined by linearization.

5. Control Design: Linear Quadratic Regulator (LQR)

5.1 Overview

LQR is a method to compute the optimal feedback gain matrix K such that u=-Kx minimizes the cost function:

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt$$

Q: state weighting matrix (penalizes deviations from desired states)
R: control weighting matrix (penalizes large control efforts)

5.2 Riccati Equation

LQR solves the algebraic Riccati equation:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

Then:

$$K = R^{-1}B^T P$$

This ensures stability and optimal performance for a linear time-invariant system.

6. Simulation & Control Loop

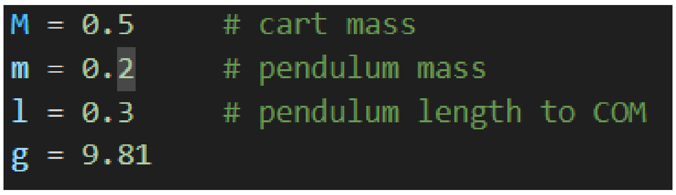
Each time step:

1. Measure the state $\mathbf{x}(t)$
2. Compute control input $u(t) = -K\mathbf{x}(t)$
3. Update system using $\dot{\mathbf{x}} = A\mathbf{x} + Bu$

This is integrated over time using numerical method solve_ivp.

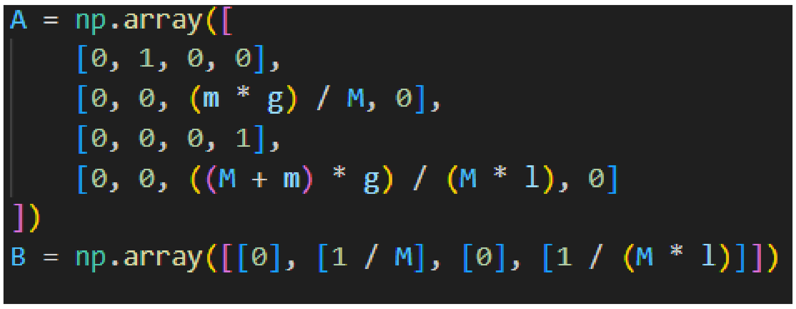
7. Python implementation

7.2 The physical parameters used in modeling the dynamics:



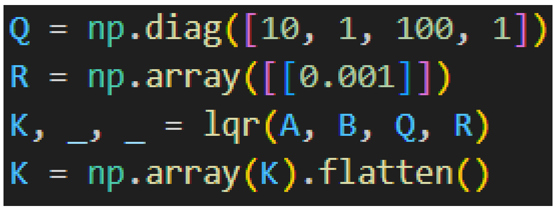
```
M = 0.5      # cart mass
m = 0.2      # pendulum mass
l = 0.3      # pendulum length to COM
g = 9.81
```

7.3 The state space matrices:



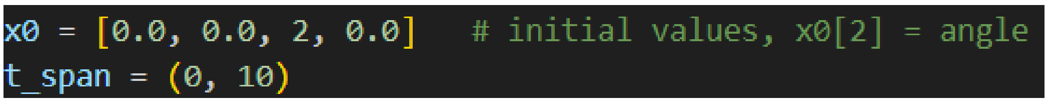
```
A = np.array([
    [0, 1, 0, 0],
    [0, 0, (m * g) / M, 0],
    [0, 0, 0, 1],
    [0, 0, ((M + m) * g) / (M * l), 0]
])
B = np.array([[0], [1 / M], [0], [1 / (M * l)]])
```

7.4 LQR design:
The cost matrices Q and R are defined, then the optimal feedback gain matrix K is computed using the LQR method.



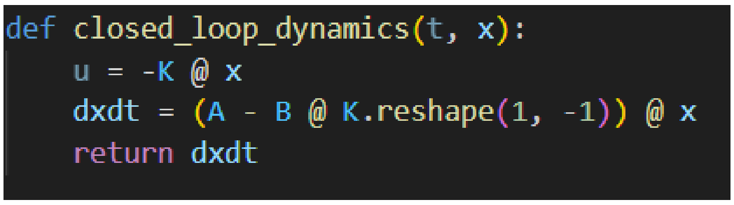
```
Q = np.diag([10, 1, 100, 1])
R = np.array([[0.001]])
K, _, _ = lqr(A, B, Q, R)
K = np.array(K).flatten()
```

7.5 Simulation Setup



```
x0 = [0.0, 0.0, 2, 0.0]   # initial values, x0[2] = angle
t_span = (0, 10)
```

7.6 Define system dynamics



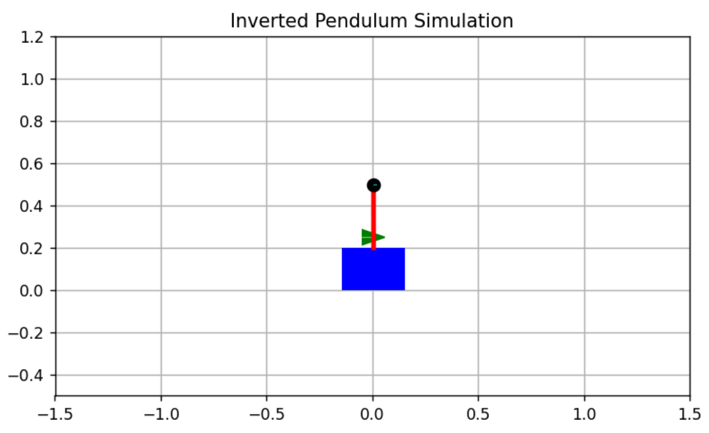
```
def closed_loop_dynamics(t, x):
    u = -K @ x
    dxdt = (A - B @ K.reshape(1, -1)) @ x
    return dxdt
```

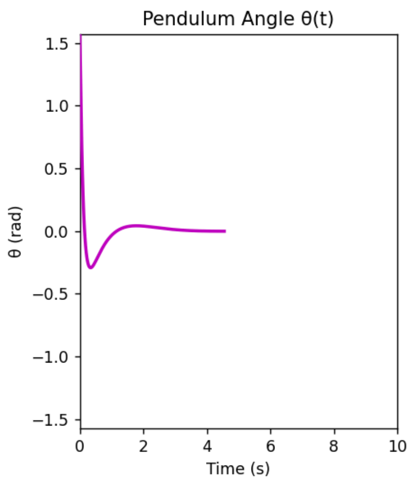
7.7 Numeric integration

```python
sol = solve_ivp(closed_loop_dynamics, t_span, x0, t_eval=t_eval)
```

8. Result:

The pendulum simulation starts with a small angle theta and quickly stabilizes.





9. Full Python code:

https://github.com/hschoofs/AMS_inverted_pendulum/tree/main/code/python

10. Matlab code:

https://github.com/hschoofs/AMS_inverted_pendulum/tree/main/code/matlab

11. Sources:

- https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling
- chatgpt.com