Introduction

The inverted pendulum on a cart is a classic benchmark problem in control engineering and robotics. It represents the challenge of stabilizing an inherently unstable, nonlinear system. The objective is to apply a controlled force to the cart to keep the pendulum balanced in its upright position. This report details the exploration of different control strategies, from classical controller design to modern machine learning techniques, to solve this problem.

Part 1: Controller Research

a. Three Common Controller Types

1. **PID Controller (Proportional–Integral–Derivative):** A classical feedback controller that applies a corrective force based on the present error (P), the accumulation of past errors (I), and a prediction of future errors (D).
2. **LQR Controller (Linear Quadratic Regulator):** An optimal state-feedback controller designed using a linearized state-space model of the system. It works by minimizing a quadratic cost function that balances state deviations and the required control effort.
3. **Reinforcement Learning (RL)-based Controller:** A modern, data-driven approach where a control policy is learned through direct interaction with the system. The controller (or "agent") receives rewards for actions that lead to stable, upright balance and learns to maximize its cumulative reward over time.

b. Features, Pros, and Cons

Controller	Major Features	Pros	Cons
PID	Error-correction based on P, I, and D terms.	Simple to implement and understand. Effective near the upright equilibrium point.	Tuning is often manual and system-specific. Performance degrades with large disturbances and strong nonlinearities.
LQR	State-space model-based optimal control.	Systematic design process via cost-weight tuning. Balances performance and control effort optimally.	Requires an accurate mathematical model. Performance suffers if the system deviates far from the linearization point.
RL-based	Data-driven learning through trial and error.	Can handle complex nonlinearities and adapt to uncertainties without a model.	Requires large amounts of data and computation for training. Risk of instability during the learning phase.

Can outperform classical methods.

c. Recommended Controller

For this specific problem, the **Linear Quadratic Regulator (LQR)** is recommended. It leverages the well-understood linearized dynamics of the pendulum near its stable point, provides a systematic and optimal tuning method compared to the trial-and-error nature of PID tuning, and strikes a good balance between performance, robustness, and implementation complexity.

Part 2: Simulation and Classical Control

a. System Implementation

The inverted pendulum dynamics were implemented in Python using an object-oriented approach. The simulation utilizes the **Runge-Kutta 4th order (RK4)** method for accurately integrating the equations of motion over time.

b. Hand-Tuned PID Controller

A standard PID controller was designed and implemented. Through a manual tuning process, the following gains were selected:

- **Kp (Proportional):** 80
- **Ki (Integral):** 1
- **Kd (Derivative):** 20

c. Simulation Results

The system was simulated for 10 seconds, with the pendulum starting from an initial tilt of 30 degrees ($\pi/6$ radians). The hand-tuned PID controller resulted in a **performance metric (integrated absolute error) of 16.891**. As shown in the comparison plot below, these gains failed to stabilize the pendulum, resulting in sustained oscillations around the upright position.

Part 3: Machine Learning for PID Tuning

a. ML Model for Automated Tuning

To automate and improve the PID tuning process, a **Multi-layer Perceptron (MLP) Regressor** from the scikit-learn library was chosen. The model was designed to predict the performance metric of the system given a set of PID gains (Kp, Ki, Kd) as input.

b. Model Training

A dataset was generated by running 300 simulations with randomly selected PID gains. The MLP model was trained on this dataset to learn the complex relationship between the gains

and the resulting system performance. The model was trained with hidden layer sizes of (32, 32) for 2000 iterations.

c. ML-Tuned Controller Results

The trained model was then used as a performance predictor to efficiently search for optimal gains. It evaluated 200 candidate gain combinations and identified the set predicted to yield the best performance. The best gains found were:

- **Kp (Proportional):** 12.27
- **Ki (Integral):** 6.03
- **Kd (Derivative):** 4.83

The simulation with these ML-suggested gains resulted in a **performance metric of 17.801**. More importantly, this controller successfully stabilized the pendulum, quickly damping the initial disturbance and maintaining the upright position.

Part 4: Comparison and Conclusion

Comparison of Results

The plot below provides a clear visual comparison between the two controllers.

- The **Hand-tuned PID (blue line)** results in continuous, unstable oscillations, failing to stabilize the system.
- The **ML-suggested PID (orange line)** effectively dampens the initial disturbance and brings the pendulum to a stable, upright position (theta = 0) in under 3 seconds.

While the calculated performance metric for the ML-tuned controller is slightly higher, the qualitative result is vastly superior. The metric for the hand-tuned system is skewed because the error oscillates around zero, while the ML-tuned system successfully eliminates the error entirely.

Conclusion

This analysis demonstrates the power of combining classical control theory with modern machine learning techniques. While manual PID tuning can be challenging and yield suboptimal results, an ML-based approach can automate the process and discover highly effective control parameters. ML-based tuning enhances controller performance and adaptability, though it requires an initial investment in data generation and model training. For robust control systems, a hybrid approach is often ideal: using classical methods like PID or LQR for a baseline and then leveraging ML to fine-tune the parameters for superior performance.