# Deep Learning (1470)

Randall Balestriero

Class 24: Model Based RL

# Recap!

# Recap: Model-Free RL

DQN


A2C / A3C

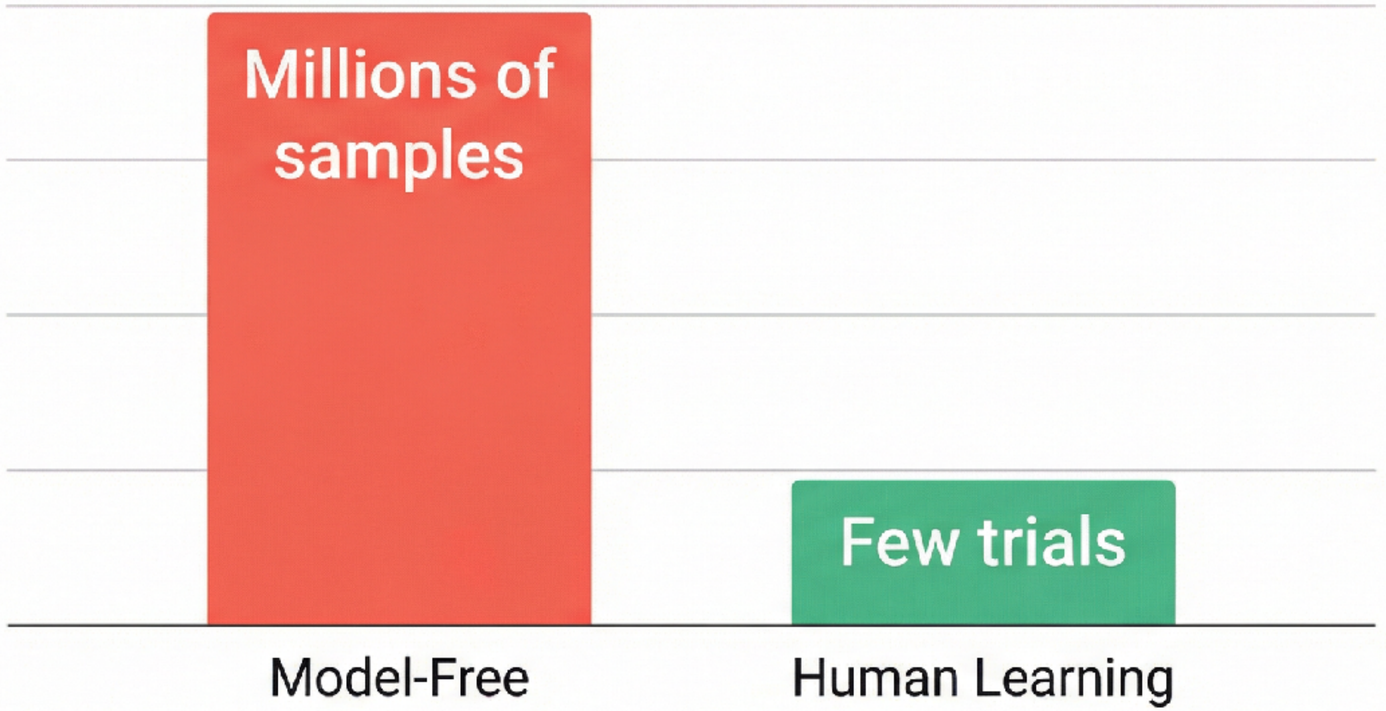
PPO

GRPO

Key idea: Learn policy or value function directly from experience

No explicit model of the environment

# The Problem: Sample Inefficiency




⚠ Real-world interaction is expensive

⚠ Dangerous in robotics and autonomous systems

⚠ Slow iteration in complex environments

**Can we learn more efficiently by understanding how the world works?**

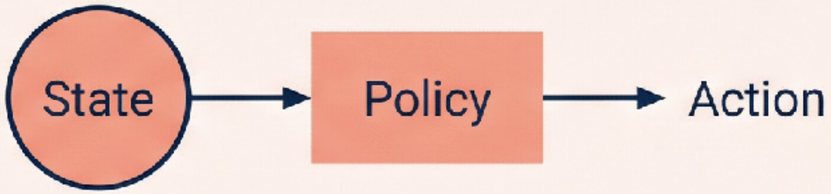
# The Key Insight

**What if the agent learns how the world works?**

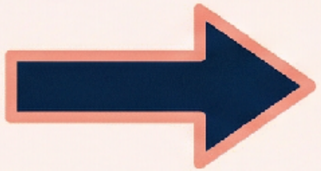

- Practice in imagination
- Plan before acting
- Generalize to new situations

# Model-Free vs Model-Based

## Model-Free




- Learn: policy or value function
- No dynamics model



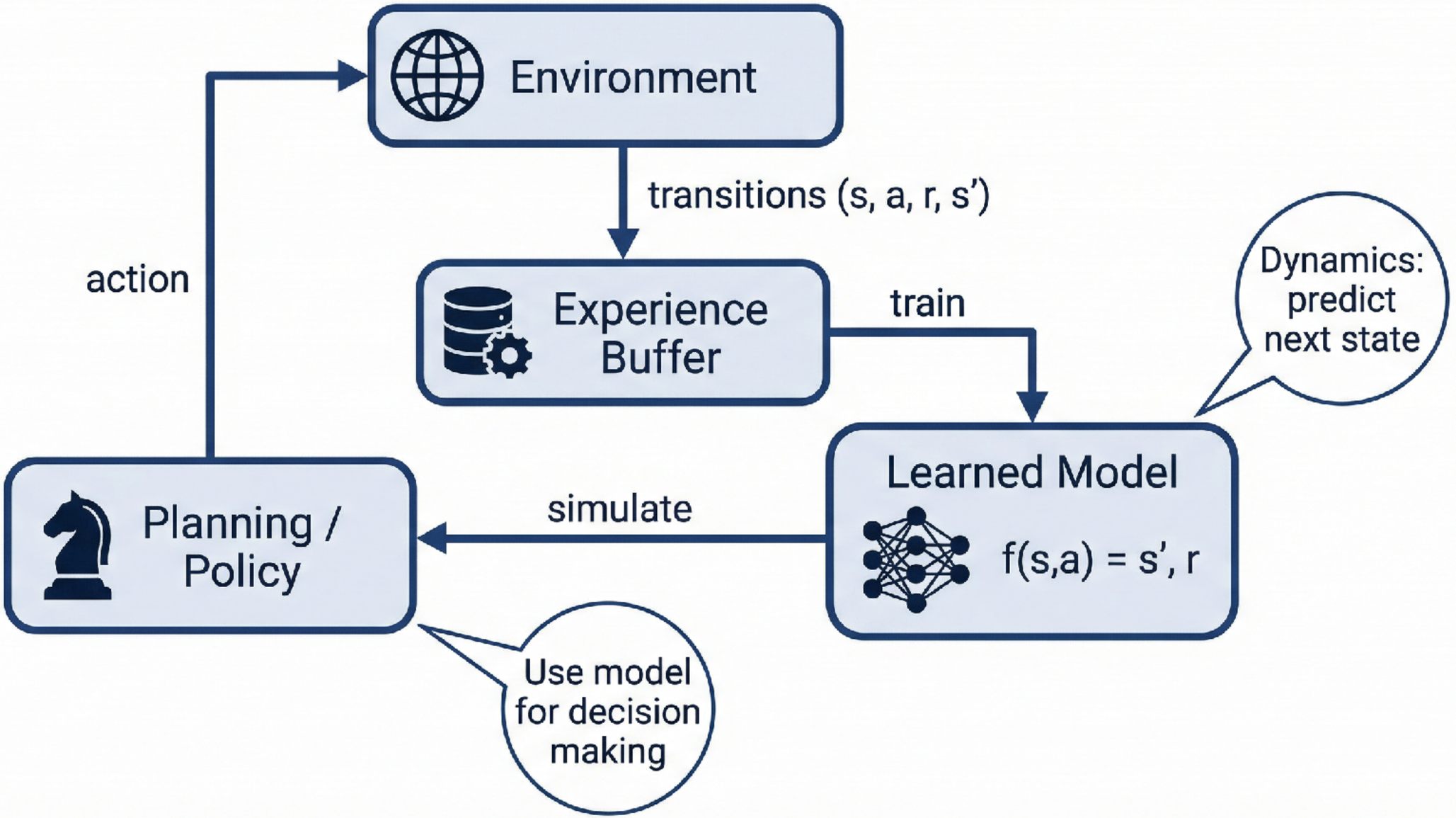
## Model-Based


- Learn: dynamics model
- Plan using the model

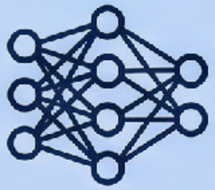
Trade-off: Simplicity vs Sample Efficiency

# If we can predict the next state, can we act?

# The Model-Based RL Framework



# Two Core Components



## 1. The Model

Learn environment dynamics

Transition: $s_{t+1} = f(s_t, a_t)$
Reward: $r_t = g(s_t, a_t)$

- Note: Can be deterministic or probabilistic

## 2. The Planner

Use model for decisions

- Shooting methods
- Model Predictive Control
- Policy optimization
- Tree search

**Model + Planner = Model-Based RL**

# What Can We Do with a Model?

## Model Predictive Control

Optimize actions over short horizon

## Planning and Search

Explore future possibilities

## Data Augmentation

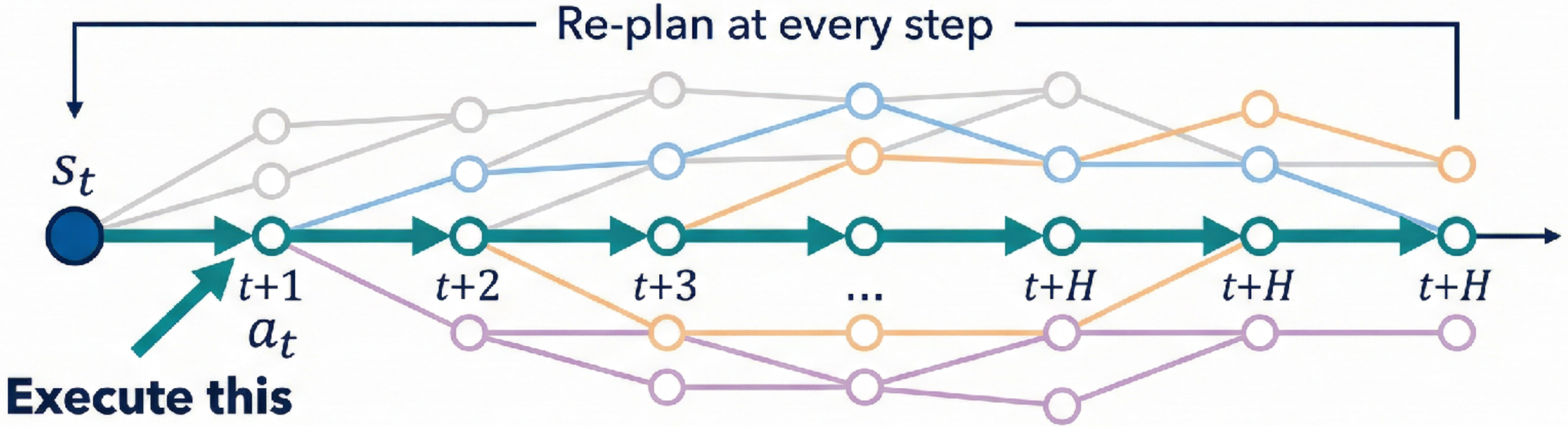
Generate synthetic experience

## Policy Improvement

Train policy on imagined rollouts

# Model Predictive Control (MPC)

The workhorse of model-based control




1. Optimize over horizon $H$

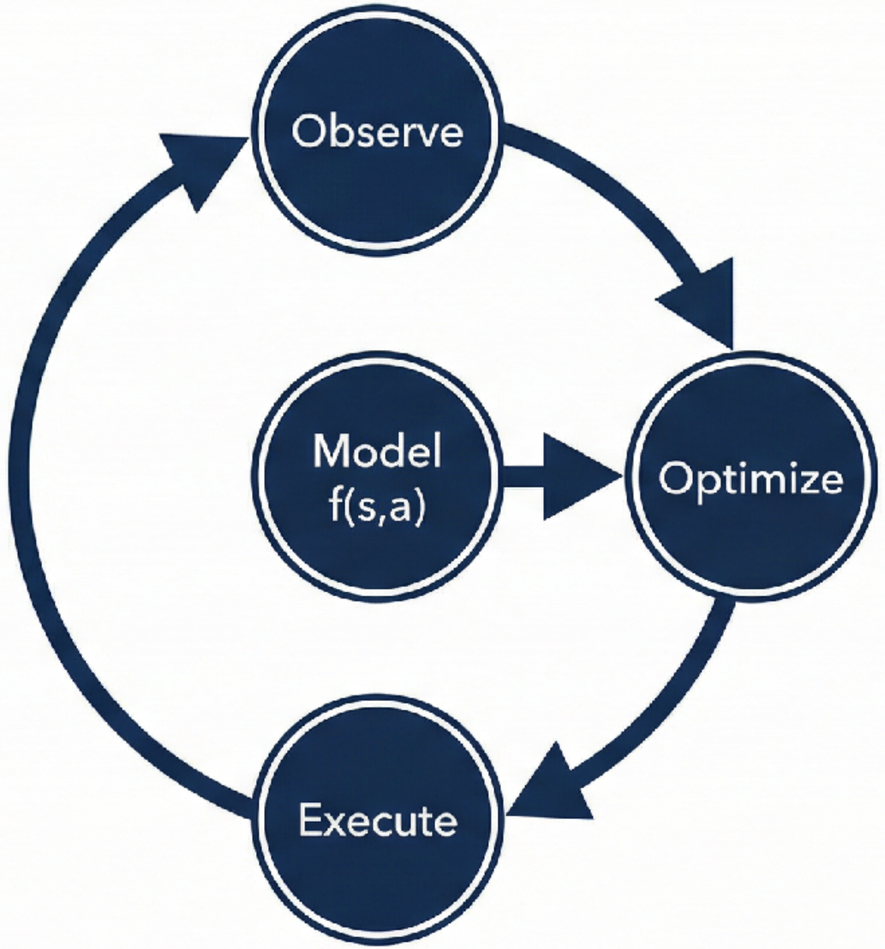
2. Execute first action only

3. Repeat with new state

Receding horizon control

# MPC: The Algorithm

```
At each timestep t:
1. Observe current state s_t
2. Optimize action sequence:
    a* = argmax Sum of r(s_i, a_i)
    subject to: s_i+1 = f(s_i, a_i)
3. Execute first action a*_0
4. Return to step 1
```




**Key insight: Re-planning compensates for model errors**

# Simple Case: Linear Quadratic Regulator (LQR)

When model-based RL has a closed-form solution

## Linear Dynamics

$$\mathbf{s}_{t+1} = \mathbf{A}\,\mathbf{s}_t + \mathbf{B}\,\mathbf{a}_t$$

**A**: state transition matrix
**B**: control matrix

## Quadratic Cost

$$J = \sum (\mathbf{s}^T \mathbf{Q} \mathbf{s} + \mathbf{a}^T \mathbf{R} \mathbf{a})$$

**Q**: state cost matrix
**R**: control cost matrix

This combination yields an optimal solution in closed form!

# LQR: The Closed-Form Solution

Optimal Policy (linear feedback):

$$a_t = -K s_t$$

K = Feedback gain matrix

Gain matrix K:

$$K = (R + B^T P B)^{-1} B^T P A$$

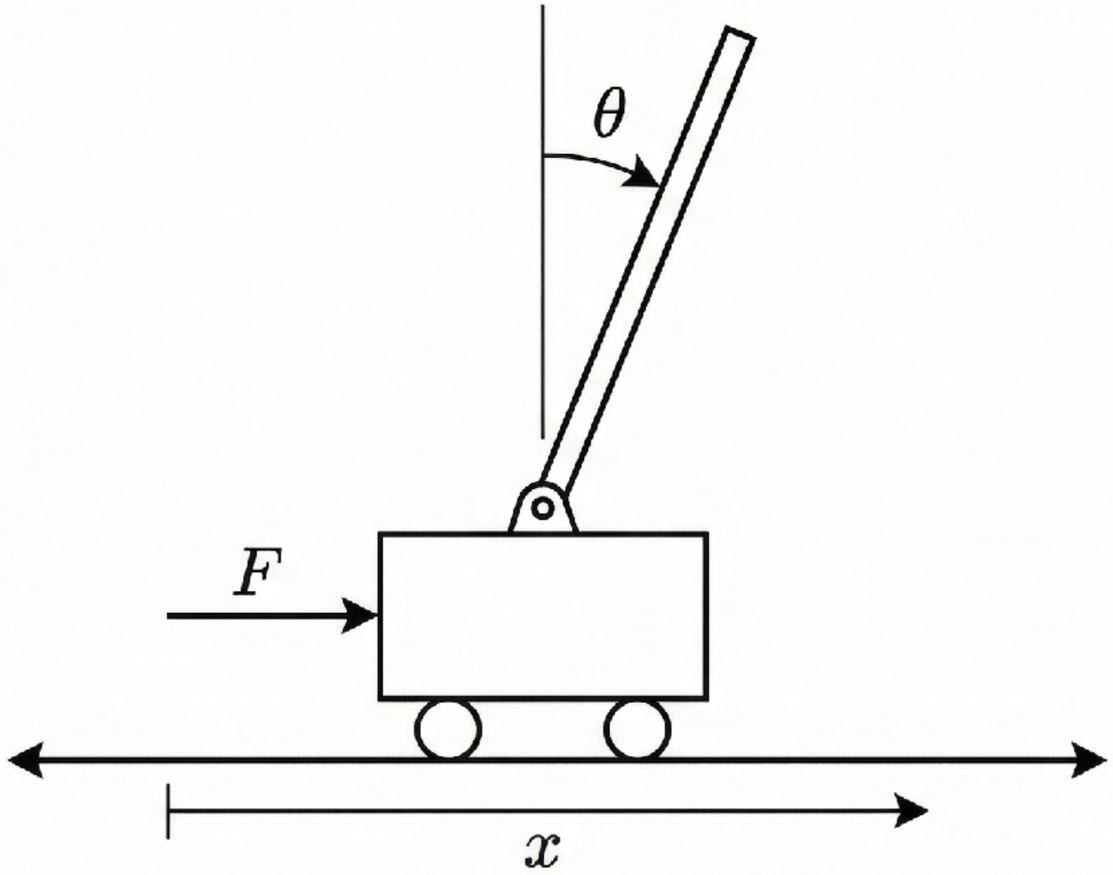
P = Value function matrix

P solves the Riccati equation:

$$P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A$$

Key: Optimal control is a linear function of state

# Example: Cartpole Control




**State vector s**

$\mathbf{s} = [x, \dot{x}, \theta, \dot{\theta}]$

position, velocity, angle, angular velocity

**Control**

$\mathbf{a} = F$ (force on cart)

**Goal**

- Keep pole upright: $\theta$ near 0
- Minimize control effort

Linearize around upright position, then apply LQR

# Cartpole LQR: The Matrices

**A**: State Transition

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -(m^*g)/M & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & (M+m)^*g/(M^*l) & 0 \end{bmatrix}$$

**B**: Control Input

$$\begin{bmatrix} 0 \\ 1/M \\ 0 \\ -1/(M^*l) \end{bmatrix}$$

**Q**: State Cost

$$\begin{bmatrix} q\_x & 0 & 0 & 0 \\ 0 & q\_v & 0 & 0 \\ 0 & 0 & q\_theta & 0 \\ 0 & 0 & 0 & q\_omg \end{bmatrix}$$

**R**: Control Cost

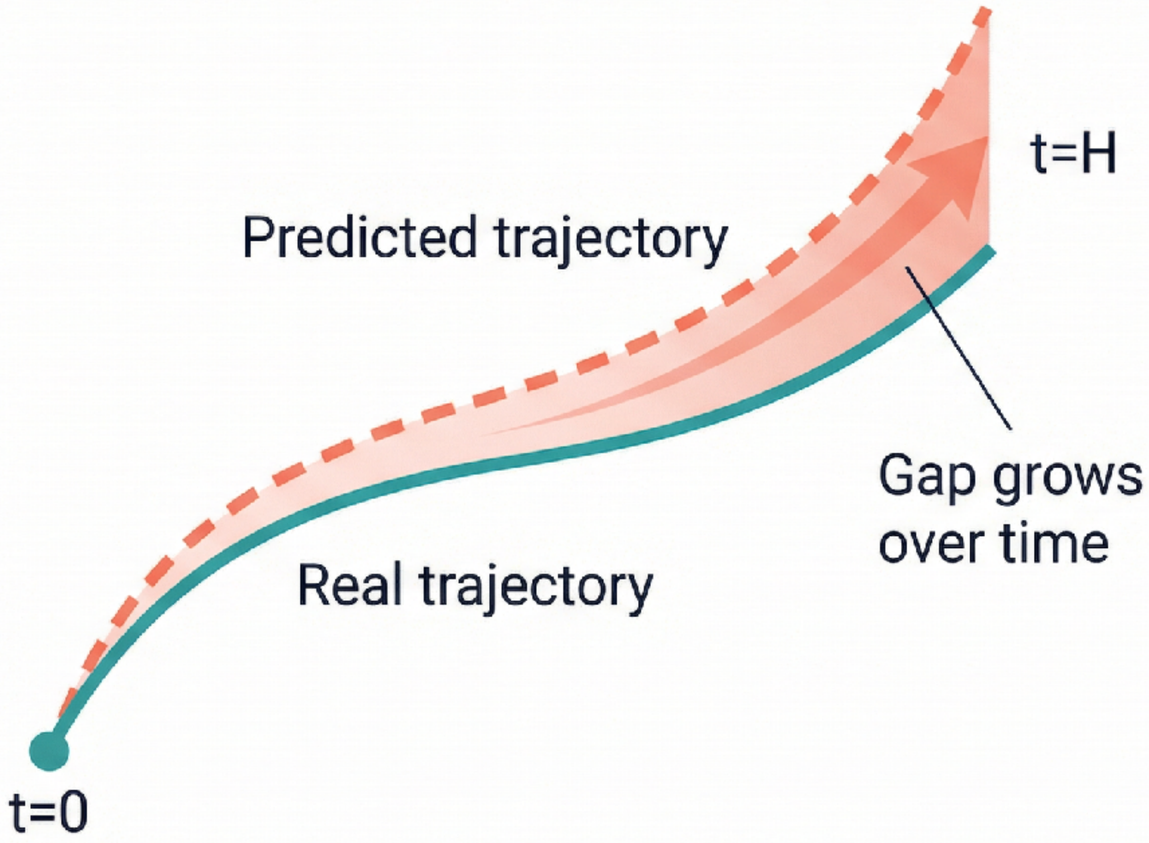
$$R = r\_u$$

Parameters: M=cart mass, m=pole mass, l=length, g=gravity

# Any limitation for long-horizon planning?

```
At each timestep t:
1. Observe current state s_t
2. Optimize action sequence:
    a* = argmax Sum of r(s_i, a_i)
    subject to: s_i+1 = f(s_i, a_i)
3. Execute first action a*_0
4. Return to step 1
```

# Challenge: Model Errors Compound




**Small errors accumulate**
Error per step times H steps

**Model exploitation**
Policy finds unrealistic shortcuts

**Distribution shift**
Model trained on different data than policy uses

**Solutions:** Short horizons, uncertainty estimation, re-planning

# With or Without Reward Prediction?

## Reward-Predicting

- Learn dynamics AND reward
- Can plan full trajectories
- Tied to one task
- Examples: Dreamer, MBPO

## Reward-Free

- Learn dynamics only
- Reward given later
- Transfer to any task
- Examples: Plan2Explore

**Trade-off:** Task-specific optimization vs Multi-task flexibility

See you on Friday!