Markov Decision Processes
A Unifying Perspective on MPC and RL

Jasper Hoffmann

University of Freiburg

Predict, Control, Learn: Combining Model Predictive Control and Reinforcement Learning
Tutorial Session at ECC 2026, Reykjavík, July 2026

universität freiburg

NTNU | Norwegian University of Science and Technology

University of Zurich[UZH]

ROBOTICS & PERCEPTION GROUP

Real-World Control Gets Messy Fast

data
constraints
uncertainty
safety
simulation
disturbances
deployment risk
engineering priors
exploration
policy learning
generalization
limited compute
sample efficiency

One promising direction is MPCRL:
MPC structure + RL adaptation.
This tutorial provides overview, theory, and software to get started.

ECC26 TUTORIAL

The Predict, Control, Learn Crew

University of Freiburg

Katrin Baumgärtner

Jonathan Frey

Leonard Fichtner

Jasper Hoffmann

Joschka Bödecker

Moritz Diehl

NTNU

Dirk Reinhardt

Sebastien Gros

University of Zurich

Rudolf Reiter

Davide Scaramuzza

Important Cornerstones of this Tutorial

Survey

Classification and terminology for MPC+RL combinations.

survey paper

Research

Theoretical and algorithmic foundations behind the tutorial.

Differentiable NMPC
Frey et al. 2025

MPC as policy/value model
Gros et al. 2019

Optimality conditions
Anand et al. 2025

Software

Tools for building learning predictive controllers.

- `acados`: fast NMPC solvers
- `leap-c`: differentiable MPC/RL workflows

`leap-c`

Tutorial Roadmap

	Talk	*Presenter*	*Duration*
➤	**Markov Decision Processes – a unifying perspective on MPC and RL**	Jasper Hoffmann	20min
	Synthesis of Model Predictive Control and Reinforcement Learning	Rudolf Reiter	30min
	Reinforcement Learning over MPC: Why does it work?	Dirk Reinhardt	30min
	Differentiable NMPC with `acados`	Katrin Baumgärtner	20min
	MPCRL with `leap-c`	Jasper Hoffmann	20min

At the end of each talk there will be time for questions!

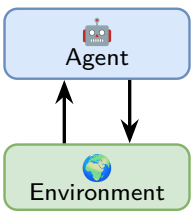
Tutorial webpage for slides and more!

Two Communities, One Problem

Reinforcement Learning

- learns from interaction or data
- represents policies and value functions explicitly
- expensive training, cheap deployment

Markov Decision Process



Model Predictive Control

- plans online with an engineering model
- solves a finite-horizon optimization problem
- cheap setup, more expensive deployment

Different traditions, same core task: choose actions over time to minimize cumulative cost.

Complementary Strengths

MPC contributes structure

- direct handling of state and input constraints
- stability and robustness tools from control theory
- interpretable objectives, models, and predictions
- adaptation by changing model, cost, or constraints

RL contributes learning

- fast inference with learned policies
- flexible function approximation for complex states
- improvement from simulations or measured data
- can handle uncertainty well!?

Tutorial theme: combine MPC structure with RL flexibility.

Outline

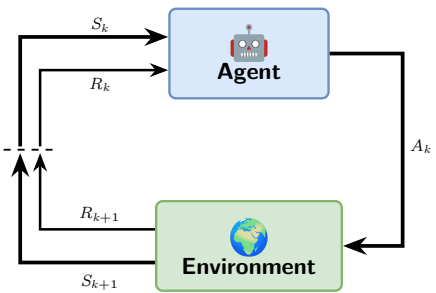
Agent and Environment

Interaction Loop:

- Time steps k: $0, 1, 2, \ldots$
- States: $S_0, S_1, S_2, \ldots$
- Actions: $A_0, A_1, A_2, \ldots$
- Rewards: $R_0, R_1, R_2, \ldots$

Given an initial state $S_0 = s_0$ we get an interaction process:

$$S_0, A_0, R_0, S_1, A_1, R_1, S_2 \ldots \quad .$$



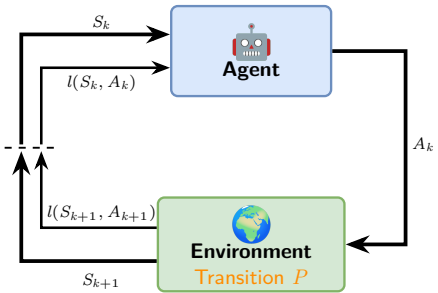
Markov Decision Processes – Dynamics

- The **Markov assumption**: the next state depends on the past only through the current state and action.
- Thus, a transition kernel P can describe the next-state distribution:

$$S_{k+1} \sim P(\cdot \mid S_k, A_k).$$

- Deterministic dynamics are a special case:

$$S_{k+1} = f(S_k, A_k).$$



Markov Decision Processes – Costs

- The reward is a scalar feedback signal from the environment.
- Instead of maximizing a reward signal R_k in this tutorial **we minimize a cost** $l : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

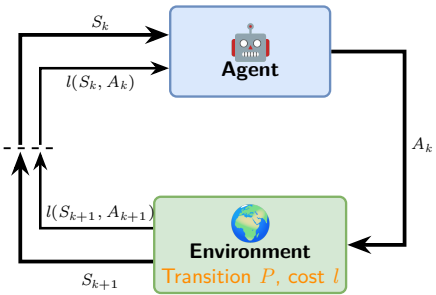
$$R_k = l(S_k, A_k)$$

- Agent should minimize the **discounted cumulative cost**

$$l(S_0, A_0) + \gamma l(S_1, A_1) + \gamma^2 l(S_2, A_2) + \dots,$$

where $\gamma \in [0, 1)$.

- Discount factor makes infinite horizon possible. It emphasizes more short term costs than long term costs.

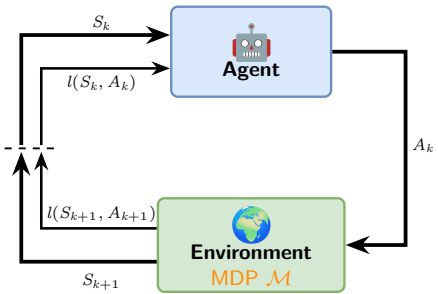


Markov Decision Processes – Summary

Definition

A Markov decision process (MDP) is a 5-tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, l, \gamma)$, where

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $P(\cdot \mid s, a)$ is the next-state distribution (transition kernel),
- $l : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the cost function,
- $\gamma \in [0, 1)$ is the discount factor.



Policies and Optimal Control

Policy

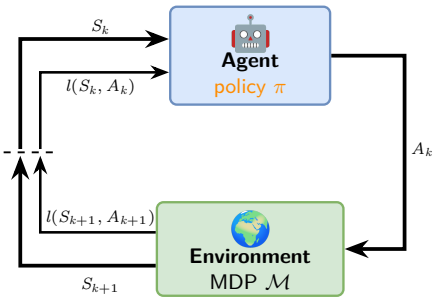
A policy maps states to actions, possibly randomly:

$$\pi : \mathcal{S} \to \mathrm{Dist}(\mathcal{A}), \qquad A_k \sim \pi(\cdot \mid S_k).$$

Optimal Control Problem

Given MDP $\mathcal{M}$, find an optimal policy $\pi^\star$ that minimizes expected discounted cost from each initial state s:

$$\pi^\star \in \arg\min_\pi \; \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k l(S_k, A_k) \mid S_0 = s \right].$$



Value and Action-Value Functions

Value Functions

$$V^\pi(s) \coloneqq \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k l(S_k, A_k) \mid S_0 = s\right],$$

$$Q^\pi(s,a) \coloneqq \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k l(S_k, A_k) \mid S_0 = s,\ A_0 = a\right].$$

Optimal Value Functions

$$V^\star(s) \coloneqq \min_\pi V^\pi(s),$$

$$Q^\star(s,a) \coloneqq \min_\pi Q^\pi(s,a).$$

Recovering Opt. Actions from $Q^\star$

$$\operatorname{supp}\big(\pi^\star(\cdot \mid s)\big) \subseteq \arg\min_a Q^\star(s,a).$$

- support = actions with nonzero probability
- optimal policies share $V^\star, Q^\star$
- deterministic optimal policies exist

Outline

From Optimal Control to Computational Methods

Optimal control problem of an MDP

$$\pi^\star \in \arg\min_\pi \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k l(S_k, A_k) \,\middle|\, S_0 = s\right], \qquad A_k \sim \pi(\cdot \mid S_k),\ S_{k+1} \sim P(\cdot \mid S_k, A_k)$$

MPC approximation

- explicit prediction model f
- finite horizon and terminal value $\bar{V}$
- optimization over a open-loop action plan $u_0, \ldots, u_{N-1}$ instead of policies π.
- constraints and engineering structure

RL approximation

- transition data (S, A, l, S^+)
- value functions V_w, Q_w
- neural network policies π_θ
- temporal-difference learning and stochastic gradients

Same objective, different computational approximations.

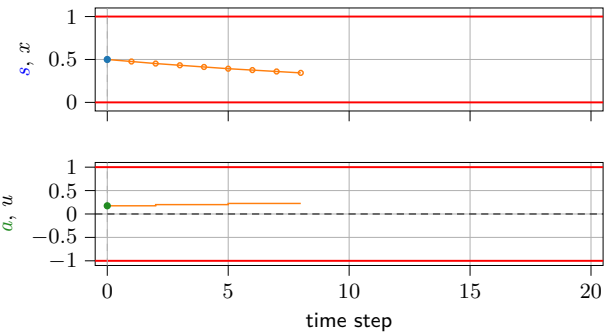
MPC: Plan, Act, Repeat

At current state s

$$\min_{x,u} \quad \bar{V}(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k),$$
$$h(x_k, u_k) \geq 0, \quad x_0 = s.$$

- optimize open-loop sequence $u_{0:N-1}$ over horizon N
- predict with model f and terminal value $\bar{V}$
- apply only $a = u_0^\star$, observe, repeat

$$\pi^{\text{MPC}}(s) = u_0^\star$$



prediction vs. realized states and applied actions

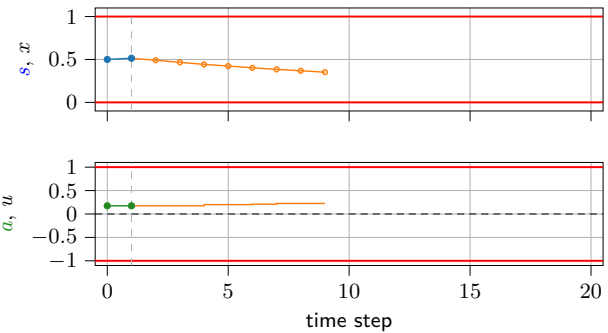
MPC: Plan, Act, Repeat

At current state s

$$\min_{x,u} \quad \bar{V}(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k),$$
$$h(x_k, u_k) \geq 0, \quad x_0 = s.$$

- optimize open-loop sequence $u_{0:N-1}$ over horizon N
- predict with model f and terminal value $\bar{V}$
- apply only $a = u_0^\star$, observe, repeat

$$\pi^{\text{MPC}}(s) = u_0^\star$$



prediction vs. realized states and applied actions

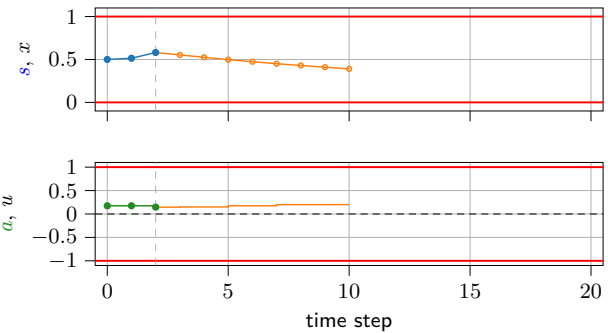
MPC: Plan, Act, Repeat

At current state s

$$\min_{x,u} \quad \bar{V}(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k),$$
$$h(x_k, u_k) \geq 0, \quad x_0 = s.$$

- optimize open-loop sequence $u_{0:N-1}$ over horizon N
- predict with model f and terminal value $\bar{V}$
- apply only $a = u_0^\star$, observe, repeat

$$\pi^{\text{MPC}}(s) = u_0^\star$$



prediction vs. realized states and applied actions

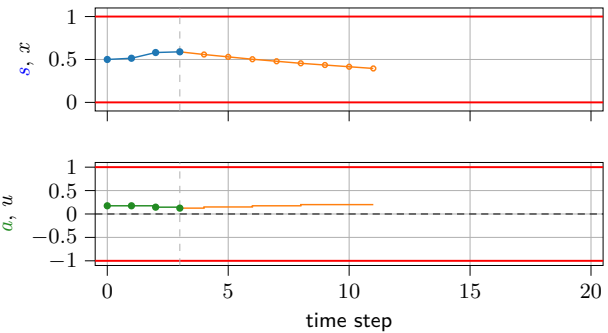
MPC: Plan, Act, Repeat

At current state s

$$\min_{x,u} \quad \bar{V}(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k),$$
$$h(x_k, u_k) \geq 0, \quad x_0 = s.$$

- optimize open-loop sequence $u_{0:N-1}$ over horizon N
- predict with model f and terminal value $\bar{V}$
- apply only $a = u_0^\star$, observe, repeat

$$\pi^{\text{MPC}}(s) = u_0^\star$$



prediction vs. realized states and applied actions

RL: Train From Data, Then Act

Training

1. Collect transition data (S, A, S^+) from
   - $\mathcal{M}$ real system
   - $\mathcal{M}^{\text{sim}}$ simulator
   - $\mathcal{D}$ offline interaction data
2. Fit policy/value approximations, often with neural networks:

$$\pi_\theta, \qquad V_w, \qquad Q_w.$$

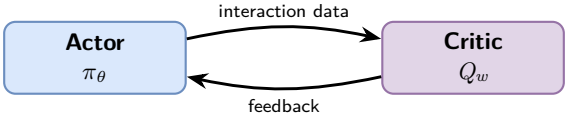
Inference

At deployment, a learned policy π is often just one neural network (NN) forward pass:

Plain RL often moves computation from deployment to training.

MPCRL can tune MPC policies that still plan online.

RL with Actor-Critic Methods



Critic: evaluate actions

- learn from transitions (S, A, S^+)
- approximate Q^{π_θ}: $Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$
- fit a Bellman target, $A^+ \sim \pi_\theta(\cdot \mid S)$

$$\min_w \mathbb{E}\Big[\big(l + \gamma Q_{\bar{w}}(S^+, A^+) - Q(S,A)\big)^2\Big]$$

Actor: choose actions

- policy samples $A \sim \pi_\theta(\cdot \mid S)$
- choose actions reducing predicted cost
- critic provides the training signal

$$\min_\theta \mathbb{E}_{A\sim\pi_\theta(\cdot|S)}[Q_w(S,A)]$$

Contrasting both Approaches

	MPC	**RL**
Main objects	finite-horizon OCP	policy, value functions, actor and critic
Information source	explicit model, costs, constraints	data, simulator, interaction
Computation	online optimization	offline or continual training
Typical strength	structure, constraints, interpretability	flexibility, cheap inference, data adaptation
Typical challenge	online cost and model requirements	safety, generalization, guarantees

These largely orthogonal strengths motivate combinations of MPC and RL.

Outline

Summary

What we saw so far

- MPC and RL are two views on control: planning with models vs. learning from data.
- MDPs provide the common language: states, actions, dynamics, costs, and policies.
- MPC defines a policy by repeatedly solving finite-horizon optimal control problems.
- RL learns policies and value functions from interaction data.

What comes next

- How MPC and RL can be combined in different architectures.
- Why learning MPC parameters can improve closed-loop behavior.
- How differentiable programming enables learning through MPC.
- How software tools make these ideas practical.

Big picture: MPC and RL have largely orthogonal strengths; MDPs give us the language to compare and combine them.

Any questions?