

Topics in Reinforcement Learning:
AlphaZero, ChatGPT, Neuro-Dynamic Programming,
Model Predictive Control, Discrete Optimization, Applications
Arizona State University
Course CSE 691, Spring 2025

Links to Textbook, Videolectures, and Slides at
<http://web.mit.edu/dimitrib/www/RLbook.html>

Prof. Dimitri P. Bertsekas (dimitrib@mit.edu)
and
Dr Yuchao Li (yuchaoli@asu.edu)

Lecture 4

POMDP, Systems with Changing Parameters, Adaptive Control
Model Predictive Control

- 1 Problem Formulations - Review
- 2 Partial State Observation Problems
- 3 Problems with Changing Parameters - Adaptive Control
- 4 Model Predictive Control
- 5 What We Have Done - Where We Are Going - What We Will Not Cover

Review of Problem Formulation. An Important Example: POMDP

An informal recipe: First define the controls (the easy part), then the stages (and info available at each stage), and then the states (the harder part)

- Define as state x_k something that “summarizes” the past for purposes of future optimization, i.e., **as long as we know x_k , all past information is irrelevant.**
- **Rationale:** The controller applies action that depends on the state. So the state must subsume all info that is useful for decision/control.
- **The ultimate test:** Write a correct DP algorithm (with well-defined system equation, cost per state, constraints, prob. descriptions, etc). **Pay attention to the math!**

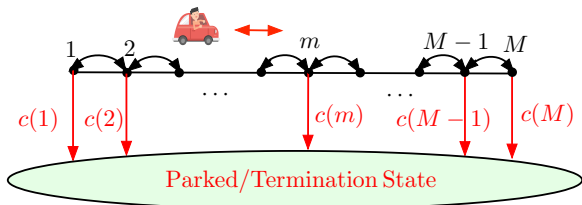
There may be multiple useful ways to define a valid state

- An important example is **POMDP** (Partial Observation Markovian Decision Problems).
- At time k , instead of observing x_k , we obtain a measurement z_k that is “related” to x_k . So x_k is not observable - **we need to redefine the state!**
- At time k all we have is the “information state”

$$I_k = (z_0, u_0, z_1, u_1, \dots, z_k, u_{k-1})$$

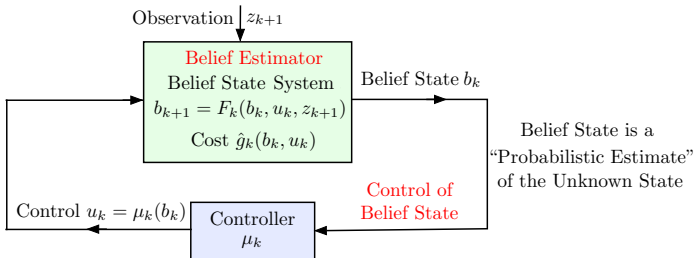
- It can serve as state. Other possibilities exist (e.g., the “belief state”; see next).

Parking with a Deadline: An Example of Partial State Observation



- At each time step, move one spot in either direction. Decide to park or not at spot m (if free) at cost $c(m)$. **If we have not parked by time N there is a large cost C**
- We observe the free/taken status of only the spot we are in. Parking spots may change status at the next time step with some probability.
- The free/taken status of the spots is “estimated” in a “probabilistic sense” based on the observations (the free/taken status of the spots visited ... when visited)
- **What should the “state” be?** It should summarize all the info needed for the purpose of future optimization
- **First candidate for state:** The entire information vector up to the present time.
- **Another candidate:** The “belief state”, i.e., the conditional probabilities of the free/taken status of all the spots: $p(1), p(2), \dots, p(M)$, conditioned on all the observations so far

Partial State Observation Problems: Reformulation via Belief State

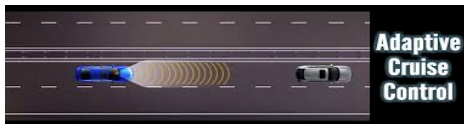


- **We introduce a belief estimator:** $F_k(b_k, u_k, z_{k+1})$ is the next belief state, given current belief state b_k , u_k is applied, and observation z_{k+1} is obtained (if not mathematically available, there are methods to estimate it by simulation)
- The reformulated DP algorithm has the form

$$J_k^*(b_k) = \min_{u_k \in U_k} \left[\hat{g}_k(b_k, u_k) + E_{z_{k+1}} \left\{ J_{k+1}^*(F_k(b_k, u_k, z_{k+1})) \mid b_k, u_k \right\} \right]$$

- $J_k^*(b_k)$ denotes the optimal cost-to-go starting from belief state b_k at stage k
- U_k is the control constraint set at time k
- $\hat{g}_k(b_k, u_k) = E_{x_k, w_k} \{ g_k(x_k, u_k, w_k) \mid b_k, u_k \}$ is the expected cost of stage k : with distribution of (x_k, w_k) determined by b_k and the distribution of w_k

Changing Problem Parameters: Adaptive Control (1960s →)



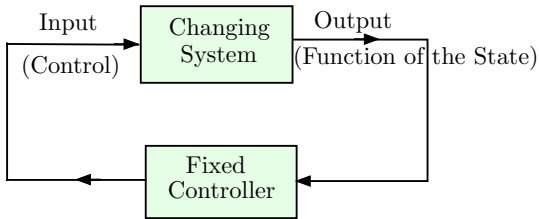
Example: A cruise control-type problem (keep velocity close to a target level \bar{x})

- Control car velocity evolution: $x_{k+1} = ax_k + bu_k$ ($a < 1$ models friction, wind drag, etc, $b > 0$ depends on road, number of passengers, etc)
- Cost over N stages: $(x_N - \bar{x})^2 + \sum_{k=0}^{N-1} ((x_k - \bar{x})^2 + ru_k^2)$, where $r > 0$ is given
- ... but a and b are changing all the time; they may be measured with error

Adaptive control deals with such situations. Three possibilities:

- Ignore the changes in parameters; design a fixed controller that is robust (“works” for a broad range of parameters).
- Try to estimate the parameters, and use the estimates to modify the controller
 - ▶ On-line replanning by optimization; modify the controller to make it optimal for the current set of estimates (a certainty equivalence approach).
 - ▶ On-line replanning by rollout with a base policy whose cost values are computed using the current parameter estimates. This is a simpler (approximate) reoptimization.
- View the adaptive control problem as a POMDP and try to deal approximately.

Robust Control

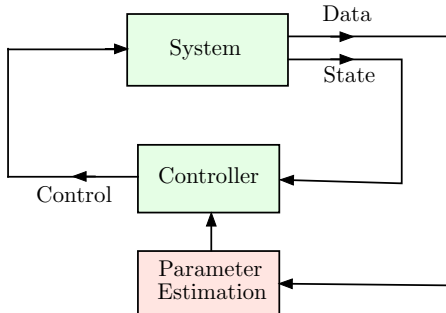


- **We ignore the changes in the system**
- Make no attempt to estimate/learn them
- Hope that a fixed controller will yield "acceptable" performance throughout the range of system changes

Unsophisticated ... but there is an important time-honored successful example

- **PID control** (Proportional-Integral-Derivative)
- Involves three parameters, which are tuned experimentally/heuristically
- Applies to single input-single output case (output: the error from some "set point")
- No math model of the system is needed

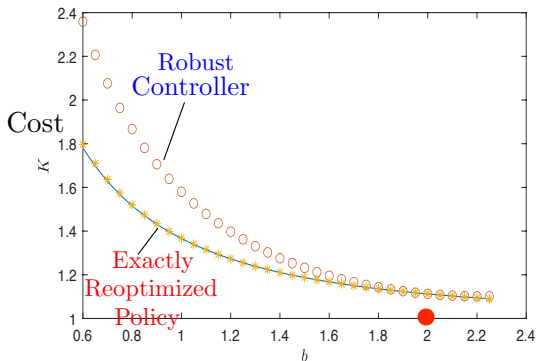
On-Line Replanning by Optimization (Indirect Adaptive Control)



Two-phase alternation: System identification \leftrightarrow Controller reoptimization

- Introduce on-line estimation/identification of changing parameters
- Recompute the controller so it is optimal for the current set of parameters
- This can be time-consuming
- There are some serious issues regarding reliable parameter identification (simultaneously with control); see the textbook

Comparison of Robust Control and On-Line Replanning

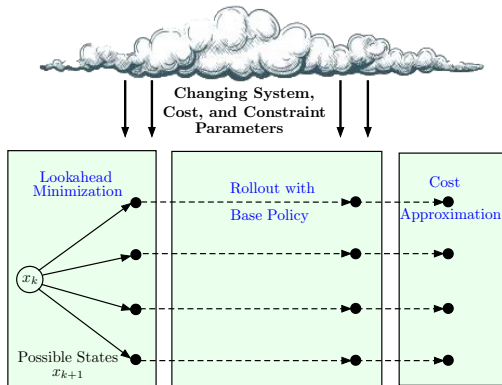


One-dimensional linear-quadratic example:

$$x_{k+1} = x_k + bu_k, \quad \text{Cost} = \lim_{N \rightarrow \infty} \sum_{k=0}^{N-1} (x_k^2 + ru_k^2)$$

Quadratic cost coefficient as b changes. Robust controller is optimal for $b = 2$

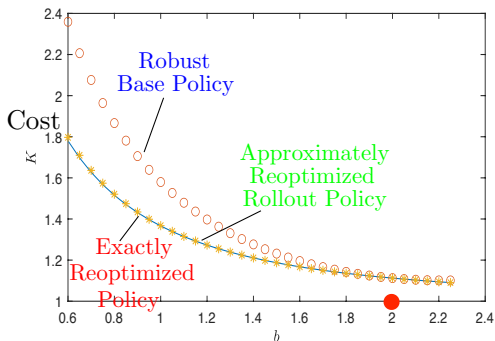
On-Line Replanning by Rollout/Approximation in Value Space



Use a robust policy as a base policy for rollout
No controller reoptimization; this is faster

- Introduce new parameter estimates in the lookahead minimization and the rollout
- Continue to use the same base/robust policy
- Possibly recalculate the base/robust policy in the background

Comparison of On-Line Replanning by Reoptimization and by Rollout



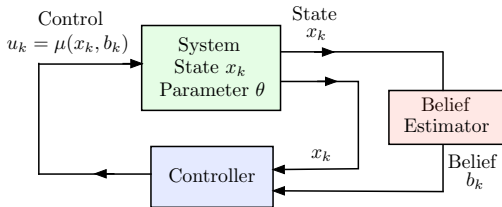
Performance comparison of on-line replanning by rollout and by optimization
Note the effect of Newton's method

One-dimensional linear-quadratic example:

$$x_{k+1} = x_k + bu_k, \quad \text{Cost} = \lim_{N \rightarrow \infty} \sum_{k=0}^{N-1} (x_k^2 + ru_k^2)$$

Quadratic cost coefficient as b changes. Base policy is optimal for $b = 2$

A (More Principled) POMDP Approach to Adaptive Control



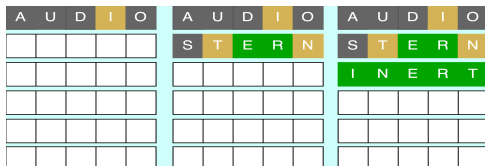
Deterministic system $x_{k+1} = f(x_k, \theta, u_k)$, $\theta \in \{\theta^1, \dots, \theta^m\}$: unknown parameter

- View θ as part of an augmented state (x_k, θ) that is partially observed
- **Belief state:** $b_{k,i} = P\{\theta = \theta^i \mid I_k\}$ (estimated on-line), where $I_k = (x_0, \dots, x_k, u_0, \dots, u_{k-1})$ is the information vector
- **Exact DP algorithm for optimal cost function J_k^* :**

$$J_k^*(I_k) = \min_{u_k} \sum_{i=1}^m b_{k,i} \left(g(x_k, \theta^i, u_k) + J_{k+1}^*(I_k, f(x_k, \theta^i, u_k), u_k) \right)$$

- **Approximation in value space:** Use approximation $\tilde{J}^i(f(x_k, \theta^i, u_k))$ in place of $J_{k+1}^*(I_k, f(x_k, \theta^i, u_k), u_k)$. Minimize over u_k to obtain one-step lookahead policy
- **Example 1:** \tilde{J}^i is the cost function of the optimal policy corresponding to θ^i
- **Example 2:** \tilde{J}^i is the cost function of a known policy assuming $\theta = \theta^i$ (this is **rollout**)

An Example: The Wordle Puzzle



θ : the unknown mystery word

x_k : list of possible mystery words (given the guesses so far)

u_k : guess word selected at time k

- Find a mystery word θ using a minimal number of successive 5-letter guess words
- **View θ as part of an augmented state (x_k, θ) that is partially observed**
- Joint work with S. Bhambri and A. Bhattacharjee (started as term paper for the 2022 offering of this class); see ArXiv paper on-line
- Apply rollout with one of several base heuristics
- To be discussed in more detail at a later lecture

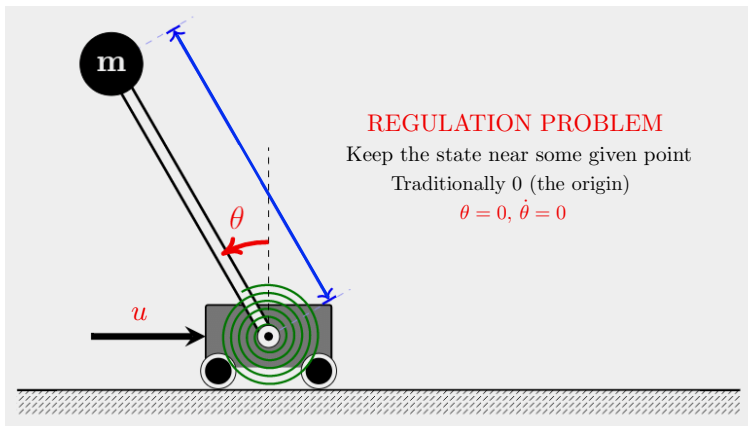
Rollout Performance = 3.5231 vs the Optimal = 3.5084 average # of guesses
Within < 0.5% more guesses from the optimal policy - **On-line answer within 1-3 secs**

IT SCALES WITH PROBLEM SIZE

A Fifteen-Minute Break

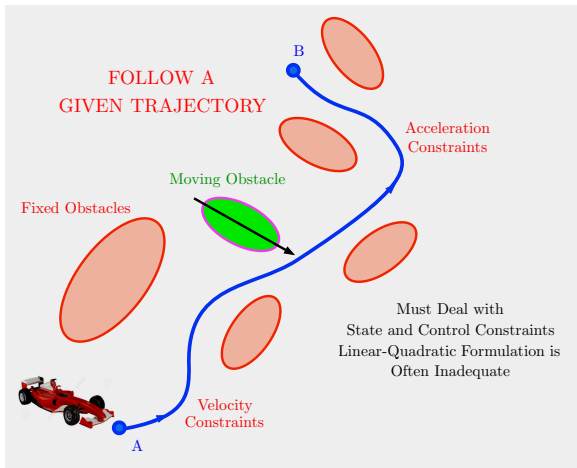
Catch our breath and think about issues relating to the first half of the lecture.
Ask questions when you return.

Classical Control Problem I: Control Around a Reference Point



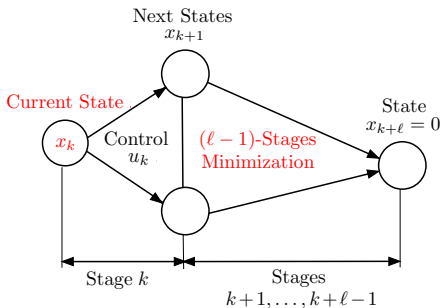
Keep the state of the system close to a reference point (usually 0)

Classical Control Problem II: Path Planning



Keep the state of the system close to a planned trajectory
State constraints can be very important
On-line path replanning may be needed

Model Predictive Control - A Form of Approximation in Value Space



$$\text{System: } x_{k+1} = f(x_k, u_k)$$

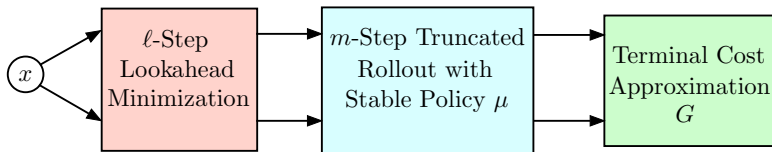
$$\text{Cost: } g(x_k, u_k) \geq 0, \text{ for all } (x_k, u_k)$$

The system can be kept at the origin at zero cost by some control

Consider undiscounted infinite horizon; we want to keep the system near 0

- **Original form of MPC:** We minimize the cost function over the next ℓ stages while requiring $x_{k+\ell} = 0$
- At the current state x_k , we apply the first control of the minimizing sequence, discard the other controls
- **This is rollout w/ base heuristic the min that drives $x_{k+\ell}$ to 0 in $(\ell - 1)$ steps**
- Well-suited for on-line replanning
- We neglect for the moment (the often very important) state constraints

Model Predictive Control with Terminal Cost Approximation



- In a common variant of MPC, **we use a nonnegative terminal cost $G(x_{k+\ell})$** in the ℓ -stage MPC problem, instead of driving the state to 0 in ℓ steps:

$$\min_{u_t, t=k, \dots, k+\ell-1} \left[G(x_{k+\ell}) + \sum_{t=k}^{k+\ell-1} g(x_t, u_t) \right]$$

- This can be viewed as approximation in value space with multistep lookahead
- **Truncated rollout with some base policy may also be introduced**
- The problem may also include state constraints, in which case we obtain one of the most general forms of MPC
- **When is the MPC policy stable?** G must be in the region of stability; see the textbook. Multistep lookahead and truncated rollout help in this respect.

Model Predictive Control with State Constraints

- It is often important to deal with **additional state constraints of the form $x_k \in X$** , where X is some subset of the state space
- Then the MPC problem to be solved at the k th stage must be modified
- **Assuming that the current state x_k belongs to X** , the MPC problem is

$$\min_{u_t, t=k, \dots, k+\ell-1} \left[G(x_{k+\ell}) + \sum_{t=k}^{k+\ell-1} g(x_t, u_t) \right],$$

subject to the control constraints

$$u_t \in U(x_t), \quad t = k, \dots, k + \ell - 1,$$

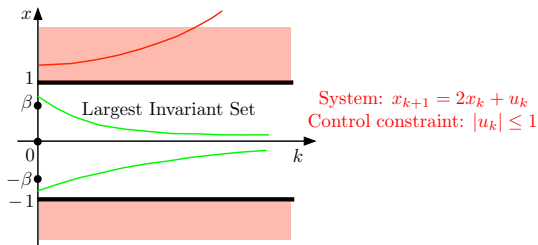
and the state constraints

$$x_t \in X, \quad t = k + 1, \dots, k + \ell$$

- The control \tilde{u}_k thus obtained is required to generate a state $x_{k+1} \in X$
- Important difficulty: **There is no guarantee that this problem has a feasible solution for all initial states $x_k \in X$** . There are ways to deal with this (see the textbook).

There are also approximations to the state constraints $x_t \in X$, using **control barrier functions**. These are computationally more convenient.

A One-Dimensional Example for MPC with State Constraints



- Consider state constraints of the form $x_k \in X$, for all k , where

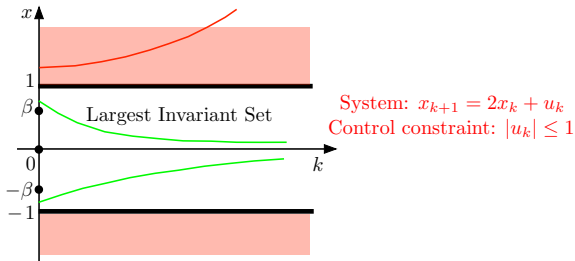
$$X = \{x \mid |x| \leq \beta\}$$

- If $\beta > 1$, the state constraint cannot be satisfied for all initial states $x_0 \in X$.
- Reason: If we take $x_0 = \beta > 1$, then $2x_0 > 2$ and $x_1 = 2x_0 + u_0$ will satisfy $x_1 > x_0 = \beta$ for any value of u_0 with $|u_0| \leq 1$.
- Arguing similarly, the entire sequence of generated states $\{x_k\}$ will satisfy

$$x_{k+1} > x_k \quad \text{for all } k, \quad x_k \uparrow \infty.$$

- The state constraint can be satisfied only for x_0 in the set $\hat{X} = \{x \mid |x| \leq 1\}$
- Sets like \hat{X} are called **invariant** (see the next slide)

Invariance Requirement for the State Constraints



- To guarantee feasibility of the MPC problem with state constraints $x_k \in X$ for all k , **an invariance condition must be satisfied by X**
for every $x \in X$, there exists $u \in U(x)$ such that $f(x, u) \in X$
- **How do we compute an invariant subset of a given constraint set?**
- This is necessarily an **off-line calculation**. Cannot be easily performed during on-line play
- It turns out that given X , there exists a **largest possible invariant subset of X** and it can be computed in the limit with an algorithm that resembles value iteration.
- There also other/simpler possibilities for computing invariant subsets of X

Overview of What we Have Done

We aimed for an overview of the approximate DP/RL landscape; **a foundation for deeper development of other RL topics (and getting you started on your term paper).**

We have described in varying levels of depth the following:

- The algorithmic foundation of **exact DP** in all its major forms: deterministic and stochastic, discrete and continuous, finite and infinite horizon.
- **Approximation in value space with one-step and multistep lookahead**, the workhorse of RL, which underlies its major success stories, including AlphaZero.
- The fundamental division between **off-line training and on-line play** in the context of approximation in value space. **Their synergy through Newton's method.**
- The fundamental methods of **policy iteration and rollout**, the former being primarily an off-line method, and the latter being primarily a less ambitious on-line method. Connections with Newton's method.
- Some **major models with a broad range of applications**, such as discrete optimization, POMDP, multiagent problems, adaptive control, and model predictive control.
- The use of **function approximation**, which has been a recurring theme in our presentation. We have hinted at several points some of the principal schemes for approximation, based on neural networks and feature-based architectures.

The first chapter of the textbook provides a foundational platform for exploring at a deeper level various algorithmic methodologies, such as:

- **Rollout, its variants, and its applications**; e.g., discrete/deterministic, multiagent, etc; **Chapter 2**.
- **Sequential decision making in special contexts involving changing system parameters**, sequential estimation, Bayesian optimization, and simultaneous system identification and control; **Chapter 2**.
- **Off-line training for approximation in value and policy space** using neural networks and other approximation architectures; **Chapter 3**.
- Stochastic algorithms, such as temporal difference methods and Q-learning, used for off-line policy evaluation, in the context of approximate policy iteration.
- Sampling methods to collect data for off-line training, in the context of cost and policy approximations.
- Statistical estimates and efficiency enhancements of various sampling methods used in simulation-based schemes.
- A deeper exploration of control system design methodologies such as model predictive control, and its applications in robotics and automated transportation.

Remember there are two types of term papers;

- **A mini-research project** (a team of up to 3 students is allowed)
- **Read-and-report**: A critical synthesis of 2-3 related research papers (about 8-15 pages, no team collaboration, NO ChatGPT please)

Detailed guidelines have been posted on Canvas

If you wish to consult with us about your ideas, please send us a writeup of 1-2 pages by email (not through Canvas), and we may schedule a meeting if needed.

A one-page proposal is expected by the end of spring break.

We will cover:

- Deterministic rollout and variations
- Rollout for stochastic problems

Homework: Think about your term paper

Watch videolecture 5 from 2023 ASU course offering