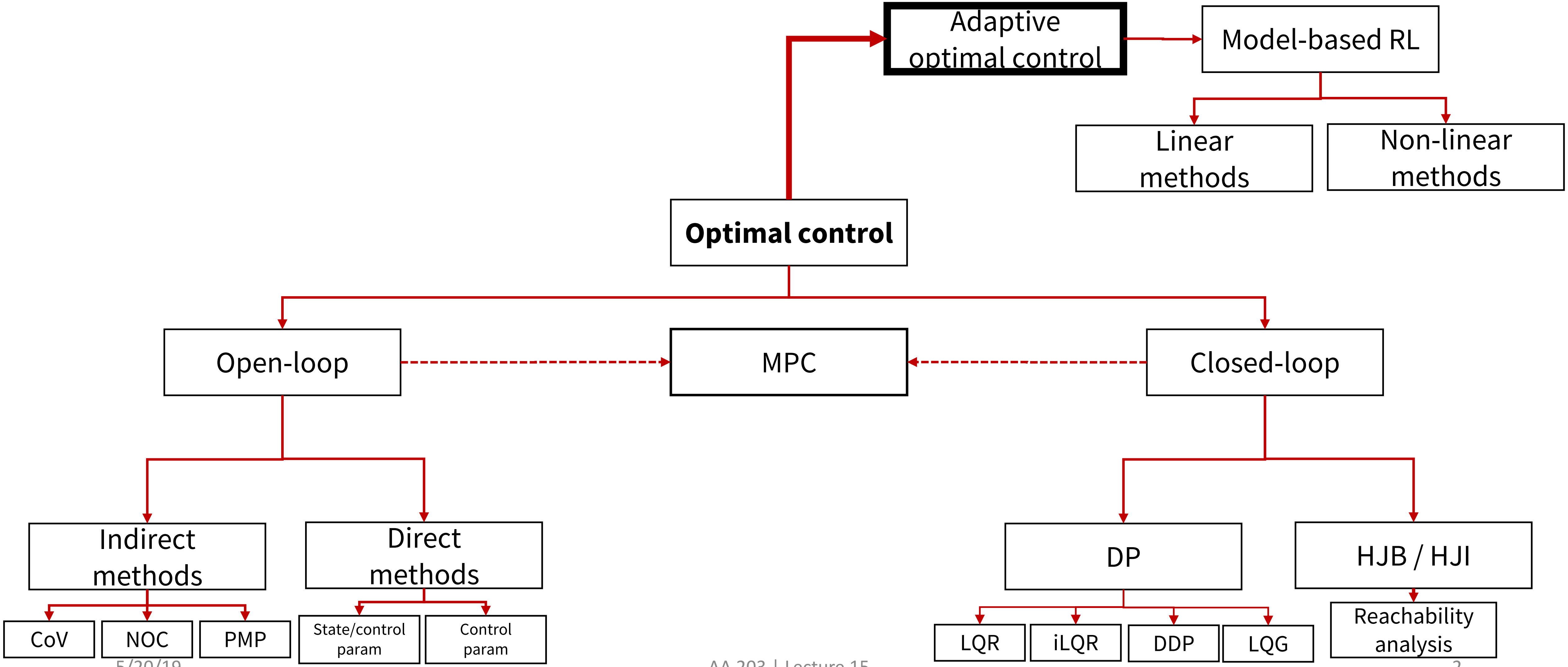
AA203
Optimal and Learning-based Control

Adaptive Optimal Control

Stanford University

Autonomous Systems Lab

Roadmap


Adaptive optimal control
Model-based RL
Linear methods
Non-linear methods
Optimal control
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI
CoV
NOC
PMP
State/control param
Control param
LQR
iLQR
DDP
LQG
Reachability analysis

Problem statement

- Up until now, we have aimed to control a (possibly stochastic) system

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)$$

under a given cost function, subject to state or action constraints

- For remainder of the class, we will look at controlling a system of the form

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k; \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is an **unknown** vector of parameters governing state evolution

Approaches

If we don’t know the exact state evolution, what should we do? Many options:

- In many cases, when the unknown parameters have only a small effect, a feedback controller will adequately compensate for model error
- We can use robust control approaches (e.g. minimax control strategies)
- We can use observed state transitions to attempt to estimate $\boldsymbol{\theta}$ or improve our control strategy

What can we learn?

- We can directly attempt to estimate $\boldsymbol{\theta}$, then use optimal control strategies to plan a controller given the model
  - Commonly referred to as *indirect adaptive control* or *model-based reinforcement learning*
- Learning the value function is not useful because it is not actionable, but can learn the Q function

$$Q(\mathbf{x}, \mathbf{u}) = E[c(\mathbf{x}, \mathbf{u}) + J(\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}; \boldsymbol{\theta}))]$$

and choose actions via maximizing

- Can directly learn the policy, π
  - Commonly referred to as *direct adaptive control* or *model-free reinforcement learning*

How does learning happen?

- We will mostly look at the case in which we attempt to learn $\boldsymbol{\theta}$, but will touch briefly on the other cases
- Three possible learning settings:
  - **"Zero" episodes**: the system identification approach, in which learning is done based on data gathered before operation
  - **One episode**: want to learn and re-optimize our controller online -> this is the standard setting for adaptive control
  - **Multiple episodes**: interact with the environment in episodes, in which the system is reset at the start of each episode; learning and policy optimization can happen between episodes -> this is the standard setting for reinforcement learning

System identification for learning-based control

- For many problems, we don't need to learn online
- A standard control engineering pipeline is to do experiments in advance to build a *data-driven* model of the dynamics
- Then, we can use this model for planning and control without further learning
- Relies on having an engineer in the loop for learning, designing experiments, resetting the system, etc.
- *Linear regression* is one of the main system id tools

Linear least-squares

Consider linear relation

$$\mathbf{y} = H\boldsymbol{\theta} + \mathbf{v}$$

where $\mathbf{y}$ is a given $N \times 1$ vector, H is a given $N \times n$ matrix, $\boldsymbol{\theta}$ is a *unknown* $n \times 1$ vector, and $\mathbf{v}$ is the residual

- We assume that $N \geq n$, thus the system is *overdetermined*
- A least-squares solution is one that minimizes the length of the residual vector, that is

$$||\mathbf{y} - H\boldsymbol{\theta}||^2$$

Linear least-squares solution

Theorem: A vector $\boldsymbol{\theta}^*$ is a minimizer of the cost function

$$J(\boldsymbol{\theta}) \coloneqq \|\mathbf{y} - H\boldsymbol{\theta}\|^2$$

If, and only if, it satisfies the always consistent *normal equations*

$$H'H\boldsymbol{\theta}^* = H'\mathbf{y}$$

Proof: The proof follows by differentiation

Linear least-squares solution

Lemma: When H has full rank n, there is a unique $\boldsymbol{\theta}^*$ satisfying the normal equations, which is given by

$$\boldsymbol{\theta}^* = (H'H)^{-1}H'\mathbf{y}$$

Proof: The proof entails showing that H being full rank implies that $H'H$ is non-singular

Statistical version of least-squares

- In many applications (such as system id), $\boldsymbol{\theta}$ is a deterministic but unknown vector while $\mathbf{v}$ is a random noise vector, with known mean and covariance, say $E[\mathbf{v}] = 0$ and $E[\mathbf{v}\mathbf{v}'] = R$
- In this case, the least-squares estimator

$$\boldsymbol{\theta}^* = (H'H)^{-1}H'\mathbf{y}$$

will also be random, with mean

$$E[\boldsymbol{\theta}^*] = E[(H'H)^{-1}H'\mathbf{y}] = \boldsymbol{\theta}$$

- Thus, least-squares is an *unbiased* estimator

Linear regression for system id

Assume the unknown parameters appear linearly in the model

$$\mathbf{y} = \Phi(\mathbf{x}, \mathbf{u})\boldsymbol{\theta} + \mathbf{v}$$

where $\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n]$ is a vector of measured outputs, $\Phi = [\phi(\mathbf{x}_0, \mathbf{u}_0)', \phi(\mathbf{x}_1, \mathbf{u}_1)', \dots, \phi(\mathbf{x}_N, \mathbf{u}_N)']'$ is a matrix of regressors, and $\mathbf{v}$ represents i.i.d., zero-mean, constant variance noise

The goal is to find $\boldsymbol{\theta}^*$ that minimizes squared error criterion

$$||\mathbf{y} - \Phi\boldsymbol{\theta}||^2 \longleftarrow \text{Least-squares problem}$$

Linear regression for system id

- As seen before, the solution is

$$\boldsymbol{\theta}^* = (H'H)^{-1}H'\mathbf{y}$$

- Gauss-Markov theorem: $\boldsymbol{\theta}^*$ is the **best linear unbiased estimator** (for any noise distribution that obeys assumptions)
- If noise distribution is Gaussian, $\boldsymbol{\theta}^*$ is the maximum likelihood estimator

Example: first order model

- Consider system with dynamics

$$x(t+1) = ax(t) + bu(t) + v(t)$$

- Linear regression representation
  - $\phi_t = [x(t), u(t)], t = 0, \ldots, N$
  - $\theta = [a, b]'$

- Practically, least squares can be written in recursive form for efficiency

Performance questions

The system identification approach leads to several questions:

- How much data is required to learn the model? How can we quantify a “good” estimate? We care about controller performance, not model accuracy, so do we require an accurate model?
- How should we design the inputs used for data collection? What if an engineer can’t intervene to prevent system failure during data collection?
- What if our system does not fall in the class of systems we are considering?

Adaptive control

- Broadly, adaptive control aims to perform *online adaptation* of the policy to improve performance
- This can be done via directly updating the policy or updating the model and re-optimizing or re-computing the controller
- Most adaptive control work does not consider the *optimal adaptive control* problem; they focus on proving stability of the coupled controller and adaptive component

Adaptive control approaches

Encompasses a large variety of techniques, including :

- Model reference adaptive control (MRAC)
- Model identification adaptive control (MIAC)
- Dual control
- Model-free
  - policy adaptation
  - iterative learning control

Model reference adaptive control (MRAC)

- A model reference adaptive controller is composed of four parts:
  1. A plant containing unknown parameters
  2. A *reference model* for compactly specifying the desired output
  3. A feedback control law containing adjustable parameters
  4. An adaptation mechanism for updating the adjustable parameters

- The reference model provides the ideal plant response which the adaptation mechanism should seek in adjusting the parameters

Example of MRAC control

- Consider the control of a mass on a frictionless surface

$$m\ddot{x} = u$$

- Assume a human operator provides the positioning command $r(t)$ to the control system
- A reasonable way of specifying the ideal response of the controlled mass to the external command $r(t)$ is to use the reference model

$$\ddot{x}_m + \lambda_1 \dot{x}_m + \lambda_2 x_m = \lambda_2 r(t)$$

where the reference model output x_m is the *ideal* output

Example of MRAC control

- If the mass is known exactly, one can achieve perfect tracking via

$$u = m(\ddot{x}_m - 2\lambda\dot{\tilde{x}} - \lambda^2\tilde{x})$$

  where $\lambda > 0$ and $\tilde{x} \coloneqq x - x_m$ is the tracking error

- This control leads to exponentially convergent tracking dynamics

$$\ddot{\tilde{x}} + 2\lambda\,\dot{\tilde{x}} + \lambda^2\tilde{x} = 0$$

Example of MRAC control

- If the mass is *not* known exactly, we can use the control law
$$u = \hat{m}(\ddot{x}_m - 2\lambda\dot{\tilde{x}} - \lambda^2\tilde{x})$$
which contains the adjustable parameter $\hat{m}$
- This control leads to the closed-loop dynamics
$$m\dot{s} + \lambda m s = \tilde{m}v$$
where:
  - s is a combined tracking error measure, defined by $s = \dot{\tilde{x}} + \lambda\tilde{x}$
  - the signal quantity v is given by $v = \ddot{x}_m - 2\lambda\dot{\tilde{x}} - \lambda^2\tilde{x}$
  - and the parameter estimation error is $\tilde{m}$= $\hat{m} - m$
- The tracking error is related to the parameter error via a stable filter

Example of MRAC control

- One way of adjusting parameter $\hat{m}$ is to use the (nonlinear) update law

$$\dot{\hat{m}} = -\gamma v s$$

where $\gamma > 0$ is called the *adaptation gain*

- Stability and convergence can be analyzed via Lyapunov theory
- Consider Lyapunov function candidate

$$V = \frac{1}{2}\left[ms^2 + \frac{1}{\gamma}\widetilde{m}^2\right]$$

- Its derivative is $\dot{V} = -\lambda m s^2$
- Thus $s \to 0$, and hence $\tilde{x} \to 0$ and $\dot{\tilde{x}} \to 0$

MRAC

- An excellent reference for systematic MRAC design is: Jean-Jacques Slotine, Weiping Li, Applied Nonlinear Control, Chapter 8
- If the reference signal $r(t)$ is very simple, such as zero or a constant, it is possible for many vectors of parameters, besides the ideal parameter vector, to lead to tracking error convergence
- However, if the reference signal $r(t)$ is so complex that only the "true" parameter vector can lead to tracking convergence, then one shall have parameter convergence -> *persistent excitation condition*

Model identification adaptive control

- MIAC (also referred to as self-tuning) simply combines model estimation with a controller that uses the estimated model
- Important distinction between *certainty-equivalent* and *cautious* approaches
  - **Certainty-equivalent:** maintains point estimate of model and uses that model for policy selection/optimization. Note that unlike the LQG setting, certainty-equivalence is sub-optimal.
  - **Cautious:** Maintains measure of estimator uncertainty, incorporates the uncertainty into the controller. This is often overly robust because it does not account for future info gain!

MRAC vs. MIAC

- MRAC and MIAC arise from two different perspectives:
  1. parameters in MRAC are updated so as to minimize tracking error between the plant output and the reference model output
  2. parameters in MIAC are updated so as to minimize the data-fitting error
- MIAC controllers are in general more flexible, as one can couple various controllers with various estimators
- However, correctness of MIAC controllers is more difficult to guarantee, as if the signals are not rich, the estimated parameters may not be close to the "true" values, and stability and convergence may not be ensured
- In contrast, for MRAC, stability and convergence are usually guaranteed *regardless* of the richness of the signals

Dual control

- Most adaptive control is “passive”: it does not incorporate the value of information or actively explore
- Dual control augments the state with the estimate of the unknown parameters, and uses the joint dynamics
- By performing DP in this “hyperstate”, one can find a controller that optimally probes/explores the system, or, in other words, optimally trades off exploration/exploitation
- Practically, designing dual controllers is difficult, so sub-optimal exploration heuristics are used

Example: adaptive control of the LQ system

- Consider simplest possible linear-quadratic discrete-time system, of the form

$$x(t+1) = ax(t) + bu(t) + w(t)$$

with cost $c(x, u) = qx^2 + ru^2$

- One of the first proposed approaches was Simon [1956], in the context of inventory control
- He proposed a certainty-equivalent approach, where a and b are recursively computed via least squares, and the optimal policy is computed with respect to those estimates

Example: adaptive control of the LQ system

- Becker et al. [1985] showed that this approach can converge to incorrect estimates with positive probability, which can lead to suboptimal performance!
- Alternative approach: dual control. We will assume that the only unknown parameter is b to further simplify the problem. Then, the hyperstate is $z_k = \left[x_k, \hat{b}_k\right]'$. Note that the problem is nonlinear, so can not use LQR for control design
- Bar-Shalom and Tse [1976] solve the problem for two timesteps, but a general solution is difficult. This is the simplest possible problem!

Example: adaptive control of the LQ system

- Passive adaptive control may converge to a poor solution and dual control is largely intractable, so common practice is to turn to exploration heuristics
- To achieve convergence of the estimator, we need persistence of excitation [Lai and Wei, 1982]
- Many approaches to guaranteeing asymptotic convergence

Example: adaptive control of the LQ system

- A variety of schemes exist for performing exploratory actions
  - White noise injection into control [Becker et al., 1985]
  - “Strategic” injection of white noise into control [Lai and Wei, 1987]: inject white noise into control at certain times
  - Cost-biased estimator [Campi and Kumar, 1998]: adds term to estimator that biases model estimate toward lower cost models
  - “Bet on the best”: act with respect to best non-falsified model (optimism) [Bittanti and Campi, 2006]
- These methods all analyzed in terms of asymptotic convergence. Recently, considerable work on finite-horizon performance or regret analysis, see [Abbasi-Yadkori and Szepesvari, 2011] or recent work by Recht et al.

Next time

- Reinforcement learning for linear systems