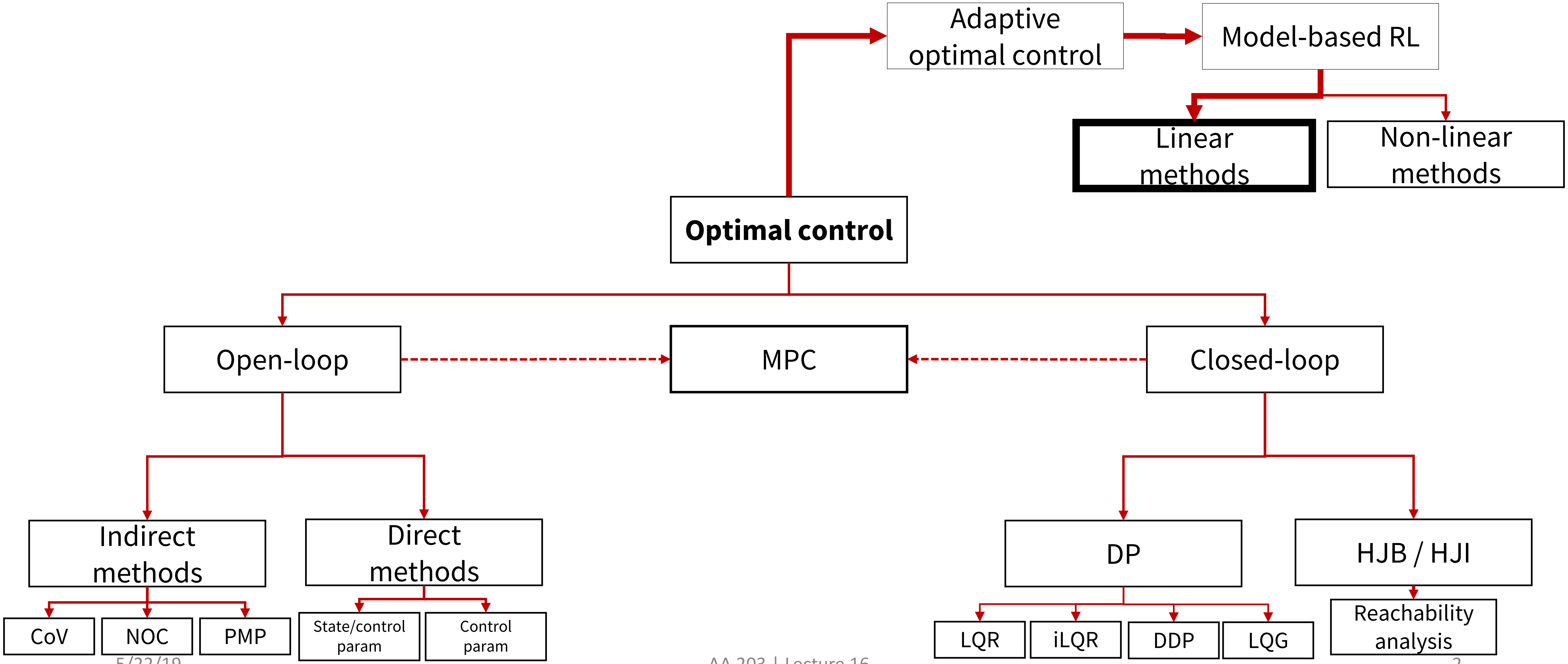
AA203
Optimal and Learning-based Control

Linear methods for model-based RL

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

Model learning for control

- Linear vs. nonlinear models
  - In linear systems, **local model knowledge is global model knowledge**: data we gather is universally valid
  - In nonlinear systems, local model knowledge is inaccurate elsewhere
  - This can lead to a planner/controller exploiting a model that is inaccurate
  - Nonlinear regression is hard in general; more expressivity for model class generally means more possible inaccuracies for a controller to exploit
- Today: we will talk about model-based RL for linear problems, and locally linear methods for nonlinear problems
- Next two weeks: approaches for nonlinear model-based RL

Model-based RL for linear systems

- Linear systems without constraints: can use adaptive LQR
  - For non-quadratic reward, can use iLQR, the estimation of the model does not change
- Linear system with constraints
  - In this setting, can turn to learning-based MPC (among other methods)
- Nonlinear problems in which performance along a trajectory is sufficient (as opposed to global policy): can use locally-linear models (e.g. time indexed collection of linear models)

Adaptive LQR

Given an initial model $\hat{A}, \hat{B}; D = \emptyset$

for $j = 1, \ldots$

  Perform Riccati recursion to compute optimal policy for current $\hat{A}, \hat{B}$

  for $k = 0, \ldots, N-1$

    $\mathbf{u}_k = F_k \mathbf{x}_k$

    $\mathbf{x}_{k+1} = A\,\mathbf{x}_k + B\,\mathbf{u}_k + \mathbf{w}_k$

    $D \leftarrow D \cup \{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})\}$

  end

  Compute $\hat{A}, \hat{B}$ given D via least-squares

end

Adaptive LQR extensions

- Many possible extensions beyond baseline algorithm
  - Can add exploration to action selection (e.g., add white noise)
  - Can regularize the model estimation
    - Ridge regression: minimize $\sum_k || A\,\mathbf{x}_k + B\mathbf{u}_k - \mathbf{x}_{k+1}||_2^2 + \; ||A||_2^2 + ||B||_2^2$
    - LASSO regression: minimize $\sum_k || A\,\mathbf{x}_k + B\mathbf{u}_k - \mathbf{x}_{k+1}||_2^2 + \lambda_1||A||_1 + \lambda_2||B||_1$
  - May wish to sacrifice some performance to be robust to poor models: can incorporate model uncertainty into action selection (e.g., [Dean et al., 2018])

- For a review of modern continuous control with an emphasis on the LQ problem, see Recht [2018]

Learning-based MPC

- Learning-based MPC has been a large topic of study in MPC for the past decade; examples include
  - A. Aswani, H. Gonzalez, S.S. Sastry, and C. Tomlin. “Provably safe and robust learning-based model predictive control.” *Automatica*, 2013.
  - U. Rosolia and F. Borrelli. "Learning Model Predictive Control for Iterative Tasks. A Data-Driven Control Framework." *IEEE Transactions on Automatic Control* (2017).
  - M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli. "Adaptive MPC for Iterative Tasks." In *IEEE Conference on Decision and Control*, 2018.
  - T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. “Learning-based Model Predictive Control for Safe Exploration.” In *IEEE Conference on Decision and Control,* 2018.

Learning-based MPC

Main lines of research:

- Learning-based MPC (LMPC): improve the control design (e.g., the penalty term and the terminal constraint set) using data -> connection to iterative learning

- Adaptive MPC (AMPC): learn and improve the uncertain part of the model to improve controller performance -> connection to adaptive control

- Adaptive, learning MPC: combine LMPC with AMPC

- Note: terminology is not consistent in the literature!

Adaptive MPC for iterative tasks

- Consider uncertain LTI system

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + E\boldsymbol{\theta}_a + \mathbf{w}_t$$

  subject to the constraint

$$F\mathbf{x}_t + G\mathbf{u}_t \leq h$$

- $\mathbf{w}_t \in W$ is random process noise, and $\boldsymbol{\theta}_a$ is an *unknown*, constant offset
- It is assumed that the same task is executed repeatedly, with iteration cost

$$V_j = \sum_{t=0}^{\infty} c(\mathbf{x}_t^j, \mathbf{u}_t^j)$$

Adaptive MPC for iterative tasks

- The goal is to design via MPC a controller that solves the problem

$$V_j(\mathbf{x}_S) = \min_{\mathbf{u}_0^j, \mathbf{u}_1^j(\cdot)\ldots} \sum_{t=0}^{\infty} c(\bar{\mathbf{x}}_t^j, \mathbf{u}_t^j(\bar{\mathbf{x}}_t^j))$$

Nominal state

$$\text{subject to} \quad \mathbf{x}_{t+1}^j = A\mathbf{x}_t^j + B\mathbf{u}_t^j(\mathbf{x}_t^j) + E\boldsymbol{\theta}_a + \mathbf{w}_t^j,$$

$$F\mathbf{x}_t^j + G\mathbf{u}_t^j \leq h, \qquad \forall \mathbf{w}_t^j \in W$$

$$\mathbf{x}_0^j = \mathbf{x}_S$$

- Goal: use data to learn unknown offset $\boldsymbol{\theta}_a$, while exploiting iterative nature of the problem to improve performance at next iteration

Adaptive MPC for iterative tasks

- Consider affine state feedback policies of the form

$$\mathbf{u}_t^j(\mathbf{x}_t^j) = K(\mathbf{x}_t^j - \bar{\mathbf{x}}_t^j) + \mathbf{v}_t^j$$

- Nominal state evolves according to

$$\bar{\mathbf{x}}_{t+1}^j = A\bar{\mathbf{x}}_t^j + B\,\mathbf{v}_t^j$$

- Error state ($\mathbf{e}_t^j := \mathbf{x}_t^j - \bar{\mathbf{x}}_t^j$) evolves according to

$$\mathbf{e}_{t+1}^j = (A + BK)\mathbf{e}_t^j + E\boldsymbol{\theta}_a + \mathbf{w}_t^j, \qquad \mathbf{e}_0^j = \mathbf{0}$$

Adaptive MPC for iterative tasks

- Define feasible parameter set
  $$\Theta^j = \{\boldsymbol{\theta}: \mathbf{x}_t^i - A\mathbf{x}_{t-1}^i - B\mathbf{u}_{t-1}^i - E\boldsymbol{\theta} \in W, \forall i \in [0, \dots, j-1], \forall t \geq 0\}$$
  clearly: $\Theta^{j+1} \subseteq \Theta^j$
- Reformulation of constraints:
  $$F\bar{\mathbf{x}}_t^j + G\mathbf{v}_t^j + (F + GK)\mathbf{e}_t^j \leq h$$
- The challenge is to ensure robust satisfaction of the above constraint for all $\mathbf{w}_t^j \in W$ and in the presence of unknown offset $\boldsymbol{\theta}_a$

Adaptive MPC for iterative tasks

$$\min_{\mathbf{v}^j_{t|t},\ldots,\mathbf{v}^j_{t+N-1|t}} \sum_{k=t}^{t+N-1} c(\bar{\mathbf{x}}^j_{k|t}, \mathbf{v}^j_{k|t}) + \boxed{P^{j-1}(\bar{\mathbf{x}}^j_{t+N|t})}$$

quantifies the performance of the closed-loop trajectories in the previous iterations -> guarantees *iterative performance*

subject to $\bar{\mathbf{x}}^j_{k+1|t} = A\bar{\mathbf{x}}^j_{k|t} + B\,\mathbf{v}^j_{k|t}$

$$F\bar{\mathbf{x}}^j_{k|t} + G\mathbf{v}^j_{k|t} \le h - \max_{\mathbf{e}_t \in \boxed{\mathcal{E}^j}} (F + GK)\mathbf{e}_t$$

minimal robust positive invariant set for the error dynamics with respect to set Θ^j -> guarantees *adaptation*

$$\bar{\mathbf{x}}^j_{t|t} = \bar{\mathbf{x}}^j_t, \qquad \bar{\mathbf{x}}^j_{t+N|t} \in \boxed{\mathcal{CS}^{j-1}}$$

collection of all nominal state trajectories up to iteration $j-1$ that have converged to the origin

- The controller applies

$$\mathbf{u}^j_t(\mathbf{x}^j_t) = K(\mathbf{x}^j_t - \bar{\mathbf{x}}^j_t) + \mathbf{v}^{j,*}_{t|t}$$

Adaptive MPC for iterative tasks

1. Initialize feasible parameter set Θ^j, compute initial minimal robust positive invariant set $\mathcal{E}^j$, and set $t = 0$
2. Compute $\mathbf{v}_{t|t}^{j,*}$ and apply control $\mathbf{u}_t^j(\mathbf{x}_t^j)$ to the system
3. Set $t = t + 1$, and return to step 2 until the end of jth iteration
4. At the end of jth iteration, update set $\mathcal{CS}^j$ for next iteration
5. Set $j = j + 1$. Return to step 1.

Summary

This algorithm showcases a number of key ideas underlying learning-based and adaptive MPC:

- Iterative learning
- Model adaptation
- Robust constraint satisfaction
- Construction of a safe terminal constraint set
- Penalty term promoting improved performance

Model-based RL using locally-linear models

- If we have nonlinear dynamics but only care about local performance for a task, we can learn time-varying linear models
- For finite horizon control task, $k = 0, \dots, N$, can learn collection of models $\{A_k, B_k\}_{k=0}^{N-1}$, where each model is locally valid around a trajectory
- Can then use standard linear optimal control techniques, e.g. LQR
- Still requires careful application: as you roll out the controller, control optimization will move your trajectory away from the nominal trajectory, possibly resulting in model error

Linear models in apprenticeship learning

- Time-varying locally-linear models have seen application especially in apprenticeship learning (i.e. learning from demonstration)
  - Atkeson and Schaal [1997] use a handful of experiments to have a robot swing up and balance a pendulum successfully
  - Apprenticeship is necessary to initialize trajectory and thus avoid model mismatch

- Linearly-parameterized models:
  - Abbeel et al. identify linear parameters of nonlinear dynamics from apprenticeship (demonstration of goal maneuver by pilot), then used DDP for control
  - This project resulted in successful flight of autonomous helicopter through highly dynamic maneuvers

- Spatially-indexed models:
  - "Locally weighted linear regression", Schaal + Atkeson have many works; e.g. [1994] used LWLR for robot juggling
  - Kolter et al. [2008] use spatially-index models for autonomous driving

Next time

- Methods for nonlinear regression