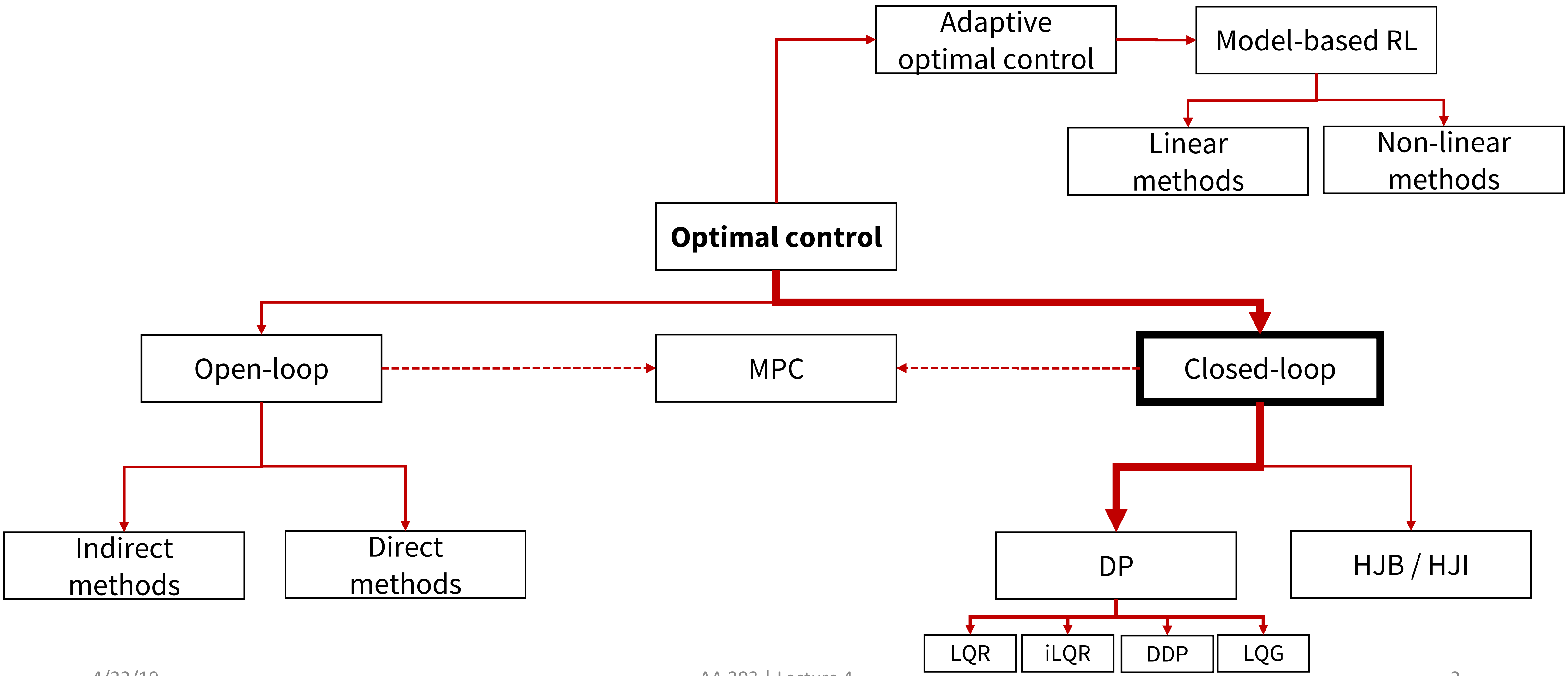
AA203
Optimal and Learning-based Control

iLQR, stochastic optimal control, LQG

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
LQR
iLQR
DDP
LQG

LQR extensions

- Linear tracking problems
- LQR with cross-quadratic cost and affine dynamics
- Non-linear tracking problems
- Using LQR to solve non-linear optimal control problems
  - Iterative LQR
  - Differential dynamic programming

Linear tracking problems

- Imagine you are given a *nominal trajectory*

$$(\overline{\boldsymbol{x}}_0, \ldots, \overline{\boldsymbol{x}}_N), (\overline{\boldsymbol{u}}_0, \ldots, \overline{\boldsymbol{u}}_{N-1})$$

- Assume nominal trajectory satisfies linear dynamics
- Linear tracking problem: find policy to minimize cost

$$\frac{1}{2}(\boldsymbol{x}_N - \overline{\boldsymbol{x}}_N)^T H(\boldsymbol{x}_N - \overline{\boldsymbol{x}}_N) + \frac{1}{2}\sum_{k=0}^{N-1}[(\boldsymbol{x}_k - \overline{\boldsymbol{x}}_k)^T Q(\boldsymbol{x}_k - \overline{\boldsymbol{x}}_k) + (\boldsymbol{u}_k - \overline{\boldsymbol{u}}_k)^T R(\boldsymbol{u}_k - \overline{\boldsymbol{u}}_k)]$$

- Then define dev*iation variables*

$$\delta\boldsymbol{x}_k \coloneqq \boldsymbol{x}_k - \overline{\boldsymbol{x}}_k \text{ and } \delta\boldsymbol{u}_k \coloneqq \boldsymbol{u}_k - \overline{\boldsymbol{u}}_k$$

  and solve standard LQR with respect to deviation variables

LQR with cross-quadratic cost & affine dynamics

- Consider the LQR problem with the generalized cost

$$\frac{1}{2}\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T R_k \boldsymbol{u}_k + \boldsymbol{u}_k^T H_k \boldsymbol{x}_k + \boldsymbol{q}_k^T \boldsymbol{x}_k + \boldsymbol{r}_k^T \boldsymbol{u}_k + c_k$$

- and dynamics

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{d}_k$$

- We can derive an *affine* optimal feedback law for this system via DP recursion

LQR with cross-quadratic cost & affine dynamics

- The cost-to-go at time k takes the form

$$\frac{1}{2}\boldsymbol{x}_k^T P_k \boldsymbol{x}_k + \boldsymbol{p}_k^T \boldsymbol{x}_k + p_k$$

- Optimal control takes the form $\boldsymbol{u}_k^* = \boldsymbol{l}_k + L_k \boldsymbol{x}_k$ with

$$\boldsymbol{l}_k = -(R_k + B_k^T P_{k+1} B_k)^{-1}(\boldsymbol{r}_k + \boldsymbol{p}_{k+1}^T B_k + \boldsymbol{d}_k P_{k+1} B_k)$$

$$L_k = -(R_k + B_k^T P_{k+1} B_k)^{-1}(B_k^T P_{k+1} A_k + H_k)$$

- Equations for the constant/linear/quadratic cost-to-go terms are unwieldy but not hard to derive, and are given in the lecture notes

Nonlinear tracking problems

- Imagine you are given a *feasible nominal trajectory*

$$(\bar{\boldsymbol{x}}_0, \dots, \bar{\boldsymbol{x}}_N), (\bar{\boldsymbol{u}}_0, \dots, \bar{\boldsymbol{u}}_{N-1})$$

- The tracking cost is still quadratic, but the dynamics are now nonlinear

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$$

- To apply LQR, we can linearize around the nominal trajectory

$$\boldsymbol{x}_{k+1} \approx f(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) + \underbrace{\frac{\partial f}{\partial \boldsymbol{x}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)}_{A}\overbrace{(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k)}^{\delta\bar{\boldsymbol{x}}_k} + \underbrace{\frac{\partial f}{\partial \boldsymbol{u}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)}_{B}\overbrace{(\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k)}^{\delta\bar{\boldsymbol{u}}_k}$$

- And apply LQR to the deviation variables (with dynamics $\delta\bar{\boldsymbol{x}}_{k+1} = A\delta\bar{\boldsymbol{x}}_k + B\delta\bar{\boldsymbol{u}}_k$)

Non-linear optimal control problem

- Consider now non-linear optimal control problem

$$\min_{\mathbf{u}} \sum_{k=0}^{N-1} c(\mathbf{x}_k, \mathbf{u}_k)$$
$$\text{subject to } \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$$

- Can we apply LQR-techniques to approximately solve it?

Iterative LQR

- Imagine you are given a *feasible nominal trajectory*

$$(\bar{\boldsymbol{x}}_0, \dots, \bar{\boldsymbol{x}}_N), (\bar{\boldsymbol{u}}_0, \dots, \bar{\boldsymbol{u}}_{N-1})$$

- Linearize the dynamics around feasible trajectory

$$\boldsymbol{x}_{k+1} \approx f(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) + \frac{\partial f}{\partial \boldsymbol{x}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k) + \frac{\partial f}{\partial \boldsymbol{u}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)(\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k)$$

- And Taylor expand cost function around feasible trajectory

$$c(\delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k) = c_k + \underbrace{c_{\boldsymbol{x},k}^T}_{\boldsymbol{q}_k} \delta\boldsymbol{x}_k + \underbrace{c_{\boldsymbol{u},k}^T}_{\boldsymbol{r}_k} \delta\boldsymbol{u}_k + \frac{1}{2}\delta\boldsymbol{u}_k^T \underbrace{c_{\boldsymbol{uu},k}^T}_{R_k} \delta\boldsymbol{u}_k + \frac{1}{2}\delta\boldsymbol{x}_k^T \underbrace{c_{\boldsymbol{xx},k}^T}_{Q_k} \delta\boldsymbol{x}_k + \delta\boldsymbol{u}_k^T \underbrace{c_{\boldsymbol{ux},k}^T}_{H_k} \delta\boldsymbol{x}_k$$

Iterative LQR

- By optimizing over deviation variables (using results for LQR with cross-quadratic cost & affine dynamics), we obtain new solution:

$$\{\bar{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k^*\} \text{ and } \{\bar{\boldsymbol{u}}_k + \delta \boldsymbol{u}_k^*\}$$

- We can then re-linearize and Taylor expand around this new trajectory, and iterate!

Iterative LQR

- Backward pass ($k = N$ to 0):
  - Compute locally linear dynamics, locally quadratic cost around nominal trajectory
  - Solve local approximation of DP recursion to compute control law
  - Compute cost-to-go
- Forward pass ($k = 0$ to N):
  - Use control law to update nominal trajectory
- Iterate until convergence

Algorithmic details

- Need to make sure that the new state / control stay close to the linearization point
  - Add extra penalty on deviations
- Need to decide on termination criterion
  - For example, one can stop when improvement is “small”
- Method can get stuck in local minima → “good” initialization is often critical
- Cost matrices may not be positive definite
  - Regularize them until they are

Differential Dynamic Programming(DDP)

- iLQR first approximates dynamics and cost, then performs exact DP recursion
- DDP instead approximates DP recursion directly
  - Define change in cost-to-go J_k under perturbation $(\delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k)$ as

$$Q(\delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k) := c(\bar{\boldsymbol{x}}_k + \delta\boldsymbol{x}_k, \bar{\boldsymbol{u}}_k + \delta\boldsymbol{u}_k) + J_{k+1}(f(\bar{\boldsymbol{x}}_k + \delta\boldsymbol{x}_k, \bar{\boldsymbol{u}}_k + \delta\boldsymbol{u}_k))$$

  - Then, second order expansion

$$Q(\delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k) \approx \frac{1}{2}\begin{bmatrix} 1 \\ \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \end{bmatrix}^T \begin{bmatrix} 2Q_k & Q_{\boldsymbol{x},k}^T & Q_{\boldsymbol{u},k}^T \\ Q_{\boldsymbol{x},k} & Q_{\boldsymbol{xx},k} & Q_{\boldsymbol{ux},k}^T \\ Q_{\boldsymbol{u},k} & Q_{\boldsymbol{ux},k} & Q_{\boldsymbol{uu},k} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \end{bmatrix}$$

Differential Dynamic Programming`(DDP)

- The optimal control perturbation is

$$\delta \boldsymbol{u}_k^* = \text{argmin}_{\delta \boldsymbol{u}} Q(\delta \boldsymbol{x}_k, \delta \boldsymbol{u})$$

- Leveraging the approximation, one can re-use LQR results and find that the optimal deviation is

$$\delta \boldsymbol{u}_k^* = \boldsymbol{l}_k + L_k \delta \boldsymbol{x}_k$$

- Algorithm proceeds via same forward/backward passes as iLQR

Stochastic optimal control problem (discrete time)

- System: $\boldsymbol{x}_{k+1} = f_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k), k = 0, \ldots, N-1$
- Control constraints: $\boldsymbol{u}_k \in U(\boldsymbol{x}_k)$
- Probability distribution: $P_k(\cdot \mid \boldsymbol{x}_k, \boldsymbol{u}_k)$ of $\boldsymbol{w}_k$
- Policies: $\pi = \{\pi_0 \ldots, \pi_{N-1}\}$, where $\boldsymbol{u}_k = \pi_k(\boldsymbol{x}_k)$
- Expected Cost:

$$J_\pi(\boldsymbol{x}_0) = E_{\boldsymbol{w}_k, k=0,\ldots,N-1}\left[g_N(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} g_k(\boldsymbol{x}_k, \pi_k(\boldsymbol{x}_k), \boldsymbol{w}_k)\right]$$

- Stochastic optimal control problem

$$J^*(x_0) = \min_\pi J_\pi(\boldsymbol{x}_0)$$

Key points

- Discrete-time model
- Markovian model
- Objective: find optimal closed-loop
- Additive cost (central assumption)
- Risk-neutral formulation

Other communities use different notation: Powell, W. B. *AI, OR and control theory: A Rosetta Stone for stochastic optimization*. Princeton University, 2012.

http://castlelab.princeton.edu/Papers/AIOR_July2012.pdf

Principle of optimality

- Let $\pi^* = \{\pi_0^*, \pi_1^*, \ldots, \pi_{N-1}^*\}$ be an optimal policy
- Consider tail subproblem

$$E\left[g_N(\boldsymbol{x}_N) + \sum_{k=i}^{N-1} g_k(\boldsymbol{x}_k, \pi_k(\boldsymbol{x}_k), \boldsymbol{w}_k)\right]$$

and the tail policy $\{\pi_i^*, \ldots, \pi_{N-1}^*\}$

Principle of optimality: The tail policy is optimal for the tail subproblem

The DP algorithm (stochastic case)

Intuition

- DP first solves ALL tail subproblems at the final stage
- At generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length

The DP algorithm (stochastic case)

The DP algorithm

- Start with

$$J_N(\boldsymbol{x}_N) = g_N(\boldsymbol{x}_N)$$

and go backwards using

$$J_k(x_k) = \min_{\boldsymbol{u}_k \in U(\boldsymbol{x}_k)} E_{w_k}\left[g_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) + J_{k+1}\left(f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)\right)\right]$$

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

- Then $J^*(\boldsymbol{x}_0) = J_0(\boldsymbol{x}_0)$ and optimal policy is constructed by setting $\pi_k^*(\boldsymbol{x}_k) = \boldsymbol{u}_k^*$

Example: Inventory Control Problem (1/3)

- Stock available $x_k \in \mathbb{N}$, inventory $u_k \in \mathbb{N}$, and demand $w_k \in \mathbb{N}$
- Dynamics: $x_{k+1} = \max(0, x_k + u_k - w_k)$
- Constraints: $x_k + u_k \leq 2$
- Probabilistic structure: $p(w_k = 0) = 0.1, p(w_k = 1) = 0.7$, and $p(w_k = 2) = 0.2$
- Cost

$$E\left[\underbrace{0}_{g_3(x_3)} + \sum_{k=0}^{2} \underbrace{(u_k + (x_k + u_k - w_k)^2)}_{g_k(x_k, u_k, w_k)} \right]$$

Example: Inventory Control Problem (2/3)

- Algorithm takes form

$$J_k(x_k) = \min_{0\leq u_k \leq 2-x_k} E_{w_k}[u_k + (x_k + u_k - w_k)^2 + J_{k+1}(\max(0, x_k + u_k - w_k))]$$

for $k = 0,1,2$

- For example

$$J_2(0) = \min_{u_2=0,1,2} E_{w_2}[u_2 + (u_2 - w_2)^2] =$$

$$\min_{u_2=0,1,2} u_2 + 0.1(u_2)^2 + 0.7(u_2-1)^2 + 0.2(u_2-2)^2$$

which yields $J_2(0) = 1.3$, and $\pi_2^*(0) = 1$

Example: Inventory Control Problem (3/3)

Final solution:

- $J_0(0) = 3.7$,
- $J_0(1) = 2.7$, and
- $J_0(2) = 2.818$

Problems with imperfect state information

- Now the controller, instead of having perfect knowledge of the state, has access to observations $\boldsymbol{z}_k$ of the form
$$\boldsymbol{z}_0 = h_0(\boldsymbol{x}_0, \boldsymbol{v}_0), \qquad \boldsymbol{z}_k = h_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k), \qquad k = 1,2, \ldots, N-1$$

- The random observation disturbance is characterized by a given probability distribution
$$P_{\boldsymbol{v}_k}(\cdot \mid \boldsymbol{x}_k, \ldots, \boldsymbol{x}_0, \boldsymbol{u}_{k-1}, \ldots, \boldsymbol{u}_0, \boldsymbol{w}_{k-1}, \ldots, \boldsymbol{w}_0, \boldsymbol{v}_{k-1}, \ldots, \boldsymbol{v}_0)$$

- The initial state $\boldsymbol{x}_0$ is also random and characterized by given $P_{\boldsymbol{x}_0}$

Control policies

- Define the *information vector* as

$$\boldsymbol{I}_k = (\boldsymbol{z}_0, \dots, \boldsymbol{z}_k, \boldsymbol{u}_0, \dots, \boldsymbol{u}_{k-1}), \qquad \boldsymbol{I}_0 = \boldsymbol{z}_0$$

- Focus is now on *admissible* policies $\pi_k(\boldsymbol{I}_k) \in U_k$

- We want then to find an admissible policy that minimizes

$$J_\pi = E_{\substack{\boldsymbol{x}_0, \boldsymbol{w}_k, \boldsymbol{v}_k \\ k=0,\dots,N-1}} \left[g_N(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} g_k(\boldsymbol{x}_k, \pi_k(\boldsymbol{I}_k), \boldsymbol{w}_k) \right]$$

Solution strategies

1. Reformulation as a perfect state information problem (main idea: make the information vector the state of the system)
   - Main drawback: state has *expanding* dimension!
2. Reason in terms of sufficient statistics, i.e., quantities that ideally are smaller than $\boldsymbol{I}_k$ and yet summarize all its essential content
   - Main example: conditional probability distribution $P_{\boldsymbol{x}_k|\boldsymbol{I}_k}$
   - Condition probability distribution leads to a decomposition of the optimal controller in two parts:

$$\pi_k\left(P_{\boldsymbol{x}_k|\boldsymbol{I}_k}\right)$$

Actuator

Estimator

LQG

Discrete LQG: find admissible control policy that minimizes

$$E\left[\boldsymbol{x}_N' Q \boldsymbol{x}_N + \sum_{k=0}^{N-1} (\boldsymbol{x}_k' Q_k \boldsymbol{x}_k + \boldsymbol{u}_k' R_k \boldsymbol{u}_k)\right]$$

subject to

- the dynamics $\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{w}_k$
- the measurement equation $\boldsymbol{z}_k = C_k \boldsymbol{x}_k + \boldsymbol{v}_k$

and with $\boldsymbol{x}_0, \{\boldsymbol{w}_k\}, \{\boldsymbol{v}_k\}$, independent and Gaussian vectors (and in addition $\{\boldsymbol{w}_k\}, \{\boldsymbol{v}_k\}$ zero mean)

LQG – solution

Let

- $M_k := E[\boldsymbol{w}_k \boldsymbol{w}_{\boldsymbol{k}}']$
- $N_k := E[\boldsymbol{v}_k \boldsymbol{v}_k']$
- $S := E[(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])']$

LQG – solution

The optimal controller is $\boldsymbol{u}_k = F_k \hat{\boldsymbol{x}}_k$, where

- F_k is the LQR gain
- $\hat{\boldsymbol{x}}_{k+1} = A_k \hat{\boldsymbol{x}}_k + B_k \boldsymbol{u}_k + \Sigma_{k+1|k+1} C'_{k+1} N_{k+1}^{-1} (\boldsymbol{z}_{k+1} - C_{k+1}(A_k \hat{\boldsymbol{x}}_k + B_k \boldsymbol{u}_k))$
- $\hat{\boldsymbol{x}}_0 = E[\boldsymbol{x}_0] + \Sigma_{0|0} C'_0 N_0^{-1} (\boldsymbol{z}_0 - C_0 E[\boldsymbol{x}_0])$
- and matrices $\Sigma_{k|k}$ are *precomputable* (given in the lecture notes)

- Key property: the estimation portion of the optimal controller is an optimal solution of the problem of estimating the state x_k assuming no control takes place, while the actuator portion is an optimal solution of the control problem assuming perfect state information $\rightarrow$ separation principle

Next time

- HJB and continuous-time LQR