Fall 2022 ME424 Modern Control and Estimation

Lecture Note 10
Dynamic Programming &
Linear Quadratic Regulator

Prof. Wei Zhang
Department of Mechanical and Energy Engineering
SUSTech Institute of Robotics
Southern University of Science and Technology

zhangw3@sustech.edu.cn
https://www.wzhanglab.site/

Outline

- General Discrete-Time Optimal Control Problem
- Short Introduction to Dynamic Programming
- Linear Quadratic Regulator

- Closed-loop eigenvalues roughly indicate system response, but do not represent all factors:
  - Same eigenvalues may also have different transient responses
  - We often want control input to be small, which cannot be formally addressed with eigenvalue assignment approach

- Metric-based controller design
  - Represent design objectives in terms a **cost function**
  - Cost functions typically **penalize**
    - state deviation from 0
    - Large control effort
  - These are conflicting goals: larger control can often drive state to zero faster

General Discrete-Time Optimal Control Problem

- Dynamics: $x_{k+1} = f(x_k, u_k)$
- State constraints: $x_k \in X$
- Control constraints: $u_k \in U(x_k)$
- Controller (Control law): $\mu_k: X \to U$
- Control Horizon: $[0, N]$
- Control policy vs. control inputs:
  - Control policy: a sequence of control laws
  - Control inputs: a sequence of control actions

General Discrete-Time Optimal Control Problem

- Closed-loop Dynamics under policy $\pi = \{\mu_0, \mu_1, \dots\}$

- Quantify performance of controller through cost function
  - Running (stage) cost: $l(x_k, u_k)$
  - Terminal cost: $g(x_N)$
  - N-horizon cost: $J_N(x_0, u) = g(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$
  - Infinite horizon cost: $J_\infty(x_0, u) = \sum_{k=0}^{\infty} l(x_k, u_k)$

- **Finite Horizon Optimal Control** $(N < \infty)$
  - For given initial state $z \in \mathrm{R}^n$, find the control input $u_0, u_1, , \ldots, u_{N-1}$ to
    - **Minimize:** $J_N(z, u)$
    - **subject to:** $u_k \in U(x_k)$, control constraint

      $x_{k+1} = f(x_k, u_k)$, $x_0 = z$ system dynamics constraints

    - Here: $U(x_k)$ is the set of state-dependent control action

      e.g. $U(x) = \{u \leq 2x\}$

    - Optimizers $\{u_0^*, \ldots, u_{N-1}^*\}$ depends on the initial state z

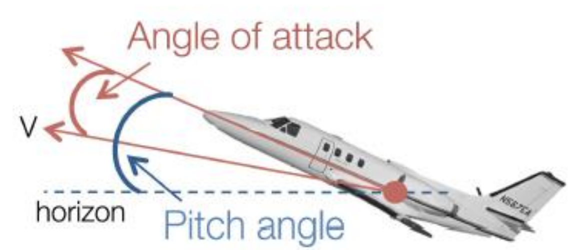
Example I: Cessna Citation Aircraft

Linearized continuous-time model: (at altitude of 5000m and a speed of 128.2 m/sec)

$$\dot{x} = \begin{bmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & \end{bmatrix} x + \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x$$

- Input: elevator angle
- States: x_1: angle of attack, x_2: pitch angle, x_3: pitch rate, x_4: altitude
- Outputs: pitch angle and altitude
- Constraints: elevator angle ± 0.262rad $(\pm 15^\circ)$, elevator rate ± 0.349 rad/s $(\pm 20^\circ/s)$, pitch angle ± 0.650 rad $(\pm 37^\circ)$



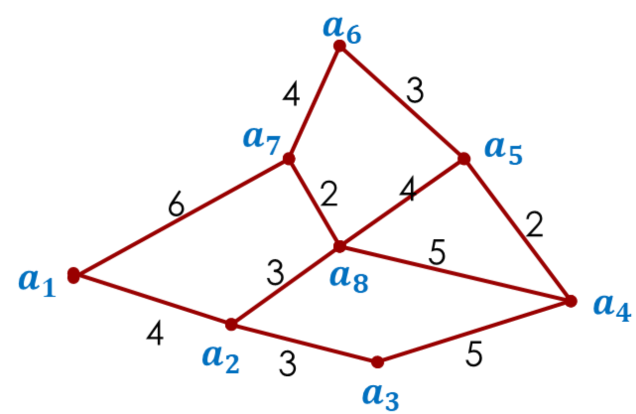
- Open-loop response is unstable (open-loop poles: $0,\ 0,\ -1.5594 \pm 2.29i$)

Example I: Cessna Citation Aircraft

- Obtain DT-Model $dt = 0.25s$
- Choose cost func:
- Choose constraint set:
- Overall optimal control problem:

Example II: Shortest Path Problem

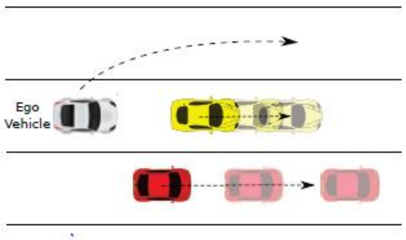
- $X = \{a_1, \dots, a_8\}$; $U(x)$: possible next site to visit
- $x_{k+1} = f(x_k, u_k)$



- Running cost: $l(z, u\ ) =$
- Terminal cost: $g(z) =$
- Optimal control problem:

Example III: Motion Planning for Autonomous Vehicle

- Consider unicycle kinematic model: state $x = (p_x, p_y, \theta, v)$, control $u = (\omega, \alpha)$
- Dynamics: $\dot{x} = \begin{bmatrix} v cos(\theta) \\ v sin(\theta) \\ \omega \\ \alpha \end{bmatrix}$



- Control Goal: Track a give reference $(p_k^d,\ v_k^d,\ \theta_k^d\ )$

Outline

- General Discrete-Time Optimal Control Problem
- **Short Introduction to Dynamic Programming**
- Linear Quadratic Regulator

Dynamic Programming (DP):

- Most important tool for solving deterministic and stochastic optimal control problems
- **Divide & conquer:** The N-horizon optimal solution depends on the $N-1$ horizon optimal solution, which in turns depend on the $N-2$ horizon optimal solution …
- We solve 0-horizon first, then 1-horizon, …, eventual solve the N-horizon optimal control problem.
- The divide & conquer approach is grounded by a fundamental principle: **Bellman's principle of optimality**

Any segment along an optimal trajectory is also optimal among all the trajectories joining the two end points of the segment

Dynamic Programming (DP)

- For arbitrary integer $j \geq 0$, the j-horizon optimal control problem:

$$V_j(z) = \min_{u_0,\ldots,u_{j-1}} \left\{ g(x_j) + \sum_{k=0}^{j-1} l(x_k, u_k) \right\},$$

subject to $\quad x_{k+1} = f(x_k, u_k), \quad x_0 = z$

$\quad u_k \in U(x_k), \quad k = 0, \ldots, j-1$

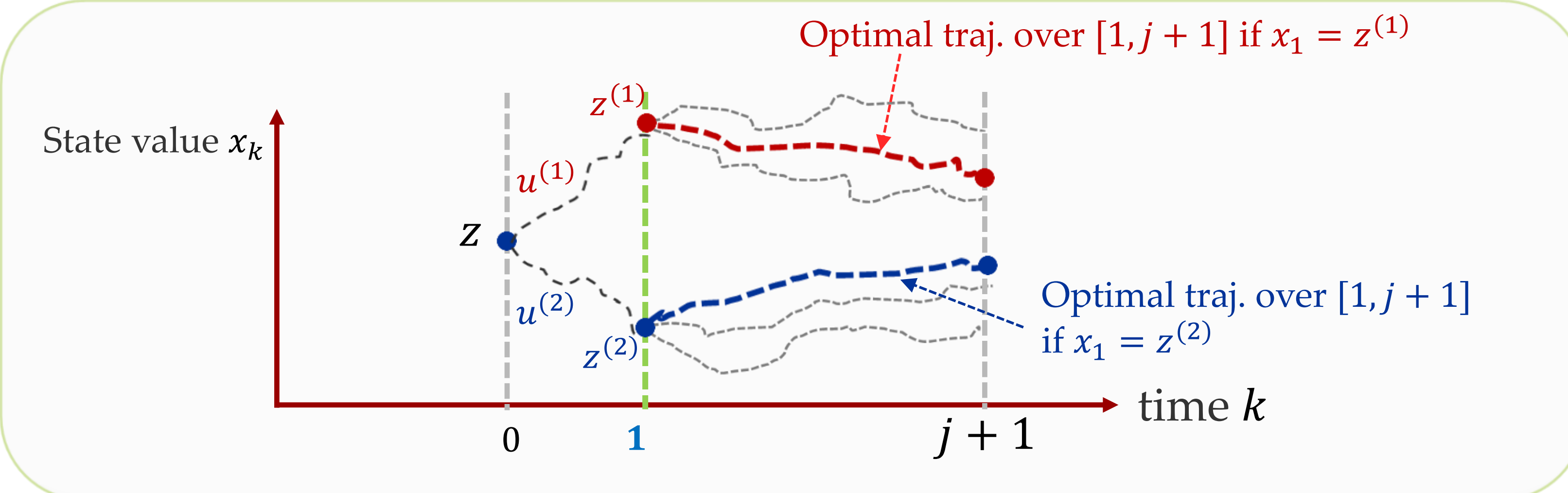
- $V_j^*(z)$: ***j*-horizon value function, i.e.** minimum cost if sys starts from state z when there are j steps left to reach final time

- Let $u_0^*, u_1^*, \ldots, u_{j-1}^*$ is the optimal solution to the above prob. If system is at state z when there are j steps left, the first step of the optimal control is u_0^*, the second step is u_1^*, ….

Dynamic Programming: Value Iteration

- Value Iteration: Compute $V_N(z)$ iteratively from $V_0(z)$
- 0-horizon problem (degenerate case):
- 1-horizon problem
- 2-horizon problem:

- Now suppose we are given $V_j(z)$, need to derive $V_{j+1}(z)$



- What is the optimal control for $j+1$ horizon?
  - Suppose available controls at time 0 are $U(z) = \{u^{(1)}, u^{(2)}\}$
  - Need to compare: $l(z, u^{(1)}) + V_j\left(f(z, u^{(1)})\right)$ and $l(z, u^{(2)}) + V_j\left(f(z, u^{(2)})\right)$
  - The optimal control: $\mu_{j+1}^*(z) = argmin_{u \in U(z)}\{l(z,u) + V_j(f(z,u))\}$
  - The minimum cost: $V_{j+1}(z) = \min_{u \in U(z)}\{l(z,u) + V_j(f(z,u))\}$
- **$\mu_{j+1}^*(z)$: has the following two meanings**
  - the first optimal control action for a $j+1$ horizon problem with initial state z
  - the optimal control action when the system is at state z and there are j+1 steps to go

Value Iteration Algorithm

- System dynamics: $x_{k+1} = f(x_k, u_k)$ with $u_k \in U(x_k)$
- Determine u by solving optimization problem:

Minimize: $J_N(z,u) = g(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$

subject to: control constraint $u_k \in U(x_k)$,

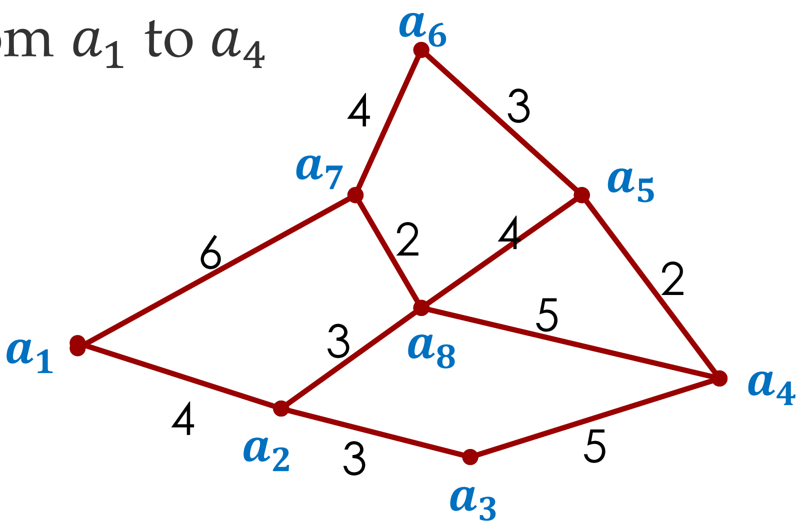
system dynamics $x_{k+1} = f(x_k, u_k)$, $x_0 = z$

- Solve problem through **value iteration**: (namely, iteratively compute the value function for 0-horizon, 1-horizon, …, N-horizon problems)

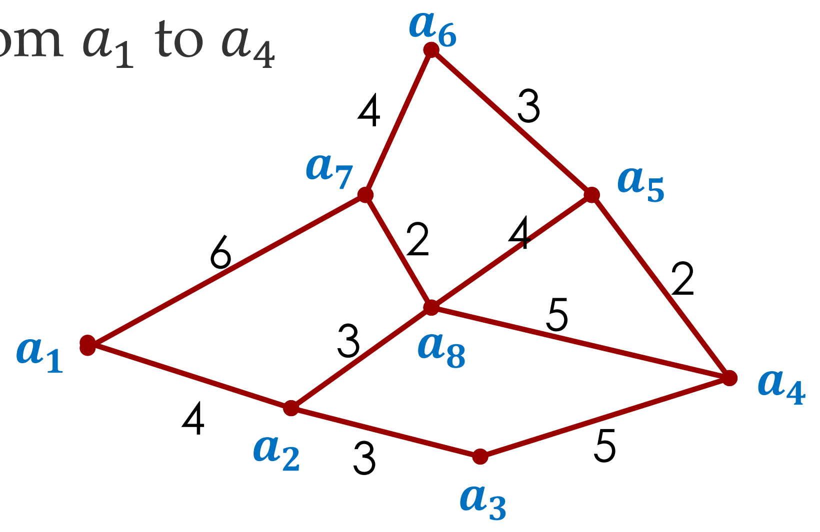
- **Step 0**: (0-horizon): $V_0(z) = g(z)$
- **Step *j***: given $V_j(z)$ and the optimal control laws $\mu_j^*(z), \mu_{j-1}^*, (z)$ … , $\mu_0^*(z)$ for the remaining j steps, compute:
  - $V_{j+1}(z) = \min_{u \in U(z)} \{l(z,u) + V_j(f(z,u))\}$
  - $\mu_{j+1}^*(z) = argmin_{u \in U(z)} \{l(z,u) + V_j(f(z,u))\}$
- $j \leftarrow j + 1$, until $j = N$

- Value iteration algorithm output:
  - Value functions: $V_0(z), \dots, V_N(z)$
  - Optimal control laws: $\mu_j^*(z) = argmin_{u \in U(z)}\{l(z,u) + V_{j-1}f(z,u)\}, j = 1, \dots, N$
    - The optimal control action if sys is at z and there are j steps to go
- How to use these control laws?
  - Optimal system trajectory
  - Time 0: $x_0 = \hat{x}$ → control action: $u_0^* = \mu_N^*(\hat{x})$
  - Time 1: $x_1^* = f(\hat{x}, u_0^*)$ → control action: $u_1^* = \mu_{N-1}^*(x_1^*)$
  - Time 2: $x_2^* = f(x_1^*, u_1^*)$ → control action: $u_2^* = \mu_{N-2}^*(x_2^*)$
  - $\vdots$
  - Time $N-1$: $x_{N-1}^* = f(x_{N-2}^*, u_{N-2}^*)$ → control action: $u_{N-1}^* = \mu_1(x_{N-1}^*)$
  - Time N: $x_N^* = f(x_{N-1}^*, u_{N-1}^*)$
- In general: at time k: **optimal control** $\boldsymbol{u_k^* = \mu_{N-k}^*(x_k^*)}$

- **Example:** Find shortest path from a_1 to a_4
  - $V_0(z) =$
  - $V_1(z) =$



- **Example:** Find shortest path from a_1 to a_4



- **Outline**

  - General Discrete-Time Optimal Control Problem

  - Short Introduction to Dynamic Programming

  - **Linear Quadratic Regulator**

- **Linear Quadratic Regulator (LQR):**
  - N-horizon LQR: Find control sequence $u_0, u_1, \dots, u_{N-1}$ to minimize $J_N(z,u)$, subject to **linear dynamics constraints**:
    $$x_{k+1} = Ax_k + Bu_k, \qquad x_0 = z$$
    where : $J_N(x_0, u) = x_N^T Q_f x_N + \sum_{k=0}^{N-1}\left[x_k^T Q x_k \; + u_k^T R u_k \right]$
  - Infinite-horizon LQR: Find control sequence $u_0, u_1, \dots,$ to minimize $J_\infty(x_0, u)$ subject to linear dynamics constraints: $x_{k+1} = Ax_k + Bu_k$
    where $J_\infty(x_0, u) = \sum_{k=0}^{\infty}\left[x_k^T Q x_k^T + u_k^T R u_k^T\right]$
  - $z^T P z$: quadratic cost term, penalizing deviation from 0, e.g.:
    - if $P = I$, then $z^T P z = \left|\left|z\right|\right|^2$
    - if $P = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, then $z^T P z = z_1^2 + 2z_2^2$, penalizes z_2 more than z_1

- Solution of LQR using Dynamic Programming (DP)
  - $V_0(z) = z^T Q_f z$
  - Suppose at j-horizon value function is: $V_j(z) = z^T P_j z$

    Compute $(j+1)$-horizon value function using DP

$$V_{j+1}(z) = \min_{u \in R^m} \{ l(z,u) + V_j(f(z,u)) \}$$
$$= \min_{u \in R^m} \{ z^T Q z + u^T R u + (Az + Bu)^T P_j (Az + Bu) \}$$
$$= \min_{u \in R^m} \{ u^T (R + B^T P_j B) u + 2 z^T A^T P_j B u + z^T (Q + A^T P_j A) z \}$$
$$\triangleq \min_{u \in R^m} h(u)$$

- $\frac{\partial h}{\partial u}(u) = 2u^T (R + B^T P_j B) + 2 z^T A^T P_j B = 0$

Optimizer: $\mu_{j+1}^*(z) = -(R + B^T P_j B)^{-1} B^T P_j A z \triangleq -K_{j+1} z$

where $K_{j+1} = (R + B^T P_j B)^{-1} B^T P_j A$

- Derivation (cont.)
  - $V_{j+1}(z) = \min_{u \in R^m} h(u) = h(u^*)$

  $= (-K_j z)^T (R + B^T P_j B)(-K_j z) + 2z^T A^T P_j B(-K_j z) + z^T (Q + A^T P_j A) z$

  $= z^T \left(Q + A^T P_j A - A^T P_j B (R + B^T P_j B)^{-1} B^T P_j A\right) z$

  $\triangleq z^T P_{j+1} z$

  where $P_{j+1} \triangleq Q + A^T P_j A - A^T P_j B (R + B^T P_j B)^{-1} B^T P_j A$

- If at time k, the state is at x_k, then the optimal control applied at time k is

$$u_k^* = \mu_{N-k}^*(x_k) = K_{N-k} x_k$$

- **Summary of LQR**
  - Value function is given by: $V_j(z) = z^T P_j z$, where P_j is given by the so-called Riccati recursion:

$$P_{j+1} = Q + A^T P_j A - A^T P_j B\left(R + B^T P_j B\right)^{-1} B^T P_j A$$

  - To compute the LQR controller:
    - Start from initial matrix: $P_0 = Q_f$
    - Riccati recursion: $P_j \leftarrow P_{j-1}$
    - Compute optimal feedback gain: $K_j = \left(R + B^T P_{j-1} B\right)^{-1} B^T P_{j-1} A$
  - Apply LQR controller:
    - Start from an IC: x_0
    - For $k = 0, \dots, N-1$
      - Compute: $u_k^* = -K_{N-k} x_k^*$,
      - $x_{k+1}^* = A x_k^* + B u_k^*$

- Infinite horizon case:
  - It can be proved that if (A, B) is controllable and (A, G) is observable, where $Q = G^T G$, then as $N \to \infty$,
  - $P_j \to P^*$, and $K_j \to K^*$, with $|\lambda(A - BK^*)| < 1$
  - P^* and K^* satisfy the algebraic equations:

$$P^* = A^T \left[P^* - P^* B\left(R + B^T P^* B\right)^{-1} B^T P^*\right] A + Q$$

$$\mathrm{K}^* = \left(R + B^T P^* B\right)^{-1} B^T P^* A$$

Coding Example