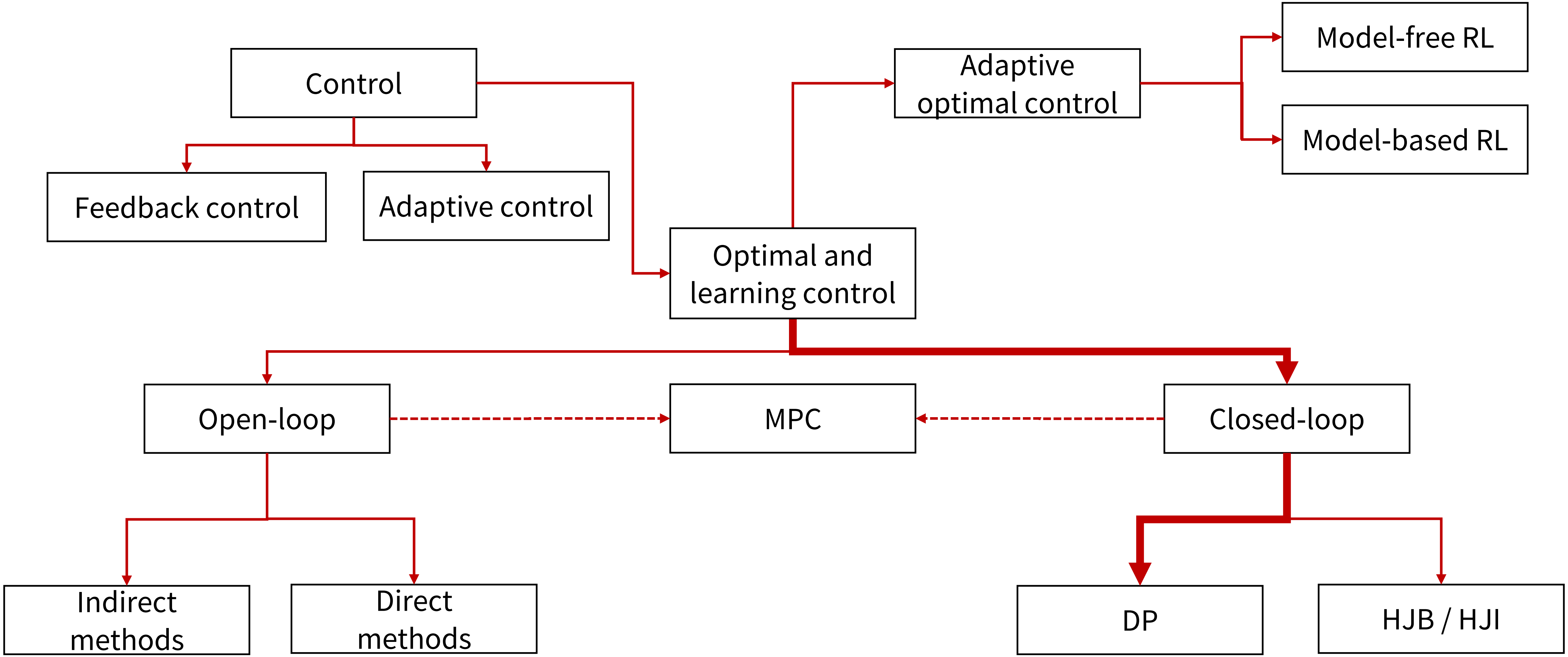
AA203
Optimal and Learning-based Control

Dynamic programming, discrete LQR

Stanford University

Autonomous Systems Lab

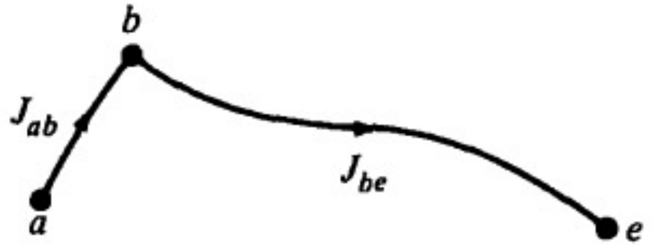
Roadmap


Control
Feedback control
Adaptive control
Optimal and learning control
Adaptive optimal control
Model-free RL
Model-based RL
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI

Principle of optimality

The key concept behind the dynamic programming approach is the principle of optimality

Suppose optimal path for a multi-stage decision-making problem is



- first decision yields segment $a - b$ with cost J_{ab}
- remaining decisions yield segments $b - e$ with cost J_{be}
- optimal cost is then $J_{ae}^* = J_{ab} + J_{be}$

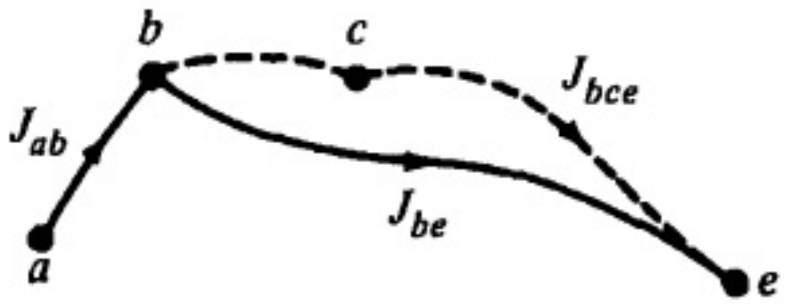
Principle of optimality

- Claim: If $a - b - e$ is optimal path from a to e, then $b - e$ is optimal path from b to e
- *Proof*: Suppose $b - c - e$ is the optimal path from b to e. Then

$$J_{bce} < J_{be}$$

and

$$J_{ab} + J_{bce} < J_{ab} + J_{be} = J_{ae}^*$$



Contradiction!

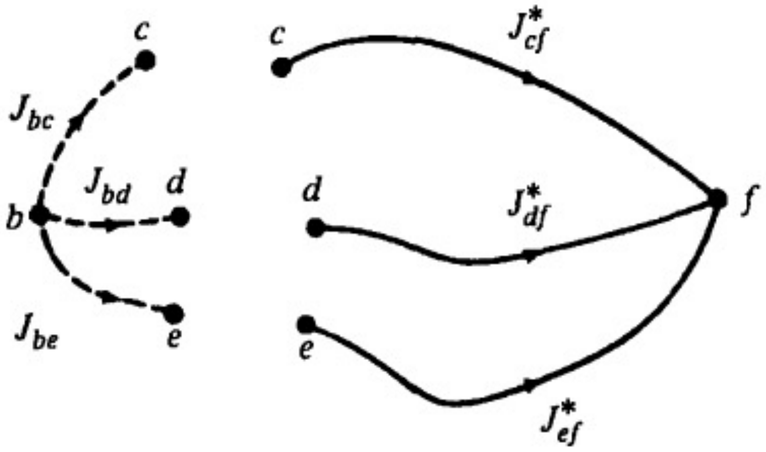
Principle of optimality

Principle of optimality (for discrete-time systems): Let $\pi^* := \{\pi_0^*, \pi_1^*, \ldots, \pi_{N-1}^*\}$ be an optimal policy. Assume state $\mathbf{x}_k$ is reachable. Consider the subproblem whereby we are at $\mathbf{x}_k$ at time k and we wish to minimize the cost-to-go from time k to time N. Then the truncated policy $\{\pi_k^*, \pi_{k+1}^*, \ldots, \pi_{N-1}^*\}$ is optimal for the subproblem

- tail policies optimal for tail subproblems
- notation: $\pi_k^*(\mathbf{x}_k) = \pi^*(\mathbf{x}_k, k)$

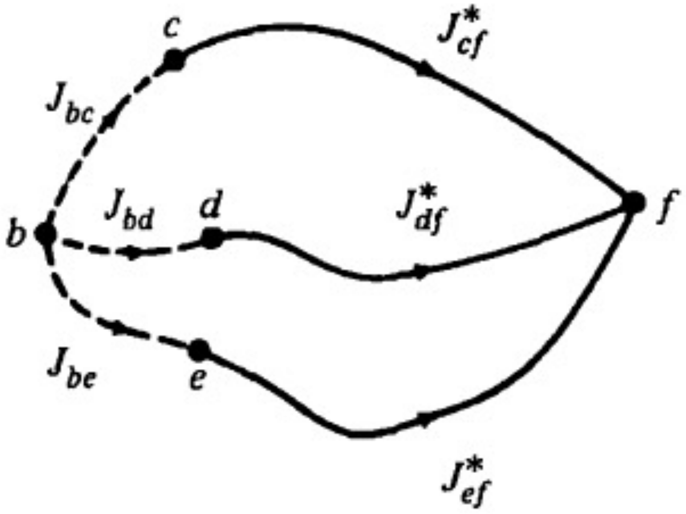
Applying the principle of optimality

Principle of optimality: if $b - c$ is the initial segment of the optimal path from b to f, then $c - f$ is the terminal segment of this path



Hence, the optimal trajectory is found by comparing:

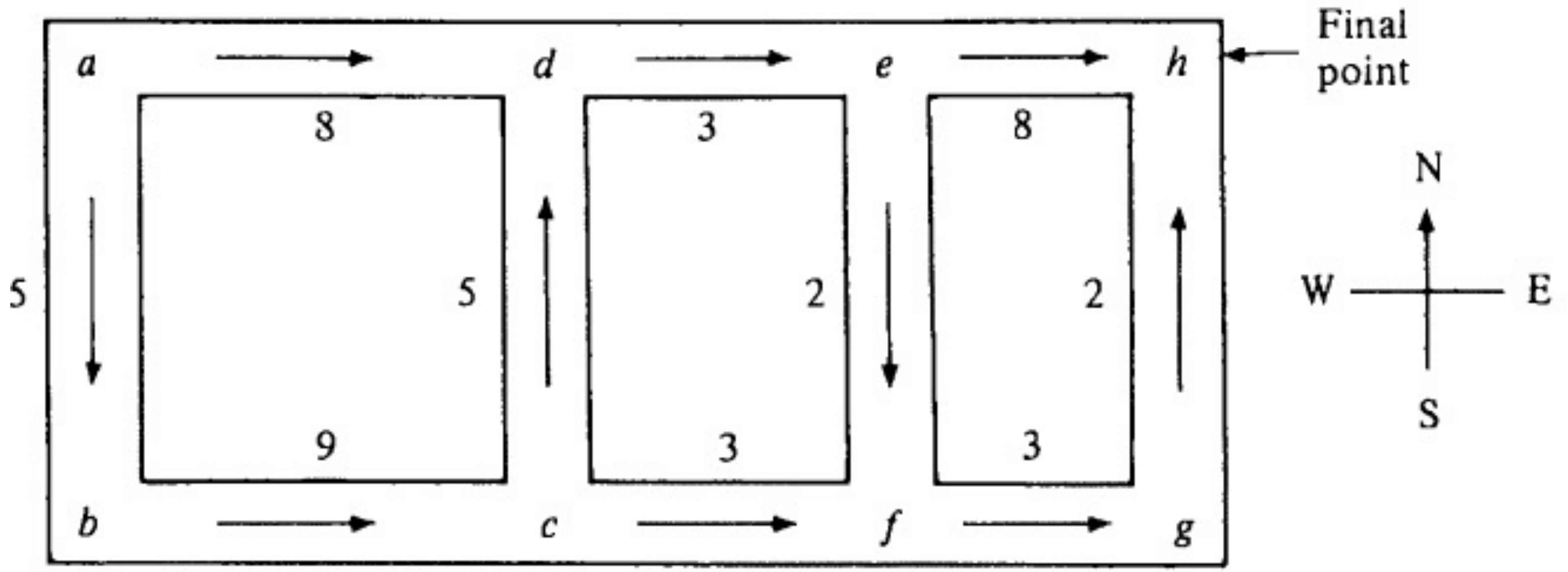
$$C_{bcf} = J_{bc} + J^*_{cf}$$
$$C_{bdf} = J_{bd} + J^*_{df}$$
$$C_{bef} = J_{be} + J^*_{ef}$$



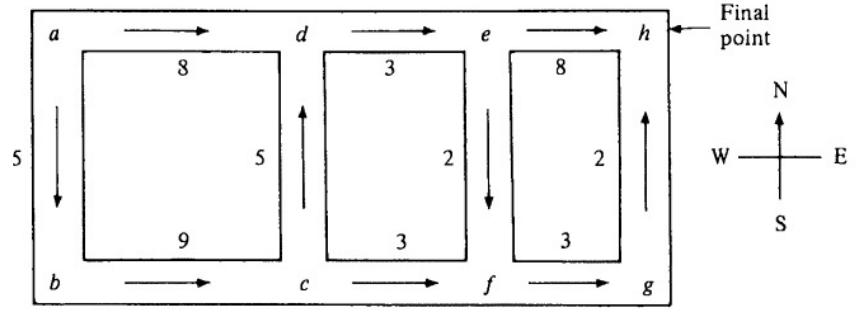
Applying the principle of optimality

- need only to compare the concatenations of immediate decisions and optimal decisions $\rightarrow$ significant decrease in computation / possibilities
- in practice: carry out this procedure backward in time

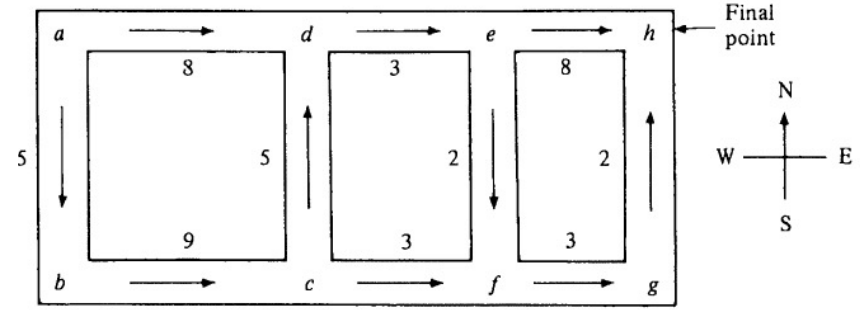
Example



Example



Example



Optimal cost: 18; Optimal path: $a \rightarrow\ d \rightarrow\ e \rightarrow\ f \rightarrow\ g \rightarrow\ h$

DP Algorithm

- Model: $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, k), \quad \mathbf{u}_k \in U(\mathbf{x}_k)$
- Cost: $J(\mathbf{x}_0) = h_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} g(\mathbf{x}_k, \pi_k(\mathbf{x}_k), k)$

DP Algorithm: For every initial state $\mathbf{x}_0$, the optimal cost $J^*(\mathbf{x}_0)$ is equal to $J_0(\mathbf{x}_0)$, given by the last step of the following algorithm, which proceeds backward in time from stage $N-1$ to stage 0:

$$J_N(\mathbf{x}_N) = h_N(\mathbf{x}_N)$$

$$J_k(\mathbf{x}_k) = \min_{\mathbf{u}_k \in U(\mathbf{x}_k)} g(\mathbf{x}_k, \mathbf{u}_k, k) + J_{k+1}\big(f(\mathbf{x}_k, \mathbf{u}_k, k)\big), \qquad k = 0, \dots, N-1$$

Furthermore, if $\mathbf{u}_k^* = \pi_k^*(\mathbf{x}_k)$ minimizes the right hand side of the above equation for each $\mathbf{x}_k$ and k, the policy $\{\pi_0^*, \pi_1^*, \dots, \pi_{N-1}^*\}$ is optimal

Comments

- discretization (from differential equations to difference equations)
- quantization (from continuous to discrete state variables / controls)
- global minimum
- constraints, in general, simplify the numerical procedure
- optimal control in closed-loop form
- curse of dimensionality

Discrete LQR

- Canonical application of dynamic programming for control
- One case where DP can be solved analytically (in general, DP algorithm must be performed numerically)

Discrete LQR: select control inputs to minimize

$$J_0(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_N^T Q_N \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + 2\mathbf{x}_k^T S_k \mathbf{u}_k\right)$$

subject to the dynamics

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k, \qquad k \in \{0, 1, \dots, N-1\}$$

assuming

$$Q_k = Q_k^T \succeq 0, \quad R_k = R_k^T \succ 0, \quad \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \succeq 0 \quad \forall k$$

Discrete LQR

Many important extensions, some of which we'll cover later in this class

- Tracking LQR: $\mathbf{x}_k, \mathbf{u}_k$ represent small deviations ("errors") from a nominal trajectory (possibly with nonlinear dynamics)
- Cost with linear terms, affine dynamics: can consider today's analysis with augmented dynamics

$$\mathbf{y}_{k+1} = \begin{bmatrix} \mathbf{x}_{k+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A_k & c_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \mathbf{u}_k = \tilde{A}\mathbf{y}_k + \tilde{B}\mathbf{u}_k$$

Discrete LQR – brute force

Rewrite the minimization of

$$J_0(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_N^T Q_N \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + 2\mathbf{x}_k^T S_k \mathbf{u}_k\right)$$

subject to dynamics

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k, \qquad k \in \{0, 1, \ldots, N-1\}$$

as…

Discrete LQR – brute force

$$\min_{\mathbf{x}_k,\mathbf{u}_k} \quad \frac{1}{2}\begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_0\\ \mathbf{x}_1\\ \mathbf{u}_1\\ \vdots\\ \mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\\ \mathbf{x}_N\end{bmatrix}^T \begin{bmatrix} Q_0 & S_0 & & & & & & \\ S_0^T & R_0 & & & & & & \\ & & Q_1 & S_1 & & & & \\ & & S_1^T & R_1 & & & & \\ & & & & \ddots & & & \\ & & & & & Q_{N-1} & S_{N-1} & \\ & & & & & S_{N-1}^T & R_{N-1} & \\ & & & & & & & Q_N \end{bmatrix} \begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_0\\ \mathbf{x}_1\\ \mathbf{u}_1\\ \vdots\\ \mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\\ \mathbf{x}_N\end{bmatrix}$$

$$\text{s.t.} \quad \begin{bmatrix} -I & & & & & & \\ A_0 & B_0 & -I & & & & \\ & & A_1 & B_1 & -I & & \\ & & & & \ddots & & \\ & & & & A_{N-1} & B_{N-1} & -I \end{bmatrix} \begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_0\\ \mathbf{x}_1\\ \mathbf{u}_1\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\\ \mathbf{x}_N\end{bmatrix} + \begin{bmatrix}\mathbf{x}_0\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix} = \mathbf{0}$$

Discrete LQR – brute force

Defining suitable notation, this is

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\mathbf{z}^T W \mathbf{z} \\ \text{s.t.} \quad & C\mathbf{z} + \mathbf{d} = \mathbf{0} \end{aligned}$$

with solution from applying NOC (also SOC in this case, due to problem convexity):

$$\begin{bmatrix} \mathbf{z}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} W & C^T \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ -\mathbf{d} \end{bmatrix}$$

Discrete LQR – dynamic programming

First step:

$$J_N^*(\mathbf{x}_N) = \frac{1}{2} x_N^T Q_N x_N = \frac{1}{2} x_N^T P_N x_N$$

Going backward:

$$\begin{aligned}
J_{N-1}^*(\mathbf{x}_{N-1}) &= \min_{\mathbf{u}_{N-1}} \frac{1}{2} \left(\begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix}^T \begin{bmatrix} Q_{N-1} & S_{N-1} \\ S_{N-1}^T & R_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix} + \mathbf{x}_N^T P_N \mathbf{x}_N \right) \\
&= \min_{\mathbf{u}_{N-1}} \frac{1}{2} \left(\begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix}^T \begin{bmatrix} Q_{N-1} & S_{N-1} \\ S_{N-1}^T & R_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix} + \right. \\
&\qquad \left. (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1})^T P_N (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}) \right)
\end{aligned}$$

Discrete LQR – dynamic programming

Unconstrained NOC:

$$\begin{aligned}\nabla_{u_{N-1}} J_{N-1}(\mathbf{x}_{N-1}) &= R_{N-1}\mathbf{u}_{N-1} + S_{N-1}^T\mathbf{x}_{N-1} + \\ &\qquad B_{N-1}^T P_N (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}) = \mathbf{0} \\ \implies \mathbf{u}_{N-1}^* &= -(R_{N-1} + B_{N-1}^T P_N B_{N-1})^{-1}(B_{N-1}^T P_N A_{N-1} + S_{N-1}^T)\mathbf{x}_{N-1} \\ &:= F_{N-1} x_{N-1}\end{aligned}$$

Note also that:

$$\nabla_{u_{N-1}}^2 J_{N-1}(\mathbf{x}_{N-1}) = R_{N-1} + B_{N-1}^T P_N B_{N-1} \succ 0$$

Discrete LQR – dynamic programming

Plugging in the optimal policy:

$$\begin{aligned} J^*_{N-1}(\mathbf{x}_{N-1}) &= \frac{1}{2}\mathbf{x}^T_{N-1}\big(Q_{N-1} + A^T_{N-1}P_N A_{N-1} - \\ &\qquad (A^T_{N-1}P_N B_{N-1} + S_{N-1})(R_{N-1} + B^T_{N-1}P_N B_{N-1})^{-1}(B^T_{N-1}P_N A_{N-1} + S^T_{N-1})\big)\,\mathbf{x}_{N-1} \\ &:= \frac{1}{2}\mathbf{x}^T_{N-1}P_{N-1}\mathbf{x}_{N-1} \end{aligned}$$

Algebraic details aside:

- Cost-to-go (equivalently, “value function”) is a quadratic function of the state at each step
- Optimal policy is a time-varying linear feedback policy

Discrete LQR – dynamic programming

Proceeding by induction, we derive the Riccati recursion:

1. $P_N = Q_N$
2. $F_k = -(R_k + B_k^T P_{k+1} B_k)^{-1}(B_k^T P_{k+1} A_k + S_k^T)$
3. $P_k = Q_k + A_k^T P_{k+1} A_k -$
$$(A_k^T P_{k+1} B_k + S_k)(R_k + B_k^T P_{k+1} B_k)^{-1}(B_k^T P_{k+1} A_k + S_k^T)$$
4. $\pi_k^*(\mathbf{x}_k) = F_k \mathbf{x}_k$
5. $J_k^*(\mathbf{x}_k) = \dfrac{1}{2}\mathbf{x}_k^T P_k \mathbf{x}_k$

Compute policy backwards in time, apply policy forward in time.

Next time

- Stochastic DP

$$V^*(x) = \max_u \left(R(x,u) + \gamma \sum_{x' \in \mathcal{X}} T(x'|x,u)\, V^*(x') \right)$$