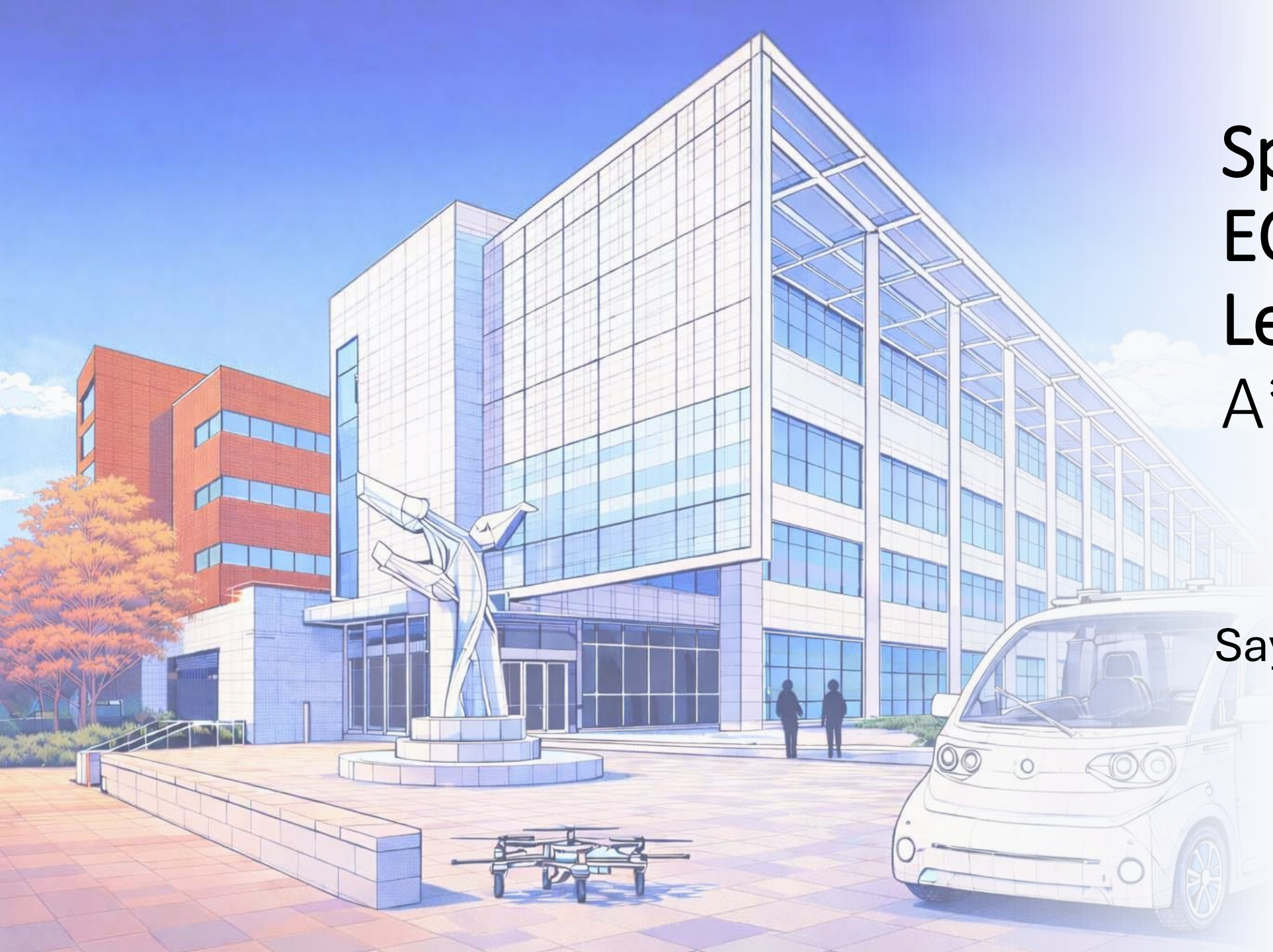


Spring 26
ECE484
Lecture 15
A*->LQR

Sayan Mitra

Lecture Roadmap

A* Search

Where we've been

Discrete graph search
f(n) = g(n) + h(n)
h(n): cost-to-go guess
Optimal + complete
Routing on road networks
Robot planning on 2D/3D

Dynamic Programming

The bridge

Bellman's principle
Value function V*(x)
optimal cost-to-go
Continuous spaces
u*(x): best action

LQR

Where we're going

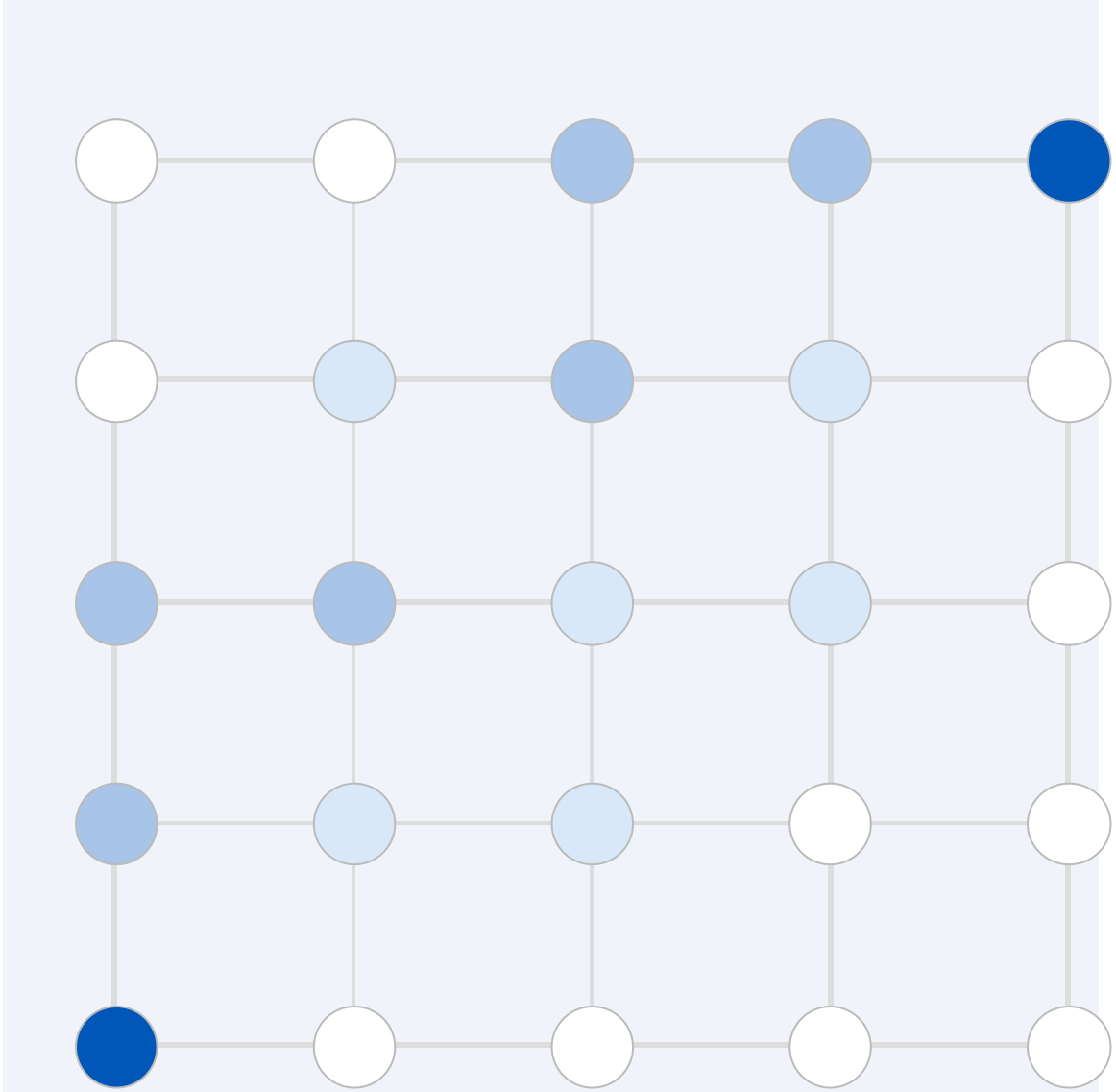
LTI $x_{t+1} = Ax_t + Bu_t$
Quadratic cost
Analytic optimal via Riccati
Drone tracks a reference trajectory

Review: A* Search

Key idea: $f(n) = g(n) + h(n)$
Expand nodes in order of estimated total cost.
Admissibility $h(n) \le h^*(n)$ guarantees optimality.

- Graph $G = (V, E)$, start s, goal g
- Edge cost $w(u, v) \ge 0$
- Heuristic $h(n)$: **estimated cost to go**

Guarantees
- Complete — finds a solution if one exists
- Optimal — minimum-cost path
- Efficient — h prunes the frontier vs. Dijkstra



Dynamic Programming for Planning

Example: a drone must fly A → B minimizing fuel. What is the cheapest path from any position?

Setup

- State $x_t \in \mathbb{R}^n$ (position, velocity, …)
- Control $u_t \in \mathbb{R}^m$ (thrust, torques, …)
- Dynamics $x_{t+1} = f(x_t, u_t)$
- Stage cost $\ell(x_t, u_t)$, e.g. $\|u\|^2$
- Terminal cost $\ell f(x_T)$

Goal: Find $u^*(x)$ minimizing total cost $J = \Sigma_t \ell(x_t, u_t) + \ell f(x_T)$

Value function: $V^*(x)$ = *minimum cost achievable* from state x under the optimal policy.
Same as the optimal heuristic cost-to-go h^* in A*

Key insight: If we know $V^*(x')$ for all successors x', the optimal action at x follows from a single one-step minimization — no full graph search needed.

Bellman's Principle (1957)

"Whatever the initial decision, the remaining decisions must constitute an optimal policy with regard to the resulting state."

The Bellman equation:

$$V^*(x) = \min_u [\, \ell(x, u) + V^*(f(x, u)) \,]$$

cost-to-go = stage cost + future cost

Connection to A*: $V^*(x)$ is the perfect heuristic $h^*(x)$. A* estimates it — DP computes it exactly, for every x.

Value Iteration (preview): Initialize $V(x) = 0$. Iterate backwards:
$V(x) \leftarrow \min_u [\, \ell(x,u) + V(f(x,u)) \,]$ until convergence.

Richard Ernest Bellman

https://en.wikipedia.org/wiki/Richard_E._Bellman

Value Iteration: Iterative computation of V

Initialize $V_0(x) = 0$ for all x

For k = 0, 1, 2, …

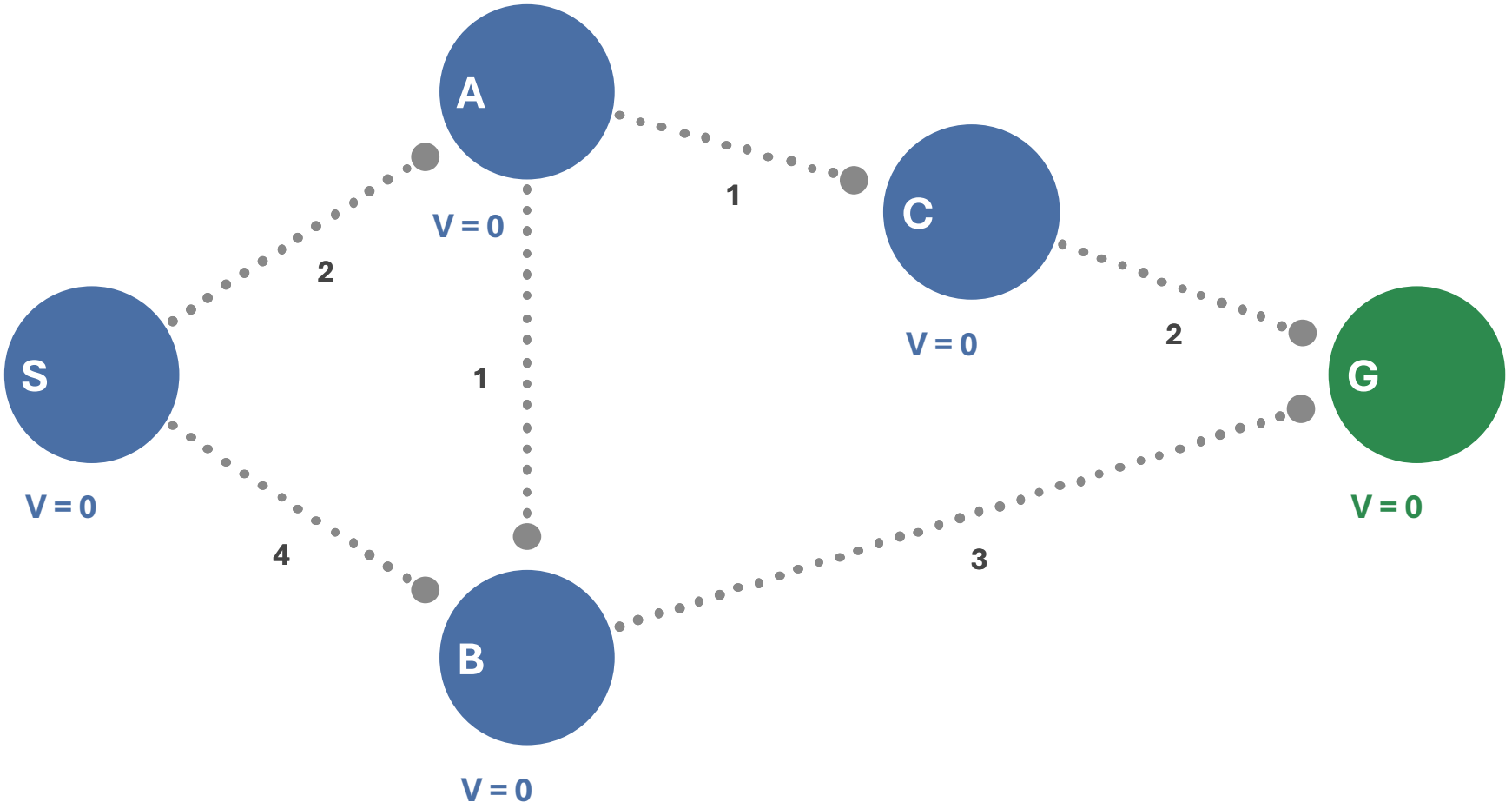
$V_{k+1}(x) = \min_u [\ \ell(x, u) + V_k(f(x, u))\ ]$ for all x

Stop when $\|V_{k+1} - V_k\|_\infty < \varepsilon$

Extract policy: $u^*(x) = \text{argmin\_u} [\ \ell(x, u) + V^*(f(x, u))\ ]$

Each iteration propagates cost information one step *further back in time*.
After k iterations, $V_k(x)$ is the optimal cost for a k-step horizon. As $k \to \infty$ it converges to the infinite-horizon optimum.

Value Iteration — V_0 — initialize all to 0

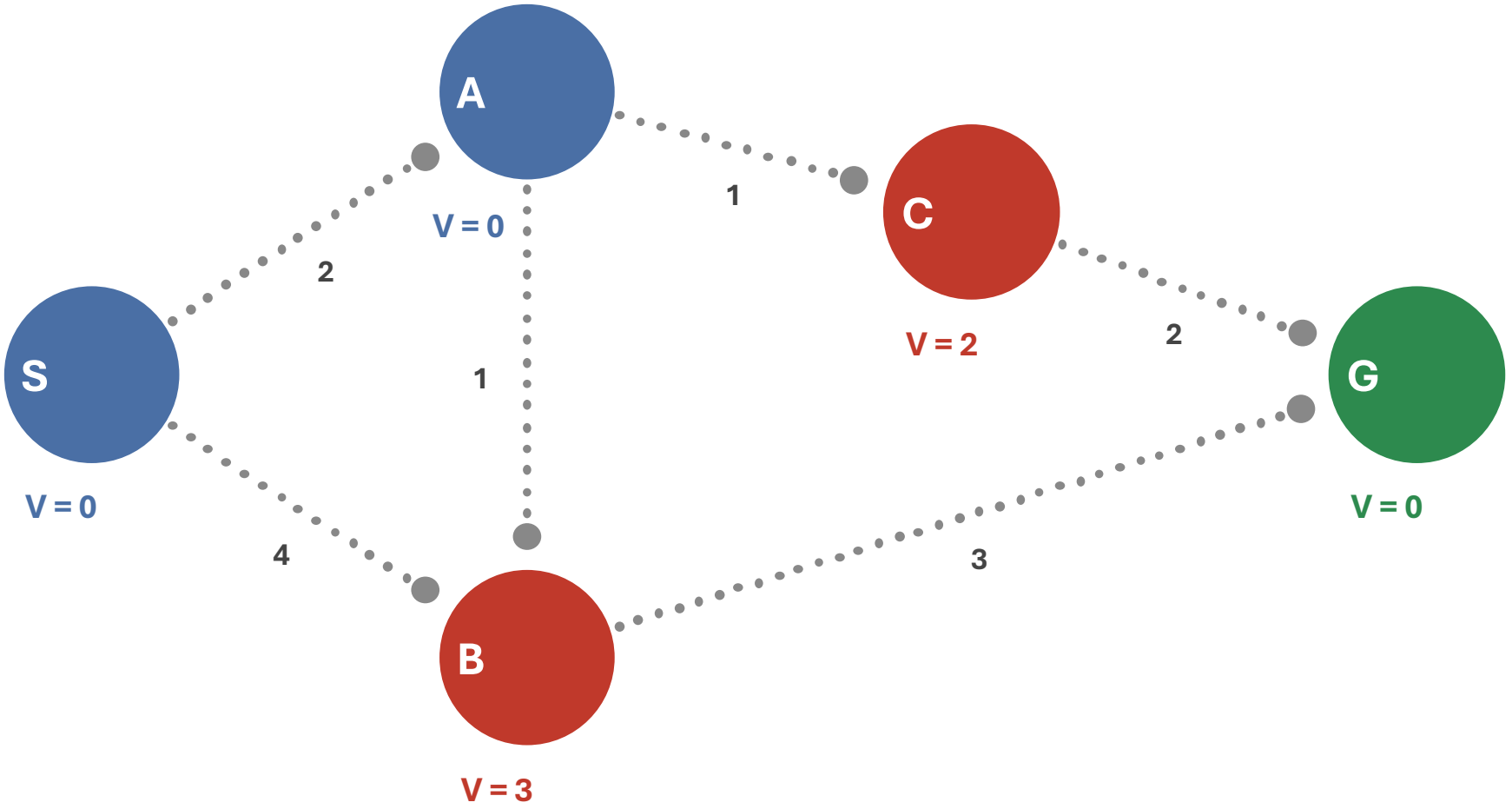


Initialize all V(x) = 0.

G is the goal; V(G) = 0 always (absorbing state).

No cost info has propagated yet.

Value Iteration — V_1 — k = 1



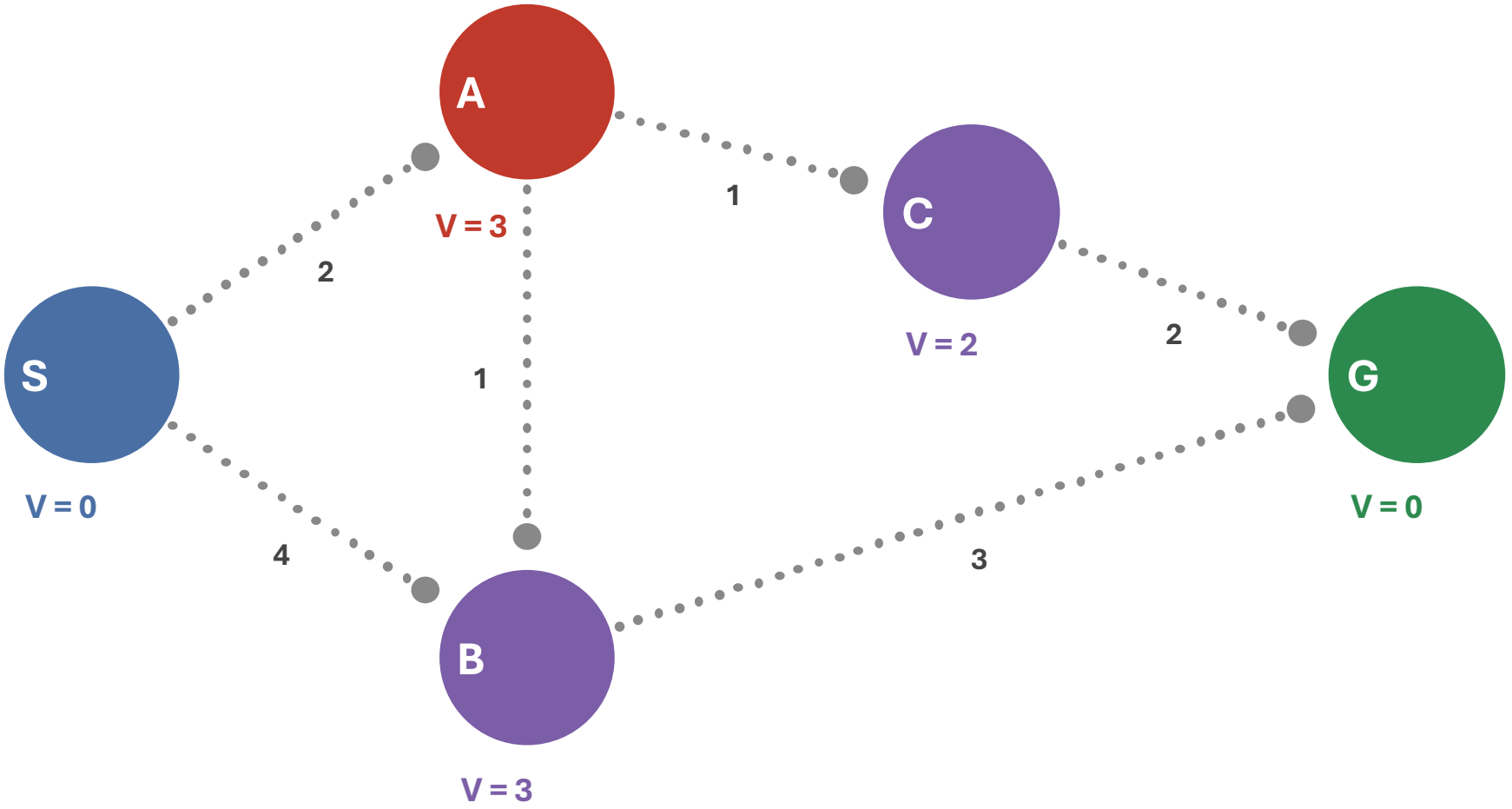
$V_1(C) = \ell(C \to G) + V_0(G)$
$= 2 + 0 = 2$

$V_1(B) = \ell(B \to G) + V_0(G)$
$= 3 + 0 = 3$

S, A: no 1-step path to G.

Goal (V=0 fixed) | Updated this step | Converged | Not yet reached

Value Iteration — V_2 — k = 2

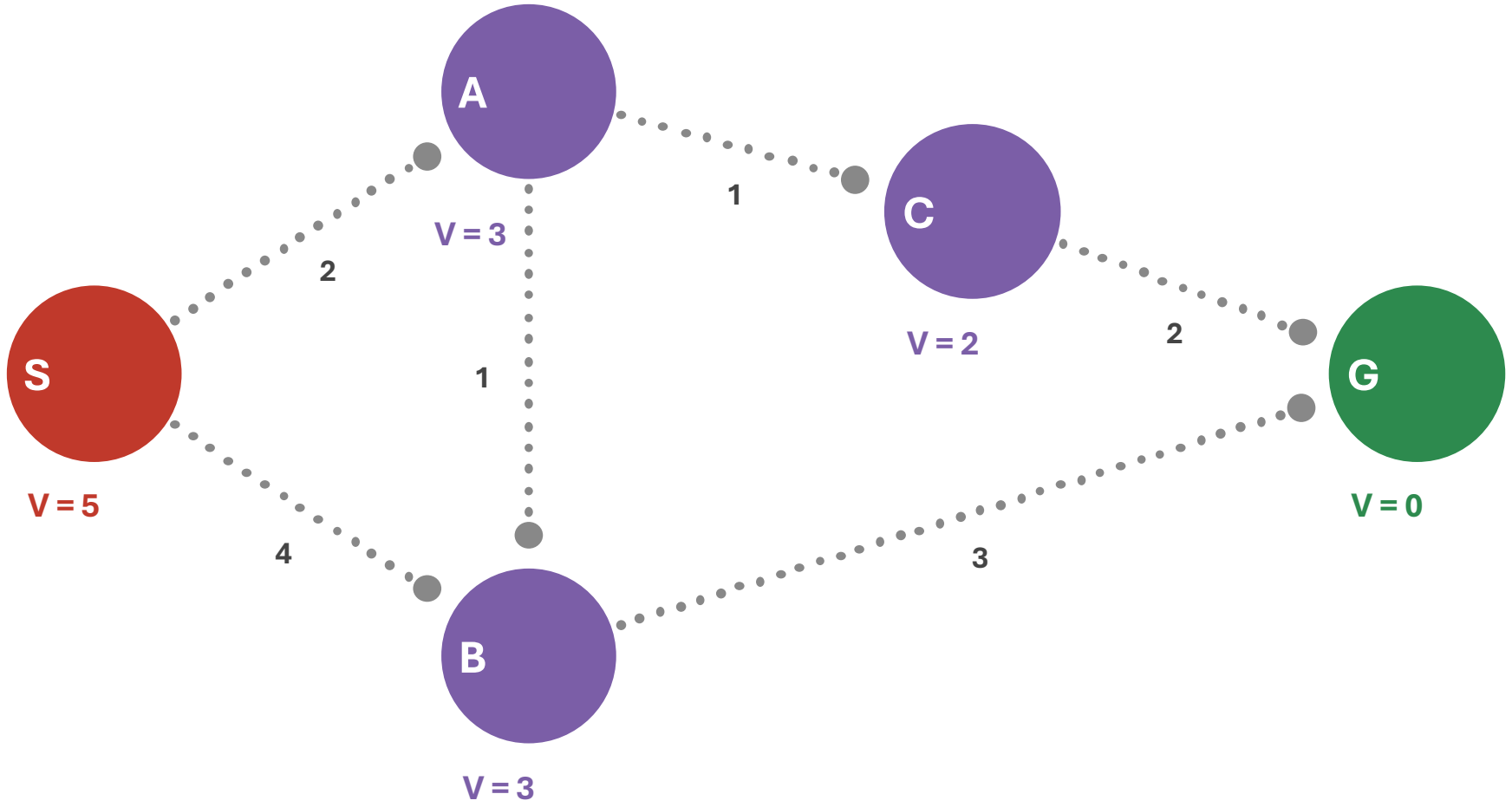


$V_2(A) = \min($
$\ell(A \to C) + V_1(C),$
$\ell(A \to B) + V_1(B)$
$)$
$= \min(1+2, 1+3)$
$= 3$

B, C converged.

Goal (V=0 fixed)
Updated this step
Converged
Not yet reached

Value Iteration — $V_3 = V^*$ — converged



$V_3(S) = \min($
$\ell(S \to A) + V_2(A),$
$\ell(S \to B) + V_2(B)$
$)$
$= \min(2+3, 4+3)$
$= 5$

Converged ✓
Optimal path:
S→A→C→G
(total cost 5)

Goal (V=0 fixed)
Updated this step
Converged
Not yet reached

Value Iteration: Convergence

States X, actions U, dynamics $f : X \times U \to X$, cost $\ell(x, u) \geq 0$, **discount factor $\gamma \in (0, 1)$**

The Bellman operator $T : (X \to \mathbb{R}) \to (X \to \mathbb{R})$

$T(V(x)) = \min_u [\, \ell(x, u) + \gamma \cdot V(f(x, u)) \,]$

T maps a value function V_k to a "one-step improved" value function V_{k+1}.

To discuss convergence/stability of the Bellman operator we need a metric on V

Sup-norm (uniform norm): $\|V\|_\infty = \sup_x |V(x)|$ measures worst-case across states

Theorem (Contraction & Convergence): T is a contraction mapping in $\|\cdot\|_\infty$ with γ:
$\|TV - TV'\|_\infty \leq \gamma \|V - V'\|_\infty$ for all V, V'

Corollary: V* is the unique fixed point: $TV^* = V^*$
Value iteration converges geometrically: $\|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty$

Proof: The Contraction Step

Goal: show $\|TV - TV'\|_\infty \le \gamma \|V - V'\|_\infty$

Fix any state $x \in X$. We need to bound $|(TV)(x) - (TV')(x)|$.
Expand the definitions:

$(TV)(x) = \min_u [\, \ell(x,u) + \gamma V(f(x,u)) \,]$
$(TV')(x) = \min_u [\, \ell(x,u) + \gamma V'(f(x,u)) \,]$

Key lemma: $|\min_u g(u) - \min_u h(u)| \le \max_u |g(u) - h(u)|$

Apply the lemma with $g(u) = \ell + \gamma V(x')$ and $h(u) = \ell + \gamma V'(x')$, $x' = f(x,u)$:

$|(TV)(x) - (TV')(x)|$

$\le \max_u |\, \gamma V(f(x,u)) - \gamma V'(f(x,u)) \,|$ [Key Lemma]
$= \gamma \cdot \max_u |V(f(x,u)) - V'(f(x,u))|$ [γ is a constant]
$\le \gamma \cdot \sup_x |V(x) - V'(x)|$ [$f(x,u)$ is a state, sup absorbs $\max_u$]
$= \gamma \|V - V'\|_\infty$ [Def on ∞ norm]

Taking sup over all x: $\|TV - TV'\|_\infty \le \gamma \|V - V'\|_\infty$ ■

Value Iteration: Practical Takeaways

Choosing the discount factor γ

γ close to 1 → slow contraction, more iterations, long-horizon planning

γ close to 0 → fast contraction, few iterations, short-sighted policy

$\gamma = 1$ → no contraction guarantee; use only with finite-horizon formulation

Stopping criterion: $\|V_{k+1} - V_k\|_\infty < \varepsilon$. The error bound: $\|V_k - V^*\|_\infty \leq \varepsilon \gamma / (1 - \gamma)$

Small γ makes this bound tight with few iterations.

Finite-horizon case: With T steps and $\gamma = 1$: run exactly T backward sweeps.

$V_k(x)$ is optimal for a $(T - k)$-step problem — exact in T steps, no convergence needed.

Cost of one iteration: Each update $V_{k+1}(x) = \min_u [\ell + V_k(f(x,u))]$ is $O(|U|)$ per state.

Total per iteration: $O(|X| \cdot |U|)$. For a d-dimensional grid with n cells/dim: $O(n^d \cdot |U|)$.

This is the curse of dimensionality — motivating LQR's $O(n^3)$ analytic alternative.

From DP to LQR: Problem Setup

Value iteration works in principle but requires discretizing the state space — which we saw is infeasible for high-dimensional systems.

Can we solve the Bellman equation analytically?

Yes — if we add structure:

- Linear dynamics
- Quadratic cost

This gives the Linear Quadratic Regulator (LQR).

Dynamics (discrete-time LTI): $x_{t+1} = A\, x_t + B\, u_t$

$x_t \in \mathbb{R}^n$ state, $u_t \in \mathbb{R}^m$ control

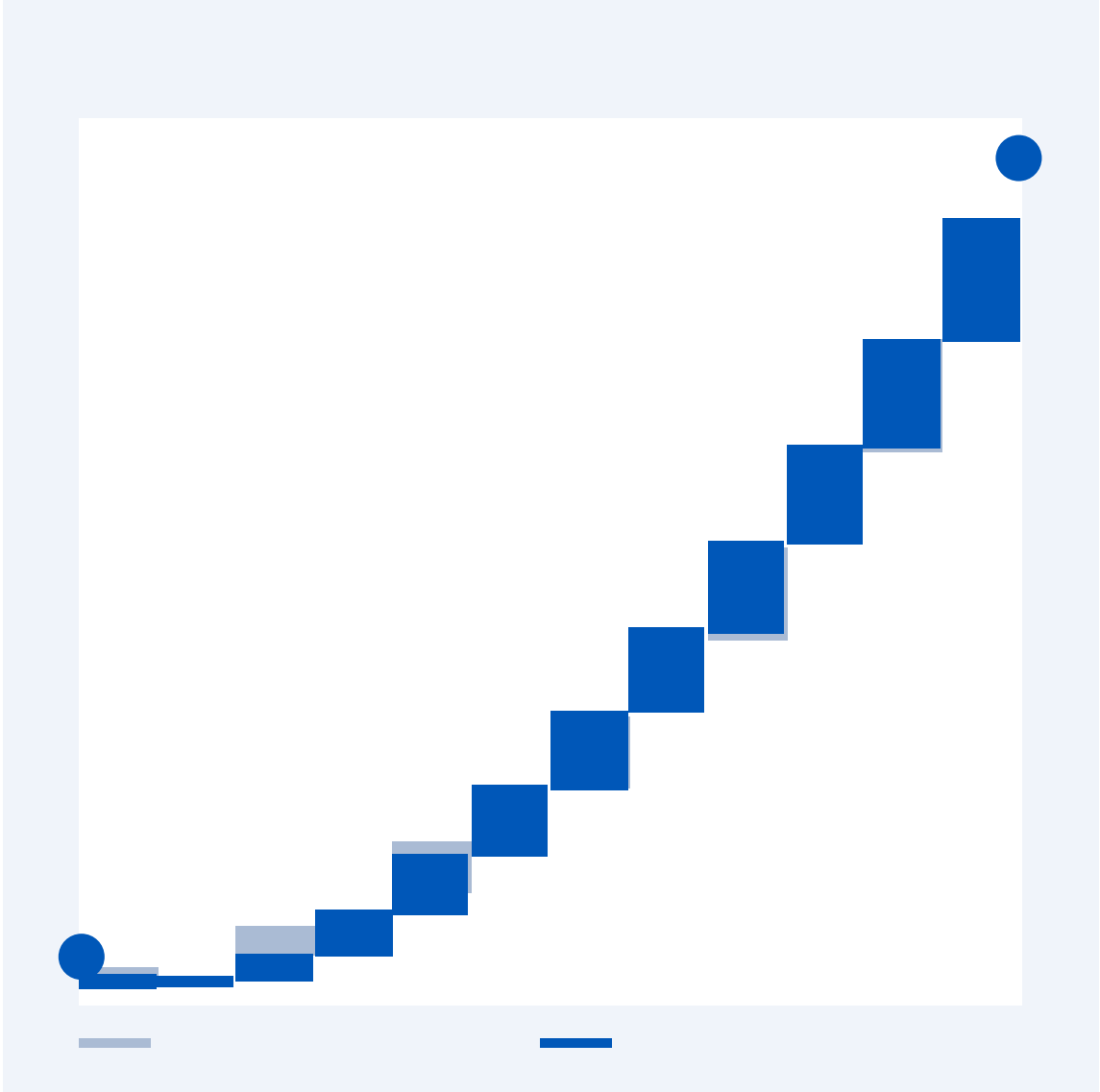
$A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ known system matrices

Example: Drone Trajectory Tracking with LQR

Given: reference trajectory $\bar{x}_t, \bar{u}_t$ from start to goal, design a feedback controller that keeps the drone on the reference despite disturbances.

Step 1 — Linearize around the reference
Let $\delta x_t = x_t - \bar{x}_t$, $\delta u_t = u_t - \bar{u}_t$. The error dynamics are: $\delta x_{t+1} = A\,\delta x_t + B\,\delta u_t$
where $A = \partial f/\partial x|_{\bar{x},\bar{u}}$ and $B = \partial f/\partial u|_{\bar{x},\bar{u}}$
(Jacobians along the reference).

Step 2 — Solve the DARE for P, compute K
Step 3 — Apply feedback
$\delta u_t = -K\,\delta x_t \rightarrow u_t = \bar{u}_t - K(x_t - \bar{x}_t)$
$\delta x = x_t - \bar{x}_t$
LQR applies $u = \bar{u}_t - K\,\delta x_t$



LQR: Cost Structure

Quadratic cost (finite horizon, N steps)

$$J = x_n^T Q_f x_n + \Sigma_{t=0}^{N-1} (x_t^T Q x_t + u_t^T R u_t)$$

$Q \in \mathbb{R}^{n \times n}$, $Q \succcurlyeq 0$ state penalty
$R \in \mathbb{R}^{m \times m}$, $R \succ 0$ control effort penalty
$Q_f \succcurlyeq 0$ terminal state penalty

Why quadratic cost?
- Penalizes deviation from zero symmetrically
- $R \succ 0$ ensures we cannot use unbounded control
- Q, R are design parameters: trading off accuracy vs. effort
- Drone: Q = diag(qx, qy, qz, ...) penalises position error; R penalises thrust

Why linear dynamics?
- LTI systems are closed under the quadratic ansatz $V^*(x) = x^T P x$
- Substituting into Bellman yields an algebraic equation for P — no grid needed

The Discrete Algebraic Riccati Equation (DARE)

Derivation sketch

Ansatz/template: suppose $V^*_t(x) = x^T P_t x$ for some $P_t \succcurlyeq 0$.

Substitute into the Bellman equation:

$V_{k+1}(x) = \min_u [\ell(x, u) + V_k(f(x, u))]$

$x^T P_t x = \min_u [\, x^T Q x + u^T R u + (Ax+Bu)^T P_{t+1} (Ax+Bu) \,]$

Minimize over u — the objective is quadratic in u with $R \succ 0$, so set the gradient to zero:

$\partial/\partial u [\, u^T R u + (Ax+Bu)^T P_{t+1}(Ax+Bu) \,] = 0$

$2Ru + 2B^T P_{t+1} (Ax + Bu) = 0$

$u^* = -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A\, x$

Substitute u^* back. Collecting $x^T(\cdot)x$ terms gives a recursion for P_t — the discrete-time Riccati recursion.

The LQR Solution

Optimal gain matrix: $K = (R + B^T P B)^{-1} B^T P A$

Optimal control law: $u^*_t = -K x_t$

K is computed once offline — applying the policy is just a matrix-vector multiply each timestep.

Interpretation: $K \in \mathbb{R}^{m \times n}$ maps the full state to control input.

- Larger Q / R → larger K → more aggressive correction
- Larger R / Q → smaller K → more conservative, lower effort

Cost achieved: $J^* = x_0^T P x_0$ (P encodes the cost-to-go as a quadratic form over state space)

Stability of the LQR Closed-Loop System

Closed-loop dynamics: substituting $u^* = -Kx$ into $x_{t+1} = Ax_t + Bu_t$:
$x_{t+1} = (A - BK)\, x_t \quad =: \quad A_{cl} x_t$

Stable iff $|\lambda_i(A - BK)| < 1$ for all i.

Theorem: If (A, B) is stabilizable and (A, Q^½) is detectable, LQR renders $A_{cl} = A - BK$ asymptotically stable.

Proof via Lyapunov: use $V(x) = x^T P x$.
$V(x_{t+1}) - V(x_t) = -x_t^T (Q + K^T R K) x_t \leq 0$

Detectability prevents Q from hiding unstable modes → V is a strict Lyapunov function.

Key point: solving the DARE is $O(n^3)$ — a single matrix equation, not a grid sweep. This is why LQR scales where value iteration cannot.

LQR Stability: Key Conditions

Key conditions for the stability theorem

Stabilisability of (A, B):
All unstable modes of A are reachable by B — we can actually control them.

Detectability of (A, Q^½):
All unstable modes are penalised by Q — we are not ignoring them in the cost.

Result:
$|\lambda_i(A - BK)| < 1$ for all i → asymptotically stable

LQR guarantees this holds automatically under stabilisability + detectability.

All closed-loop poles $\lambda(A - BK)$ lie strictly inside the unit disk.

Limitations of LQR

1 Nonlinear dynamics — LQR requires $f(x,u) = Ax+Bu$. Real systems are nonlinear; linearisation works locally but the guarantee breaks down far from the operating point.

2 No constraint handling — The optimal $u^* = -Kx$ can be arbitrarily large. Actuators saturate, states must stay within safe bounds — LQR ignores both.

3 No obstacle avoidance — The cost $J = \Sigma\, x^TQx + u^TRu$ has no notion of forbidden regions. LQR will drive through an obstacle on the optimal unconstrained path.

4 Global linearisation error — With gain scheduling there is no certificate the switched system is stable overall; transitions can be destabilising.

5 Stochastic / partial observability — If only noisy outputs $y = Cx+v$ are available, LQR must be paired with a Kalman filter (LQG).

When to stop using LQR: large deviations from linearisation · hard actuator/state constraints · obstacle-rich environments · strong nonlinearities