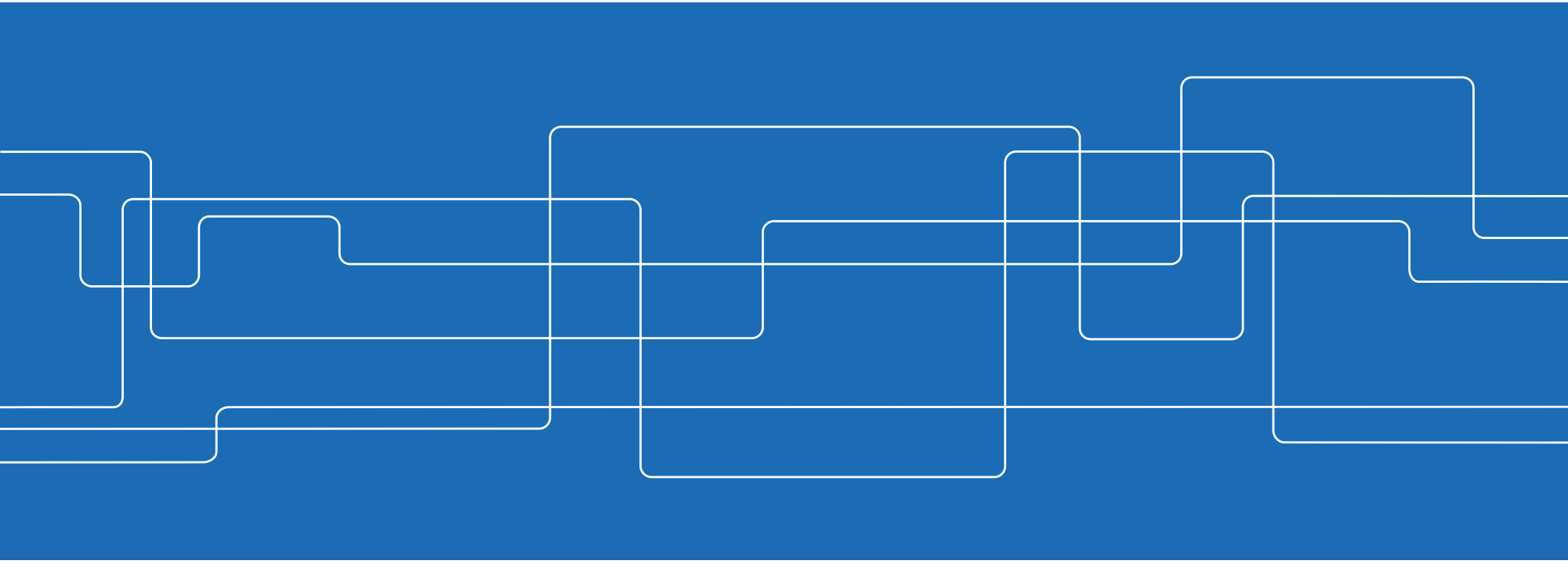
Lecture 4: Dynamic programming and LQR

Mikael Johansson
KTH - Royal Institute of Technology



Outline

- Linear quadratic regulation
- Finite-horizon open-loop solution via least-squares
- A first attempt with receding-horizon control
- Finite-horizon feedback solution via dynamic programming

Finite-horizon linear-quadratic regulation

Given the *linear* system

$$x_{t+1} = Ax_t + Bu_t,$$

Find control sequence

$$U_N = \{u_0, u_1, \cdots, u_{N-1}\}$$

that minimizes the *quadratic* cost function

$$J(U_N) = \sum_{k=0}^{N-1} (x_k^T Q_1 x_k + u_k^T Q_2 u_k) + x_N^T Q_f x_N := J'(U_N) + x_0^T Q_1 x_0$$

for given state cost, control cost, and final cost matrices

$$Q_1 \succeq 0, \quad Q_2 \succ 0, \quad Q_f \succeq 0$$

N is called the *horizon* of the problem. Note the final state cost.

Finite-horizon LQR on standard form

On our standard form for finite-time optimal control problems

$$\begin{array}{ll} \underset{\{u_0,\ldots,u_{N-1}\},\{x_0,\ldots,x_N\}}{\text{minimize}} & \sum_{t=0}^{N-1} x_t^T Q_1 x_t + u_t^T Q_2 u_t + x_N^T Q_f x_N \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t \qquad t = 0, \ldots, N-1 \\ & x_0 = x_{\text{init}} \end{array}$$

Since $Q_1 \succeq 0$, $Q_2 \succ 0$ and $Q_f \succeq 0$, objective is convex and quadratic in

$$z = (u_0, \ldots, u_{N-1}, x_0, x_1, \ldots, x_N)$$

In addition, constraints are linear, so problem is convex QP.

Can simplify problem by eliminating x using prediction equations.

Finite-horizon LQR via least squares

$X_N = \begin{pmatrix} x_1, \dots, x_N \end{pmatrix}$ is a linear function of x_0 and $U_N = \begin{pmatrix} u_0, \dots, u_{N-1} \end{pmatrix}$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & 0 & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} A \\ \vdots \\ A^N \end{bmatrix} x_0$$

Can express as

$$X_N = \mathcal{H}_N U_N + \hbar_N$$

where $\mathcal{H}_N \in \mathbb{R}^{Nn \times Nm}$, $\hbar_N \in \mathbb{R}^{Nn \times n}$

Finite-horizon LQR via least squares

Can express J' as a quadratic function of U

$$J'(U_N) = X_N^T \underbrace{\begin{bmatrix} Q_1 & 0 & \dots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & Q_1 & 0 \\ 0 & \cdots & 0 & Q_f \end{bmatrix}}_{\bar{Q}_1} X_N + U_N^T \underbrace{\begin{bmatrix} Q_2 & 0 & \dots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & Q_2 & 0 \\ 0 & \cdots & 0 & Q_2 \end{bmatrix}}_{\bar{Q}_2} U_N =$$

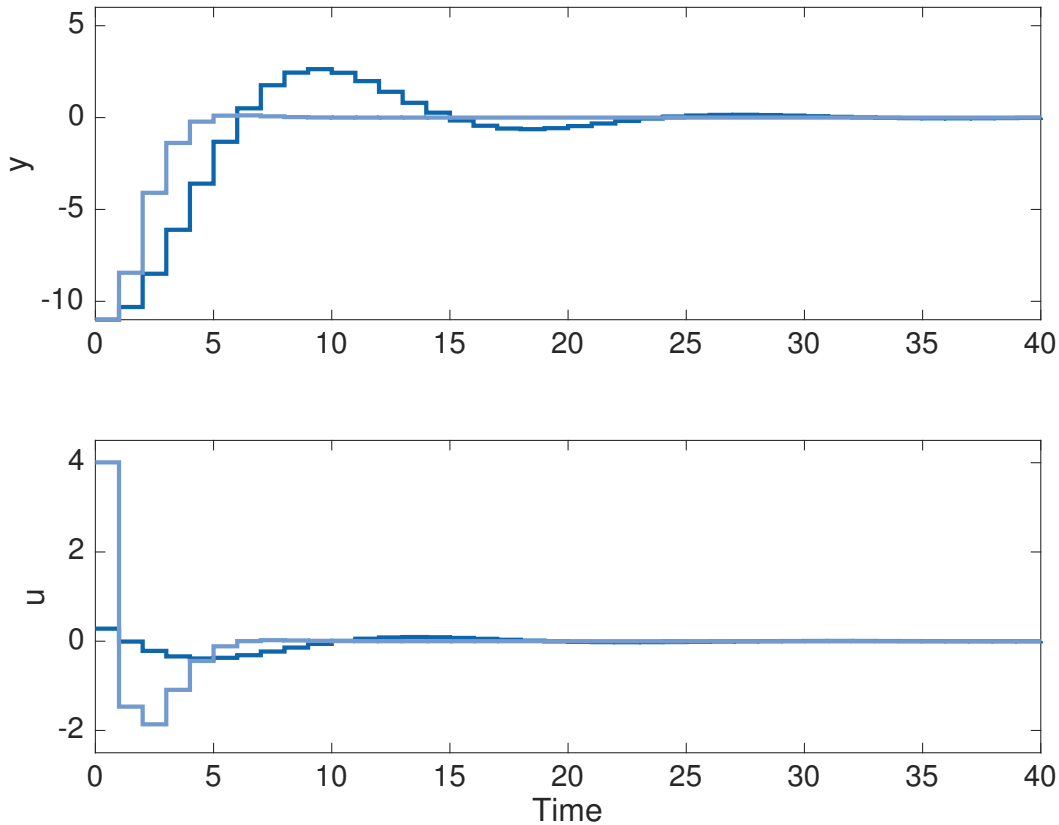
$$= (\mathcal{H}_N U_N + \hbar_N)^T \bar{Q}_1 (\mathcal{H}_N u_N + \hbar_N) + U_N^T \bar{Q}_2 U_N =$$

$$= U_N^T \underbrace{(\mathcal{H}_N^T \bar{Q}_1 \mathcal{H}_N + \bar{Q}_2)}_{P_{LQ}} U_N + 2U_N^T \underbrace{\mathcal{H}_N^T \bar{Q}_1 \hbar_N}_{q_{LQ}} + \underbrace{\hbar_N^T \bar{Q}_1 \hbar_N}_{r_{LQ}}$$

So optimal solution is $U_N^\star = -P_{LQ}^{-1} q_{LQ}$ (a linear function of x_0)

Finite-horizon LQR of mechanical system from Lecture 3

Finite-horizon LQR with $Q_1 = Q_f = 1$, and $Q_2 = \rho I$ with $\rho = 1, 100$



Open-loop control - can we derive LQ-optimal feedback solution?

Receding-horizon LQR

Can also introduce feedback is via receding horizon principle:

In every sample t,

- measure $x(t) = x_t$
- solve finite-horizon LQR problem

$$\begin{array}{ll} \text{minimize} & \sum_{k=0}^{N-1} (x_{t+k}^T Q_1 x_{t+k} + u_{t+k}^T Q_2 u_{t+k}) + x_{t+N}^T Q_f x_{t+N} \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t, \end{array}$$

  by forming P_{LQ} and $q_{LQ}(x_t)$ and computing $U_t = -P_{LQ}^{-1} q_{LQ}$
- apply first component of U_t, wait until next sample and repeat

Potential problem: may be unstable, far from long-term optimal.

Potential instability of receding-horizon LQR

Consider a discrete-time linear system with

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and the receding-horizon problem given by

$$Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q_2 = 1, \quad Q_f = Q_1$$

The one-step optimal receding-horizon control is

$$u_t = -\underbrace{(1 + B^T B)^{-1} B^T A}_{L} x_t = \begin{bmatrix} 1 & 0 \end{bmatrix} x_t$$

which yields an unstable closed-loop system

$$x_{t+1} = (A - BL)x_t = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} x_t$$

What went wrong?

Can ensure stability and performance of receding-horizon LQR by

- increasing the planning horizon N, and/or
- selecting the terminal penalty $x_N^T Q_f x_N$ properly

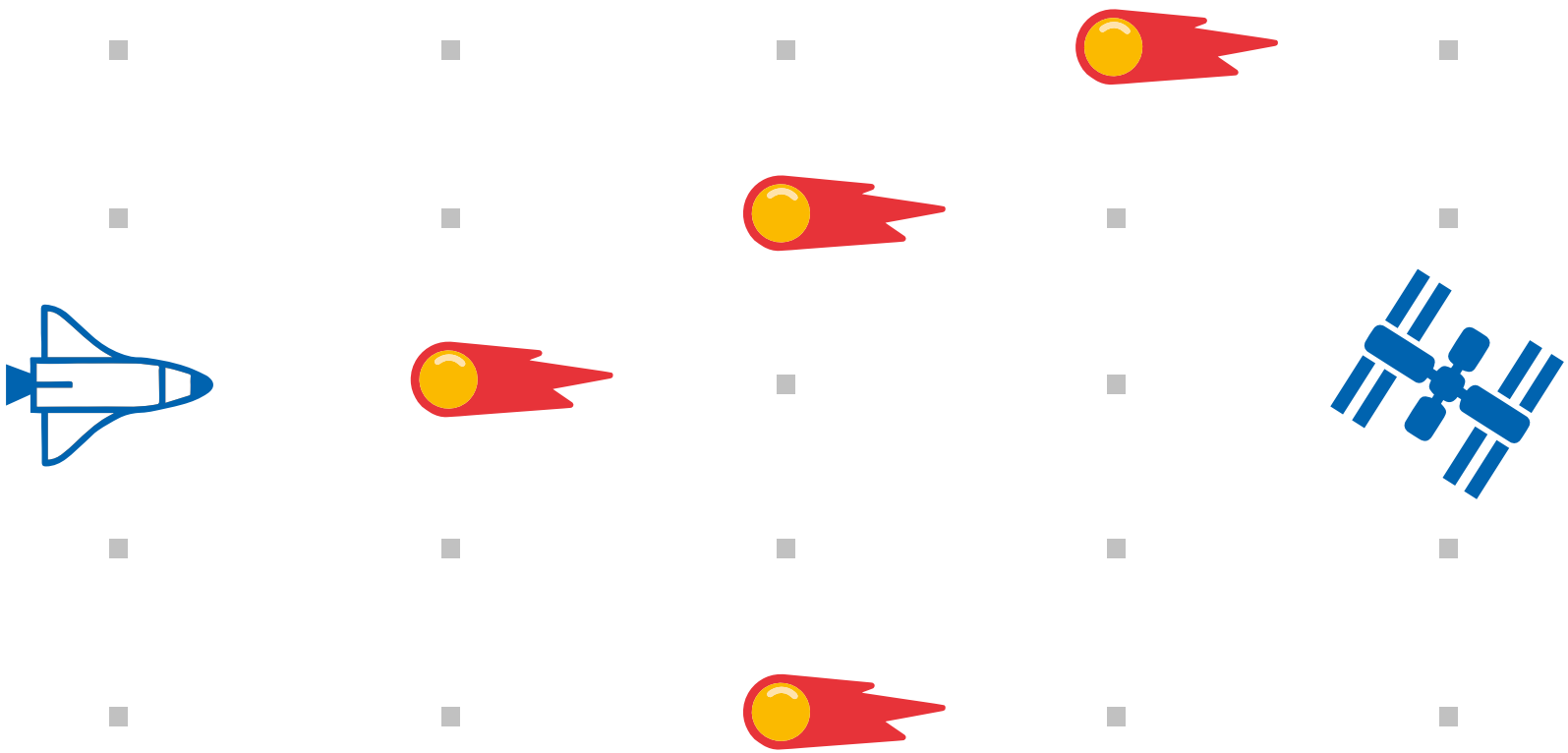
Need to work a little more to understand how (and why).

Outline

- Linear quadratic regulation
- Finite-horizon open-loop solution via least-squares
- A first attempt with receding-horizon control
- Finite-horizon feedback solution via dynamic programming

Dynamic programming: introductory example

Simple game: dock the space shuttle with the space station

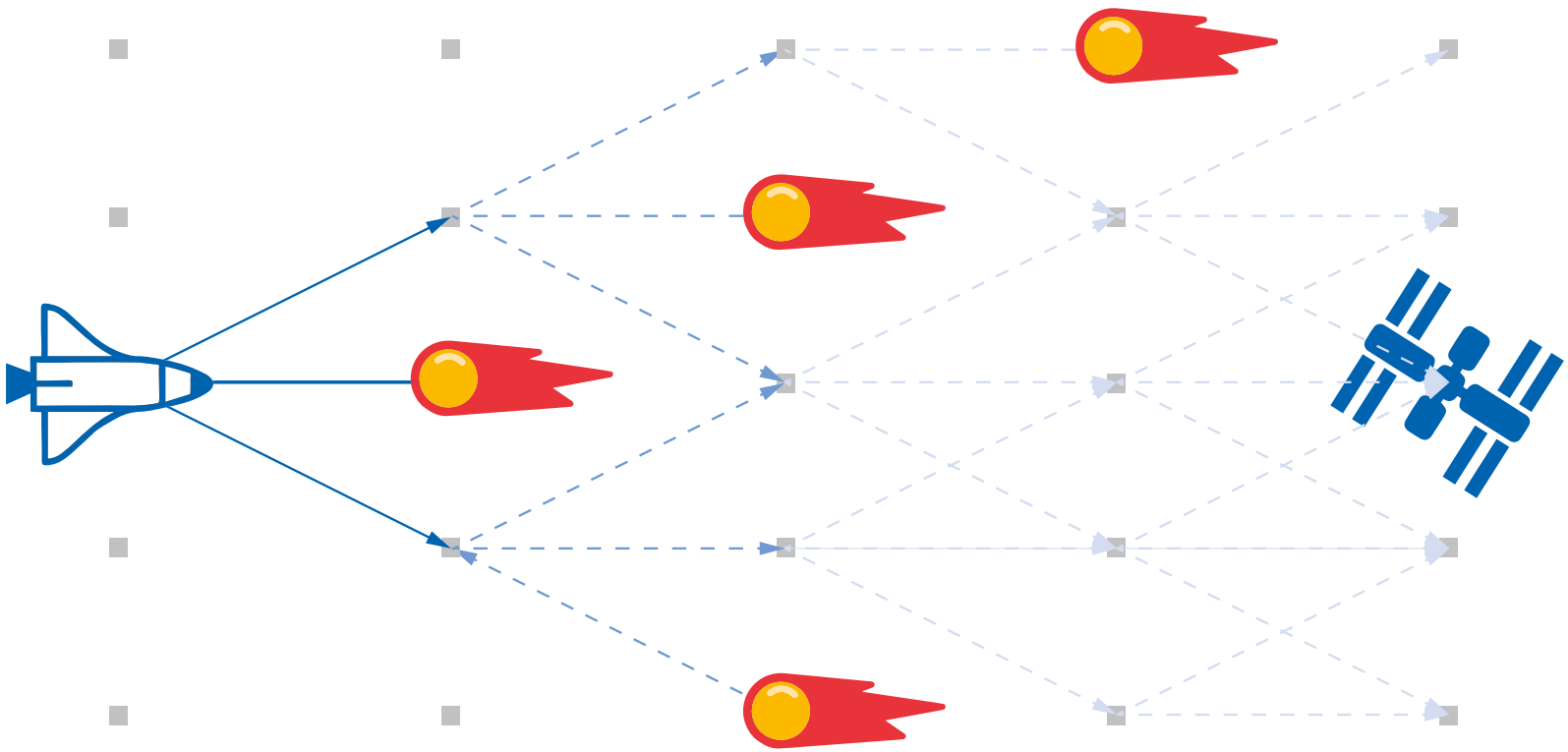


Dynamics: $x_{t+1} = x_t + 1, \quad y_{t+1} = y_t + u_t.$

Aim: find $u_t \in \{-1, 0, 1\}$ that reaches goal, minimizes $\sum_{t=0}^{3} |u_t|$.

Dynamic programing: introductory example

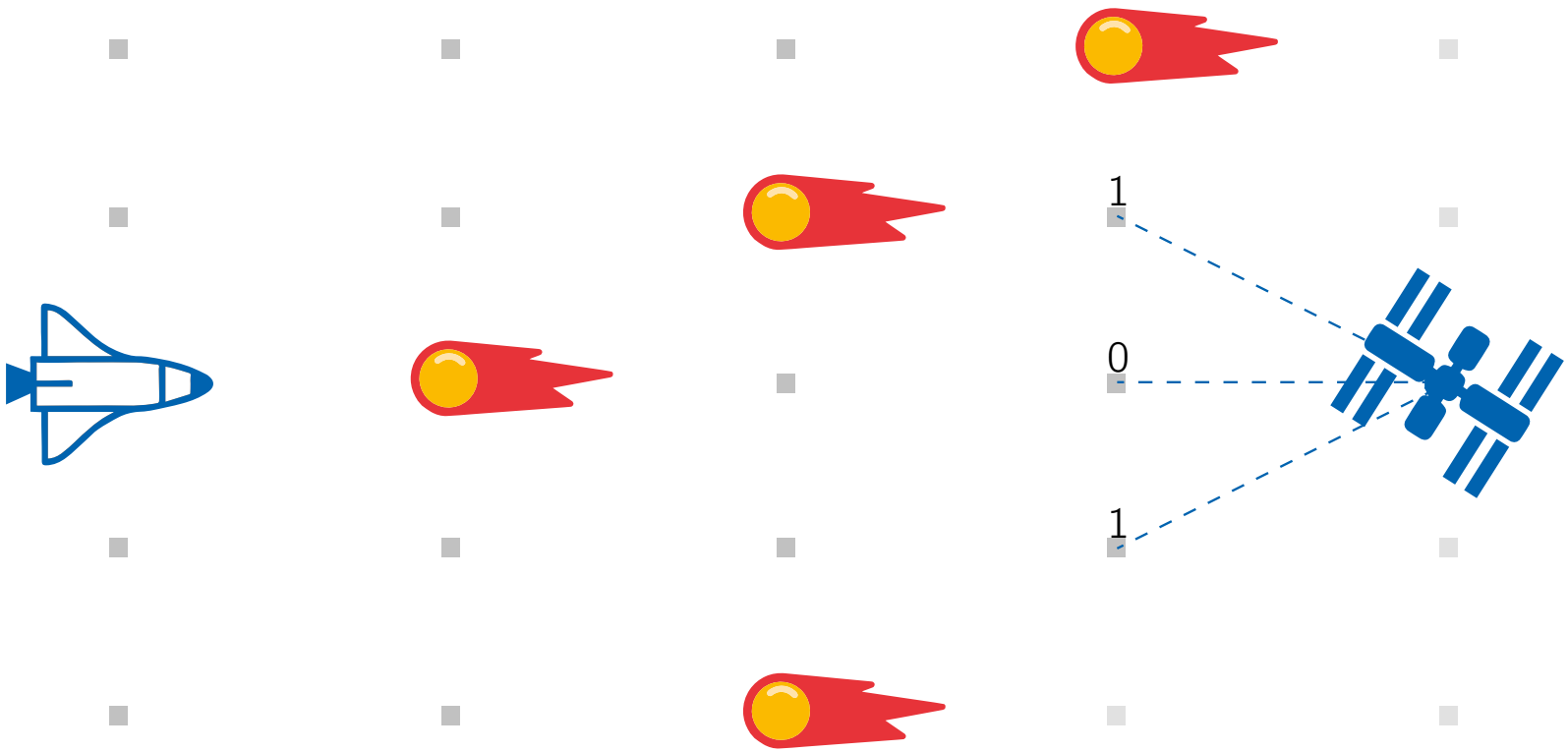
Planning forward in time is difficult, no idea of "value" of next state



May need to enumerate all paths and evaluate their cost.

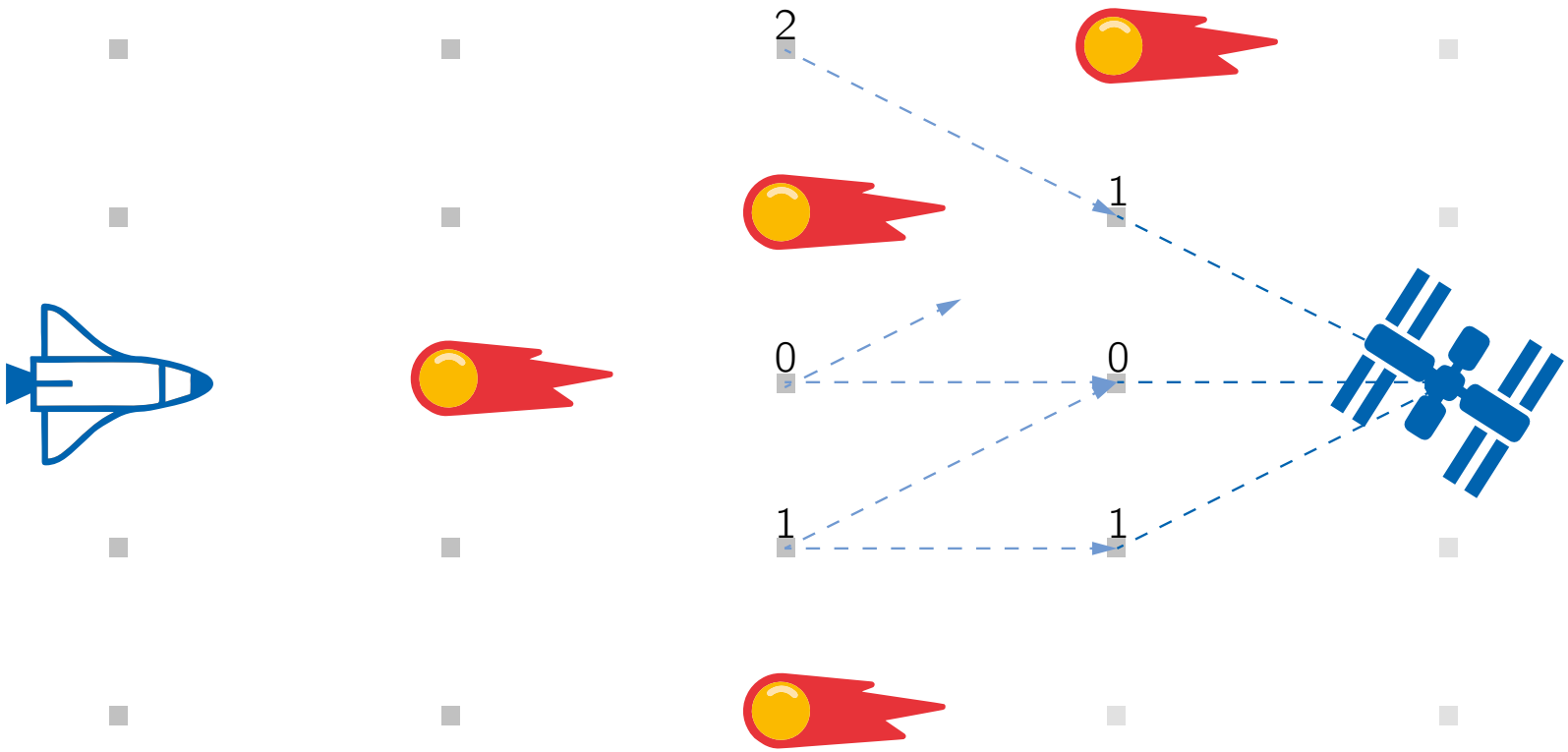
Dynamic programming: introductory example

Let's try to work our way backwards, keeping track of "cost-to-go"



Dynamic programming: introductory example

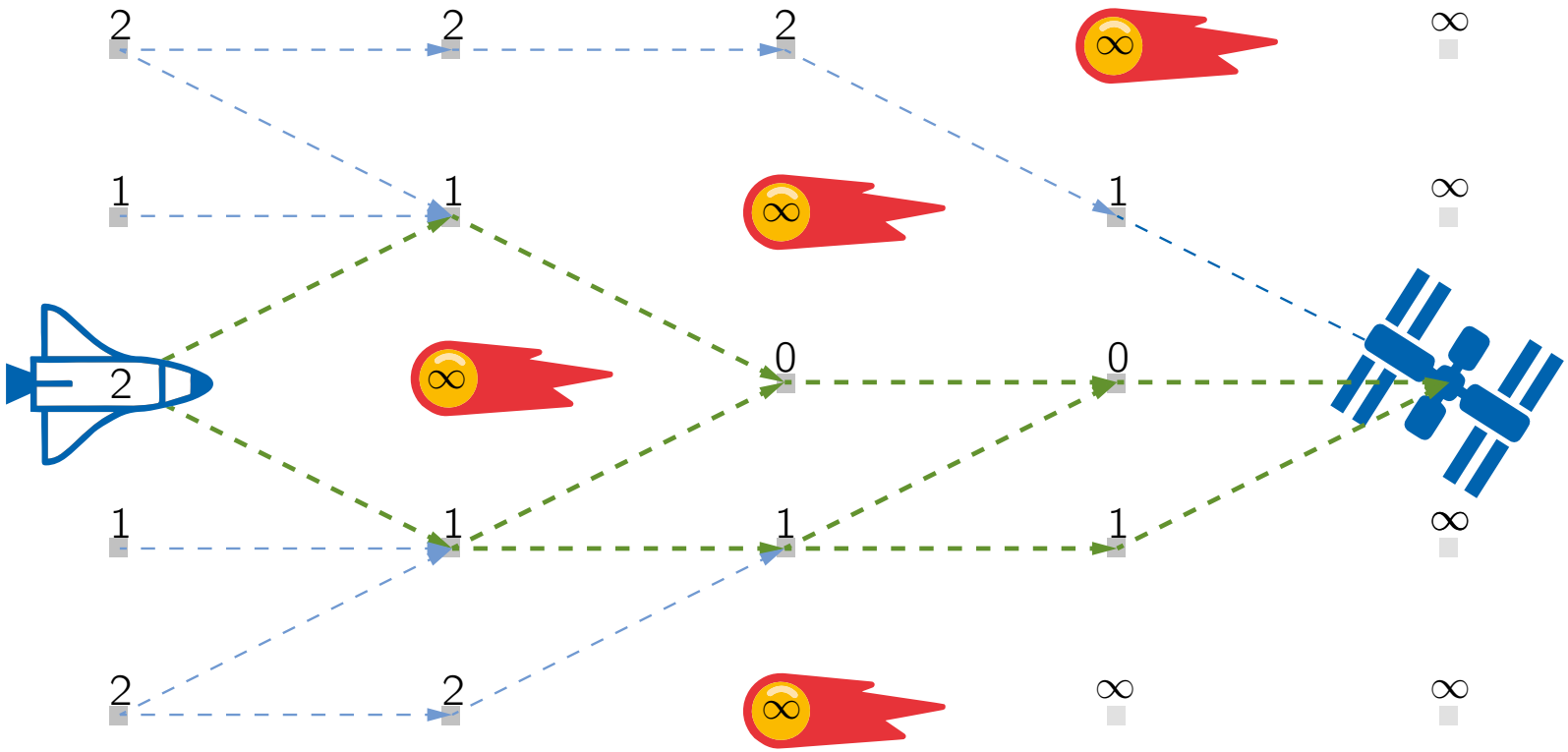
Sufficient to keep track of "optimal cost-to-go"



Can identify (and eliminate) suboptimal actions and paths.

Dynamic programming: introductory example

Final solution, optimal paths and cost-to-go for all states/times



Dynamic programming

Dynamic programming: a technique for sequential optimization.

Based on Bellman's principle of optimality:

> *For an optimal policy, no matter what the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

Value of next state can be characterized by its optimal "cost-to-go".

- a function of the state, often computed recursively, backwards in time
- optimal action guided by current cost and cost-to-go from next state

Dynamic programming algorithm

For every x_0, the finite-horizon optimal cost is given by the cost-to-go function $v_0(x_0)$, defined from the last step of the following algorithm:

$$
\begin{aligned}
v_N(x) &= g_N(x) \\
v_k(x) &= \min_{u_k \in U_k} \{g_k(x, u_k) + v_{k+1}(f_k(x, u_k))\} \qquad k = N-1, N-2, \ldots, 0
\end{aligned}
$$

If $u_k^\star = \varphi_k^\star(x)$ minimizes the right-hand-side for each x and each k, then

$$\varphi^\star = \{\varphi_0^\star, \ldots, \varphi_{N-1}^\star\}$$

is an optimal policy for the optimal control problem. Moreover,

$$
v_k(x) = \begin{array}{ll} \min\limits_{u_k, \ldots, u_{N-1}} & \sum_{t=k}^{N-1} g_t(x_t, u_t) + g_N(x_N) \\ \text{subject to} & x_{t+1} = f_t(x_t, u_t),\ x_k = x \end{array}
$$

Note. State constraints accounted for by letting $g_k(x, u_k) = +\infty$ if $x \notin X_k$.

Dynamic programming comments

Dynamic programming is very powerful if

(a) we have an efficient representation of the cost-to-go functions $v_k(x)$

(b) we have an efficient way to solve the optimization problems

$$\underset{u_k \in U_k}{\text{minimize}} \quad g_k(x, u_k) + v_{k+1}(f_k(x, u_k))$$

For LQR, v_k are quadratic and minimization admits an explicit solution:

- can treat large system dimension and planning horizons

In design project, v_k is represented by a table, minimization is non-convex:

- can only deal with 2-3 dimensions, and limited horizon lengths

Finite-horizon LQR via dynamic programming

First, note that $v_N(x) = x^T P_f x$.

For $t \in 0, \ldots, N-1$, sssume that $v_{t+1}(z) = z^T P_{t+1} z$ for some $P_{t+1} \succeq 0$ (this assumption holds for $t = N-1$, with $P_{t+1} = P_f$)

Then,

$$v_t = z^T Q_1 z + \min_w \left(w^T (Q_2 + B^T P_{t+1} B) w + 2 z^T A^T P_{t+1} B w + z^T A^T P_{t+1} A z \right) =$$
$$= z^T \left(Q_1 + A^T P_{t+1} A - A^T P_{t+1} B (Q_2 + B^T P_{t+1} B)^{-1} B P_{t+1} A \right) z := z^T P_t z$$

so v_t is also quadratic. Note that the optimal control

$$u_t^\star = -(Q_2 + B^T P_{t+1} B)^{-1} B^T P_{t+1} A x_t$$

is a linear function of the current state.

Finite-horizon LQR solution

Algorithm 1 pseudocode for finite-horizon LQR via dynamic programming

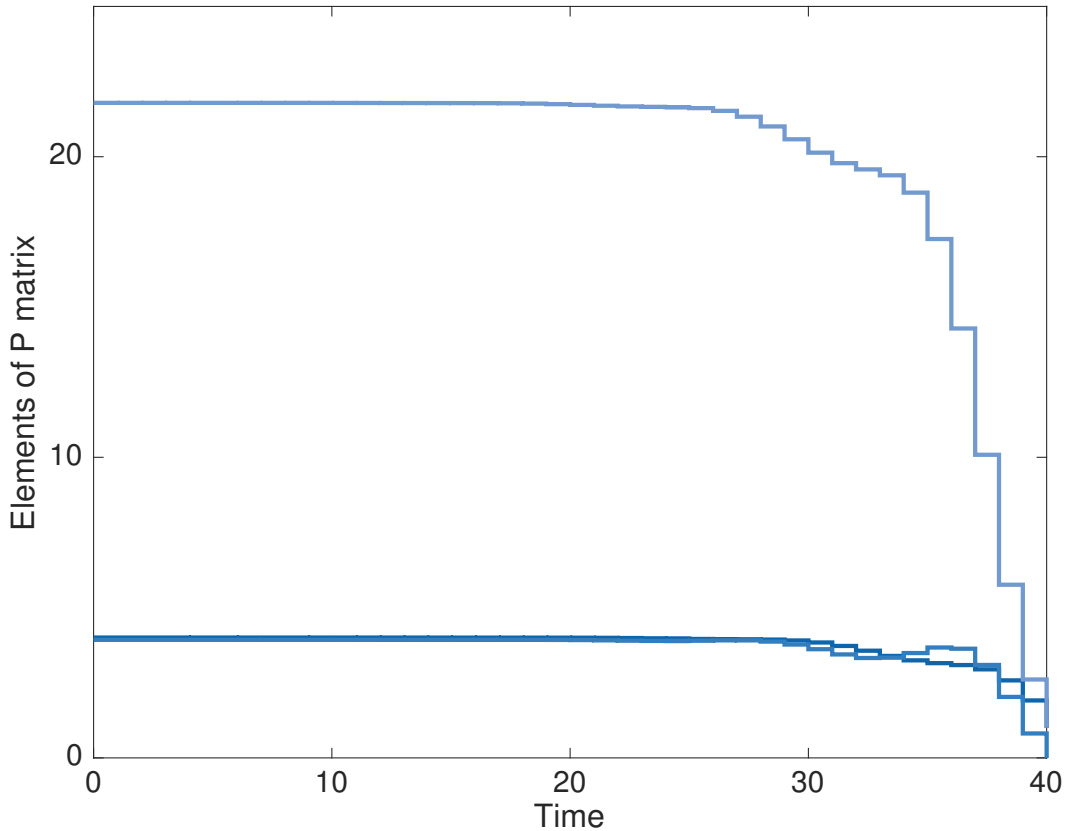
$P_N := Q_f$
for $t = N, N-1, \ldots, 1$ **do**
  $P_{t-1} := Q_1 + A^T P_t A - A^T P_t B (Q_2 + B^T P_t B)^{-1} B^T P_t A$
end for
for $t = 0, \ldots, N-1$ **do**
  $L_t := (Q_2 + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$
  $u_t^\star := -L_t x_t$
end for

Notes:

- recursion for minimum cost-to-go runs backwards in time
- optimal control is linear in the current state, gain depends on cost-to-go

Example: rapid convergence of Riccati recursion

Elements of P_t for for mechanical system (using $Q_1 = Q_f = I$, $Q_2 = 1$)



P recursion, run backwards in time from $N = 40$, converges quickly:

- key for understanding how to "fix" receding-horizon LQR

DP as minimization of multi-stage functions

DP can be interpreted as recursive minimization of multi-stage functions

$$J(z_1, \ldots, z_N; z_0) = g_0(z_0, z_1) + g_1(z_1, z_2) + \cdots + g_{N-1}(z_{N-1}, z_N) + g_N(z_N)$$

Letting $v_N(z) = g_N(z)$, we proceed backwards via

$$v_{N-1}(z_{N-1}) = \min_{z_N} \; g_{N-1}(z_{N-1}, z_N) + g_N(z_N)$$

We consider z_{N-1} as a parameter and find the optimizer $z_N^\star(z_{N-1})$. Then

$$v_{N-1}(z_{N-1}) = g_{N-1}(z_{N-1}, z_N^\star(z_{N-1})) + g_N(z_N^\star(z_{N-1}))$$

We can now continue to compute the cost-to-go in stage $N-2$:

$$v_{N-2}(z_{N-2}) = \min_{z_{N-1}} \; g_{N-2}(z_{N-2}, z_{N-1}) + v_{N-1}(z_{N-1})$$

Summary

- Linear quadratic regulation
- Finite-horizon open-loop solution via least-squares
- A first attempt with receding-horizon control
- Finite-horizon feedback solution via dynamic programming

Reading instructions: Lecture notes Chapter 3.3.