Lecture 1

Linear quadratic regulator: Discrete-time finite horizon

- LQR cost function
- multi-objective interpretation
- LQR via least-squares
- dynamic programming solution
- steady-state LQR control
- extensions: time-varying systems, tracking problems

LQR problem: background

discrete-time system $x(t+1) = Ax(t) + Bu(t)$, $x(0) = x_0$

problem: choose $u(0), u(1), \ldots$ so that

- $x(0), x(1), \ldots$ is 'small', $i.e.$, we get good *regulation* or *control*
- $u(0), u(1), \ldots$ is 'small', $i.e.$, using small *input effort* or *actuator authority*

- we'll define 'small' soon
- these are usually competing objectives, $e.g.$, a large u can drive x to zero fast

linear quadratic regulator (LQR) theory addresses this question

LQR cost function

we define *quadratic cost function*

$$J(U) = \sum_{\tau=0}^{N-1} \left(x(\tau)^T Q x(\tau) + u(\tau)^T R u(\tau) \right) + x(N)^T Q_f x(N)$$

where $U = (u(0), \ldots, u(N-1))$ and

$$Q = Q^T \geq 0, \qquad Q_f = Q_f^T \geq 0, \qquad R = R^T > 0$$

are given *state cost*, *final state cost*, and *input cost* matrices

- N is called *time horizon* (we'll consider $N = \infty$ later)
- first term measures *state deviation*
- second term measures *input size* or *actuator authority*
- last term measures *final state deviation*
- Q, R set relative weights of state deviation and input usage
- $R > 0$ means any (nonzero) input adds to cost J

LQR problem: find $u_{\text{lqr}}(0), \ldots, u_{\text{lqr}}(N-1)$ that minimizes $J(U)$

Comparison to least-norm input

c.f. least-norm input that steers x to $x(N) = 0$:

- no cost attached to $x(0), \ldots, x(N-1)$
- $x(N)$ must be exactly zero

we can approximate the least-norm input by taking

$$R = I, \qquad Q = 0, \qquad Q_f \text{ large, } e.g., \; Q_f = 10^8 I$$

Multi-objective interpretation

common form for Q and R:

$$R = \rho I, \qquad Q = Q_f = C^T C$$

where $C \in \mathbf{R}^{p \times n}$ and $\rho \in \mathbf{R}$, $\rho > 0$

cost is then

$$J(U) = \sum_{\tau=0}^{N} \|y(\tau)\|^2 + \rho \sum_{\tau=0}^{N-1} \|u(\tau)\|^2$$

where $y = Cx$

here $\sqrt{\rho}$ gives relative weighting of output norm and input norm

Input and output objectives

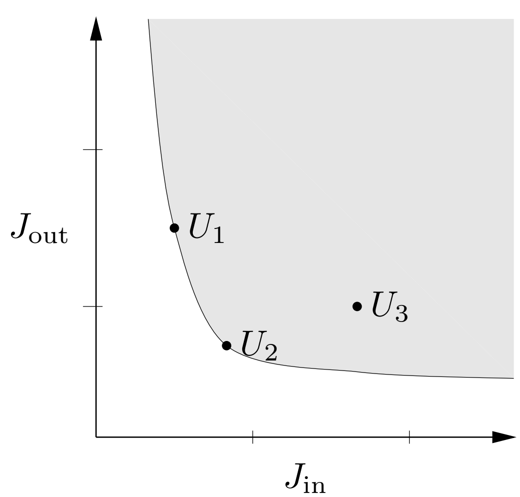
fix $x(0) = x_0$ and horizon N; for any input $U = (u(0), \ldots, u(N-1))$ define

- input cost $J_{\text{in}}(U) = \sum_{\tau=0}^{N-1} \|u(\tau)\|^2$
- output cost $J_{\text{out}}(U) = \sum_{\tau=0}^{N} \|y(\tau)\|^2$

these are (competing) objectives; we want *both* small

LQR quadratic cost is $J_{\text{out}} + \rho J_{\text{in}}$

plot $(J_{\text{in}}, J_{\text{out}})$ for all possible U:



- shaded area shows $(J_{\text{in}}, J_{\text{out}})$ achieved by some U
- clear area shows $(J_{\text{in}}, J_{\text{out}})$ not achieved by any U

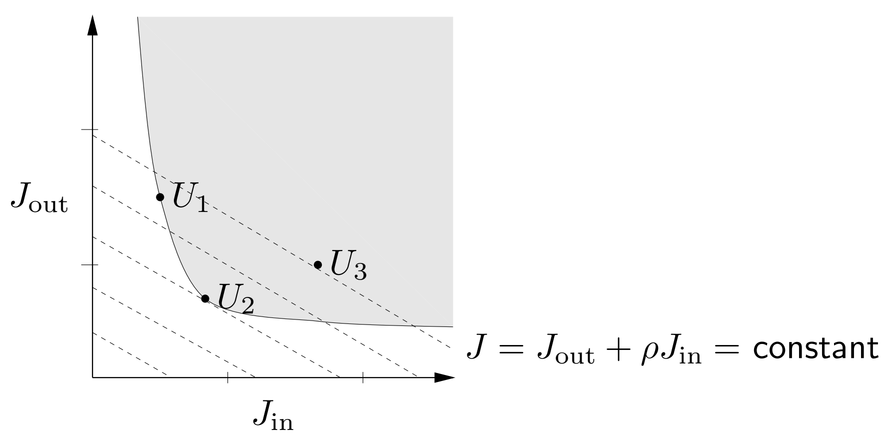
three sample inputs U_1, U_2, and U_3 are shown

- U_3 is worse than U_2 on both counts (J_{in} and J_{out})
- U_1 is better than U_2 in J_{in}, but worse in J_{out}

interpretation of LQR quadratic cost:

$$J = J_{\text{out}} + \rho J_{\text{in}} = \text{ constant}$$

corresponds to a line with slope $-\rho$ on $(J_{\text{in}}, J_{\text{out}})$ plot



- LQR optimal input is at boundary of shaded region, just touching line of smallest possible J
- u_2 is LQR optimal for ρ shown
- by varying ρ from 0 to $+\infty$, can sweep out *optimal tradeoff curve*

LQR via least-squares

LQR can be formulated (and solved) as a (large) least-squares problem

note that $X = (x(0), \ldots x(N))$ is a *linear function* of $x(0)$ and $U = (u(0), \ldots, u(N-1))$:

$$\begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} = \begin{bmatrix} B & 0 & \cdots & \\ AB & B & 0 & \cdots \\ \vdots & \vdots & & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix} + \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} x(0)$$

can express as $X = GU + Hx(0)$, where $G \in \mathbf{R}^{Nn \times Nm}$, $H \in \mathbf{R}^{Nn \times n}$

can express LQR cost as

$$\begin{aligned} J(U) &= \left\| \mathbf{diag}(Q^{1/2}, \ldots, Q^{1/2}, Q_f^{1/2})(GU + Hx(0)) \right\|^2 \\ &+ \left\| \mathbf{diag}(R^{1/2}, \ldots, R^{1/2})U \right\|^2 \end{aligned}$$

this is just a (big) least-squares problem

this solution method requires forming and solving a least-squares problem with size that *grows* with N

Dynamic programming solution

- gives an efficient, recursive method to solve LQR least-squares problem
- useful and important idea on its own

for $t = 0, \ldots, N$ define the **value function** $V_t : \mathbf{R}^n \to \mathbf{R}$ by

$$V_t(z) = \min_{u(t),\ldots,u(N-1)} \sum_{\tau=t}^{N-1} \left(x(\tau)^T Q x(\tau) + u(\tau)^T R u(\tau) \right) + x(N)^T Q_f x(N)$$

subject to $x(t) = z$, $x(\tau+1) = Ax(\tau) + Bu(\tau)$

- $V_t(z)$ gives the minimum LQR cost-to-go, starting from state z at time t
- $V_0(x_0)$ is min LQR cost (from state x_0 at time 0)

we will find that

- V_t is quadratic, *i.e.*, $V_t(z) = z^T P_t z$, where $P_t = P_t^T \geq 0$
- P_t can be found recursively, working backwards from $t = N$
- the LQR optimal u is easily expressed in terms of P_t

cost-to-go with no time left is just final state cost:

$$V_N(z) = z^T Q_f z$$

thus we have $P_N = Q_f$

Dynamic programming principle

now suppose we know $V_{t+1}(z)$

what is the optimal choice for $u(t)$?

choice of $u(t)$ affects

- current cost incurred (through $u(t)^T Ru(t)$)
- where we land, *i.e.*, $x(t+1)$ (hence, the min-cost-to-go from $x(t+1)$)

dynamic programming (DP) principle:

$$V_t(z) = \min_w \left(z^T Qz + w^T Rw + V_{t+1}(Az + Bw) \right)$$

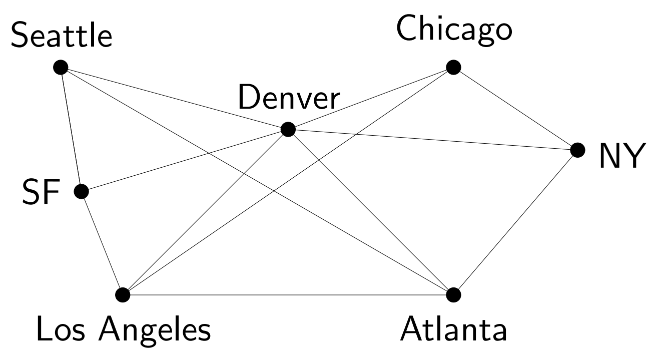
- $z^T Qz + w^T Rw$ is cost incurred at time t if $u(t) = w$;
  $V_{t+1}(Az + Bw)$ is min cost-to-go from where you land at $t+1$
- follows from fact that we can minimize in any order:

$$\min_{w_1,\ldots,w_k} f(w_1,\ldots,w_k) = \min_{w_1} \underbrace{\left(\min_{w_2,\ldots,w_k} f(w_1,\ldots,w_k) \right)}_{\text{a fct of } w_1}$$

in words:
min cost-to-go from where you are = min over
(current cost incurred + min cost-to-go from where you land)

Example: path optimization

- edges show possible flights; each has some cost
- want to find min cost route or path from SF to NY



dynamic programming (DP):

- $V(i)$ is min cost from airport i to NY, over all possible paths
- to find min cost from city i to NY: minimize sum of flight cost plus min cost to NY from where you land, over all flights out of city i
(gives optimal flight out of city i on way to NY)
- if we can find $V(i)$ for each i, we can find min cost path from any city to NY
- DP principle: $V(i) = \min_j (c_{ji} + V(j))$, where c_{ji} is cost of flight from i to j, and minimum is over all possible flights out of i

HJ equation for LQR

$$V_t(z) = z^T Q z + \min_w \left(w^T R w + V_{t+1}(Az + Bw) \right)$$

- called DP, Bellman, or Hamilton-Jacobi equation
- gives V_t recursively, in terms of V_{t+1}
- any minimizing w gives optimal $u(t)$

DP has many applications beyond LQR, *e.g.*,

- optimal flow control in communication networks
- optimization in finance

we know $V_N(z) = z^T P_N z$ where $P_N = Q_f$

by DP,

$$V_{N-1}(z) = z^T Q z + \min_w \left(w^T R w + (Az + Bw)^T P_N (Az + Bw) \right)$$

can solve by setting derivative w.r.t. w to zero:

$$2w^T R + 2(Az + Bw)^T P_N B = 0$$

hence optimal w is

$$w^* = -(R + B^T P_N B)^{-1} B^T P_N A z$$

and so

$$V_{N-1}(z) = z^T Qz + w^{*T} Rw^* + (Az + Bw^*)^T P_N (Az + Bw^*)$$

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

(after some ugly algebra)

we conclude that V_{N-1} is quadratic: $V_{N-1}(z) = z^T P_{N-1} z$ where

$$P_{N-1} = Q + A^T P_N A - A^T P_N B (R + B^T P_N B)^{-1} B^T P_N A$$

this recursion works for all t:

once we know $V_t(z) = z^T P_t z$ is quadratic, we find that V_{t-1} is as well, $i.e.$, $V_{t-1}(z) = z^T P_{t-1} z$, with

$$P_{t-1} = Q + A^T P_t A - A^T P_t B (R + B^T P_t B)^{-1} B^T P_t A$$

together with $P_N = Q_f$, we can find $P_0, \ldots, P_N$ by recursion (backwards in time)

called **Riccati recursion** for P_t

and the optimizing w is

$$w^* = -(R + B^T P_t B)^{-1} B^T P_t A z$$

Summary of LQR solution via DP

1. set $P_N := Q_f$
2. for $t = N, \ldots, 1$,

$$P_{t-1} := Q + A^T P_t A - A^T P_t B (R + B^T P_t B)^{-1} B^T P_t A$$

3. define $K_t := -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$
4. optimal u is given by $u_{\mathrm{lqr}}(t) = K_t x(t)$

comments:

- optimal u is a linear function of the state (called *linear state feedback*)
- recursion for min cost-to-go runs backwards in time
- solves least-squares problem with $(N+1)m$ variables much faster than direct least-squares method

LQR example

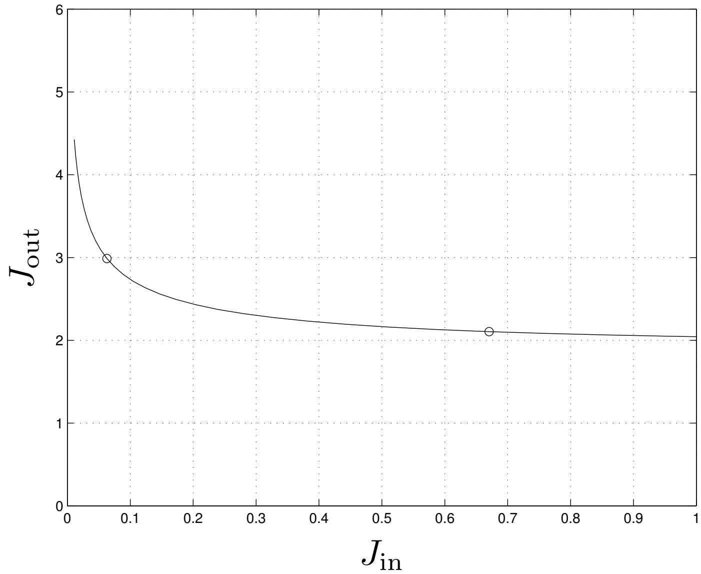
2-state, single-input, single-output system

$$x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \qquad y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

with initial state $x(0) = (1, 0)$, horizon $N = 20$, and weight matrices

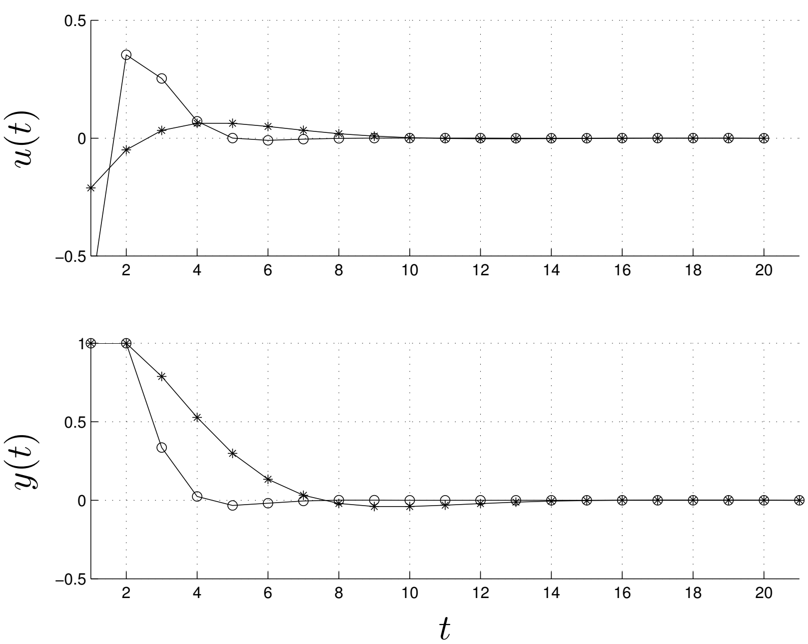
$$Q = Q_f = C^T C, \qquad R = \rho I$$

optimal trade-off curve of $J_{\rm in}$ vs. $J_{\rm out}$:



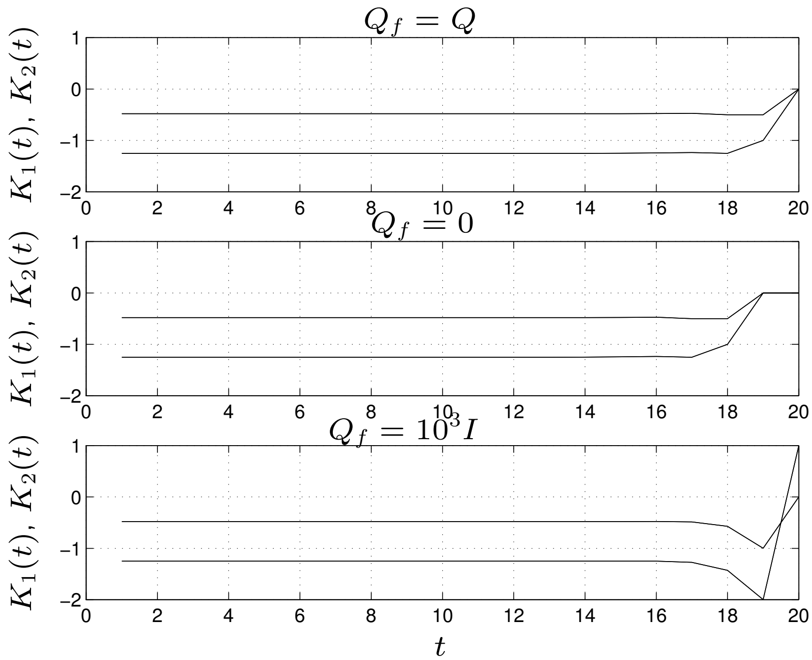
circles show LQR solutions with $\rho = 0.3$, $\rho = 10$

u & y for $\rho = 0.3$, $\rho = 10$:



optimal input has form $u(t) = K(t)x(t)$, where $K(t) \in \mathbf{R}^{1\times 2}$

state feedback gains vs. t for various values of Q_f (note convergence):



Steady-state regulator

usually P_t rapidly converges as t decreases below N

limit or steady-state value $P_{\rm ss}$ satisfies

$$P_{\rm ss} = Q + A^T P_{\rm ss} A - A^T P_{\rm ss} B (R + B^T P_{\rm ss} B)^{-1} B^T P_{\rm ss} A$$

which is called the (DT) algebraic Riccati equation (ARE)

- $P_{\rm ss}$ can be found by iterating the Riccati recursion, or by direct methods
- for t not close to horizon N, LQR optimal input is approximately a linear, constant state feedback

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

(very widely used in practice; more on this later)

Time-varying systems

LQR is readily extended to handle time-varying systems

$$x(t+1) = A(t)x(t) + B(t)u(t)$$

and time-varying cost matrices

$$J = \sum_{\tau=0}^{N-1} \left(x(\tau)^T Q(\tau) x(\tau) + u(\tau)^T R(\tau) u(\tau) \right) + x(N)^T Q_f x(N)$$

(so Q_f is really just $Q(N)$)

DP solution is readily extended, but (of course) there need not be a steady-state solution

Tracking problems

we consider LQR cost with state and input offsets:

$$\begin{aligned} J &= \sum_{\tau=0}^{N-1} (x(\tau) - \bar{x}(\tau))^T Q (x(\tau) - \bar{x}(\tau)) \\ &+ \sum_{\tau=0}^{N-1} (u(\tau) - \bar{u}(\tau))^T R (u(\tau) - \bar{u}(\tau)) \end{aligned}$$

(we drop the final state term for simplicity)

here, $\bar{x}(\tau)$ and $\bar{u}(\tau)$ are given desired state and input trajectories

DP solution is readily extended, even to time-varying tracking problems

Gauss-Newton LQR

nonlinear dynamical system: $x(t+1) = f(x(t), u(t))$, $x(0) = x_0$

objective is

$$J(U) = \sum_{\tau=0}^{N-1} \left(x(\tau)^T Q x(\tau) + u(\tau)^T R u(\tau) \right) + x(N)^T Q_f x(N)$$

where $Q = Q^T \geq 0$, $Q_f = Q_f^T \geq 0$, $R = R^T > 0$

start with a guess for U, and alternate between:

- linearize around current trajectory
- solve associated LQR (tracking) problem

sometimes converges, sometimes to the globally optimal U

some more detail:

- let u denote current iterate or guess
- simulate system to find x, using $x(t+1) = f(x(t), u(t))$
- linearize around this trajectory: $\delta x(t+1) = A(t)\delta x(t) + B(t)\delta u(t)$

$$A(t) = D_x f(x(t), u(t)) \qquad B(t) = D_u f(x(t), u(t))$$

- solve time-varying LQR tracking problem with cost

$$\begin{aligned} J &= \sum_{\tau=0}^{N-1} (x(\tau) + \delta x(\tau))^T Q (x(\tau) + \delta x(\tau)) \\ &+ \sum_{\tau=0}^{N-1} (u(\tau) + \delta u(\tau))^T R (u(\tau) + \delta u(\tau)) \end{aligned}$$

- for next iteration, set $u(t) := u(t) + \delta u(t)$