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 $x_{t+1} = Ax_t + Bu_t$, $x_0 = x^{\mathrm{init}}$

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 $u_0^{\mathrm{lqr}}, \ldots, u_{N-1}^{\mathrm{lqr}}$ 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

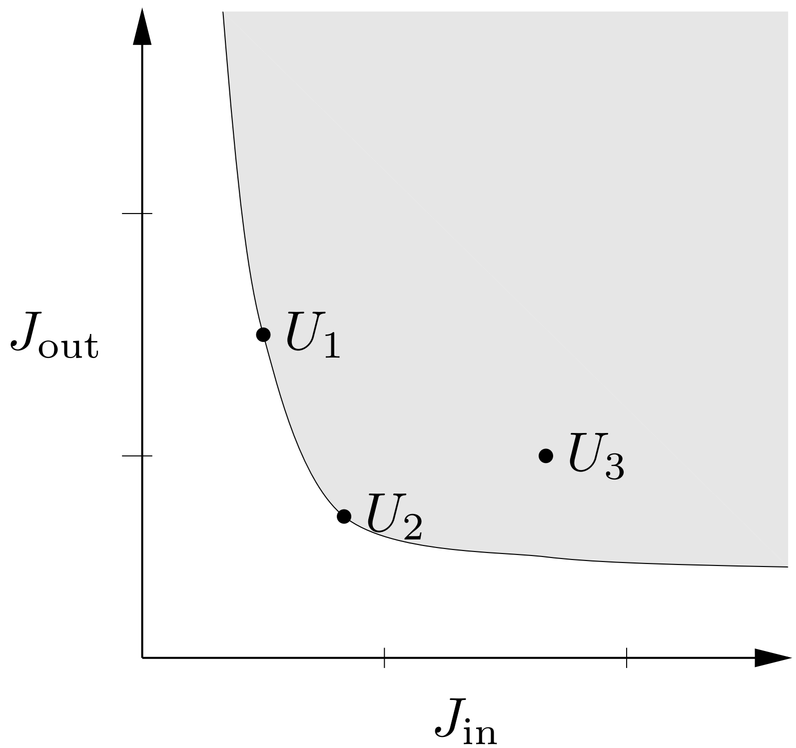
fix $x_0 = x^{\mathrm{init}}$ and horizon N; for any input $U = (u_0, \ldots, u_{N-1})$ define

- input cost $J_{\mathrm{in}}(U) = \sum_{\tau=0}^{N-1} \|u_\tau\|^2$
- output cost $J_{\mathrm{out}}(U) = \sum_{\tau=0}^{N} \|y_\tau\|^2$

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

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

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

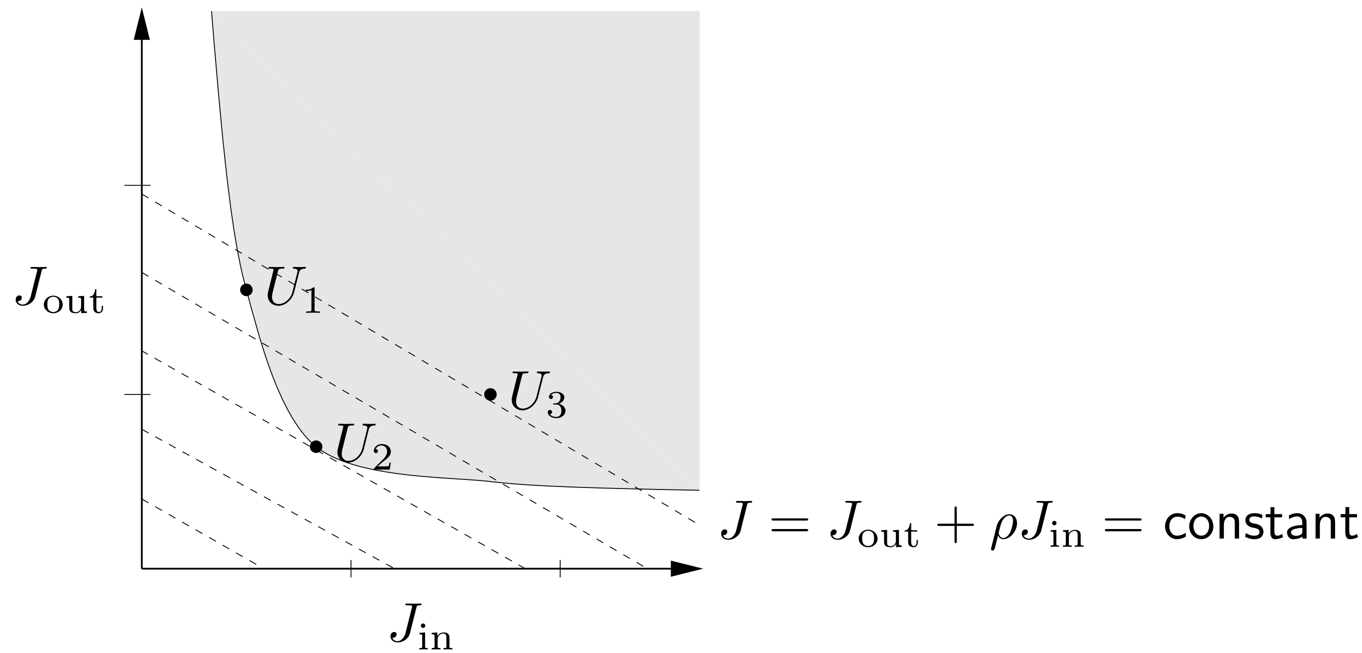
three sample inputs U_1, U_2, and U_3 are shown

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

interpretation of LQR quadratic cost:

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

corresponds to a line with slope $-\rho$ on $(J_{\rm in}, J_{\rm out})$ 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 least-squares problem

$X = (x_0, \dots x_N)$ is a *linear function* of x_0 and $U = (u_0, \dots, u_{N-1})$:

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

express as $X = GU + Hx_0$, where $G \in \mathbf{R}^{Nn \times Nm}$, $H \in \mathbf{R}^{Nn \times n}$

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 $N(n+m) \times Nm$

using a naive method ($e.g.$, QR factorization), cost is $O(N^3nm^2)$

Dynamic programming solution

- gives an efficient, recursive method to solve LQR least-squares problem; cost is $O(Nn^3)$
- (but in fact, a less naive approach to solve the LQR least-squares problem will have the same complexity)
- useful and important idea on its own
- same ideas can be used for many other problems

Value function

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 $x_t = z$, $x_{\tau+1} = A x_\tau + B u_\tau$, $\tau = t, \ldots, T$

- $V_t(z)$ 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 backward 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 $u_t^T R u_t$)
  - where we land, x_{t+1} (hence, the min-cost-to-go from x_{t+1})
- **dynamic programming (DP) principle:**

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

  - $z^T Q z + w^T R w$ 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 $t + 1$

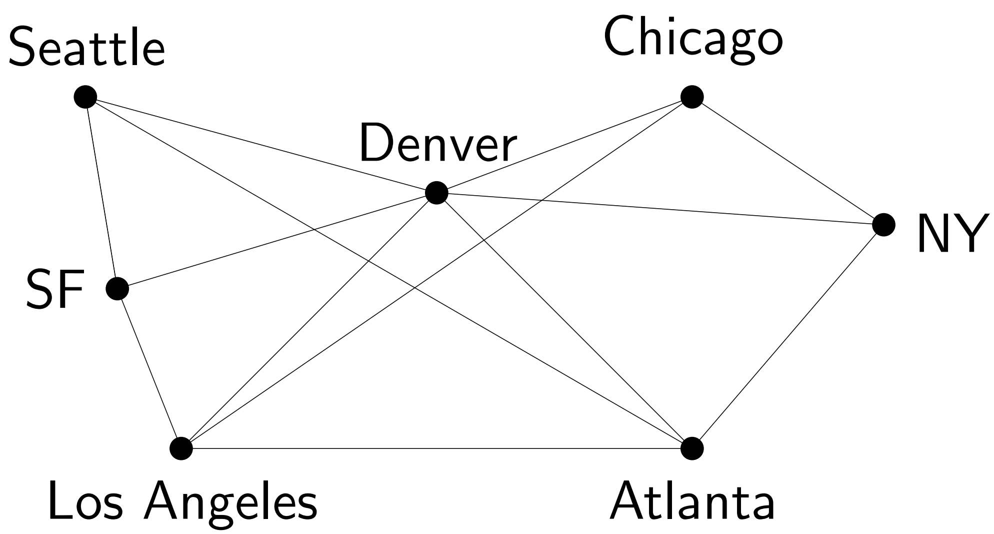
- follows from fact that we can minimize in any order:

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

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:

$$u_t^{\mathrm{lqr}} = \underset{w}{\operatorname{argmin}} \left(w^T R w + V_{t+1}(Az + Bw) \right)$$

- let's assume that $V_{t+1}(z) = z^T P_{t+1} z$, with $P_{t+1} = P_{t+1}^T \geq 0$

- we'll show that V_t has the same form

- by DP,

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

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

$$2w^T R + 2(Az + Bw)^T P_{t+1} B = 0$$

- hence optimal input is

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

- and so (after some ugly algebra)

$$\begin{aligned} V_t(z) &= z^T Q z + w^{*T} R w^* + (Az + Bw^*)^T P_{t+1} (Az + Bw^*) \\ &= z^T \left(Q + A^T P_{t+1} A - A^T P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A \right) z \\ &= z^T P_t z \end{aligned}$$

where

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

- easy to show $P_t = P_t^T \geq 0$

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. for $t = 0, \ldots, N-1$, define $K_t := -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$

4. for $t = 0, \ldots, N-1$, optimal u is given by $u_t^{\mathrm{lqr}} = K_t x_t$

- optimal u is a linear function of the state (called *linear state feedback*)
- recursion for min cost-to-go runs backward in time

LQR example

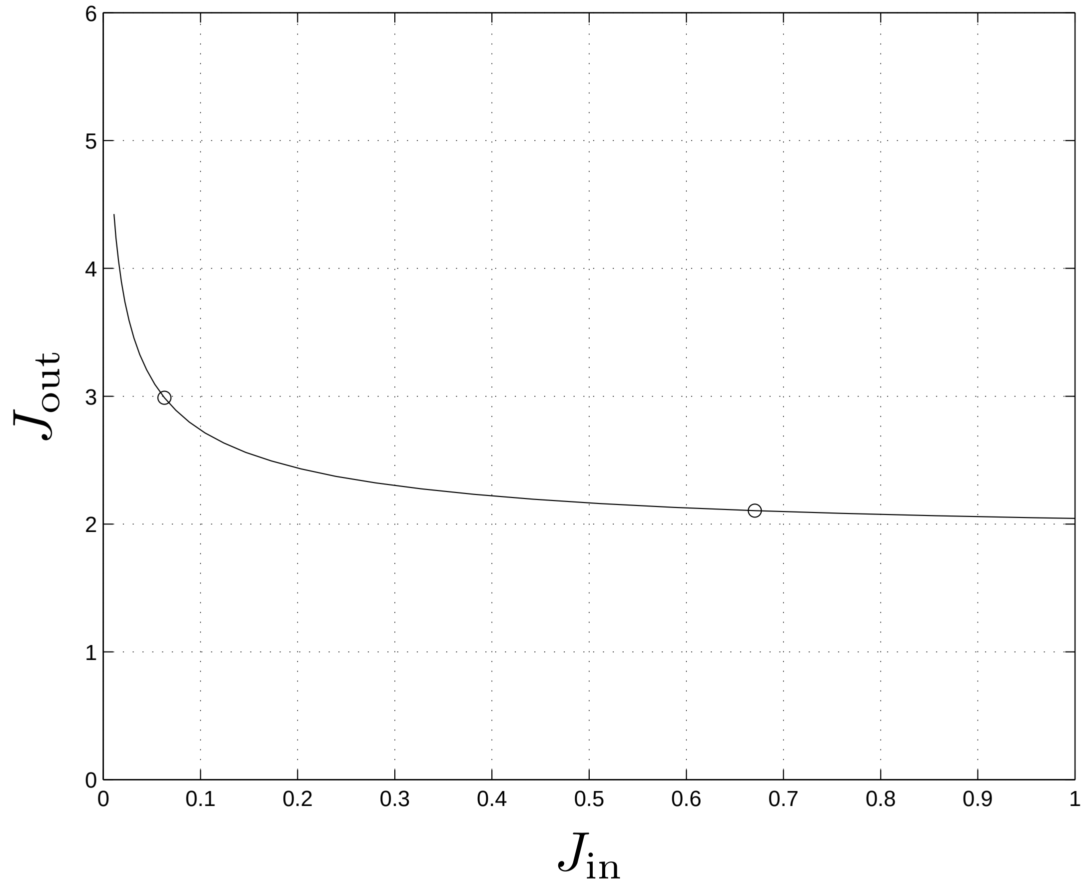
2-state, single-input, single-output system

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t, \qquad y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} x_t$$

with initial state $x_0 = (1, 0)$, horizon $N = 20$, and weight matrices

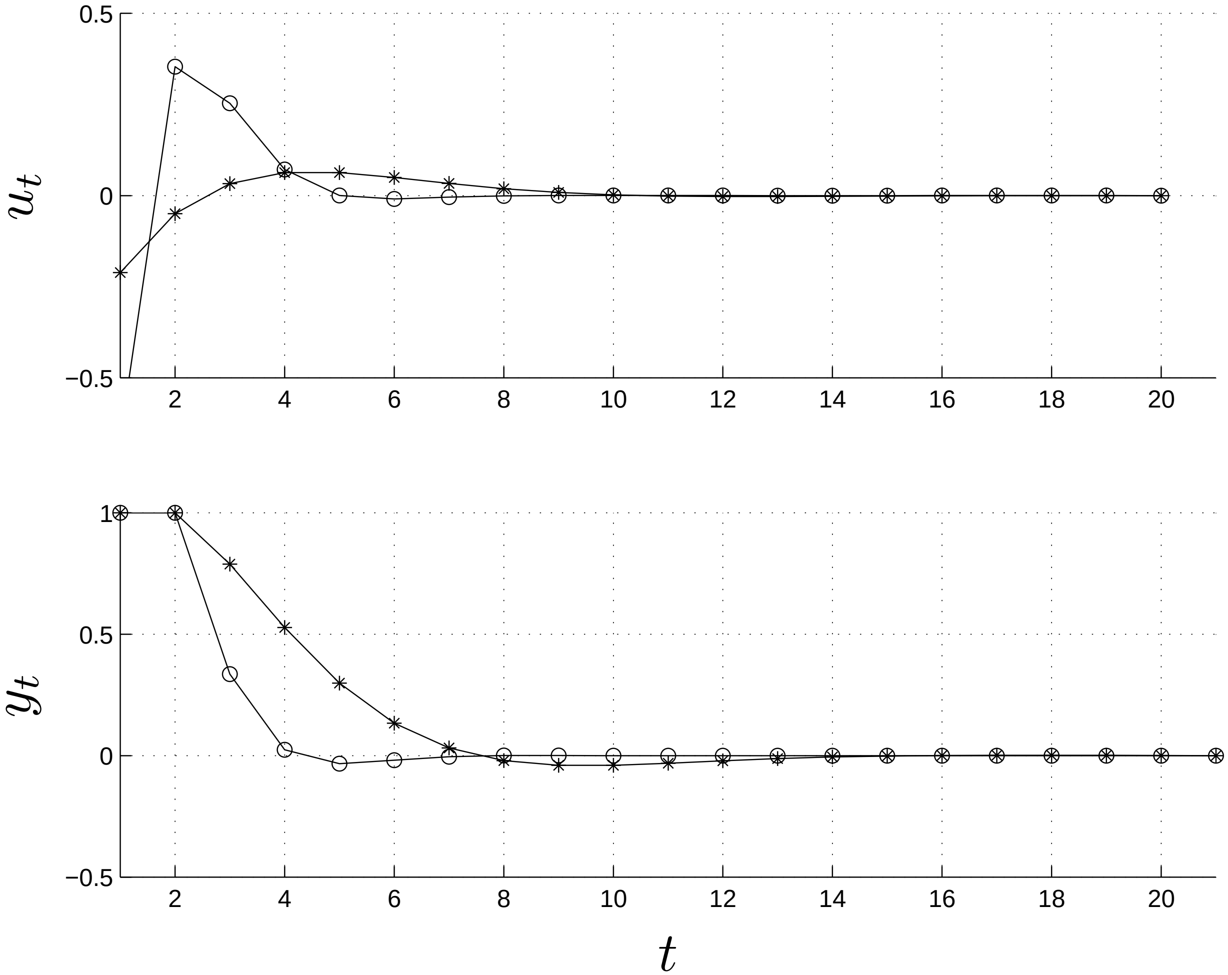
$$Q = Q_f = C^T C, \qquad R = \rho I$$

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



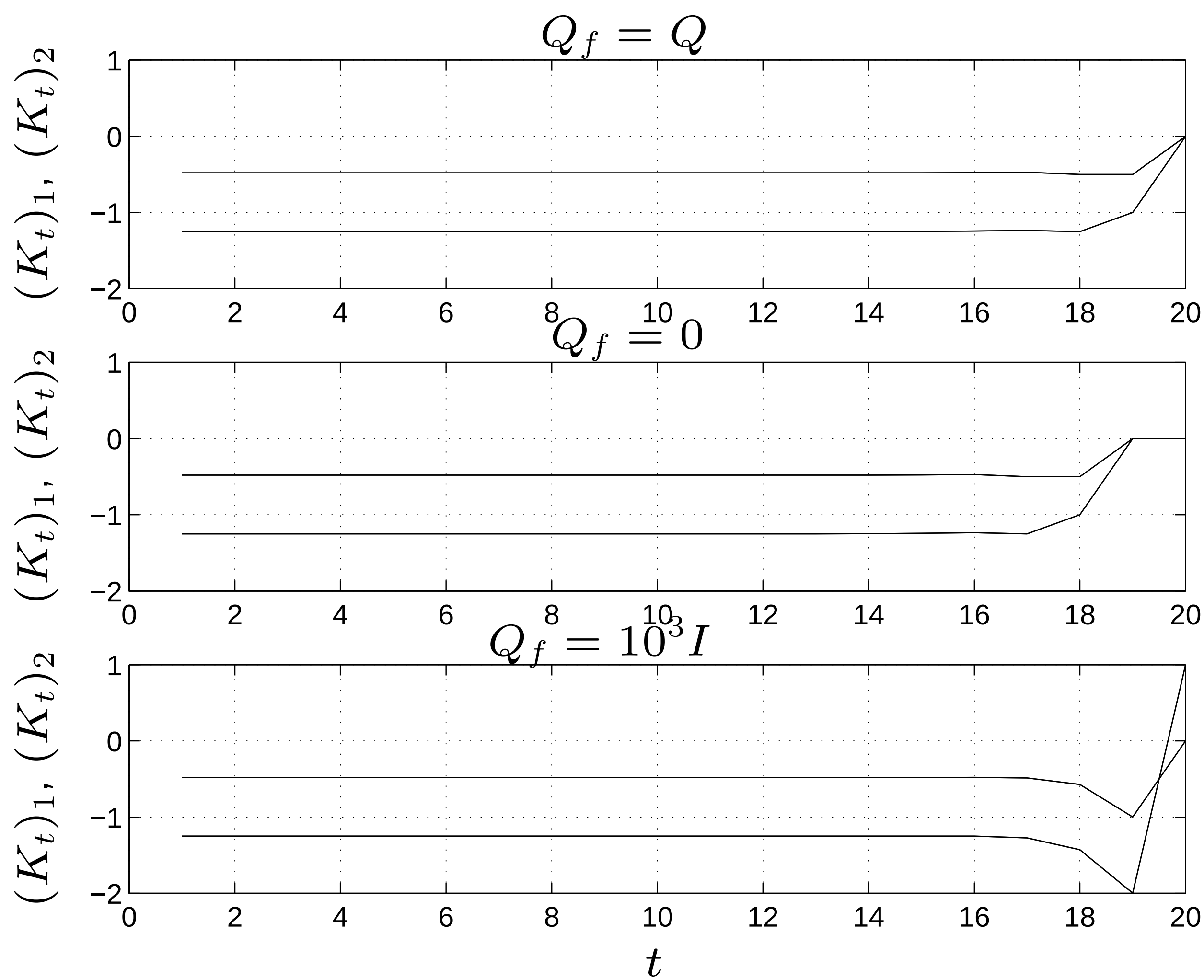
circles show LQR solutions with $\rho = 0.3$, $\rho = 10$

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: $x_{t+1} = f(x_t, u_t)$, $x_0 = x^{\mathrm{init}}$

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 $u_t := u_t + \delta u_t$