CALIFORNIA INSTITUTE OF TECHNOLOGY
Control and Dynamical Systems

CDS 110b

R. M. Murray **Linear Quadratic Regulators** 11 January 2006

i

This lecture provides a brief derivation of the linear quadratic regulator (LQR) and describes how to design an LQR-based compensator. The use of integral feedback to eliminate steady state error is also described. Two design examples are given: lateral control of the position of a simplified vectored thrust aircraft and speed control for an automobile.

1 Review from last lecture

Optimal control problem:

$$\min_u \underbrace{\int_0^T L(x,u)\,dt}_{\text{integrated cost}} + \underbrace{V\big(x(T),u(T)\big)}_{\text{final cost}}$$

subject to the constraint

$$\begin{aligned} &\dot x = f(x,u) \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m \\ &\psi(x(T)) = 0 \quad \leftarrow \text{terminal constraint} \end{aligned}$$

Hamiltonian:

$$H = L + \lambda^T f = L + \sum \lambda_i f_i$$

Necessary conditions:

$$\dot x_i = \frac{\partial H}{\partial \lambda_i} \qquad -\dot\lambda_i = \frac{\partial H}{\partial x_i} \qquad \underbrace{\begin{aligned} &x(0) \text{ given, } \psi(x(T)) = 0 \\ &\lambda(T) = \frac{\partial V}{\partial x}(x(T)) + \frac{\partial \psi^T}{\partial x}\nu \end{aligned}}_{\textit{Boundary conditions (2n total)}}$$

and

$$H(x^*(t), u^*(t), \lambda^*(t)) \le H(x^*(t), u, \lambda^*(t)) \quad \forall \quad u \in \Omega$$

2 Linear Quadratic Regulator

The finite horizon, linear quadratic regulator (LQR) is given by

$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n, u \in \mathbb{R}^n, x_0 \text{ given}$$
$$\tilde{J} = \frac{1}{2}\int_0^T \left(x^T Q_x x + u^T Q_u u\right)\, dt + \frac{1}{2}x^T(T)P_1 x(T)$$

where $Q_x \geq 0$, $Q_u > 0$, $P_1 \geq 0$ are symmetric, positive (semi-) definite matrices. Note the factor of $\frac{1}{2}$ is left out, but we included it here to simplify the derivation. Gives same answer (with $\frac{1}{2}x$ cost).

Solve via maximum principle:

$$H = x^T Q_x x + u^T Q_u u + \lambda^T(Ax + Bu)$$

$$\begin{aligned}
\dot{x} &= \left(\frac{\partial H}{\partial \lambda}\right)^T &&= Ax + Bu \qquad x(0) = x_0 \\
-\dot{\lambda} &= \left(\frac{\partial H}{\partial x}\right)^T &&= Q_x x + A^T\lambda \qquad \lambda(T) = P_1 x(T) \\
0 &= \frac{\partial H}{\partial u} &&= Q_u u + \lambda^T B \quad \implies \quad u = -Q_u^{-1}B^T\lambda.
\end{aligned}$$

This gives the optimal solution. Apply by solving *two point boundary value problem* (hard).

Alternative: guess the form of the solution, $\lambda(t) = P(t)x(t)$. Then

$$\dot{\lambda} = \dot{P}x + P\dot{x} = \dot{P}x + P(Ax - BQ_u^{-1}B^T P)x$$
$$-\dot{P}x - PAx + PBQ_u^{-1}BPx = Q_x x + A^T Px.$$

This equation is satisfied if we can find $P(t)$ such that

$$-\dot{P} = PA + A^T P - PBQ_u^{-1}B^T P + Q_x \qquad P(T) = P_1$$

Remarks:

1. This ODE is called *Riccati ODE*.
2. Can solve for $P(t)$ backwards in time and then apply

   $$u(t) = -Q_u^{-1}B^T P(t)x.$$

   This is a (time-varying) *feedback* control $\implies$ tells you how to move from *any* state to the origin.

3. Variation: set $T = \infty$ and eliminate terminal constraint:

$$\begin{aligned} J &= \int_0^\infty (x^T Q_x x + u^T Q_u u)\,dt \\ u &= -\underbrace{Q_u^{-1} B^T P}_{K} x \qquad \text{Can show } P \text{ is } \textit{constant} \\ 0 &= PA + A^T P - PBQ_u^{-1} B^T P + Q_x \end{aligned}$$

   This equation is called the *algebraic Riccati equation*.

4. In MATLAB, `K = lqr(A, B, Q_x, Q_u)`.

5. Require $Q_u > 0$ but $Q_x \geq 0$. Let $Q_x = H^T H$ (always possible) so that $L = \int_0^\infty x^T H^T H x + u^T Q_u u\,dt = \int_0^\infty \|Hx\|^2 + u^T Q_u u\,dt$. Require that (A, H) is *observable*. Intuition: if not, dynamics may not affect cost $\implies$ ill-posed.

3 Choosing LQR weights

$$\dot{x} = Ax + Bu \qquad J = \int_0^\infty \overbrace{\left(x^T Q_x x + u^T Q_u u + x^T S u\right)}^{L(x,u)} dt,$$

where the S term is almost always left out.

Q: How should we choose Q_x and Q_u?

1. Simplest choice: $Q_x = I$, $Q_u = \rho I \implies L = \|x\|^2 + \rho\|u\|^2$. Vary ρ to get something that has good response.

2. Diagonal weights

$$Q_x = \begin{bmatrix} q_1 & & \\ & \ddots & \\ & & q_n \end{bmatrix} \qquad Q_u = \rho \begin{bmatrix} r_1 & & \\ & \ddots & \\ & & r_n \end{bmatrix}$$

   Choose each q_i to given equal effort for same "badness". Eg, x_1 = distance in meters, x_3 = angle in radians:

$$\begin{aligned} &1 \text{ cm error OK} \implies q_1 = \left(\frac{1}{100}\right)^2 && q_1 x_1^2 = 1 \text{ when } x_1 = 1 \text{ cm} \\ &\frac{1}{60} \text{ rad error OK} \implies q_3 = (60)^2 && q_3 x_3^2 = 1 \text{ when } x_3 = \frac{1}{60} \text{ rad} \end{aligned}$$

   Similarly with r_i. Use ρ to adjust input/state balance.

3. Output weighting. Let $z = Hx$ be the output you want to keep small. Assume (A, H) observable. Use

$$Q_x = H^T H \qquad Q_u = \rho I \qquad \Longrightarrow \text{ trade off } \|z\|^2 \text{ vs } \rho\|u\|^2$$

4. Trial and error (on *weights*)

References

[1] K. J. Åström and R. M. Murray. *Analysis and Design of Feedback Systems*. Preprint, 2005. Available at `http://www.cds.caltech.edu/~murray/am05`.

[2] B. Friedland. *Control System Design: An Introduction to State Space Methods*. Dover, 2004.

[3] F. L. Lewis and V. L. Syrmos. *Optimal Control*. Wiley, second edition, 1995.

A MATLAB example: Caltech ducted fan

```
% L12_2dfan.m - ducted fan example for L12.2
% RMM, 14 Jan 03

%%
%% Ducted fan dynamics
%%
%% These are the dynamics for the ducted fan, written in state space
%% form.
%%

% System parameters
J = 0.0475; % inertia around pitch axis
m = 1.5; % mass of fan
r = 0.25; % distance to flaps
g = 10; % gravitational constant
gamma = 0.51; % counterweighted mass
d = 0.2;   % damping factor (estimated)
l = 0.05; % offset of center of mass

% System matrices (entire plant: 2 input, 2 output)
A = [  0     0     0     1     0     0;
       0     0     0     0     1     0;
       0     0     0     0     0     1;
```

```
       0     0  -gamma -d/m    0     0;
       0     0     0    0   -d/m    0;
       0     0 -m*g*l/J  0     0     0 ];

B = [  0     0;
       0     0;
       0     0;
      1/m    0;
       0    1/m;
      r/J    0    ];

C = [  1     0    0     0     0     0;
       0     1    0     0     0     0 ];

D = [  0     0;   0     0];

%%
%% Construct inputs and outputs corresponding to steps in xy position
%%
%% The vectors xd and yd correspond to the states that are the desired
%% equilibrium states for the system.  The matrices Cx and Cy are the
%% corresponding outputs.
%%
%% The way these vectors are used is to compute the closed loop system
%% dynamics as
%%
%% xdot = Ax + B u =>xdot = (A-BK)x + K xd
%%         u = -K(x - xd)    y = Cx
%%
%% The closed loop dynamics can be simulated using the "step" command,
%% with K*xd as the input vector (assumes that the "input" is unit size,
%% so that xd corresponds to the desired steady state.
%%

xd = [1; 0; 0; 0; 0; 0];  Cx = [1 0 0 0 0 0];
yd = [0; 1; 0; 0; 0; 0];  Cy = [0 1 0 0 0 0];

%%
%% LQR design
%%

% Start with a diagonal weighting
Q1 = diag([1, 1, 1, 1, 1, 1]);
R1a = 0.1 * diag([1, 1]);
K1a = lqr(A, B, Q1, R1a);
```

```
% Close the loop: xdot = Ax + B K (x-xd)
H1ax = ss(A-B*K1a,B(:,1)*K1a(1,:)*xd,Cx,0);
H1ay = ss(A-B*K1a,B(:,2)*K1a(2,:)*yd,Cy,0);
figure(1); step(H1ax, H1ay, 10);
legend('x', 'y');

% Look at different input weightings
R1b = diag([1, 1]); K1b = lqr(A, B, Q1, R1b);
H1bx = ss(A-B*K1b,B(:,1)*K1b(1,:)*xd,Cx,0);

R1c = diag([10, 10]); K1c = lqr(A, B, Q1, R1c);
H1cx = ss(A-B*K1c,B(:,1)*K1c(1,:)*xd,Cx,0);

figure(2); step(H1ax, H1bx, H1cx, 10);
legend('rho = 0.1', 'rho = 1', 'rho = 10');

% Output weighting
Q2 = [Cx; Cy]' * [Cx; Cy];
R2 = 0.1 * diag([1, 1]);
K2 = lqr(A, B, Q2, R2);

H2x = ss(A-B*K2,B(:,1)*K2(1,:)*xd,Cx,0);
H2y = ss(A-B*K2,B(:,2)*K2(2,:)*yd,Cy,0);
figure(3); step(H2x, H2y, 10);
legend('x', 'y');

%%
%% Physically motivated weighting
%%
%% Shoot for 1 cm error in x, 10 cm error in y.  Try to keep the angle
%% less than 5 degrees in making the adjustments.  Penalize side forces
%% due to loss in efficiency.
%%

Q3 = diag([100, 10, 2*pi/5, 0, 0, 0]);
R3 = 0.1 * diag([1, 10]);
K3 = lqr(A, B, Q3, R3);

H3x = ss(A-B*K3,B(:,1)*K3(1,:)*xd,Cx,0);
H3y = ss(A-B*K3,B(:,2)*K3(2,:)*yd,Cy,0);
figure(4); step(H3x, H3y, 10);
legend('x', 'y');

%%
```

```
%% Velocity control
%%
%% In this example, we modify the system so that we control the
%% velocity of the system in the x direction.  We ignore the
%% dynamics in the vertical (y) direction.  These dynamics demonstrate
%% the role of the feedforward system since the equilibrium point
%% corresponding to vd neq 0 requires a nonzero input.
%%
%% For this example, we use a control law u = -K(x-xd) + ud and convert
%% this to the form u = -K x + N r, where r is the reference input and
%% N is computed as described in class.
%%

% Extract system dynamics: theta, xdot, thdot
Av = A([3 4 6], [3 4 6]);
Bv = B([3 4 6], 1);
Cv = [0 1 0]; % choose vx as output
Dv = 0;

% Design the feedback term using LQR
Qv = diag([2*pi/5, 10, 0]);
Rv = 0.1;
Kv = lqr(Av, Bv, Qv, Rv);

% Design the feedforward term by solve for eq pt in terms of reference r
T = [Av Bv; Cv Dv]; % system matrix
Nxu = T \ [0; 0; 0; 1]; % compute [Nx; Nu]
Nx = Nxu(1:3); Nu = Nxu(4); % extract Nx and Nu
N = Nu + Kv*Nx; % compute feedforward term

%%
%% Design #1: no feedforward input, ud
%%

Nv1 = [0; 1; 0];
Hv1 = ss(Av-Bv*Kv, Bv*Kv*Nx, Cv, 0);
step(Hv1, 10);

%%
%% Design #2: compute feedforward gain corresponding to equilibrium point
%%

Hv2 = ss(Av-Bv*Kv, Bv*N, Cv, 0);
step(Hv2, 10);
```

```
%%
%% Design #3: integral action
%%
%% Add a new state to the system that is given by xidot = v - vd.  We
%% construct the control law by computing an LQR gain for the augmented
%% system.
%%

Ai = [Av, [0; 0; 0]; [Cv, 0]];
Bi = [Bv; 0];
Ci = [Cv, 0];
Di = Dv;

% Design the feedback term, including weight on integrator error
Qi = diag([2*pi/5, 10, 0, 10]);
Ri = 0.1;
Ki = lqr(Ai, Bi, Qi, Ri);

% Desired state (augmented)
xid = [0; 1; 0; 0];

% Construct the closed loop system (including integrator)
Hi = ss(Ai-Bi*Ki,Bi*Ki*xid - [0; 0; 0; Ci*xid],Ci,0);
step(Hi, 10);
```