Goals:
• Derive the linear quadratic regulator and demonstrate its use

Reading:
• Friedland, Chapter 9 (different derivation, but same result)
• RMM course notes (available on web page)
• Lewis and Syrmos, Section 3.3

Homework #2
• Design LQR controllers for some representative systems
• Due Wed, 18 Jan by 5 pm, in box outside 109 Steele
Review from last lecture

Trajectory Generation

Controller

Process

Estimator

Trajectory Generation via Optimal Control:

\[
\begin{align*}
\dot{x} &= f(x, u) \quad x = \mathbb{R}^n \\
x(0) \text{ given} &\quad u \in \Omega \subset \mathbb{R}^p \\
J &= \int_0^T L(x, u) \, dt + V(x(T)) \\
\psi(x(T)) &= 0
\end{align*}
\]

Today: focus on special case of a linear quadratic regulator

\[
\begin{align*}
\dot{x} &= Ax + Bu \quad x = \mathbb{R}^n \\
x(0) \text{ given} &\quad u \in \mathbb{R}^p \\
J &= \int_0^T x^T Q x + u^T R u \, dt + x(T)^T P_1 x(T) \\
\text{no terminal constraints}
\end{align*}
\]
Linear Quadratic Regulator (finite time)

Problem Statement

\[ \dot{x} = Ax + Bu \quad x = \mathbb{R}^n \]
\[ x(0) \text{ given} \quad u \in \mathbb{R}^p \]

\[ J = \frac{1}{2} \int_0^T (x^T Q x + u^T R u) \, dt + \frac{1}{2} x^T(T) P_1 x(T) \]

- Factor of 1/2 simplifies some math below; optimality is not affected

Solution: use the maximum principle

\[ H = x^T Q x + u^T R u + \lambda^T (Ax + Bu) \]

\[ \dot{x} = \left( \frac{\partial H}{\partial \lambda} \right)^T = Ax + Bu \quad x(0) = x_0 \]

\[ -\dot{\lambda} = \left( \frac{\partial H}{\partial x} \right)^T = Q x + A^T \lambda \quad \lambda(T) = P_1 x(T) \]

\[ 0 = \frac{\partial H}{\partial u} = Ru + \lambda^T B \quad \implies \quad u = -R^{-1} B^T \lambda. \]

- This is still a two point boundary value problem \( \Rightarrow \) hard to solve
- Note that solution is linear in \( x \) (because \( \lambda \) is linear in \( x \), treated as an input)
Simplified Form of the Solution

Can simplify solution by guessing that $\lambda = P(t) x(t)$

\[-\dot{\lambda} = \left( \frac{\partial H}{\partial x} \right)^T = Qx + A^T \lambda \quad \lambda(T) = P_1 x(T)\]

\[
\begin{align*}
\dot{\lambda} &= \dot{P} x + P \dot{x} = \dot{P} x + P (Ax - BR^{-1}B^T P)x \\
&\Downarrow \\
-\dot{P} x - PAx + PBR^{-1}BPx &= Qx + A^T Px.
\end{align*}
\]

Solution exists if we can find $P(t)$ satisfying

\[-\dot{P} = PA + A^T P - PBR^{-1}B^T P + Q \quad P(T) = P_1\]

• This equation is called the *Riccati ODE*; matrix differential equation
• Can solve for $P(t)$ backwards in time and then apply $u(t) = -R^{-1} B P(t) x$
• Solving $x(t)$ forward in time gives optimal state (and input): $x^*(t), u^*(t)$
• Note that $P(t)$ can be computed once (ahead of time) $\Rightarrow$ allows us to find the optimal trajectory from different points just by re-integrating state equation with optimal input
Finite Time LQR Summary

Problem: find trajectory that minimizes

\[ \dot{x} = Ax + Bu \quad x = \mathbb{R}^n \]

\[ x(0) \text{ given} \quad u \in \mathbb{R}^p \]

\[ J = \frac{1}{2} \int_0^T \left( x^T Q x + u^T R u \right) dt + \frac{1}{2} x^T(T) P_1 x(T) \]

Solution: time-varying linear feedback

\[ u(t) = -R^{-1} B P(t)x. \]

\[ -\dot{P} = PA + A^T P - PB R^{-1} B^T P + Q \quad P(T) = P_1 \]

- Note: this is in feedback form ⇒ can actually eliminate the controller (!)
Infinite Time LQR

Extend horizon to $T = \infty$ and eliminate terminal constraint:

\[ \dot{x} = Ax + Bu \quad x = \mathbb{R}^n \]

$x(0)$ given \quad \quad $u \in \mathbb{R}^p$

\[ J = \int_0^\infty (x^T Q x + u^T R u) \, dt \]

Solution: same form, but can show $P$ is constant

\[ u = K x \quad K = -R^{-1} B^T P \]

\[ 0 = PA + A^T P - PBR^{-1}B^T P + Q \]

Remarks

- In MATLAB, $K = \text{lqr}(A, B, Q, R)$
- Require $R > 0$ but $Q \geq 0$ + must satisfy “observability” condition
- Alternative form: minimize “output” $y = H x$

\[ L = \int_0^\infty x^T H^T H x + u^T R u \, dt = \int_0^\infty \| H x \|^2 + u^T R u \, dt \]

- Require that $(A, H)$ is observable. Intuition: if not, dynamics may not affect cost $\Rightarrow$ ill-posed. We will study this in more detail when we cover observers
Applying LQR Control

Application #1: trajectory generation
- Solve for \((x_d, y_d)\) that minimize quadratic cost over finite horizon (requires linear process)
- Use local controller to regulate to desired trajectory

Application #2: trajectory tracking
- Solve LQR problem to stabilize the system to the origin \(\Rightarrow\) feedback \(u = Kx\)
- Can use this for local stabilization of \(\text{any}\) desired trajectory
- Missing: so far, have assumed we want to keep \(x\) small (versus \(x \to x_d\))
LQR for trajectory tracking

Goal: design local controller to track \( x_d \):

Approach: regulate the error dynamics

- Let \( e = x - x_d \), \( v = u - u_d \) and assume \( f(x, u) = f(x) + g(x)u \) (simplifies notation)

\[
\dot{e} = \dot{x} - \dot{x}_d = f(x) + g(x)u - f(x_d) + g(x_d)u_d \\
= f(e + x_d) - f(x_d) + g(e + x_d)(v + u_d) - g(x_d)u_d \\
= F(e, v, x_d(t), u_d(t))
\]

- Now linearize the dynamics around \( e = 0 \) and design controller \( v = Ke \)
- Final control law will be \( u = K(x - x_d) + u_d \)
- Note: in general, linearization will depend on \( x_d \Rightarrow u = K(x_d)x \) ← “gain scheduling”
Choosing LQR weights

Most common case: diagonal weights

\[ Q = \begin{bmatrix} q_1 & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & q_n \end{bmatrix} \quad R = \rho \begin{bmatrix} r_1 & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & r_n \end{bmatrix} \]

- Weight each state/input according to how much it contributes to cost
- Eg: if error in \( x_1 \) is 10x as bad as error in \( x_2 \), then \( q_1 = 10 \cdot q_2 \)
- OK to set some state weights to zero, but all input weights must be > 0
- Remember to take units into account: eg for ducted fan if position error is in meters and pitch error is in radians, weights will have different “units”

Remarks

- LQR will \textit{always} give a stabilizing controller, but no guaranteed margins
- LQR shifts design problem from loop shaping to weight choices
- Most practical design uses LQR as a first cut, and then tune based on system performance
Example: Ducted Fan

Stabilization:
- Given an equilibrium position \((x_d, y_d)\) and equilibrium thrust \(f_{2d}\), maintain stable hover
- Full state available for feedback

Tracking:
- Given a reference trajectory \((x_r(t), y_r(t))\), find a feasible trajectory \(\vec{x}_d, u_d\) and a controller \(u = \alpha(x, x_d, u_d)\) such that \(x \to x_d\)

Equations of motion

\[
\begin{align*}
m\ddot{x} &= f_1 \cos \theta - f_2 \sin \theta - c_d, x(\theta, \dot{x}) \\
m\ddot{y} &= f_1 \sin \theta + f_2 \cos \theta - mg - c_d, y(\theta, \dot{y}) \\
J\ddot{\theta} &= rf_1 - mg \sin \theta - c_d, \theta(\theta, \dot{\theta})
\end{align*}
\]

LQR design: see lqr_dfan.m (available on course web page)
Variation: Integral Action

Limitation in LQR control: perfect tracking requires perfect model
- Control law is \( u = K (x - x_d) + u_d \Rightarrow u_d \) must be perfect to hold \( e = 0 \)
- Alternative: use integral feedback to give zero steady state error

\[
\frac{d}{dt} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ y - r \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ Cx - r \end{bmatrix} \quad \text{integral of (output) error}
\]

- Now design LQR controller for extended system (including integrator weight)

\[
u = K(x - x_d) - K_i z + u_d
\]

equilibrium value \( \Rightarrow y = r \Rightarrow 0 \) steady state error
Example: Cruise Control

\[ m \ddot{v} = F - F_d \]

Linearized around \( v_0 \):

\[ \ddot{v} = a \ddot{v} - mg\theta + mb \ddot{u} \]

\[ F = \alpha_n u T(\alpha_n v) \]

\[ F_d = mgC_r + \frac{1}{2} \rho C_v Av^2 + mg\theta, \quad y = v = \ddot{v} + v_0 \]

Step 1: augment linearized (error) dynamics with integrator

\[
\frac{d}{dt} \begin{bmatrix} \ddot{v} \\ z \end{bmatrix} = \begin{bmatrix} \frac{a}{m} & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{v} \\ z \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix} u + \begin{bmatrix} -g \\ 0 \end{bmatrix} \theta + \begin{bmatrix} 0 \\ r - v_0 \end{bmatrix}
\]

Step 2: choose LQR weights and compute LQR gains

\[
Q = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix} \quad R = \rho \quad \rightarrow \quad K = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}
\]

Note: linearized about \( v_0 \) but try to maintain speed \( r \) (near \( v_0 \))

Step 3: implement controller

\[
\dot{z} = v - r
\]

\[
u = u_0 + k_1 (v - r) + k_2 z \quad \text{PI controller}
\]
Application #1: trajectory generation
- Solve for \((x_d, y_d)\) that minimize quadratic cost over finite horizon
- Use local controller to track trajectory

Application #2: trajectory tracking
- Solve LQR problem to stabilize the system
- Solve algebraic Riccati equation to get state gain
- Can augment to track trajectory; integral action

\[ J = \frac{1}{2} \int_0^T (x^T Q x + u^T R u) \, dt \]
\[ + \frac{1}{2} x^T(T) P_1 x(T) \]

\[ J = \int_0^\infty (x^T Q x + u^T R u) \, dt \]
Announcements

Mailing list
• If you didn’t get e-mail about TA office hours, send email to murray@cds

Late homework policy
• No late homework without prior permission
• Usually willing to give a few extra days the first time you ask
• Sickness, conferences and other unavoidable conflicts usually work

Lecture recordings
• Will be posting audio recordings of lectures (along with slides) on web site