10. The Linear Quadratic Regulator

• The Algebraic Riccati equation
• Infinite-horizon problems
• Time-varying systems and tracking problems
• The steady-state regulator
• Example: force on mass
• Summary of LQR solution via DP
• The Riccati recursion
• Solving the Hamilton-Jacobi equation
• Example: path optimization
• Dynamic programming
• Constrainted optimization formulation
• The LQR problem formulation
• Regulation and the Least squares formulation of regulation
The Key Points of This Section

- The idea of regulation: keep the output small, using as little input as possible.
- Multi-objective problem: allows trade-off to be made between input effort and regulation.
- Idea of regulation: keep the output small, using as little input as possible.
- Controller is linear state feedback $u(t) = K_0 x(t)$.
- Solution is Riccati recursion; much faster to compute.
- Instead of solve it via dynamic programming, we often use the steady-state solution; to find it, solve the Algebraic Riccati Equation:

$$\dot{X}(t) = X(t) P + P X(t)^T$$

where $P$ is the steady-state solution of the Algebraic Riccati Equation. The controller is then given by

$$u(t) = -K_0 x(t)$$

This solution is much faster to compute than solving the full Riccati recursion.
\[ x(t+1) = Ax(t) + Bu(t) \]

\[ x(0) = x_0 \]

\[ y(t) = Cx(t) \]

**Regulation**

- **Objective**: Keep small over \( t = 0, \ldots, N-1 \).
- **Objective**: Using low input effort.

\[ J_{\text{out}} = \sum_{t=0}^{N-1} \| y(t) \|_2^2 \]

\[ J_{\text{in}} = \sum_{t=0}^{N-1} \| u(t) \|_2^2 \]

**Multiobjective problem**

\[ (t)x_0 + (t)\hat{\theta} \]

\[ (t)nB + (t)xV = (I + t)x \]

**Regulation**
Least-squares solution is open-loop: does not use measurements of \( x(t) \) on \( t = 0, \cdots, N \).

\[
\begin{bmatrix}
0 \\
0 x W
\end{bmatrix} + \text{bias}_n \begin{bmatrix}
I n \wedge T
\end{bmatrix} = 0 x W + \text{bias}_n T = \begin{bmatrix}
\text{bias}_n \| n \| \\
\text{bias}_n \| n \|
\end{bmatrix} = (\text{bias}_n)_\text{in} J n + (\text{bias}_n)_\text{out} J
\]

Multiobjective least squares problem:

\[
0 x W + \text{bias}_n T =
\]

as before we have

\[
\begin{bmatrix}
0 \\
0 x W
\end{bmatrix} + \text{bias}_n \begin{bmatrix}
I n \wedge T
\end{bmatrix} = 0 x W + \text{bias}_n T = \begin{bmatrix}
\text{bias}_n \| n \| \\
\text{bias}_n \| n \|
\end{bmatrix} = (\text{bias}_n)_\text{in} J n + (\text{bias}_n)_\text{out} J
\]

Least-squares formulation
The Linear Quadratic Regulator

\[ J(u_{\text{seq}}) = J_{\text{out}}(u_{\text{seq}}) + \mu J_{\text{in}}(u_{\text{seq}}) = \sum_{t=0}^{N-1} \| y(t) \|^2 + \mu \| u(t) \|^2 \]

where

\[
(N)x^T \hat{Q}_L(N)x + \left( (\mu) n^T \hat{H}_L(n) + (\mu) x^T \hat{O}_L(x) \right) = \sum_{t=0}^{0} = (b_{\text{eq}} n) f
\]

we'll use the slightly more general cost function

\[
(\mu) n^T \hat{H}_L(n) + (\mu) x^T \hat{O}_L(x) = \sum_{t=0}^{0} = (b_{\text{eq}} n) f
\]

\[
\sum_{t=0}^{0} = (b_{\text{eq}} n) f
\]

\[
\sum_{t=0}^{0} = (b_{\text{eq}} n) f
\]

are called state cost, final state cost, and input cost matrices.

0 < H \quad 0 \geq f \hat{O} \quad 0 < \hat{O}
The Linear Quadratic Regulator (LQR) problem:

\[
J(u_{seq}) = \sum_{t=0}^{N-1} (x(t)^T Q x(t) + u(t)^T R u(t)) + x(N)^T Q_f x(N)
\]

- \( Q_f \) is called the time horizon.
- First term measures state deviation.
- Second term measures input size or actuator authority.
- Last term measures final state deviation.
- We often use \( Q = Q_f = C^T C \) and \( R = \mu I \) for relative weights of state deviation and input usage.
- \( R > 0 \) means any (nonzero) input adds to the cost.

\[
\ell = R \text{ and } \ell L \ell = \ell = \ell
\]

\[
\ell L \ell = R \text{ and } \ell L = \ell = \ell
\]

\[
J = \sum_{t=0}^{N-1} (x(t)^T Q x(t) + u(t)^T R u(t)) + x(N)^T Q_f x(N)
\]

Cost Function:

\[
\sum_{t=0}^{N-1} (x(t)^T Q x(t) + u(t)^T R u(t)) + x(N)^T Q_f x(N)
\]
Constrained Optimization

The LQR problem can be written as the constrained optimization

$$\begin{array}{l}
\text{minimize} \quad N \sum_{t=0}^{N-1} \left( x(t)^T Q x(t) + u(t)^T R u(t) \right) + x(N)^T Q f x(N) \\
\text{subject to} \quad x(0) = x_0 \\
\{ x(t+1) = A x(t) + B u(t) \} \quad \text{for} \quad t = 0, \ldots, N-1 \\
\text{the variables are} \quad u(0), \ldots, u(N-1), x(0), \ldots, x(N) \\
\text{the dynamics are} \quad \{ I - N \} x + (I - N) n = (I + n) x \\
\text{subject to} \quad 0 x = (0) x \\
\text{minimize} \quad \left( N x^T \bar{O}_L(N) x + \left( (I - N) n B + (I - N) x A \right) \right) \underbrace{0}_{1 - N} = \underbrace{n}_{0} x
\end{array}$$

Constrained Optimization

The LQR problem can be written as the constrained optimization.
Dynamic Programming Solution

• Gives an efficient, recursive method to solve LQR problem

\[
V_t(z) = \min_{u(t), \ldots, u(N-1)} \sum_{\tau=t}^{N-1} (x(\tau)^T Q x(\tau) + u(\tau)^T R u(\tau)) + x(N)^T Q f x(N)
\]

subject to

\[
\begin{align*}
&x(t+1) = Ax(t) + Bu(\tau) \\
&x(1) = z
\end{align*}
\]

For \( t = 0, \ldots, N \), define the value function \( V_t(z) \) by

\( (\perp)nB + (\perp)xAV = (1 + \perp)x \) and \( z = (\perp)x \)

• Gives the min-cost-to-go \( (z)^0V \)

• \( (z)^0V \) is the min LQR cost

• Gives a feedback solution

• Useful and important idea on its own
We'll find that

\[ V_t(z) = z^T P_t z, \]

where \( P_t \geq 0 \). The optimal cost-to-go with no time left is just final state cost

\[ f \mathcal{O} = N P \]

\[ f \mathcal{O}_t \mathcal{O} = (z)^N \Lambda \]

can be found recursively, working backward from \( t = N \).

\[ N = f \]

\[ 0 < \frac{1}{P} \] and \[ \frac{1}{P} = \frac{1}{P} \]

where \( \frac{1}{P} \) is quadratic, i.e.,

\[ \Lambda \]

Therefore, we'll find that

Dynamic Programming
The Linear Quadratic Regulator

S. Lall, Stanford 2007.11.06.01

dynamic programming principle

\[
(\mathbf{m} \mathbf{B} + \mathbf{z} \mathbf{A})^{1+i} \Lambda + m \mathbf{R}_L m + z \mathbf{D}_L z \right)^m \min = (z)^i \Lambda
\]

dynamic programming principle

where \( \mathbf{I} + \mathbf{q} \mathbf{x} \)  

where we land, i.e., 

current cost incurred (through (1) \( \mathbf{n} \mathbf{R} \mathbf{H}_L (\mathbf{q}) \mathbf{n} \)) 

choice of \( \mathbf{n} \mathbf{R} \mathbf{H}_L (\mathbf{q}) \mathbf{n} \) affects \( \mathbf{q} \) 

now suppose we know \( (z)^i \mathbf{A} \)
Let's look at this again!

**Dynamic Programming Principle**

\[
V_t(z) = \min_w w^T Q z + w^T R w + V_{t+1}(A z + B w)
\]

In words, a function of \( w_1, \ldots, w_k \) is minimized in any order:

\[
\min_{w_1, \ldots, w_k} f(w_1, \ldots, w_k) = \left( \min_{w_1} \min_{w_2, \ldots, w_k} f(w_1, \ldots, w_k) \right)
\]

\[
(m B + z A)^{t+1} \Lambda + m H_L m + z \hat{O}_L z
\]

\[
\min_w = (z)^{t+1/2} \Lambda
\]

current cost incurred + min-cost-to-go from where you land

\[
\min_{w_1, \ldots, w_k} f(w_1, \ldots, w_k) = \min_{w_1, \ldots, w_k} f(w_1, \ldots, w_k)
\]

follows from fact that we can minimize in any order

\[
m = (t)^n m \text{ is cost incurred at time } t \text{ if you land at } t + 1\]

\[
m H_L m + z \hat{O}_L z
\]
Example: Path Optimization

- You can fly from point $i$ to point $j$ at cost $c_{ij}$.
- Want to find min cost route or path from Seattle to Boston.

Seattle ➔ San Francisco ➔ Los Angeles ➔ Denver ➔ Chicago ➔ Miami
The Linear Quadratic Regulator

\[ (\ell \Lambda + \ell_1 V)_{\text{adjacent}} \min_{j} = (\ell) \Lambda + \ell_1 V \]

From point \( \ell \) to \( \ell \) where

\[ (\ell_1 V)_{\text{adjacent}} \min_{j} = \ell_1 \Lambda \]

dynamic programming principle

Boston

If we know \( (\ell) \Lambda \) for all \( \ell \), then we can find min cost route from any point to Boston.

value function

dynamic programming approach
Hamilton-Jacobi equation

\[ V_t(z) = \min_w (z^T Q z + w^T R w + V_{t+1}(A z + B w)) \]

- called DP, Bellman, or Hamilton-Jacobi equation
- gives \( \min \) any minimizing \( w \)
- recursively, in terms of \( V_{t+1} \)
- DP has many applications beyond LQR, e.g.,
  - optimal flow control in communication networks
  - optimal control in communication networks
  - optimization in finance

\[ \left( (mB + zA)^{t+1} + mH_n m + zO_z \right)^m = (z)^t \Lambda \]
\[ V^N P L \left( B L P L B + R \right) - = 1^{do} m \]

hence optimal \( m \) is

\[ 0 = B^N P L (m B + z \Lambda) z + R \Lambda z \]

solve by setting derivative w.r.t. \( m \) to zero

\[
\left( (m B + z \Lambda)^N P L (m B + z \Lambda) + m R L m \right)^m \min + z \Theta L z = (z)^{1-N} \Lambda
\]

by DP

\[ \int \Theta = N P \text{ where } z^N P L z = (z)^N \Lambda \text{ we know} \]

solving the Hamilton-Jacobi equation
solving the Hamilton-Jacobi equation

\[ V_{\text{N}} = \left( z^T Q z + w^T R w \right)^{\text{opt}} + (Az + Bw_{\text{opt}})^T P_N (Az + Bw_{\text{opt}}) \]

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

we conclude that \( V_{\text{N}} \) is quadratic:

\[ V_{\text{N}} - 1(z) = (z)^{1-N}\Lambda \]

(after some ugly algebra)

\[ z(1-N)B_L z = (z)^{1-N}\Lambda \]

\[ \left( \begin{array}{c} 1^d \text{d}m_B + zV \end{array} \right)^{N} B_L \left( \begin{array}{c} 1^d \text{d}m_B + zV \end{array} \right)^{1^d} m + zO_L z = (z)^{1-N}\Lambda \]

and so

solving the Hamilton-Jacobi equation
\[(t)xAV^{t+\frac{1}{2}}B_{t-1} (B^{t+\frac{1}{2}}A_{t-1} + R) = (t)^{\text{opt}}\]

called Riccati recursion for \(P_t\); then the optimal control input is

\[
V_t \cdot \frac{\partial}{\partial t} P_t - (B^{t+\frac{1}{2}}A_{t-1} + R)B^{t+\frac{1}{2}}A_{t-1}V - A^{t+\frac{1}{2}}B^{t+\frac{1}{2}}V + \hat{O} = P_t^{-1} P_t^{-1}
\]

where

\[
z^{t-\frac{1}{2}} B_{t-1}z = (z)^{t-\frac{1}{2}} A^{t-\frac{1}{2}}
\]

\[
z^{t-1} B_{t-1}z = (z)^{t-1} A^{t-1}
\]

once we know \(A\), \(z\) is quadratic, we find that \(A^{t-1}\) is as well, this recursion works for all \(t\)
Summary of LQR Solution via DP

1. set $P_N = Q$

2. for $t = N-1, \ldots, 0$
   - compute $P_t$ from $P_{t+1}$ via
     \[ P_t = P_{t+1} - A P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A \]

3. define $K_t = - (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$

4. optimal $u$ is given by $u(t) = K_t x(t)$
properties of LQR solution

- Optimal \( u \) is a linear function of the state (called linear state feedback).
- Recursion for min cost-to-go runs backward in time.
- Solves least squares problem in \( m \) variables much faster than direct least squares method.
- At run-time, we need only know \( x \); we don’t need to know \( x(0) \).
- \( \{K_0, \ldots, K_{N-1}\} \) computed in advance.

\[
\begin{align*}
(\mathcal{Q}) x^{t} Y & = (\mathcal{Q}) n \\
(\mathcal{Q}) n B + (\mathcal{Q}) x V & = (1 + \mathcal{Q}) x
\end{align*}
\]
\[ I^n = R \quad \text{and} \quad f_0C = f_0 = 0 \]

Corresponds to

\[
(N) f(n) + \left( \begin{array}{c} (t) n \end{array} + (t) n \right) \sum_{t=0}^{T} = \left( \begin{array}{c} b \end{array} \right) \in \left( \begin{array}{c} b \end{array} \right) + \left( \begin{array}{c} b \end{array} \right) \in \left( \begin{array}{c} b \end{array} \right)
\]

Cost function

\[
(t) x \begin{bmatrix} 0 & 1 \end{bmatrix} = (t) f
\]

\[
(t) n \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + (t) x \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = (1 + t) x
\]

Newton's law discretized, with sample period \( n \) gives

\[ 1 \]

Example: Force on mass
example: states and inputs

for $\mu = 1$, the controller uses less control effort.

for $\mu = 10$, the controller uses more control effort.

Example: States and Inputs
Example: trade-off curve
Example: Gains for various $Qf$; note convergence.

$I_{100} = fQ$

$I = fQ$

$0 = fQ$

State feedback gains for various $Qf$; note convergence.
steady-state regulator usually converges as $t$ decreases below $N$ limit or steady-state value $P_{ss}$ satisfies

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

which is called the (discrete-time) Algebraic Riccati Equation (ARE)

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

state feedback

for $t$ not close to the horizon, LQR optimal control is approximately a linear, constant state feedback

$$u(t) = K_{ss} x(t)$$

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

very widely used in practice

$P_{ss}$ can be found by iterating the Riccati recursion, or by direct methods

which is called the (discrete-time) Algebraic Riccati Equation (ARE)

limit of steady-state value $P_{ss}$ satisfies usually $P_{ss}$ rapidly converges as $t$ decreases below $N$

steady-state regulator
LQR is readily extended to handle time-varying systems.

\[
L(t)x(t) + n(t)B + x(t)V = (I + t)x
\]

and time-varying cost matrices

\[
J = \sum_{t=0}^{T-1} x(t)^T Q(t)x(t) + u(t)^T R(t)u(t)
\]

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

\[
((L(t)\hat{\mathcal{O}}_L(t) - \mathcal{O}_L(t)L(t))x + \left((n(t)B + x(t)V)\mathcal{O}_L(t)x\right)
\]
Tracking Problems

we consider LQR cost with state and input offsets. (we drop the final state term for simplicity)

\[ \big( (t)n - (t)n \big) L \big( (t)n - (t)n \big) \sum_{t=0}^{\tau} + \big( (t)x - (t)x \big) L \big( (t)x - (t)x \big) \sum_{t=1}^{\tau-1} = f \]

we consider LQR cost with state and input offsets

Trackng Problems
we have an infinite number of variables

\[ 0 < R \quad 0 < \mathcal{Q} \]

with given constant state and input weighting matrices

\[
\left( (t) n R L(t) n + (t) x \mathcal{Q} L(t) x \right) \sum_{t=0}^{\infty} = J
\]

Infinite-horizon LQR problem: choose \( \{0\} n \); \( \{1\} n \) to minimize

\[ 0 x = (0) x \quad (t) n B + (t) x A = (1 + t) x \]

Infinite Horizon LQR Problem
The finiteness of the cost problem: it's possible that $J = \infty$ for all possible input sequences $u(0), u(1), \ldots$ for example:

$$x(t+1) = 2x(t) + 0u(t)$$

This cannot happen if $(A, B)$ is controllable; then for any $x_0$ there is an input sequence $u = t(I - u)n, \ldots, (I)n, (0)n$ that steers $x$ to zero at $t$ and keeps it there:

$$I = (0)x, \quad (t)n0 + (t)xz = (I + t)x$$

For example:

$$\cdots (I)n, (0)n$$

Problem: It's possible that $\infty = \int$ for all possible input sequences $u$, finiteness of the cost.
Define the value function $V$: $\mathbb{R}^n \rightarrow \mathbb{R}$

$V(z) = \min_{u(0), u(1), ...} \sum_{t=0}^{\infty} (x(t) \mathbf{Q} x(t) + u(t) \mathbf{R} u(t))$

subject to

$x(0) = z, x(t+1) = \mathbf{A} x(t) + \mathbf{B} u(t)$ for all $t \geq 0$

$\min_{z \in \mathbb{R}} z = \infty$ is the min-cost-to-go, starting from state $z$

$\Lambda$ is the min-cost-to-go, which is always $\infty$, infinite horizon problem is shift-invariant.
\[ V(z) = \min_w \left( w^T Q z + w^T R w + V(Az + Bw) \right) \]

and the minimizing \( w \) is

\[ \left( (MB + zA) L (MB + zA) + mR_L m + z \hat{O}_L z \right)^m_{\text{min}} = z L z \]

(can be argued directly from first principles)

Fact: \( \Lambda \) is quadratic, i.e.,

\[ 0 \preceq P \text{ and } P = P^T \text{ where } z L z = (z) \Lambda \]

The HJ equation is

\[ \text{Hamilton-Jacobi equation} \]
The Linear Quadratic Regulator

Riccati equation

\[ z^T P z = z^T Q z + w^T \text{opt} R w \text{opt} + (Az + Bw \text{opt})^T P (Az + Bw \text{opt}) = z^T (Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A) z \]

this must hold for all \( z \), so \( P \) satisfies the ARE

\[ AV_{P_L B_{L-1}} (B_{P_L B_{L}} + R) B_{P_L V_{L}} V_{P_L V_{L}} - AV_{P_L V_{L}} + \tilde{O} = 0 \]

compared to finite-horizon LQR problem

- value function and optimal state feedback gains are time-invariant
- we don't have a recursion to compute \( P \); we only have the ARE

\[ (1^{do} m B + z V)_{P_L} (1^{do} m B + z V) + 1^{do} m R 1^{do} m + z \tilde{O}_{L} z = z P_{L} z \]

the HJ equation is

Riccati equation
so infinite-horizon LQR is the same as steady-state finite-horizon LQR

$$P_k - 1 = Q + A^T P_k A - A^T P_k B (R + B^T P_k B)^{-1} B^T P_k A$$

converges to $P$ as $k \to -\infty$, from any initial condition $P_0$.

the Riccati recursion

the ARE has exactly one positive semidefinite solution $P$ if $A, B$ is controllable and $A, C$ is observable (recall $C = CT$).

Solving the ARE
1. solve the ARE

\[ P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A \]

in Matlab, can use `dlqr`

2. set \( K \) to

\[ K = -(R + B^T P B)^{-1} B^T P A \]

and use state-feedback controller

\[ u(t) = K x(t) \]
If \( A, B \) is controllable and \( A, C \) is observable, then the closed-loop system is stable.

If \( A, B \) is controllable, \( A, C \) is observable, then the closed-loop system is stable.

Optimal control is
\[
\begin{align*}
I &= \mathcal{H} \quad 0 = \mathcal{O} \\
(\dot{t})n + (\dot{t})xZ &= (I + \dot{t})x
\end{align*}
\]

Is closed-loop system stable? Look at the example:

\[
(\dot{t})x(YB + V) = (I + \dot{t})x
\]

We have the controller
\[
(\dot{t})nB + (\dot{t})xV = (I + \dot{t})x
\]

The closed-loop system