Linear Quadratic Regulator (LQR)

- Remember Gains: $K_p$ and $K_d$
- LQR is an automated method for choosing \textbf{OPTIMAL} gains
- Optimal with respect to what?
  - Some (quadratic) function of state (e.g. Minimize distance to goal)
  - Some (quadratic) function of control input (e.g. Minimize torque)
  - A relative weighting of both
LQR Basis

- Linear Dynamic System:
  \[
  \dot{x} = Ax(t) + Bu(t)
  \]

\[
\begin{bmatrix}
\dot{\theta} \\
\dot{\phi}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
0 & -\frac{1}{J}
\end{bmatrix}
\begin{bmatrix}
\theta \\
\phi
\end{bmatrix} + \begin{bmatrix}
0 \\
\frac{1}{J}
\end{bmatrix} \tau
\]

\[
J\dot{\theta} + b\dot{\phi} = \tau
\]

\[
\theta = -\frac{b}{J}\dot{\theta} + \frac{1}{J}\tau
\]

LQR Basis

- Linear Dynamic System:
  \[
  \dot{x} = Ax(t) + Bu(t)
  \]

- Quadratic Performance Criterion:
  \[
  C(x, u) = \frac{1}{2}x^TQx + \frac{1}{2}u^TRu
  \]

Symmetric Positive Definite (Typically Diagonal) Matrices

\[
Q = \begin{bmatrix}
q_0 & q_1 & \cdots \\
 & q_1 & \cdots \\
 & & \ddots & \ddots \\
 & & & q_n
\end{bmatrix}
\]

\[
R = \begin{bmatrix}
r_0 & r_1 & \cdots \\
 & r_1 & \cdots \\
 & & \ddots & \ddots \\
 & & & r_0
\end{bmatrix}
\]
**LQR Basis**

- Linear Dynamic System: \[ \dot{x} = Ax(t) + Bu(t) \]
- Quadratic Performance Criterion: \[ C(x, u) = \frac{1}{2} x^T Q x + \frac{1}{2} u^T R u \]
- Gain Matrix: \[ K = R^{-1} B^T P \] \[ u = K x = R^{-1} B^T P x \]
- Algebraic Riccati Equation: \[ P \text{ is a solution of ARE} \]

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

---

**Detour: Shortest Path Search**

Consider a digraph \( G = (V, A) \) with nonnegative arc distance \( c_{ij} \)

---

S. Joo (sungmoon.joo@cc.gatech.edu) 11/04/2014
**Detour: Shortest Path Search**

Naïve approach:
- Find all paths from s(start) to g(goal)
- Evaluate the length of each s ~ g path
- Find the minimum length
- The number of s ~ g paths increases exponentially.
  (Infeasible for large graph)

**Observation:**
- If the s ~ g shortest path passes by node p, the subpath (s, p) and (p, g) are shortest path from s to p and from p to g, respectively

\[
d^*(g) = \text{length of shortest path from s to g} \\
d^*(g) = \min_{i \in V \setminus g} \{d^*(i) + c_{ig}\}
\]
Detour: Shortest Path Search

Forward Dynamic Programming
- Let \( j \) be a node at stage \( k \), \( D_k(j) \) be the shortest distance from \( s \) to \( j \)

\[
D_k(j) = \min \{ D_{k-1}(i) + c_{ij} | i \in \text{stage } k - 1 \}
\]

\[ k = 0, 1, \ldots \]

Detour: Shortest Path Search

Backward Dynamic Programming
- Let \( D_k(j) \) be the shortest distance from node \( j \) at stage \( k \) to \( g \)

\[
D_k(j) = \min \{ D_{k+1}(i) + c_{ji} | i \in \text{stage } k + 1 \}
\]

\[ k = N, N - 1, \ldots \]
What is the ‘Value’ of a State?

MDP – Accumulated rewards, Want to maximize
Shortest distance problem – Cost-to-Go, Want to minimize

Value Iteration is Dynamic Programming

What would Bellman do?
Dynamic Programming

$$J^*(x_t, u_t) = \min_u [C(x_t, u_t) + J^*(x_{t+1}, u_{t+1})]$$

Optimal Policy is obtained!!
Dijkstra '59

Dijkstra's Algorithm is a Special Case of Dynamic Programming!
: DP + greedy search with Cost-to-Come
: a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs

*Dijkstra vs A*

S. Joo (sungmoon.joo@cc.gatech.edu) 11/04/2014 13

Back to LQR

- Linear Dynamic System: \( \dot{x} = Ax(t) + Bu(t) \)
- Quadratic Performance Criterion: \( C(x,u) = \frac{1}{2}x^TQx + \frac{1}{2}u^TRu \)
- Gain Matrix: \( K = R^{-1}B^TP \) \( u = Kx = R^{-1}B^TPx \)
- Algebraic Riccati Equation:

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

S. Joo (sungmoon.joo@cc.gatech.edu) 11/04/2014 14
Define a Value Function

- State Transitions: \( \dot{x} = Ax(t) + Bu(t) \)
- Costs: \( C(x_\tau, u_\tau) = \frac{1}{2} x^T(\tau) Q x(\tau) + \frac{1}{2} u^T(\tau) R u(\tau) \)
- Minimize Cost For All Future Time
  \[ J = \int_0^\infty C(x_\tau, u_\tau) d\tau \]
- Minimize Cost After Time = \( t \)
  \[ J(x_t, u_t) = \int_t^\infty C(x_\tau, u_\tau) d\tau \]

What would Bellman do?

Minimize Cost After Time = \( t \)
\[ J^*(x_t, u_t) = \min_u \int_t^\infty C(x_\tau, u_\tau) d\tau \]

Critical Step!

\[ J^*(x_t, u_t) = \min_u [C(x_t, u_t) \delta t + J^*(x_{t+\delta t}, u_{t+\delta t})] \]

S. Joo (sungmoon.joo@cc.gatech.edu) 11/04/2014
Substitute for Value in Right Hand Side

\[ J(x_t, u_t) = C(x_t, u_t) \delta t + J(x_{t+\delta t}, u_{t+\delta t}) \]

Assume \( V \) is smooth and does not depend explicitly on \( t \):

\[
J(x_{t+\delta t}, u_{t+\delta t}) = J(x_t, u_t) + \frac{\delta J}{\delta x} \delta x \delta t \\
= J(x_t, u_t) + \frac{\delta J}{\delta x} (Ax_t + Bu_t) \delta t
\]

\[ J(x_t, u_t) = C(x_t, u_t) \delta t + J(x_t, u_t) + \frac{\delta J}{\delta x} (Ax_t + Bu_t) \delta t \]

One More Fact/Assumption

Value function has the following form (\( P \) is symmetric, positive definite)

\[ J(x_t, u_t) = \frac{1}{2} x^T P x \]
One More Fact/Assumption

Value function has the following form ($P$ is symmetric, positive definite)

$$J(x_t, u_t) = \frac{1}{2} x^T P x$$

Why? It is an infinite sum of symmetric, positive definite matrices.

Linear Algebra Theorems:
- Sum of symmetric matrices = symmetric
- Sum of positive definite matrices = positive definite

Derivative with respect to $x$

Value function has the following form ($P$ is symmetric, positive definite)

$$J(x_t, u_t) = \frac{1}{2} x^T P x$$

$$\frac{\delta J}{\delta x} = x^T P$$
Putting it All Together

\[ J(x_t, u_t) = C(x_t, u_t)\delta t + J(x_{t+1}, u_{t+1}) + \frac{\delta J}{\delta x}(Ax_t + Bu_t)\delta t \]

\[ C(x, u) = \frac{1}{2}x^TQx + \frac{1}{2}u^TRu \]

\[ \frac{\delta J}{\delta x} = x^TP \]

How do we minimize this with respect to \( u \)?

\[ J(x_t, u_t) = \frac{1}{2}x_t^TQx_t\delta t + \frac{1}{2}u_t^TRu_t\delta t + J(x_{t+1}, u_{t+1}) + x_T^TP(Ax_t + Bu_t)\delta t \]

\[ \frac{\delta J}{\delta u} = 0 \]

\[ \frac{\delta J}{\delta u} = 0 + u_t^TR + \frac{\delta J}{\delta u} + x_t^TPB \]

\[ u_t^TR + x_t^TPB = 0 \]
Therefore Gain Matrix: $K = ?$

$u_t^T R + x_t^T PB = 0$

Nice Exercise # 1

$u_t = K x_t \quad \text{so} \quad K = -R^{-1}B^T P$

The Rest is Simple Linear Algebra

Recall Bellman’s Equation:

$$J(x_t, u_t) = \frac{1}{2} x_t^T Q x_t + \frac{1}{2} u_t^T R u_t + J(x_t, u_t) + x_t^T P (Ax_t + Bu_t) u_t$$

Recall our solution for $u_t$:

$$u_t = -R^{-1}B^T P x_t$$

Cancel some terms and replace every instance of $u_t$ with the control of $x_t$

$$0 = \frac{1}{2} x_t^T Q x_t + \frac{1}{2} (-R^{-1}B^T P x_t)^T R (-R^{-1}B^T P x_t) + x_t^T P (Ax_t + B(-R^{-1}B^T P x_t))$$
\[ 0 = \frac{1}{2} x_t^T Q x_t + \frac{1}{2} (-R^{-1} B^T P x_t)^T R (-R^{-1} B^T P x_t) + x_t^T P A x_t - x_t^T B R^{-1} B^T P x_t \]

**Exercise 2**

\[ 0 = \frac{1}{2} x_t^T Q x_t - \frac{1}{2} x_t^T P B R^{-1} B^T P x_t + x_t^T P A x_t \]

\[ 0 = Q - P B R^{-1} B^T P + P A + A^T P \]
There’s that Riccati Equation!

- Linear Dynamic System: \( \dot{x} = Ax(t) + Bu(t) \)
- Quadratic Performance Criterion: \( C(x, u) = \frac{1}{2} x^T Q x + \frac{1}{2} u^T R u \)
- Gain Matrix: \( K = R^{-1} B^T P \) \( u = K x = R^{-1} B^T P x \)
- Algebraic Riccati Equation:

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

What did we just show?

**LQR is a Special Case of Dynamic Programming!**

- Given a Linear System
- Quadratic Cost Criterion