1 Linear Quadratic Regulator (LQR)

Consider: Why is it called “Linear Quadratic Regulator”? Consider: What are the inputs, design variables, and outputs of an LQR design problem?

1.1 The Linear Quadratic Regulator

Setup. Consider designing a control policy for a discrete linear time-invariant dynamical system. One way to design a control policy \( u = f(x) \) is to write down a cost function which penalizes the overall state deviation, input, and terminal cost.

LQR Problem. We seek to minimize our cost functional subject to our dynamics constraints:

\[
J(x) = \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) + x_N^T P_N x_N
\]

where \( Q, R, P > 0 \), and \( U := \{u_0, u_1, \ldots, u_{N-1}\} \).

1.2 LQR Solution via Dynamic Programming Principle

Let \( * \) denote optimal, not complex conjugate transpose. Consider the Dynamic Programming equations, aka Bellman, aka Hamilton-Jacobi equations:

\[
J(x) = \min u_k \left\{ x_k^T Q x_k + u_k^T R u_k + J(x_{k+1}) \right\}
\]

The above expression gives \( J_k \) recursively, in terms of \( J_{k+1} \). Any minimizing control \( u_k \) gives optimal \( J_k \).

Find recursive form:

1. At final time, we know \( J_N = x_N^T P_N x_N \).
2. By Bellman principle, \( J_k = \min_{u_k} \left( x_k^T Q x_k + u_k^T R u_k + J_{k+1}(x_{k+1}) \right) \).

October 23, 2020

3. We solve for \( u_k \) by taking derivative w.r.t. \( u_k \) and setting to zero (we start omitting \( k \)’s from \( x \) and \( u \) temporarily):

\[
2x_k^T R + 2x_k^T B_k x_{k+1} + P_k = 0
\]

Consider: How do we know taking derivative w.r.t. \( u \) and setting to zero will give us the global optimal solution?

4. Hence optimal control law is:

\[
x_k^* = \arg \min_{x_k} x_k^T Q x_k + \frac{1}{2} x_k^T B_k x_{k+1} \quad \text{By def.}
\]

5. Substitute the optimal control law into the cost function to find the associated optimal cost:

\[
J_k^* = x_k^T Q x_k + \frac{1}{2} x_k^T B_k x_{k+1} - \frac{1}{2} x_{k+1}^T B_k^T \left( R + \frac{1}{2} B_k^T B_k \right)^{-1} B_k x_k
\]

1.3 Summarizing DP Solution

1. Set \( P_N := Q \).
2. For \( k = N-1, \ldots, 1 \):

\[
P_k := \left( R + \frac{1}{2} P_{k+1} B_k^T B_k \right)^{-1} B_k^T P_{k+1}
\]

3. For \( k = 0, \ldots, N-1 \), optimal \( u_k \) is given by \( u_k^* = -B_k x_k \).

Remarks:

Optimal \( u \) is a linear function of the state (called linear state feedback).

Recurrence for min cost-to-go runs backward in time:

\[
J_k^* = \min_{u_k} \left( x_k^T Q x_k + u_k^T R u_k + J_{k+1}(x_{k+1}) \right)
\]

October 23, 2020

More on Problem 1:

and the cost matrix \( R = \frac{1}{2} \). Let \( x_0 = (1, 1) \) and \( N = 20 \). How do you think the closed-loop sequence \( y(t) \) and control efforts \( u(t) \) for different cost matrices \( Q \) compare?

\[
A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

Use MATLAB to simulate the system.

October 23, 2020
1.4 Infinite Horizon DT LQR

If \( N \to \infty \), we can still apply the dynamic programming principle. However, the recursive equation reaches a steady state solution in this case:

\[
\begin{align*}
K_{\infty} &= (R + B^T P_{\infty} B)^{-1} B^T P_{\infty} A \\ P_{\infty} &= Q + R + B K_{\infty} B^T
\end{align*}
\]

This will occur in Eq. 15, but this ARE

1.5 State Afline Systems

Problem 2. Suppose that we have an affine system

\[ x_{k+1} = Ax_k + Bu_k + e, \quad k \in \{0, 1, \ldots, N\} \tag{9} \]

Can we design an LQR controller for this open loop system? Here:

Consider the vector \([x_1, x_2, \ldots, x_N]\) and \(Q, R \geq 0\). Note that this gives a linear system of Eq. 15.

1.6 Trajectory Following For LTI Systems

Suppose that now we want to derive a control policy that maintains the deviation of our system’s trajectory from a reference trajectory \((Q^*_{\text{ref}}, u^*_{\text{ref}}), i \in \{0, 1, \ldots, N\} \) (in other words, we want to follow the reference...
trajectory as closely as possible). The cost function in this case is given by:

\[
J(x, u) = \sum_{k=0}^{N-1} \left( (x_k - x_{d,k})^T Q (x_k - x_{d,k}) + (u_k - u_{d,k})^T R (u_k - u_{d,k}) \right) + (x_N - x_{d,N})^T Q (x_N - x_{d,N}).
\]  

(10)

Problem 3. Derive the optimal LQR control policy and cost-to-go function that minimize (10) subject to the system dynamics in (1). Do we need to make any assumptions about the reference trajectory \((x_{d,0}, x_{d,1}, \ldots, x_{d,N})\)?

2 Extension: Iterative LQR (ILQR), Trajectory Following for Non-linear Systems

What if our dynamical system isn’t linear? Can the ideas from LQR be helpful? Consider designing a control policy for a discrete nonlinear dynamical system:

\[
x_{k+1} = f(x_k, u_k), \quad t \in \{0, 1, \ldots, N\}
\]  

(11)

\[x_k = u^\text{ref}.
\]

Assume we have been given a specific state trajectory \(x_{0}^0, x_{1}^0, \ldots, x_{N}^0\) that we want to control our system to. We can control our system along this trajectory if and only if

\[
\exists \delta x_{0}^0, \delta x_{1}^0, \ldots, \delta x_{N-1}^0 \text{ s.t. } x_{k}^0 = f(x_{k-1}^0, \delta x_{k}^0), \forall k \in \{0, N-1\}
\]  

Our ILQR problem looks really similar but now we have to consider these nonlinear dynamics constraints:

\[
\min_{u^0} \sum_{k=0}^{N-1} \left( (x_k - x_{d,k})^T Q (x_k - x_{d,k}) + (u_k - u_{d,k})^T R (u_k - u_{d,k}) \right) + (x_N - x_{d,N})^T Q (x_N - x_{d,N}).
\]  

(13)

Can we leverage anything we learned from working with linear dynamics? Yes! First... Consider: What is the difference between linearizing about an equilibrium point and linearizing about a trajectory?

Let’s transform our nonlinear system into a linear time-varying dynamical system. We will linearize around each of the desired trajectory states, i.e. perform a first-order Taylor expansion around the trajectory:

\[
x_{k+1} = f(x_k, u_k) + \frac{\partial f(x_k, u_k)}{\partial x} (x_k - x_{d,k}) + \frac{\partial f(x_k, u_k)}{\partial u} (u_k - u_{d,k})
\]  

(13)

\[x_{k+1} - x_{d,k+1} = A (x_k - x_{d,k}) + B (u_k - u_{d,k})
\]

(14)

\[x_{k+1} = A x_k + B u_k + x_{d,k+1}
\]

(15)

Let’s put to these linearized dynamics into our ILQR problem:

\[
\min_{u^0} \sum_{k=0}^{N-1} \left( (x_k - x_{d,k})^T Q (x_k - x_{d,k}) + (u_k - u_{d,k})^T R (u_k - u_{d,k}) \right) + (x_N - x_{d,N})^T Q (x_N - x_{d,N}).
\]  

(16)

s.t. \(x_{k+1} = A x_k + B u_k + x_{d,k+1}, \quad t \in \{0, 1, \ldots, N\}\)

Summarizing the ILQR Algorithm.

- until convergence
  - get corresponding state sequence \(x_{0}^0, \ldots, x_{N}^0\) for the current best guess of the optimal control
  - linearize the dynamics by computing a first-order Taylor expansion of the dynamics model
  - use the ILQR backups to solve for the optimal control policy
  - \(x_{0}^0, \ldots, x_{N}^0\) record the best guess of the optimal control policy

Consider: Does our solution to (16) give the global solution? Not necessarily. Our model is still ill. \(x_{d,k} = f(x_{d,k})\) can get stuck at local min.耐久と構造

For more info on LQR derivation: https://stanford.edu/class/ee363/lectures/lqr.pdf
For more practice with implementing LQR: EE221A (fall)
For more practice with ILQR related to stochastic control and robust design: ME233 (spring)