Tutorial on LQR and iLQR

Fernando Palafox
fernandopalafox@utexas.edu

1 Introduction

This tutorial derives the Linear Quadratic Regulator (LQR) and its nonlinear counterpart, the iterative LQR (iLQR). Unlike many tutorials, this derivation does not assume the value functions are quadratic. Instead, it shows how this quadratic form arises naturally from the problem formulation, making the logic of the derivation clearer. A JAX implementation with a variety of examples is available at github.com/fernandopalafox/iLQR.

2 Linear Quadratic Regulators (LQR)

2.1 Derivation

Consider the following optimization problem

$$\min_{u} \quad \frac{1}{2}x_T^\top Q_T x_T + \frac{1}{2}\sum_{t=1}^{T-1} x_t^\top Q_t x_t + u_t^\top R_t u_t \tag{1a}$$

$$x_{t+1} = A_t x_t + B_t u_t \tag{1b}$$

where x_0 is known. For brevity, we'll refer to the objective as $J(u)$. To solve this problem, we will use Bellman's Principle of Optimality. We begin by defining the value function as $V_k(x_k)$, a function which gives us the cost-to-go at a given step, given the optimal actions u^*. Then, we work backwards from the last state and compute the cost to go at the last timestep:

$$V_T(x_T) = \frac{1}{2}x_T^\top Q_T x_T \tag{2a}$$

$$= \frac{1}{2}x_T^\top S_T x_T. \tag{2b}$$

Similarly, for the state at $t = T-1$

$$V_{T-1}(x_{T-1}) = \min_{u_{T-1}} \frac{1}{2}x_{T-1}^T Q_{T-1} x_{T-1} + \frac{1}{2}u_{T-1}^\top R_{T-1} u_{T-1} + V_T(x_T). \tag{3}$$

Note that $V_T(x_T)$ can be written as a function of x_{T-1} and u_{T-1} by using the dynamics in eq. (1b).

$$V_T(x_T) = \frac{1}{2}(A_{T-1}x_{T-1} + B_{T-1}u_{T-1})^\top S_T(A_{T-1}x_{T-1} + B_{T-1}u_{T-1}) \tag{4}$$

Now $V_{T-1}(x_{T-1})$ is only a function of u_{T-1} and we can solve for u^*_{T-1} by setting the derivative of value, with respect to controls, to zero. That is,

$$\frac{\partial V_{T-1}}{\partial u_{T-1}} = R_{T-1}u_{T-1} + B_{T-1}^\top S_T B_{T-1}u_{T-1} + B_{T-1}^\top S_T A_{T-1}x_{T-1} = 0 \tag{5}$$

Then the optimal control is given by

$$u^*_{T-1} = -(R_{T-1} + B_{T-1}^\top S_T B_{T-1})^{-1} B_{T-1}^\top S_T A_{T-1}x_{T-1} \tag{6a}$$

$$\equiv K_{T-1}x_{T-1} \tag{6b}$$

where K_{T-1} is known as the feedback matrix.

By substituting u^*_{T-1} into eq. (3) we can write the value function at $t = T-1$ as a quadratic function of x_{T-1}:

$$V_{T-1}(x_{T-1}) = \frac{1}{2}x_{T-1}^\top S_{T-1}x_{T-1} \tag{7}$$

where

$$S_{T-1} = Q_{T-1} + K_{T-1}^\top R_{T-1}K_{T-1} + (A_{T-1} + B_{T-1}K_{T-1})^\top S_T(A_{T-1} + B_{T-1}K_{T-1}) \tag{8}$$

Now that we have $V_{T-1}(x_{T-1})$, we can use the same procedure to compute u^*_{T-2}. The exact procedure is described in line 11.

Algorithm 1: Linear Quadratic Regulator (LQR)

Input: Initial state x_0, dynamics A_t, B_t, cost matrices Q_t, R_t

1 **Backward Pass: Compute gains**
2 $S_T \leftarrow Q_T$
3 **for** $t = T$ **to** 1 **do**
4 $\quad K_{t-1} \leftarrow -(R_{t-1} + B_{t-1}^\top S_t B_{t-1})^{-1} B_{t-1}^\top S_t A_{t-1}$ `// Feedback gain`
5 $\quad S_{t-1} \leftarrow$ *eq.* (8) `// Value function update`
6 **end**
7 **Forward Pass: Roll out optimal control**
8 **for** $t = 0$ **to** $T - 1$ **do**
9 $\quad u_t \leftarrow K_t x_t$ `// Feedback policy`
10 $\quad x_{t+1} \leftarrow A_t x_t + B_t u_t$ `// Dynamics step`
11 **end**

Output: Optimal trajectory x, optimal controls u, and feedback gains K

3 Iterative Linear Quadratic Regulator (iLQR)

If the cost $J(u)$ and dynamics are nonlinear, we can use the iterative Linear Quadratic Regulator (iLQR). iLQR approximates cost and dynamics along a nominal trajectory and then computes an optimal trajectory and feedback matrices in terms of deviations from the nominal controls. This algorithm is referred to as iterative because we don't just solve it once: once we compute our first set of optimal controls, we roll them out, and then approximate the costs and dynamics again—but this time along the new trajectory. Over time, this algorithm pushes the calculated controls (and associated state trajectory) to state space that both minimizes cost and accurately approximates the dynamics.

Let the nonlinear dynamics be defined as $x_{t+1} = f(x_t, u_t)$. We begin by defining a nominal control trajectory $U = \{u_0, ..., u_{T-1}\}$. We can obtain it by solving a lower fidelity dynamics model (e.g., a linear model solved using LQR), randomly guessing, or simply using all zeros. Given U and an initial state x_0 we can compute the nominal state trajectory $X = \{x_0, ..., x_T\}$by rolling out the dynamics f.

The next step is to approximate the cost and dynamics by building a Taylor expansions about the nominal state and control trajectories. The cost will be approximated by a quadratic function, and the dynamics will be linearized. The overall approach is to use the approximations to perform LQR in terms of *deviations* from the nominal trajectory.

The first-order Taylor expansion in terms of state and control deviations δx_k and δu_k is given by

$$x_{t+1} + \delta x_{t+1} = f(x_t + \delta x_t, u_t + \delta u_t) \tag{9a}$$

$$\approx f(x_t, u_t) + \frac{\partial f}{\partial x}\bigg|_{x_t, u_t} \delta x_t + \frac{\partial f}{\partial u}\bigg|_{x_t, u_t} \delta u_t \tag{9b}$$

Therefore, deviations from the nominal state trajectory are described by the dynamical system:

$$\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t \tag{10}$$

where $A_t = \frac{\partial f}{\partial x}\big|_{x_t, u_t}$ and $B_t = \frac{\partial f}{\partial u}\big|_{x_t, u_t}$. This is a linear system which means we can easily apply LQR to it.

The next step is to form a quadratic approximation of the cost function. A general nonlinear cost $J(U)$ is given by

$$J(U) = \ell_T(x_T) + \sum_{t=1}^{T-1} \ell(x_t, u_t) \tag{11}$$

where ℓ_T is the terminal cost and ℓ is the stage cost. Using a quadratic Taylor expansion about the nominal trajectory we get

$$\delta J \approx \frac{1}{2}\delta x_T^\top Q_T \delta x_T + q_T^\top \delta x_T + \sum_{i=1}^{T-1} \frac{1}{2} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^\top \begin{bmatrix} Q_t & H_t \\ H_t^\top & R_t \end{bmatrix} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} + \begin{bmatrix} q_t \\ r_t \end{bmatrix}^\top \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} \tag{12}$$

where

$$
\begin{aligned}
Q_t &= \left.\frac{\partial^2 \ell}{\partial x^2}\right|_{x_t,u_t} \\
R_t &= \left.\frac{\partial^2 \ell}{\partial u^2}\right|_{x_t,u_t} \\
H_t &= \left.\frac{\partial^2 \ell}{\partial x \partial u}\right|_{x_t,u_t} \\
H_t^\top &= \left.\frac{\partial^2 \ell}{\partial u \partial x}\right|_{x_t,u_t} \\
q_t &= \left.\frac{\partial \ell}{\partial x}\right|_{x_t,u_t} \\
r_t &= \left.\frac{\partial \ell}{\partial u}\right|_{x_t,u_t}.
\end{aligned} \tag{13}
$$

Unlike a linear approximation, a positive-definite quadratic approximation has a minimum, which is important when solving for the optimal control in the Ricatti recursion. It is also relatively tractable since it doesn't include any higher-order tensor terms.

Now let's solve the dynamics programming problem. As in vanilla LQR, we solve for controls backwards in time (since we initially only know what the optimal cost at the final state looks like). That is, if we want to solve for the controls at $t = T - 1$ we

1. Calculate the value function at time T, i.e., $\delta V_T(x_T)$
2. Set the derivative of the Q function at time $T - 1$ (which is in terms of δu_{T-1} and $\delta V_T(x_T)$) to zero.
3. Solve for u^*_{T-1}
4. Compute value function at time $T - 1$.
5. Repeat

We begin by finding the value at the last timestep

$$
\begin{aligned}
\delta V_T(\delta x_T) &= \frac{1}{2}\delta x_T^\top Q_T \delta x_T + q_T^\top \delta x_T && \text{(14a)} \\
&= \frac{1}{2}\delta x_T^\top S_T \delta x_T + s_T^\top \delta x_T + c_T && \text{(14b)}
\end{aligned}
$$

Recall δx_T is a function of the state and controls at the previous step, i.e., $\delta x_T = A_{T-1}\delta x_{T-1} + B_{T-1}\delta u_{T-1}$

Next, we plug $\delta V_T(\delta x_T)$ into the Q function at $t = T - 1$

$$
\delta \mathbb{Q}_{T-1} = \frac{1}{2}\begin{bmatrix}\delta x_{T-1} \\ \delta u_{T-1}\end{bmatrix}^\top \begin{bmatrix}Q_{T-1} & H_{T-1} \\ H_{T-1}^\top & R_{T-1}\end{bmatrix}\begin{bmatrix}\delta x_{T-1} \\ \delta u_{T-1}\end{bmatrix} + \begin{bmatrix}q_{T-1} \\ r_{T-1}\end{bmatrix}^\top \begin{bmatrix}\delta x_{T-1} \\ \delta u_{T-1}\end{bmatrix} + \delta V_T(\delta x_T) \tag{15a}
$$

And now we can set the derivative of $\delta\mathbb{Q}_{t-1}$ with respect to u_{t-1} to zero, and solve for δu^*_{T-1}:

$$\delta u^*_{T-1} = \underbrace{-(R_{T-1} + B^\top_{T-1} S_T B_{T-1})^{-1}(H^\top_{T-1} + B^\top_{T-1} S_T A_{T-1})}_{K_{T-1}} \delta x_{T-1}$$
$$\underbrace{-(R_{T-1} + B^\top_{T-1} S_T B_{T-1})^{-1}(B^\top_{T-1} s_T + r_{T-1})}_{d_{T-1}} \tag{16a}$$
$$= K_{T-1}\delta x_{T-1} + d_{T-1} \tag{16b}$$

By plugging δu^*_{T-1} in eq. (15), the new value function is given by

$$\delta V_{t-1}(x_{t-1}) = \frac{1}{2}\delta x^\top_{T-1} S_{T-1} \delta x_{T-1} + s^\top_{T-1}\delta x_T + c_{T-1} \tag{17}$$

where

$$\begin{aligned}
S_{T-1} &= Q_{T-1} + K^\top_{T-1} R_{T-1} K_{T-1} + H^\top_{T-1} K_{T-1} + K^\top_{T-1} H_{T-1} + Z^\top_{T-1} S_T Z_{T-1} \\
s_{T-1} &= q_{T-1} + K^\top_{T-1} r_{T-1} + (K^\top_{T-1} R^\top_{T-1} + H_{T-1}) d_{T-1} + Z^\top_{T-1}(S^\top_T B_{T-1} d_{t-1} + s_T) \\
c_{T-1} &= \frac{1}{2} d^\top_{T-1}(R_{T-1} + B_{T-1} S_T B_{T-1}) d_{T-1} + (r^\top_{T-1} + s^\top_{T-1} B_{T-1}) d_{T-1} + c_T
\end{aligned} \tag{18}$$

where $Z_{T-1} = A_{T-1} + B_{T-1}K_{T-1}$ In practice, for any t, it is not necessary to compute the constant term c_{t-1} because it has no effect on the computation of the optimal control δu^*_{t-1}. After computing δV_{T-1} we have everything we need to compute δu^*_{T-2} and can recursively compute the rest of the optimal controls δU^*.

A couple important points: First, unlike LQR, in iLQR we don't apply the full controls δU^*. Remember: we're using a linearized approximation of the dynamics for planning, which is not perfect. Therefore, we apply the following modified version of the controls:

$$\delta u^*_t = K_t \delta x_t + \alpha d_t \tag{19}$$

This control is in terms of a feedback term $K_t\delta x_t$ and feedforward term d_t weighted by α,. The feedforward term is weighted by α because even though δu^*_t is guaranteed to be an optimal solution to the approximate problem, we have no such guarantees for the actual nonlinear problem. And if the problem is particularly nonlinear, taking the full step ($\alpha = 1$) might take the controls into a region where the approximation is no longer a good one. Therefore, we implement a line-search which finds an $\alpha \in [0, 1]$ which we know will decrease the overall cost. The full iLQR procedure is shown in line 33. For an implementation with examples take a look at this repo github.com/fernandopalafox/iLQR.

4 Acknowledgments

Thanks Brian Jackson and Taylor Howell for this helpful tutorial. [1].

Algorithm 2: Iterative Linear Quadratic Regulator (iLQR)

Input: Initial state x_0, nominal control trajectory u, dynamics function $f(x_t, u_t)$, cost function $J(x, u)$

while *not converged* **do**
  Rollout
  for $t = 0$ **to** $T - 1$ **do**
    $x_{t+1} \leftarrow f(x_t, u_t)$ `// Rollout`
  end
  Backward Pass
  $S_T \leftarrow Q_T$
  $\Delta J = 0$
  for $t = T$ **to** 1 **do**
    $A_{t-1}, B_{t-1} \leftarrow$ eq. (10) `// Linear approx.`
    $Q_{t-1}, R_{t-1}, H_{t-1}, q_{t-1}, r_{t-1} \leftarrow$ eq. (13) `// Quadratic approx.`
    $K_{t-1}, d_{t-1} \leftarrow$ eq. (16) `// Policy parameters`
    $S_{t-1}, s_{t-1}, c_{t-1} \leftarrow$ eq. (18) `// Value function update`
    $\Delta J \leftarrow \Delta J - \frac{1}{2} d_{t-1}^\top (R_{T-1} + B_{T-1} S_T B_{T-1}) d_{t-1}$ `// Pred. cost decrease`
  end
  Line-search w/ Armijo Rule
  $\alpha \leftarrow 1$
  $J_0 \leftarrow J(x, u)$
  while $\alpha \geq \alpha_{min}$ **do**
    $\hat{x}_0 \leftarrow x_0$
    for $t = 0$ **to** $T - 1$ **do**
      $\hat{u}_t \leftarrow u_t + \alpha d_t + K_t(x_t - \hat{x}_t)$
      $\hat{x}_{t+1} \leftarrow f(\hat{x}_t, \hat{u}_t)$
    end
    $J_\alpha \leftarrow J(\hat{x}, \hat{u})$
    if $J_\alpha \leq J_0 + \eta \alpha \Delta J$ **then**
      $x, u \leftarrow \hat{x}, \hat{u}$ `// New nominal trajectory`
      break line-search loop
    end
    $\alpha \leftarrow \alpha * \beta$
  end
  Convergence Check
end

Output: Optimal trajectory x, optimal controls u, and feedback gains K

References

[1] B. Jackson and T. Howell, *Al-ilqr tutorial*, 2019.