

CILQR 时空联合规划算法详解

王泰翔

相比上述传统的来自无人机的规划算法，CILQR(Constrained Iterative LQR)则是较新的基于车辆运动学或动力学的优化方法，相比于以上的基函数规划方法其有几个明显的优点：

- 把轨迹规划问题处理成一个轨迹最优控制问题，直接考虑的运动学约束和控制量约束，理论上可以直接兼容覆盖泊车场景
- CILQR 可以直接在自车坐标系下进行规划，因而不需要去对感知的提供的车道线去做平滑了，而是可以直接使用视觉提供的车道线的 $C0, C1, C2, C3$ 参数来作为控制参考输入，和边界条件
- 整个求解算法，完全使用基础的矩阵运算，不需要使用任何求解器依赖库（只需要 EIGEN 库），算法对于测试和开发过程的 debug 完全是白盒，利于分析和工程化的优化改进

同时，相比于传统的 IPM 等通用非线性优化方法，CILQR 在车辆轨迹规划的特殊应用场景下的求解效率也更为高效,因为：

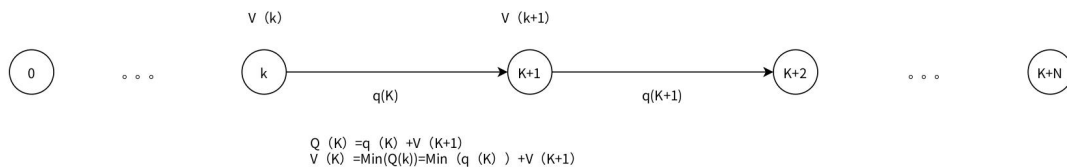
- 利用了动态轨迹的方法将全局优化问题转换为一个个独立的局部优化问题，简化了问题
- 相比于 DDP，CILQR 对车辆运动学约束省去了对 hessian 矩阵的计算，而只保留了一阶展开（只计算雅可比矩阵），所以计算速度更快。（对应车辆运动约束这个特殊的形式，计算 hessian 矩阵消耗的时间要远高于从二阶收敛降为一阶收敛多做的迭代次数）

CILQR 方法在传统线性 LQR 算法的基础上：

- 通过引入线性化迭代求解，来解决目标函数和系统状态方程的非线性问题
- 通过引入 barrier function，来解决不等式约束的问题

首先介绍传统 LQR 算法

基于动态规划方法的 LQR 算法得到的控制律是一系列的 pair，理解为一个离散的马尔科夫链的序列优化求解问题，而基于黎卡提方程计算的控制律是无限时域的，求解只是单步长。



车辆的轨迹可以理解为在离散的时间序列上，给定一个步长的控制量，车辆状态从当前状态转移到下一个状态的过程。而整个轨迹规划过程，即从当前状态，寻找一个最优控制序列，将车辆从当前状态，经过指定的时间步数，转移到目标状态的问题。

使用动态规划思想，我们可以将认为：

- 某个节点到终端节点的路径的最小累计 cost 即为该节点的状态价值 $V(k)$
- 从 k 节点给定一个控制量，其从当前 k 状态根据系统运动方程转移到 $k+1$ 状态产生的 cost 为 $q(k)$
- 假如从 k 步给定一个任意的控制量，车辆转移到 $k+1$ 状态，而从 $k+1$ 状态到终端状态，我们确保能够一直选择最优控制量，那么我们就可以认为当前 k 步在给定控制量下的动作价值 $Q(k) = q(k) + V(k+1)$
- 这样寻找一个最优的控制序列 u, u, \dots, u 的问题就可以结构成一个嵌套的从终端状态寻找最优控制量的子问题。 V, q, Q 均为标量代表 cost
- 如果优化求解问题的目标函数可以被写成 X, U 的二次型函数，且运动学等式约束可以写成 X, U 的线性组合，那这就一个标准的 LQR 优化问题。

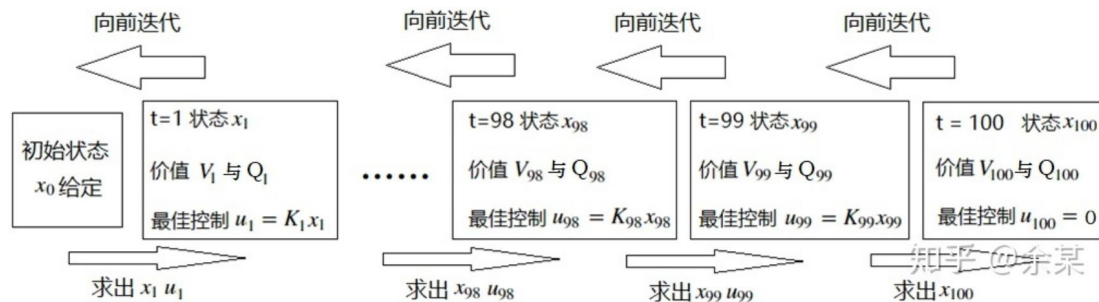
$$\min_{u_1, u_2, u_3, \dots, u_n} = \sum_{k=1}^n C(k)$$

$$X_{k+1} = F \begin{bmatrix} X_k \\ U_k \end{bmatrix} + f$$

$$C(k) = \frac{1}{2} \begin{bmatrix} X_k \\ U_k \end{bmatrix}^T \begin{bmatrix} C_{xx} & C_{xu} \\ C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} X_k \\ U_k \end{bmatrix} + \begin{bmatrix} X_k \\ U_k \end{bmatrix}^T \begin{bmatrix} D_x \\ D_u \end{bmatrix}$$

其中 $C_{ux} = C_{xu}^T$, C 是代价(cost) F 是状态转移方程, f 是前馈

这样从终端状态 $X(N)$ 我们可以求出上一步的最优控制反馈增益 $K(k-1)$, 以及对应的状态价值矩阵 $C(k-1)$, 这样一步步往前推, 直到初始状态 $X(0)$. 然后根据 $K(0)$ 正向求出每一步的最优控制量和最优状态。



• 按照上述的 backward, forward 优化求解方式, 我们可以以如下例子, 进行如下推导归纳:

◦ 在终端状态 (举例为第 100 步) 下, 因为是我们的目标状态, 因此没有 cost, 因此, 可以认为 $V(100) = 0$

◦ 则在第 99 步时, 其 action value, 是:

$$Q(99) = \frac{1}{2} \begin{bmatrix} X_{99} \\ U_{99} \end{bmatrix}^T \begin{bmatrix} C_{xx} & C_{xu} \\ C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} X_{99} \\ U_{99} \end{bmatrix} + \begin{bmatrix} X_{99} \\ U_{99} \end{bmatrix}^T \begin{bmatrix} D_x \\ D_u \end{bmatrix} + V(100)$$

要得到第 99 步的最优控制, 是的 $Q(99)$ 最小, 则可以对 U 求导, 进而找到动作价值的极值点: $\frac{\partial Q(99)}{\partial U_{99}} = 0$ 可以得到最优控制:

$$\begin{aligned} U_{99}^* &= K_{99} X_{99} + k_{99} \\ K_{99} &= -C_{uu}^{-1} C_{xu} \\ k_{99} &= -C_{uu}^{-1} D_u \end{aligned}$$

将最优控制量, 即可得到 99 步的 state value, 这里只需要状态量了, 同时注意 $\frac{1}{2}$ 依然存在, 注意矩阵维度

$$V(99) = \frac{1}{2} X_{99}^T V_{99} X_{99} + X_{99}^T v_{99} + const$$

其中:

$$\begin{aligned} V_{99} &= C_{xx} + K_{99} C_{ux} + C_{xu} K_{99} + K_{99}^T C_{uu} K_{99} \\ v_{99} &= c_{xu} k_{99} + K_{99}^T C_{uu} K_{99} + D_x + K_{99}^T D_u \end{aligned}$$

再次往前推进, 计算第 98 步, 其 action value:

$$Q(98) = \frac{1}{2} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \begin{bmatrix} C_{xx} & C_{xu} \\ C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \begin{bmatrix} D_x \\ D_u \end{bmatrix} + V(99)$$

$$= \frac{1}{2} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \begin{bmatrix} C_{xx} & C_{xu} \\ C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \begin{bmatrix} D_x \\ D_u \end{bmatrix} + \frac{1}{2} X_{99}^T V_{99} X_{99} + X_{99}^T v_{99} + const$$

又因为: $X_{99} = F \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + f$

因此可以展开写成:

$$Q(98) = \frac{1}{2} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \begin{bmatrix} C_{xx} & C_{xu} \\ C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \begin{bmatrix} D_x \\ D_u \end{bmatrix} + \frac{1}{2} (F \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + f)^T V_{99} (F \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + f) + (F \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + f)^T v_{99} + const$$

再次变形， $Q(98)$ 可以写成：

$$Q(98) = \frac{1}{2} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \tilde{Q} \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix} + \begin{bmatrix} X_{98} \\ U_{98} \end{bmatrix}^T \tilde{q} + const$$

注意这里又变成了二次型，利用之前二次型对控制量求偏导的方式，可以复用，再次求解最优控制律

$$\tilde{Q} = \begin{bmatrix} C_{xx} & C_{xu} \\ C_{ux} & C_{uu} \end{bmatrix} + F^T V_{99} F = \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix}$$

$$\tilde{q} = \begin{bmatrix} D_x \\ D_u \end{bmatrix} + \frac{1}{2} F^T V_{99} f + \frac{1}{2} F^T V_{99}^T f + F^T v_{99} = \begin{bmatrix} q_x \\ q_u \end{bmatrix}$$

注意： $\frac{1}{2} F^T V_{99} f + \frac{1}{2} F^T V_{99}^T f = F^T V_{99} f$

因为 $V_{99} = C_{xx} + K_{99} C_{ux} + C_{xu} K_{99} + K_{99}^T C_{uu} K_{99}$

因此 $V_{99}^T = V_{99}$

再次对 $Q(98)$ 对 U 求偏导，获得该步的最佳控制率

$$U_{98}^* = K_{98} X_{98} + k_{98}$$

$$K_{98} = -Q_{uu}^{-1} Q_{xu}$$

$$k_{98} = -Q_{uu}^{-1} q_u$$

将控制律代回 cost 中可得：

$$V(98) = \frac{1}{2} X_{98}^T V_{98} X_{98} + X_{98}^T v_{98} + const$$

$$V_{98} = Q_{xx} + K_{99} Q_{ux} + Q_{xu} K_{99} + K_{99}^T Q_{uu} K_{99}$$

$$v_{98} = Q_{xu} k_{99} + K_{99}^T Q_{uu} K_{99} + q_x + K_{99}^T q_u$$

可以理解成每次计算 V_{98} 、 v_{98} 是为了得到二次型的 \tilde{Q} 和 \tilde{q} 。以此方式，不断往前类推，直至达到初始状态 $x(0)$ 。

PIPELINE

BACKWARD：

for $t = T$ to 1:

$$Q_t = C_t + F_t^T V_{t+1} F_t$$

$$q_t = c_t + F_t^T V_{t+1} f_t + F_t^T v_{t+1}$$

$$Q(x_t, u_t) = const + \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T Q_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T q_t$$

$$u_t \leftarrow \arg \min_{u_t} Q(x_t, u_t) = K_t x_t + k_t$$

$$K_t = -Q_{u_t, u_t}^{-1} Q_{u_t, x_t}$$

$$k_t = -Q_{u_t, u_t}^{-1} q_{u_t}$$

$$V_t = Q_{x_t, x_t} + Q_{x_t, u_t} K_t + K_t^T Q_{u_t, x_t} + K_t^T Q_{u_t, u_t} K_t$$


$$v_t = q_{x_t} + Q_{x_t, u_t} k_t + K_t^T Q_{u_t} + K_t^T Q_{u_t, u_t} k_t$$

$$V(x_t) = const + \frac{1}{2} x_t^T V_t x_t + x_t^T v_t$$

知乎 @王沃河

FORWARD :

到这里，求出了每一步的最优控制律，从初始状态开始就可以逐步推导出每一步的最优状态。


$$\begin{aligned} &\text{for } t = 1 \text{ to } T: \\ &\quad \mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t \\ &\quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \end{aligned}$$

ILQR 的迭代求解思路

回顾 LQR 的求解推导过程，之所以可以这么推导，是因为代价函数是标准线性二次型，同时系统状态转移方程是 \mathbf{x} , \mathbf{u} 的线性组合：

$$\begin{aligned} C(k) &= \frac{1}{2} \begin{bmatrix} \mathbf{X}_k \\ \mathbf{U}_k \end{bmatrix}^T \begin{bmatrix} C_{xx} & C_{xu} \\ C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} \mathbf{X}_k \\ \mathbf{U}_k \end{bmatrix} + \begin{bmatrix} \mathbf{X}_k \\ \mathbf{U}_k \end{bmatrix}^T \begin{bmatrix} D_x \\ D_u \end{bmatrix} \\ \mathbf{X}_{k+1} &= \mathbf{F} \begin{bmatrix} \mathbf{X}_k \\ \mathbf{U}_k \end{bmatrix} + \mathbf{f} \end{aligned}$$

但是在轨迹规划问题中，cost 函数有可能包括非线性项，同时系统状态方程通常是非线性的。

新非线性模型：

$$\begin{aligned} \mathbf{X}_{k+1} &= f(\mathbf{X}_k, \mathbf{U}_k) \\ C_{k+1} &= l(\mathbf{X}_k, \mathbf{U}_k) \end{aligned}$$

因此：先把两个非线性函数线性化。

对于 cost 函数 l 我们进行二阶泰勒展开。

- 如果我们把系统方程 f 只做一阶泰勒展开，那这种方式就是 ILQR，如果把 f 函数也做二阶展开，那这种方式就是 DDP（微分动态规划）。所以 ILQR 是 DDP 的特例
- 使用 DDP 的方式，优化迭代是按二次速度收敛的，所以其需要的迭代次数更少，但是单次迭代的计算量更大。ILQR 则反之。对于轨迹规划和 MPC 控制这种问题而言，ILQR 效果相对而言更高。

新非线性模型线性化：

$$\begin{aligned} f(\mathbf{x}_t, \mathbf{u}_t) &= f(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix} \\ l(\mathbf{x}_t, \mathbf{u}_t) &= l(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} l(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix}^T \nabla_{\mathbf{x}_t, \mathbf{u}_t}^2 l(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix} \end{aligned}$$

展开模型转误差模型后变成标准二次型和线性模型：

$$\begin{aligned} f(\mathbf{x}_t, \mathbf{u}_t) - f(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) &= \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix} \\ l(\mathbf{x}_t, \mathbf{u}_t) - l(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) &= \nabla_{\mathbf{x}_t, \mathbf{u}_t} l(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix}^T \nabla_{\mathbf{x}_t, \mathbf{u}_t}^2 l(\widehat{\mathbf{x}}_t, \widehat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \widehat{\mathbf{x}}_t \\ \mathbf{u}_t - \widehat{\mathbf{u}}_t \end{bmatrix} \end{aligned}$$

通过对两个函数的线性化处理，我们发现在两个节点之间的 cost 的增量又变成标准二次型，而系统状态的变化量又变成了输入增量的线性组合。所以我们有可以使用 LQR 的求解方式了。这样我们就可以把原问题转化为，针对原有轨迹上每一步的控制计算最优修正量 $\delta \mathbf{u}$ ，使得每一步的对应 cost 下降最快。（换个角度考虑，如果我们能保证，每一步的控制量经过修正，每一步 cost 都能以最快的速度降低，经过多步的反复计算，我们一定能够收敛到一个最优的轨迹上）。同时以为 $\delta \mathbf{u}$ 足够小，我们就可以在该状态上进行线性化。因此，同样使用 action value 和 state value 的概念，我们需要优化的是，每一步上的 action value 的增量。

ILQR 的求解过程由三步组成，在每次优化中，这三步将被反复执行，直至总的 cost 收敛，这也是 ILQR 中 iterative 的来源。

ILQR 的第一个步骤

初始轨迹和梯度向量/海塞矩阵计算

根据初始轨迹提供的初始状态量，求解对应点上的目标函数的一阶梯度向量，和海塞矩阵和系统状态方程的雅可比矩阵。

由初始轨迹（即上一次迭代的规划轨迹，注意这里是轨迹而不是参考，这里意思是泰勒展开点，要区分展开点和参考点，这很重要）

可以计算得到每一个节点上的雅可比矩阵和海森矩阵(这个地方看不懂的话后续会实例展示)

$$l_k = \begin{bmatrix} l_x \\ l_u \end{bmatrix}_{x_k, u_k} = \begin{bmatrix} \frac{\partial l}{\partial x} \\ \frac{\partial l}{\partial u} \end{bmatrix}_{x_k, u_k}$$

$$L_k = \begin{bmatrix} l_{xx} & l_{xu} \\ l_{ux} & l_{uu} \end{bmatrix}_{x_k, u_k} = \begin{bmatrix} \frac{\partial^2 l}{\partial x \partial x} & \frac{\partial^2 l}{\partial x \partial u} \\ \frac{\partial^2 l}{\partial u \partial x} & \frac{\partial^2 l}{\partial u \partial u} \end{bmatrix}_{x_k, u_k}$$

$$F_k = \begin{bmatrix} f_x \\ f_u \end{bmatrix}_{x_k, u_k} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial u} \end{bmatrix}_{x_k, u_k}$$

ILQR 的第二个步骤：Backward 推导：

$$\delta Q(k) = \frac{1}{2} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix}^T \begin{bmatrix} l_{xx} & l_{xu} \\ l_{ux} & l_{uu} \end{bmatrix} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix} + \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix}^T \begin{bmatrix} l_x \\ l_u \end{bmatrix} + \delta V(k+1)$$

$$\delta V(k+1) = \frac{1}{2} \delta X_{k+1}^T V_{k+1} \delta X_{k+1} + \delta X_{k+1}^T v_{k+1} + const$$

$$\delta X(k+1) = \begin{bmatrix} f_x & f_u \end{bmatrix} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix}$$

带入展开后得：

$$\delta Q(k) = \frac{1}{2} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix}^T \begin{bmatrix} l_{xx} & l_{xu} \\ l_{ux} & l_{uu} \end{bmatrix} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix} + \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix}^T \begin{bmatrix} l_x \\ l_u \end{bmatrix} + \delta V(k+1)$$

$$+ \frac{1}{2} \left(\begin{bmatrix} f_x & f_u \end{bmatrix} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix} \right)^T V_{k+1} \left(\begin{bmatrix} f_x & f_u \end{bmatrix} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix} \right)^T$$

$$+ \left(\begin{bmatrix} f_x & f_u \end{bmatrix} \begin{bmatrix} \delta X_k \\ \delta U_k \end{bmatrix} \right)^T v_{k+1} + const$$

重新整理得：

$$\delta Q(k) = \frac{1}{2} \begin{bmatrix} 1 \\ \delta X_k \\ \delta U_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta X_k \\ \delta U_k \end{bmatrix}$$

$$Q_x = l_x + f_x^T v_{k+1}$$

$$Q_u = l_u + f_u^T v_{k+1}$$

$$Q_{xx} = l_{xx} + f_x^T V_{k+1} f_x$$

$$Q_{xu} = l_{xu} + f_x^T v_{k+1} f_u$$

$$Q_{ux} = l_{ux} + f_u^T V_{k+1} f_x$$

$$Q_{uu} = l_{uu} + f_u^T V_{k+1} f_u$$

对 $\delta Q(k)$ 求 δU_k 偏导，得到最优控制率：

$$\begin{aligned}\delta U_k^* &= K_k \delta X_k + k_k \\ K_k &= -Q_{uu}^{-1} Q_{ux} \\ k_k &= -Q_{uu}^{-1} Q_u\end{aligned}$$

带入得：

$$\begin{aligned}V_{k+1} &= Q_{xx} + K_k^T Q_{uu} K_k + K_k^T Q_{ux} + Q_{ux}^T K_k \\ v_{k+1} &= Q_x + K_k^T Q_{uu} k_k + K_k^T Q_u + Q_{ux}^T k_k\end{aligned}$$

背景知识：

但是存在一个问题： Q_{uu}^{-1} 。这个 Q_{uu} 虽然实对称但不一定满秩或者可能行列式为 0（非正定），因此要使用 LM 系数法对 K_k, k_k 进行修正。

```
Q_uu_evals, Q_uu_evecs = np.linalg.eig(Q_uu)
Q_uu_evals[Q_uu_evals < 0] = 0.0
Q_uu_evals += lamb
Q_uu_inv = np.dot(Q_uu_evecs, np.dot(np.diag(1.0/Q_uu_evals), Q_uu_evecs.T))
```

方式详解：首先，通过使用 `np.linalg.eig()` 函数计算 Q_{uu} 的特征值和特征向量。然后，将所有小于零的特征值截断为零，以确保 Q_{uu} 是正定矩阵（因为协方差矩阵必须是正定的）。接下来，加上一个较小的正常数 λ ，以确保 Q_{uu} 可逆。这个正常数通常被称为正则化参数，它的作用是防止矩阵奇异，或者说防止逆矩阵中的元素过大。最后，通过对特征向量和特征值进行运算，计算出 Q_{uu} 的伪逆 Q_{uu}^{-1} 。这里使用 `np.dot()` 函数和 `np.diag()` 函数来实现矩阵的乘法和对角线提取。需要注意的是，在计算伪逆时，分母中的特征值应该是经过修正后的，即加上了正则化参数 λ 的特征值。同时对于一个对称矩阵 Q_{uu} ，其特征向量构成的矩阵是正交矩阵，即 $v v^{-1} = I$

$$A v = v \lambda$$

可得：

$$A^{-1} = v \lambda^{-1} v^{-1}$$

技巧：

λ 是 >0 的系数，当 λ 接近于 0 时，迭代优化收敛的更快，但是系统容易不稳定；当 λ 值很大时，则系统可以变得更稳定，不会发散，但是系统收敛的更慢，需要更多步的迭代，计算时间增长。因此，需要在每一步的 backward pass 时动态的去调整 λ 参数：

- 在接近最优点附近时，逐渐调小 λ
- 而当原有 Q_{uu} 非正定时，则需要快速增加 λ ，并重新去尝试 backward pass

我们按照如下方法，动态的去调整系数 λ ：

Forward 推导：

通过 backward 步得到的最优控制率 K_k, k_k ，我们就可以从初始状态，再一步步反推，得到新的修正过的轨迹。然后以新的轨迹，回到第一步，以新的轨迹点上的状态，重新计算对应点上的雅可比矩阵和海塞矩阵。

$$X_0 = X_{start}$$

每一个节点上上新的控制量，可以在旧轨迹控制量的基础上，做相应修正，然后算出新的状态量 (\widehat{U}_k 是展开点)

$$\begin{aligned}U_k &= \widehat{U}_k + K_k \delta X_k \\ X_{k+1} &= f(X_k, U_k)\end{aligned}$$

解决非线性函数的优化问题。还有最后一个问题，ILQR 无法直接考虑不等式约束，而这在轨迹规划中是必须考虑的。在 CILQR 中，我们将不等式约束，通过一个 **barrier function** 转换成等效 cost 来处理。

$$\begin{aligned} d(U_k) &\leq 0 \\ d(X_k) &\leq 0 \end{aligned}$$

举例： $-5 \leq U_k \leq 5$ 要写成： $d_1(U_k) = U_k - 5, d_3(U_k) = -5 - U_k$

以上为在每个时间步上对状态和控制的约束条件，而且约束函数 d 也可以是非线性函数。

首先我们构造一个 barrier function，将约束函数处理成一个可导方程形式。Barrier function 有不同的形式可用，比如（指数函数，log 函数，平方函数），以下以指数函数为例：

$$b(x_k) = q_1 e^{q_2 d(x_k)}$$

当 $d(x_k) \geq 0$ 时，barrier function 就会趋于无穷大，从而实现等效约束，但是由于 $d(x_k)$ 和 $b(x_k)$ 都是非线性函数，我们还不能直接使用，需要对其在 x_k 状态下进行线性化：将其 x_k 处对 \hat{x}_k 进行泰勒二阶展开：

原文忽略二次高阶项，只保留一次项：

$$\delta b(x_k) = \delta b(x_k + \delta x_k) - \delta b(x_k) = \delta x_k^T \frac{\delta b}{\delta x} + \frac{1}{2} \delta x_k^T \frac{\delta^2 b}{\delta x^2} \delta x_k$$

这里重新写下 CILQR 下的优化求解流程：

Repeat 如下步骤，直至 V 收敛：

Step1：

- 根据初始的轨迹，计算每个节点上的雅可比矩阵，海塞矩阵
- 根据初始轨迹，计算在每个节点上的状态、控制量的 barrier function
- 将 barrier cost 项加入到原始的 cost 矩阵中，对目标函数的梯度矩阵和海塞矩阵更新

CILQR 实战例子：

状态量： $[x, y, v, \phi]$ 控制量： $[a, \omega]$

数学模型：

$$\begin{aligned} \dot{x} &= (v + 0.5at)\cos(\phi) & f_1 \\ \dot{y} &= (v + 0.5at)\sin(\phi) & f_2 \\ \dot{v} &= a & f_3 \\ \dot{\phi} &= \omega & f_4 \end{aligned}$$

则对于参考轨迹的任意一个参考点，用 r 表示，上式可以改写为

$$\dot{X} = f(X, U) \implies \dot{X}_r = f(X_r, U_r)$$

对参考点泰勒展开：

$$\dot{X} - \dot{X}_r = f(X_r, U_r) + \frac{\partial f(X, U)}{\partial X}(X - X_r) + \frac{\partial f(X, U)}{\partial U}(U - U_r)$$

其中 $\frac{\partial f(X, U)}{\partial X}$ ：

$$A = \frac{\partial f(X, U)}{\partial X} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \phi} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \phi} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \phi} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial \phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cos(\phi) & -(v + 0.5a\Delta t)\sin(\phi) \\ 0 & 0 & \sin(\phi) & (v + 0.5a\Delta t)\cos(\phi) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \frac{\partial f(X, U)}{\partial U} = \begin{bmatrix} \frac{\partial f_1}{\partial a} & \frac{\partial f_1}{\partial \omega} \\ \frac{\partial f_2}{\partial a} & \frac{\partial f_2}{\partial \omega} \\ \frac{\partial f_3}{\partial a} & \frac{\partial f_3}{\partial \omega} \\ \frac{\partial f_4}{\partial a} & \frac{\partial f_4}{\partial \omega} \end{bmatrix} = \begin{bmatrix} 0.5\Delta t \sin(\phi) & 0 \\ 0.5\Delta t \cos(\phi) & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

则 $\dot{X} - \dot{X}_r = f(X_r, U_r) + \frac{\partial f(X, U)}{\partial X}(X - X_r) + \frac{\partial f(X, U)}{\partial U}(U - U_r)$ 可以改成误差模型：

新状态量 X_{new} : $\begin{bmatrix} x - x_r \\ y - y_r \\ v - v_r \\ \phi - \phi_r \end{bmatrix}$ 控制量 U_{new} : $\begin{bmatrix} a - a_r \\ \omega - \omega_r \end{bmatrix}$

$$\tilde{X} = \begin{bmatrix} \dot{x} - \dot{x}_r \\ \dot{y} - \dot{y}_r \\ \dot{v} - \dot{v}_r \\ \dot{\phi} - \dot{\phi}_r \end{bmatrix} = AX_{new} + BU_{new}$$

模型线性化：

$$\frac{X_{new(k+1)} - X_{new(k)}}{\Delta t} = AX_{new(k)} + BU_{new(k)}$$

得到：

$$X_{new(k+1)} = (\Delta t A + I)X_{new(k)} + \Delta t BU_{new(k)}$$

解释：哪怕原模型已经是线性模型也需要再次线性化，因为约束和代价是二次型都是误差模型，因此状态量必须对应的上。

代价函数一共有两种，一种是和参考的代价，一个是和硬约束的代价，以控制量举例：

一、Ref cost

$$cost(u) = (u - u_r)^2 = u^2 - 2u_r u + const$$

2 阶泰勒展开变成和模型相同控制量的误差模型：（ u_r 是参考，而 u_0 是展开点，即上一时刻规划轨迹同时时间戳的点）

$$cost(u) - cost(u_0) = (2u_0 - 2u_r)(u - u_0) + \frac{2}{2!}(u - u_0)^2$$

二、Barrier Function：

以控制量举例：

$$error = u - u_{bound} < 0$$

以上一次规划结果计算 error 以及 $q_1 e^{q_2 error}$ 的一阶导和二阶导

$$cost(u) = q_1 e^{q_2(u - u_{bound})} = C(u)$$

2 阶泰勒展开：

$$cost(u) - cost(u_0) = \frac{\partial C}{\partial a}(u_0)(u - u_0) + \frac{1}{2!} \frac{\partial^2 C}{\partial a^2}(u_0)(u - u_0)^2$$