基于最优控制 iLQR 的时空联合规划

iLQR大循环

0. 迭代次数k=0->M；

1. $f(x,u)$线性化，计算f_x,f_u。如果有必要，代价函数(x, u)展开成二次型；

iLQR内循环

2. 反向传播，$i=N\to 0$,更新反馈增益$\{K_0,K_1,\ldots,K_N\}$，

1) $$\begin{cases} Q_{u_i} = g_u + f_u^T V_{x_{i+1}} \\ Q_{x_i x_i} = g_{xx} + f_x^T V_{x_{i+1}x_{i+1}} f_x \\ Q_{xu} = g_{xu} + f_x^T V_{x_{i+1}x_{i+1}} f_u \\ Q_{ux} = g_{ux} + f_u^T V_{x_{i+1}x_{i+1}} f_x \\ Q_{uu} = g_{uu} + f_u^T V_{x_{i+1}x_{i+1}} f_u \end{cases}$$

2) $$\begin{cases} K_i = -Q_{uu}^{-1} Q_{ux} \\ d_i = -Q_{uu}^{-1} Q_u \end{cases}$$

3) $\delta u_i{}^* = -Q_{uu}^{-1}\left(Q_{ux}\delta x_i + Q_{u_i}\right) = K\delta x_i + d$

4) $$V_{\mathbf{x}}(i) = Q_{\mathbf{x}} - Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}$$
$$V_{\mathbf{xx}}(i) = Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}.$$

iLQR内循环

3. 正向传播，$i=0\to N$,使用非线性方程$\dot{x}=f(x,u)$，更新$\{x_0,x_1,\ldots,x_N\}$

$$\delta x_k = x_k^i - x_k^{i-1}$$

$$\delta u_k = d_k + K_k \delta x_k$$

$$u_k^i = u_k^{i-1} + \delta u_k$$

$$x_{k+1}^i = f(x_k^i, u_k^i)$$

4. 计算每一次迭代的代价，J^k,J^{k+1},并计算是否满足收敛标准

$$\left|\frac{J^{(k+1)} - J^{(k)}}{J^{(k)}}\right| < \epsilon_J$$

目　录

1 最优控制背景介绍

2 函数极值和乘子法

3 最优控制变分法

4 极小值原理

5 动态规划

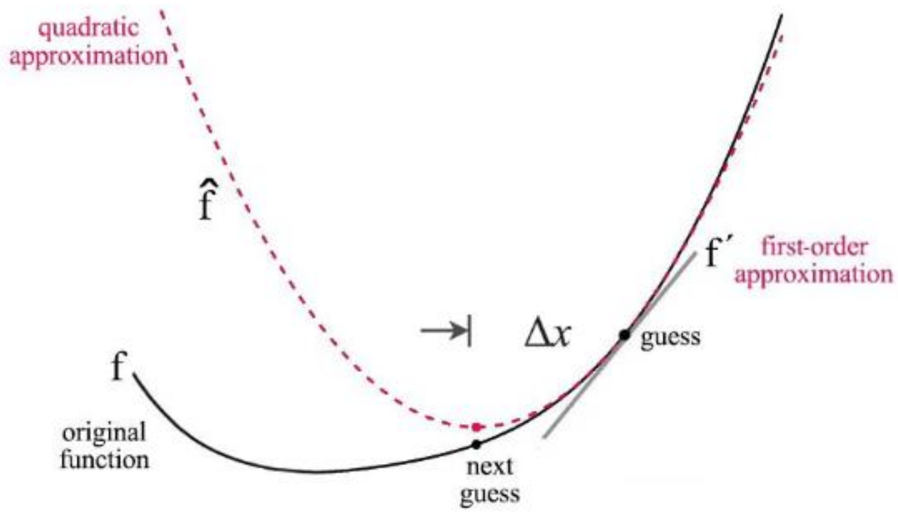
6 LQR

7 ilqr 推导

7.1 背景

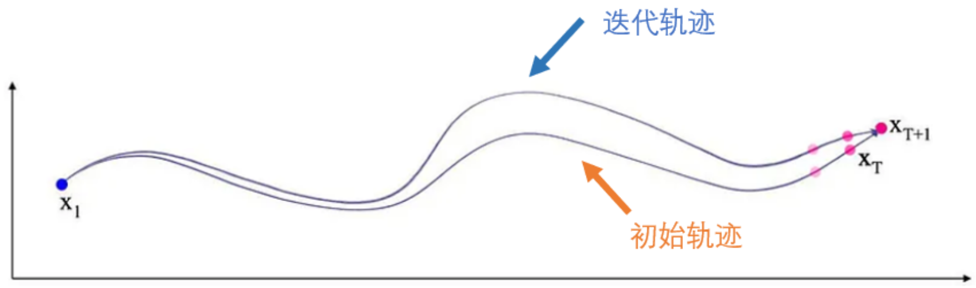
对于非线性模型，如何去解最优控制问题呢？一种思路是使用线性化的方式，来求解近似解。我们在凸优化中已经了解到，对于一个非线性的函数，为了计算极值，我们将原来的函数，线性近似或者二次函数近似，从而计算近似函数的下降点，通过迭代，最终得到原来函数的极值。

近年来，使用 iLQR/DDP 来计算轨迹优化问题越来越流行。其实 ilqr 的思想也借鉴了数值优化中的迭代思想，对于一个非线性极值问题，我们使用二阶泰勒展开去近似原始函数，求解每一次近似函数的极值，通过这种方式迭代地计算非线性曲线的极值。如下图所示。



对于一个 iLQR 问题，因为$\dot{x} = f(x,u)$是非线性的，此时如果直接在原始的 LQR 计算框架中，无法得到解析解，此时 cost-to-go 函数$V(x)$是一个复杂的高维度的非线性函数。观察 LQR 框架，Q(x,u) 和 V(x)都是二次的函数，所以，我们将$f(x,u)$线性化，将复杂的 cost-to-go 函数$V(x)$泰勒展开取二阶，来近似原来的$V(x)$，这样就可以使用 LQR 框架来计算轨迹。为了求解最优轨迹，我们先给定一条轨迹，然后设定需要的代价

函数 $\mathrm{cost}(x,u)$ ，在每一次迭代中都计算最优解，然后 cost 函数用来调整解，直到收敛。如下图所示。



DDP/ilqr 的概念和推导，在 1960 年代就出现了[1]，当时还叫做 DDP[2]，然后这个古老的技术，在二十一世纪继续发光发热，在机器人的轨迹优化领域依然有着不俗的表现，现在又继续简化 ddp，重新定义了一个新的名字 ilqr[3]。

7.2 线性化

Lqr 问题一个良好的性质是，状态方程是线性的，cost function 是二次的，可以通过里卡提方程得到解析解。而对于非线性方程，得不到解析解，只能通过数值解来计算了。所以，在处理非线性状态方程问题时，可以将状态方程线性化，转换成常规 lqr 可以求解的方式。

这里特别的，一般求解求解连续的 lqr 问题，使用 HJB 方程来计算。在 iLQR 中，我们一般将连续的状态方程转换成离散方程，从而使用动态规划原理来计算轨迹。

[1] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. Elsevier, 1970；

[2] D. Mayne, “A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems,” International Journal of Control, vol. 3, no. 1, pp. 85–95, 1966

[3] Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization

离散状态方程

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

代价函数

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \quad s.t.\ \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

如果将上面的求和公式展开，并且将状态转移方程代入，那么代价函数可以写成

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots + c(f(f(\ldots)\ldots), \mathbf{u}_T)$$

这是一个不包含状态变量 x，只包含控制量 u 的方程，为了求出最优解，可以对控制量$u = \{u_0, u_1, \ldots, u_N\}$进行求导，但是，由于 cost function 嵌套太多，求导比较困难，所以，此时利用到了贝尔曼最优性原理来解决这个问题。即未来状态只和当前状态有关，而和过去状态无关。

所以，本来是计算整个序列{u_0, u_1,u_n}的过程，使用了 bellman 原理，

$$Q_k = g(x_k, u_k) + V_{k+1}(x_{k+1})$$

$$V_k = \min\{Q_k\}$$

要想使得 k 阶段的 cost-to-go V_k 最优，所以只需要Q_k取极小值即可，因为$V_{k+1}(x_{k+1})$在上一个阶段已经计算出来了， 此时Q_k和x_k, u_k有关，状态量是受控制量影响的，所以计算得到最优的u_k即可。

经过上述操作，动态规划将计算$u = \{u_0, u_1, \ldots, u_N\}$的过程，转变成了计算单步$u_k$的过程，然后 k=N->0，迭代地计算下去，整体地控制量也就可以得到，这样求解就简

单多了。然后利用上 DP 的整个 backward pass 和 forward pass，就可以得到整个解。

对于 ilqr 而言，将非线性的状态转移方程、目标函数泰勒展开，然后使用 lqr 求解方式去计算近似解，得到最后结果。

泰勒展开：

$$f(\mathbf{x}_t,\mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t,\mathbf{u}_t} f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$c(\mathbf{x}_t,\mathbf{u}_t) \approx c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}^T \nabla^2_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

移项：

$$f(\mathbf{x}_t,\mathbf{u}_t) - f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \approx \nabla_{\mathbf{x}_t,\mathbf{u}_t} f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$c(\mathbf{x}_t,\mathbf{u}_t) - c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \approx \nabla_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}^T \nabla^2_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

新增加符号$\delta x, \delta u$

$$\bar{f}(\delta\mathbf{x}_t,\delta\mathbf{u}_t) = \underbrace{\mathbf{F}_t}_{\nabla_{\mathbf{x}_t,\mathbf{u}_t} f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t)} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix} \qquad \bar{c}(\delta\mathbf{x}_t,\delta\mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}^T \underbrace{\mathbf{C}_t}_{\nabla^2_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t)} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}^T \underbrace{\mathbf{c}_t}_{\nabla_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t)}$$

$$\delta\mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$$
$$\delta\mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$$

这样，整个状态转移方程、目标函数变成和 lqr 一样的形式。

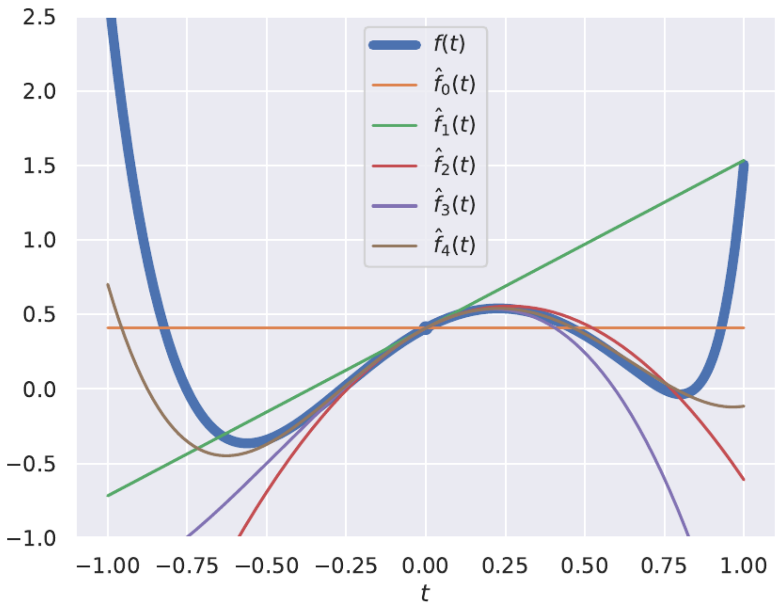
状态转移方程也可以写成下面的形式

$$f(x_k + \delta x, u_k + \delta u) = f(x) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial u}\delta u + \textit{高阶项}$$

其中，

$$\frac{\partial f}{\partial x} = f_x, \frac{\partial f}{\partial u} = f_u$$

如图所示，对于非线性方程，通过泰勒展开，可以对原始的函数有不同程度的局部近似，



7.3 离散化

为了在数值上计算解，通常需要将方程离散化，连续状态方程转换成离散方程，可以假设在一个采样周期 T 内，控制量 u 保持不变，也就是一阶保持器的形式，在这个假设下，我们可以通过系统的解析解，给直接离散化[4]。但是，有时候很难知道系统的解析解，所以使用近似离散化的方式。

根据导数的定义：

$$\dot{x}(t_0) = \lim_{\Delta t \to 0} \frac{x(t_0 + \Delta t) - x(t_0)}{\Delta t}$$

[4] 现代控制理论，刘豹，第二章

现讨论 $t_0 = kT$ 到 $t = (k+1)T$ 这一段的导数，有：

$$\dot{x}(kT) \approx \frac{x((k+1)T) - x(kT)}{T}$$

以此代入 $\dot{x}(t) = Ax(t) + Bu(t)$ 中，得

$$\frac{x((k+1)T) - x(kT)}{T} = Ax(kT) + Bu(kT)$$

$$x((k+1)T) = (TA + 1)x(kT) + TBu(kT)$$

$$G(T) \approx TA + 1$$

$$H(T) \approx TB$$

得到

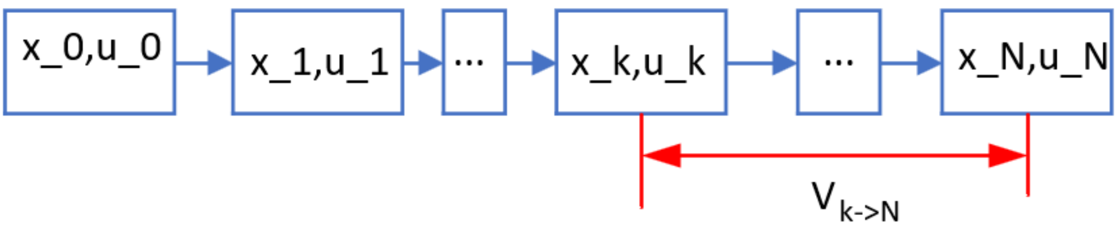
离散时间状态空间表达式为：

$$x(k+1) = G(T)x(k) + H(T)u(k)$$

$$y(k) = Cx(k) + Du(k)$$

7.4 Backward pass

动态规划中 2 个常用的概念，动作价值函数 Q(x,u)和价值函数 V(x)。



如图所示，阶段 k 到阶段 N，最优的 cost 叫做价值函数，或者 cost to go，$V_{k\to N}$，简写成V_k。

在阶段 k 这里，可以u_k可以有多种选择，那么每一种选择都会带来不同的 cost，所以，在这里又新增加一个概念叫做 action-value 函数 Q，影响 Q 取值的变量包括 x 和 u，所以一般写成$Q(x_k,u_k)$。

动作值函数$Q(x_k,u_k)$，有 2 个代价组成。一个代价是，在阶段 k，系统处于状态 x，当前动作 u 带来的代价 g(x, u)。另一个代价是上一个阶段 k+1 的最优代价 cost to go。所以，Q 表达式为

$$Q(x_i,u_i) = g(x_i,u_i) + V(x_{i+1}) = g(x_i,u_i) + V(f(x_i,u_i))$$

在 k 阶段，如果得到最优的u_k，那么也就得到最优的V_k。写成表达式，

$$V(x_i) = \min\{Q(x_i,u_i)\}$$

Q 泰勒展开

$$\begin{aligned}Q_k + \delta Q_k &= Q(x_k+\delta x, u_k+\delta u)\\ &\approx Q(x_k,u_k) + \frac{\partial Q}{\partial x}\bigg|_{x_k,u_k}(x-x_k) + \frac{\partial Q}{\partial u}\bigg|_{x_k,u_k}(u-u_k)\\ &\quad + \frac{1}{2}(x-x_k)^T\frac{\partial^2 Q}{\partial x^2}\bigg|_{x_k,u_k}(x-x_k) + \frac{1}{2}(u-u_k)^T\frac{\partial^2 Q}{\partial u^2}\bigg|_{x_k,u_k}(u-u_k)\\ &\quad + \frac{1}{2}(u-u_k)^T\frac{\partial^2 Q}{\partial u\partial x}\bigg|_{x_k,u_k}(x-x_k) + \frac{1}{2}(x-x_k)^T\frac{\partial^2 Q}{\partial x\partial u}\bigg|_{x_k,u_k}(u-u_k)\end{aligned}$$

δQ继续整理

$$\delta Q = Q_x^T\delta x + Q_u^T\delta u + \frac{1}{2}(\delta x)^T Q_{xx}\delta x + \frac{1}{2}(\delta u)^T Q_{ux}\delta x + \frac{1}{2}(\delta x)^T Q_{xu}\delta u + \frac{1}{2}(\delta u)^T Q_{uu}\delta u$$

δQ可以写成

$$: \frac{1}{2}\begin{bmatrix}\delta x_i\\ \delta u_i\end{bmatrix}^T\begin{bmatrix}Q_{x_i^2} & Q_{x_i u_i}\\ Q_{u_i x_i} & Q_{u_i^2}\end{bmatrix}\begin{bmatrix}\delta x_i\\ \delta u_i\end{bmatrix} + \begin{bmatrix}Q_{x_i}\\ Q_{u_i}\end{bmatrix}^T\begin{bmatrix}\delta x_i\\ \delta u_i\end{bmatrix}$$

继续整理

$$= \frac{1}{2}\begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}$$

价值函数 V 泰勒展开

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

所以

$$\delta V = \frac{\partial V}{\partial x}\delta x + \frac{1}{2}(\partial x)^T \frac{\partial^2 V}{\partial x^2}\delta x = V_x^T \delta x + \frac{1}{2}(\partial x)^T V_{xx}\delta x$$

δQ也可以将泰勒展开式移项，表达为

$$\begin{aligned} \delta Q &= Q(x_k + \delta x, u_k + \delta u) - Q_k \\ &= g(x_i + \delta x, u_i + \delta u) + V(f(x_i + \delta x, u_i + \delta u) - g(x_i, u_i) \\ &\quad + V(f(x_i, u_i) \end{aligned}$$

δQ的表达式中，其中

$$Q_x^T = \frac{\partial Q}{\partial x} = \frac{\partial(g(x_i, u_i) + V(f(x_i, u_i)))}{\partial x}$$

上式子继续化简

$$Q_x^T = \frac{\partial g}{\partial x} + \frac{\partial V(f(x_i, u_i))}{\partial x}$$

为了求解上式，将 g 展开（代价函数有时的符号也可以为l），

$$
\begin{aligned}
\ell(x_i+\delta x_i, u_i+\delta u_i) &= \ell(x_i,u_i)+\delta\ell(x_i,u_i) \\
&\approx \ell(x_i,u_i)+\frac{\partial\ell(x_i,u_i)}{\partial x_i}\delta x_i+\frac{\partial\ell(x_i,u_i)}{\partial u_i}\delta u_i \\
&+\frac{1}{2}\delta x_i^T\frac{\partial^2\ell(x_i,u_i)}{\partial x_i^2}\delta x_i+\frac{1}{2}\delta u_i^T\frac{\partial^2\ell(x_i,u_i)}{\partial u_i^2}\delta u_i \\
&+\frac{1}{2}\delta x_i^T\frac{\partial^2\ell(x_i,u_i)}{\partial x_i\partial u_i}\delta u_i+\frac{1}{2}\delta u_i^T\frac{\partial^2\ell(x_i,u_i)}{\partial u_i\partial x_i}\delta x_i \\
&\triangleq \ell(x_i,u_i)+\ell_{x_i}^T\delta x_i+\ell_{u_i}^T\delta u_i+\frac{1}{2}\delta x_i^T\ell_{x_i^2}\delta x_i+\frac{1}{2}\delta u_i^T\ell_{u_i^2}\delta u_i \\
&+\frac{1}{2}\delta x_i^T\ell_{x_iu_i}\delta u_i+\frac{1}{2}\delta u_i^T\ell_{u_ix_i}\delta x_i
\end{aligned}
$$

所以，继续定义符号

$$
\begin{aligned}
\frac{\partial g}{\partial x} &= g_x^T & \frac{\partial^2 g}{\partial u^2} &= g_{uu} \\
\frac{\partial g}{\partial u} &= g_u^T & \frac{\partial^2 g}{\partial x\partial u} &= g_{xu} \\
\frac{\partial^2 g}{\partial x^2} &= g_{xx} & \frac{\partial^2 g}{\partial u\partial x} &= g_{ux}
\end{aligned}
$$

所以，根据向量的链式求导法则，

$$Q_x^T=\frac{\partial g}{\partial x}+\frac{\partial V(x_{i+1}=f(x_i,u_i))}{\partial x}=g_x^T+\frac{\partial V(x_{i+1})}{\partial x}|_{x_{i+1}}\frac{\partial f}{\partial x}=g_x^T+V_{x_{i+1}}^Tf_x$$

已知Q_x^T，根据矩阵运算性质，计算他的转置矩阵，

$$Q_x=(Q_x^T)^T=\ (g_x^T+V_{x_{i+1}}^Tf_x)^T=g_x+f_x^TV_{x_{i+1}}$$

同理，写出其他几项

$$
\begin{cases}
Q_u=g_u+f_u^TV_{x_{i+1}} \\
Q_{xx}=g_{xx}+f_x^TV_{x_{i+1}x_{i+1}}f_x \\
Q_{xu}=g_{xu}+f_x^TV_{x_{i+1}x_{i+1}}f_u \\
Q_{ux}=g_{ux}+f_u^TV_{x_{i+1}x_{i+1}}f_x \\
Q_{uu}=g_{uu}+f_u^TV_{x_{i+1}x_{i+1}}f_u
\end{cases}
$$

根据 bellman 最优性原理，可以对系统的状态进行反向传播，即从终端状态$\{x_N,u_N\}$计算到$\{x_0,u_0\}$。

首先计算终端的 Q, V。

$$V_N(x_N+\delta x_N)=V_N(x_N)+\frac{\partial V_N}{\partial x}\delta x_N+\frac{1}{2}(\partial x)^T\frac{\partial^2 V_N}{\partial x^2}\delta x$$

对于末端x_N而言，一般不再施加控制, 所以控制量$u_N = 0$, 此时的最优值函数往往是一个平方项，

$$V_N(x_N) = g_N(x_N) = \frac{1}{2}(x_N - x_{rN})^T R_N (x_N - x_{rN})$$

那么，

$$\frac{\partial V_N}{\partial x} = R_N(x_N - x_{rN}) = V_{x_N}^T$$
$$\frac{\partial^2 V_N}{\partial x^2} = R_N = V_{xx_N}$$

这样的话，得到终点的V_x^T，V_{xx}，就可以得到前一个点的Q_u，Q_{xx}，Q_{xu}等值。

接下来继续分析剩余状态。已知动作值函数 Q，为了计算最优控制序列，对δu直接求导[5]，

$$\delta Q = Q_x^T \delta x + Q_u^T \delta u + \frac{1}{2}(\delta x)^T Q_{xx} \delta x + \frac{1}{2}(\delta u)^T Q_{ux} \delta x + \frac{1}{2}(\delta x)^T Q_{xu} \delta u + \frac{1}{2}(\delta u)^T Q_{uu} \delta u$$

为了使得δQ*最小*，令导数为 0，这里需要参考矩阵的求导方式，会存在一些矩阵变换内容，

$$\frac{\partial \delta Q}{\partial \delta u} = Q_u + \frac{1}{2} Q_{ux} \delta x + \frac{1}{2} Q_{xu}^T \delta x + Q_{uu} \delta u = Q_u + Q_{ux} \delta x + Q_{uu} \delta u = 0$$

得到最优δu ，

$${\delta u_k}^* = -Q_{uu}^{-1}\left(Q_{ux} \delta x_k + Q_{u_k}\right) = K \delta x_k + d$$

其中，

$$\begin{cases} K = -Q_{uu}^{-1} Q_{ux} \\ d = -Q_{uu}^{-1} Q_u \end{cases}$$

得到最优δu 之后，将其代入到δQ中，

[5] 这里要参考变分法，欧拉-拉格朗日方程，可以自行查阅最优控制内容

$$
\begin{aligned}
&= \frac{1}{2}\begin{bmatrix} \delta x_i \\ K_i\delta x_i + d_i \end{bmatrix}^T \begin{bmatrix} Q_{x_i^2} & Q_{x_iu_i} \\ Q_{u_ix_i} & Q_{u_i^2} \end{bmatrix} \begin{bmatrix} \delta x_i \\ K_i\delta x_i + d_i \end{bmatrix} + \begin{bmatrix} Q_{x_i} \\ Q_{u_i} \end{bmatrix}^T \begin{bmatrix} \delta x_i \\ K_i\delta x_i + d_i \end{bmatrix} \\
&= \frac{1}{2}\delta x_i^T Q_{x_i^2}\delta x_i + \frac{1}{2}(K_i\delta x_i + d_i)^T Q_{u_i^2}(K_i\delta x_i + d_i) \\
&+ \frac{1}{2}\delta x_i^T Q_{x_iu_i}(K_i\delta x_i + d_i) + \frac{1}{2}(K_i\delta x_i + d_i)^T Q_{u_ix_i}\delta x_i \\
&+ Q_{x_i}^T\delta x_i + Q_{u_i}^T(K_i\delta x_i + d_i) \\
&= \frac{1}{2}\delta x_i^T[Q_{x_i^2} + K_i^T Q_{u_i^2}K_i + Q_{x_iu_i}K_i + K_i^T Q_{u_ix_i}]\delta x_i \\
&+ [Q_{x_i} + K_i^T Q_{u_i^2}d_i + Q_{x_iu_i}d_i + K_i^T Q_u]^T\delta x_i \\
&+ \frac{1}{2}d_i^T Q_{u_i^2}d_i + Q_{u_i}^T d_i
\end{aligned} \tag{20}
$$

知道上面这个式子，我们如何得到V_x^T，V_{xx}？我们呢继续分析。

利用 bellman 最优性原理，Q 和 V 的关系是

$$
\begin{gathered}
Q_k = g_k + V_{k+1} \\
V_k = \min Q_k
\end{gathered}
$$

δQ和δV的关系是什么呢？在Q_k是最优动作值函数时，那么$V_k(x_k) = Q_k(x_k)$，那么即使再增加一个小扰动δx，$V_k(x_k+\delta x) = Q_k(x_k+\delta x)$，

$$V_k(x_k+\delta x) = V_k(x_k) + \delta V_k$$

$$Q_k(x_k+\delta x) = Q_k(x_k) + \delta Q_k$$

所以，$\delta V_k = \delta Q_k$，

观察公式（20），此时方程和δx有关，和δu无关。再结合δV方程，

$$\delta Q = \delta V = \frac{\partial V}{\partial x}\delta x + \frac{1}{2}(\partial x)^T\frac{\partial^2 V}{\partial x^2}\delta x + \textit{高阶项} = V_x^T\delta x + \frac{1}{2}(\partial x)^T V_{xx}\delta x + \Delta V$$

上式也和δx有关，和δu无关，和公式（20）联立，

继续整理，得到

$$
\begin{aligned}
\Delta V(i) &= \quad -\tfrac{1}{2} Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} && \text{(7a)}\\
V_{\mathbf{x}}(i) &= Q_{\mathbf{x}} \ -Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} && \text{(7b)}\\
V_{\mathbf{xx}}(i) &= Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}. && \text{(7c)}
\end{aligned}
$$

这里，ΔV和δx无关，有什么用呢？我们暂时不管，后面会分析。

所以，对于反向传播而言，如果知道$k+1$阶段的$V_x(k+1), V_{xx}(k+1)$，那么就能得到 k 阶段的Q_x, Q_u, Q_{xx}, Q_{xu}，那么继续可以得到 k 阶段的反馈增益K, d，那么继续得到δu_k，然而由于δx_k的具体值是未知的，所以这里并不能确定具体的δu_k值。但是没关系，这里不知道δu_k具体值并不影响整体计算。然后此时再刷新 k 阶段的$V_x(k), V_{xx}(k)$，进入下一轮循环。

整个更新流程是：

1） $$\begin{cases} Q_{u_k} = g_u + f_u^T V_{x_{k+1}} \\ Q_{x_k x_k} = g_{xx} + f_x^T V_{x_{k+1} x_{k+1}} f_x \\ Q_{xu} = g_{xu} + f_x^T V_{x_{k+1} x_{k+1}} f_u \\ Q_{ux} = g_{ux} + f_u^T V_{x_{k+1} x_{k+1}} f_x \\ Q_{uu} = g_{uu} + f_u^T V_{x_{k+1} x_{k+1}} f_u \end{cases}$$

2） $$\begin{cases} K_k = -Q_{uu}^{-1} Q_{ux} \\ d_k = -Q_{uu}^{-1} Q_u \end{cases}$$

3） $$\delta u_k{}^* = -Q_{uu}^{-1}\left(Q_{ux}\delta x_k + Q_{u_k}\right) = K\delta x_k + d$$

4） $$\begin{aligned} V_{\mathbf{x}}(i) &= Q_{\mathbf{x}} \ -Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} \\ V_{\mathbf{xx}}(i) &= Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}. \end{aligned}$$

在反向传播中，并没有更新具体的 x 值和 u 值，但是更新了反馈增益 K,后续在正向传播的时候就可以利用上。

7.5 Forward pass

和常规的 LQR 不同，iLQR 的 forward pass 的时候需要使用非线性的状态转移方程来更新。在反向传播的时候，已经更新了反馈增益序列$\{K_1, K_2, \ldots, K_N\}$，根据方程

$$\delta u_k{}^* = -Q_{uu}^{-1}\left(Q_{ux}\delta x_k + Q_{u_k}\right) = K\delta x_k + d$$

如果已知δx，那么就可以得到$\delta u_k{}^*$。

使用两次迭代轨迹点 k 的差分，更新δx_k，注意，这里并不是使用轨迹上前后两个点的位置差分，而是使用两条轨迹点，例如，在迭代 i-1 次，得到一条轨迹，在迭代 i 次，也会产生一条轨迹，那么在轨迹点 k 处，有

$$\delta x_k = x_k^i - x_k^{i-1}$$

这样就可以更新$\delta u_k{}^*$。因为，

$$\delta u_k = d_k + K_k \delta x_k$$

然后更新控制量u_k，因为，

$$u_k^i = u_k^{i-1} + \delta u_k$$

再根据状态转移方程，使用当前点的状态值和控制值，来更新下一个 k+1 轨迹点的值

$$x_{k+1}^i = f(x_k^i, u_k^i)$$

这样一直增大 k 的值，整条轨迹就得到更新。

7.6 iLQR 的迭代

在一次循环中，轨迹得到更新，那么 cost 和值函数 V 也得到更新，为了让 cost 最小，继续微调δu的值，直到算法收敛。假设已经给出一个较好的初始轨迹，我们对初始轨迹进行调整优化即可。收敛的准则是，

4. Evaluate the trajectory cost before and update the optimization process, $J^{(k)} = J(\mathbf{X}^{(k)}, \boldsymbol{U}^{(k)})$, $J^{(k+1)} = J(\mathbf{X}^{(k+1)}, \boldsymbol{U}^{(k+1)})$.

5. The algorithm is terminated upon convergence based on the change in the cost function:

$$\left|\frac{J^{(k+1)} - J^{(k)}}{J^{(k)}}\right| < \epsilon_J \tag{6.12}$$

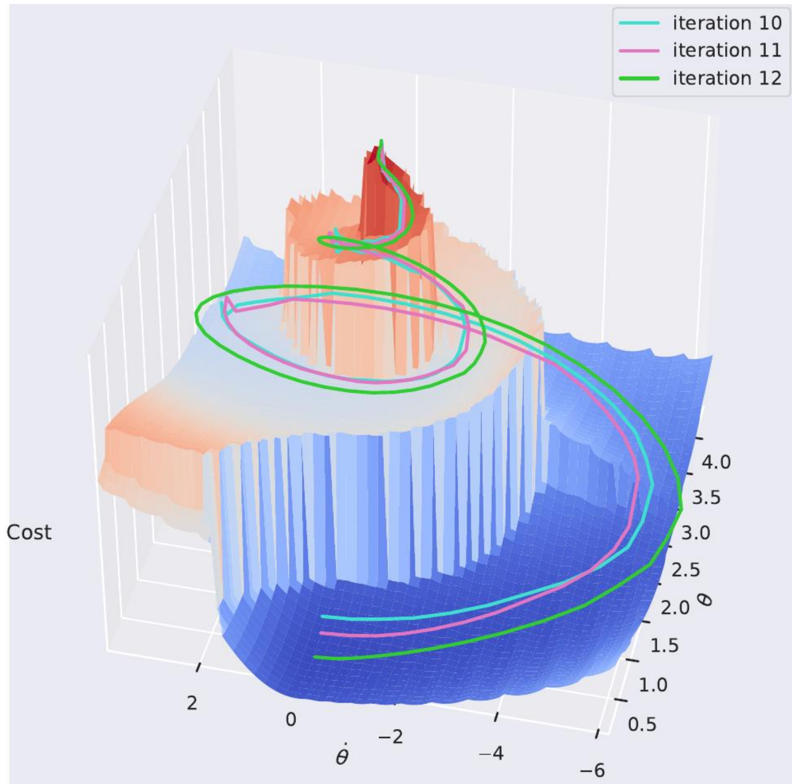
where ϵ_J is the convergence threshold.

即可以使用最优值函数 V 来计算，起点到终点的 cost to go 是 $J = V_{0->N}$，每一次迭代，都有一个 $J = V_{0->N}$，然后使用前后两次迭代的性能指标，判断是否收敛。

经过上面的推导，iLQR 可以存在 3 个循环，

- 一个外循环，负责调整δu的值，从而改善轨迹性能；
- 一个内循环，用来反向传播计算反馈增益；
- 还有一个内环，正向传播计算具体的δu，δx, 以及u, x；

为了形象化 ilqr 过程，可以参考下图



iLQR大循环

0. 迭代次数k=0->M;

1. $f(x,u)$线性化，计算f_x, f_u。如果有必要，代价函数(x, u)展开成二次型;

iLQR内循环

2. 反向传播，$i = N \to 0$,更新反馈增益$\{K_0, K_1, \dots, K_N\}$,

1) $$\begin{cases} Q_{u_i} = g_u + f_u^T V_{x_{i+1}} \\ Q_{x_i x_i} = g_{xx} + f_x^T V_{x_{i+1}x_{i+1}} f_x \\ Q_{xu} = g_{xu} + f_x^T V_{x_{i+1}x_{i+1}} f_u \\ Q_{ux} = g_{ux} + f_u^T V_{x_{i+1}x_{i+1}} f_x \\ Q_{uu} = g_{uu} + f_u^T V_{x_{i+1}x_{i+1}} f_u \end{cases}$$

2) $$\begin{cases} K_i = -Q_{uu}^{-1} Q_{ux} \\ d_i = -Q_{uu}^{-1} Q_u \end{cases}$$

3) $$\delta u_i^* = -Q_{uu}^{-1}(Q_{ux}\delta x_i + Q_{u_i}) = K\delta x_i + d$$

4) $$V_{\mathbf{x}}(i) = Q_{\mathbf{x}} - Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}$$
$$V_{\mathbf{xx}}(i) = Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}.$$

iLQR内循环

3. 正向传播，$i = 0 \to N$,使用非线性方程$\dot{x} = f(x,u)$，更新$\{x_0, x_1, \dots, x_N\}$

$$\delta x_k = x_k^i - x_k^{i-1}$$

$$\delta u_k = d_k + K_k \delta x_k$$

$$u_k^i = u_k^{i-1} + \delta u_k$$

$$x_{k+1}^i = f(x_k^i, u_k^i)$$

4. 计算每一次迭代的代价，J^k, J^{k+1},并计算是否满足收敛标准

$$\left|\frac{J^{(k+1)} - J^{(k)}}{J^{(k)}}\right| < \epsilon_J$$

7.7 iLQR vs DDP

ilqr 和 ddp 主要不同是，$f(x,u)$泰勒展开，一个取一阶，一个取二阶。所以，和 ddp 相比，ilqr 不再具有 2 阶收敛性。然而，使用二阶计算，会增加计算负担，影响最后的计算性能。

7.8 Line search

线搜索，是一种最优化中常用的技术，主要包括两步，

1. 确定迭代方向，梯度方向，牛顿方向，交叉乘子方向；
2. 确定搜索步长，只有选择了合适的步长，才能保证迭代值能够一直下降。如果步长大了，可能难以收敛，甚至发散；如果步长小了，可能收敛太慢。

对于 ilqr，通过方向传播，我们已经确定了迭代方向，为了确定步长，我们观察

7.9 正则化

8 约束处理

8.1 外点罚函数法

8.2 内点罚函数法

8.3 增广拉格朗日乘子法

8.4 交叉乘子法

8.5 SQP 法

9 AL-iLQR

10 基于 ilqr/ddp 的车辆轨迹规划

11 参考

Deriving Differential Dynamic Programming (imgeorgiev.com)

Cs285

Cmu 16-745

12 答疑