下面是 MIT《欠驱动机器人学》第8章 **线性二次型调节器（LQR）** 的中文整理翻译。原文内容很长，我保留主要公式与推导逻辑，省略网页中的交互代码、链接和少量研究性备注。MIT 原文

第8章：线性二次型调节器 LQR

一般来说，求解连续系统的动态规划问题非常困难。但对于“线性系统 + 凸二次代价函数”这类特殊问题，可以比较容易地得到最优解。

最简单、最重要的情形称为：

[\boxed{\text{线性二次型调节器（LQR）}}]

LQR的基本目标是：

> 用尽可能合理的控制能量，把线性时不变系统稳定到原点。

这里的“原点”指：

[\mathbf x=0]

也就是所有状态误差都为零。

一、LQR的基本推导

考虑线性时不变系统：

[\dot{\mathbf x}=A\mathbf x+B\mathbf u]

其中：

- (\mathbf x)：状态向量；
- (\mathbf u)：控制输入；
- (A)：系统自身的动力学；
- (B)：控制输入如何影响系统。

例如两轮平衡机器人可以定义：

[\mathbf x= \begin{bmatrix} x\ \dot{x}\ \phi\ \dot{\phi} \end{bmatrix}]

分别表示位置、速度、倾角和角速度。

1. 代价函数

无限时间LQR要最小化：

[J=\int_0^\infty \left(\mathbf x^TQ\mathbf x+ \mathbf u^TR\mathbf u \right)dt]

这两部分分别表示：

[\mathbf x^TQ\mathbf x]

是状态偏差产生的代价；

[\mathbf u^TR\mathbf u]

是控制输入产生的代价。

因此LQR同时考虑：

- 尽快减小状态误差；
- 不让电机输出过大。

其中要求：

[Q=Q^T\succeq0]

表示 (Q) 是对称半正定矩阵；

[R=R^T\succ0]

表示 (R) 是对称正定矩阵。

简单理解：

- (Q) 决定“多讨厌状态误差”；
- (R) 决定“多讨厌控制输出过大”。

二、什么是最优剩余代价?

假设系统当前处于状态 (\mathbf x)，从现在开始直到未来无穷远，采用最优控制仍然需要付出的代价，记为：

[J^*(\mathbf x)]

它也叫：

- 最优价值函数；
- 最优代价函数；
- cost-to-go，未来剩余代价。

对于LQR，可以证明它具有二次型形式：

[J^*(\mathbf x)=\mathbf x^TS\mathbf x]

其中：

[S=S^T\succeq0]

例如一维情况下：

[J^{*(x)=Sx}2]

离目标越远，未来需要付出的代价越大。

它对状态的梯度是：

[\frac{\partial J^*}{\partial\mathbf x} =2\mathbf x^TS]

这个梯度表示价值函数增大最快的方向。

三、HJB方程

最优控制需要满足 Hamilton–Jacobi–Bellman 方程：

[0=\min_{\mathbf u} \left[\mathbf x^TQ\mathbf x+ \mathbf u^TR\mathbf u+ \frac{\partial J^*}{\partial\mathbf x} (A\mathbf x+B\mathbf u) \right]]

它的直观含义是：

> 当前付出的代价，加上它对未来代价的影响，必须在所有可能输入中最小。

代入：

$$J^*(\mathbf x)=\mathbf x^TS\mathbf x$$

得到关于控制输入 $\mathbf u$ 的二次函数。

因为 R 是正定矩阵，这个函数存在唯一最小值。令其对 $\mathbf u$ 的导数为零：

$$2\mathbf u^TR+2\mathbf x^TSB=0$$

解得最优输入：

$$\mathbf u^* =-R^{-1}B^TS\mathbf x$$

定义：

$$K=R^{-1}B^TS$$

于是：

$$\boxed{\mathbf u^*=-K\mathbf x}$$

这就是LQR的状态反馈控制律。

四、为什么会得到 $u=-Kx$?

价值函数：

$$J^*(\mathbf x)=\mathbf x^TS\mathbf x$$

可以想象成一个碗：

- 碗底是目标状态 $\mathbf x=0$；
- 离碗底越远，代价越大；
- 控制器希望让状态沿着代价下降的方向移动。

最陡下降方向与：

$$-S\mathbf x$$

有关。

但是系统不一定能够向任意状态方向运动，控制输入只能通过 B 影响系统。因此：

$$-B^TS\mathbf x$$

相当于把理想下降方向投影到控制输入真正能够实现的方向。

最后乘上：

$$R^{-1}$$

用来考虑不同控制输入的代价，得到：

$$\mathbf u^{*=-R}{-1}B^TS\mathbf x$$

所以LQR并不是只看当前误差，它还通过 S 考虑了系统动力学和长期代价。

五、代数Riccati方程

将最优控制律代回HJB方程，可以得到：

[A^{TS+SA-SBR{-1}B^TS+Q=0]

这叫作：

[\boxed{\text{连续时间代数Riccati方程，CARE}}]

这里未知的是矩阵 (S)。

求解过程是：

```
已知 A、B、Q、R
        ↓
求解Riccati方程得到 S
        ↓
K = R⁻¹BᵀS
        ↓
u = -Kx
```

Riccati方程中含有：

[SBR^{-1}BTS]

因此它关于 (S) 是非线性的，通常使用数值算法求解，不需要手算。

当系统满足适当的可稳定条件时，可以得到对应的正定解。

MATLAB中通常直接使用：

```
[K,S,P] = lqr(A,B,Q,R);
```

Python中可以使用相关控制库或 Drake。

六、双积分器例子

双积分器模型为：

[\dot{x}*1=x_2][\dot{x}*2=u]

可以理解为：

- (x_1)：位置；
- (x_2)：速度；
- (u)：加速度。

状态空间矩阵为：

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

选择：

[Q=I,\qquad R=1]

得到：

[K= \begin{bmatrix} 1&\sqrt3 \end{bmatrix}]

因此：

[u=-x_1-\sqrt3x_2]

它的含义是：

- 位置偏离目标时，产生返回目标的控制；
- 有运动速度时，产生阻尼作用；
- 最终使位置和速度都趋近于零。

七、用LQR局部稳定非线性系统

真实机器人一般是非线性系统：

[\dot{\mathbf x}=f(\mathbf x,\mathbf u)]

例如倒立摆中的：

[\sin\phi,\qquad \cos\phi]

都是非线性项。

选择一个平衡工作点：

[(\mathbf x_0,\mathbf u_0)]

它必须满足：

[f(\mathbf x_0,\mathbf u_0)=0]

定义相对于平衡点的误差：

[\bar{\mathbf x}=\mathbf x-\mathbf x_0][\bar{\mathbf u}=\mathbf u-\mathbf u_0]

在平衡点附近进行一阶泰勒展开：

[\dot{\bar{\mathbf x}} \approx A\bar{\mathbf x}+B\bar{\mathbf u}]

其中：

[A= \left. \frac{\partial f}{\partial\mathbf x} \right|*{\mathbf x_0,\mathbf u_0}][B= \left. \frac{\partial f}{\partial\mathbf u} \right|*{\mathbf x_0,\mathbf u_0}]

然后用LQR得到：

[\bar{\mathbf u}^*=-K\bar{\mathbf x}]

换回原始变量：

[\boxed{ \mathbf u^* = \mathbf u_0-K(\mathbf x-\mathbf x_0) }]

这就是用LQR在平衡点附近控制非线性机器人。

对于两轮平衡机器人：

- (\mathbf x_0)：机器人竖直、速度为零、位置在目标点；
- (\mathbf u_0)：维持平衡工作点需要的基础输入；
- (\mathbf x-\mathbf x_0)：当前状态相对目标状态的误差。

需要注意：

> 线性化只在工作点附近准确。

如果倾斜角很大，线性模型可能不再准确，LQR也不一定能把机器人救回来。

八、有限时间LQR

前面的LQR考虑：

$$ t\in[0,\infty) $$

如果只考虑有限时间：

$$ t\in[0,t_f] $$

代价函数变成：

$$ J= \mathbf x(t_f)^TQ_f\mathbf x(t_f) + \int_0^{t_f} \left(\mathbf x^TQ\mathbf x+ \mathbf u^TR\mathbf u \right)dt $$

其中：

$$ \mathbf x(t_f)^TQ_f\mathbf x(t_f) $$

表示终点状态的代价。

此时价值函数与时间有关：

$$ J^*(\mathbf x,t)=\mathbf x^TS(t)\mathbf x $$

控制器也随时间变化：

$$ \mathbf u^*(t) =-R^{-1}B^TS(t)\mathbf x(t) $$

即：

$$ \mathbf u^*(t)=-K(t)\mathbf x(t) $$

此时 $(S(t))$ 满足微分Riccati方程：

$$ -\dot S(t) = S(t)A+A^TS(t) -S(t)BR^{-1}B^TS(t)+Q $$

终止条件为：

$$ S(t_f)=Q_f $$

求解时需要从终止时刻 (t_f) 向前积分。

无限时间LQR相当于有限时间解的稳态情况：

$$ \dot S(t)=0 $$

于是微分Riccati方程就变成了代数Riccati方程。

九、时变LQR

如果系统矩阵随时间变化：

$$ \dot{\mathbf x} = A(t)\mathbf x+B(t)\mathbf u $$

那么仍然可以使用LQR。

此时：

$$ Q=Q(t),\qquad R=R(t) $$

也可以随时间变化，控制律为：

$$ \mathbf u=-K(t)\mathbf x $$

这种控制器称为：

$$ \boxed{\text{时变LQR，TVLQR}} $$

十、用LQR稳定一条轨迹

假设非线性系统已经有一条期望轨迹：

$$ \mathbf x_0(t),\qquad \mathbf u_0(t) $$

定义跟踪误差：

$$ \bar{\mathbf x}(t) = \mathbf x(t)-\mathbf x_0(t) $$ $$ \bar{\mathbf u}(t) = \mathbf u(t)-\mathbf u_0(t) $$

沿期望轨迹进行线性化：

$$ \dot{\bar{\mathbf x}} \approx A(t)\bar{\mathbf x} +B(t)\bar{\mathbf u} $$

得到控制器：

$$ \bar{\mathbf u}^* =-K(t)\bar{\mathbf x} $$

因此实际输入为：

$$ \boxed{ \mathbf u^*(t) = \mathbf u_0(t) -K(t)\bigl(\mathbf x-\mathbf x_0(t)\bigr) } $$

它由两部分组成：

$$ \mathbf u_0(t) $$

是沿期望轨迹运动所需的前馈输入；

$$ -K(t)(\mathbf x-\mathbf x_0(t)) $$

是修正轨迹偏差的反馈输入。

这适合用于机器人：

- 行走轨迹；
- 跳跃；
- 无人机飞行；
- 机械臂轨迹跟踪；
- 动态平衡。

十一、线性二次型跟踪 LQT

标准LQR把系统控制到：

[\mathbf x=0,\qquad \mathbf u=0]

如果希望跟踪期望状态和期望输入：

[\mathbf x_d(t),\qquad \mathbf u_d(t)]

可以使用代价函数：

[J= (\mathbf x(t_f)-\mathbf x_d(t_f))^T Q_f (\mathbf x(t_f)-\mathbf x_d(t_f))][+\int_0^{t_f} \left[(\mathbf x-\mathbf x_d)^TQ(\mathbf x-\mathbf x_d) + (\mathbf u-\mathbf u_d)^TR(\mathbf u-\mathbf u_d) \right]dt]

它惩罚的是：

- 实际状态与期望状态之间的误差；
- 实际输入与期望输入之间的误差。

这称为：

[\boxed{\text{线性二次型跟踪，LQT}}]

它比简单的 (u=-Kx) 更适合跟踪非零目标或随时间变化的轨迹。

十二、离散时间LQR

实际单片机是按照固定周期运行的，因此更常见的是离散系统：

[\mathbf x[n+1] = A\mathbf x[n]+B\mathbf u[n]]

其中 (n) 是控制周期编号。

有限时间代价为：

[J= \sum_{n=0}^{N-1} \left[\mathbf x[n]^TQ\mathbf x[n] + \mathbf u[n]^TR\mathbf u[n] \right]]

最优控制律仍然是：

[\mathbf u^*[n]=-K[n]\mathbf x[n]]

但反馈矩阵为：

[K[n] = (R+B^{TS[n]B){-1}B^TS[n]A]

对应的Riccati差分方程为：

[S[n-1] = Q+A^TS[n]A][- A^TS[n]B (R+B^{TS[n]B){-1} B^TS[n]A]

无限时间情况下，(S[n]) 收敛到固定矩阵 (S)，满足离散代数Riccati方程：

[S= Q+A^TSA - A^{TSB(R+B^{TSB)^{-1}B^{TSA}]

实际数字控制器通常应该根据离散模型计算离散LQR增益，而不是直接把连续时间增益照搬进单片机。

十三、状态和输入约束

标准LQR通常没有严格处理以下约束：

[|u|\leq u_{\max}]

例如：

- 电机最大电流；
- PWM最大值；
- 最大速度；
- 最大倾角；
- 关节限位。

标准LQR计算出的控制量可能超过电机能力。实际使用时虽然可以限幅，但限幅会改变理论闭环特性。

如果需要在优化中严格考虑这些约束，通常使用：

[\boxed{\text{模型预测控制 MPC}}]

MPC会在每个控制周期求解一段有限时间内的最优控制序列，并只执行第一个控制量，然后下一周期重新计算。

十四、整章最重要的结论

已知系统：

[\dot{\mathbf x}=A\mathbf x+B\mathbf u]

选择：

[Q\succeq0,\qquad R\succ0]

求解Riccati方程：

[A^{TS+SA-SBR}{-1}B^TS+Q=0]

计算：

[K=R^{-1}BTS]

然后使用：

[\boxed{\mathbf u=-K\mathbf x}]

对于非零平衡点：

[\boxed{ \mathbf u = \mathbf u_0-K(\mathbf x-\mathbf x_0) }]

对于期望轨迹：

[\boxed{ \mathbf u(t) = \mathbf u_0(t) -K(t)(\mathbf x-\mathbf x_0(t)) }]

一句话概括：

> LQR利用系统模型，在“状态误差尽量小”和“电机输出不要太大”之间寻找最优折中，最终自动计算状态反馈参数 (K)。