ILQR算法及其应用

2022.12.01

目 录

CONTENT

01 引言--状态空间方程的定义

状态空间是指在系统中可决定系统状态、最小数目变量的有序集合[1]。而所谓状态空间则是指该系统全部可能状态的集合。

状态空间方程即为一种将物理系统表示为一组输入、输出及状态的数学模式，而输入、输出及状态之间的关系可用许多一阶微分方程组来描述。

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)\\ \mathbf{y}(t) &= C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)\end{aligned}$$

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}(t, x(t), u(t))\\ \mathbf{y}(t) &= \mathbf{h}(t, x(t), u(t))\end{aligned}$$

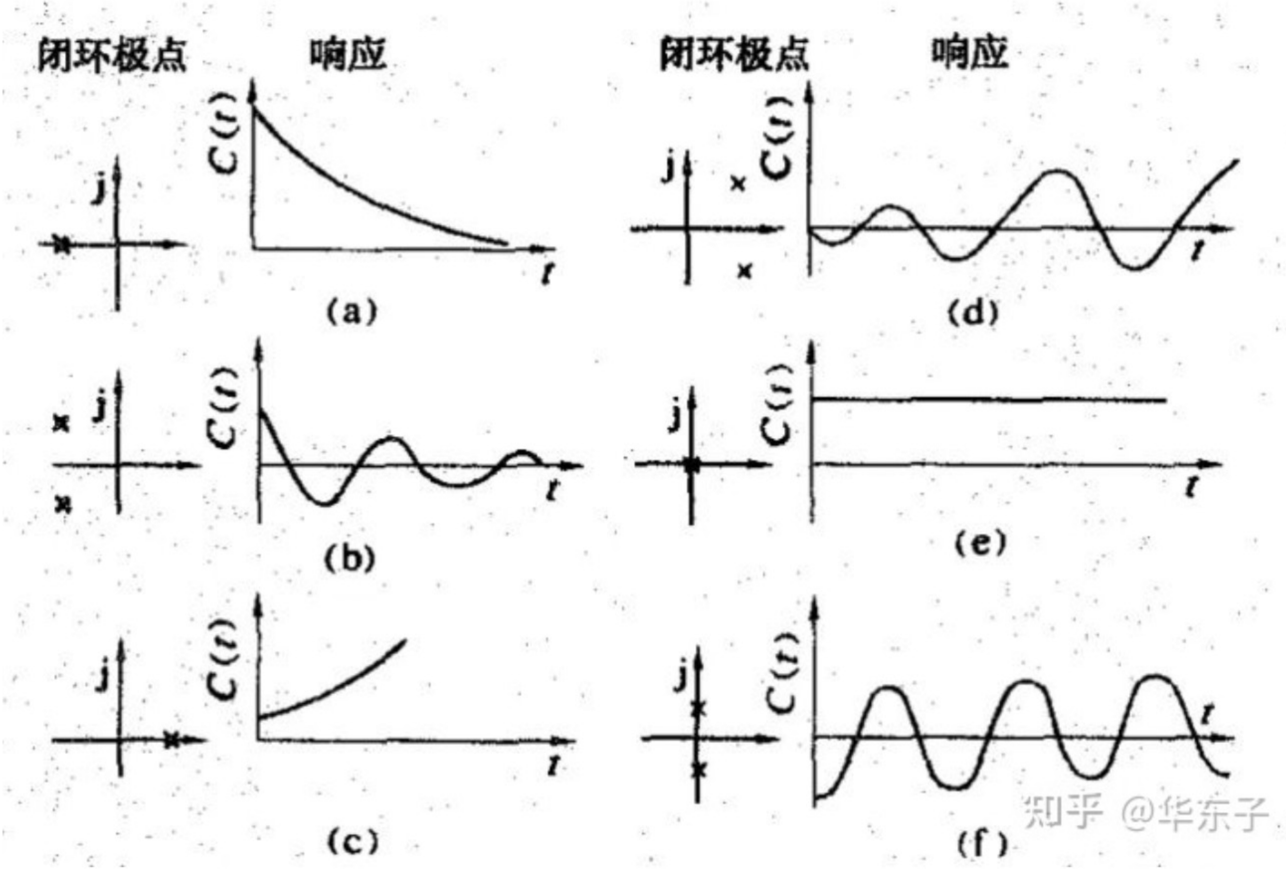
$$\begin{cases}\dot{s} = v\\ \dot{v} = T/mR - gsin(\varphi) - \mu gcos(\varphi) - kv_r^2\end{cases}$$

$$\begin{bmatrix}\dot{s}\\ \dot{v}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -kv_r/m\end{bmatrix}\begin{bmatrix}s\\ v\end{bmatrix} + \begin{bmatrix}0\\ 1/mR\end{bmatrix}T + \begin{bmatrix}0\\ -gsin(\varphi) - \mu gcos(\varphi)\end{bmatrix}$$

A矩阵　X向量　B矩阵　常数项：参考冲激输入信号

1. Get the physics right.
2. After that, it is all mathematics.

01 引言--状态空间方程与稳定性



不同的特征根，系统的响应是不同，包括稳定、不稳定、临界稳定三种状态。

$$\dot{x} = A'x$$

系统稳定的条件：

特征根（闭环极点）有负实部

注：简单的说，这些线性系统的解是以e为底的指数函数。（时间总是大于0）如果特征根实部是负数，随着时间增大，函数衰减。所以会看到一个稳定的解。

01 引言--状态空间方程与二次型最优控制

对于已知状态空间方程 $\dot{x} = Ax + Bu$ 设计状态反馈控制律为 $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t)$

$$\Rightarrow \quad \dot{x} = Ax + B(-Kx) = \boxed{(A - BK)}x \quad A_{cl}$$

$K \Rightarrow$ 改变闭环矩阵 A_{cl} 的特征值 $\Rightarrow$ 控制系统的变化

$$J = \int_0^\infty |e|\,dt = \int_0^\infty |x_o(t) - x_i(t)|\,dt$$

$$J = \int_0^\infty e^2\,dt = \int_0^\infty \left(x_o(t) - x_i(t)\right)^2 dt$$

为什么用二次型函数？
绝对值的导数不连续，数学上不太好处理.用平方也能表征距离，而且数学处理上的性能非常好。

定义二次型代价函数 $J = \int_0^\infty \boldsymbol{x}^T \boldsymbol{x} dt$

$\Rightarrow$ 当J最小化的时候，对应的K，最优的特征值

参考：https://zhuanlan.zhihu.com/p/58134063

二次型最优控制本质上是通过反馈控制来找到闭环系统最优的特征值

目 录

CONTENT

02 LQR与ILQR算法原理--LQR算法原理

最优的特征值 ⟹ 最优的状态反馈K　　如何求解最优的状态反馈系数矩阵K？

对于闭环系统状态空间方程 $\dot{x} = Ax + Bu$　　$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t)$

定义二次型损失函数为：$J = \int_0^\infty \boxed{(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})} dt$　控制能量最小和误差最小的权衡

一种离散化的形式

这个LQR问题如何求解？　$x_{t+1} = Ax_t + Bu_t, x_0 = x_{init}$　　$J(U) = \sum_{\tau=0}^{N-1}(x_\tau^T Q x_\tau + u_\tau^T R u_\tau) + x_N^T Q_f x_N$

上式可以改写为如下形式：

一种紧凑的形式

$$x_{t+1} = F * \begin{bmatrix} x_t \\ u_t \end{bmatrix}，其中F = [A \quad B]$$

$$J(x,u) = \sum_{t=0}^{N}\left(\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T C_t \begin{bmatrix} x_t \\ u_t \end{bmatrix}\right)，其中C_t = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}$$

根据二次损失与线性转移关系，将 J 展开之后可以得到一个关于 x0 与 u0,u1,u2,…,uT 的正定二次函数。根据正定二次函数的凸性，必有唯一的极值。

02 LQR与ILQR算法原理--LQR算法原理

$$x_{t+1} = F_t * \begin{bmatrix} x_t \\ u_t \end{bmatrix} \qquad J(x,u) = \sum_{t=0}^{N} \left(\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T C_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} \right)$$

这个LQR问题如何求解？

动态规划算法：解决多阶段决策过程最优化的一种有效的数学方法

定义价值函数V，其中包括$V(x_t, u_t)$以及$V(x_t)$

$V(x_t, u_t)$：表示在状态x_t，选择动作u_t，后续的最小损失

$V(x_t)$：表示在状态x_t对应的最小损失，等于选择最优动作u_t对应的$V(x_t, u_t)$

令t=N $\quad V(x_N) = \begin{bmatrix} x_N \\ u_N \end{bmatrix}^T C_N \begin{bmatrix} x_N \\ u_N \end{bmatrix} = x_N^T Q_N x_N$

令t=N-1 $\quad V(x_{N-1}, u_{N-1}) = \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}^T C_{N-1} \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix} + V(x_N)$

$$= \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}^T C_{N-1} \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}^T F_{N-1}^T C_N F_{N-1} \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}$$

$$= \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}^T (C_{N-1} + F_{N-1}^T C_N F_{N-1}) \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}$$

参考：https://jonathan-hui.medium.com/rl-lqr-ilqr-linear-quadratic-regulator-a5de5104c750

02 LQR与ILQR算法原理--LQR算法原理

令t=N $\quad V(x_N) = \begin{bmatrix} x_N \\ u_N \end{bmatrix}^T C_N \begin{bmatrix} x_N \\ u_N \end{bmatrix} = x_N^T \boxed{Q_N} x_N$ $\quad \to P_N$

$P_{N-1} \Rightarrow \begin{bmatrix} P_{x_{N-1},x_{N-1}} & P_{x_{N-1},u_{N-1}} \\ P_{u_{N-1},x_{N-1}} & P_{u_{N-1},u_{N-1}} \end{bmatrix}$

令t=N-1 $\quad V(x_{N-1}, u_{N-1}) = \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}^T \boxed{(C_{N-1} + F_{N-1}^T P_N F_{N-1})} \begin{bmatrix} x_{N-1} \\ u_{N-1} \end{bmatrix}$

$V(x_{N-1}, u_{N-1})$对u_{N-1}求偏导：$P_{u_{N-1},u_{N-1}} u_{N-1} + P_{u_{N-1},x_{N-1}} x_{N-1} = 0$

$u_{N-1} = K_{N-1} * x_{N-1}$其中，$K_{N-1} = P_{u_{N-1},u_{N-1}}^{-1} P_{u_{N-1},x_{N-1}}$

$V(x_{N-1}) = V(x_{N-1}, u_{N-1}) |\, u_{N-1} = K_{N-1} * x_{N-1}$ $\quad P_{N-2}$

令t=N-2 $\quad V(x_{N-2}, u_{N-2}) = \begin{bmatrix} x_{N-2} \\ u_{N-2} \end{bmatrix}^T \boxed{(C_{N-2} + F_{N-2}^T P_{N-1} F_{N-2})} \begin{bmatrix} x_{N-2} \\ u_{N-2} \end{bmatrix}$

$u_{N-2} = K_{N-2} * x_{N-2}$其中，$K_{N-2} = P_{u_{N-2},u_{N-2}}^{-1} P_{u_{N-2},x_{N-2}}$

初始化$P_N = C_N$，
从N-1到0向后迭代求出K_{N-1}到K_0

$$\begin{pmatrix} P_t = C_t + F_t^T P_{t+1} F_t \\ K_t = P_{u_t,u_t}^{-1} P_{u_t,x_t} \end{pmatrix}$$

$\Longrightarrow$

已知初始状态x_0，
从0到N向前迭代求出u_t到x_t

$$\begin{pmatrix} u_t = K_t * x_t \\ x_{t+1} = F_t * \begin{bmatrix} x_t \\ u_t \end{bmatrix} \end{pmatrix}$$

02 LQR与ILQR算法原理--ILQR算法原理

$$x_{t+1} = F_t * \begin{bmatrix} x_t \\ u_t \end{bmatrix} \qquad J(x,u) = \sum_{t=0}^{N} \left(\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T C_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} \right)$$

（Linear Quadratic Regulator）

局限性：
（1）系统必须是线性的，而实际系统大多是非线性的；
（2）损失函数必须是二次型。

ILQR：迭代LQR，将LQR扩展到非线性系统

$$x_{t+1} = f(x_t, u_t) \qquad J(x,u) = \sum_{t=0}^{N} c(x_t, u_t),\ f(x_t, u_t)\text{与}c(x_t, u_t)\text{is Nonlinear}$$

第一步：泰勒展开，线性化

初始化的状态通过参考轨迹/参考线计算得到

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

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

02 LQR与ILQR算法原理--ILQR算法原理

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

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

$$f(\mathbf{x}_t,\mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

LQR形式

$$c(\mathbf{x}_t,\mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

初始化$P_N = C_N$，从N-1到0向后迭代求出K_{N-1}到K_0

$$\begin{cases} P_t = C_t + F_t^T P_{t+1} F_t \\ p_t = c_t + F_t^T P_{t+1} f_t + F_t^T p_{t+1} \\ K_t = P_{u_t,u_t}^{-1} P_{u_t,x_t} \\ k_t = P_{u_t,u_t}^{-1} p_{u_t} \end{cases}$$

第二步：LQR求解

已知初始状态x_0，从0到N向前迭代求出u_t到x_t

$$\begin{cases} u_t = K_t * (x_t - \hat{x}_t) + k_t + \hat{u}_t \\ x_{t+1} = f(x_t, u_t) \end{cases}$$

生成的$u_0 \cdots u_{N-1}$，$x_0 \cdots x_N$作为下次迭代的$\hat{u}_0 \cdots \hat{u}_{N-1}$，$\hat{x}_0 \cdots \hat{x}_N$

第三步：迭代（滚动优化）

02 LQR与ILQR算法原理--算法对比分析

算法	问题公式化	求解耗时	目标函数	约束
LQR	$x_{t+1} = Ax_t + Bu_t + C$ $J = \sum_{t=0}^{T}(x_t' Q x_t + u_t' R u_t + x_t' P u_t + q' x_t + r' u_t)$	0.1ms级别	二次型函数	线性等式约束
QP	$\min f(\mathbf{x}) = 1/2\mathbf{x}^{\mathrm{T}}\mathrm{H}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x}, \mathbf{x} \in \mathrm{R}^{\mathrm{n}}$ $\mathrm{s.t.}\ \mathbf{Ax} \le \mathbf{b}$	ms级别	二次型函数	线性不等式约束
ILQR	状态转移关系：$x_{t+1} = f(x_t, u_t)$ 目标：极小化总损失 $J = \sum_{t=0}^{T} C(x_t, u_t)$	10ms级别	其他非线性函数	非线性等式约束
SQP	$\min \mathrm{f(X)}$ $\mathrm{s.t.}\quad \mathrm{g_u(X)} \le 0\ \ (\mathrm{u} = 1, 2, ..., \mathrm{p})$ $\mathrm{h_v(X)} = 0\ \ (\mathrm{v} = 1, 2, ..., \mathrm{m})$	100ms级别	其他非线性函数	非线性不等式约束

LQR和ILQP其实是特殊的QP和SQP问题，QP问题求解一般采用OSQP(交替方向乘子法ADMM)、qpOASES，而LQR/ILQR采用动态规划的方法、求解速度更快。
OSQP和qpOASES参考：https://zhuanlan.zhihu.com/p/464676135

目 录

CONTENT

03 ILQR算法典型应用实例--轨迹规划

[1] J. Chen, W. Zhan, and M. Tomizuka, “Autonomous Driving Motion Planning with Constrained Iterative LQR,” IEEE Transactions on Intelligent Vehicles, pp. 1–1, 2019.（伯克利）

Problem 2 (Discrete-time Finite-horizon Motion Planning Problem):

$$x^*, u^* = \arg\min_{x,u} \left\{ \phi(x_N) + \sum_{k=0}^{N-1} L^k(x_k, u_k) \right\} \quad (2a)$$

$$\text{s.t.} \quad x_{k+1} = f^k(x_k, u_k), k = 0, 1, \ldots, N-1 \quad (2b)$$

$$x_0 = x_{start} \quad (2c)$$

$$g^k(x_k, u_k) < 0, k = 0, 1, \ldots, N-1 \quad (2d)$$

$$g^N(x_N) < 0 \quad (2e)$$

1 不等式约束不满足ILQR形式

2 采用罚函数的方法表示

3 将约束转换到目标函数中

An alternative way to handle constraints is introducing penalties. The basic idea is to use a barrier function to shape the constraint functions (2d) and (2e):

$$c(x, u) = b(g(x, u)) \quad (11)$$

Then the barrier function is added to the objective function. The ideal barrier function is the indicator function:

$$b^*(g(x, u)) = \begin{cases} \infty, g(x, u) \geq 0 \\ 0, g(x, u) < 0 \end{cases} \quad (12)$$

In this paper, we use a logarithmic barrier function:

$$b(g(x, u)) = -\frac{1}{t} \log(-g(x, u)) \quad (14)$$

03 ILQR算法典型应用实例--轨迹规划

Problem 4 (Motion Planning Problem for Autonomous Driving):

$$x^*, u^* = \arg\min_{x,u} \phi(x_N) + \sum_{k=0}^{N-1} L^k(x_k, u_k) \quad (30a)$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k), k = 0, 1, \ldots, N-1 \quad (30b)$$

$$x_0 = x_{start} \quad (30c)$$

$$d(x_k, O_j^k) > 0, k = 1, 2, \ldots, N. j = 1, 2, \ldots, m \quad (30d)$$

$$\underline{u} < u_k < \overline{u}, k = 1, 2, \ldots, N-1 \quad (30e)$$

2 目标函数

1 车辆运动学模型约束

$$\begin{aligned} v_1 &= v_0 + aT_r \\ \theta_1 &= \theta_0 + \int_0^l \kappa ds = \theta_0 + \kappa l \\ x_1 &= x_0 + \int_0^l \cos(\theta_0 + \kappa s) ds = x_0 + \frac{\sin(\theta_0+\kappa l) - \sin(\theta_0)}{\kappa} \\ y_1 &= y_0 + \int_0^l \sin(\theta_0 + \kappa s) ds = x_0 + \frac{\cos(\theta_0) - \cos(\theta_0+\kappa l)}{\kappa} \end{aligned} \quad (31)$$

We can now write down the vehicle dynamic equation as:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{k+1} \\ \theta_{k+1} \end{bmatrix} = f\left(\begin{bmatrix} x_k \\ y_k \\ v_k \\ \theta_k \end{bmatrix}, \begin{bmatrix} a \\ \kappa \end{bmatrix} \right) \quad (32)$$

1) Acceleration: The term:

$$c_k^{acc} = w_{acc} u_k^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} u_k$$

3) Velocity Tracking: The term:

$$c_k^{vel} = w_{vel} \left(\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T x_k - v^r \right)^T \left(\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T x_k - v^r \right)$$

3 避免碰撞约束

1) Acceleration Constraint: The acceleration is bounded according to the engine force limit and the braking force limit. This constraint is formulated as:

4 执行器约束

$$a_{low} \le u_k^T \begin{bmatrix} 1 \\ 0 \end{bmatrix} \le a_{high} \quad (38)$$

where $a_{high} > 0$ is the largest acceleration the engine can provide, and $a_{low} < 0$ is the largest deceleration the brake can provide.

2) Steering Angle Constraint: The steering angle is bounded according to the steering angle limit:

$$-\bar{s} \le u_k^T \begin{bmatrix} 0 \\ 1 \end{bmatrix} \le \bar{s} \quad (39)$$

where $\bar{s}$ is the largest steering angle value of the vehicle.

（1）*引入了车辆运动学模型约束；
（2）求解速度更快；

2) Steering Angle: The term:

$$c_k^{steer} = w_{steer} u_k^T \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} u_k$$

4) Reference Tracking: The term c_k^{ref} is the cost to penalize the distance from the ego vehicle to the reference trajectory. A commonly used method to define reference is to set a sequence of reference points $x^r = [x_0^r, x_1^r, \ldots, x_N^r]$. The cost penalizes the distance from the trajectory point x_k of the ego vehicle to the reference trajectory point x_k^r:

$$c_k^{ref} = (x_k - x_k^r)^T Q_k^r (x_k - x_k^r) \quad (36)$$

3) Obstacle Avoidance Constraint: The obstacle avoidance constraint includes avoiding the moving obstacles and static obstacles. For example, how to overtake the front vehicle safely, and how to avoid parked vehicles on the roadside. The non-convexity of the constraints makes it difficult to deal with.

In this paper the obstacles are represented by polygons, and the vehicles are represented by rectangles. The core of the collision avoidance constraints (30d) is the distance from ego vehicle to the polygon O_j, which can be calculated as:

$$d(x_k, O_j) = \min_{y \in O_j} d(x_k, y), y \notin O_j \quad (40)$$

TABLE I
RUNTIME COMPARISON WITH SQP

Algorithm	Time/Iters (s)	# of Iters	Total Time (s)
SQP	0.3477	43	14.95
CILQR	0.0086	21	0.18

目 录

CONTENT

04 结合BUS业务的思考--精准控制

业务背景： 特定场景精准控制方案（精准进站、自动托挂钩（拖车）；自动充电（香港机场）；自动泊车）

精准控制：更适合控制跟随的轨迹+规划控制配合

（1）在少约束的场景下，采用ILQR进行轨迹规划，可以同时考虑速度和path；

速度和path配合更好：转急弯前降低速度，起步转大弯的时候低速过弯；

规划结果更符合车辆运动学，不超出车辆执行能力。

（2）精准控制场景下，强化规划和控制的配合，具备挪车微动的功能。

规划和控制配合更好：低速高精度跟随，精准停车，及时replan；

THANKS