百度 Apollo 自动驾驶控制算法分析

慕博文

July 16 2019

I. 算法简介

Apollo 是由百度发起的一个高效、灵活的基于自动驾驶的测试和研发的平台。Apollo 的主要逻辑为局部规划 + 轨道跟踪。在局部规划时，由低精度模型 + 长周期规划组成。在跟踪控制层面上是有高精度模型 + 短周期规划组成。本文主要是对于 Apollo 控制算法中的 MPC（模型预测控制）和 LQR（线性二次调节器）这两种路径跟踪控制器进行分析与比较。

MPC（模型预测控制）是一种先进的过程控制方法，在满足一定约束条件的前提下，用于实现控制的过程。他的实现依赖于过程的动态线性模型。在控制时域内，MPC 将主要对当前时刻进行优化，但同时也会考虑未来时刻，求取当前时刻的最优解，然后反复优化，从而实现在整个时域内的优化求解。

LQR（线性二次调节器）是现代控制理论中发展十分成熟的一种状态空间设计方法。LQR 算法可得到线性反馈的最优控制规律，易于形成闭环最优控制。LQR 的最优设计是指设计出的状态反馈器 K 要使二次型目标函数 J 取最小值，而 K 有权矩阵 Q 与 R 唯一决定，在汽车不同的速度下，Q 与 R 的不同会对算法的效果有着重要的影响，所以 Q，R 的选值十分重要。

MPC 的研究对象是以现代控制理论中的以状态空间方程的形式给出的线性系统。对于 MPC 控制器，他的研究对象既可以是线性系统，也可以是非线性系统。LQR 的研究对象是以状态空间方程形式给出的线性系统。然而由于对效率和时耗的考虑，通常会将非线性系统线性化进行计算。

在 Apollo 算法中，LQR 和 MPC 都选用单车动力学模型作为研究对象。单车系统为非线性系统，但 LQR 和 MPC 控制器的目的是为了求最优控制解，在优化过程中都会将其线性化，所以也可以说 Apollo 中研究的对象都为线性系统。

II. 汽车数学模型

Apollo 的 MPC 和 LQR 路径跟踪算法中都考虑到的汽车运动学（kinematic）和动力学模型（dynamic）的影响。由于汽车动力学模型更能分析车辆的平顺性和车辆操控的稳定性，其更能体现出车辆轮胎及其相关部件的受力情况。在 Apollo 的算法中，都主要由动力学模型进行纵向速度控制（控制轮胎转速）和横向航向控制（控制轮胎转角）。下文将对模型进行分析得出状态方程，并对其进行离散化。

A. 汽车动力学模型

百度 Apollo 使用单车模型的假设下并做出如下假设

- 只考虑纯侧偏轮胎特性，忽略轮胎里的纵横向耦合关系
- 使用单车模型（在前后轴，左右轮胎合并成一个轮胎），不考虑载荷的左右转移
- 忽略纵向空气动力学

模型如图 1.

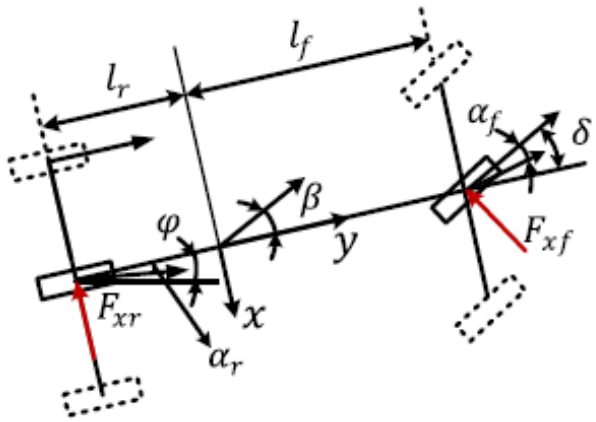


Fig. 1: 汽车动力学模型

根据牛顿定律，对沿 x 轴，y 轴和 z 轴做受力分析：在 x 轴方向上：

$$ma_x = F_{xf} + F_{xr} \tag{1}$$

在 y 轴方向上：

$$ma_y = F_{yf} + F_{yr} \tag{2}$$

在 z 轴方向上：

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{3}$$

在这里 I_z 是车辆绕 z 轴转动的转动惯量，l_f 和 l_r 是车辆已重心为重心的前后悬长度，m 是整车质量，a_x 和 a_y 是在 x 轴和 y 轴上的加速度。

结合对车辆横向动力学的分析1, 在 y 轴的加速度 a_y 由两部分组成：y 轴方向的唯一相关的加速度 $\ddot{y}$ 和向心加速度 $V_x\dot{\psi}$

$$a_y = \ddot{y} + V_x\dot{\psi} \tag{4}$$

带入公式 2 可变为：

$$m(\ddot{y} + V_x\dot{\psi}) = F_{yf} + Fyr \tag{5}$$

对于轮胎转向力，假设由于轮胎受到的横向压力，轮胎会有一个很小的滑移角。轮胎横向所受的力假设为一个关

于轮胎滑移角的线性的方程。对于前后轮胎所受的力 F_{yf}, F_{yr} 和轮胎侧滑角 a_f, a_r 定义为:

$$F_{yf} = C_f \times a_f = C_f \times (\delta - \theta_{V_f}) \tag{6}$$

$$F_{yr} = C_r \times a_r = C_r \times (-\theta_{V_f}) \tag{7}$$

q 其中 δ 是前轮转角; C_f 和 C_r 是前后轮的侧偏刚度(cornering stiffness)。由于车辆前后各两个轮胎, 所以受力要乘 2。

θ_{V_f} 和 θ_{V_r} 计算为:

$$tan(\theta_{V_f}) = \frac{V_y + l_f\dot{\psi}}{V_x} \tag{8}$$

$$tan(\theta_{V_r}) = \frac{V_r + l_f\dot{\psi}}{V_x} \tag{9}$$

在小角度时 $tan(\theta) = \theta$,$V_y = \dot{y}$。综合上述公式, 可得一个关于车辆横向距离和航向角的状态方程:

$$\ddot{y} = -\frac{2C_{ar} + 2C_{af}}{mV_x}\dot{y} + (-V_x - \frac{2C_{af}l_f - C_{ar}l_r}{mV_x})\dot{\psi} + \frac{C_f}{m}\delta \tag{10}$$

$$\ddot{\psi} = -\frac{2C_{af}l_f - 2C_{ar}l_r}{I_zV_x}\dot{y} + \frac{2C_{af}l_f^2 + 2C_{ar}l_r^2)}{I_zV_x}\dot{\psi} + \frac{2C_{af}l_f}{I_z}\delta \tag{11}$$

B. 方向盘控制模型

由于车辆的横向控制主要是通过控制轮胎转角实现, 对于驾驶原来说, 最直接的就是控制方向盘的角度。因此, 在建立车辆动力学模型时, 可以用相对道路的方向和距离误差为状态变量的动力学模型1。在百度控制算法中, e_1 为横向误差, 即车辆质心距车道中心线的距离, e_2 为航向误差, 车辆纵向速度为 V_x, 车转弯半径为 R。其中期望转角速度:

$$\dot{\psi}_{des} = \frac{V_x}{R} \tag{12}$$

所需横向加速度:

$$a_{y_{des}} = \frac{V_x^2}{R} = V_x\dot{\psi}_{des} \tag{13}$$

横向加速度误差:

$$\ddot{e}_1 = a_y - a_{y_{des}} = \ddot{y} + V_x(\dot{\psi} - \dot{\psi}_{des}) \tag{14}$$

横向速度误差:

$$\dot{e}_1 = \dot{y} + V_x(\psi - \psi_{des}) \tag{15}$$

航向误差:

$$e_2 = \psi - \psi_{des} \tag{16}$$

根据之前的数学公式, 可以得到 4 个横向控制状态变量, 横向误差 (lateral error), 横向误差率 (lateral error rate), 航向误差 (heading error), 航向误差率 (heading error rate) 分别对应状态:

$$X_c = \begin{bmatrix} e_1 & \dot{e}_1 & e_2 & \dot{e}_2 \end{bmatrix}^T \tag{17}$$

状态方程为:

$$\dot{X} = A_cX_c + B_c\delta + C_c\dot{\psi}_{des} \tag{18}$$

其中

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{af}+2C_{ar}}{mV_x} & \frac{2C_{af}+2C_{ar}}{m} & -\frac{2C_{af}l_f+2C_{ar}l_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_{af}l_f-2C_{ar}l_r}{I_zV_x} & \frac{2C_{af}l_f-2C_{ar}l_r}{I_z} & -\frac{2C_{af}{l_f}^2+2C_{ar}{l_r}^2}{I_zV_x} \end{bmatrix}$$

$$B_c = \begin{bmatrix} 0 \\ \frac{2C_{af}}{m} \\ 0 \\ \frac{2C_{af}l_f}{I_z} \end{bmatrix}, C_c = \begin{bmatrix} 0 \\ -\frac{2C_{af}l_f-2C_{ar}l_r}{mV_x} - V_x \\ 0 \\ -\frac{2C_{af}{l_f}^2+2C_{ar}{l_r}^2}{I_zV_x} \end{bmatrix} \tag{19}$$

对于横向误差的计算, 从图2可得, 横向误差 e_1 为:

$$e_1 = y' = dy * cos\theta_{des} - dx * sin\theta_{des} \tag{20}$$

横向误差率 $\dot{e}_1$:

$$\dot{e}_1 = V_x * sin\Delta\theta = V_x * \sin e_2 \tag{21}$$

航向误差 e_2 为:

$$e_2 = \theta - \theta_{des} \tag{22}$$

航向误差率 $\dot{e}_2$ 为:

$$\dot{e}_2 = \dot{\theta} - \dot{\theta}_{des} \tag{23}$$

其中 $\dot{\theta}$ 为车辆的转角速度, 是由车身传感器测得, $\dot{\theta}_{des}$ 是期望车辆转角速度, 由规划参数可得:

$$\dot{\theta}_{des} = V_{des} * k_{des} \tag{24}$$

其中 V_{des} 为期望车速, k_{des} 为期望道路曲率。

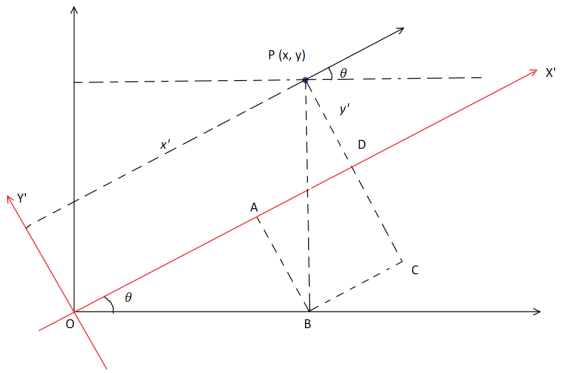


Fig. 2: 坐标变换

III. 百度 MPC 算法

MPC 算法是一种在满足一定约束条件的前提下, 用来实现过程控制, 通过对汽车模型的分析和预测来进行滚动优化, 尽可能减小模型输出与参考值的差距。在控制时域内, 对当前时刻进行优化, 并同时考虑未来的时刻, 求取当前时刻的最优控制解, 从而实现对整个时域进行优化求解。

在新的采样时刻，基于被控对象的实际输出，对输出进行校正，然后进行新的优化。防止模型失配或外界干扰导致的输出与期望差距过大。

A. 横纵控制状态变量

Apollo 的状态控制变量既考虑了横向的误差，也考虑了在 frenet 坐标下纵向的误差, 如图2。在已有的横向状态控制变量中加入了两个纵向控制变量：纵向位置误差 (station error) 和速度误差（speed error）。其中位置误差的计算方法为：

$$station\ error = -(dx * cos\psi_{des} + dy * sin\psi_{des}) \tag{25}$$

速度误差为：

$$speed\ error = V_{des} - V * cos\Delta\psi/k \tag{26}$$

Apollo MPC 的控制变量为：

$$control\ state = \begin{bmatrix} \delta_f \\ \alpha \end{bmatrix} \tag{27}$$

其中 δ_f 为前轮转角，α 为加速度补偿。由于在实际场景中，车辆实际速度与期望速度可能存在偏差，于是有一个加速度补足。于是 Apollo MPC 的输入状态为：

$$state = \begin{bmatrix} lateral\ error \\ lateral\ error\ rate \\ heading\ error \\ heading\ error\ rate \\ station\ error \\ speed\ error \end{bmatrix} \tag{28}$$

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{2C_{af}+2C_{ar}}{mV_x} & \frac{2C_{af}+2C_{ar}}{m} & -\frac{2C_{af}l_f+2C_{ar}l_r}{mV_x} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{af}l_f-2C_{ar}l_r}{I_zV_x} & \frac{2C_{af}l_f-2C_{ar}l_r}{I_z} & -\frac{2C_{af}{l_f}^2+2C_{ar}{l_r}^2}{I_zV_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_c = \begin{bmatrix} 0 & 0 \\ \frac{2C_{af}}{m} & 0 \\ 0 & 0 \\ \frac{2C_{af}l_f}{I_z} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, C_c = \begin{bmatrix} 0 \\ -\frac{2C_{af}l_f-2C_{ar}l_r}{mV_x} - V_x \\ 0 \\ -\frac{2C_{af}{l_f}^2+2C_{ar}{l_r}^2}{I_zV_x} \\ 0 \\ 1 \end{bmatrix} \tag{29}$$

B. 预测状态模型分析

在百度的算法中，其采用了泰勒展开线性化的方式，在任意点，只保留一阶项，忽略高阶项。假设车辆参考系统在任意时刻的状态和控制量满足方程：

$$\dot{X}_r = f(X_r, U_r) \tag{30}$$

保留一阶项并忽略高阶项后可得：

$$\Delta\dot{X} = A(t)\Delta X + B(t)\Delta U \tag{31}$$

其中 Δ 代表着其现在状态与期望状态的误差。由于需要控制的是一个离散的系统但是所得的状态方程是一个连续的系统，于是需要对其进行离散化。在 Apollo 的算法中，其运用了双线性离散化的方法（bilinear discretize ）将其离散化后可得：

$$x(k+1) = A_d x(k) + B_d(k)u(k) + C_d \tag{32}$$

其中 $A_d = (I - TA/2)^{-1}(I + TA/2), B_d = TB, C_d = TC * \dot{\psi}$(heading error rate).

C. MPC 设计

为了设计 MPC 控制算法去跟踪轨迹，需要预测汽车未来每一步的状态。关于未来状态的预测决定了控制输入的大小和关于这些状态的矩阵的大小。在 baidu 的 Apollo 中，他的预测维度为 10（N_p=10），即预测未来十个阶段内的状态变量。假设现在时间为 k, $k > 0$; 未来状态方程可写为：

$$X_a(k+1), ..., X_a(k+m), ..., X_a(k+N_p) \tag{33}$$

其中 $X_a(k+m)$ 是预测的状态变量根据当前状态 $X_a(k)$。在 k+m 时所计算. 定义 ΔU 代表未来输入的控制变量递增于时间点 k, 相对于现在的状态表示为：

$$\Delta U_m = [\Delta u(k), ..., \Delta u(k+m), ..., \Delta u(k+N_p-1)]^T \tag{34}$$

根据 ΔU_m 状态方程 (A_a, B_a, C_a)[4] 所得在未来迭代的关系为:

$$\begin{aligned} &x_a(k+1) = A_a x_a(k) + B_a\Delta u(k) + C_a \\ &x_a(k+2) = A_a^2 x_a(k) + A_a B_a\Delta u(k) + B_a\Delta u(k+1) + C_a + A_a C_a \\ &\cdots \\ &x_a(k+N_p) = {A_a}^{N_p} x_a(k) + \sum_{i=k}^{N_p-1} {A_a}^i B_a\Delta u(N_p-1-i) + ... \\ &\quad + \sum_{i=k}^{N_p-1} {A_a}^i C_a \end{aligned} \tag{35}$$

之后就可以根据未来 10 个阶段的预测转换成矩阵形式:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{10} \end{bmatrix} = \tilde{A} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_9 \end{bmatrix} + \tilde{K} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_9 \end{bmatrix} + \tilde{C} \tag{36}$$

其中,

$$\tilde{A} = \begin{bmatrix} A & A^2 & \dots & A^{10} \end{bmatrix} \tag{37}$$

$$\tilde{K} = \begin{bmatrix} B & & & \\ AB & B & & \\ \vdots & & B & \\ A^9B & A^8B & \dots & B \end{bmatrix} \tag{38}$$

$$\tilde{C} = \begin{bmatrix} C \\ C + AC \\ \vdots \\ C + AC + \cdots + A^9C \end{bmatrix} \tag{39}$$

D. 滚动优化

根据给定的轨迹 $P_r(k)$, 可以找到车辆当前位姿相对于给定轨迹的误差并在线根据当前误差进行滚动优化，通过根据某种指标从而达到求出当前控制最优解。因此，滚动优化可能不会得到全局最优解，但是却能对每一时刻的状态进行最及时的响应，达到局部最优 [1]。

在百度的 MPC 算法中，把非线性的目标函数转换为二次规划问题，形式如下

$$\begin{cases} min\ q(x) = 0.5U^T HU + G^T U \\ s.t. \\ M_1 U = N_1 \\ M_2 U \leq N_2 \\ b_1 \leq U \leq b_2 \end{cases} \tag{40}$$

根据路径跟踪的目的，最后总结出目标函数为:

$$J(x) = (X - X_{ref})^T \tilde{Q}(X - X_{ref}) + U^T \tilde{R} U \tag{41}$$

其中，$\tilde{Q}$ 和 $\tilde{R}$ 为状态量和控制量的权重矩阵，X_{ref} 为状态变量的参考值，由于状态变量已经是实际值与预计值的偏差量，所以 X_{ref} 为零矩阵。代入之前的离散过后的模型:

$$\begin{aligned} &J(x) = (\tilde{A}X + \tilde{K}U + \tilde{C} - X_{ref})^T \tilde{Q}(\tilde{A}X + \tilde{K}U + \tilde{C} - X_{ref}) \\ &+ U^T \tilde{R} U \end{aligned} \tag{42}$$

令 $e = \tilde{A}X + \tilde{C} - X_{ref}$，则:

$$\begin{aligned} &J(x) = (e + \tilde{K})^T \tilde{Q}(e + \tilde{K}) + U^T \tilde{R} U \\ &= e^T \tilde{Q} e + e^T \tilde{Q} \tilde{K} U + (\tilde{K}U)^T \tilde{Q} e + (\tilde{K}U)^T \tilde{Q} \tilde{K} U + U^T \tilde{R} U \\ &\approx 2(\tilde{K}U)^T \tilde{Q} e + (\tilde{K}U)^T \tilde{Q} \tilde{K} U + U^T \tilde{R} U \\ &= U^T (\tilde{K}^T \tilde{Q} \tilde{K} + \tilde{R}) U + 2(U\tilde{K})^T \tilde{Q}(\tilde{A}X + \tilde{C} - X_{ref}) \end{aligned} \tag{43}$$

其中忽略 $e^T \tilde{Q} e$ 是因为其不包含控制变量。代入之前的二次规划的目标函数, $H = \tilde{K}^T \tilde{Q} \tilde{K} + \tilde{R}, G = (\tilde{K})^T \tilde{Q}(\tilde{A}X + \tilde{C} - X_{ref})$。

二次规划的约束条件是由最大前轮转角 (δ_f) 和最大加速度 (α) 组成。

$$\delta_f = \begin{cases} \delta_{f\,max}, & \delta_f(k) > \delta_{f\,max} \\ \delta_f, & \delta_f(k) \in [-\delta_{f\,max}, \delta_{f\,max}] \\ -\delta_{f\,max}, & \delta_f(k) < -\delta_{f\,max} \end{cases} \tag{44}$$

同理最大加速度限制为:

$$\alpha = \begin{cases} \alpha_{max}, & \alpha(k) > \alpha_{max} \\ \alpha, & \alpha(k) \in [-\alpha_{max}, \alpha_{max}] \\ -\alpha_{max}, & \alpha(k) < -\alpha_{max} \end{cases} \tag{45}$$

E. QP 求解器

Apollo 将 MPC 的优化求解转化为二次规划 (Quadratic Programming, QP) 问题 [2]，且 QP 问题的求解采用 QPOASES 的方法。通过 QP 求解器，就可以算出根据所给路径所算得的未来前轮转角和加速度补足。在实际模拟时，我采用了 MATLAB 中的 quadprog 来对 QP 问题进行求解。计算出的控制序列有 N_p 组解，然后将序列的第一组解作为后反馈角度输出给控制系统。

IV. 百度 LQR 路径跟踪

与 MPC 不同的是 LQR 是在一个固定的时域上求解, 并且在一个时域内只有一个最优解。在百度 Apollo 中 LQR 控制算法主要在横向控制上体现。根据之前的数学公式，LQR 的控制变量是 4 个横向控制状态变量，横向误差 (lateral error)，横向误差率 (lateral error rate)，航向误差 (heading error)，航向误差率 (heading error rate) 分别对应状态:

$$X_c = \begin{bmatrix} e_1 & \dot{e}_1 & e_2 & \dot{e}_2 \end{bmatrix}^T \tag{46}$$

在经过双线性离散后求得空间状态方程为:

$$\dot{X} = A_c X_c + B_c \delta + C_c \dot{\psi}_{des} \tag{47}$$

定义目标函数:

$$J(x) = \sum_{t=0}^{\infty} x^T Q x + U^T R U \tag{48}$$

其中 Q 为状态权重系数，R 为控制量权重系数。在百度 Apollo 中，在不同的的速度有着不同的 Q，R 的取值，于是采取查表法 (lookup Table) 根据目标速度跟新 Q，R 权重的大小。

之后对其进行滚动优化。LQR 的求解，假设输入为 $u = -Kx$，得出:

$$J(x) = \sum_{t=0}^{\infty} x^T (Q + K^T R K) x \tag{49}$$

为了找到 K, 假设存在一个常量矩阵 P，并且系统是稳定的 $(t- > \infty, x(t) = 0)$:

$$K = (R + B^T P B)^{-1} B^T P A \tag{50}$$

同时矩阵 P 满足黎卡提方程:

$$P = A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A + Q \tag{51}$$

在经过一系列的迭代后找到符合要求的最优解 K, 并得出反馈输出 u。

V. 前馈转角

在百度 Apollo 和 MPC 中在原有的反馈控制量的基础上都会加上一个前馈角度，使系统趋于稳定的同时让系统的横向误差为零。对其的计算为:

$$kv = \frac{lr * mass}{2 * cf * wheelbass} - \frac{lf * mass}{2 * cr * wheelbass} \tag{52}$$

$$\delta_{ff} = wheelbass * k + kv * v * v * k \tag{53}$$

其中 k 为道路曲率，δ_{ff} 为前馈转角。

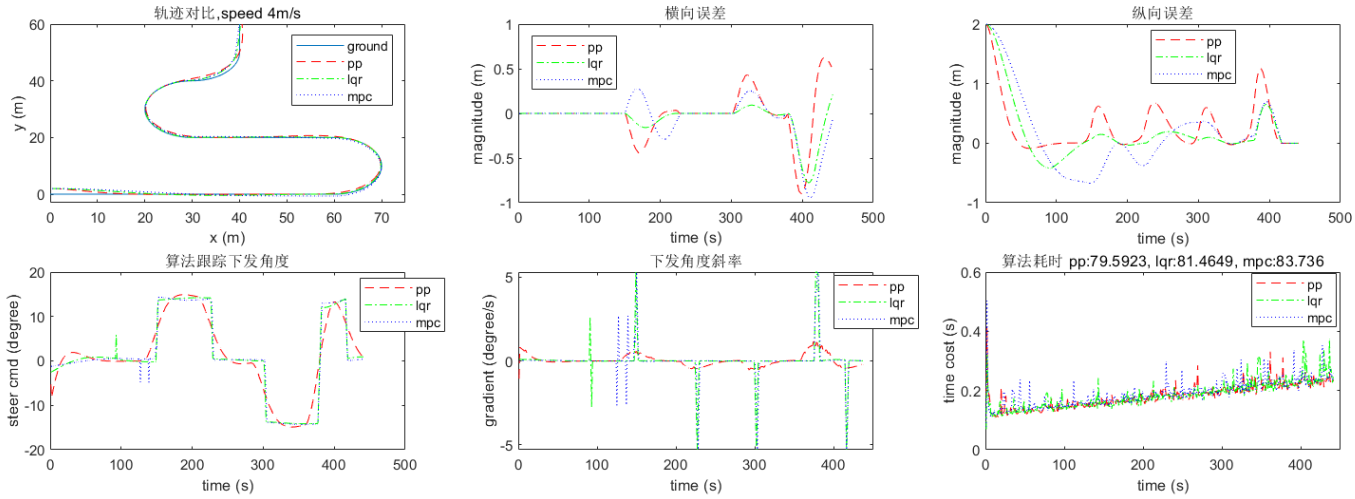


Fig. 3: 速度 4m/s 时 PP,LQR,MPC 三种算法对比

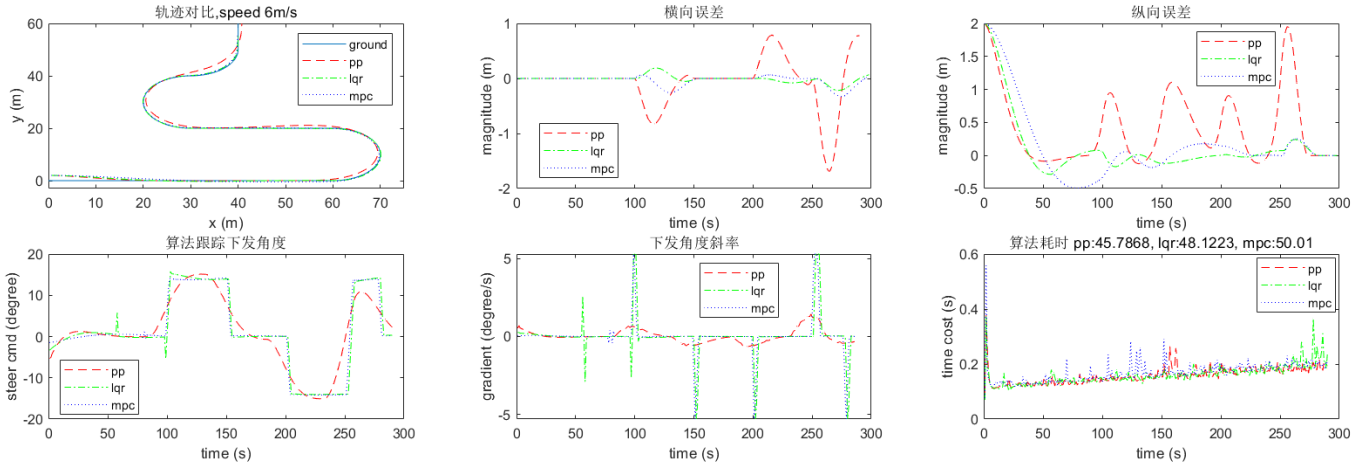


Fig. 4: 速度 6m/s 时 PP,LQR,MPC 三种算法对比

VI. 道路跟踪控制算法的模拟与分析

为了检测百度 Apollo 路径跟踪的算法，我在 MATLAB 环境下对这几种算法进行模拟。在复现时一共考虑了三种算法的对比：PP, Apollo LQR, Apollo MPC。模拟的环境是一个没有障碍物的赛道，赛道由直道和弯道组合而成。车辆会先从距离赛道原点横向两米的位置为起点，向预定道路靠拢。模拟的目标是为了让汽车成功尽可能的平滑和正确的跟随给定轨迹到达终点。在模拟中我使用汽车运动学模型对车辆的位姿进行更新，每次模拟的取样时间是 0.1 秒。

标志	描述	值 [单位]
m	总车质量	2016 [kg]
I_z	车辆转动惯量	3250 $[kg \cdot m^2]$
V	车速	4,6,10 [m/s]
l_f	车辆重心到前轮距离	1.25 [m]
l_r	车辆重心到后轮距离	1.25 [m]
C_f	前轮侧偏刚度	155494.663 [N/rad]
C_r	后轮侧偏刚度	155494.663 [N/rad]

TABLE I: 汽车模型数据

A. 算法效果分析

为了测试不同算法在直线和弯道上的效果，我在道路上测试了慢、中、快三种速度 (4m/s, 6m/s, 10m/s)，并对其效果进行分析。TABLE I [3] 定义了车辆模型的参数

1) 跟踪精度对比：从图片.3, 图片.4 和图片.5 中可以看到横向和纵向三种算法在跑赛道时的对比。在低速时 (4m/s) 从横向误差来看，Apollo LQR 的横向误差表现最好，其次是 MPC，PP 的效果略差。从纵向误差来看，MPC 在并向道路时反应较慢，LQR 在纵向误差表现最好，PP 算法纵向表现不佳。随着车辆速度的提升，MPC 的横向误差表现会超过 LQR，纵向误差会和 LQR 表现持平。在高速条件下，可以看出 PP 算法的跟踪精度会随着速度的增加而变差。

2) 跟踪平顺度对比：从对比图的算法下发角度和其斜率对比图可以看出，在低速时，LQR 和 MPC 算法的下发角

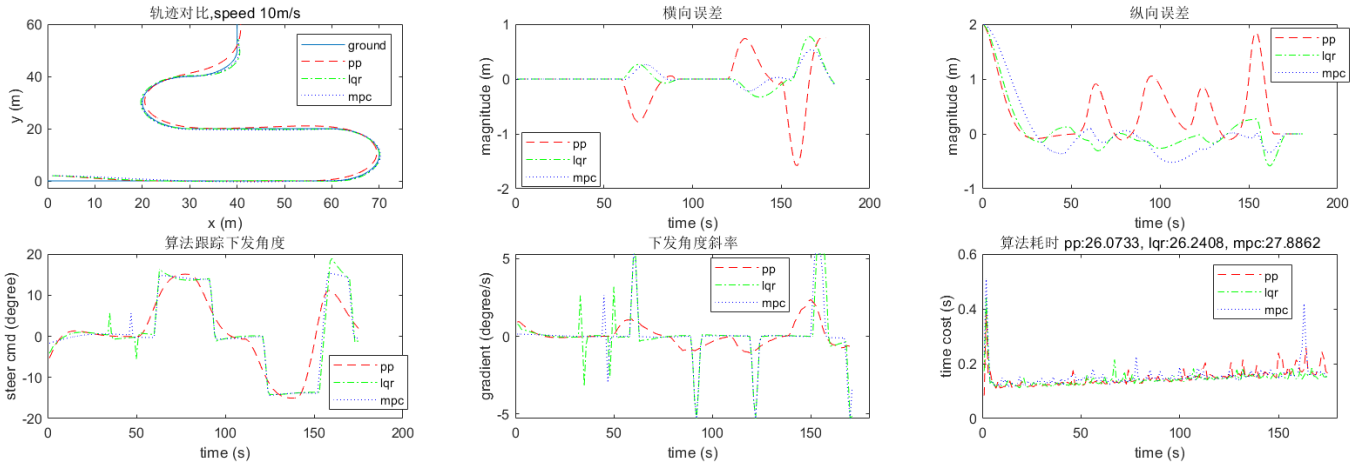
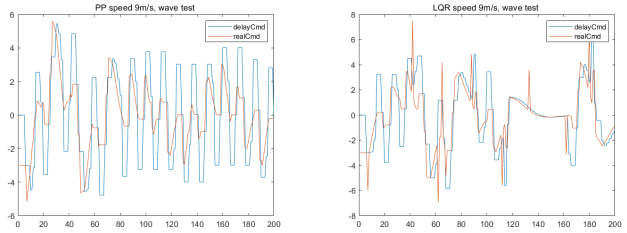


Fig. 5: 速度 10m/s 时 PP,LQR,MPC 三种算法对比

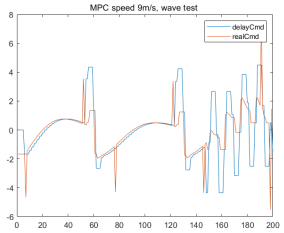
度的斜率在大部分情况为零，但在转弯时这两种算法的下发角度比 PP 算法更加激进。而 PP 算法的整体下发角度是递进的，并没有极大的下发角度梯度。随着速度的增加，MPC 和 LQR 算法的转弯次数有所增加，PP 的转角整体有所增加，但是整体的趋势不变。因此，LQR 和 MPC 在大部分情况展现了很好的跟踪平顺性，但在转弯时会比较激进。

3) *算法耗时对比*: 通过各个算法耗时的对比，可以看出，MPC 和 LQR 的计算时间明显高于 PP 算法，并且在特定情况下（如急转弯）MPC 会比 LQR 消耗更多的计算时间，但在快速连续转弯的情况下，耗时好于 LQR。



(a) PP 算法

(b) LQR 算法



(c) MPC 算法

Fig. 6: 不同算法下对车辆转角超调时的效果对比

4) *超调情况下算法对比*: 为了模拟汽车实际转向角在每次梯度方向变化时的延迟与超调。我通过手动编程设定延迟步数并加上了转向的超调和整体反应延迟，对于算法在对这种情况的稳定性进行模拟。从图6中，可以看出车辆转向角的延迟与超调这三种算法都会有一定影响。PP 算法在当轮胎转向时发生延迟和超调时，会产生长时间的“画龙”和延迟现象。LQR 算法在前期有震荡的情况，但在一定时间之后也会有所缓解。MPC 算法在起步阶段影响较小，但在一定时间之后会有震荡加剧的情况。

VII. 总结

通过这次模拟，百度 Apollo 的 MPC 和 LQR 算法展现出了很好的道路跟随能力。尤其在高速情况下，百度 MPC 算法由于考虑了横向和纵向，两个方向的误差，所以在横纵效果上有着更好的效果。MPC 和 LQR 在算法的平顺性上也能够在大部分时刻平滑的进行驾驶。车辆转角命令下达的延时和超调对三种算法都有一定的影响。另外在这次模拟中，LQR 相对于 MPC 算法有着更好的表现。因为 LQR 对整个时域进行优化求解，且求解过程中假设控制量不受约束，但是实际情况下，控制量是有约束的。而 MPC 通常在比整个时域更小的时间窗口中解决优化问题，因此可能获得次优解，并且其对线性没有任何的假设，能够处理硬约束以及非线性系统偏离其线性化工作点的迁移。

在测试时发现，通过调整 MPC 参数，可以减小仿真结果的跟踪误差。不同的权重矩阵的参数会对各个状态指标产生直接的影响。这意味着 QP 问题的解决方案是在跟踪误差较少和更平滑、更高水平的转向角度输入之间进行一些权衡。

在模拟中，依然有一些实际特征需要进一步研究和改进。首先由于车辆位姿的更新是通过汽车运动学模型，但 MPC 和 LQR 的模型是汽车运动学模型，因此在更新汽车位姿时可能会有一定的影响，尤其在高速阶段。其次，在实际的情况下，期望速度和实际速度可能因为外部问题而产生偏差，因此 Apollo MPC 算法可能在速度保持上，会有着更好的效果。

References

[1] Joao Manoel Gomes da Silva Jr. Felipe K¨uhne, Walter Fetter Lages. Model predictive control of a mobile robot using linearization. Federal University of Rio Grande do Sul.

[2] Andreas Potschka Hans Georg Bock Moritz Diehl Hans Joachim Ferreau, Christian Kirches. qpoases:

a parametric active-set algorithm for quadratic programming. Springer-Verlag Berlin Heidelberg and Mathematical Optimization Society, 2014.

[3] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Transactions on Vehicular Technology, 66(2):952–964, Feb 2017.

[4] 徐威龚建伟, 姜岩. 无人驾驶车辆模型预测控制 [m]. 2014.