Apollo车辆控制仿真程序--matlab实现

本文档很多部分参考了以下博客系列文章：

https://blog.csdn.net/u013914471/article/details/83824490

1 期望轨迹生成

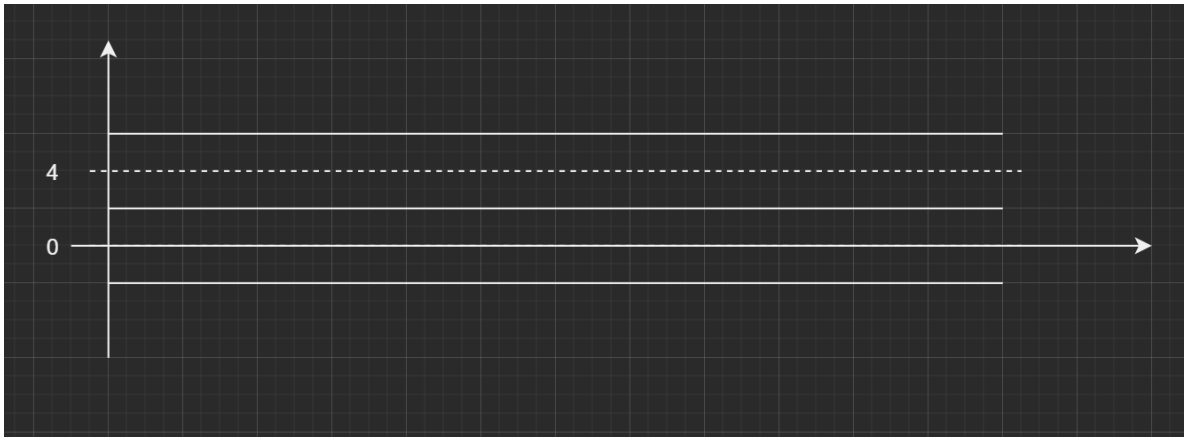
1.1 道路

仿真环境设置的道路由2车道组成。

道路方向和x轴方向平行。

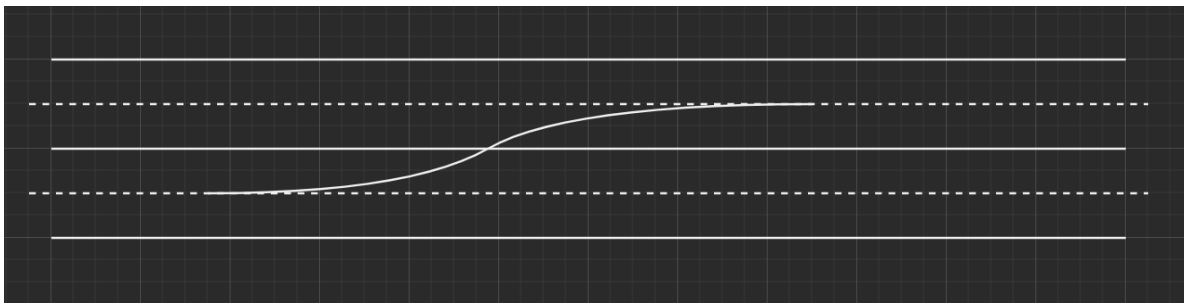
车道1中心线的y坐标是4

车道2中心线的y坐标是0



1.2 变道轨迹的生成

变道轨迹采用5次多项式拟合



横向运动用5次多项式来表示

$$l(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

根据初始状态和末状态的已知条件：

$$
\begin{gathered}
l(t_0) \\
\dot{l}(t_0) \\
\ddot{l}(t_0) \\
l(t_1) \\
\dot{l}(t_1) \\
\ddot{l}(t_1)
\end{gathered}
$$

根据6个已知条件可以得到6个方程，解出来6个参数。

纵向运动用4次多项式来表示

$$
s(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4
$$

根据初始状态和末状态的已知条件：

$$
\begin{gathered}
s(t_0) \\
\dot{s}(t_0) \\
\ddot{s}(t_0) \\
\dot{s}(t_1) \\
\ddot{s}(t_1)
\end{gathered}
$$

根据5个已知条件可以得到5个方程，解出来5个参数。

之后得到每个t时刻的s和l，即得到了所有轨迹点的frenet坐标，之后再根据地图信息将frenet坐标转换成xy坐标

由于这里道路是直线，很容易进行frenet坐标和xy坐标之间的转换。

如果道路是曲线，那么还需要显式地进行frenet坐标和xy坐标之间的转换。

在程序中，通过生成多段轨迹之后再组合成一段轨迹作为期望轨迹来进行控制仿真。

2 车辆动力学模型

车辆动力学仿真模型采用侧向二自由度模型

$$
\frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \varphi \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{af}+2C_{ar}}{mV_x} & 0 & -V_x - \frac{2C_{of}\ell_f - 2C_{ar}\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_{af}\ell_f - 2C_{ar}\ell_r}{I_I V_x} & 0 & -\frac{2C_{af}\ell_f^2 + 2C_{ar}\ell_r^2}{I_z V_x} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \varphi \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{af}}{m} \\ 0 \\ \frac{2\ell_f C_{af}}{I_z} \end{bmatrix} \delta
$$

假设车身朝向记为速度方向

仿真步骤：

1 根据当前速度更新下一步位置

注意：之前的模型忽略了Vy，由于侧向速度不大，一般对仿真影响不大。

$$
\begin{aligned}
x_{k+1} &= x_k + V_x cos\phi T + Vycos(\phi + \frac{\pi}{2})T \\
y_{k+1} &= y_k + V_x sin\phi T + Vysin(\phi + \frac{\pi}{2})T
\end{aligned}
$$

2 根据当前速度和加速度更新下一步的速度

$$
v_{k+1} = v_k + aT
$$

3 根据侧向状态空间模型更新状态

$$X_{k+1} = A_c X_k + B_c u$$

Ac和B_c是离散之后的状态空间矩阵

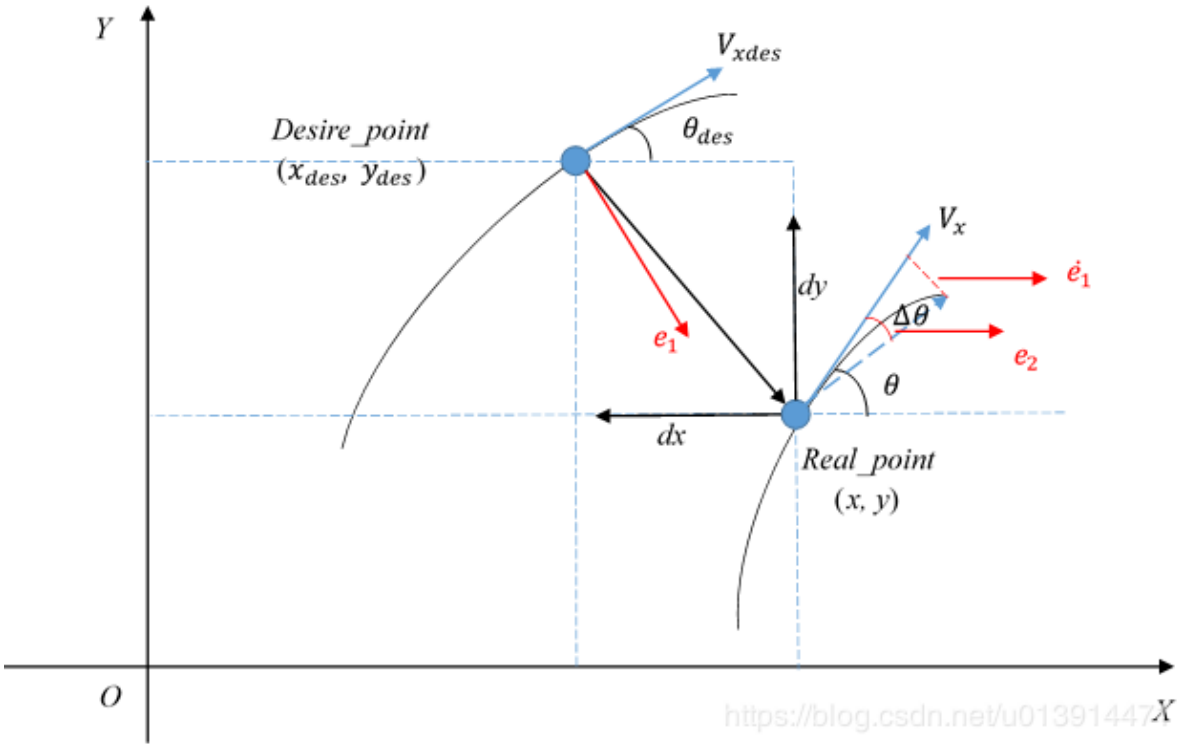
$$\frac{X_{k+1} - X_k}{T} = AX_k + Bu$$
$$X_{k+1} = X_k + AX_kT + BuT$$
$$A_c = AT + I$$
$$B_c = BT$$

根据这个仿真程序可以得到：车辆在固定转向输入时是做圆周运动的，并且半径与理论计算的半径一致。

车辆在固定转向输入时转弯半径计算公式

$$\rho = \left(1 - \frac{m}{2\ell^2}\frac{\ell_f C_{\alpha f} - \ell_r C_{\alpha r}}{C_{\alpha f} C_{\alpha r}} V^2\right)\frac{\ell}{\delta_0}$$

3 横向PID预瞄控制



车辆跟踪控制中的误差计算。

$$\begin{cases} e_1 = dy * \cos\theta_{\text{des}} - dx * \sin\theta_{\text{des}} \\ \dot{e_1} = V_x * \sin\Delta\theta = V_x * \sin e_2 \\ e_2 = \theta - \theta_{\text{des}} \\ \dot{e_2} = \dot{\theta} - \dot{\theta}_{\text{des}} \end{cases}$$

$e_1, \dot{e_1}, e_2, \dot{e_2}$分别对应

- 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

在横向PID预瞄控制中，这个期望的点和当前实际的点有一个时间或者位置上的偏移量，因此叫做预瞄。

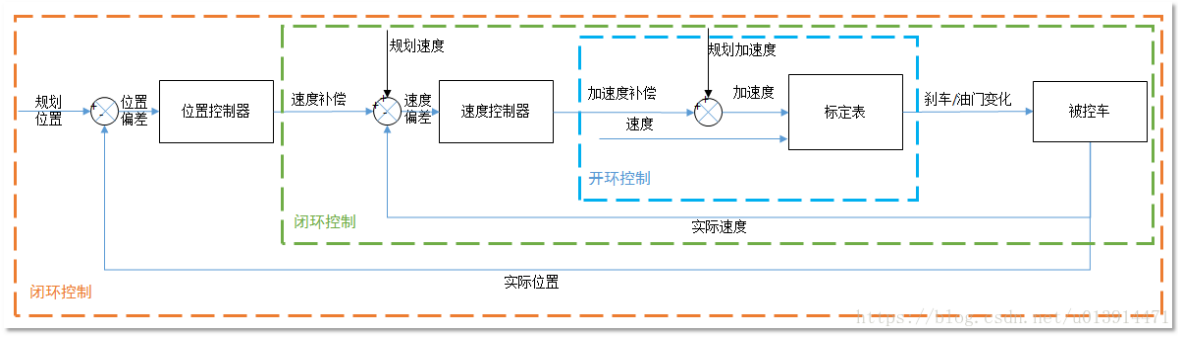
横向控制仅用到横向误差 lateral_error

用这个误差值求出P控制量和I控制量，作为方向盘转角输入即可

4 横纵向PID预瞄控制

仅有横向控制时，总速度是保持恒定的，但是由于在规划变道路线时，如果保持x方向速度恒定，那么在变道过程中，由于y方向速度的存在，和速度是大于原来直行时的速度的。这就会导致横向控制看起来能够跟踪期望轨迹，但是在时间上会落后期望轨迹的。因此引入纵向速度控制

apollo中纵向控制的示意图



我这里做的稍有区别，规划器没有输出加速度，速度控制器输出的控制量，直接作为加速度输入即可。

station_error=$-(dx * cos\theta_{des} + dy * sin\theta_{des})$

speed_error=$V_{des} - V * cos\Delta\theta/k$

5 横向LQR控制

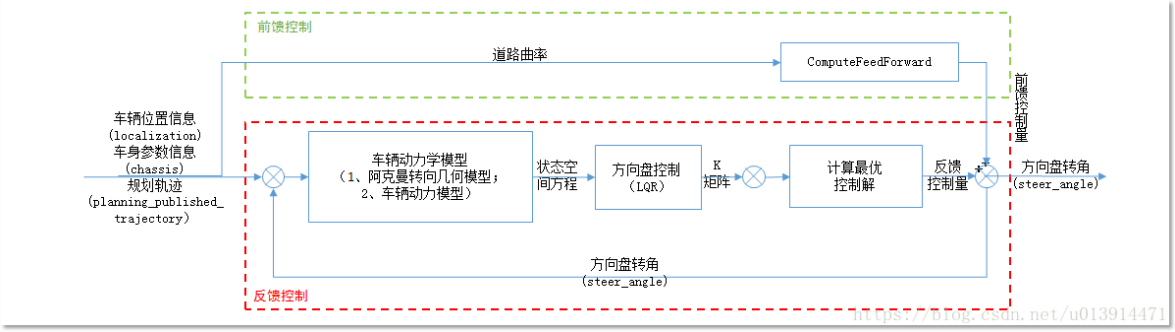
误差状态空间模型

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{af}+2C_{ar}}{mV_x} & \frac{2C_{af}+2C_{ar}}{m} & \frac{-2C_{af}\ell_f+2C_{ar}\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_{af}\ell_f-2C_{ar}\ell_r}{I_xV_x} & \frac{2C_{af}\ell_f-2C_{ar}\ell_r}{I_x} & -\frac{2C_{af}\ell_f^2+2C_{ar}\ell_r^2}{I_xV_x} \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2C_{\alpha f}\ell_f}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ -\frac{2C_{af}\ell_f-2C_{ar}\ell_r}{mV_x} - V_x \\ 0 \\ -\frac{2C_{af}\ell_f^2+2C_{ar}\ell_r^2}{I_zV_x} \end{bmatrix} \dot{\varphi}_{des}$$

误差状态变量分别对应

- 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

横向LQR控制基本按照apollo的思路来：



6 横纵向MPC控制

横纵向MPC控制，将纵向控制误差模型和横向控制误差模型集成到一起。

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \\ e_3 \\ e_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{2C_{af}+2C_{ar}}{mV_x} & \frac{2C_{af}+2C_{ar}}{m} & \frac{-2C_{af}\ell_f+2C_{ar}\ell_r}{mV_x} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{af}\ell_f-2C_{ar}\ell_r}{I_zV_x} & \frac{2C_{af}\ell_f-2C_{ar}\ell_r}{I_z} & -\frac{2C_{af}\ell_f^2+2C_{ar}\ell_r^2}{I_zV_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \\ e_3 \\ e_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{2C_{af}}{m} & 0 \\ 0 & 0 \\ \frac{2C_{af}\ell_f}{I_z} & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \delta \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{2C_{af}\ell_f-2C_{ar}\ell_r}{mV_x} - V_x \\ 0 \\ -\frac{2C_{af}\ell_f^2+2C_{ar}\ell_r^2}{I_zV_x} \\ 0 \\ 1 \end{bmatrix} \dot{\varphi}_{des}$$

误差状态变量分别对应

- 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
- 速度误差speed_error
- 位置误差station_error

有了这个线性误差模型就能够进行MPC控制

设误差模型为：

$e_{k+1} = Ae_k + Bu$

设某一时刻误差状态为E_k

那么假设未来一段时间内的控制输入为$u_{k+1}, u_{k+2} \ldots u_{k+n}$，合并记成$U$向量

那么未来这段时间内的误差状态为$e_{k+1}, e_{k+2} \ldots e_{k+n}$，合并记成$E$向量

那么可以设计一个目标函数

$J = \sum_{k=0}^{n} e_k^T Q e_k + U^T R U$

J最终可以表达成U的二次型

$J = \frac{1}{2} U^T H U + F U + Y$

那么就可以通过二次规划算法求解出未来一段时间内最佳控制变量U

根据控制窗口（一般为1）来施加控制。到下一个时间步再重复这个过程。

关于MPC控制详细过程可以参考mpc文档1和mpc文档2