The Model

Student describes their model in detail. This includes the state, actuators and update equations.

Answer:

Based on the previous lessons, my model contains these states: $[x, y, \varphi, \mathrm{v}, \mathrm{cte}, \mathrm{e}\varphi]$, the are used to model position of vehicle, orientation of vehicle, how fast of vehicle and the error between the real state and predicted state.

To control a car, we just need to control forward and backward as well as orientation. in other words, for decision making and control algorithm, we just need to control the acceleration of our car as well as the steering angle, such as $[\delta, a]$.

Finally, my update model are shown below.

$$\begin{cases} x_{t+1} = x_t + v_t * \cos(\varphi_t) * dt \\ y_{t+1} = y_t + v_t * \sin(\varphi_t) * dt \\ \varphi_{t+1} = \varphi_t + \dfrac{v_t}{L_f}\delta_t * dt \\ v_{t+1} = v_t + a_t * dt \\ cte_{t+1} = f(xt) - y_t + (vt \sin(e\varphi_t) * dt) \\ e\varphi_{t+1} = \varphi_t - \varphi_{des_t} + \left(\dfrac{v_t}{L_f} * \delta_t * dt \right) \end{cases}$$

Timestep Length and Elapsed Duration (N & dt)

Student discusses the reasoning behind the chosen *N* (timestep length) and *dt* (elapsed duration between timesteps) values. Additionally the student details the previous values tried.

Based on the theory of MPC, bigger T will lead to better performance. In fact, the environment around car always changes, so for autonomous driving, predicting the state in further future is meaningless. So I set T to 1s. You know, turn down dt will make the car move smoother, but consider the time consumption of Ipopt, I set dt equal to 0.1. By the way, I have try to other combine of N and dt, In fact, when dt was decreased to 0.1. Keep on to decrease it will not increase the performance of algorithm besides make the solver slower.

Polynomial Fitting and MPC Preprocessing

A polynomial is fitted to waypoints.

If the student preprocesses waypoints, the vehicle state, and/or actuators prior to the MPC procedure it is described.

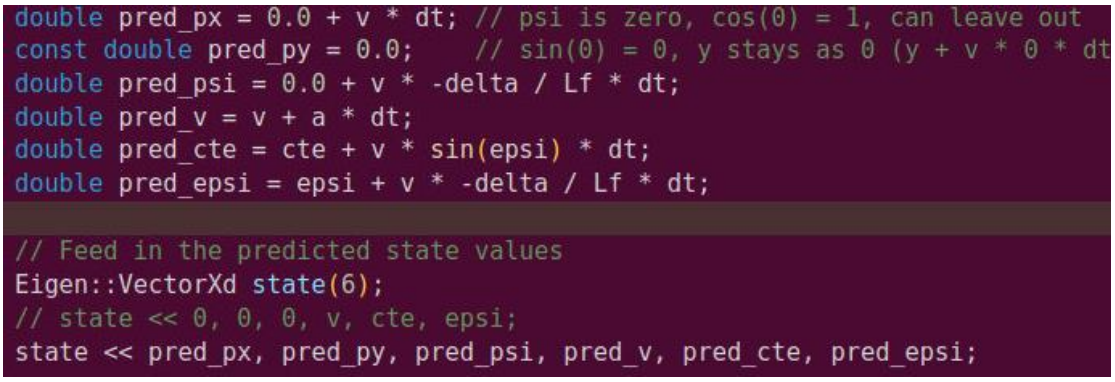
In the waypoints pre-processing module, I use the formula to transform data from vehicle coordinate system to ground coordinate system.

$$\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_p \\ \sin\theta & \cos\theta & -y_p \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix}$$

Model Predictive Control with Latency

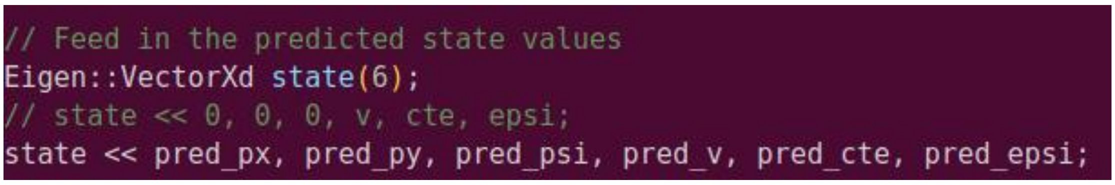
The student implements Model Predictive Control that handles a 100 millisecond latency. Student provides details on how they deal with latency.

In the state update steps, before I input the state to the solve, I add the latency to predicted states as shown below.



```
double pred_px = 0.0 + v * dt; // psi is zero, cos(0) = 1, can leave out
const double pred_py = 0.0;    // sin(0) = 0, y stays as 0 (y + v * 0 * dt
double pred_psi = 0.0 + v * -delta / Lf * dt;
double pred_v = v + a * dt;
double pred_cte = cte + v * sin(epsi) * dt;
double pred_epsi = epsi + v * -delta / Lf * dt;

// Feed in the predicted state values
Eigen::VectorXd state(6);
// state << 0, 0, 0, v, cte, epsi;
state << pred_px, pred_py, pred_psi, pred_v, pred_cte, pred_epsi;
```



```
// Feed in the predicted state values
Eigen::VectorXd state(6);
// state << 0, 0, 0, v, cte, epsi;
state << pred_px, pred_py, pred_psi, pred_v, pred_cte, pred_epsi;
```