**CarND-Controls-MPC**

Vehicle Model

The model I used in this MPC project is a kinematic bicycle model which neglects the complex interactions between the tires and road.   
The equations of the model are shown below:

x[t] = x[t-1] + v[t-1] \* cos(psi[t-1]) \* dt

y[t] = y[t-1] + v[t-1] \* sin(psi[t-1]) \* dt

psi[t] = psi[t-1] + v[t-1] / Lf \* delta[t-1] \* dt

v[t] = v[t-1] + a[t-1] \* dt

cte[t] = f(x[t-1]) - y[t-1] + v[t-1] \* sin(epsi[t-1]) \* dt

epsi[t] = psi[t] - psides[t-1] + v[t-1] \* delta[t-1] / Lf \* dt

Here x and y are the position of the car on the track, psi is the direction in which the car is headed, v is velocity of the car, cte is the cross-track error & epsi is the orientation error. These values are considered the state of the model. In addition to that, Lf is the distance between the center of mass of the car and the front wheels.

The model input (actuators) are the car's acceleration(throttle) a, and the steering angle delta.

The goal of MPC method is to find throttle acceleration and steering angle which minimizes the cost function while minimizing cross track, heading, and velocity errors. A further improvement is to make the turns as smooth as possible. Additionally, the vehicle velocity should not change too radically. The goal of this final loop is to make control decisions smoother. The next control input should be like the current one

Timestep Length and Elapsed Duration

The number of points N and the time interval dt define the prediction horizon T = N \* dt. The number of points impacts the performance of controller, short prediction horizon lead to more responsive controllers, but are less accurate and lead to more instable when chosen too short. Long prediction horizon generally leads to smoother controls. For a given prediction horizon shorter time step dt although lead to more accurate controls, it will slow down the computation latency.

In this project, I start to try N =25 and dt =0.05, but it became erratic. After trying many times, finally I chose N =10 and dt = 0.1 as it displayed much better results over other values.

Polynomial Fitting and MPC Preprocessing

The waypoints provided by the simulator are transformed to the car’s coordinate system. First shifting the origin to the current position of the car, then a 2D rotation is applied on the relative waypoints, finally transfer into car coordinate system.

X' = cos(psi) \* (ptsx[i] - x) + sin(psi) \* (ptsy[i] - y);

Y' = -sin(psi) \* (ptsx[i] - x) + cos(psi) \* (ptsy[i] - y);

Here X' and Y' are the waypoints in the vehicle coordinate system.

A 3rd polynomial is fitted to the transformed waypoints. The coefficients are used to calculate the cte and epsi.

Model Predictive Control with Latency

There are two common approaches to handle the delays in control problem.

1. Latency is considered by constraining the controls to the values of the previous iteration for the duration of the latency. Thus, the optimal trajectory is computed starting from the time after the latency period. So, the dynamics during the latency period is still calculated according to the vehicle model.

2. The position of the car is estimated based on its current speed and heading direction by propagating the position of the car forward until the expected time when actuations are expected to have an effect. This approach is intuitionistic, and easy to handle.

So to handle the actuator latency, I used the state values which are calculated by the model and delay interval replace the initial one.

x\_delay = x0 + (v\* cos(psi0) \* delay);

y\_delay = y0 + (v\* sin(psi0) \* delay);

psi\_delay = psi0 + (v\*delta/Lf \*delay);

v\_delay = v + a\* delay;

cte\_delay = cte0 + (v\*sin(epsi0)\*delay);

epsi\_delay = epsi0 + (v \* delta\* delay/Lf);