Term 2 – Project 5 - Model Predictive Control (MPC)

The model

I used the Kinematic model for the Model Predictive Control (MPC) project. Kinematic models are simplifications of dynamic models that ignore tire forces, gravity, and mass. This simplification reduces the accuracy of the models, but it also makes them more tractable.

The state of the vehicle is defined by x, y coordinates, orientation ψ and velocity v. Car also have 2 actuators (Steering wheel, Throttle pedal). So there are 2 Control inputs, 'delta' for steering angle and 'a' for acceleration. Actuators change the state of vehicle over time.

The model uses previous timestamp state and actuation to calculate state of current timestamp using equations below:

$$
\begin{aligned}
x_{t+1} &= x_t + v_t * cos(\psi_t) * dt \\
y_{t+1} &= y_t + v_t * sin(\psi_t) * dt \\
\psi_{t+1} &= \psi_t + \frac{v_t}{L_f} * \delta_t * dt \\
v_{t+1} &= v_t + a_t * dt \\
cte_{t+1} &= f(x_t) - y_t + v_t * sin(e\psi_t) * dt \\
e\psi_{t+1} &= \psi_t - \psi des_t + \frac{v_t}{L_f} * \delta_t * dt
\end{aligned}
$$

L_f measures the distance between the front of the vehicle and its center of gravity. I assumed L_f value of 2.67 for all the calculations.

Timestep Length and Elapsed Duration (N & dt)

The prediction horizon (T) is the duration over which future predictions are made. T is the product of two variables, N and dt. N is the number of time steps in the horizon. dt is how much time elapses between actuations. N also determines the number of variables optimized by the MPC. This is the major driver of computational cost.

N, dt, and T are hyper parameters which I tuned for the model predictive controller. I followed the guideline that T should be as large as possible, while dt should be as small as possible. MPC attempts to approximate reference trajectory by means of discrete paths between actuations. Larger values of dt result in less frequent actuations, which makes it harder to accurately approximate a continuous reference trajectory.

I could determine the reasonable range for T as 1 second by trial and error. When I started tuning dt and N for various values, the car exhibited different behavior. For example when I selected N =20 and dt =0.05, the car was going off the road. When I started reducing the N value and increasing dt, it stabilized when N=10 and dt=0.1.

Polynomial Fitting and MPC Preprocessing

The coordinates are converted to vehicle's coordinate system for calculations. This resulted in having vehicle at origin (0,0) and also orientation angle becomes zero.

3rd degree polynomials are common which fits most roads. I used the already defined function polyfit() in main.cpp to fit transformed waypoints to a 3^{rd} degree polynomial function.

Model Predictive Control with Latency

In a real car, an actuation command won't execute instantly; there will be a delay as the command propagates through the system. This is a problem called "latency". Model Predictive Controller can adapt quite well to this problem because we can model this latency in the system.

To account for the latency, extra step is added to predict the vehicles position after 100 milliseconds. The function defined in main.cpp, predicted_state() considers the latency of 0.1 second. The predicted state is computed using the same model equations and then the predicted state is passed to MPC.Solve() function for computations.