Term2: Project5 - Model Predictive Control

In this project, a Model Predictive control algorithm is used to drive a car around a simulated track such that the Cross-track error(CTE) and error in orientation(EPSI) are minimized.

The MPC algorithm is run on a simulated track provided by Udacity from which calculate CTE and error in orientation (EPSI).

1. **Attachments**: Attached updated MPC.cpp and main.cpp files along with a small recorded video clip. Note that there was some extra latency induced on the processor when the simulator was run simultaeously with video recording. It looks better when run alone.

2. **Model**:

a) The state vector includes the following six components:

x - x co-ordinate, **y**- y co-ordinate, **psi** - angle of orientation wrt origin, **v** - speed of the vehicle, **cte**- cross track error, **epsi** - error in psi

b) The actuators/control parameters are steering angle(delta) between -25 and 25 degrees and throttle (acceleration/decceleration) between -1 and 1.

c) A Kinematic motion model is used as the motion model with the following state update equations. They predict the state t based on the state at t-1.

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

d) At each time step, the control variables for the next N time steps are optimized to minimize cost(objective) function. Cost function is the sum of deltas of CTE , EPSI wrt predicted curve(polynomial) and velocity(compared to reference velocity). Also other values like difference of consecutive steer angles are added to promote smooth driving. All this is fed to an optimizer called Ipopt which returns the optimum set of steer angles and throttle values for the next N time steps.

3. **Timestep Length and Elapsed Duration (N & dt)**:

As the number of variables and constraints for the optimizer increase with N, it drives up the computational cost. So, it's better to choose an optimum value for N such that it estimates the curvature(polynomial) upto a good enough distance. I started with 25 given in the lesson, then 20, 15 and finally reduced it to 10. Along with the obvious reduction in compiling time during 'make' step, 10 also provided better results.

dt is the elapsed time duration between each timestep. In this project, we finetune the MPC algorithm for a latency of 100ms (0.1 second) to simulate real life delay in actuator commands and vehicle response. To take care of this latency, the algorithm is tweaked to get the actuator values 0.1 seconds ahead. So, it makes sense to set dt to 0.1 second for easy implementation of the logic in the code.

4. **Polynomial Fitting**:

Polyfit and Polyeval functions provided in the main.cpp file are used to fit to track waypoints and calculate cte and epsi.

5. **MPC with latency**:

As described above, the latency of 100ms is taken care of by getting thetake the actuator values 0.1 seconds ahead. Lines 111-114,MPC.cpp

6. **Finetuning**:

The steering delta portion of the cost function is increased 2000 times to provide smooth drving. After some trial and error, noticed the cost function values that are related to steering are more important than CTE for a smooth ride. So, multiplied the CTE cost by 0.15. Lines 58-75,MPC.cpp