Project 6: CarND-MPC-Project

Model Predictive Control (MPC) Project

The goals / steps of this project are the following:

- Implement Model Predictive Control (MPC) model to drive the car around the track.
- Cross Track Error(CTE) is not given in this project, it has to be calculated in the project implementation
- Also take of the latency calculations between actuations

Model (state, actuators and update equations)

Used the kinematic bicycle mode in this MPC project, this model ignores the gravity, tire forces and mass. At lower speed, Kinematic model lowers the accuracy of the actual values, this implementation uses the following equations for the model provided in the course:

```
// Recall the equations for the model:
// 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
```

x, y	: Position of the vehicle
v	: Velocity which vehicle moves
psi	: Orientation of the vehicle
cte	: Cross track error
epsi	: Orientation error of psi
Lf	: distance from the front of the vehicle and center of gravity of vehicle
a	: Acutators like steering wheel

State of the vehicle is defined as:

```
state << 0.0, 0.0, 0.0, v, cte, epsi;
{ px, py, psi, v, cte, epsi }
```

Time-step Length and Elapsed Duration (N & dt) & Polynomial Fitting and MPC preprocessing

MPC approximates the trajectory with the third order polynomial and with Nstates with N-1 changes of actutator using prediction horizon T.

T is the product of the N and dt,

N: Number of time-steps in the horizon

dt: time lapses between 2 actuations

The CTE is calculated using polynomial function and epsi is –ve arctan for the first derivative of point x = 0/ MPC control predicts from vector state N to N01 actutator vectors for predicion of horizon T.

FG_eval class provides the polynomial cost function:

w0*cte^2 + w1*epsi^2 + w2*(v - ref_v) + w3*(delta(t+1)-delta(t))^2 + w4*(a(t+1)-a(t))^2 + w5*a^2 + w6*delta^2

and the hyper parameters are as follows:

w0, w1, w2, w3, w4, w5, w6 are the weights

w0, w1, w2 are used for cte, epsi and distance to desired target speed

w3 and w4 are for control smoothness of steering and acceleration

w5 and w6 are used for minimizing the use of steering and accelartion

Model Predictive Control with Latency

Model predictive control handles 100ms latency provided by the simulator latency between the sensors and processing. The steering angle and throttle are not changed during latency with 100 ms.

```
// Steering angle is not changed during latency
    for (int i = delta_start; i < delta_start + latency; i++) {
        vars_lowerbound[i] = steering_offset;
        vars_upperbound[i] = steering_offset;
    }

  // Acceleration/deceleration is not changed during latency
    for (int i = a_start; i < a_start + latency; i++) {
        vars_lowerbound[i] = throttle;
        vars_upperbound[i] = throttle;
    }
```

Steering offset and the actutator/throttle used during the different transtions of the states from t-1 to t states. Handling the actuators after latency, as the transtions between t+2 and t+3 states are used. This optimization solves the problem faster than 50ms, during the normal conditions of driving.

Files Submitted & Code Quality

Required Files:

Submission includes all required files
My project includes the following files:

- MPC.cpp
- MPC.h
- main.cpp
- CMakeLists.txt
- Writeup_report.pdf
- Removed the video , as it took larger space. And it creates issue in submitting to udacity

Conclusion, Future Work

- Will try different from Kinematic model in doing this MPC project
- Increase the speed of vehicle to higher values and test the MPC model and fine tune in further