**CarND-Controls-MPC**

In this project, the implementation of MPC will enable the car to drive around a track.

The algorithm is able to estimate when the car should accelerate or slow down based on the complexity of the turn.

**Model:**

***State*:** x, y are positions, psi and v are orientation and velocity of the car respectively. All these values are received from the simulator.

***Actuators***: “delta” is steering angle and “a” is the throttle. Solver computes these variables which are passed to the simulator.

***New State***: This is obtained from the previous state by using the following equations:

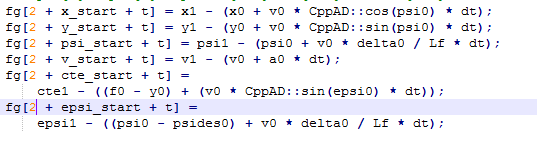
x=x+v\*cos(psi)\*dt

y=y+v\*sin(psi)\*dt

v=v+a\*dt

psi=(v \* tan(-delta) / Lf) \* dt + ((a \* tan(-delta) / (2 \* Lf)) \* pow(dt, 2));

**Note**: The problem is based on a bicycle vehicle model. Effects such as inertia, torque etc are not taken into account. Equations are as shown below:



**Where:**

v- Velocity,

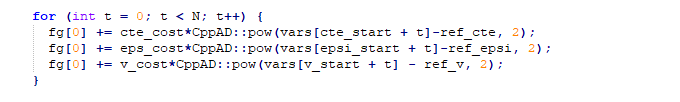
psi- heading direction

cte-cross track error

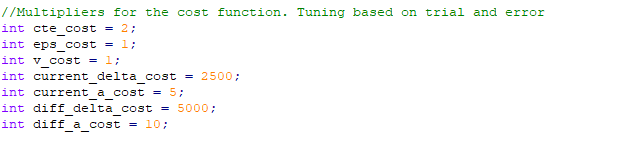
epsi- orientation error

Lf- distance between c.g and front wheels.

**Errors** have been calculated using cost factors and a bit of tuning.



These cost parameters have been manually tuned. More emphasis has been placed on steering angle and steering angle change.

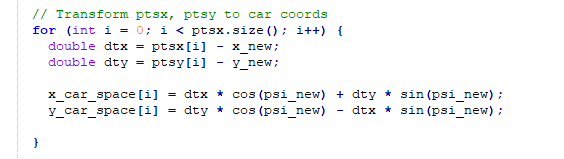


**Choosing Dt and N:**

These were chosen manually by observing how the car drives. Smaller values of N(10-20) showed promising results. Higher values of N (40) tried previously did not have a smooth motion of the car. If dt is too large, then the car could cover a lot of distance before receiving input from the actuator. This could lead to the car travelling smoothly along straight lanes and off road along the curves. dt=0.1 showed good results.

**Polynomial Fitting and MPC Preprocessing**:

The values read from the simulator have been converted to the vehicle coordinates by using the transformation formula and CTE was calculated.



**Latency:**

The car is assumed to travel at the current speed and direction before receiving any inputs from the MPC. The below line makes the thread sleep before sending back to the simulator.



After this, the future x,y,psi,v are calculated .