The Model Predictive Control Algorithm

The MPC algorithm is an improvement over the PID Control algorithm that we implemented. At a very high level, the MPC algorithm predicts the next state in the trajectory of a vehicle, given the current state. This prediction is done by an optimization procedure. In a way, this is very similar to how Machine learning algorithms work. The prediction in a machine learning algorithm is an optimization function that minimizes the error between an expected value and the actual value. In the MPC the CTE is that error. Details on how the algorithm is implemented.

1. The current state of a given vehicle is captured. In this project we use the following state variables
    a. Px, py - the positional co-ordinates
    b. V - The velocity
    c. Psi - The orientation
    d. Cte - The cross track error that measures the difference between the desired position and the actual position, i.e expeted px, py and the actual px,py
    e. Epsi - The orientation error that measures the difference in the direction/orientation.
2. Compute the errors - cte and epsi are computed using a fitted polynomial at px=0 and the artcan of the derivative of that polynomial.
3. Predict the next state using a kinematic model
4. Minimize the cte and epsi using ipopt

The kinematic model: Uses the physical control aspects of a car namely the steering angle(delta) and the throttle(a) which is a combination of acceleration and braking. The model, as described in the course:

```
px` = px + v * cos(psi) * dt
py` = py + v * sin(psi) ( dt)
psi` = psi + v / Lf * (-delta) * dt
v` = v + a * dt
cte` = cte - v * sin(epsi) * dt
epsi` = epsi +  v / Lf * (-delta) * dt
```

The cost function and penalty weights are as given below. I ended up with the weights given below mostly by trial and error.

```
cost = A * cte^2 + B * epsi^2 + C * (v - vmax)^2 +
      D * delta^2 + E * a^2 + F * (a` - a)^2 +  G * (delta` - delta)^2
```

This is integrated over multiple timesteps. I ended up with a weight of 1 for D and a weight of 2000 for G.

Timestep length and frequency : I chose a timestep of 0.1 and a frequency of 10 as 0.1 is the actual latency between consecutive states (100 ms). I started off with a N of 25 and eventually settled for a value of 10 which seems to work quite well.