**The Model**

In main.cpp:

First the simulator provide these data

1. Reference trajectory points
   1. vector<double> ptsx = j[1]["ptsx"];
   2. vector<double> ptsy = j[1]["ptsy"];
2. Car state
   1. double px = j[1]["x"];
   2. double py = j[1]["y"];
   3. double psi = j[1]["psi"];
   4. double v = j[1]["speed"];
   5. double delta = j[1]["steering\_angle"];
   6. double a = j[1]["throttle"];

Now we have reference trajectory points and current car state we need to solve MPC model to find best car trajectory that match the reference trajectory with minimum cost. we need to find actuators values that achieve this best car trajectory

The cost we have (Cte, psi, v, steering angle(delta), acceleration(a), delta difference, a difference).

To simplify calculation we will transform reference trajectory points from global coordinates to vehicle coordinates. then find the 3rd order polynomial that fit reference trajectory points,

**auto coeffs = polyfit(ptsx\_ref, ptsy\_ref, 3);**

Then we can easily find the cte if we evaluate the polynomial at px = 0

**double cte = polyeval(coeffs, 0);**

and find orientation error by find the tangent line of reference trajectory at px = 0; which is the derivative of polynomial at px = 0; and find the psi of this tangent line

**double epsi = -atan(coeffs[1]);**

**Now we have the complete car sate at this time**

**(px, py, psi, v, cte, epsi)**

#### **Accounting for Latency**

We then need to predict the car state at time (t0 + latency(100 ms)) this will be the intial car state that will feed the MPC, by applying the kenmatic car model

**(px, py, psi, v, cte, epsi)**

**Because psi then the predicted state will be as the following**

// predict vehicle state at (current time + dt)

const double current\_px = v \* dt;

const double current\_py = 0;

const double current\_psi = - v \* delta \* dt / Lf;

const double current\_v = v + a \* dt;

const double current\_cte = cte + v \* sin(epsi) \* dt;

const double current\_epsi = epsi + current\_psi;

**( current\_px, current\_py, current\_psi, current\_v, current\_cte, current\_epsi)**

Then we solve the MPC to find the [steering angle, throttle]

// MPC Solve

auto mpcResult = mpc.Solve(currentState, coeffs);

**Timestep Length and Elapsed Duration (N & dt)**

By tuning N & dt i found value of dt = 0.2 work fine on my machine actually my machine is ubuntu on virtualbox so the graphics is very slow, I found also N = 10 is fine to drive safely also its reasonable horizon = dt \* N = 0.2 \* 10