Model Predictive Control (MPC)

MPC Algorithm calculates state variables (x, y, psi, v, CTE, epsi) for N points with delta time (dt) by optimizing (minimizing) cost with constraints. Fig. 1 shows the model, cost, and constraints. From MPC solver, we can get actuator delta and accelerator values along with predicated (x,y) points.

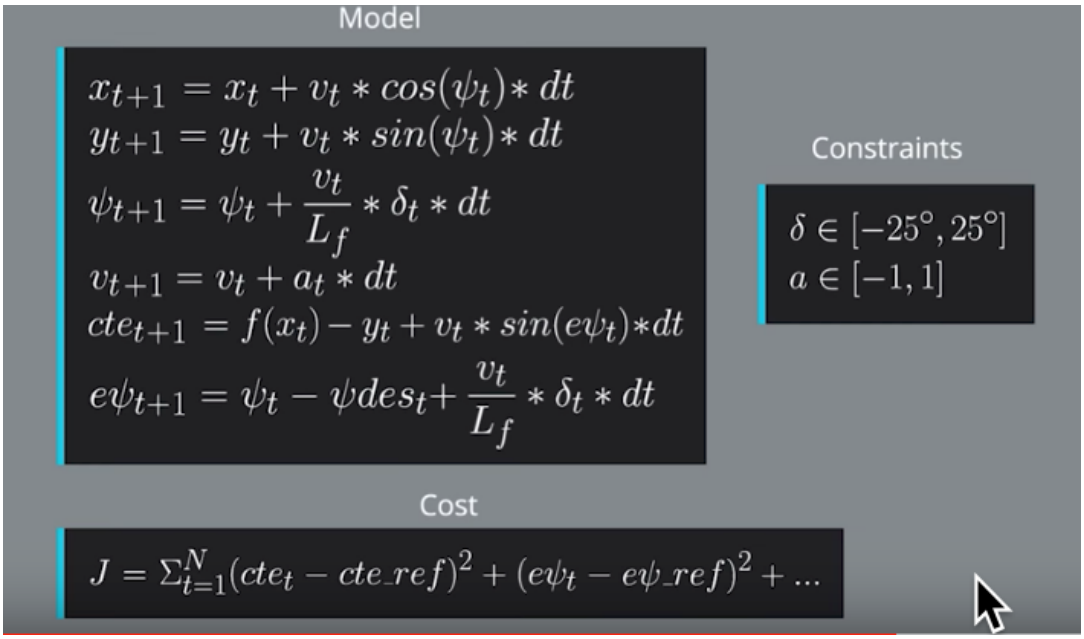


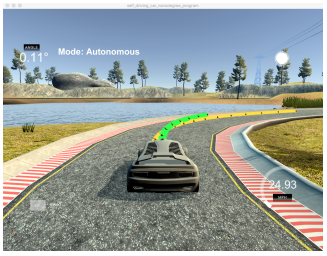
Fig. 1 Model, Cost, and Constraints

Weights of cost are important. CTE and error of PSI have the most important weight 2000 as we want to them as close as possible. I use 50 for delta angle and acceleration. I use weight 200 for sequence between delta angle change for smooth turn and 100 for acceleration change. I set the reference velocity ref_v to 40 as desired velocity.

```
// Reference State Cost
for (auto i = 0; i < N; i++) {
    fg[0] += 2000.0*CppAD::pow(vars[cte_start+i], 2);
    fg[0] += 2000.0*CppAD::pow(vars[epsi_start+i], 2);
    fg[0] += CppAD::pow(vars[v_start+i]-ref_v, 2);
}
// Miminize actuators
for (auto i = 0; i < N-1; i++) {
    fg[0] += 50*CppAD::pow(vars[delta_start+i], 2);
```

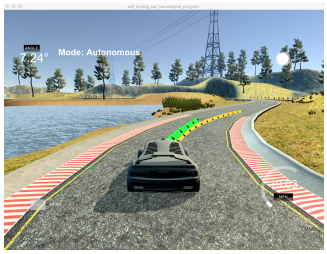
```
    fg[0] += 50*CppAD::pow(vars[a_start+i], 2);
}
// Miminize the gap of sequences of actuators
for (auto i = 0; i < N-2; i++) {
    fg[0] += 200*CppAD::pow(vars[delta_start+i+1]-vars[delta_start+i], 2);
    fg[0] += 100*CppAD::pow(vars[a_start+i+1]-vars[a_start+i], 2);
}
```

N and delta time (dt) are also important parameters to tune. I use N=10 and dt = 0.1. Larger N and smaller dt potentially give more accuracy results. However, it will take more time to comput. When I used N=11 and dt=0.09, it failed to solve because of running out of time. When a solver fails, I simply return zeros for delta and acceleration. Following images show screenshots from various N and dt combinations. N = 8 and dt=.125 still can finish safely. N = 5 fails in the beginning.



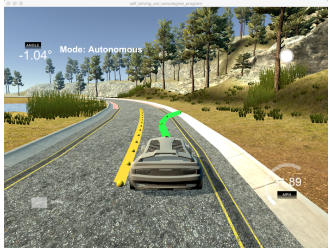
N = 10 dt = 0.1

N = 9 dt = 0.11

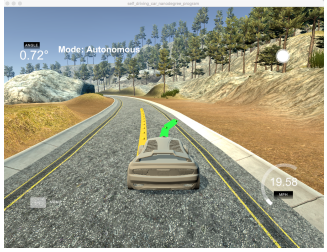


N = 8 dt = 0.125

N = 7 dt = 0.13



N = 6 dt = 0.17

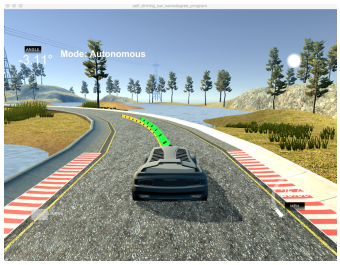


N = 5 dt = 0.2

Latency is to simulate time between actuator been commanded and taking effects. When it's too high, car might be out of control. With 150ms, it ran out of control after bridge. With 200ms, it ran out of control in the beginning. The program is default to 100ms. Car runs faster with less latency. Even 0ms, it drivers safely on my Mac.

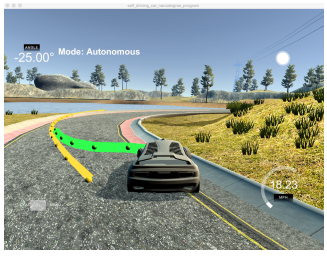
0 ms

1ms



5ms

10 ms



150 ms

200 ms

Before waypoints passed to MPC solver, we transform them from global coordinate to the vehicle coordinates. I also use the utility polynomial routine to get coefficients by order of three. Then I pass state variables and polynomial coefficients to the MPC solver.

From MPC solver, I get actuator delta angle, acceleration, and predicted positions. I need to convert delta angle from degree to radian and limit both delta and acceleration to [-1.0 … 1.0]. The predicated points will be shown in green in the simulator. I make 16 waypoints with distance of 2.0 in between in yellow in the simulator.

To ease testing, I added three optional parameters to the command line.
>./mpc <latency=100 ms> <N=10> <dt=0.1>

>./mpc 10 8 0.125
It runs 10ms latency, N=8, and dt=0.125.
>./mpc
It runs with default values, 100ms latency, N=10, and dt=0.1.