**Model details**

*In the current model, the state include the x and y, psi, speed, cte and epsi, with the size of N, and the actuators include the delta and acceleration, with the size of N-1. They are set as below.*

size\_t x\_start = 0;

size\_t y\_start = x\_start + N;

size\_t psi\_start = y\_start + N;

size\_t v\_start = psi\_start + N;

size\_t cte\_start = v\_start + N;

size\_t epsi\_start = cte\_start + N;

size\_t delta\_start = epsi\_start + N;

size\_t a\_start = delta\_start + N - 1;

*The cost for fg[0] is first set to 0.*

fg[0] = 0;

*Below are update of cost based on the reference state.*

for (int t = 0; t < N; t++) {

fg[0] += CppAD::pow(vars[cte\_start + t], 2);

fg[0] += CppAD::pow(vars[epsi\_start + t], 2);

fg[0] += CppAD::pow(vars[v\_start + t] - ref\_v, 2);

}

*The update of cost to minimize the use of actuators.*

for (int t = 0; t < N - 1; t++) {

*The update of cost to minimize the delta. In order to reduce the swing of the car from one side to another, weight of 2000 was applied in here, otherwise the car kept swinging.*

fg[0] += 2000 \* CppAD::pow(vars[delta\_start + t], 2);

*The update of cost to minimize the acceleration. Weight of 20 was applied to prevent the sudden change of speed, otherwise the speed increase too fast.*

fg[0] += 20 \* CppAD::pow(vars[a\_start + t], 2);

}

*Cost Functions to minimize the value gap between sequential actuations. Weight of 500 was applied to reduce the sudden change of delta, otherwise the steering angle might change too fast.*

for (int t = 0; t < N - 2; t++) {

fg[0] += 500 \* CppAD::pow(vars[delta\_start + t + 1] - vars[delta\_start + t], 2);

fg[0] += CppAD::pow(vars[a\_start + t + 1] - vars[a\_start + t], 2);

}

*The rest of the constraints were setup as below*

for (int t = 1; t < N; t++) {

*The state at time t+1, the same as in the lesson quiz*

AD<double> x1 = vars[x\_start + t];

AD<double> y1 = vars[y\_start + t];

AD<double> psi1 = vars[psi\_start + t];

AD<double> v1 = vars[v\_start + t];

AD<double> cte1 = vars[cte\_start + t];

AD<double> epsi1 = vars[epsi\_start + t];

*The state at time t, the same as in the lesson quiz*

AD<double> x0 = vars[x\_start + t - 1];

AD<double> y0 = vars[y\_start + t - 1];

AD<double> psi0 = vars[psi\_start + t - 1];

AD<double> v0 = vars[v\_start + t - 1];

AD<double> cte0 = vars[cte\_start + t - 1];

AD<double> epsi0 = vars[epsi\_start + t - 1];

*Setup the actuation at time t.*

AD<double> delta0 = vars[delta\_start + t - 1];

AD<double> a0 = vars[a\_start + t - 1];

*Calculate the f0 and psides0 based on the 3 order poly fit equation*

AD<double> f0 = coeffs[0] + coeffs[1] \* x0 + coeffs[2] \* CppAD::pow(x0, 2) + coeffs[3] \* CppAD::pow(x0, 3);

AD<double> psides0 = CppAD::atan(coeffs[1] + 2 \* coeffs[2] \* x0 + 3 \* coeffs[3] \* CppAD::pow(x0, 2));

*Based on the equations for the model:*

*x\_[t+1] = x[t] + v[t] \* cos(psi[t]) \* dt*

*y\_[t+1] = y[t] + v[t] \* sin(psi[t]) \* dt*

*psi\_[t+1] = psi[t] + v[t] / Lf \* delta[t] \* dt*

*v\_[t+1] = v[t] + a[t] \* dt*

*cte[t+1] = f(x[t]) - y[t] + v[t] \* sin(epsi[t]) \* dt*

*epsi[t+1] = psi[t] - psides[t] + v[t] \* delta[t] / Lf \* dt*

*Their cost were updated by the functions below.*

fg[1 + x\_start + t] = x1 - (x0 + v0 \* CppAD::cos(psi0) \* dt);

fg[1 + y\_start + t] = y1 - (y0 + v0 \* CppAD::sin(psi0) \* dt);

*Note: Multiply the steering value by -1 before sending it back to the server*

fg[1 + psi\_start + t] = psi1 - (psi0 + v0 \* delta0 / Lf \* dt);

fg[1 + v\_start + t] = v1 - (v0 + a0 \* dt);

fg[1 + cte\_start + t] = cte1 - ((f0 - y0) + (v0 \* CppAD::sin(epsi0) \* dt));

fg[1 + epsi\_start + t] = epsi1 - ((psi0 - psides0) + v0 \* delta0 / Lf \* dt);

**Reasons behind the chosen N (timestep length) and dt (elapsed duration between timesteps) values.**

*The N and dt I set is 20 and 0.1. Actually, the model also works when the N and dt are set as (10, 0.05) or (10, 0.1) or other values. When N and dt were set to different values, the weight of the cost functions need to be adjusted by case. For example, when N = 20 and dt = 0.1, the weight for fg[0] += 2000 \* CppAD::pow(vars[delta\_start + t], 2) should be set to around 2000, while when N = 10 and dt = 0.05, the weight 100 works fine for fg[0] += 100 \* CppAD::pow(vars[delta\_start + t], 2). Of course, there are different combinations of weights work for the model, as the whole model with several states and actuators could be buffered very well with their cost functions.*

*The reason I set the N = 20 and dt = 0.1 is because this combination give a pretty decent mpc trajectory, which seems long enough to predict the turns on the way while not too long to cost too much calculation power of the computer. The running result also showed that N = 20 and dt = 0.1 are a proper set of the numbers for the current cost functions.*