Writeup for Model predictive control Project

Part of Udacity self driving car nanodegree

Sven Eriksson

September 11, 2017

Text in headings and in bold comes from the project rubric. My answer to the question are written just underneath.

1 Compilation

Your code should compile. Code must compile without errors with cmake and make. Given that we've made CMakeLists.txt as general as possible, it's recommend that you do not change it unless you can guarantee that your changes will still compile on any platform.

Have not modified the CMakeList.txt and it compiles and run outside of CLion (the IDE I am using). So it should work.

2 Implementation

2.1 The Model

Student describes their model in detail. This includes the state, actuators and update equations.

Used the model described in the previous sections: 'Vehicle Models' and 'Model Predictive Control'.

State consists of 2D-position (x and y), speed (v), heading (ψ), crosstrack error (cte), and orientation error ($e\psi$) . The actuators were acceleration (a)

and steering (δ).

L_f is a vehicle specific parameter and a factor between the turning rate and the steering angle time speed. $f(x)$ is that value (distance right/left) of the polynomial at distance x. The polynomial is fitted according to given waypoint in the middle of the road that has been translated into the vehicle coordinate system. $\psi des(x)$ is the direction of the tangent to the polonomial at a certain x.

The update function was:

$$
\begin{aligned}
x_{t+1} &= x_t + v_t \cdot cos(\psi)dt \\
y_{t+1} &= y_t + v_t \cdot sin(\psi)dt \\
\psi_{t+1} &= \psi_t + \frac{v_t}{L_f}\delta_t dt \\
v_{t+1} &= v_t + a_t dt \\
cte_{t+1} &= f(x_t) - y_t + v_t * sin(e\psi_t)dt \\
e\psi_{t+1} &= \psi_t - \psi des(x_t) + \frac{v_t}{L_f}\delta_t dt
\end{aligned}
$$

2.2 Timestep Length and Elapsed Duration (N & dt)

Student discusses the reasoning behind the chosen N (timestep length) and dt (elapsed duration between timesteps) values. Additionally the student details the previous values tried.

dt was chosen to be 0.1s as that was the minimum time between actuation.

The following values for N was tested: 20, 15, 10, 8, 6, and 5.

The different values were tested with cost function tuned with N=20 and dt=0.1.

The yellow line is made up of the waypoints that are provided and the green line is made up of x and y values from different timesteps from the MPC solution.

N=20: Works well, the cars stays within the track and the green line mostly aligns with the yellow line.
N=15: Works well, the cars stays within the track and the green line mostly

aligns with the yellow line.
N=10: Works well, the cars stays within the track but the green line diverts from the yellow far ahead.
N=8: Like, 10 but the green line diverts even earlier and more.
N=6: Does not work at all, leaves track early.
N=5: Does not work at all.

N=20 was picked as it worked without any problems and I did not see any advantage of using a lower N. There is probably an advantage is reducing the number of timestep to save on computations but I had no problems with computational time.

2.3 Polynomial Fitting and MPC Preprocessing

A polynomial is fitted to waypoints. If the student preprocesses waypoints, the vehicle state, and/or actuators prior to the MPC procedure it is described.

A cubic polynomial was fitted to provided wayponts. The waypoints where first expressed in x and y of the vehicle coordinate system before the polynomial was fitted.

As 100ms latency before actuation existed I did one update of the state with the previously decided actuator values before providing the state at t+1 to the MPC.

After reading a discussion on the forum I attempted to average the actuation of the first three timesteps of the MPC. So I avarage the suggested actuations of t+1, t+2, and t+3.

2.4 Model Predictive Control with Latency

The student implements Model Predictive Control that handles a 100 millisecond latency. Student provides details on how they deal with latency.

See previous subsection, with a single state update to time t+1 before using MPC. Also taking the avarage of several timesteps of actuator values.

3 Simulation

The vehicle must successfully drive a lap around the track. No tire may leave the drivable portion of the track surface. The car may not pop up onto ledges or roll over any surfaces that would otherwise be considered unsafe (if humans were in the vehicle). The car can't go over the curb, but, driving on the lines before the curb is ok.

I decided to hand the project in with a target speed of 50mph. I did attempt to tune the cost function for higher speeds but the car did occationally leave the track. As there was no requirement on speed I left it as 50mph and continued with the other projects.