CarND-Controls-MPC

Self-Driving Car Engineer Nanodegree Program

The Model

The Model Predictive Control relies on a model to predict system state. Control signals, actuators, are optimized over a prediction horizon in order to achieve desired objectives. In this project I used a kinematic model of vehicle.

The model state consists of six parameters:

- x - x-coordinate of vehicle
- y - y-coordinate of vehicle
- psi - angle of vehicle orientation
- v - vehicle's linear velocity
- cte - cross-track error which is the distance between the center of the road and the vehicle's position
- epsi - orientation error that is difference between vehicle orientation and trajectory curvature.

Also there two actuators:

- delta - steering
- a - throttle

The next state at time step, t+1, is computed by the following equations:

$$x_{t+1} = x_t + v * cos(psi_t) * dt$$
$$y_{t+1} = y_t + v * sin(psi_t) * dt$$
$$psi_{t+1} = psi_t - \frac{v_t}{L_f} * delta_t * dt$$
$$v_{t+1} = v_t + a_t * dt$$
$$cte_{t+1} = cte_t + v_t * sin(epsi_t) * dt$$
$$epsi_{t+1} = epsi_t - \frac{v_t}{L_f} * delta_t * dt$$

subscript t - value at time t, and t+1 - value at time t+1.
dt is elapsed duration between t and t+1.
L_f is a distance between the front of the vehicle and its center of gravity.

Note that in equations for psi_{t+1} and $epsi_{t+1}$ I subtract the second term because in the simulator a positive value implies a right turn and a negative value implies a left turn.

Timestep Length and Elapsed Duration

The prediction horizon is defined by timestep length, N, and elapsed duration, dt.

The timestep length should cover 1-3 seconds of predictions. Longer prediction prone to higher accumulated errors as further predictions would add up on the errors from previous ones.

Elapsed duration discretizes the continuous model. Lower duration provides more accurate predictions, particularly at higher speed of vehicle.

However too small duration would require more computation time as optimizer would have to compute actuators for more time steps. Normally, computation time should not be greater than elapsed duration.

Initially, I set elapsed duration to 0.05 seconds and N to 20, so the prediction horizon is 0.05 * 20 = 1 second. It worked but I found that a longer prediction horizon provides better navigation. When I increased timestep length up to 30 (prediction horizon of 1.5 seconds) the computation time caused latency to the point when overall performance decreased. Then I tried elapsed duration 0.1 seconds and N = 15 (prediction horizon of 1.5 seconds). The results were satisfying and I proceeded with this settings. 0.1 seconds also corresponds to 100msec latency of actuators. Using elapsed duration less than latency leads to incorrect modeling as actuators cannot be changed for the latency period. Though this can be resolved by constraining change of actuators for the latency period at optimization if elapsed duration is small enough to discretize it.

While predicting vehicle behaviors for several steps in future allows more accurate controlling in comparison with PID controller, multiple step prediction inevitably inaccurate due to various factors, such as variations in control and computation latency, limitations of used kinematic which does not account for friction, gravity and other forces, discretization errors, and etc. To deal with this problem, model predictive control assumes that only first optimization step actuators are actually applied and the optimization for entire horizon is performed at each timestep.

Polynomial Fitting and MPC Preprocessing

The vehicle should drive along a reference path defined by waypoints. As the waypoints have discrepancy with actual trajectory I fit the waypoints to a 3d degree polynomial (main.cpp, line 168). I used the polynomial to compute cross-track and orientation errors.

It is easier to calculate the current step cross-track and orientation errors in vehicle’s coordinate system. Vehicle’s coordinate system is also used to display predicted and reference trajectories. With these assumptions I decided to perform modeling in vehicle’s coordinate system. As waypoints are provided in map’s coordinate I transform them to vehicle's coordinates (main.cpp, lines 160-163).

Model Predictive Control with Latency

As noted above predictive model may take noticeable computation time. In addition, controls may not be applied immediately and have some latency while the commands propagate through the system. The project models 100msec latency by delaying execution for 100msec prior to sending commands to the simulator.

As I have chosen elapsed duration equal to latency, I don’t need additionally constrain the optimization variables to account for the latency. However, I still need to adjust the initial state variables as the state at which command would be applied would differ from the state reported by the simulator. I use the model to predict the state of vehicle after latency period assuming that actuators are unchanged (main.cpp, lines 148-155). I do this in map’s coordinates and then use new vehicle state to transform waypoint coordinates to the vehicle’s coordinate system.

If my modeled elapsed duration, dt, would be less than latency, I would have to constrain actuator variables to stay unchanged for the duration of latency through the whole prediction horizon.

MPC in action

https://youtu.be/EP-hJTyfKNs