MPC Controller for Path Following

In this project, an MPC ("Model Predictive Control") controller is used to calculate steering and gas/brake pedal inputs for a virtual vehicle driving on a simulated test track. This MPC controller uses optimization over the space of possible future movements to guide the vehicle along a reference path (= center of the road).

Motion Model

In order for the MPC controller to be able to simulate the future behavior of the vehicle, a motion model is needed. I favored a simple kinematic motion model over a more complex dynamic motion model, because of faster evaluation times and less effort for model parameterization.

The state variables of the chosen kinematic model are x- and y-position, heading angle ψ and velocity v. The discretized state update equations from time step t to time step $t+1$ look like this:

$$\begin{aligned} x_{t+1} &= x_t + v_t * cos(\psi_t) * dt \\ y_{t+1} &= y_t + v_t * \sin(\psi_t) * dt \\ \psi_{t+1} &= \psi_t + v_t / L_f * \delta * dt \\ v_{t+1} &= v_t + a_t * dt \end{aligned}$$

Here δ is the steering angle of the front wheels, L_f is a constant factor relating this steering angle to the change in heading and a is the longitudinal acceleration. The model inputs are therefore given by δ and a, which correspond to input from steering (→ lateral motion) and input from gas/brake pedal (→ longitudinal motion).

MPC Parameters

There are two main parameters determining the accuracy and computational effort of the MPC simulation:

- The number of simulation steps N and
- The time difference dt between two simulation steps t and $t+1$

The smaller dt, the smaller the discretization error from assuming constant state values in-between simulation steps becomes. But at the same time, in order to still run the simulation for an adequate time horizon into the future (e.g. $1.5\ s$), a bigger and bigger number of simulation steps N is needed, which leads to higher and higher computational effort and therefore higher and higher execution time of the MPC control loop. Consequently, a compromise between simulation accuracy and algorithm runtime has to be found.

In this project, I settled for $N = 8$ and $dt = 0.15$. This leads to a simulated trajectory length of $1.2\ s$ which allowed the vehicle to safely and efficiently (a bit of corner cutting) traverse all segments of the test track while still having a low execution time of typically $15\ ms$ (on my PC) for one MPC control loop. I wanted the execution time to be as low as possible to avoid adding additional delay time to the system.

Before settling for my final values, I experimented with different parameter combinations. Here are some examples:
Increasing N to 14 while fixing dt at 0.15 resulted in a prediction horizon of $2.1\ s$. This increased the average MPC loop time to about $25\ ms$ which still doesn't sound very high compared to $100\ ms$ of

actuator delay. However, when entering a narrow curve, the optimization algorithm now had spikes of up to $150\ ms$ execution time, which resulted in the vehicle oscillating and sometimes even leaving the road completely.
Fixing N at 8 and decreasing dt to $0.05\ s$ gave a prediction horizon of only $0.4\ s$. The problem now was that the trajectory, that has been optimized by the MPC algorithm, considered changes in the reference path (center of the road) very late, so that the vehicles driving seemed jittery and not very natural or efficient.

Time Delay

As I have mentioned earlier, the vehicle simulator features an artificial time delay of $100\ ms$ between setting control signals (steering + accelerating/decelerating) and the vehicle receiving these signals. This means, that the vehicle is essentially driving along the track using "old" input signals that have been calculated at least $100\ ms$ ago.

If this time delay is not accounted for, the vehicle becomes unstable or even crashes at higher velocities. Therefore, the current vehicle state $[x, y, \psi, v]$ is predicted into the future using the state update equations (see section "Motion Model"), bevor passing it to the MPC algorithm. In addition to the actuator time delay, the prediction also incorporates the average execution time of the MPC control loop.

Coordinate Transformation

To provide the MPC algorithm with a reference trajectory, the waypoints of the reference path (= center of the road) are transformed into the predicted vehicle coordinate system (i.e. the vehicle coordinate system that has been accounted for time delays) and a cubic polynomial is fitted through them. This polynomial is also used to approximate the initial cross track error and orientation error before finally calling the MPC solver.