Self-driving Car Nanodegree

Project: Model Predictive Control

Model

State of the car:

Car's state at a given time consists of:

- spatial position (px, py),
- axle orientation ψ,
- speed v.

The values of the first two components depend on the choice of a stationary coordinate system.

Actuators:

The state of the car is controlled using two actuator inputs, which are constrained to have values between -1.0 and 1.0:

- throttle, a,
- steering, $-\delta$.

Update equations:

Given the current state of the car and the actuator inputs, the state of the car after time dt is predicted by assuming the kinematic model discussed in the lectures. This model implies the following updates for the state of the car:

$$
\begin{aligned}
px &\leftarrow px + v * cos(\psi) * dt \\
py &\leftarrow py + v * sin(\psi) * dt \\
\psi &\leftarrow \psi + \frac{v}{L_f} * \delta * dt \\
v &\leftarrow v + a * dt
\end{aligned}
$$

Control

Assuming the above model and a reference trajectory, the goal is to find the best actuator inputs to control the state of the car so that the car stays as close and as aligned as possible to the reference trajectory. In other words, the following errors must be as close as possible to 0:

- cross-track error, cte: shortest distance from the car to the reference trajectory,
- orientation error, $e\psi$: angle between car's axle and the reference trajectory.

Remark that the kinematic model described above implies that these errors evolve as follows:

$$cte \leftarrow cte + v * sin(e\psi) * dt$$
$$e\psi \leftarrow e\psi + \frac{v}{L_f} * \delta * dt$$

To address the control problem, the state of the car is predicted over a certain time in future, T, by assuming a set of actuator inputs applied N times at time intervals of length dt, and the predicted state-trajectory is assigned a cost function which takes into account

- the cross-track and orientation errors:
$$C_{cte} = \sum_{i=0}^{N-1}(cte_i)^2, \quad C_{e\psi} = \sum_{i=0}^{N-1}(e\psi_i)^2,$$
- deviations from a reference speed, v_{ref} :
$$C_v = \sum_{i=0}^{N-1}(v_i - v_{ref})^2,$$
- and moderation of
  - actuator inputs:
  $$C_a = \sum_{i=0}^{N-1}(a_i)^2, \quad C_\delta = \sum_{i=0}^{N-1}(\delta_i)^2$$
  - and their transitions:
  $$C_{diff_a} = \sum_{i=0}^{N-2}(a_{i+1} - a_i)^2, \quad C_{diff_\delta} = \sum_{i=0}^{N-2}(\delta_{i+1} - \delta_i)^2$$

Additionaly, a regularization term, λ, is used to emphasize or de-emphasize the contribution of the different cost components to the total cost.

$$C = C_{cte} + C_{e\psi} + \lambda * C_{diff_\delta} + \frac{1}{\lambda}(C_v + C_a + C_\delta + C_{diff_a})$$

The control problem now reduces to finding the optimal set of actuator inputs so as to minimize the above cost function, subject to the constraints on the actuator inputs. The optimal actuator inputs thus obtained for the first step are then passed on to control the car in the current moment.

Regularization term (λ)

To prevent the car from wobbling, it is particularly important to emphasize the cost associated with steering transition, C_{diff_δ} . However, it is also important not to over-emphasize this cost component, so that the large steering transitions remain possible at sharp turns.

$\lambda = 10.0$ works well in this project.

Timestep Length and Elapsed Duration (N & dt)

Remark that $T = N * dt$.

Suppose that the reference trajectory is given by the simulator for a distance, d, which can be small near turns and large along straight paths. Assuming an average speed of around v_{ref} , this imposes a natural upper limit on T of around $\frac{d}{v_{ref}}$. This upper limit in the current project is observed to be around 1.2.

At the same time, it is important that T is large enough so that the current actuator inputs are sufficiently forward-looking to anticipate the future curvature of the road. A reasonable lower limit here would be 0.8.

Now, N and dt could be tuned for this range of T.

A smaller dt means more frequent actuations, and therefore more flexibility to approximate the reference trajectory with desirable speed and actuator attributes. However, a smaller dt forces N to be large in order to maintain T within a reasonable range, thus increasing the computational cost, and hence, latency in a real-world scenario. Another reason why a small dt might lead to a wobbly motion of the car in the current project is because the entire MPC computation is done from an estimated post-latency initial state: therefore, a progressively smaller dt doesn't lead to a corresponding gain in accuracy, but it probably leads to a loss in robustness.

It is also observed that N should be greater than or equal to 10 for a good approximation of the reference trajectory.

In the current project, the range of values which are observed to work fine are: dt between 0.08 and 0.12, N between 10 and 15, subject to T between 0.8 and 1.2.

Latency

A straigtforward way to deal with a latency of 0.1 is to estimate the state post latency from the current state and the current actuators using the kinematic model described earlier. This estimated state post latency is then taken to be the initial state from where all further processing is done.

Polynomial Fitting and MPC Preprocessing

After estimating the car's state post latency, the coordinate frame is then shifted to that of the car in this state. In particular, this implies that the post latency (px, py, ψ) is $(0, 0, 0)$ in the new coordinate system. The waypoints for the reference trajectory given by the simulator are also transformed to the new coordinate system, and the reference trajectory is approximated by a 3rd degree polynomial. After computing cte and $e\psi$, a vector consisting of these two error terms and the car's state is fed to the MPC optimizer, which returns the optimal actuators for the post latency state of the car.