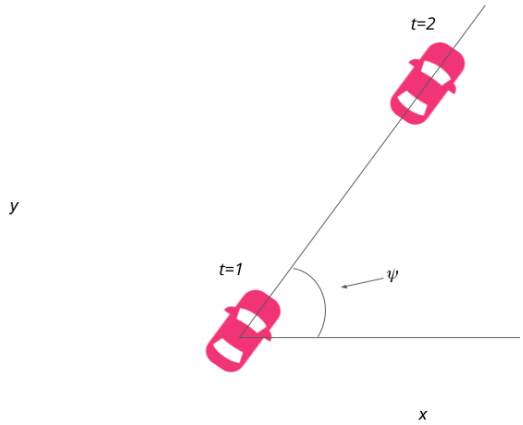
Model Predictive Control Project

Vehicle Model:

Global Kinematic Model



$$x_{t+1} = x_t + v_t cos(\psi_t) * dt$$

$$y_{t+1} = y_t + v_t sin(\psi_t) * dt$$

$$\psi_{t+1} = \psi_t + \frac{v_t}{L_f}\delta * dt$$

$$v_{t+1} = v_t + a_t * dt$$

Source: *Udacity*

Controller Model:

The controller mainly consists of three constructs:

1. Constraints
2. State and Actuator variables
3. Cost equation

1. Constraints: These are the equations that the model must adhere by while solving for outputs. These comprise of state space and error update equations. e.g. a constraint based on state representing vehicle 'x' position can be represented as:

$$x_{t+1} = x_t + v_t * cos(\psi_t) * dt$$

which, if to be satisfied, can be put as:

$$x_{t+1} - (x_t + v_t * cos(\psi_t) * dt) = 0$$

This is repeated for all four states as well as cross track error (cte) and vehicle orientation error (e_psi).

2. State and Actuator Variables: The state variables comprise of vehicle (x,y) position, velocity and orientation. The actuator variables are throttle (a) and steering angle (delta).

3. Cost equation: The optimizer tries to minimize the cost by finding the optimum set of actuator variables. The cost is a sum of 'undesirable' constructs, for instance, squared sum of cross track error. The following terms are considered for cost calculation, which the solver will try to minimize:

a) Cross track error squared

b) Vehicle orientation error squared

c) actuator command (steer and throttle) squared (to minimize big actuator commands)

d) actuator command (steer and throttle) delta squared (to minimize big actuator command rates)

e) speed error squared (to minimize error in speed tracking)

The above-mentioned terms are further multiplied with gains as a result of parameter tuning, thus prioritizing one term over another while doing cost calculation.

N and dt

The N and dt values were finalized to be 10 and 0.05 s respectively. The idea was to have a long enough horizon estimation (0.5 s in this case) for the vehicle to have ample track stretch to estimate the average vehicle performance and decide the optimum control outputs. The constraint on these parameters is the computational cost, which scales with N and dt. With too big N (and/or too small dt), the controller would fail to do the real time computation for a given horizon length. While at the same time, having a large dt would introduce inaccuracy in the model estimation, as it assumes the control inputs to be constant over the dt time period, assuming the linear variation of states during this time period, thus, longer the dt is, more inaccurate would be this assumption.

Polynomial Fitting and MPC Preprocessing

The reference way-points and vehicle position received from the simulator are in global coordinate system. Once the relative position of way-points w.r.t vehicle position, as expressed in vehicle position is calculated, is then translated to vehicle coordinate system (using vehicle orientation 'psi'). Furthermore, the relative position of way-points is also fitted into a 3^{rd} order polynomial, which is then used to for doing further calculations like cross track error and vehicle orientation error.

Model Predictive Control with Latency

There are two sources of latency observed in the simulation. One is introduced manually (a 100 ms sleep) in the simulator to simulate a real system where actuator have a lag between command and actual actuation while the other is specific to the Ubuntu virtual machine, which when communicates with the simulator running in windows, shows sum latency, which results in a delay in state feedback values coming from simulator and actuator commands going to it. To counter this, there is a two-step adjustment being done in the code.

First is in the main.cpp, where, we update the state feedback, accounting for the time elapsed during latency, and thus, sending in these updated (or extrapolated) values to the optimizer. Secondly, in the Solver, we introduce a delay (equivalent to latency) specifically in the actuator commands when modeling the constraints, so that the optimizer is forced to account for latency in its calculations.