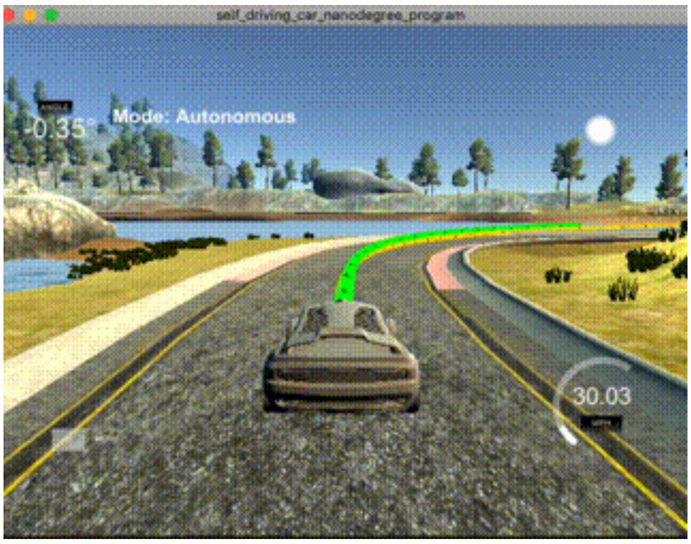
# Term 2, Project 5: Model Predictive Control (MPC)

Overview of the result:



## Project Setup

The project was based on a simulator based on the Unitiy engine, that connects with the created MPC via websocket. At each tick, the simulator sends the current state and actuator values, along with the next 6 waypoints of a given reference trajectory. On the other hand, the MPC needs to provide the actuator values to the simulator as response, which were throttle and steering angle in this project.

## Overview of the MPC

A model predictive controler derive the controls to be taken in the current state with the help of a given reference trajectory and a Constraint Optimization Problem (COP).

The model predictive controler solves a COP at each time it is executed, i.e. asked for the next control commands. The COP models the predicted next `N` states of the car as variables that need to be assigned with values. It is the task of the used constrant solver to find these values. The timespan between the states is `dt`. Both, `N` and `dt` are parameters, that need to be tuned.

The variables cannot take random variables, as the they should realistically model the next steps of our vehicle, who's movements follow a specific movement model. Thus, the value of a state at timestamp `t` is constrained by the value of the preceeding state at timestamp `t-1` and the controls taken at the preceeding timestamp `t-1`. To implement this, the COP introduces `N-1` bilateral constraints between each pair of neighboring states in the prediction horizon. The constraints used in this project are defined by the Kinematic Model equations, i.e. a movement model for vehicles that does not consider forces and usually works well at low speeds.

Every instantiation of the constrained variables (i.e. our states) that satisfies the imposed constraints would be a valid trajectory that our vehicle could theoretically follow. The controls in the first state of such an instantiation would be the next controls that our vehicle should take. However, we don`t want to follow the vehicle a random trajectory, but the one that fits best to the defined reference trajectory.

For that reason, a `cost variable` is introduced, that considers the deviation of the trajectory suggested by the state variables of the COP and the reference trajectory, as well as characteristics of the given trajectory suggested by the state variables of the COP, e.g. how smooth it turns around curves and how well it sticks to a reference velocity. Finally, we search for that instantiation of the variables that satisfies the constraints of the movement model between suceeding states and minimizes the `cost variable` at the same time.

Compared to a PID controler, the MPC brings several advantages:

- latency of actuators can be considered in the controls
- Multiple Input / Multiple Output controllers can be naturally modelled by a PID controller, contrary to PID-based systems, where several PID controllers would need to be tuned independently but mutually influencing each others input and output. This renders the configuration process very complex.

## The Kinematic Model

Two movement models are for choice, in order to model the bilateral constraints between succeeding vehicle states of the COP:

- A kinematic model, that does not respect any forces like grip of the tires or centrifugal fores.
- A dynamic model, that respects all involved forces.

A state at time t contains the following elements:

- x_{t}: The x coordinate of the vehicle at time t
- y_{t}: The y coordinate of the vehicle at time t
- \psi_{t}: The yaw angle of the vehicle at time t
- v_{t}: The speed of the vehicle at time t
- cte_{t}: The cross-track error of the vehicle at time t
- e\psi: The deviation of the yaw angle of the vehicle from the desired yaw angle

The domain of each of this variables are the real numbers.

Furthermore, the car has two actuators: throttle and steering angle. Thus, two controls are to be taken at a state at time t comprise:

- \delta_{t}: The steering angle to be applied at timestamp t, values are angles in rad, representing

angles between -25 to +25 degrees. These boundaries are vehicle-specific.
- a_{t}: The acceleration control to be applied at timestamp t, values are in the interval [-1; 1], also determined by the actuator of the vehicle.

For sake of simplicity, this project applies a kinimatic bicycle model with the following constraints:

- x*{t+1} = x*{t} + \v * \cos(\psi_{t}) * dt
- y*{t+1} = y*{t} + \v * \sin(\psi_{t}) * dt
- v*{t+1} = v + a*{t} * dt
- \psi*{t+1} = psi*{t} - (\frac{v*{t}}{Lf}) * \delta*{t} * dt
- cte*{t+1} = (f(x*{t}) - y*{t}) - v*{t} * \sin(e\psi) * dt
- e\psi*{t+1} = (\psi*{t} - \atan(f'(x*{t})) + v*{t} * (\frac{v_{t}}{Lf}) * dt

Here, Lf is a characteristic of the vehicle, i.e. the distance of the front axle to the center of mass. f(x) is a 3rd order polynomial fitted to the next 6 given waypoints of the track. f'(x) is the first derivative of this function and accordingly \atan(f'(x)) is the angle of the slope at point x.

## The Cost Function

The defined COP optimizes a cost function, that sums over all states N:

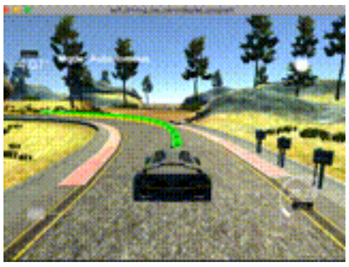
- The squared CTE of each state, weighted by a factor of 20.
- The squared e\psi of each state, weighted by a factor of 20.
- The squared difference of v_{t} from a reference speed, set to 50., weighted by a factor of 1.5
- The squared difference of steering control \delta of the current state and that of the previous state, weighted by a factor of 600.
- The squared value of the steering angle \delta, weighted by a factor of 20

The cost for deviation from the reference speed is necessary to prevent the car from stopping in case it predicts a perfect trajectory with minimal cte and e\psi on straight parts of the track. Alternatively, the distance to the furthest away waypoint could also be punished.

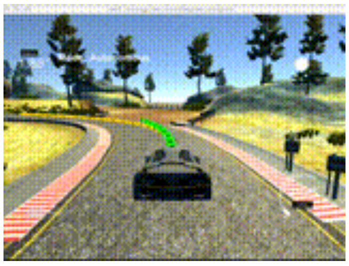
Without imposed latency, the last two cost factors were not necessary to drive around the track savely. However, with the imposed latency of 100ms, the car started to oscillate extremely, which could be compensated by adding these factors and tuning the lengths of the look-ahead horizon, as described below.

## Choosing the Paramaters

The final choice for the parameters was `N=20` and `d=0.1s`, which corresponds to a look-ahead horizon of 2s:

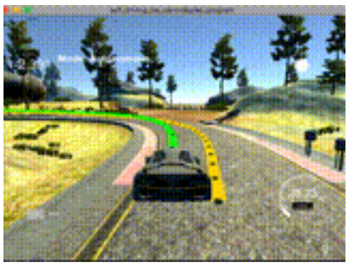


With values of `N=10` and `d=0.1s`, the look-ahead of the optimizer was to short, especially when adding the 100ms latency. In that case, the vehicle took quite extreme steering maneuvers, even when punishing that in the cost function. Result was a vehicle oscilating around the reference trajectory and a much lower speed:



The background of this phenomena is, that in case of a short look-ahead horizon like `N=10` and `d=0.1s`, a higher cte must be compensated faster than in a longer look-ahead horizon, where the later states would yield a low cte resulting in a low overall cte.

In case of longer look-ahead horizons like 3s with like `N=12` and `d=0.25s`, the constraint solver was not able to yield suitable solutions within the given time, resulting in unsafe steering behaviour:



## Preprocessing of Waypoints

Before applying the MPC procedure, the waypoints received from the simulator were transformed to the vehicle coordinate system, i.e. an system where the vehicles center of mass is at the origin, the x-axis describes the longitudinal axis of the vehicle and the y-axis the latitudinal axis of the vehicle. This simplifies the update equations and helps to fit the polynomial to the given waypoints, which would not be possible if the waypoints would all have the same y coordinate.

## Handling Latency

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

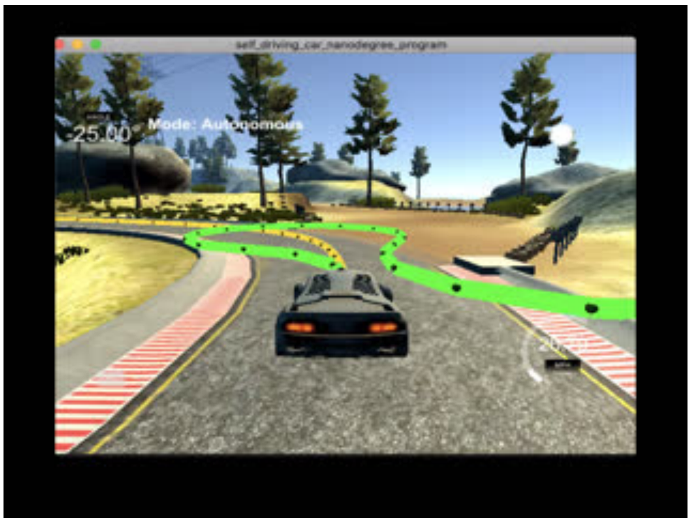
On assumption of this project was, that the actuators of the vehicle have a latency of 100ms. To cope with that, the MPC is not directly applied to the current state received from the vehicle, but to prediction that was created ased on the current state and the equations of the kinematic bicycle model with a value for dt of 100ms.

This ensures, that the MPC finds that controls, that fit that state that fit the estimated future state (100ms later) of the vehicle at best.

## Further Tweaks

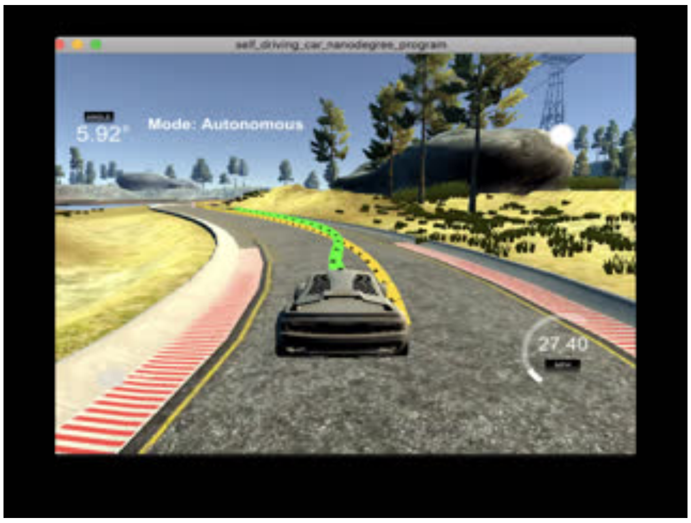
In order to accelerate the constraint solving process, the variables for the MPC executed at timestamp t are initialized with the solution of the MPC executed at time t-1. This helps to reduce find good solutions faster in sharp curves.

Initialization with 0 (without the twek applied) occasionally led to situations where no proper solution was found by the solver:



Initialization with best solution of last state:

Further, the constraints imposed by the equations of the kinematic bicycle model were transformed to soft-constraints, allowing the constraint solver a tollerance of +- 0.01 in the solution. This further improved the behaviour in steep curves. Not using this tweak resulted in more inaccurate trajectories:


self_driving_car_nanodegree_program
ANGLE
5.92°
Mode: Autonomous
27.40
MPH