

# Implementation

## The Model

The model used is a Kinematic model neglecting the complex interactions between the tires and the road. The model equations are as follow:

```
x[t] = x[t-1] + v[t-1] * cos(psi[t-1]) * dt
y[t] = y[t-1] + v[t-1] * sin(psi[t-1]) * dt
v[t] = v[t-1] + a[t-1] * dt
psi[t] = psi[t-1] + v[t-1] / Lf * delta[t-1] * dt
cte[t] = f(x[t-1]) - y[t-1] + v[t-1] * sin(epsi[t-1]) * dt
epsi[t] = psi[t] - psides[t-1] + v[t-1] * delta[t-1] / Lf * dt
```

Where the state of the model:

- $x, y$  : Car's position.
- $\psi$  : Car's heading direction.
- $v$  : Car's velocity.
- $cte$  : Cross-track error.
- $\epsilon$  : Orientation error. two values are the model output:
- $a$  : Car's acceleration (throttle).
- $\delta$  : Steering angle.
- $L_f$  is the distance between the car of mass and the front wheels.
- The objective is to find the acceleration (  $a$  ) and the steering angle(  $\delta$  ) in the way it will minimize an objective function that is the combination of different factors:
- Square sum of  $cte$  and  $\epsilon$  .

## Timestep Length and Elapsed Duration (N & dt)

- The prediction horizon is defined by the number of points(  $N$  ) and the time interval(  $dt$  ).
- After trying with  $N$  from 10 to 20 and  $dt$  100 to 500 milliseconds, to have a better result tuning the other parameters I decided to choose them fixed to 10 and 100 milliseconds.
- The number of points impacts the controller performance as well. I tried to keep the horizon around the same time the waypoints were on the simulator.
- With too many points the controller starts to run slower, and some times it went wild very easily.

## Polynomial Fitting and MPC Preprocessing

- The waypoints provided by the simulator are transformed to the car coordinate system in main.cpp.
- A 3rd-degree polynomial is fitted to the transformed waypoints. These polynomial coefficients are used to calculate the  $cte$  and  $\epsilon$  later on.
- They are used by the solver as well to create a reference trajectory.

## Model Predictive Control with Latency

- To handle actuator latency, the state values are calculated using the model and the delay interval. These values are used instead of the initial one.
- The code implementing that could be found in main.cpp.