## The Model

The model used is a Kinematic model where state includes:

* position `x`
* position `y`
* orientation `psi`
* velocity `v`
* cross-track error `cte`
* orientation error `epsi`

and control inputs:

* steering angle `delta`
* acceleration `a`

The model of the vehicle follows equations:

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]/Lf \* delta[t] \* dt;

v[t+1] = v[t] + a[t] \* dt;

cte[t+1] = f(x[t]) - y[t] + v[t] \* sin(epsi[t]) \* dt;

epsi[t+1] = psi[t] - psi\_des + v[t]/Lf \* delta[t] \* dt;

where values at timestep [t+1] is based on values at timestep [t] after dt seconds, Lf is the distance between the front of the vehicle and the center of gravity

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

The prediction horizon `T` is the product of `N` and `dt` where *N* is the number of timesteps in the horizon, *dt* is how much time elapses between actuations. Initially started with `N`=20, `dt`=0.05. With higher `N` values (20, 25), the vehicle went above the reference trajectory and would begin to oscillate wildly and drive off the track whereas lower value fell short causing the vehicle to drive off the track straight away. Finally, the choice with `N = 10` and `dt = 0.1` seemed to give good result.

## Polynomial Fitting and MPC Preprocessing

The waypoints from the simulator are transformed to the vehicles coordinate system which are then fitted to a polynomial of 3rd order. The output polynomial coefficients are used to initialize the states and passed to the MPC to find the trajectory.

## Model Predictive Control with Latency

To counter the latency problem, the state values are calculated using the model with delay interval. These values are used forced as the initial states.