MPC Project

The goals / steps of this project are the following:

- Implementing Model Predictive Control to drive the car around the track safely.
- The CTE will be calculated given the readings from the simulator.
- At each iteration, the current state of the vehicle is fed to the MPC Solve function, from which we calculate the new steering angle and acceleration.
- We also have to handle the 100 ms latency between actuations commands.

Rubric Points:

- **The Model.**

  We are using a kinematic model in this project, where all dynamic forces are neglected (gravity, friction, etc..). The vehicle state consists of 6 parameters: x position (x), y position (y), heading angle (Ψ), velocity (v), cross-track error (cte) and the orientation error (eΨ).

  The car is controlled by two actuators: steering wheel (δ) that controls the heading angle of the car, and the gas/break pedals (a) (which we consider here as one) controlling the acceleration.

  At each time step, the state is updated as follows:

  - $x_{t+1} = x_t + v_t * \cos(\Psi_t) * dt$
  - $y_{t+1} = y_t + v_t * \sin(\Psi_t) * dt$
  - $\Psi_{t+1} = \Psi_t + v_t * \delta_t / L_f * dt$
  - $v_{t+1} = v_t + a_t * dt$

- $cte_{t+1} = cte_t + v_t * \sin(e\Psi_t) * dt$
- $e\Psi_{t+1} = e\Psi_t + v_t * \delta_t / L_f * dt$

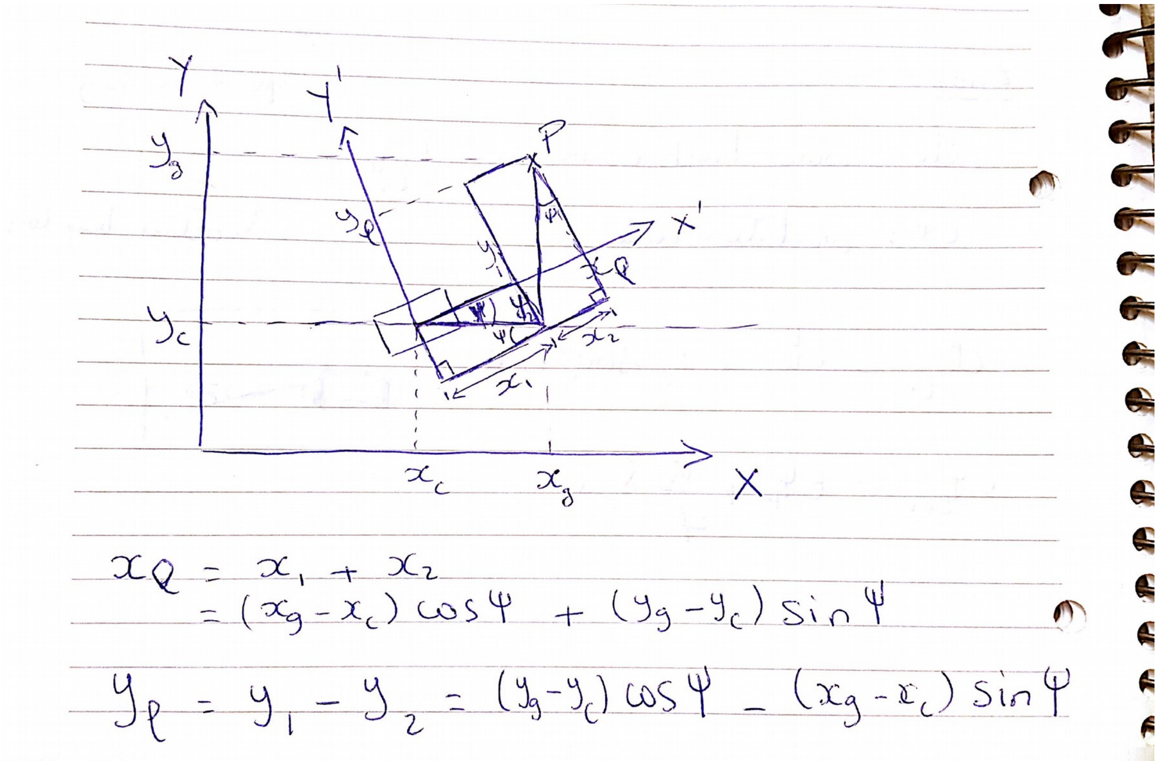
Note: L_f is the distance between the front of the vehicle and its center.

- **Timestep length and elapsed duration (N & dt).**

  These hyperparameters were chosen by trial and error. I followed the advice of the lectures that T (N*dt) should be a maximum of a few seconds, so I set it to 1.25 seconds as in the MPC Quizz (N =25 and dt = 0.05). But I didn't get good results.
  So I tried many different values for T and different combinations of N and dt such as:
  - T = 4, N = 20, dt = 0.2
  - T = 3, N = 20, dt = 0.15
  - T = 3, N = 15, dt = 0.2
  - T = 2, N = 10, dt = 0.2
  - T = 2, N = 20, dt = 0.1
  - T = 1.5, N = 15, dt = 0.1
  - **T = 1.5, N = 10, dt = 0.15**

  The last combination gave me the best results, where the car was able to drive around the track at a speed of 50.

- **Polynomial fitting and MPC preprocessing**
  Since the server sends positions in map or global coordinates, we have to transform them to car coordinates. This is done by simple geometric and trigonometric equations shown in the following image:



Where P is the waypoint we want to transform, x_g and y_g are its global coordinates, while x_l and y_l are its local coordinates which we want to calculate. X_c and Y_c define the car position in global coordinates. And Ψ is the angle of the car's heading.

- **Model Predictive Control with latency.**
  To handle the 100ms latency between actuators commands, I combined two methods suggested in the forums, the first is to use a dt larger than the latency, so I chose 0.15. The second is to use the previous values of the actuators (from the previous timestep). When I applied only of these methods, I couldn't get good results, but combining the two made the car drive smoother.