MP2

problem 4

For the longitudinal controller, we computed the curvature of the track first using a difference of target angles. Let ϕ_1 be the world-coordinate angle between the car and the next target waypoint. Let ϕ_2 be the world-coordinate angle between the car and the second next waypoint.

$$\phi_1 = \text{atan2}(x_c - x_{r1}, y_c - y_{r1})$$
$$\phi_2 = \text{atan2}(x_c - x_{r2}, y_c - y_{r2})$$

Then the target speed should be a function of the difference of these angles:

$$u_v = s(|\phi_2 - \phi_1|)$$

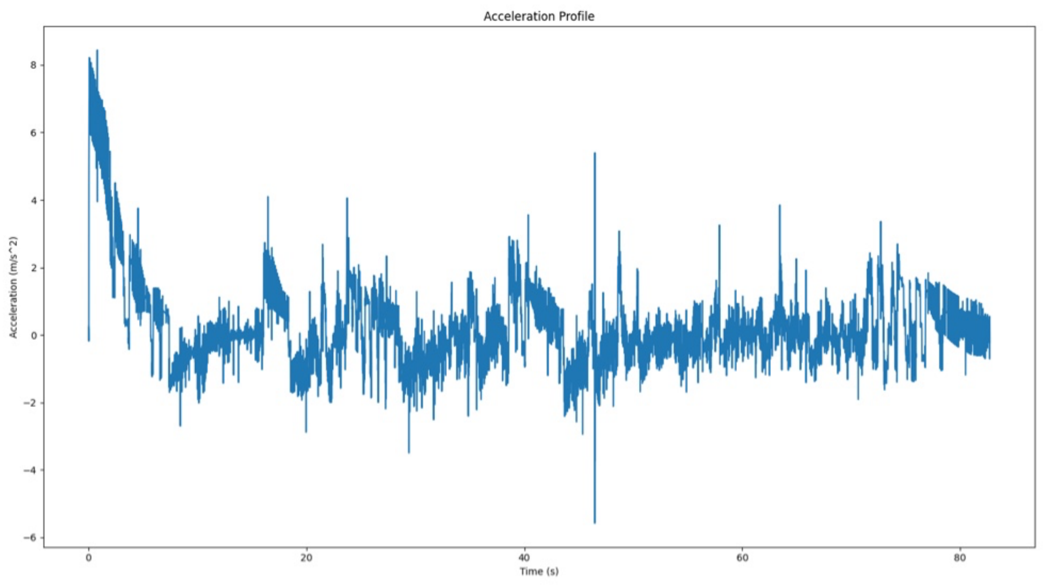
For our implementation, we simply derived a lookup table:

$s(\theta)$ m/s	θ rad
20	[0, 0.03)
18	(0.03, 0.04)
16	(0.04, 0.1)
15	(0.1, 0.15)
14	(0.15, 0.25)

The reasoning behind this algorithm is we want to calculate the rate of steering in the immediate future. If the angle difference is large, then the car will need to steer hard after reaching the next waypoint. If the angle difference is small, then the future trajectory is likely straight. Note that this algorithm is dependent on the spacing between the waypoints, thus care must be taken when generating the waypoint list.

To improve efficiency, we started with a highly conservative LUT of linear speeds ranging from 5-10 m/s. As we tested the algorithm, we gradually raised the speed at each angular difference until the car became unstable. Then, we lowered the specific speed setpoints that caused instability and repeated the process. Lastly, we lowered the speed setpoints that the car was struggling to meet.

problem 5



Our controller crossed the threshold of 0.5G when it was just launched. The vehicle started at a straight line, so we expected it to reach the maximal speed there, which lead to a high acceleration. After that the acceleration almost never exceed the threshold.

problem 6

For the lateral controller, we decided on a weighted interpolation between the next two target waypoints. The weight follows the equation:

$$\alpha = wr_1 + (1-w)r_2$$
$$w = min(0, 1 - \frac{1}{1+l_d-D})$$

Where D is the “proximity distance” at which the car considers a waypoint “met” and stops pursuing it. This function produces a number that asymptotically approaches 1 as l_d goes to infinity and rapidly drops to 0 when $l_d \leq D$. The reasoning is that we want to steer towards the next waypoint as accurately as possible until it is no longer safe to do so, in which case the car should ignore the waypoint and smoothly transition to following the next one after.

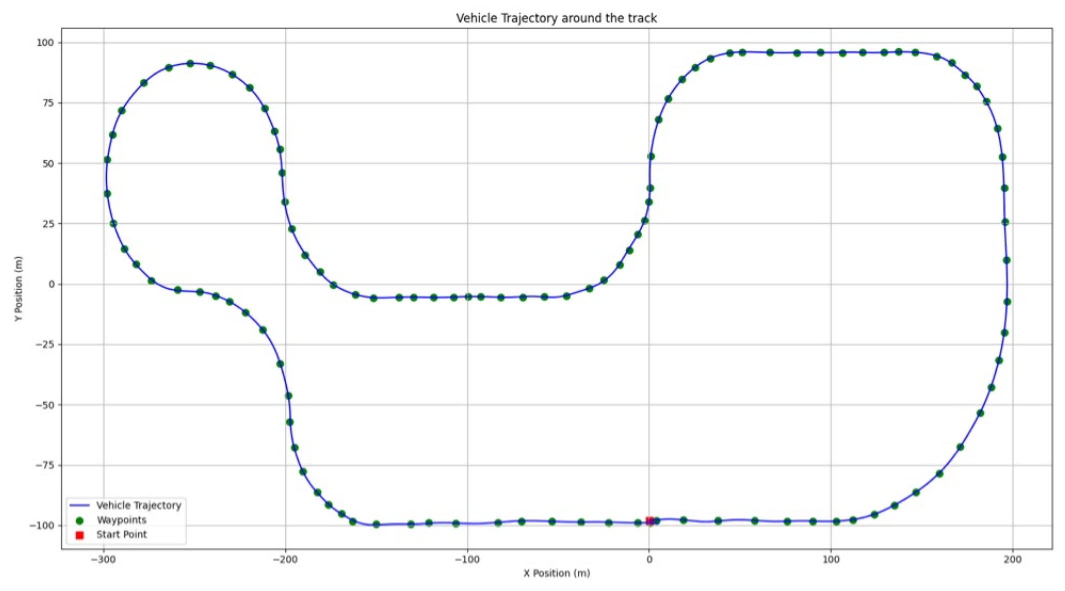
We noticed that the pure pursuit equation encounters a Lipschitz discontinuity as the distance l_d approaches 0, causing instability in the car’s steering. Thus, we quickly discarded the option of looking ahead to the next waypoint. In fact, we specifically guarded against this scenario by introducing a “minimum lookahead distance” D to prevent the car from pursuing any waypoints that are too close.

While we carefully considered the option of looking a fixed distance ahead, we decided that this algorithm may cause the car to miss the immediate next waypoint and moved to a nonlinear weight model.

We also considered the third option of looking a velocity-dependent distance ahead, as rapidly changing velocity could also rapidly change the lookahead angle, causing even more instability. This option could also be algorithmically expensive to implement.

We believe that an ideal algorithm will be a function of the positions of the next N waypoints as well as a “trajectory tolerance” term, rather than vehicle state. This is because arriving at the waypoints is a hard requirement, while increasing speed is only a secondary goal. The function would efficiently compute the optimal steering based on a weighted average of the angles to the next N waypoints, with the required tolerance deciding the weights. In addition, the steering error term would also be communicated to the longitudinal controller for deciding the target speed.

problem 7



problem 8

https://github.com/Yao-Xinchen/ECE484/blob/master/mp2/mp2.webm

mp2.webm
File mp2/mp2.webm
Yao-Xinchen / ECE484 Branch master