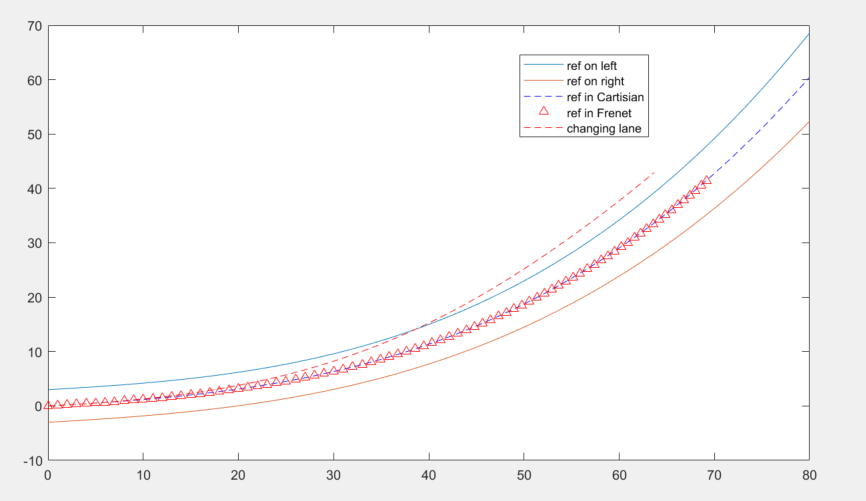
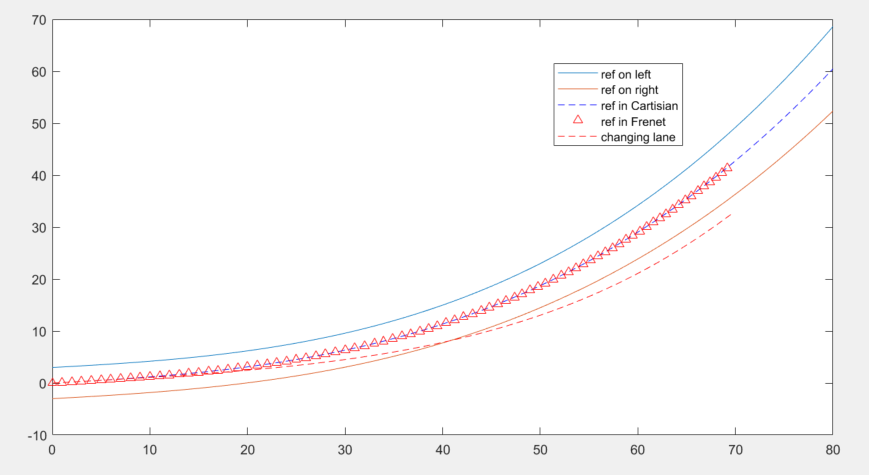
Test report

This report offers the test result in trajectory generation functionals.

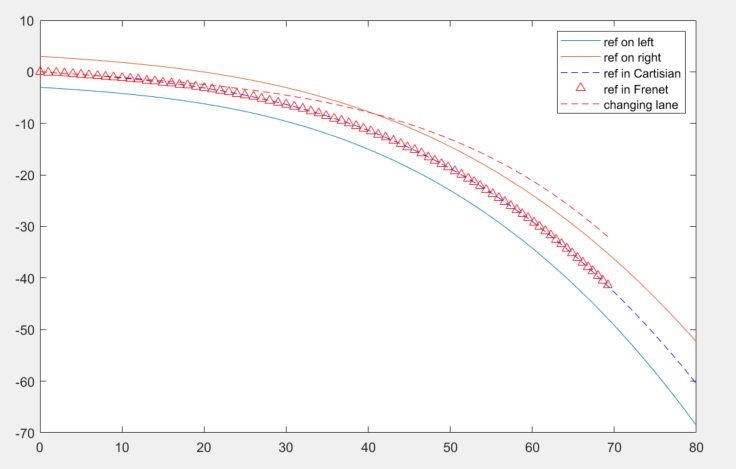
1. Lane changing functions test
2. Lane change trajectory generation test. We generate the four cases where we apply lane change functionals.
   1. Vehicle turning left, with lane changing left



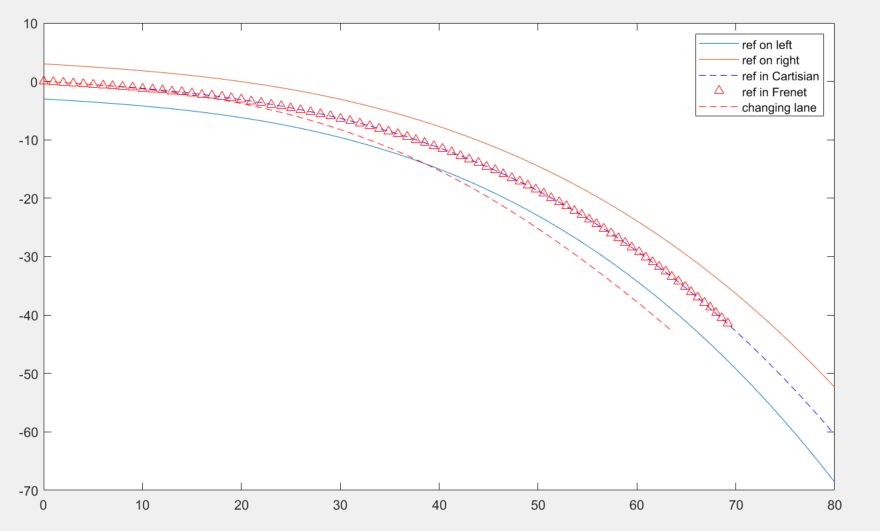
* 1. Vehicle turning left, with lane changing right



* 1. Vehicle turning right, with lane changing left



* 1. Vehicle turning right, with lane changing right



1. Comparison between two different center line generation

We have two difference methods in generating a sequence of center line waypoints for future use. One is, based on current polynomials of lanes on the left and right, discretely sample on Cartesian coordinate along x axis to get the y axis value, and calculate its mean value. And the other method is discretely sample under Frenet coordinate, which means it integrate along s direction. At each sample point, get the respective x value on each lane polynomial. We provide both methods realization and make comparison numerically.

* 1. Using Cartesian coordinate to generate a sequence of center line waypoints.



* 1. Using Frenet coordinate to generate a sequence of center line waypoints.



As you can see in the figure, the upper triangle line is using Frenet method, while the blue dashed line is using Cartesian method. They almost overlapped. As for the calculation time, we also make a comparison as follows:

The prerequisite is that both methods generate a sequence of trajectory of 80m, and the sampling resolution is about 0.2m, which means about 400 points will be generated. The calculation time is (with CPU performance i7-9750H @2.60GHz and RAM 32G):

Using Cartesian method:

*reference\_1 generation time is: 1.680400 ms*

Using Frenet method:

*reference\_2 generation time is: 4.659700 ms*

1. Kappa calculation

In our module, we provide two difference kappa calculation methods, one is deducted by pure math, while the other uses discrete interpolation.

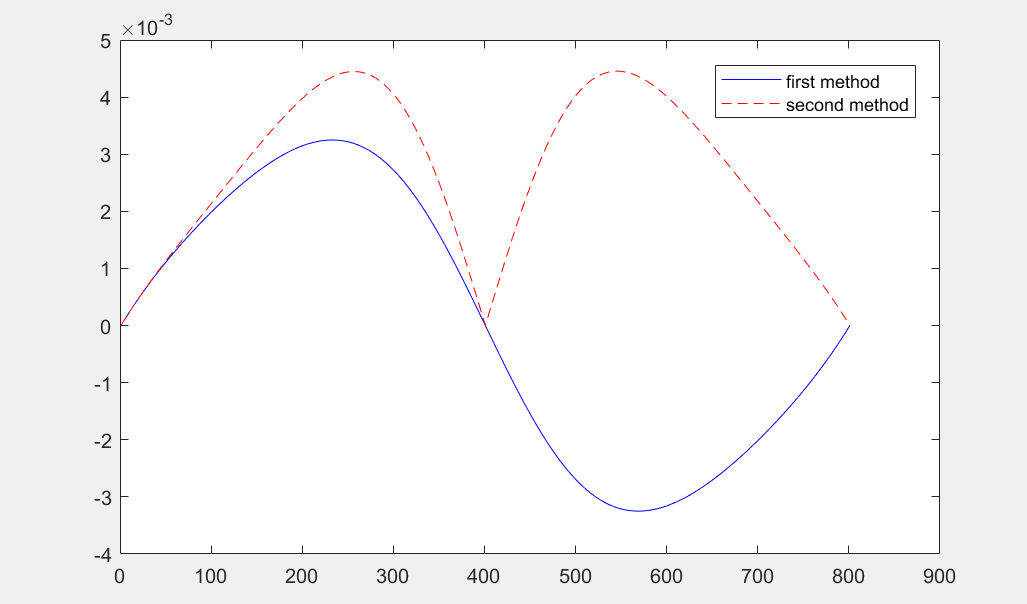
The first one use close form equation as follows:

The other method is as follows:

where

and

In the next few steps of path smoothing, we decided to trust the first method. And the two methods comparison is as follows:



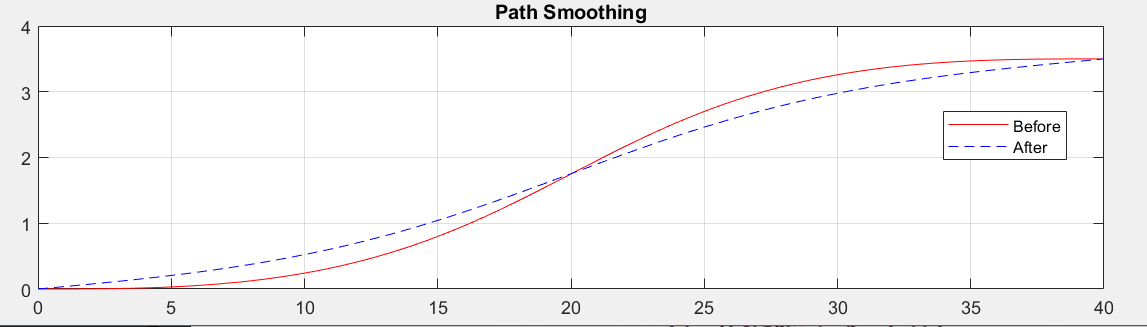
1. Path smoothing method:

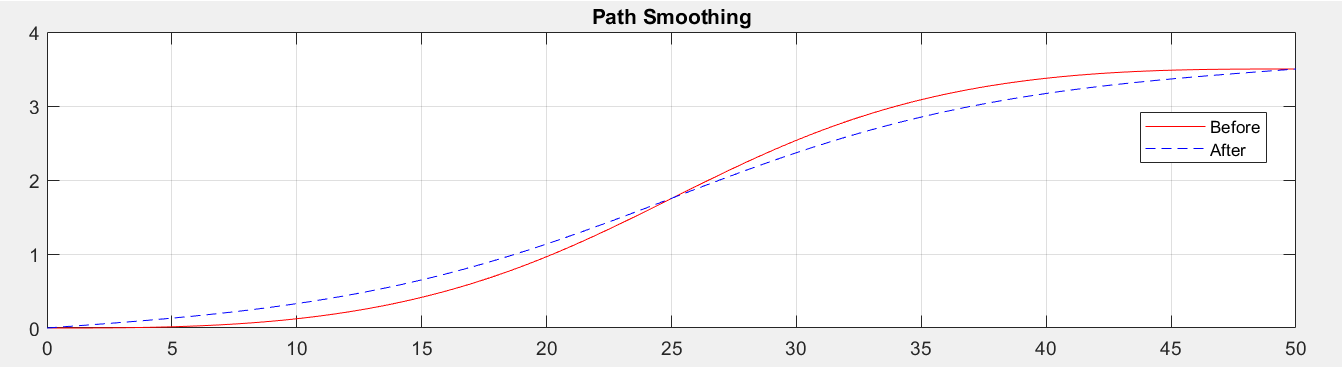
Since the original trajectory generated by Bezier curve cannot guarantee that the curve can satisfy the vehicle kinematic constraints, say, the max curvature constraints. Therefore, we designed a path smoother which aims to minimize 1) the distance between the origin reference line and the optimized trajectory. 2) the smoothness of the optimized trajectory.

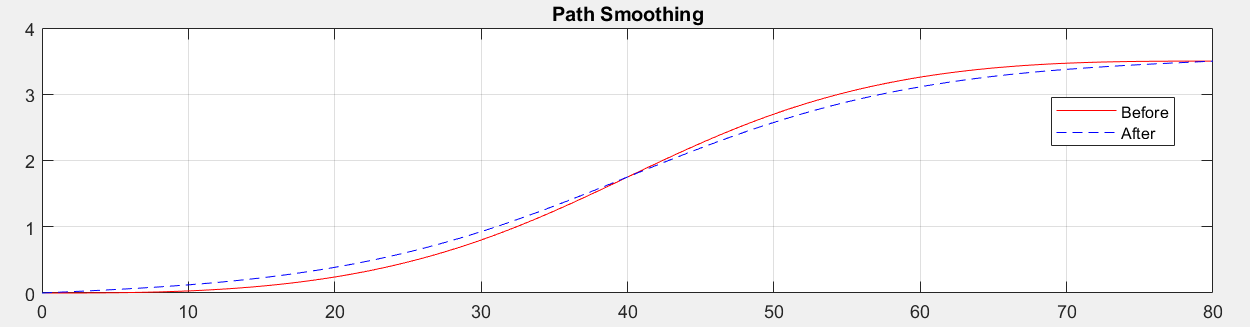
In this smoother, we penalized the norm of the offset between reference line and optimized trajectory, and the norm of . The objective function is as follows:

We use gradient descent method (do derivative over ) to minimize objective function by iterative search.

We also add some stopping criteria to ensure the robustness of the method. The first term is constrained by add a maximum offset between the reference line and optimized trajectory, one the offset increases upper to the upper bound, the iteration will be forced to stop. In terms of curvature, we add a maximum curvature constraint as well. In each iteration, we will check the curvature of the optimized trajectory, once all the points on the trajectory satisfy the curvature limit, iteration will also be stopped. To fix the running time of the iterative search part, we lastly constrained the maximum iteration number as a constant. Several examples of the smoother performance is as follows, where we are smoothing the lane changing reference line, with lateral offset 3.5m, and longitudinal variation between 50m to 80m.







As for the calculation time added by smoothing module, we use a simple example to show the approximate running time. In this example, we smooth an original Bezier curve with lateral offset of 3.5m, and longitudinal distance of 50m, the smoothing parameter alpha is set as 0.01, and beta 0.7, the running time is 62.135 *ms*, and the iteration loop number is 124.

1. Sampling method

In this section, we want to discuss how we use sampling-based method in our design. This section is organized as follows: 1) we discuss difference rules in do sampling-based planning. 2) we divide on-road planner scenario by scenarios and introduce different sampling method in dealing with these scenarios. 3) we introduce a simplified trajectory generation method to help realize Lane Changing Assistant (LCA). 4) we discuss how to make sampling more efficient. 5) we discuss how to choose cost function terms.

1). Difference sampling rules

In sampling-based method, we should always obey the rule that we combine two dimensions at a time. For example, we can sample x(t) and y(t), and combine them together to get spatial-temporal profile with time information. By do derivative over time the velocity profile can also be generated. Another method is to sample y(x), and x(t).

In this section, we discuss our method of candidate trajectory generation method which is based on lattice sampling (kind of), notice that we do not precompute the lattice heap matrix. In our functionals, this is realized and simulated in frenet\_Bezier\_Sampling.m file, and we also provide a simpler functional where a Bezier curve is individually generated once for Lane Change Assistant (LCA) use which is realized and simulated in frenet\_Bezier\_simpleRule.m.

We divide our on-road planner scenario by scenario, more specifically, merging, velocity keeping, following, and stopping.

1. Bezier curve sampling method
2. Quintic polynomial sampling method
   1. lateral trajectory planning
   2. Longitudinal trajectory planning

In this section, we……

Following is a test on the robustness of using quintic polynomial to do longitudinal trajectory planning. The test file named as *test\_on\_long\_poly.m* is the simulation code for testing such polynomial in longitudinal direction. By changing the fixed boundary conditions, we can plot the longitudinal profile of x(t), v(t). combining this two figure together, we can get s(v), and when spatial-temporal profile is given, we can search respective v at specific s in the s(v) graph. As we mentioned above, we do not use such method in dealing with merging tasks, but it works well in other scenarios.

We use the file to do a simple test in test\_on\_laneChanging.slx. The assumption for this test is as follows: we have two adjacent lane vehicles on the right-hand side moving at the same speed as ego vehicle, which is 30 kph. The ego vehicle has the same longitudinal distance as the first neighbor vehicle that is closest to ego vehicle, and the other vehicle is about 50 meters ahead of the first neighbor car. The test scenario is that the ego vehicle is going to merge into the area between the two neighbor vehicles. This test file can generate a speed profile alongside the longitudinal *s*, and it can be used in *test\_on\_laneChanging.slx* file for Carmaker simulation. This test file has not added sampling-based method longitudinal planning functionals and it also does not support such functional simulation. Therefore, it can only test on very specific situation, say, fixed boundary conditions. The merits of this test file are a brief view of how quintic polynomials performs in longitudinal planning and have a look at the changing lane functional performance. Later, when sampling-based functionals are developed, the sampling strategies can also be tested under the similar framework.

The result of this test file is shown in the following figures.





In merging scenarios, the most important thing is velocity control, therefore, a precise velocity tracking controller is critical, it can greatly affect the system performance in merging maneuvers. The following figure is the velocity and steering angle during the merging process. We can see that the ego vehicle velocity increase first and then decrease. Since we only have a simple test with boundary condition fixed, we can not penalize high acceleration during merge, actually, we do not have any control on the duration acceleration. That is why we need to introduce sampling-based method so that we can penalize high acceleration through sampling and cost function minimization.

