Model Predictive Control (MPC)

Compilation

- *Your code should compile:* Code must compile without errors with cmake and make.
    - Confirmed.

Implementation

- *The Model*: Student describes their model in detail. This includes the state, actuators and update equations.
    - Overview: The Model Predictive Control algorithm utilizes a collection of waypoints to fit a discretized trajectory curve that can aid in determining the approximate travel path. The path is comprised of N-states that are optimized to reduce the error between the waypoint trajectory and planned path.
    - State: Each state is a collection of values that indicate the cars location (x/y), orientation (psi), velocity (v), cross-track error (cte), and orientation error (epsi). For a given time horizon, determined by the product of the number of states and elapsed duration between states, each state is approximated to reduce the cost/error from the polynomial fit of the waypoints.
    - Actuators: The actuators apply forces to the car to force a change in state. In the MPC model, both steering angle (delta) and acceleration (a) were used to change orientation and speed of the vehicle.
    - Update: By modeling N different states the update equations must account for the cars kinematics between one time step to the next. In the MPC model, a given time the N-states will be optimized to reduce the error between them and the desired waypoint path. Once the best next time step state (t+1) is known, the required actuations can be determined to meet that state (subject to the physical model constaints, e.g. steering angle)
- *Time step Length and Elapsed Duration (N & dt):* Student discusses the reasoning behind the chosen N (time step length) and dt (elapsed duration between time steps) values. Additionally, the student details the previous values tried.
    - The length and duration parameters were chosen such that the vehicle was able to predict a smooth trajectory with granularity great enough to resolve tighter turns at speed. The time step parameter, N, determines how many state variable points there in the prediction vector. The elapsed duration parameter, dt, is the amount of time between two state prediction point at a given velocity. Multiplying the two parameters yields the duration of the current prediction. The more points there are for a given duration between points, the greater the time horizon and ability to adjust the actuators to position the car in the proper location. As speed increases, more points and shorter durations are necessary to resolve the faster changes in the road.
    - Note that once a trajectory is predicted, the vehicle will perform the necessary actuations, and then re-predict a new trajectory. This means that all of the predicted points are not the actual path of the vehicle, but serve as a method for minimizing the error in predicted trajectories and assigning more accurate actuations.
    - Started with 25 points with an elapsed duration of 0.05 seconds and worked my way down to 10 points and 0.1 seconds. The goal was to reduce the computational complexity as much as possible for a given speed (velocity reference of 30), although, computer could handle more without lag. As speeds were increased, I added more points to help account for the faster time horizon.

- *Polynomial Fitting and MPC Preprocessing*: A polynomial is fitted to waypoints. If the student preprocesses waypoints, the vehicle state, and/or actuators prior to the MPC procedure it is described.
    - Based upon some of the feedback given on the project Slack channel, I found that transforming the global x/y coordinates of the waypoints to the cars local reference frame was a helpful method to determine the polynomial coefficients, determine state, and account for latency. The transformation involved subtracting the cars global position from each of the way points and changing the coordinate system.
    - To fit the polynomial, I passed the converted waypoints to the provided 'polyfit' function, which returned the necessary coefficients. The coefficients could then be used to compute the cross-track and heading error in the MPC optimization.
    - The waypoint polynomial and predicted state values were both assembled in the proper structure and passed to the simulator for visualization.
- *Model Predictive Control with Latency*: The student implements Model Predictive Control that handles a 100-millisecond latency. Student provides details on how they deal with latency.
    - The latency was added by simply adjusting the current state provided by the simulator output to account for the time delay between actuator input and output (i.e., time from turning the steering wheel to the tires turning). Adding velocity, steering angle, and acceleration adjustments to the state accounts for the amount of change that happened during that small latency period (e.g., if the car is traveling at 55 feet/second and there is a 0.1 second delay, the car has actually driven an additional 5.5 feet in that time).

Simulation

- *The vehicle must successfully drive a lap around the track*: No tire may leave the drivable portion of the track surface. The car may not pop up onto ledges or roll over any surfaces that would otherwise be considered unsafe (if humans were in the vehicle). The car can't go over the curb, but, driving on the lines before the curb is ok.
    - Confirmed.