Project 10 Reflection (MPC Controller)

1. Student describes their model in detail. This includes the state, actuators and update equations.

   State:
   In my model, I had 4 state variables. Py and Px represented the current position of the vehicle. Psi represented the going direction of the vehicle. V represented the velocity/speed of the vehicle. I also included 2 more variables to evaluate the cost model. This included CTE which was the shortest distance the vehicle should take to reach its destination. I also used the EPSI variable to represent the difference between current and desired orientation.

   Actuators:
   I used two actuators in my model: Steer_value and Throttle. Steer_value controlled the steering of the front wheels, thus controlled the orientation of the vehicle. Throttle controlled the acceleration of the car.

   Update Equations:
   I referenced the class material to come up with the update equations in my code. Fairly straightforward.

2. Student discusses the reasoning behind the chosen N (timestep length) and dt (elapsed duration between timesteps) values. Additionally the student details the previous values tried.

   N - At first I chose an N of 35, which resulted in the car being more willing to go off the tracks and significantly longer compute time. I moved my N value down to 25, but the compute time was still too long. After more trial and error, I settled on 10, which provided a good elapsed time value for the car to reevaluate its path and stay on the road.

   Dt - Finding dt also presented challenges. I didn't want it to be too small, or it would have been computationally difficult because more steps need to run. Alternatively, I also didn't want the value to be too big. I chose a value of .1 after trial and error as well.

3. 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.

   With each new timestep, the waypoints were received and preprocessed into coordinates after the update step. A polynomial was then fitted to those points. The coefficients were then passed to the MPC to calculate CTE and EPSI.

4. The student implements Model Predictive Control that handles a 100 millisecond latency. Student provides details on how they deal with latency.

   To handle the 100 millisecond latency, I updated the state before using it in the model. Reference main.cpp to see the implementation of this code.