MPC Questions

Model Description:

The purpose of this model is to implement an MPC controller, which is a dynamic control model. This allows for elements such as forces and velocity to be taken into account.

Code:

After defining variables, the model first uses the standard physics kinematic equations to determine the change in state during the delay time of 100 ms (Main.cpp lines 100 – 104). Simple kinematic equations may be used with the average velocity, angle and time to estimate how far the vehicle will move during 0.1 second timeframe.

Position = initial position + velocity * delay.

Angle = initial angle – steer value/Lf * delay.

Velocity = Velocity + acceleration * delay.

For this model, I next changed the frame of reference to the vehicle's frame of reference in order to simplify calculations (Main.cpp lines 114-115).

The next step is to fit the model to a curve. This can be directly seen in the simulator. Polyfit and Polyevel was used to fit the coefficients to the model and determine the crosstrack error (Main.cpp lines 126-129).

Next, the state vector (x) is set. Because the model uses the vehicles frame of reference, px, py and psi are all assumed to be 0.

Next, the state and coefficients are sent to mpc.cpp to determine the variables.

8 timestep lengths of .15 ms duration were chosen for the model through trail and error.

The model from the classroom was used to determine the cost equations (mpc.cpp lines 62-76). However, the weights needed to be tuned. I used a high weight (2300) for the state cost weights. This is essential for assuring a safe speed and turn radius. A weight of 4 was chosen to minimize the use of the actuators. To minimize the value gap, I used values of 200 and 8.

In lines 88 -135 of mpc.cpp, the initial constraints are set using the equations from the class. The equations for f0 and psides0 had to be modified to suite a second order curve.

The output in lines 264-270 is stored in a vector based on the number of iterations chosen.

Timestep Length and Duration:

These were determined mostly through trial and error. I began with an N of 10, moved to 12, then settled at 8. A bigger N requires more computing power. A smaller dt requires more computing power. I began with a dt of 01., then settled at .13.

Waypoints

This procedure is established in the top section.