Model Predictive Control

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

The state is constructed as an Eigen VectorXd matrix of 1 row and 6 columns. This includes the x and y coordinates of the car position, the angle of the car's heading, the velocity of the car, the cross track error of the car from the predetermined waypoints, which is along the center of the track, and the error of the car's heading direction.
The actuators are the car's steering values and its throttle value, with positive values being acceleration and negative values being breaking.
The model equations are as follows:
x_[t+1] = x[t] + v[t] * cos(psi[t]) * dt
y_[t+1] = y[t] + v[t] * sin(psi[t]) * dt
psi_[t+1] = psi[t] + v[t] / Lf * delta[t] * dt
v_[t+1] = v[t] + a[t] * dt
cte[t+1] = f(x[t]) - y[t] + v[t] * sin(epsi[t]) * dt
epsi[t+1] = psi[t] - psides[t] + v[t] * delta[t] / Lf * dt

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.

For tuning the number of waypoints and the length between way I went with the manual twittle method and tried different numbers. For N I've tried 100, 50, 25, 12, 10, 9, and 8. For dt I've tried 0.5, 0.2, 0.1337, and 0.125. I've set my parameters based on my observations of how well the car does on the track.

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

To handle a 100 millisecond latency I've used the average of multiple steering and throttle values in order to smooth out the actuator controls.