Model Predictive Control

The Model

The Model used in this project is the Kinematic Model of the vehicle, which ignores things like gravity, tire forces, and mass. Using this model instead of a Dynamic model reduces the accuracy, but is a simplification that is a good approximation to the actual vehicle dynamics at low and moderate speeds.

The state is the x position, y position, the heading of the vehicle, the velocity, and both the cross track error and the heading error. A vehicle has 3 actuators (steering wheel, brake pad, throttle), but they can be represented as two control inputs. The update equations require the state at the current time step as well as the control inputs at the current time step.

```
// 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
```

Time step length and Elapsed Duration (N & dt)

The N and dt values I choose are 10 and 0.1 respectively. These values were chosen from having the understanding that the prediction horizon, N*dt, should be in the range of a few seconds due to the predicting too far into the future won't make sense due a changing environment. I also understood that dt, being the time that elapses between actuations, needed to be taken into consideration when looking at the actuators latency. I also wanted to have a low dt in order to better optimize the actuation values used to keep the vehicle on the reference trajectory. I tried values of N from 5 to 30, which resulted in a variety of horizons when optimizing actuator values. I felt that around 10 time steps led to great results, while also not being too long into the future.

Polynomial Fitting and MPC Preprocessing

Prior to the MPC, the waypoints retrieved from the simulator are in the Maps Coordinate system, which needs to be transformed into the Vehicles coordinate system in order to easier calculate the Cross Track Error and the heading error for the MPC.

```
for (int i = 0; i < ptsx.size(); i++){

  double dx = ptsx[i] - px;
  double dy = ptsy[i] - py;

  vehicle_coordinates_x(i) = (dx*cos(psi) + dy*sin(psi) );
  vehicle_coordinates_y(i) = (-dx*sin(psi) + dy*cos(psi) );
}
```

Model Predictive Control with Latency

I handled the 100ms of actuator latency by setting my dt to 100ms, and considering the previous time steps actuations when predicting the next time steps velocity, heading, cross track error, and heading error. This way there is no latency between the current actuation and the value considered in the update equations.