PROJECT SPECIFICATION

Model Predictive Control (MPC)

By Satinderjit Singh

Compilation

Your code should compile.

Code must compile without errors with `cmake` and `make`.
Given that we've made CMakeLists.txt as general as possible, it's recommend that you do not change it unless you can guarantee that your changes will still compile on any platform.

Code compilers with regular cmake and make.
The src.zip has all the *.cpp and *.h files

Implementation

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

Model Predictive Control is simulating actuator inputs to predict trajectory and select trajectory with minimum cost

The MPC model consists of equations which update the state of the vehicle from current tilmestep “t” to next timestep “t+1”

1/ The state consists of

X = x location
Y = y location
Psi = orientation
V = Velocity
CTE = cross track error
Epsi = Error in orientation

2/ Constraints are defined for actuations.

e.g.
Steering constraints = -25deg to + 25deg
And
Throttle = - 1 to +1

3/ The cost function
We want to minimize cost function
Cost can consist of CTE, EPSI, Reference velocity, Distance between starting and ending point etc.

The current state is passed to the Model and then optimizer comes with set of actuations which will minimize the cost function.

MPC model
Timestep Length and Elapsed Duration (N & 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.

N = 20; dt = 0.5 same as in lesson
The car was braking a lot and then finally stopped at it reached side of the road.

N = 20; dt = 0.1; reduced dt so we calculate more often
The car reached max speed of 66MPH on the track

N = 10; dt = 0.1; reduced N so we calculate less points
The car reached max speed of 63 MPH on the track

N = 10; dt = 0.2; increased dt so we calculate less often
The car reached max speed of 65 MPH on the track

Final value chosen

N = 10
dt = 0.2

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.

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 state is predicted for latency and then passed to the solver.

Following code updates the state from current state to latency

```
// Predict state after latency before passing to the solver
double dt = 0.1;
px = v * dt;
psi = -v * steer_angle * dt / 2.67;

double cte = polyeval(coeffs, px);
double epsi = -atan(coeffs[1] + 2 * px * coeffs[2] + 3 * px * px *
coeffs[2]);

// State to initialize the solver
Eigen::VectorXd state(6);
state << px, 0.0, psi, v, cte, epsi;
```

Following code sends the latency state to the solver

```
auto vars = mpc.Solve(state, coeffs);
```

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.

The code provided drives successfully around the lap with max speed of 65MPH.