Writeup

MPC Project

Rubric Points

Compilation

1. Your code should compile.

Yes, my model can compile w/o errors with `cmake ..` and `make`

Implementation

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

Generally there are 3 parts involved in my MPC algorithm, cost function, constraints, and design variables. in the end, latency is considered to simulate the real physical world.

The design variables are the states vectors(etc,x,y,theta,v and so on) and control variables(like steering angle and throttle).

The optimizer handles the cost function and constraints, constrains is another saying of motion model. Ipopt accepts constrains in this transformed format(i.e. equality). It expects functions that evaluate to 0 when the constraints fulfilled.

In all, the optimizer is going to find the minimum value of cost function by varying the design variables under the specified constraints.

2. 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.

As introduced in the class, the prediction horizon *T* should be a few seconds, at most. Beyond that horizon, or else the adjacent environment will be changed that it won't make sense to predict any further into the future path.

After building my model successfully, I tried to turn those parameters. Purposely, I set the dt = 0.5s, and N = 20; there are a lot of lags to the respond, by car can hardly speed up and even stuck at specific point in the track,

if I put the dt at 0.01s,time lapse to the adjacent control signals is so small that It responses wildly and my car quickly went off the track. it can not make a longer path driving as it only forecast meters instead of the terrain info inside the horizon.

Finally I got the balance, set dt around 0.15.

For N, if > 20, then the forecasted route will go further and even beyond the visible range, which contribute less to the driving performance and waste loads on my computation powers. So I set it around 10, which seems reasonable.

In general,

1. smaller dt is better. (finer resolution)
2. larger N isn't always better (computational time)
3. N * dt at second level. So to balance both scale.

3. 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.

Waypoints are preprocessed as I transform the global coordinate into the local coordinates first, then feed it into the `polyfit()` function.

Prior going into the MPC procedure, I also introduced the latency module. A control signal delay is put into the motion model. So this pre-process state vector is prepared for the MPC to optimize.

4. 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.

As mentioned above, this latency was deemed as part of the pre-processing of state vector, which work flow demonstrates here:

1. Calculate cte, epsi
2. insert latency to transform state by using of motion models
3. Feed the transformed state to MPC.cpp

acceleration and steering angle will be influenced by this delayed-control in reality. So latency time 0.1s is defined. To replace dt in motion model with this latency value.

When transform the coordinates to the car coordinate system, x,y,psi becomes zero.

```
double x = 0;
double y = 0;
double npsi = 0;
```

change of sign because turning left is negative sign in simulator but positive yaw for MPC:

```
delta *= -1;
```

then implement the latency to the motion model.

```
x += v * cos(npsi) * latency;
y += v * sin(npsi) * latency; //sin(npsi)=sin(0)=0
npsi += v * delta * latency / Lf;
v += a * latency;
cte += v * sin(epsi) * latency;
epsi += v * delta * latency / Lf;
```

that’s all for my latency details.

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.

I see a positive results in this case.

Discussion

I encountered an issue of making a proper communication to visualize the path

And finally I got support form mentors in discussion forum.

my latency codes was also improved a bit with helpful hints.

This is the link of my encountered issues and solutions:

https://discussions.udacity.com/t/no-trajectory-plotted-in-simulator/373986/5