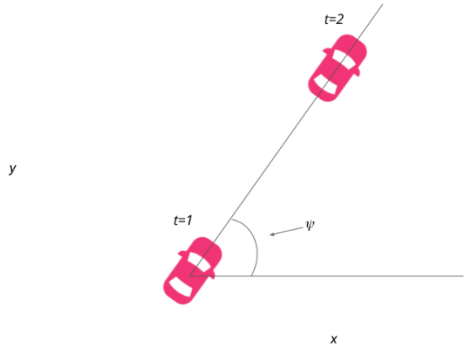
Model Predictive Control (MPC)

Kwanghyun JUNG / 16th JUL 2017

Introduction

In this project you'll implement Model Predictive Control to drive the car around the track. This time however you're not given the cross track error, you'll have to calculate that yourself! Additionally, there's a 100 millisecond latency between actuations commands on top of the connection latency.

The Model



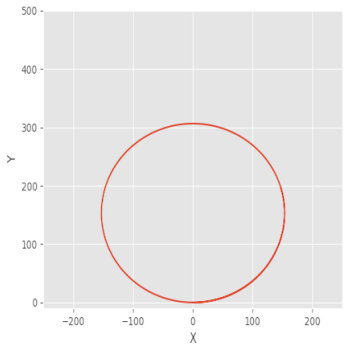
Model

$$x_{t+1} = x_t + v_t * cos(\psi_t) * dt$$
$$y_{t+1} = y_t + v_t * sin(\psi_t) * dt$$
$$\psi_{t+1} = \psi_t + \frac{v_t}{L_f} * \delta_t * dt$$
$$v_{t+1} = v_t + a_t * dt$$
$$cte_{t+1} = f(x_t) - y_t + v_t * sin(e\psi_t) * dt$$
$$e\psi_{t+1} = \psi_t - \psi des_t + \frac{v_t}{L_f} * \delta_t * dt$$

In tihs project I use Kinematic Model which is suggested in the lecture.
Postion x,y angle psy, Velocity v
Cross Track Error cte, Orientation Error epsy

Choosing Lf

L_f measures the distance between the front of the vehicle and its center of gravity. The larger the vehicle, the slower the turn rate.



In the above image the vehicle started at the origin, oriented at 0 degrees and was then simulated driving with a δ value of 1 degree and L_f value of 2.67.

Timestep Length and Elapsed Duration (N & dt)

N(timestep length) , dt (elapsed duration between timesteps)

I choose N=10, dt=0.1 (100ms)

Because Udacity Simulator gives values every 100ms, and the latency is also 100ms. So if I want to process latency, then dt = 0.1 is best choice. Because I process latency not check the time, only check the looping count.

And I decrease N into 5, 6.. but the curve is so ugly to process, and Increase the N until 20, the processing time is much needs, so car go out of the way, even backward racing.

Polynomial Fitting and MPC Preprocessing

Polynomial Fitting is processed by “polyfit()” function in main.cpp. First, call the polyfit function, then returns coefficients. Then put the coefficients values into “polyeval()” function, then returns cte value, and can gt -atan(coeffs[1]) then get epsi value, too.

```
 // Pass the x and y waypoint coordinates along the order of the polynomial.
// In this case, 3.
auto coeffs = polyfit(waypoints_xvals, waypoints_yvals, 3);


//    We can evaluate the polynomial at a x coordinate by calling `polyeval`.
//    in car coord , px = 0, py = 0,    psi = 0
double cte = polyeval(coeffs, 0);
double epsi = -atan(coeffs[1]);
```

Cost function

Cost function is calculated in the 3 parts(reference state , Minimize the use of actuators, Minimize the value gap between sequential actuations.). I think it is similar ideas from PID controller. Of course, this cte is more complicated and can give more control factors. In MPC.cpp from 51 lines to 59 lines is that parts. The multiple values are get from "trial & error". This value is so good if the max speed (ref_v) is 55 (mph). If you want fast speed, then more value's can be added or multiplied.

MPC Preprocessing

The waypoints are changed to vehicle's perspective before the processing. (Main.cpp line 107-115). After transformed, origin px=0, py=0, psi=0. So It is very simple to process to next step.

ψ Updates

In the classroom we've referred to the ψ update equation as:

$$\psi t+1=\psi t+Lfvt*\delta t*dt$$

Note if δ is positive we rotate counter-clockwise, or turn left. In the simulator however, a positive value implies a right turn and a negative value implies a left turn.
So I changed

$$\psi t+1=\psi t-Lfvt*\delta t*dt$$

MPC.cpp line 146

```
fg[1 + psi_start + t] = psi1 - (psi0 - v0 * delta0 / Lf * dt);
```

Model Predictive Control with Latency

In this project, the latency is 100ms which is given. And as I said the selection reason for dt=0.1(100ms), every step, the control part , i.e Wheel angle and Speed. So I change the (N+1)'s value to Nth value except first value. You can see that in my source file, MPC.cpp

```
AD<double> delta0 = vars[delta_start + t - 1 - latency];

AD<double> a0 = vars[a_start + t - 1 - latency];
```

The value of latency is 0 when the t =0, but it is changed to 1 when t>0. So, I always give 100ms fast control values.

Notes.

You can see my simulator video at https://youtu.be/VHd6Mn10q2g

Almost source of MPC.cpp is adopted from Udactiy quiz solution (https://github.com/udacity/CarND-MPC-Quizzes/blob/master/mpc_to_line/solution/MPC.cpp)