Reflection

PID controller project

Kwanghyun JUNG

21 JUN 2017

P- Proportinal

Steer in proportion to Cross Track Error(CTE). If CTE is increased, then P factor will be increased. Vise versa. In my source code, P controller's variables are two. p_error and p[KP] (KP=0). p_error eaquals to cte, which is given by simulator at every 0.1 second.

p_error = cte

so P controller is implemented like this

p[KP] * i_error

p[KP] is called tau in the instruction)

This is main factor of PID controller but if only use this factor then overshoot will be occurred.

D-Differential

If use P controller alone, overshoot problem occurred. It cannot affect very fast change of out side factors include CTE. If I change the wheel from very big CTE with only P controller then it always overshoot, because it is too big P control value because it doesn't regard the change of CTE. If CTE is decrease very fast, then the value of controller must be decreased compare to previous time. This is D-Controller.

I always count the difference of previous CTE and present CTE. The variable p_error always save previous CTE value before update this time. So d_error is calculated with ease.

d_error = cte - p_error

p[KD] * d_error

I- Integral

Because if CTE goes near 0, the value of P controller move to tiny value that cannot affect to steer(control). So offset is always remains. So we need use I controller which is Integral. It sums all CTE from starting until now. (Sometimes it can overflow, but this project I ignore that case.)

i_error += cte

p[KI]* i_error

PID controller

So my PID controller is implemented on PiD::TotalError()

```
double PID::TotalError() {

   return ( -p[KP]* p_error - p[KD] * d_error- p[KI] * i_error );

}
```

I used ‘ double p[3]’ array for my Twiddle routine instead Kp Kd, Ki.

TWIDDLE

Now I must find the best values for p (Kp, Kd, Ki).

I use TWIDDLE to get best coefficients of PID controller. First of all I must change algorithm wich is given by online lecture.

[Udcity online lecture TWIDDLE algorithm]

```
def twiddle(tol=0.2):
    p = [0, 0, 0]
    dp = [1, 1, 1]
    robot = make_robot()
    x_trajectory, y_trajectory, best_err = run(robot, p)

    it = 0
    while sum(dp) > tol:
        print("Iteration {}, best error = {}".format(it, best_err))
        for i in range(len(p)):
            p[i] += dp[i]
            robot = make_robot()
            x_trajectory, y_trajectory, err = run(robot, p)

            if err < best_err:
                best_err = err
                dp[i] *= 1.1
```

```
            else:
                p[i] -= 2 * dp[i]
                robot = make_robot()
                x_trajectory, y_trajectory, err = run(robot, p)

                if err < best_err:
                    best_err = err
                    dp[i] *= 1.1
                else:
                    p[i] += dp[i]
                    dp[i] *= 0.9
        it += 1
    return p
```

There is big problem to change this routine to my project. Because above routine can make robot(in this project, I can call it 'car') and run it in the routine. But in this project we cannot make car and run like above. Because it is asynchronous process start car and receive the signals(like CTE). So only can process when the signal(include CTE). So I change above routine with 'depth' concept.

Every time it received signal from car, I can call TWIDDLE routine. It saves all p values and dp values. Also depth value. If you see the routine, there are two depth of if…else clause. The first is depth 1, and the other, i.e below first else: clause is depth 2.

So, I can change the twiddle routine, you can see PID::Twiddle() in PID.cpp file.

But run twiddle needs so much time. I test from p=[0,0,0] , dp=[1,1,1] but it is so slow to get the answer. Even I must determine how to repeat same track on my twddle routine. I found some codes from Udacity community which is can reset the car server (simulator)

```
json msgJson;

msgJson["steering_angle"] = steer_value;

msgJson["throttle"] = 0.3;

auto msg = "42[\"steer\"," + msgJson.dump() + "]";ws.send(msg.data(), msg.length(), uWS::OpCode::TEXT);
```

I use total_count variable which is increased 1 every UpdateError routine, so I can control when it start twiddle and when ends. I omit some data from staring with TWIDDLE_MIN_COUNT

```
if(total_count++ > TWIDDLE_MIN_COUNT )

   total_err += (cte*cte);
```

total_err will be divided by real count (total_count-TWIDDLE_MIN_COUNT), so I get err value. This two values are define in PID.h

```
#define TWIDDLE_MAX_COUNT 100

#define TWIDDLE_MIN_COUNT 20
```

If total_count is reached TWIDDLE_MAX_COUNT then twiddle goes next round. It is repeated, repeated until user stop.

Starts from very short course, increase the time(max counts) until almost 1 lap. I used every previous best results as starting values.

And about starting values of twiddle, i.e. p[], dp[], I assumed I got all values ABS are smaller than 1. (Because the result of PID controller is [-1…1]) And integral value, p[KI] must very small, because it is more and more increased when it comes to 0 CTE, so it cannot bigger then p[KP] value or p[KD] value. p[KD] is not divided by (0.1sec). So It must be bigger than p[KP] about 10 times.

So I use start values like below.

```
p[KP] = .1;
p[KI] = .01;
p[KD] = .0;
dp[KP] = .1;
dp[KI] = 0.01;
dp[KD] = 1.;
#define TWIDDLE_MAX_COUNT 100
#define TWIDDLE_MIN_COUNT 20
```

And from next round, I use the best values which is get from previous round.

```
// 2nd -- the best vaules of 1st try (MAX 300 / MIN 100)
//0.487765/0.00518348/1.8019  dp : 0.0630247 0.00630247 0.421901\

// 3rd -- the best values of 2nd try
(MAX 550 / MIN 250)
//  p 0.545736/0.00449651/2.56212  dp : 0.0555935 0.00555935 0.372155

// 4th -- the best values of 3rd try
(MAX 850 / MIN 50)

// p 0.496042/0.00899958/2.93427  dp : 0.0441346 0.00361101 0.241729
// 5th -- the best values of 4th try
(MAX 1200/ MIN50)
//  p 0.540177/0.00574606/2.75805  dp : 0.0353915 0.00432563 0.17622
```

So the my last values is 0.540177/0.00574606/2.75805

You can see detail data twiddle_log.txt file

The result

You can see my project video

https://youtu.be/QaEXI57DZ-E