PID Controller

Tuning Kp, Kd and Ki parameters

`Kp` is the proportional gain parameter. This governs how hard the vehicle is steered toward the reference trajectory to compensate for cross track error. However, proportional only controller (specially a one with high gain) will cause oscillations to occur. I have saved a video of such proportional only controller here. (`Kp = 0.32`)

To dampen the oscillations, we need to add a derivative term `Kd`for the controller. However a small `Kd` gain will still make the controller under damped and it might eventually could just crash on a hard turn as you can see in this example of underdamped PD controller. (`Kp = 0.32, Kd = 2.0`)

However, setting a very large derivative gain will make the controller over damped and steering commands will be very jerky as controller will not like any deviation of the cte. In this example video of overdamped PD controller I have set a very large derivative gain for the same controller in first example of proportional only controller. (`Kp = 0.32, Kd= 100.0`)

A critically damped PID controller could be obtained by setting `Kd` parameter relatively in just right proportion to the `Kp` term. For this purpose, I have used twiddle algorithm iteratively over increasing throttle values. (First, I would set smaller throttle value to get rough estimation and then after establishing relatively good parameters, I would set these as starting values for twiddle again and tune for the new throttle value. If it fails to converge, I would try simply reducing or increasing both parameters about 50% and try again.)

One observation made during this exercise was that system has a very little bias if any and works just fine even without an integral part of the PID controller. So, I set a very low value (0.0001) for the `Ki` term for the completeness of controller.

Here is a video of final tuned version of the controller reaching for speeds over 94mph.

Final tuned parameter values. `Kp=0.12, Kd=3.07, Ki=0.0001`

Controlling Throttle Value

For the speed control I did not employed any complex controller. Just a simple rule to set a target throttle value of either 1.0(full throttle) or -1.0(full brake) and a simple scheme to either increment or decrement the throttle value sequentially over couple of steps. (Given the rate of throttle change it could still feel quite a jerky if you were inside the vehicle. But still it feels quite sporty on at least in the simulator.)

```
double target_throttle = 1.0;
if (speed > 50 && abs(cte) > 0.4 && abs(steer_value) > 0.3) {
  target_throttle = -1.0;
}
if (target_throttle > throttle) {
  throttle += 0.2;
}
else if(target_throttle < throttle) {
  throttle -= 0.2;
}
```

Special Observations

- I noticed that in my PC when using PID project running under Bash on Windows, it received telemetry readings on irregular intervals. This made the controller tuning much hard. So, I opted to compile PID project natively on Windows using Visual Studio project and this produced a much stable result.
- Similarly, when same project was implemented using python it quite failed to achieve considerable results even though it appeared that telemetry data stream was receiving in regular fashion.
- PID tunings will be much sensitive to the performance of the PC. For example, somewhat marginally stable controller would have slightly less desirable outcome when heavy processing load is running (e.g. When trying to screen record the output of simulator.)

Tuning Log

2018-12-25

- Initial_Params: `Kp=0.25, Kd=0.75, Ki=0, dp=[0.1, 0., 0.5], n=2000 samples, throttle = 0.3`
- Tuned Result: `Kp=0.61 Ki=0 Kd=5.49`
- Initial_Params: `Kp=0.61, Ki=0, Kd=5.49, dp=[0.2,0.,1.5], n=2000 samples, throttle = 0.5`
- Tuned Result: `Kp=0.32, Ki=0, Kd=6.84`

2018-12-26

- Initial_Params: `Kp=0.32, Ki=0 Kd: 6.84 Ki=0, dp=[0.1,0.,1.5] ,n=2000 samples, throttle =0.8 Kp=0.32`

  At this stage, it failed to converge even with using brakes=-0.5 when speed > 50 and absolutte cte > 0.8 and steer angle > 7.5 degrees. So I retried with smaller parameters since it was quiet oscillating.

- Initial_Params: `Kp=0.16, Ki=0, Kd=3.4, dp=[0.05, 0., 0.8],n=2000 samples, throttle=0.8`

  Still it didn't help to converge quickly either. However, result was much visually pleasing so I decided to stick to these values and try to push it to full throttle. This resulted in `Kp=0.27, Kd=5.08` but the result wasn't visually pleasing.

- So, I decided to try out searching from `Kp=0.13, Kd=2.5`. This produced `Kp=0.19, Kd=3.07` But during these runs it seemed `Kp=0.12, Kd=3.07` was producing much more visually pleasing behavior. So, I finally opted to use that.