PID Controller

PID Controller Project

The goals / steps of this project are the following:

a) Drive a self driving car on lake side track (using udacity simulator)
b) Implementation of PID crontroller
c) Implementation of Twiddle for hyper parameter tuning (P, I, D & constants)
d) Simulation of car running on lake side track and system behaviour debug

Image References:

Image-1 : A block diagram of PID contoller in feedback loop (generic)
Image- 2: Mathematical equation of PID contoller in feedback loop (generic)
Image- 3: PD contoller outcome & concern (python)
Image-4 : PID contoller outcome & resolution (python)
Image-5 : Simulator (udacity) for autonomous drive on a track
Image-6 : drive start with slow speed and high angle – PID behaviour
Image-7 : Vehicle drive on smooth track
Image-8 : Vehicle drive on sharp turn

➢ Files Submitted & Code Quality

Submission includes all required files for build and compile:

1. src : C++ source code for PID implementation i.e PID.cpp, PID.h, main.cpp, Twiddle.cpp and Twiddle.h
2. build : project compilation script and executable
3. Simulator Interface : JSON.hpp file in src provides a mechanism to exchange messages on a port
4. Scripts : install-ubuntu.sh, CmakeLists.txt to create WebInterface setup and build project
5. Output : Video's (*.mp4)
   - PID-Controller4.mp4 gives simulation outcome (without twidle)
   - PIDcontroller-Twiddle.mp4 (with twiddle)

➢ Submission includes functional code & other files

Sumitted project contain functional code, recorded video and outcome images of project execution.

➢ Submission code is usable and readable

YES

(A) PID Overview

PID stands for proportionate-integrate-derivative controller. It is a control loop feedback mechanism widely used in industrial control and autonomous systems. A PID controller continuously calculates an *error value* as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted *P*, *I*, and *D* respectively) which give the controller its name.

An everyday example of PID is cruise control on road vehicle. PID algorithm restores the actual speed to the desired speed in a optimal way without delay or overshoots by controlling power outcome of the vehicle engine. Project PID controller implementation is also targeted for similar objectives.

(1) PID Block diagram

Today there is universal use of the PID concept in applications requiring accurate and optimised automatic control.

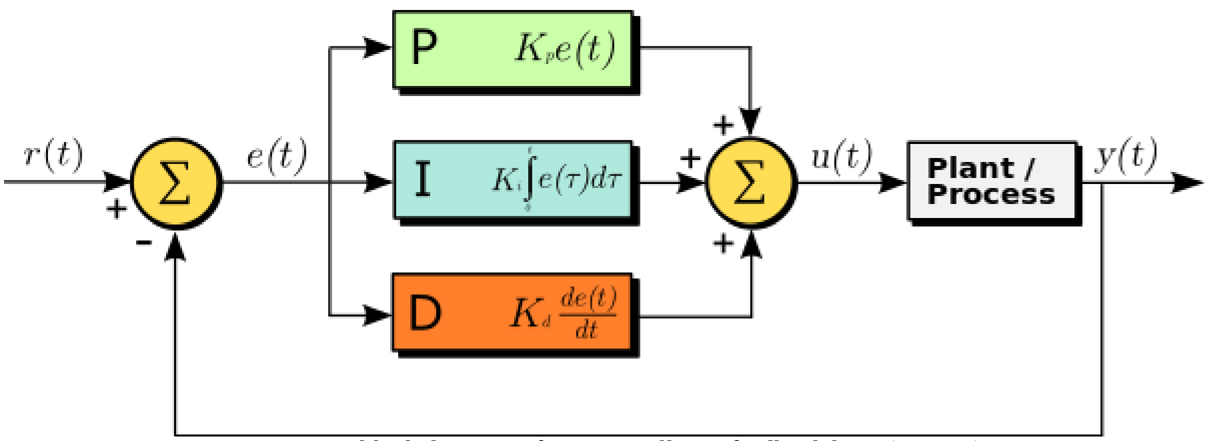


Image-1 : A block diagram of PID contoller in feedback loop (generic)

The block diagram shows the principles of how these terms (P-Proportion, I-Integral & D-Derivative) are generated and applied. It shows a PID controller, which continuously calculates an error value as the difference between a desired setpoint and a measured process variable , and applies a correction based on proportional, integral, and derivative terms.

(2) PID Mathematical Form

The overall control function can be expressed mathematically as

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(t')\,dt' + K_{\mathrm{d}} \frac{de(t)}{dt},$$

Image- 2: Mathematical equation of PID contoller in feedback loop (generic)

where , , and , all non-negative, denote the coefficients for the proportional, integral, and derivative terms respectively (sometimes denoted *P*, *I*, and *D*).

(3) PID Algorithm

PD section implementation as we have learned in classroom sessions depicts steps for calculating gains.

/* Pseudo section for PD controller** *************

```
def run(robot, tau_p, tau_d, n=100, speed=1.0):
    x_trajectory = []
    y_trajectory = []
    prev_cte = robot.y
    for i in range(n):
        cte = robot.y
        diff_cte = cte - prev_cte
        prev_cte = cte
        steer = -tau_p * cte - tau_d * diff_cte
        robot.move(steer, speed)
        x_trajectory.append(robot.x)
        y_trajectory.append(robot.y)
    return x_trajectory, y_trajectory
```

This is very similar to the P controller. We've added the `prev_cte` variable which is assigned to the previous CTE and `diff_cte`, the difference between the current CTE and previous CTE. We then put it all together with the new `tau_d` parameter to calculate the new steering value, `-tau_p * cte - tau_d * diff_cte`.

This implementation helps in achieveing quick stability as compared to alone D implementation (as learn in class), as depicted below; However introduction of D term is not enough to manage high CTE in some situations. Thats why P & D is not enough for smooth steering and need to introduce additional bias.

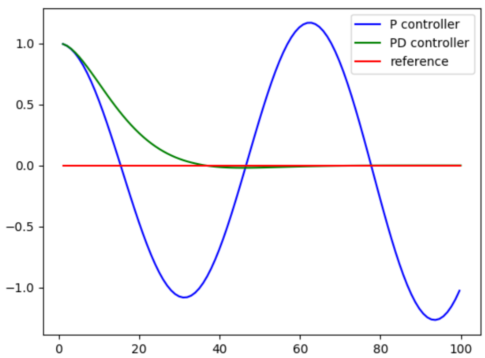


Image- 3: PD contoller outcome & concern (python)

*************/

Introduction of I in PD controller & Results;

As learned in the class session we implemented PID controller (in python) by using additional integral bias for error component summations.

/************* pesudo code for Integral part implementation**

```
def run(robot, tau_p, tau_d, tau_i, n=100, speed=1.0):
    x_trajectory = []
    y_trajectory = []

    # TODO: your code here
    prev_cte = robot.y
    sum_cte = 0
    for i in range(n):
        cte = robot.y
        sum_cte = sum_cte+cte
        diff_cte = cte - prev_cte
        prev_cte = cte
        steer = -tau_p * cte - tau_d * diff_cte - tau_i*sum_cte
        robot.move(steer, speed)
        x_trajectory.append(robot.x)
        y_trajectory.append(robot.y)

    return x_trajectory, y_trajectory

x_trajectory, y_trajectory = run(robot, 0.2, 3.0, 0.004)
n = len(x_trajectory)

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8,8))
ax1.plot(x_trajectory, y_trajectory, 'g', label='PID controller')
ax1.plot(x_trajectory, np.zeros(n), 'r', label='reference')
```

This implementation outcome bring faster stability in abrupt change.

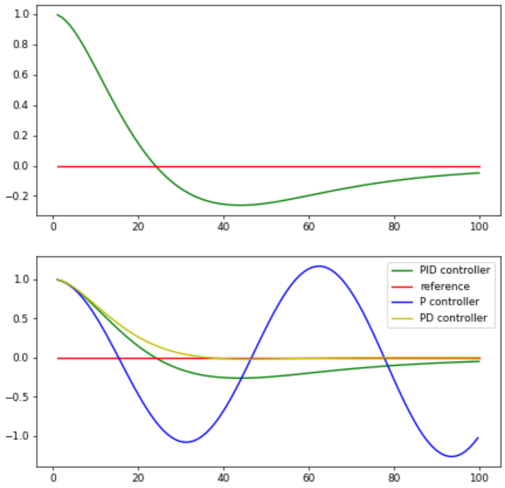


Image- 4: PID contoller outcome & resolution (python)

***/

(4) PID C++ Implementation

main.cpp – provides a standard main() function which recieves simulator message on JSON interface and process vehicle motion parameters with PID parameter setting aling with updates of cummulative errors.
PID.h – defines class PID and its destructor/constructors
PID.cpp – defines PID class & various functions of PID class i.e INIT(), UpdateError() and TotalError()
Twiddle.h – define Twiddle class and linked functions i.e init(), Restart(), Next(), UpdateError and Print parameter functions.
Twiddle.cpp – maintains 9 parameters for tunning i.e P[], dp[] and Kp[]

```
/************************ Twiddle P parameters ***************
 this->cte_limit = 4.5;

 this->p = new double[n_param];
 // Kp for PID steering
 this->p[0] = 0.03685; /* 0.04 */
 // Ki for PID steering
 this->p[1] = 0.0068858;  /* 0.00777101  */
 // Kd for PID steering
 this->p[2] = 1.4; /* 1.4 */
 // Min Target Speed
 this->p[3] = 60;    /* Incresed 57.4 */
 // Speed over cte/steering coefficient
 this->p[4] = 9.511;
 // Kp for PID speed
 this->p[5] = 0.255;   /* 0.2665  */
 // Ki for PID speed
 this->p[6] = 0.00016;  /* 0.00016 */
 // Kd for PID speed
 this->p[7] = 0.00245; /* 0.0026 */
 // cte/steering limit for max speed
  this->p[8] = 1.1255; /* 1.1355 */
********************************** similarly other parameter's defined *********/
```

- Restart() - function reset all parameters (0 value) i.e for proportionate values for steering and speed
- Next() - calculates average total squared errors and squared errors (steering and speed), also find out best erros scenario's
- UpdateError() - function used to update errors

<u>Parameters Tunning:</u>

Too much increase in k (steer, speed) values leads to drive oscillations while N is low (nearly 100) however if N increased to 1000 or more vehicle drive shifts towards very smooth drive however at sharp turns vehicle steer leads to slip out of road boundaries. A trade off with other parameters (differentiate and integrate error) tunning bring that stability. Herein twiddle implementation help a lot is stablizing system drive.

(5) Simulator Output & PID controller response

Simulator shown in picture displays run time data for – driving mode, speed and angle of motion
RED oval – display's vehicle speed in mph
Blue oval – gives angle in degree
Black oval – is a representation of backend PID prameters used by controller

Simulator herein run in autonomous mode on lake side track.

<u>Interface between a simulator (drive) and PID controller:</u>
Type – JSON
Port – 4567
Message – Standard fields for kinematics i.e. define in simulator and decoded in PID controller

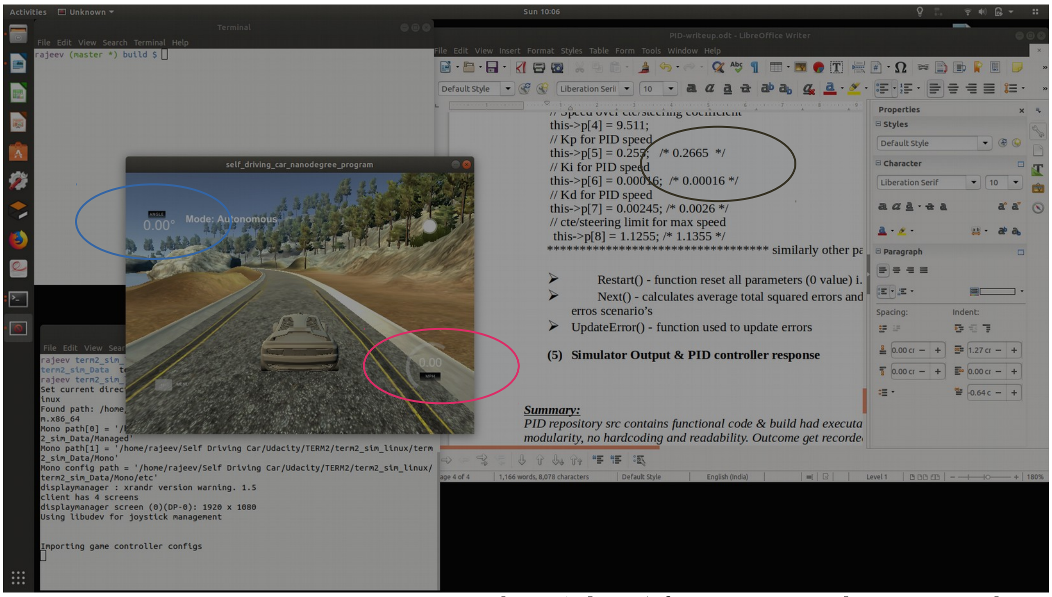


Image-5 : Simulator (udacity) for autonomous drive on a track

Begining of drive simulations & PID controller responses:

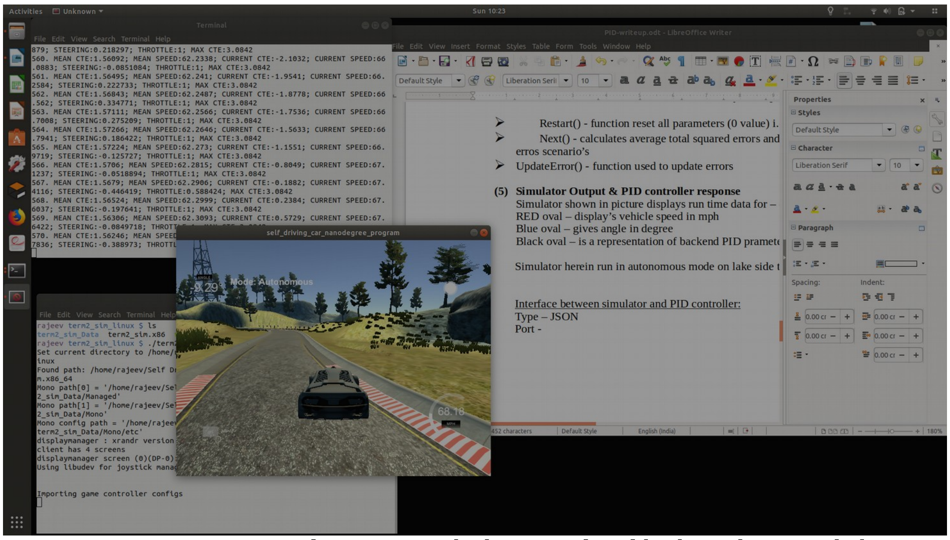
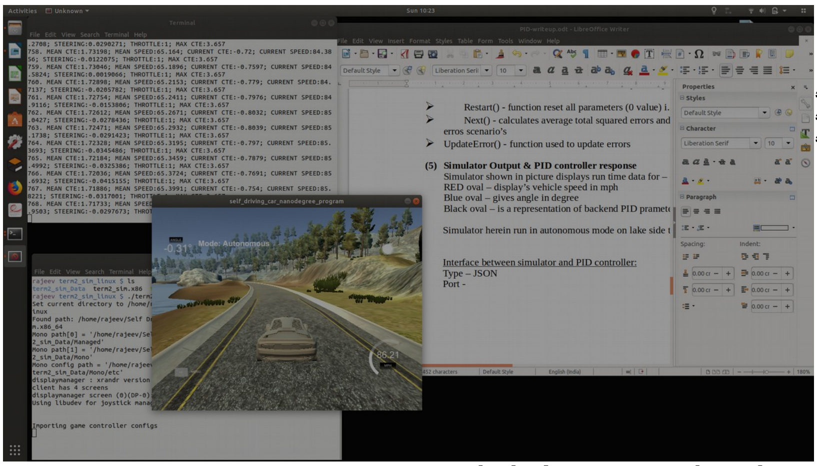


Image-6 : drive start with slow speed and high angle – PID behaviour

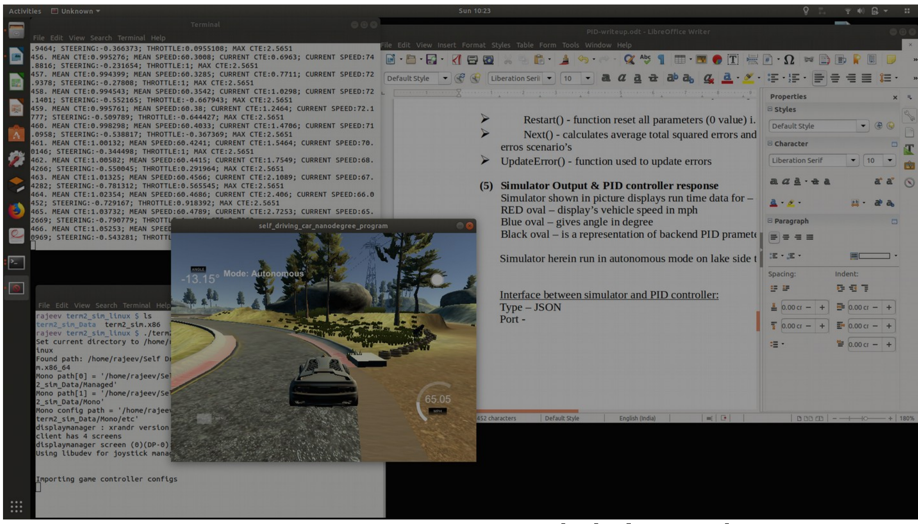
Vehicle drive on smooth track :



**** car drive smoothly***
**** Angle is nearly 0***
**** Speed gone high (85)***

Image-7 : Vehicle drive on smooth track

Vehicle drive on sharp turns :



** Vehicle touch track boundary (right)*
** Angle got adjusted*
** Speed dropped(65)*

Image-8 : Vehicle drive on sharp turn

(6) Video Outcomes

Simulation execution get recorded in mp4 format, here is recorded outcomes;

- *PID controller (Twiddle based) - https://www.youtube.com/watch?v=NXKu3brBGZo*
- *PID controller (without twiddle) - https://www.youtube.com/watch?v=wfyjxm88X8E*

(7) Debug log

***build*/outcome-logs**

/*********** Sampled outcome log section**

Listening to port 4567
1. MEAN CTE:0.577296; MEAN SPEED:0.876; CURRENT CTE:0.7598; CURRENT SPEED:0.876; STEERING:-1.09695; THROTTLE:1; MAX CTE:0.7598
2. MEAN CTE:0.577296; MEAN SPEED:1.533; CURRENT CTE:0.7598; CURRENT SPEED:2.19; STEERING:-0.0384623; THROTTLE:1; MAX CTE:0.7598
3. MEAN CTE:0.577296; MEAN SPEED:2.044; CURRENT CTE:0.7598; CURRENT SPEED:3.066; STEERING:-0.0436941; THROTTLE:1; MAX CTE:0.7598
4. MEAN CTE:0.577296; MEAN SPEED:2.628; CURRENT CTE:0.7598; CURRENT SPEED:4.38; STEERING:-0.048926; THROTTLE:1; MAX CTE:0.7598
5. MEAN CTE:0.577296; MEAN SPEED:3.1536; CURRENT CTE:0.7598; CURRENT SPEED:5.256; STEERING:-0.0541578; THROTTLE:1; MAX CTE:0.7598
6. MEAN CTE:0.577296; MEAN SPEED:3.65; CURRENT CTE:0.7598; CURRENT SPEED:6.132; STEERING:-0.0593896; THROTTLE:1; MAX CTE:0.7598
7. MEAN CTE:0.577274; MEAN SPEED:3.82794; CURRENT CTE:0.7597; CURRENT SPEED:4.8956; STEERING:-0.0644771; THROTTLE:1; MAX CTE:0.7598
8. MEAN CTE:0.57722; MEAN SPEED:3.5769; CURRENT CTE:0.7595; CURRENT SPEED:1.8196; STEERING:-0.0695595; THROTTLE:1; MAX CTE:0.7598
9. MEAN CTE:0.577077; MEAN SPEED:3.34904; CURRENT CTE:0.7589; CURRENT SPEED:1.5262; STEERING:-0.074203; THROTTLE:1; MAX CTE:0.7598
10. MEAN CTE:0.576886; MEAN SPEED:3.22138; CURRENT CTE:0.7584; CURRENT SPEED:2.0724; STEERING:-0.0795468; THROTTLE:1; MAX CTE:0.7598
11. MEAN CTE:0.576647; MEAN SPEED:3.10908; CURRENT CTE:0.7578; CURRENT SPEED:1.9861; STEERING:-0.0846027; THROTTLE:1; MAX CTE:0.7598
12. MEAN CTE:0.576322; MEAN SPEED:2.99232; CURRENT CTE:0.7568; CURRENT SPEED:1.708; STEERING:-0.089217; THROTTLE:1; MAX CTE:0.7598
13. MEAN CTE:0.575942; MEAN SPEED:2.90045; CURRENT CTE:0.7559; CURRENT SPEED:1.7979; STEERING:-0.0945288; THROTTLE:1; MAX CTE:0.7598
14. MEAN CTE:0.575434; MEAN SPEED:2.84874; CURRENT CTE:0.7542; CURRENT SPEED:2.1766; STEERING:-0.0985395; THROTTLE:1; MAX CTE:0.7598
15. MEAN CTE:0.574862; MEAN SPEED:2.82142; CURRENT CTE:0.7529; CURRENT SPEED:2.4389; STEERING:-0.104236; THROTTLE:1; MAX CTE:0.7598
16. MEAN CTE:0.574211; MEAN SPEED:2.81154; CURRENT CTE:0.7513; CURRENT SPEED:2.6634; STEERING:-0.10893; THROTTLE:1; MAX CTE:0.7598
17. MEAN CTE:0.573381; MEAN SPEED:2.83075; CURRENT CTE:0.7484; CURRENT SPEED:3.138; STEERING:-0.112157; THROTTLE:1; MAX CTE:0.7598
18. MEAN CTE:0.572453; MEAN SPEED:2.86488; CURRENT CTE:0.7461; CURRENT SPEED:3.4452; STEERING:-0.118049; THROTTLE:1; MAX CTE:0.7598
19. MEAN CTE:0.571308; MEAN SPEED:2.92206; CURRENT CTE:0.7421; CURRENT SPEED:3.9512; STEERING:-0.120632; THROTTLE:1; MAX CTE:0.7598
20. MEAN CTE:0.570042; MEAN SPEED:2.99133; CURRENT CTE:0.7389; CURRENT SPEED:4.3075; STEERING:-0.126722; THROTTLE:1; MAX CTE:0.7598
21. MEAN CTE:0.568643; MEAN SPEED:3.07192; CURRENT CTE:0.7353; CURRENT SPEED:4.6837; STEERING:-0.131092; THROTTLE:1; MAX CTE:0.7598
22. MEAN CTE:0.566939; MEAN SPEED:3.17241; CURRENT CTE:0.7288; CURRENT SPEED:5.2828; STEERING:-0.131811; THROTTLE:1; MAX CTE:0.7598
23. MEAN CTE:0.565067; MEAN SPEED:3.2825; CURRENT CTE:0.7238; CURRENT SPEED:5.7045; STEERING:-0.138711; THROTTLE:1; MAX CTE:0.7598

24. MEAN CTE:0.562432; MEAN SPEED:3.43019; CURRENT CTE:0.7084; CURRENT SPEED:6.827; STEERING:-0.128461; THROTTLE:1; MAX CTE:0.7598
25. MEAN CTE:0.559585; MEAN SPEED:3.5843; CURRENT CTE:0.7009; CURRENT SPEED:7.2829; STEERING:-0.144071; THROTTLE:1; MAX CTE:0.7598
26. MEAN CTE:0.556279; MEAN SPEED:3.75273; CURRENT CTE:0.6882; CURRENT SPEED:7.9634; STEERING:-0.141062; THROTTLE:1; MAX CTE:0.7598
27. MEAN CTE:0.552731; MEAN SPEED:3.92537; CURRENT CTE:0.6786; CURRENT SPEED:8.4142; STEERING:-0.149721; THROTTLE:1; MAX CTE:0.7598
28. MEAN CTE:0.548657; MEAN SPEED:4.10973; CURRENT CTE:0.6623; CURRENT SPEED:9.0874; STEERING:-0.144301; THROTTLE:1; MAX CTE:0.7598
29. MEAN CTE:0.544302; MEAN SPEED:4.29677; CURRENT CTE:0.6499; CURRENT SPEED:9.5338; STEERING:-0.153779; THROTTLE:1; MAX CTE:0.7598
30. MEAN CTE:0.539654; MEAN SPEED:4.48616; CURRENT CTE:0.6363; CURRENT SPEED:9.9786; STEERING:-0.155979; THROTTLE:1; MAX CTE:0.7598
31. MEAN CTE:0.534384; MEAN SPEED:4.68474; CURRENT CTE:0.6134; CURRENT SPEED:10.6421; STEERING:-0.146339; THROTTLE:1; MAX CTE:0.7598
32. MEAN CTE:0.528792; MEAN SPEED:4.88466; CURRENT CTE:0.5962; CURRENT SPEED:11.0822; STEERING:-0.157791; THROTTLE:1; MAX CTE:0.7598
33. MEAN CTE:0.522541; MEAN SPEED:5.08831; CURRENT CTE:0.5679; CURRENT SPEED:11.6051; STEERING:-0.145118; THROTTLE:1; MAX CTE:0.7598
34. MEAN CTE:0.515976; MEAN SPEED:5.28773; CURRENT CTE:0.5471; CURRENT SPEED:11.8685; STEERING:-0.158619; THROTTLE:1; MAX CTE:0.7598
35. MEAN CTE:0.50916; MEAN SPEED:5.49753; CURRENT CTE:0.5267; CURRENT SPEED:12.6308; STEERING:-0.162054; THROTTLE:1; MAX CTE:0.7598
36. MEAN CTE:0.502117; MEAN SPEED:5.71527; CURRENT CTE:0.5056; CURRENT SPEED:13.336; STEERING:-0.163778; THROTTLE:1; MAX CTE:0.7598
37. MEAN CTE:0.494839; MEAN SPEED:5.93347; CURRENT CTE:0.4825; CURRENT SPEED:13.7887; STEERING:-0.163449; THROTTLE:1; MAX CTE:0.7598
38. MEAN CTE:0.487101; MEAN SPEED:6.15147; CURRENT CTE:0.4481; CURRENT SPEED:14.2177; STEERING:-0.149447; THROTTLE:1; MAX CTE:0.7598
39. MEAN CTE:0.479286; MEAN SPEED:6.36781; CURRENT CTE:0.427; CURRENT SPEED:14.5885; STEERING:-0.17023; THROTTLE:1; MAX CTE:0.7598
40. MEAN CTE:0.471423; MEAN SPEED:6.58518; CURRENT CTE:0.4059; CURRENT SPEED:15.0627; STEERING:-0.172247; THROTTLE:1; MAX CTE:0.7598
***/

Summary:

PID repository src contains functional code & build had executables. All sections give desired result. Code has modularity, no hardcoding and readability. Outcome get recorded in a mp4 file (available in Output folder). Simulator and PID controler message exchange processing captured in debug logs which are available in repository.

Implementation of Twiddle and numerious trial of tunning paramter get helped in achieving a stable PID controller functioning to runa vehicle on lake side track. Sharp turns still make vehicle slightly unstable and push vehicle outside designated lane markings.

All execution done Ubuntu 17.10, GPU (GeForce GTX 1060)/CPU grade machine and compute execution being fairly fast.