FINAL REPORT – TRACTION CONTROL SYSTEM

LITERATURE REVIEW

Vehicles are being used to communicate from one place to another since the dawn of the Industrial revolution. It also ensures our safety while being transported irrespective of the weather conditions. Harsh weather conditions such as rain and snow lead to slipping of tires which may result in accident or even death. Irrespective of the engine, the tires ultimately produce the force to proper the engine forward. It doesn't matter what torque that is generated by the engine, if the tire slips relative to the road, traction force is lost and hence the car doesn't move forward as expected. This happens when the tires are unable to make a grip to the ground and get the required traction force to move the car forward. In the recent past, several techniques were put forth to control the car slipping factor (slip ratio) for a better, safer & controllable drive experience. To avoid the slipping of tires with respect to the road, several traction control systems (TCS) were developed. Traction control system basically helps is maintaining the wheel speed as close as to the vehicle speed so that maximum tractive force (force responsible for pushing the vehicle forward) is generated by the vehicle at any given time.

The most simple and popular way to control the torque and hence maintain the required slip ratio is through a PID controller. A PID or Proportion-Integral-Derivative controller is fed with an error function (in between the desired slip ratio and the actual slip ratio/the desired wheel speed and the actual wheel speed) and the output is a controlled torque that is being fed to the drive-train/wheels.

$$Torque = K_p \times error(t) + \ K_i \times \int_0^t error(t).dt + K_d \times \frac{derror(t)}{dt}$$

Here, K_p, K_i & K_d are the gain constants, that needs to be tuned according to the requirement.

When a slip occurs (the vehicle speed doesn't match with the wheel speed), the control algorithm tries to maintain an optimal difference between the wheel speed with the actual speed of the car (while maximizing the traction force) to prevent the car from slipping. Simple PID works, but isn't responsive enough for critical situation where the response time needs to be faster. Moreover, its difficult to get the PID gains when the road condition changes.

In the paper "FUZZY Based PID Controller for Speed Control of D.C. Motor Using LabVIEW", a simple D.C. Motor speed is being controlled by using a PID + Fuzzy control system. Here, the error between the desired wheel speed and the actual speed is being provided as an input the controller and this processes it to provide a controlled voltage to the motor. This paper, even though not related to Traction control system directly, provide a basic idea on how to create a control system by combining Fuzzy Inference logic and a PID controller to provide an input-controlled voltage (torque in case of vehicles) so that the output motor/wheel speed can be controlled. In the paper – "Traction control system using a fuzzy representation of the vehicle model", a robust control strategy has been put forth, where the engine torque is being controlled by a fuzzy state feedback controller. This controller is based on Takagi-Sugeno (TS) fuzzy representation of the vehicle dynamics, drivetrain and wheel motion. Here non-linear control strategy has been used based on the state-space representation. In order to compute the gain and to make the origin of the closed loop system asymptotically stable, Lyapunov's &H_∞ approaches are used, followed by TS fuzzy representation.

Several other paper such as ‘Development of traction control system’ & ‘Wheel Slip Control in Traction Control System for Vehicle Stability’ try to control the torque by using a throttle valve control mechanism. These types of approaches are too specific to internal of the combustion engine and are needed to be studied in details for each car type before implementation which is exhaustive. In the paper ‘Traction Control Algorithm Based on Fuzzy PID and PWM Modulation’, fuzzy + PID control strategy has been used along with PWM modulation (for braking) to control the torque & brake of the vehicle at any moment in time. Here, the error between the wheel speed & actual speed along with the derivative of the error is being fed to the fuzzy inference system for real time calculation of PID gains. Using the calculated gains and based on the error, the PID produced a controlled throttle as well as braking pressure separately and fed it to the vehicle for better control of the slip. Here the 2 fuzzy PID controller are working simultaneously to accelerate and slow down the vehicle which again is not ideal since the vehicle can lose control. With different road conditions, the desired slip ratio (maximize traction force) would be different. In the paper ‘Study on Optimal Slip Ratio Identification and Traction Control for Electric Vehicle’, the control strategy is in 2 folds. Firstly, the slip ratio is being determined by integrating the recursive least square algorithm with the forgetting factor which helps in determining the real time road adhesion coefficient (using PI integrator) and fuzzy inference to categorise the road condition based on adhesion coefficient. Secondly, based on the determined desired slip ratio, a PID controller has been used to control torque of the electric motor. The model has the advantage of determining the desired slip ratio in real time for varying road condition and hence is robust in nature. But again, adjusting the PID gains is a difficult task and lengthy.

In all these papers, different control strategy has been put forth to control the torque/braking pressure in order to control the car in case of slipping. Most of them has used PID controller as a component to effectively produce the controlled torque but determining the gains is a difficult task along with optimal slip ratio. Along with this, in all these cases, further studies, implementation & analysis need to be done. In this project, we are proposing a technique which involves a PID controller with fuzzy inference logic for real-time control of the PID’s gains based on the error (between the actual and desired slip ratio) to produce the control torque.

I. <u>Integration of Knowledge</u>

In an ideal condition, the tires of the vehicles stick perfectly with the road through the contact patch. Given the tires are made of rubber and hence elastic, it tries to regain its original shape. This restorative force provides the necessary traction force required to propel the vehicle forward. In slippery road, there is a chance that the driven tire is not perfectly stuck to the road surface and is rotating relative to the road surface. In such a situation, the free rolling velocity is different from the velocity with which the driven tires are rotating. In order to measure these relative velocities, slip ratio (k) was defined as per Eq. 1.

$$k = \frac{\dot{\theta}_{\omega} - \omega_0}{\omega_0} \qquad where\ \omega_0 = \frac{\dot{x}}{R_{L_R}} \tag{1}$$

In eq. (1), k is the slip ratio, $\dot{\theta}_{\omega}$is the driven wheel angular velocity, $\omega_0$is the free rolling velocity, $\dot{x}$ is the vehicle velocity and $R_{L_R}$is the Radius of the loaded rear tires given that the vehicle is an RWD car.

The equations of motion changes with the introduction of slip ratio and is given by the formula shown in Eq. 2 & 3.

$$(m + m_{EQ_{FT}})\ddot{x} = F_{X_R}(k) - f_r mg - 0 \cdot 5\rho C_D A\dot{x}^2 \tag{2}$$

$$I_{DT}\ddot{\theta}_{\omega R} = F_{X_R}(k)R_{T_R} - GR_{AX}GR_{TRAN}Te \tag{3}$$

m is the mass of the vehicle including the driver mass, $m_{EQ_{FT}}$ is the equivalent mass of the front train, $\ddot{x}$ is the acceleration of the vehicle, $F_{X_R}(k)$ is the tractive force value based on the current slip ratio, f_r is the rolling resistance coefficient, g is the value of gravitational acceleration, ρ is the air density, C_D is the Drag coefficient, A is the frontal dimension, $\dot{x}$ is the speed of the vehicle, I_{DT} is the mass moment of inertia of the entire drive train, $\ddot{\theta}_{\omega_R}$ is the angular acceleration of the loaded tire, $R_{T_R}$is the loaded tire radius, GR_{AX} is the gear ratio of the axle, GR_{TRAN} is the gear ratio of the transmission and Te is the input torque produced by the engine,

It has been observed that for a particular road condition, there is an optimal value k(typically around 0.2) for which the wheel's produces the maximum tractive force. At any given time and as per the road condition, the vehicle's wheel must rotate with a speed such that the slip ratio is always around this optimum value in order to provide the maximum tractive force and avoid slip. In this project, we have compared the traction control performance between a conventional PID controller and a fuzzy PID controller by controlling the input torque to achieve the optimal slip ratio.

II. <u>Methods</u>

<u>PID Controller</u>

A PID controller as explained in the previous section, is a Proportion-Integral-Derivative controller which produced a controlled output to the plant based on the error function as shown in Eq. 4. For a traction control system, the controlled output is a controlled torque being fed to the vehicles engine based on the error between the desired slip ratio and the current slip ratio.

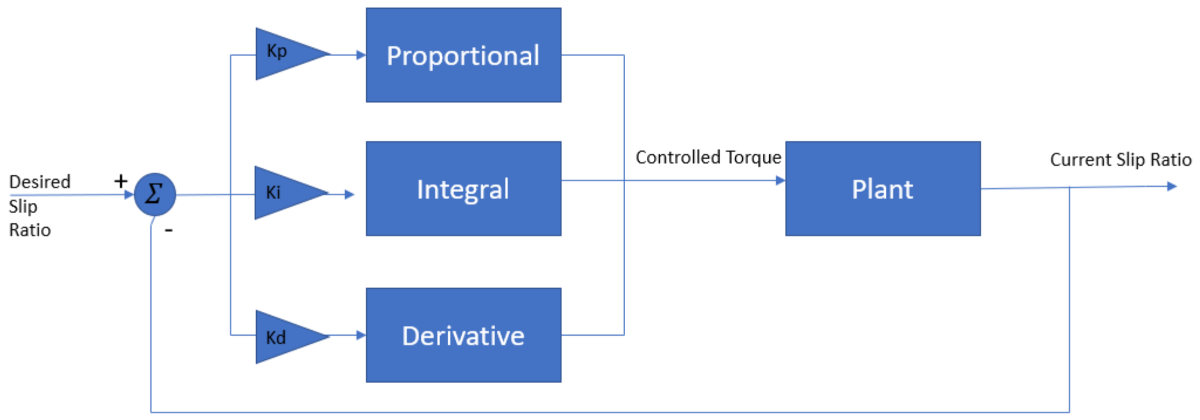


Fig. 1: PID Controller using Negative feedback structure

Based on K_p, K_i & K_d gains, it drives the current slip ratio to desired slip ratio is certain amount of time.

$$Torque = K_p \times error(t) + K_i \times \int_0^t error(t).dt + K_d \times \frac{derror(t)}{dt} \quad (4)$$

Fuzzy PID Controller

Fuzzy logic is a way to emulate human deductive thinking, the process humans use to infer conclusion from what they know. It helps in incorporating human expertise and experience while building a controller. It is a way to deal with computing in views of degrees of truth as opposed to just Boolean logic. In this project, the fuzzy logic has been used to self-tune the K_p, K_i & K_d gain values based on error function and derivative of the error function in real time.

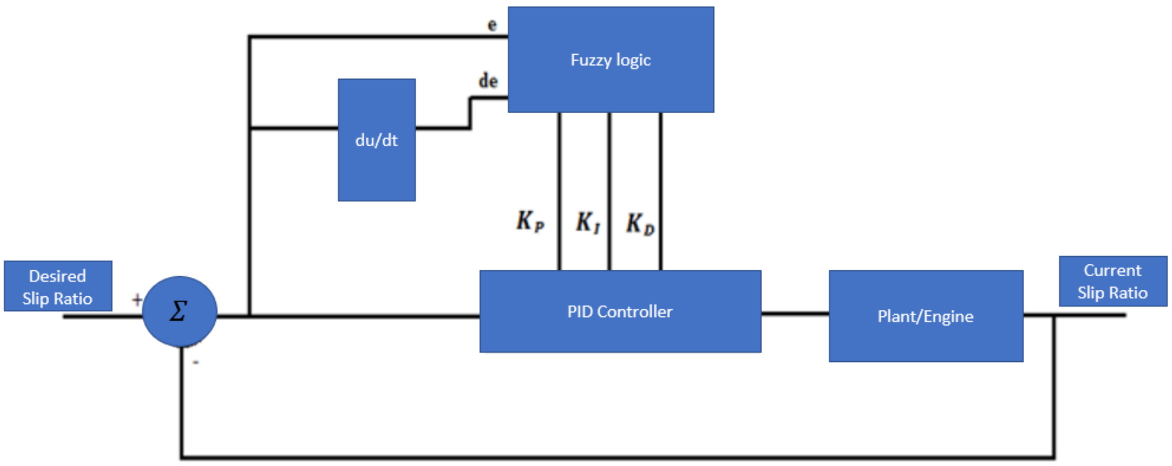


Fig. 2: Fuzzy-PID Controller using Negative feedback structure

In this project, we have kept $K_d = 0$ since the response was aperiodic in nature. 7 Fuzzy membership functions were defined for input (error, derivative of error with respect to real time) & output (K_p, K_i) which were triangular function in nature as shown in Fig. 3 & 4.

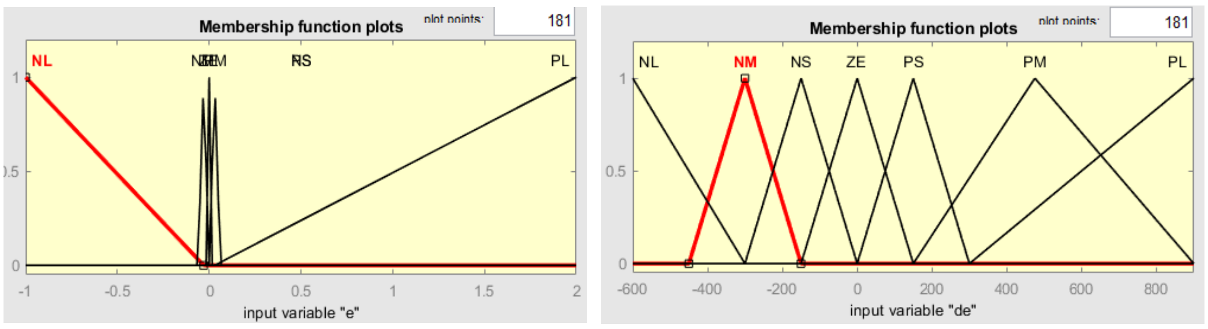


Fig 3: Membership functions for Inputs (a) Error (b) Derivative of Error with time

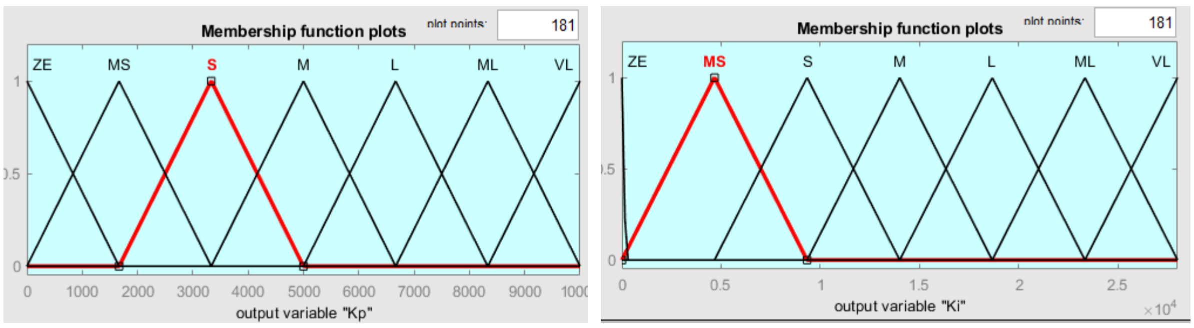


Fig 4: Membership functions for Inputs (a) Proportional gain (K_p) (b) Integral gain (K_i)

The fuzzy rules are designed in such a way that, at first when the error is large, the K_p value is large & K_i value is small. This logic ensures that the controlled torque grows faster at the beginning of the experiment. As the error start to decrease with time (Error integral increases), the K_p start to decrease and K_i start to increase. Since error start to decrease, the proportional gain will have minimal effect on the controlled torque and hence is trivial but the error sum/integral would have increases and hence the Integral gain would play a major role in keeping the controlled torque at the optimal point to maintain the optimal slip ratio.

III. Experimental Setup

In this project, in order to replicate how the controllers, perform under a slippery condition, a 2019 Corvette ZR1 (RWD) car operating in it's 1st gear has been taken into consideration. The specification sheet has been attached at the end of the report. An initial slip (i.e. $k = 0.01$ & $k = 0.15$) was introduced at the beginning of the simulation and the performance of PID and Fuzzy-PID controllers are compared. In Fig. 5, the tractive force vs. Slip ratio curve has been shown. As evident from the figure, the optimal slip ratio $k = 0.1$. The car has been given a rolling start of 5mph. In order to compare the performance of both the controllers, we will be comparing the settling time, rise time, percentage overshoot & mean square error.

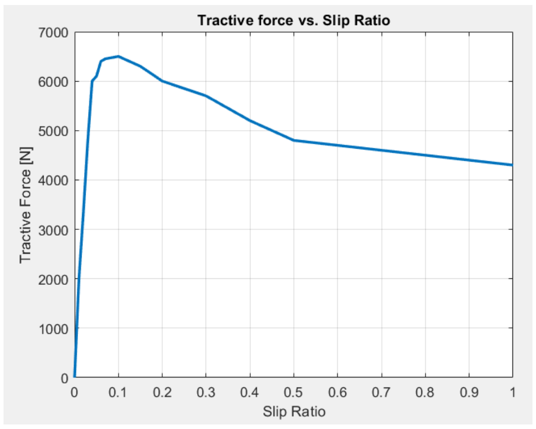


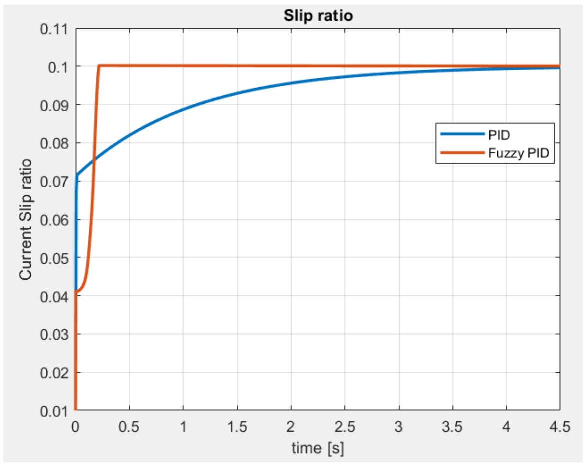
Fig. 5: Tractive Force vs. Slip Ratio Curve

In this project, we also have taken into consideration that the maximum achievable torque (controlled output of the PID controller) is 850 N-m which aligns with the specification charts.

IV. Results

For the conventional PID, the proportional, Integral and Derivative gain values were set as $K_p = 8000, K_i = 7550$ & $K_d = 0$ using trial-error method.

Initial Slip Ratio: $k = 0.01$ Initial Slip Ratio: $k = 0.15$



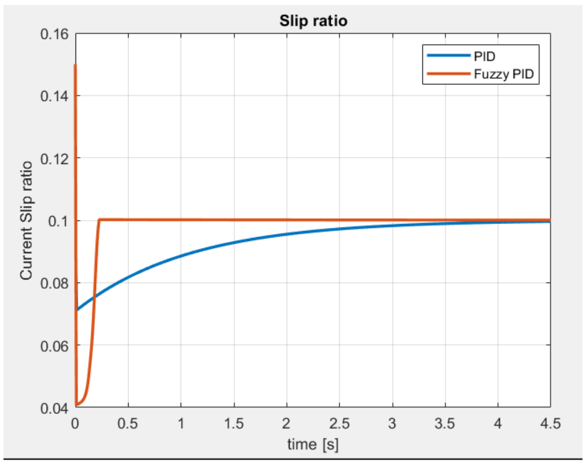
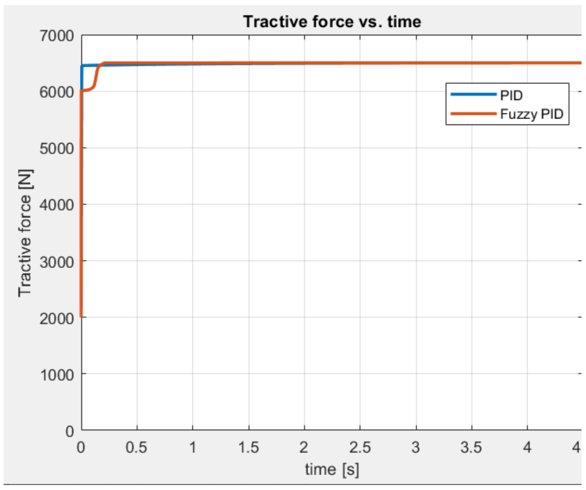


Fig. 6: Slip ratio vs. time



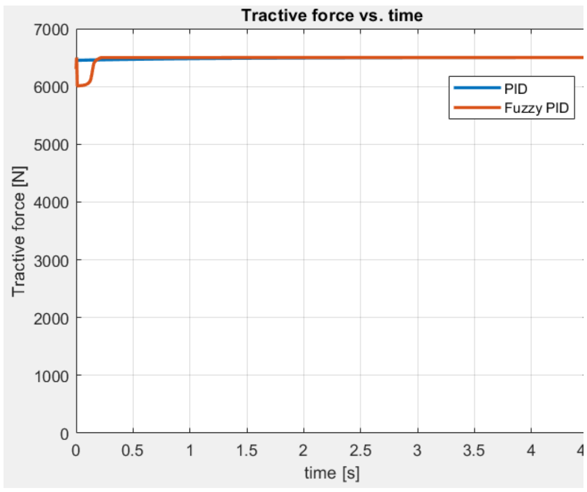
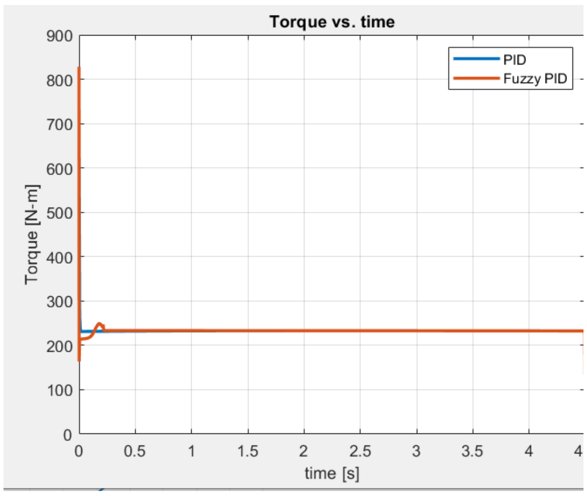


Fig. 6: Tractive force vs. time



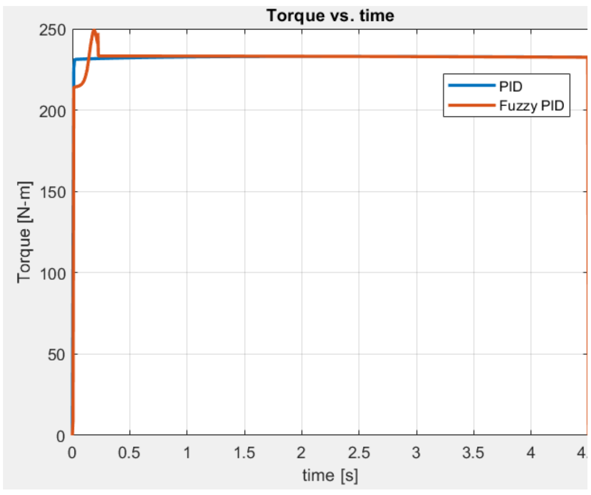


Fig. 7: Torque vs. time

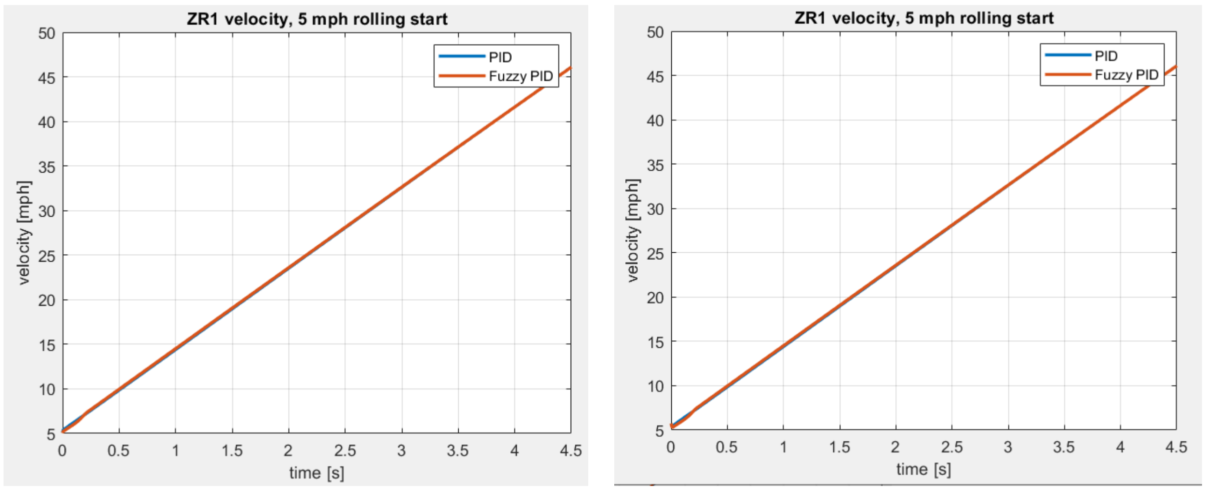


Fig. 8: Angular Velocity of wheel vs. time

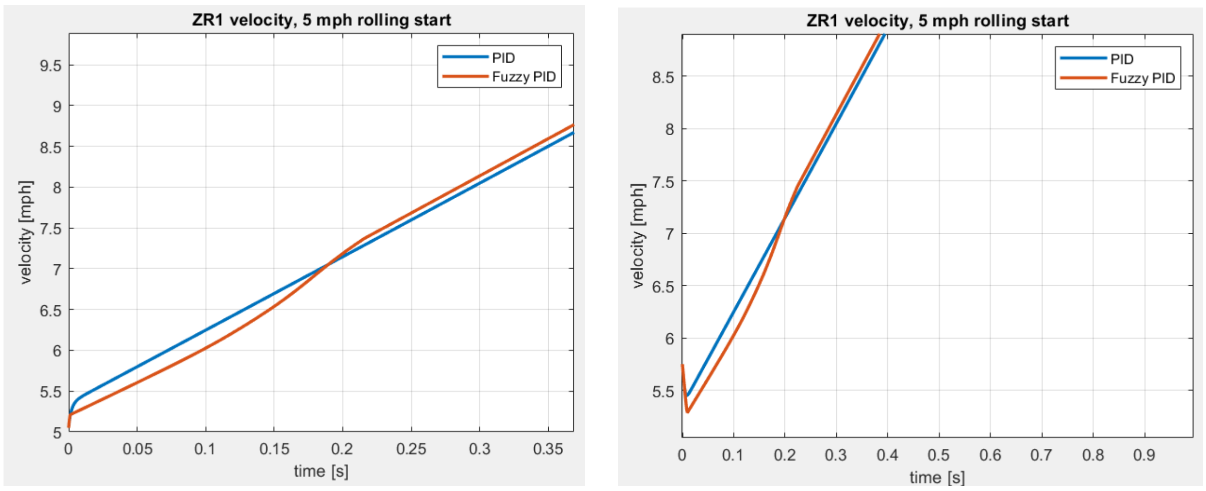


Fig. 9: Angular Velocity of wheel vs. time (zoomed)

As seen from figure 6, the fuzzy-PID controller performance is better than the convention PID controller. Fuzzy PID controller has better rise time as well as settling time when compared with the PID controller for both scenarios (initial slip, i.e. $k = 0.01$ & $k = 0.15$). With Initial slip ratio = 0.01, the settling time for fuzzy-PID controller and PID controller are 0.21 sec and 2.9 sec respectively. With Initial slip ratio = 0.01, the mean square error for fuzzy-PID controller and PID controller are 1.0636*10^{-4}e sec and 1.0090*10^{-4} respectively. With initial slip ratio = 0.15, the settling time for fuzzy-PID controller and PID controller are 0.22 sec and 2.85 sec respectively. The percentage overshoot is however greater while using a fuzzy-PID controller in this scenario ($k = 0.15$) which can cause undesirable side effects. With Initial slip ratio = 0.15, the mean square error for fuzzy-PID controller and PID controller are 1.0758*10^{-4}e sec and 1.0051*10^{-4} respectively. Figure 10 shows the proportional & integral gain values varying with respect to time according to fuzzy logic.

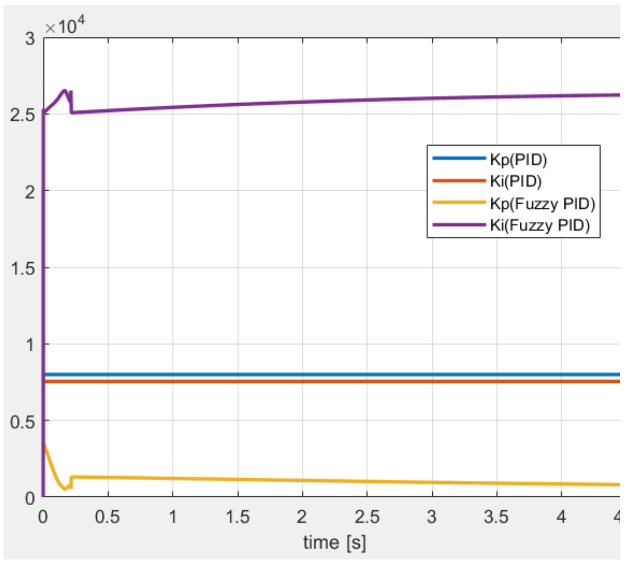


Fig. 10 – Proportional & Integral gains vs. time

V. <u>Discussion</u>

As seen from the results, the fuzzy-PID has been better rise and settling time than the PID controller. The mean square error of both the controller are similar with PID controller performing slightly better. The percentage overshoot is higher for fuzzy-PID controller with initial slip ratio = 0.15. In this case, the derivative gain needs to be introduced to minimize the percentage overshoot. In order for fuzzy-PID controller to be used in the real world, derivative gain needs to introduced inside the fuzzy inference system and extensive testing needs to be done with changing the optimum slip ratio value as well. In case of changing road condition (i.e. changing optimum slip ratio), fuzzy PID can be coupled with Recursive least square algorithm as discussed in the paper 'Study on Optimal Slip Ratio Identification and Traction Control for Electric Vehicle' to determine the optimal slip ratio in real time.

REFERENCE

- Wheel Slip Control in Traction Control System for Vehicle Stability - Jong Hyeon Park, Chan Young Kim, 1999
- Development of traction control system - Hun Sang Jung, Byung Hak Kwak, Young Jin Park,2000
- Traction Control Algorithm Based on Fuzzy PID and PWM Modulation - LiBo Chao, Liang Chu, Yang Ou, WenBo Lu, 2011
- Study on Optimal Slip Ratio Identification and Traction Control for Electric Vehicle - Qiang Gu, Xiusheng Cheng, 2011
- Traction control system using a fuzzy representation of the vehicle model - H. Dahmani, O. Pages and A. El Hajjaji,2015
- FUZZY Based PID Controller for Speed Control of D.C. Motor Using LabVIEW – Salim, Jyoti Ohri,2015

Table of Contents

APPENDIX

```
clc;clear all;close all;
```

Traction control system with Fuzzy PID

```
GR_ax1 = 2.41;                                                          %
 Gear ratio of Axle
GR_tran1 = 4.56;                                                        %
 Gear ratio of different gears

I_tran1 = 0.147;                                                        %
 Inertia of different gears
I_e1 = 0.09;                                                            %
 Inertia of engine
I_d1 = 0.12;                                                            %
 Inertia of driveshaft
I_ax1 = 0.003;                                                          %
 Inertia of Axle
I_tire1 = 1.2;                                                          %
 Inertia of tire
I_rear1 = I_tire1*2;                                                     %
 Inertia of rear wheels
thetaderedline_dot1 = 6500;                                             %
 Readline theta [RPM]
M1 = 1665;                                                              %
 mass of car
car_h1 = 1.23;                                                          %
 car's height
car_w1 = 1.97;                                                          %
 car's width
car_area1 = car_h1*car_w1;                                               %
 car's area
drag_c1 = 0.28;                                                         %
 Drag coefficient
resi_c1 = 0.015;                                                        %
 Friction coefficient
rho1 = 1.225;                                                           %
 air density
I_dt1 = I_tire1 + GR_ax1^2*I_d1 + (GR_ax1^2*GR_tran1^2*(I_e1 +
 I_tran1));  % Drivetrain Inertia

% based on Tire specification
```

```
Rt_front1 = ((19*0.0254) + (2*((30*285)/100000)))/2;                %
 Front tire radius
Rt_rear1 = ((20*0.0254) + (2*((25*335)/100000)))/2;                 %
 Rear tire radius

Meq_FT1 = (2*I_tire1)/(Rt_front1^2);
Meq1 = M1 + Meq_FT1;

fis = readfis;

% evalute torque from engine specification
P_hp1 = [0 10 40 80 160 220 240 270 330 400 480 540 620 660 670 675];
P_watt1 = P_hp1*746;
thetae_dot_rpm1 = [0 500 1000 1500 2000 2100 2300 2500 3000 3500 4000
 4500 5000 5500 6000 6500];
thetae_dot_rad1 = (2*pi/60)*thetae_dot_rpm1;
T1 = P_watt1./thetae_dot_rad1;

% evalute rolling resistance force
F_fric1 = resi_c1*M1*9.8;

N1 = 45001;                         % vector length
dt1 = 0.0001;                        % time increment
t1 = (0:dt1:(N1-1)*dt1);

xdot1 = zeros(N1,1);             % velocity vector for plot
xdot1(1) = 5/2.23694;               % initial velocity = 5mph

x1 = zeros(N1,1);                    % distance vector for plot
x1(1) = 0;                          % initial distance = 0m

thetae_dot_rad_instant1 = zeros(N1,1);
P_watt_instant1 = zeros(N1,1);
T_instant1 = zeros(N1,1);

thetaw_dot_req1 = zeros(N1,1);
sr_instant1 = zeros(N1,1);
thetaw_dot_req1(1) = xdot1(1)/Rt_rear1;
thetaw_dot_instant1 = zeros(N1,1);
t_force_instant1 = zeros(N1,1);
xddot1 = zeros(N1,1);
thetaw_ddot_instant1 = zeros(N1,1);
sr_instant1(1) = 0.01;
% calculate intial wheel angular velocity from slip ratio
thetaw_dot_instant1(1) = xdot1(1)*(sr_instant1(1)+1)*(1/Rt_rear1);

% input traction force vs slip ratio
t_force1 = [0 2000 3333 4777 6000 6100 6400 6450 6500 6300 6000 5700
 5200 5000 4800 4700 4600 4500 4400 4300];
sr1 = [0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.1 0.15 0.2 0.3 0.4 0.45
 0.5 0.6 0.7 0.8 0.9 1];

sr_desired1 = 0.1;
error1=zeros(N1,1);
```

```
error1(1) = sr_desired1 - sr_instant1(1);
Kp1 = 8000; Kd1 = 0; Ki1 = 7550;
de1 = 0; error_sum1 = 0;
for k = 2:N1
% 1st gear only
% get engine speed from wheel speed
        thetae_dot_rad_instant1(k-1) =
 (thetaw_dot_instant1(k-1)*GR_ax1*GR_tran1(1));
% PID Controller
        if k == 2
            de1 = error1(k-1);
        else
            de1 = error1(k-1) - error1(k-2);
        end
        error_sum1 = error1(k-1)*dt1 + error_sum1;
% get Kp and Ki from FIS
        [K1] = evalfis(fis,[error1(k-1) de1/dt1]);

        de_wrt_t1(k-1) = de1/dt1;
        kp1(k-1) = K1(1);
        kd1(k-1) = Kd1;
        ki1(k-1) = K1(2);

% Controlled Torque
         T_instant1(k-1) = kp1(k-1)*error1(k-1) + kd1(k-1)*(de1/dt1) +
 ki1(k-1)*error_sum1;
         % The torque can't drop's below zero, it's not possible
         if T_instant1(k-1) < 0
            T_instant1(k-1) = 0;
         end

% logic to get tractive force
        if sr_instant1(k-1) >= 0
            t_force_instant1(k-1) = interp1(sr1,t_force1, ...
                                      sr_instant1(k-1));
        else
            t_force_instant1(k-1) = 0;
        end
% calculate linear acceleration (to get linear velocity for next
 iteration)
        xddot1(k-1) = ((t_force_instant1(k-1)/Meq1(1)) - ...
                      (F_fric1/Meq1(1)) -
 ((0.5*rho1*drag_c1*car_area1/Meq1(1))* ...
                      (xdot1(k-1).^2)));

% calculate angular wheel acceleration (to get angular wheel velocity
 for next iteration)
        thetaw_ddot_instant1(k-1) = (-(t_force_instant1(k-1)*Rt_rear1/
I_dt1) + ...

 (GR_ax1*GR_tran1*T_instant1(k-1)/I_dt1));

% use euler method to calculate speed and distance & angular wheel
 speed
```

```
        xdot1(k) = xdot1(k-1) + xddot1(k-1)*dt1;
        x1(k) = x1(k-1) + xdot1(k-1)*dt1;
        thetaw_dot_instant1(k) = thetaw_dot_instant1(k-1) +
 thetaw_ddot_instant1(k-1)*dt1;

% get the actual velocity of car
        thetaw_dot_req1(k) = xdot1(k)/Rt_rear1;
% get the new Slip ratio
        sr_instant1(k) = (thetaw_dot_instant1(k) -
 thetaw_dot_req1(k))/thetaw_dot_req1(k);
        error1(k) = sr_desired1 - sr_instant1(k);
end

% for fuzzy
    de_wrt_t1(k) = de_wrt_t1(k-1);
    kp1(k) = Kp1;
    kd1(k) = Kd1;
    ki1(k) = Ki1;

Warning: Input 2 expects a value in range [-600 900], but has a value
 of 900.
```

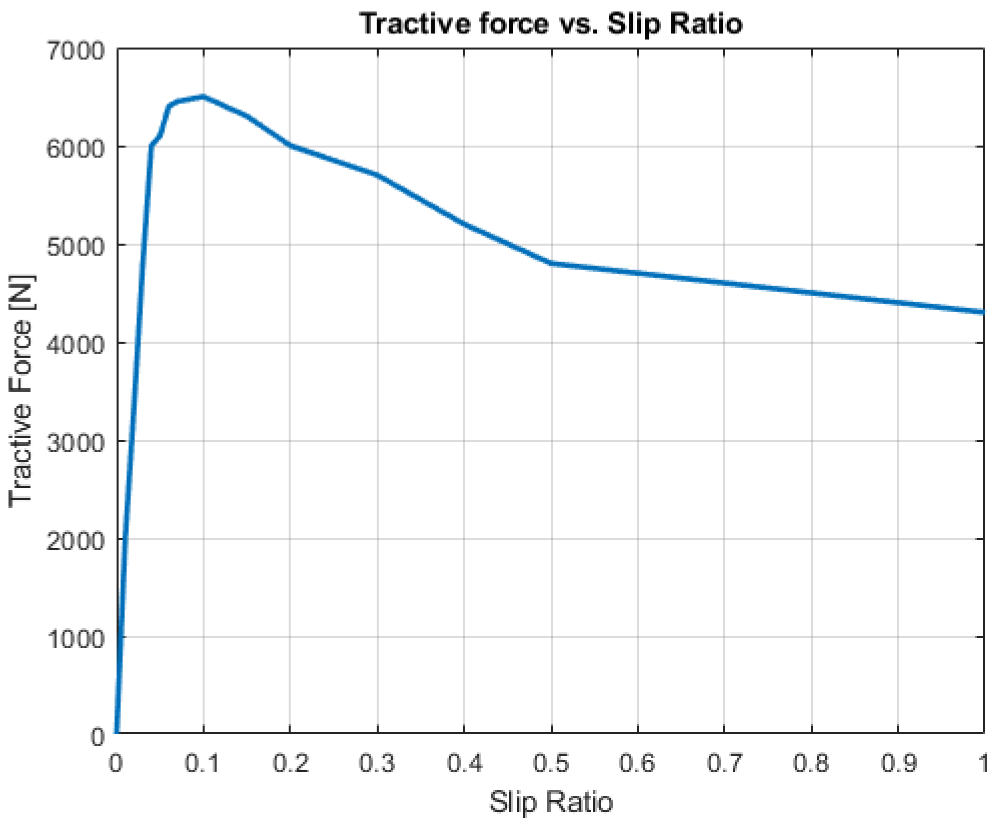
Traction control system with PID

```
GR_ax = 2.41;                                                          %
 Gear ratio of Axle
GR_tran = 4.56;                                                        %
 Gear ratio of different gears

I_tran = 0.147;                                                        %
 Inertia of different gears
I_e = 0.09;                                                            %
 Inertia of engine
I_d = 0.12;                                                            %
 Inertia of driveshaft
I_ax = 0.003;                                                          %
 Inertia of Axle
I_tire = 1.2;                                                          %
 Inertia of tire
I_rear = I_tire*2;                                                     %
 Inertia of rear wheels
thetaderedline_dot = 6500;                                             %
 Readline theta [RPM]
M = 1665;                                                              %
 mass of car
car_h = 1.23;                                                          %
 car's height
car_w = 1.97;                                                          %
 car's width
car_area = car_h*car_w;                                                %
 car's area
drag_c = 0.28;                                                         %
 Drag coefficient
```

```
resi_c = 0.015;                                                    %
 Friction coefficient
rho = 1.225;                                                       %
 air density
I_dt = I_tire + GR_ax^2*I_d + (GR_ax^2*GR_tran^2*(I_e + I_tran));  %
 Drivetrain Inertia

% based on Tire specification
Rt_front = ((19*0.0254) + (2*((30*285)/100000)))/2;                %
 Front tire radius
Rt_rear = ((20*0.0254) + (2*((25*335)/100000)))/2;                 %
 Rear tire radius

Meq_FT = (2*I_tire)/(Rt_front^2);
Meq = M + Meq_FT;

% evalute torque from engine specification
P_hp = [0 10 40 80 160 220 240 270 330 400 480 540 620 660 670 675];
P_watt = P_hp*746;
thetae_dot_rpm = [0 500 1000 1500 2000 2100 2300 2500 3000 3500 4000
 4500 5000 5500 6000 6500];
thetae_dot_rad = (2*pi/60)*thetae_dot_rpm;
T = P_watt./thetae_dot_rad;

% evalute rolling resistance force
F_fric = resi_c*M*9.8;

N = 45001;                         % vector length
dt = 0.0001;                        % time increment
t = (0:dt:(N-1)*dt);

xdot = zeros(N,1);             % velocity vector for plot
xdot(1) = 5/2.23694;               % initial velocity = 5mph

x = zeros(N,1);                    % distance vector for plot
x(1) = 0;                          % initial distance = 0m

thetae_dot_rad_instant = zeros(N,1);
P_watt_instant = zeros(N,1);
T_instant = zeros(N,1);

thetaw_dot_req = zeros(N,1);
sr_instant = zeros(N,1);
thetaw_dot_req(1) = xdot(1)/Rt_rear;
thetaw_dot_instant = zeros(N,1);
t_force_instant = zeros(N,1);
xddot = zeros(N,1);
thetaw_ddot_instant = zeros(N,1);
sr_instant(1) = 0.01;
% calculate intial wheel angular velocity from slip ratio
thetaw_dot_instant(1) = xdot(1)*(sr_instant(1)+1)*(1/Rt_rear);

% input traction force vs slip ratio
```

```
t_force = [0 2000 3333 4777 6000 6100 6400 6450 6500 6300 6000 5700
 5200 5000 4800 4700 4600 4500 4400 4300];
sr = [0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.1 0.15 0.2 0.3 0.4 0.45
 0.5 0.6 0.7 0.8 0.9 1];

sr_desired = 0.1;
error=zeros(N,1);
error(1) = sr_desired - sr_instant(1);
Kp = 8000; Kd = 0; Ki = 7550;
de = 0; error_sum = 0;
for k = 2:N
% 1st gear only
% get engine speed from wheel speed
        thetae_dot_rad_instant(k-1) =
 (thetaw_dot_instant(k-1)*GR_ax*GR_tran(1));
% Instead of getting torque from Engine speed, torque is controlled
 through
% PID Controller
        if k == 2
            de = error(k-1);
        else
            de = error(k-1) - error(k-2);
        end
        error_sum = error(k-1)*dt + error_sum;
% related to fuzzy
        de_wrt_t(k-1) = de/dt;
        kp(k-1) = Kp;
        kd(k-1) = Kd;
        ki(k-1) = Ki;

% Controlled Torque
         T_instant(k-1) = Kp*error(k-1) + Kd*(de/dt) + Ki*error_sum;
         % The torque can't drop's below zero, it's not possible
         if T_instant(k-1) < 0
            T_instant(k-1) = 0;
         end

% logic to get tractive force
        if sr_instant(k-1) >= 0
            t_force_instant(k-1) = interp1(sr,t_force, ...
                                             sr_instant(k-1));
        else
            t_force_instant(k-1) = 0;
        end
% calculate linear acceleration (to get linear velocity for next
 iteration)
        xddot(k-1) = ((t_force_instant(k-1)/Meq(1)) - ...
                      (F_fric/Meq(1)) - ((0.5*rho*drag_c*car_area/
Meq(1))* ...
                      (xdot(k-1).^2)));

% calculate angular wheel acceleration (to get angular wheel velocity
 for next iteration)
```

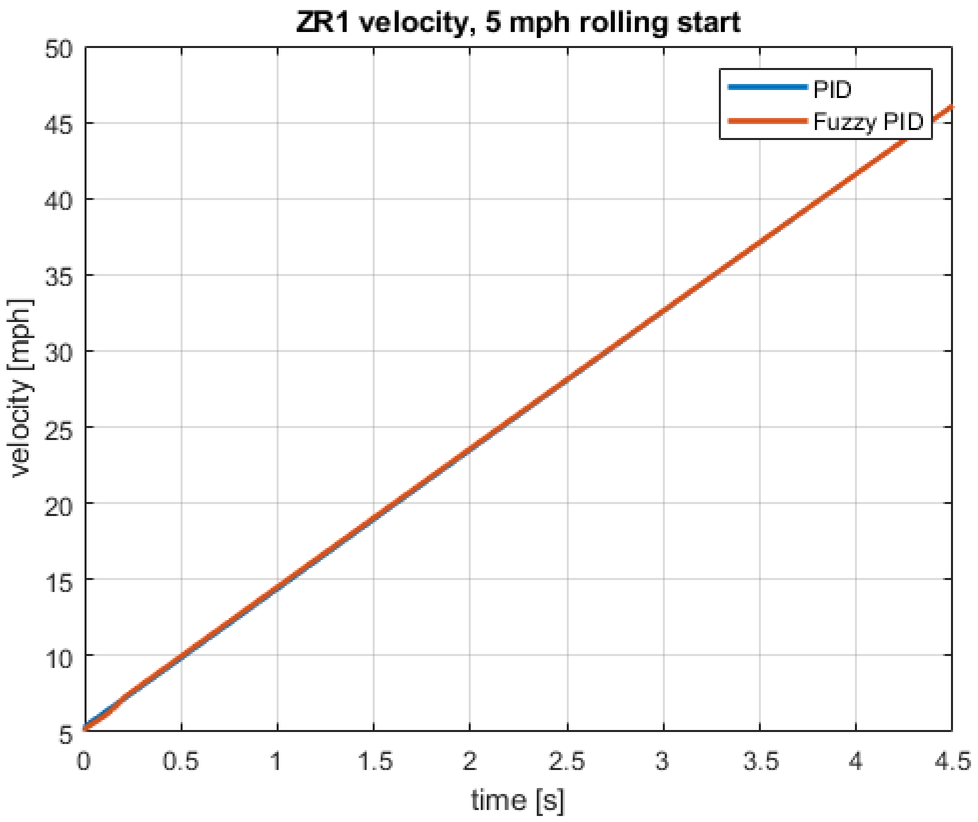
```
        thetaw_ddot_instant(k-1) = (-(t_force_instant(k-1)*Rt_rear/
I_dt) + ...
                                        (GR_ax*GR_tran*T_instant(k-1)/
I_dt));

% use euler method to calculate speed and distance & angular wheel
 speed
        xdot(k) = xdot(k-1) + xddot(k-1)*dt;
        x(k) = x(k-1) + xdot(k-1)*dt;
        thetaw_dot_instant(k) = thetaw_dot_instant(k-1) +
 thetaw_ddot_instant(k-1)*dt;

% get the actual velocity of car
        thetaw_dot_req(k) = xdot(k)/Rt_rear;
% get the new Slip ratio
        sr_instant(k) = (thetaw_dot_instant(k) - thetaw_dot_req(k))/
thetaw_dot_req(k);
        error(k) = sr_desired - sr_instant(k);
end

% for fuzzy
    de_wrt_t(k) = de_wrt_t(k-1);
    kp(k) = Kp;
    kd(k) = Kd;
    ki(k) = Ki;

figure
plot(sr,t_force,'linewi',2),xlabel('Slip Ratio'),ylabel('Tractive
 Force [N]'),
title('Tractive force vs. Slip Ratio'),grid on
```

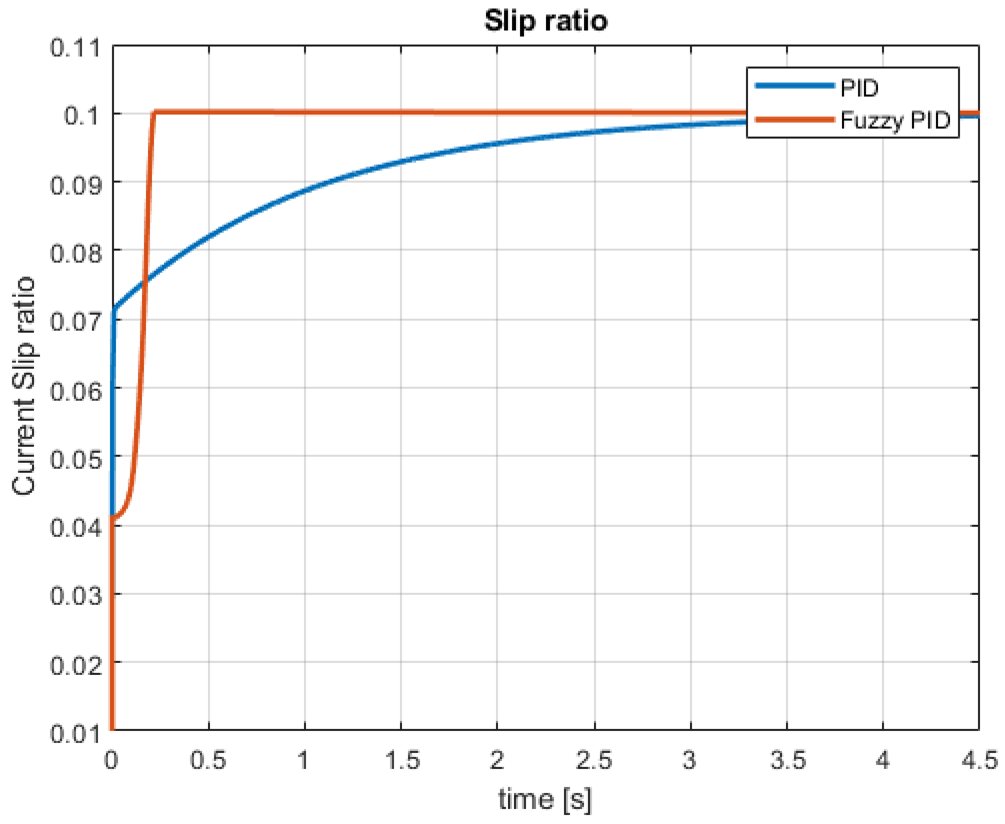


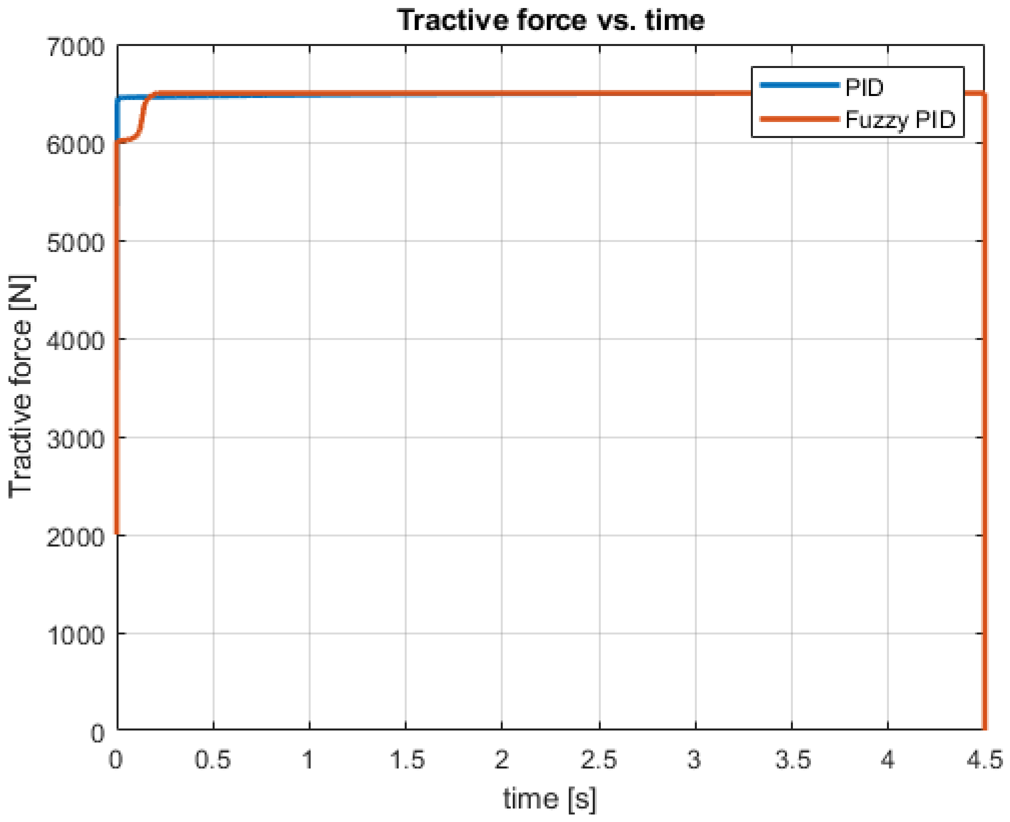
comparsion between PID and Fuzzy-PID Controller

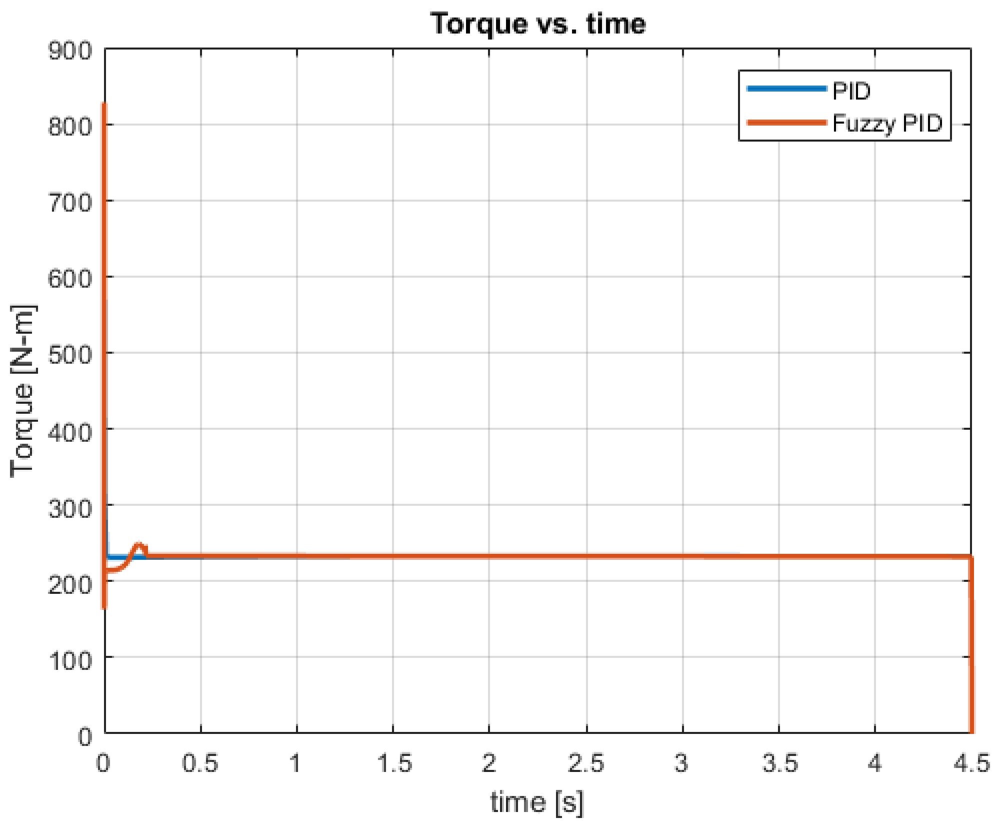
```
figure
plot(t,thetaw_dot_instant*Rt_rear*2.23694,t,thetaw_dot_instant1*Rt_rear*2.23694,'l
xlabel('time [s]'),ylabel('velocity [mph]')
title('ZR1 velocity, 5 mph rolling start'),grid on
legend('PID','Fuzzy PID')

figure
plot(t,sr_instant,t,sr_instant1,'linewi',2)
xlabel('time [s]'),ylabel('Current Slip ratio'),grid on
title('Slip ratio')
legend('PID','Fuzzy PID')

figure
plot(t,t_force_instant,t,t_force_instant1,'linewi',2),grid on
xlabel('time [s]'),ylabel('Tractive force [N]')
title('Tractive force vs. time')
legend('PID','Fuzzy PID')

figure
plot(t,T_instant,t,T_instant1,'linewi',2),grid on
xlabel('time [s]'),ylabel('Torque [N-m]')
```

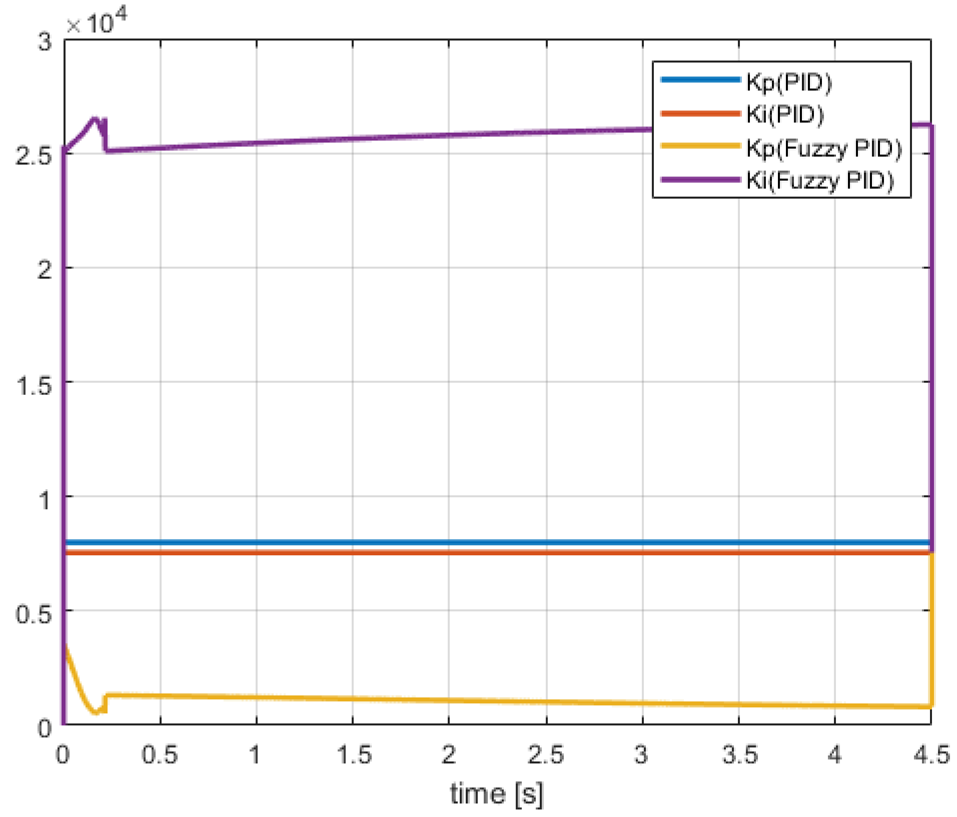
```
title('Torque vs. time')
legend('PID','Fuzzy PID')

figure
plot(t,kp,t,ki,t,kp1,t,ki1,'linewi',2),grid on
xlabel('time [s]')
legend('Kp(PID)','Ki(PID)','Kp(Fuzzy PID)','Ki(Fuzzy PID)')
```




Slip ratio
Current Slip ratio
time [s]
PID
Fuzzy PID
0.11
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5


Tractive force vs. time
Tractive force [N]
time [s]
PID
Fuzzy PID
7000
6000
5000
4000
3000
2000
1000
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5


Torque vs. time
Torque [N-m]
time [s]
PID
Fuzzy PID
0
100
200
300
400
500
600
700
800
900
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5


×10^4
time [s]
Kp(PID)
Ki(PID)
Kp(Fuzzy PID)
Ki(Fuzzy PID)
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5

Published with MATLAB® R2020b