



Optimized self-adaptive PID speed control for autonomous vehicles

Yassine Kebbaty, Naïma Aït Oufroukh, Vincent Vigneron, Dalil Ichalal,
Dominique Gruyer

► To cite this version:

Yassine Kebbaty, Naïma Aït Oufroukh, Vincent Vigneron, Dalil Ichalal, Dominique Gruyer. Optimized self-adaptive PID speed control for autonomous vehicles. 26th International Conference on Automation and Computing (ICAC 2021), Sep 2021, Portsmouth, United Kingdom. pp.1-6, 10.23919/ICAC50006.2021.9594131 . hal-03442081

HAL Id: hal-03442081

<https://hal.science/hal-03442081>

Submitted on 3 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimized self-adaptive PID speed control for autonomous vehicles

Yassine Kebbat^{1,*}, Naima Ait-Oufroukh¹, Vincent Vigneron^{1,2}, Dalil Ichalal¹, and Dominique Gruyer³

¹IBISC-EA4526, Université Evry, Université Paris-Saclay, France

²School of Applied Sciences (FCA), UNICAMP, Limeira, Brazil

³IFSTTAR/ PICS-L, Université Gustave Eiffel, France

Email*: yassine.kebbati@univ-evry.fr

Abstract—The main control tasks in autonomous vehicles are steering (lateral) and speed (longitudinal) control. PID controllers are widely used in the industry because of their simplicity and good performance, but they are difficult to tune and need additional adaptation to control nonlinear systems with varying parameters. In this paper, the longitudinal control task is addressed by implementing adaptive PID control using two different approaches: Genetic Algorithms (GA-PID) and then Neural Networks (NN-PID) respectively. The vehicle nonlinear longitudinal dynamics are modeled using Powertrain blockset library. Finally, simulations are performed to assess and compare the performance of the two controllers subject to external disturbances.

Index Terms—Autonomous Vehicle, Adaptive PID Control, Neural Networks, Genetic Algorithms, Optimization.

I. INTRODUCTION

The autonomous vehicle technology has been the main trend in the last decade. Although artificial intelligence and control researchers are quickly closing the gap to a real self-driving vehicle, several significant challenges still persist. A fully autonomous vehicle requires fully automatic steering and speed control modules. Cruise control systems (CC) control the speed of the vehicle and have been the subject of extensive research: Osman *et al.* [1] and Simorgh *et al.* [2] for instance showed that CC versions for highway driving worked only above certain speed limits as the braking part was not addressed [3]. Adaptive CC systems were introduced later on to tackle both acceleration and braking [4], these systems are divided into high level and low level controllers. The first computes the required acceleration or deceleration, and the second actuates the throttle or the brake. These systems allowed to automate vehicles even for urban driving.

Many speed controllers are available in the literature. [5] worked on the sliding mode control for vehicle platooning in uncertain topologies. [6] designed a preview controller that integrates road slope, speed profile and vehicle dynamics to form an augmented optimal control. [7] used the model predictive control (MPC) for low velocity tracking. Actually, most control systems are based on the PID controller which is easy to design and yet performs very well. However, the non-linearities of the vehicle subsystems and the variety of disturbances that face the vehicle such as road elevation, wind speed and road conditions require the PID to be adaptive to these changing parameters. [2] and [3] worked

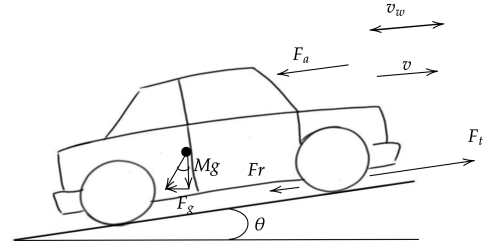


Fig. 1: Forces and disturbances affecting vehicle dynamics.

on adaptive PID control using the inverse model theory, [8] used the twiddle algorithm to tune a PID for trajectory tracking and [9] used Genetic Algorithms (GA) to tune a PID for controlling DC motors. [10] and [11] used NN-PID for frequency regulation and vehicle steering control respectively. In [12], the authors used radial basis function to adapt PID gains for the vehicle longitudinal control. The research of [13] presented an adaptive PID with GA optimization for the lateral control task. However, most of these methods are based on oversimplified models excluding or simplifying external disturbances which significantly affects the performance in real-life situations. In addition, most adaptation methods require the measurement of certain parameters that are not easily accessible. To address this issue, two PID adaptation methods were applied to control the vehicle velocity; the first method is based on a GA with a complete vehicle model taking into account external disturbances, and the second one uses a feed forward neural network (NN) that utilises only the error and control signals to adapt the PID gains. Simulations were carried to assess the performance of the designed controllers. Section II of this paper deals with the modeling of the vehicle longitudinal dynamics. The design, optimization and adaptation of the controller are presented in section III. The performance of the controller is evaluated and discussed in section IV, and section V concludes the paper and gives some perspectives for future work.

II. VEHICLE MODELING

The longitudinal control of an autonomous vehicle regulates its speed to follow a speed profile. The latter is obtained from a speed planner or profiler based on changes of the

external environment like the shape of the road, traffic signs and weather conditions. The vehicle is mainly propelled by the engine drive torque and stopped by the braking torque.

A. Vehicle Body Dynamics

The longitudinal dynamics of the vehicle are obtained by applying Newton's second law as shown in Fig. 1, this results in a simplified point mass model [14]:

$$\begin{cases} F_t = M \frac{dv}{dt} + F_a + F_g + F_r \\ F_g = Mg \sin \theta \\ F_a = \frac{1}{2} \rho C_d A v^2 \\ F_r = Mg C_r \cos \theta \end{cases} \quad (1)$$

where the parameters are defined as:

- M the total mass of the vehicle
- F_t the traction force (drive force)
- F_a the aerodynamic drag (air friction)
- F_g the downgrade force (gravity pull)
- F_r the rolling resistance force
- ρ the air density
- C_r the rolling resistance coefficient
- C_d the drag coefficient
- A the vehicle cross sectional area
- $\frac{dv}{dt}$ the vehicle acceleration/deceleration.

B. Engine Dynamics

The engine and transmission dynamics make up the vehicle's propulsion system and are highly non-linear. For electric vehicles, the engine is an electric motor that supplies power to the wheels through the transmission. The latter is linked to the electric drive through a gearbox which can be modeled under the assumption of no losses by (2):

$$\begin{cases} T_g = k_g T_e \\ \omega_g = k_g \omega_e \end{cases} \quad (2)$$

where ω_g and ω_e are the angular speeds of the gear and the engine, T_g and T_e are the gear and electric drive torques and k_g represents the ratio of the gearbox. The electric torque is generated by the electric machine and its model can be static or dynamic [15]. The dynamic model depends on the type of machine whether it is a DC or AC (synchronous/asynchronous) motor. It links the input current/voltage to the output electric torque. The static model which is used in this work and given in (3), is a data-driven model that is based on the electric drive efficiency map:

$$\begin{cases} T_e = T_{e-ref} \\ I_b = \frac{T_e \omega_g}{u_b \eta^k} \end{cases} \quad (3)$$

with $k = 1$ if $T_e \omega_g \geq 0$, $k = -1$ otherwise. T_{e-ref} represents the reference electric torque obtained from the motor efficiency map, u_b denotes the battery voltage, I_b and η are the battery current and motor efficiency respectively. Positive currents discharge the battery while negative currents, as in regenerative braking, charge the battery.

C. Wheel Dynamics

The wheel translates the gear torque into a traction force that propels the vehicle. Thus, the angular speed of the wheel determines the velocity of the vehicle as the following:

$$\begin{cases} F_{t/b} = \frac{1}{R_w} T_{g/b} \\ \omega_w = \frac{1}{R_w} V \end{cases} \quad (4)$$

where $F_{t/b}$ and $T_{g/b}$ denote either the tractive force (Fig. 1) and the drive torque or the braking force and braking torque respectively. R_w and V are the respective wheel radius and vehicle longitudinal velocity.

D. Tire Dynamics

The tires of a vehicle undergo different kinds of deformations and behave in a highly nonlinear manner, the most widely used longitudinal tire model is the magic formula or Pacejka model [16]:

$$F_x = F_z D \sin(C \tan^{-1}[Bk - E[Bk - \tan^{-1}(Bk)]]]) \quad (5)$$

with F_x and F_z being the tire longitudinal force and vertical load, k representing the wheel slip and B, C, D and E being empirical tire coefficients.

E. Brake Dynamics

The brakes of the vehicle can be either a drum or a disk system. The disk braking system has been used in this paper for both front and rear wheels. The braking torque delivered by the mechanism is given as:

$$T_b = \frac{\mu P \pi B_a^2 R_m N_{pads}}{4} \quad (6)$$

where P, R_m and B_a are the applied brake pressure, the mean radius of the brake pad and the brake actuator diameter respectively. μ and N_{pads} are the disc-pad friction coefficient and the number of pads respectively.

F. Battery modeling

The electric motor is powered by the battery voltage u_b which can be modeled by a resistor R_b in series with an open circuit voltage V_{oc} . Both of them depend on the battery state of charge (SOC):

$$\begin{cases} u_b = N_s (V_{oc}(SOC) + \frac{I_b}{N_p} R_b(SOC)) \\ SOC = \frac{1}{C_b} \int_0^t \frac{I_b}{N_p} dt \end{cases} \quad (7)$$

where N_s, N_p are the number of cells in series and in parallel respectively, and the denominator C_b is the battery capacity.

The Powertrain blockset of MATLAB has been used to build a high fidelity vehicle longitudinal model (see Fig. 2). The vehicle, motor and battery characteristic parameters were based on the Renault Zoe Vehicle (see table I). The rest of the parameters are blockset default values. Moreover, both the brake and accelerator pedal actuators are modeled as a first order system with time constants $\tau_a = 0.75s$ and $\tau_b = 1s$ respectively [4]. The controller generates either a braking or an acceleration command at a time. Based on the sign of the control signal, the switching logic sends the command to the braking system or to the engine such that a negative control signal means that the braking is activated, otherwise, acceleration is activated [12].

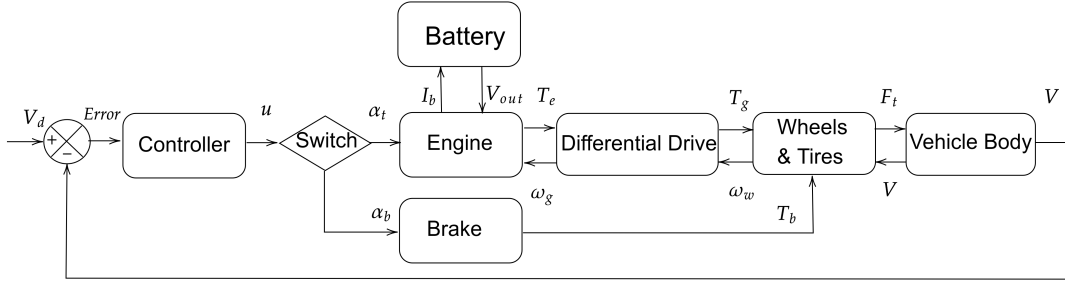


Fig. 2: Model Block Diagram

TABLE I: Simulation parameters.

M	1468kg	C_d	0.29
C_r	0.007	k_g	3.4
A	2.22m ²	R_w	0.329m
ρ	1.225kg/m ³	N_s	96
μ	0.9	g	9.81m/s ²
CG height	0.35m	N_{pads}	2
Front axle to CG	1.455m	η	95%
Rear axle to CG	1.132m	C_b	132Ah
R_m	0.1778m	N_b	2

III. CONTROLLER DESIGN

A. PID Controller

The PID controller is heavily used in the industry due to its simple design and satisfactory performance. The input to the controller is the error signal defined as the difference between the set-point and the system output as in Fig. 6. The output of the controller depends on the PID gains and is governed by the following equation:

$$u(t) = K_p e(t) + K_i \int e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (8)$$

where K_p , K_i and K_d are the proportional, integral and derivative gains and $e(t)$ is the error between set-point R and output Y . The discrete form of the PID has been used in this work where the control law is obtained using the trapezoid method as follows:

$$\begin{cases} \Delta u(k) &= \alpha e(t) + \beta e(k-1) + \gamma e(k-2) \\ \alpha &= K_p + \frac{K_d}{T_s} + \frac{K_i T_s}{2} \\ \beta &= \frac{K_i T_s}{2} - 2 \frac{K_d}{T_s} - K_p \\ \gamma &= \frac{K_d}{T_s} \end{cases} \quad (9)$$

with T_s being the sampling time.

The performance of a PID controller is measured in terms of overshoot O_s , rising time R_t and settling time S_t . It depends on the choice of gains which is not straight forward, there are several tuning techniques such as root locus, gain-phase margin and the widely used Ziegler Nichols [17]. However, these methods are generally limited to linear systems and have poor disturbance rejection. To account for the effects of road angle and wind speed on the controller performance, the designed model is simulated with GA to optimize the gains of the PID controller. The choice of this meta-heuristic optimization method is based on its good performance reported in the literature [9].

B. GA-based PID Optimization

Genetic algorithms are inspired from biological evolution theory which is based on the process of natural selection and genetics [9]. The algorithm uses population sets of multiple potential encoded solutions called chromosomes. Each chromosome is rated by a fitness function (10) that determines how good of a solution it is.

$$\text{Fitness}_{\text{value}} = \frac{1}{\text{Performance}_{\text{index}}} \quad (10)$$

GA algorithms use crossover and mutation operations (Fig. 3) to enhance the existing solutions.

The crossover operation merges existing chromosomes together, while the mutation operation modifies the encoding of existing chromosomes to yield new and better chromosomes. The fitness score of each chromosome is assigned by an objective function and only fit chromosomes are selected to evolve in the next operation. In the context of PID tuning, the objective function minimizes the error that is fed to the controller. The mean of squared error (MSE) has been used as the objective function for a consistent performance comparison between GA-PID and NN-PID.

The GA optimization tool in (Fig. 4) has been used. The process initiates a random population of a certain size containing values (chromosomes) for the PID gains (K_p , K_i and K_d) with which the vehicle longitudinal model is simulated. The fitness value of each chromosome is assessed through the MSE. Fit chromosomes with high fitness scores are selected as fit parents, while unfit chromosomes with low fitness scores are discarded. The GA then performs mutation and crossover on the best fit parents and produces a new population. The number of best fit parents to be enhanced is predefined as a percentage called elite count value. This whole process is repeated over many generations. Moreover, the first selection of population individuals is random but bounded between a lower and an upper limit. The PID gains are optimized with regard to the varying parameters of the model and new optimal PID gains are generated for each case

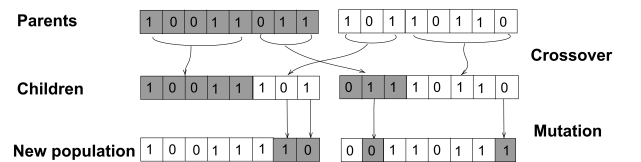


Fig. 3: Genetic operations

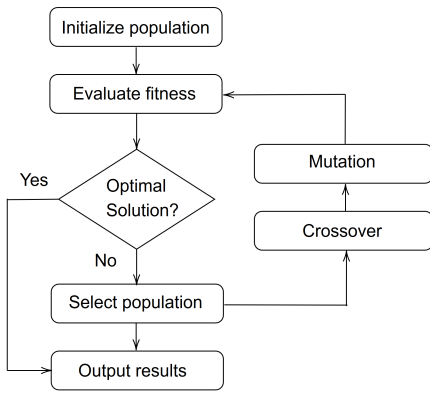


Fig. 4: Genetic algorithm process.

of the varying parameters. A variety of cases were tested and the results were stored in a small data-set. The data-set is then used in a lookup table to select the optimized gains that correspond to the varying parameters. This method allows continuous adaptation of the PID gains as the varying parameters change along a speed profile or drive cycle.

C. Neural Network PID Adaptation

The use of a NN inside the control loop makes it possible to adapt the PID gains online. This method presents a better and more suitable alternative to offline optimization with GA. The NN is trained online using the back-propagation of the error generated from previous PID gains. The architecture of the NN consists of an input layer with four inputs, a hidden layer with four neurons and an output layer with three neurons (Fig. 5). The sigmoid activation function has been used for the hidden layer, while contrary to existing works, instead of the identity, the ReLU function has been used for the output layer to avoid predicting negative gains. The overall system is given in Fig. 6.

The inputs to the hidden layer (S_j) and output layer (Z_j) are given by (11). w_{ij}^h are the weights between input and hidden layer and w_{ij}^o are the weights between hidden and output layer, n_h and n_o represent the number of hidden units and output units respectively. x_i are the inputs to the network, the outputs of the hidden layer h_j and the outputs of the network o_j are given in (12) where f and g are the sigmoid and ReLU activation functions respectively.

$$\begin{cases} S_j = \sum_{i=1}^{n+1} w_{ij}^h x_i, & \text{for } j = 1, \dots, n_h \\ Z_j = \sum_{i=1}^{n_h+1} w_{ij}^o h_i, & \text{for } j = 1, \dots, n_o \end{cases} \quad (11)$$

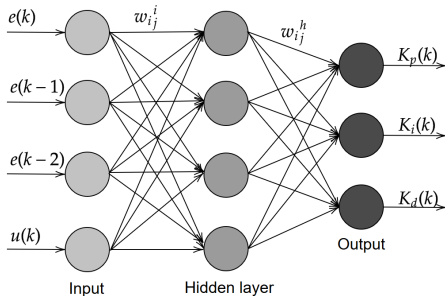


Fig. 5: Neural network architecture.

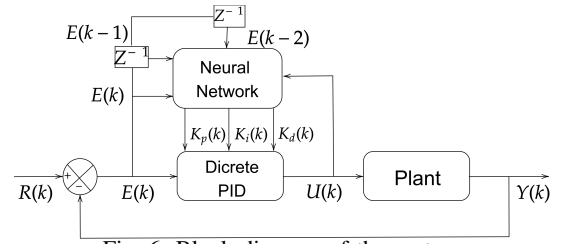


Fig. 6: Block diagram of the system

$$\begin{cases} h_j = f(S_j) \\ o_j = g(Z_j) \end{cases} \quad (12)$$

The gradient descent algorithm minimizes the MSE function $E(k) = \frac{1}{2}e(k)^2$, which is back-propagated through the network using the chain rule [13]. The gradient of the error term is given by:

$$\frac{\delta E(k)}{\delta w_{ij}^h(k)} = \frac{\delta E(k)}{\delta e(k)} \frac{\delta e(k)}{\delta y(k)} \frac{\delta y(k)}{\delta u(k)} \frac{\delta u(k)}{\delta o_j} \frac{\delta o_j}{\delta w_{ij}^h(k)} \quad (13)$$

where

$$\begin{cases} \frac{\delta E(k)}{\delta e(k)} = e(k) \\ \frac{\delta y(k)}{\delta y(k)} = 1 \\ \frac{\delta u(k)}{\delta u(k)} = \text{sign}\left(\frac{Y(k)-Y(k-1)}{U(k)-U(k-1)}\right) \\ \frac{\delta o_j}{\delta o_j} = e(k) - e(k-1) \\ \frac{\delta u(k)}{\delta o_1} = \frac{T_s}{2}(e(k) + e(k-1)) \\ \frac{\delta u(k)}{\delta o_2} = \frac{1}{T_s}(e(k) - 2e(k-1) + e(k-2)) \\ \frac{\delta o_j}{\delta w_{ij}^h} = h_i \end{cases} \quad (14)$$

The term $\frac{\delta y(k)}{\delta u(k)}$ is not accessible in the network, therefore, it is approximated by $\text{sign}\left(\frac{Y(k)-Y(k-1)}{U(k)-U(k-1)}\right)$, which shows whether the control law is improving. To avoid singularities, the expression $\text{sign}((Y(k) - Y(k-1)) \times (U(k) - U(k-1)))$ is used instead. For the hidden layer, the error term is defined as:

$$\frac{\delta E(k)}{\delta w_{ij}^i(k)} = \frac{\delta E(k)}{\delta h_j(k)} \frac{\delta h_j(k)}{\delta S_j(k)} \frac{\delta S_j(k)}{\delta w_{ij}^i(k)} \quad (15)$$

where: $i = 1, \dots, n_i + 1$, and $j = 1, \dots, n_h$.

$$\begin{cases} \frac{\delta E(k)}{\delta h_j(k)} = -e(k) \frac{\delta y(k)}{\delta u(k)} \sum_{i=1}^{n_o} \frac{\delta u(k)}{\delta o_i(k)} w_{ji}^o(k) \\ \frac{\delta h_j(k)}{\delta h_j(k)} = h_j(k)(1 - h_j(k)) \\ \frac{\delta S_j(k)}{\delta S_j(k)} = x_i(k) \end{cases} \quad (16)$$

The weights are updated according to the gradient descent rule as follows:

$$\begin{cases} w_{ij}^h(k+1) = w_{ij}^h(k) - \eta \frac{\delta E(k)}{\delta w_{ij}^h(k)} \\ w_{ij}^o(k+1) = w_{ij}^o(k) - \eta \frac{\delta E(k)}{\delta w_{ij}^o(k)} \end{cases} \quad (17)$$

with η being the learning rate.

IV. SIMULATION RESULTS AND DISCUSSION

The designed PID controllers have been tested with the vehicle longitudinal model given in section II. The GA algorithm is used to optimize the gains of the PID (GA-PID) for three scenarios:

- **Sc1** A varying speed profile without disturbances ($V_w = 0$ and $\theta = 0$).

- **Sc2** A varying speed profile subject to varying wind speed and road slope (see Fig. 7).
- **Sc3** Optimization for different values of V_{ref} , V_w and θ separately.

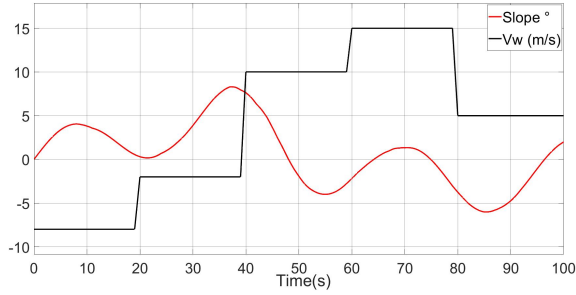


Fig. 7: Wind speed and road slope profiles

Both scenarios 1 and 2 produce a triplet of optimal gains (K_p , K_i and K_d) for the specific speed and disturbance profiles. Scenario 2 shows the effect of the disturbances (v_w and θ) on the controller performance. Scenario 3 is adaptive and more generalized and produces a data-set of optimized gains corresponding to the changing values of V_{ref} , V_w , and θ . The GA-PID is compared to the NN-PID with optimized initial weights and learning rate. Initialization of the weights and choice of the learning rate are very important stages for the performance of the NN-PID and its convergence speed. The learning rate is selected by trial and error after the first few simulations.

The GA optimization for scenarios 1 and 2 results in the triplets $\{K_{p1} = 999.75, K_{i1} = 0.1, K_{d1} = 0.3\}$ and $\{K_{p2} = 383.79, K_{i2} = 0.11, K_{d2} = 128.84\}$ respectively. The corresponding results are compared in Fig. 8-10 where the speed tracking and error variation signals are in (m/s) and the acceleration and braking signals are in percentage (%). For scenario 3, the GA-PID gains are optimized for $V_{ref} \in [0, 30]$ m/s, $\theta \in [-10^\circ, 10^\circ]$ and $V_w \in [-10, 15]$ m/s. The performance of PID adaptation using the generated data-set in a lookup table is compared to that of the NN-PID in Fig. 11-13.

In scenario 1, the GA-PID produces a smoother and faster response with no overshoots due to the absence of disturbances. On the other hand, the inclusion of road slope and wind speed profiles in scenario 2 increases the response time which corresponds to real-life situations, this results in slower tracking with slight overshoots resulting in less smooth responses.

The GA-PID in scenario 3 is smoother than the NN-PID which is more aggressive since the adaptation takes place at each iteration. On the other hand, the NN-PID is more robust to disturbances and faster because it reacts immediately whenever the error magnitude changes and adapts the PID gains to quickly reject the disturbances. Overall, both GA-PID and NN-PID insure adaptive control with very good performance and disturbance rejection.

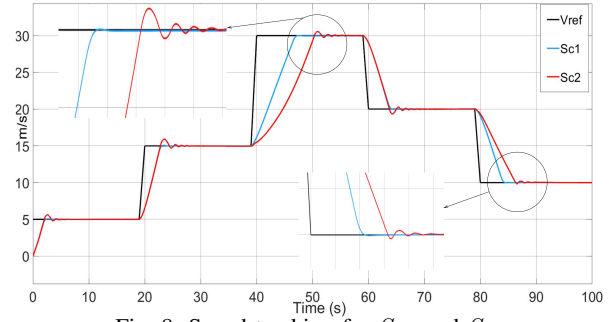


Fig. 8: Speed tracking for Sc_1 and Sc_2

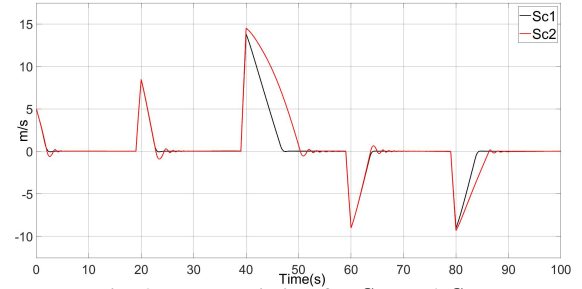


Fig. 9: Error variation for Sc_1 and Sc_2

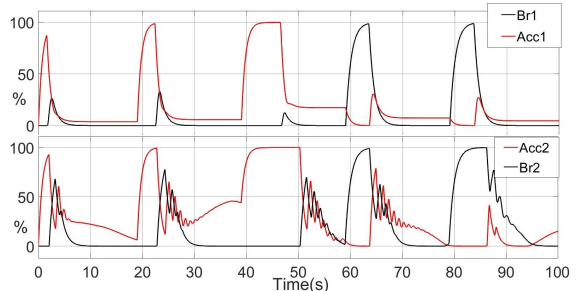


Fig. 10: Acceleration/Brake variation for Sc_1 and Sc_2

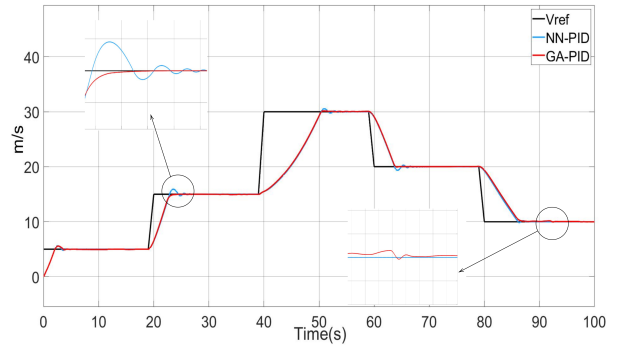


Fig. 11: Speed tracking for Sc_3 vs NN-PID.

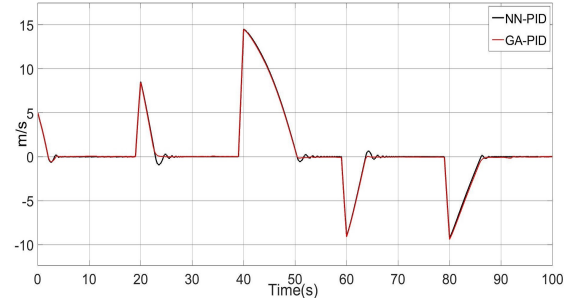


Fig. 12: Error variation for Sc_3 vs NN-PID.

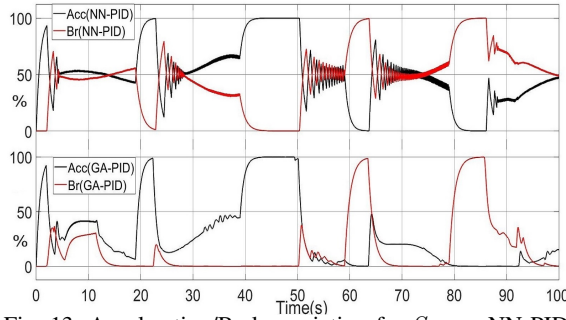


Fig. 13: Acceleration/Brake variation for Sc_3 vs NN-PID.

Table II compares the controller performance regarding the MSE error and evaluates R_t , S_t and O_s for a step response using different values for the hidden layer neurons (h) and learning rate (lr). The GA-PID in scenario 2 has the best performance, while the NN-PID with ($h=10$, $lr=0.01$) is the second best controller.

TABLE II: Controller performance for different h and lr .

Method	MSE	$R_t(s)$	$S_t(s)$	$O_s(\%)$
NN-PID ($lr=0.01$, $h = 4$)	7.596	0.348	0.445	4.6
NN-PID ($lr=0.1$, $h = 4$)	7.605	0.37	0.445	4.59
NN-PID ($lr=0.8$, $h = 4$)	7.604	0.37	0.445	4.59
NN-PID ($lr=0.01$, $h = 10$)	7.555	0.348	0.445	4.6
NN-PID ($lr=0.8$, $h = 15$)	7.597	0.348	0.445	4.59
NN-PID ($lr=0.8$, $h = 20$)	7.557	0.348	0.445	4.6
GA-PID(Sc_1)	7.563	0.87	1.164	6.68
GA-PID(Sc_2)	7.502	0.348	0.444	4.79
GA-PID(Sc_3)	7.534	0.449	0.616	1.52

V. CONCLUSIONS

This paper dealt with the longitudinal control of autonomous vehicles where the vehicle dynamics were modeled using the Powertrain blockset. The adaptive PID controller was designed using two different approaches. The first approach (GA-PID) uses GA where the MSE has been used as a cost function. The advantages of this approach are (a) optimization with a complete vehicle model unlike other approaches which use simplified models in the form of transfer function or state space representation. (b) optimization for a variety of cases and adaptation via a look-up table. The downside of this approach is the heavy calculations required. The second approach (NN-PID) was based on online learning with an MLP NN where the PID gains are adapted with back-propagation. This approach has the following advantages: (a) automatic online PID adaptation and learning capability. (b) adaptation only requires error and control signals. (c) fast instant reaction to error changes at every iteration.

These characteristics make the NN-PID more robust and better at rejecting disturbances. On the other hand, the GA-PID produces optimized PID gains in terms of response speed, precision and overshoot but does not generalize well for different cases of disturbances and set-points. The NN-PID has been found to be more efficient in terms of adaptability and generalizes better. However, its performance significantly depends on weights initialization and choice of learning rate. In general, the GA-PID was smoother and

faster than the NN-PID, while the latter was more adaptive and more robust. Future work should evaluate other optimization algorithms and address the smooth transition between different PID gains as well as explore other controllers.

REFERENCES

- [1] K. Osman, M. F. Rahmat, and M. A. Ahmad, "Modelling and controller design for a cruise control system," Proc. 2009 5th Int. Colloq. Signal Process. Its Appl. CSPA 2009, no. 1, pp. 254–258, 2009.
- [2] A. Simorgh, A. Marashian, and A. Razminia, "Adaptive PID Control Design for Longitudinal Velocity Control of Autonomous Vehicles," Proc. - 2019 6th Int. Conf. Control. Instrum. Autom. ICCIA 2019, pp. 1–6, 2019.
- [3] J. E. A. Dias, G. A. S. Pereira, and R. M. Palhares, "Longitudinal Model Identification and Velocity Control of an Autonomous Car," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 776–786, 2015.
- [4] H. Kim, D. Kim, I. Shu, and K. Yi, "Time-varying parameter adaptive vehicle speed control," IEEE Trans. Veh. Technol., vol. 65, no. 2, pp. 581–588, 2016.
- [5] F. Gao, X. Hu, S. E. Li, K. Li, and Q. Sun, "Distributed Adaptive Sliding Mode Control of Vehicular Platoon with Uncertain Interaction Topology," IEEE Trans. Ind. Electron., vol. 65, no. 8, pp. 6352–6361, 2018.
- [6] S. Xu, H. Peng, Z. Song, K. Chen, and Y. Tang, "Accurate and Smooth Speed Control for an Autonomous Vehicle," IEEE Intell. Veh. Symp. Proc., vol. 2018-June, no. Iv, pp. 1976–1982, 2018.
- [7] F. Walz and S. Hohmann, "Model predictive longitudinal motion control for low velocities on known road profiles," Veh. Syst. Dyn., vol. 58, no. 8, pp. 1310–1328, 2020.
- [8] W. Farag and Z. Saleh, "Tuning of PID track followers for autonomous driving," 2018 Int. Conf. Innov. Intell. Informatics, Comput. Technol. 3ICT 2018, pp. 1–7, 2018.
- [9] A. A. M. Zahir, S. S. N. Alhady, W. A. F. W. Othman, and M. F. Ahmad, "Genetic algorithm optimization of pid controller for brushed DC motor," Lect. Notes Mech. Eng., vol. 0, no. 9789811087875, pp. 427–437, 2018.
- [10] M. K. Debnath, R. Agrawal, S. R. Tripathy, and S. Choudhury, "Artificial neural network tuned PID controller for LFC investigation including distributed generation," Int. J. Numer. Model. Electron. Networks, Devices Fields, vol. 33, no. 5, pp. 1–17, 2020.
- [11] G. Han, W. Fu, W. Wang, and Z. Wu, "The lateral tracking control for the intelligent vehicle based on adaptive PID neural network," Sensors (Switzerland), vol. 17, no. 6, pp. 1–15, 2017.
- [12] L. Nie, J. Guan, C. Lu, H. Zheng, and Z. Yin, "Longitudinal speed control of autonomous vehicle based on a self-adaptive PID of radial basis function neural network," IET Intell. Transp. Syst., vol. 12, no. 6, pp. 485–494, 2018.
- [13] B. Zhao, H. Wang, Q. Li, J. Li, and Y. Zhao, "PID Trajectory Tracking Control of Autonomous Ground Vehicle Based on Genetic Algorithm. Proceedings of the 31st Chinese Control and Decision Conference, CCDC 2019, 3677–3682.
- [14] Z. Sun and G. G. Zhu, Design and control of automotive propulsion systems, 1st Editio. Boca Raton: CRC Press, 2014.
- [15] A. Desrevaux, M. Ruba, A. Bouscayrol, G. M. Sirbu and C. Martis, "Comparisons of Models of Electric Drives for Electric Vehicles," 2019 IEEE Vehicle Power and Propulsion Conference (VPPC), Hanoi, Vietnam, 2019, pp. 1–5.
- [16] Pacejka, H. B. (2008). Vehicle System Dynamics : International Journal of Vehicle Mechanics and Mobility. International Journal of Vehicle Mechanics and Mobility, (August 2012), 37–41.
- [17] A. Q. Ansari, Ibraheem, and S. Katiyar, "Application of ant colony algorithm for calculation and analysis of performance indices for adaptive control system," Proc. Int. Conf. Innov. Appl. Comput. Intell. Power, Energy Control. with Their Impact Humanit. CIPECH 2014, no. November, pp. 466–471, 2014.