INTERNATIONAL JOURNAL OF AUTOMOTIVE AND MECHANICAL ENGINEERING
ISSN: 2229-8649 e-ISSN: 2180-1606
VOL. 20, ISSUE 2, 2023, 10523 – 10535
DOI: https://doi.org/10.15282/ijame.20.2.2023.14.0812

RESEARCH ARTICLE

Particle-Based Optimization Algorithms for Longitudinal Control of Autonomous Vehicle: A Comparative Study

F.A. Ma'ani[1], A.R. Mahdi[1], K.A. Arfian[1] and Y.Y. Nazaruddin[1,2,*]

[1]Department of Engineering Physics, Faculty of Industrial Technology, Institut Teknologi Bandung, 40132 Bandung, Indonesia
[2]Instrumentation and Control Research Group, Faculty of Industrial Technology, Institut Teknologi Bandung, 40132 Bandung, Indonesia

ABSTRACT - In order to improve the stability and performance of an autonomous vehicle, optimization needs to be explicitly performed in the controllers, which has an essential part in the tracking system. This work proposes a novel longitudinal control optimization scheme and a novel longitudinal controller consisting of a feed-forward and feedback term. The feed-forward term is inspired by the vehicle's steady-state response, whereas the feedback term is a proportional-integral-derivative (PID) controller. Also, a model representing the longitudinal vehicle dynamics is designed based on physical phenomena affecting the vehicle. Besides, some nature-inspired optimization algorithms are used to determine the optimal model parameters and optimize the controller parameters, i.e., Particle Swarm Optimization (PSO), Accelerated PSO (APSO), Flower Pollination Algorithm (FPA), and Modified FPA (MFPA). The algorithms are compared in optimizing the longitudinal vehicle model and controller using the CARLA simulator, and stability tests are also done for each algorithm. In addition, the characteristics of several cost functions in controller optimization are inspected. The results show that the MFPA is the most stable algorithm, the proposed model represents the system satisfactorily, and optimizing the controller using a regularized cost function leads to better overall performance. Our code is available in https://github.com/fadamsyah/Particle-Based-Optimization-for-Longitudinal-Control.

ARTICLE HISTORY

Received	:	01st Oct 2021
Revised	:	22nd June 2023
Accepted	:	26th June 2023
Published	:	02nd Aug 2023

KEYWORDS
Longitudinal controller;
Autonomous vehicle;
Particle swarm optimization;
Flower pollination algorithm;
Proportional-integral-derivative

1.0 INTRODUCTION

Autonomous vehicle technology has been improved for quite some time now. Its industry has grown rapidly, which has become the answer to several technological challenges in the Industry 4.0 era. By integrating multi-sensor navigation and positioning system, intelligent decision-making and other technologies can produce a control system that can drive in multiple terrain settings, track probable dangers, and stay stable when there are disturbances. The goal of autonomous vehicle development is to improve traffic safety and driving efficiency. To achieve the self-tracking-ability of an autonomous vehicle, numerous researchers divide the control problem into two sub-categories: longitudinal and lateral control [1–3]. The longitudinal control acts as speed control, whereas the lateral control acts as lane control. This work is limited to longitudinal control for autonomous vehicles.

Proportional-integral-derivative (PID) controllers, one of the most prominent feedback control techniques, are commonly used in autonomous vehicles [1,4,5] because of their simplicity, good control effect, robustness, and ease of implementation. In the vehicle longitudinal control system, the controller is responsible for generating brake and throttle signals by analyzing the given set-point and the actual vehicle speed. [1] implements a proportional-integral (PI) controller to control autonomous vehicle speed for off-road driving and incorporates a saturation mechanism for the integral term in order to avoid windup, while [5] utilizes the standard PID controller to control the vehicle longitudinal system in simulation and real platforms in a low-speed setting. Additionally, [4] proposes an adaptive-PID controller which automatically alters the PID gains to adapt to the changes of the longitudinal system characteristic due to the environment. Yet, [4] uses a strong assumption; the vehicle speed is constant at 20 m/s, resulting in a linear longitudinal vehicle model that may limit their controller span. Unlike the previously mentioned control strategies, our controller consists of a feed-forward as well as feedback term, and we consider a data-driven non-linear vehicle longitudinal model when optimizing our proposed longitudinal controller.

The longitudinal controller must be optimized so that the system can give stable output with a desired response characteristic that fits any given set-point. For PID-based controllers, PID gains are typically tuned to ensure stability and suitable tracking components. Optimization of longitudinal control is challenging due to the inherent non-linearities of the vehicle longitudinal model. These non-linearities limit researchers in exploring optimization techniques, controller equations, and objective functions; [1] and [5] tuned their controller gains manually to achieve stability and considerably good speed tracking performance, while [4] linearized the vehicle longitudinal model and then the controller gains were chosen to meet some specific criteria. None of the mentioned works mathematically formulates aspects of passengers' comfort in their objective functions, e.g., the throttle rate of change and jerk.

*CORRESPONDING AUTHOR | Y.Y. Nazaruddin | ✉ yul@itb.ac.id

PID tuning with particle-based optimization algorithms shows promising results on various occasions [6–9]. A PSO-based PID controller performs better than one tuned with a conventional tuning method, i.e. Ziegler-Nichols, Tyreus-Luyben, and Internal Model Control (IMC), for a third-order linear system [6]. The same observation also emerges in [7], where PSO results in a better-performing PID controller than Ziegler-Nichols method for a composite control system controlling the liquid level of a linear three-tank system. In addition to the superiority of particle-based optimization algorithms for PID tuning, as these algorithms do not depend on gradients to optimize controller parameters, the algorithms allow a wide range of design choices, including incorporating regularization terms and constraints that may introduce non-linearity, singularity, and non-convex optimization problems. de Moura Oliveira et al. [8] conduct an experiment on a first-order linear system with time delay, explore various criteria in the objection function (e.g. integral of square error and integral of absolute error), and incorporates a smoothness term for the control input. Jadhav and Vadirajacharya [9] further introduce hard constraints for the parameters of 2 PID controllers in a system with dual inputs. Thus, inspired by the outstanding capabilities of the particle-based optimization for control systems, this paper explores the optimization scenario of our proposed non-linear longitudinal controller while considering several challenging aspects; a non-linear data-driven vehicle longitudinal model, constrained controller parameters, and regularization terms in the objective function which relate to passengers' comfort.

In this paper, non-linear equations are applied to represent the longitudinal vehicle model by considering physical properties such as drag force and friction force [10]. To determine the parameters in the proposed non-linear model, nature-inspired optimization methods are applied. These optimization methods can find a global optimal solution for a system which is highly non-linear and nonlinearly constrained [11]. Some methods are compared in this paper, i.e. particle swarm optimization (PSO), accelerated particle swarm optimization (APSO), flower pollination algorithm (FPA), and modified flower pollination algorithm (MFPA). Each algorithm has its capabilities and limitations that are discussed by testing the algorithms through several optimization problems which are mainly obtained from the CARLA simulator [1] [12]. There are four contributions delivered in this paper, namely:

i. Propose a longitudinal controller that has a feed-forward and feedback term.
ii. Propose a longitudinal vehicle model representing a real vehicle for simulation.
iii. Compare optimization algorithms performance and stability.
iv. Compare cost function modifications on algorithm performance.

This paper has the following structure. The second section, i.e. longitudinal controller design, discusses the proposed longitudinal controller used in the system design. The third section, i.e. optimization algorithms, reviews the optimization algorithm structure. Further, the longitudinal model optimization is described in the fourth section, i.e. model optimization, whereas the fifth section, i.e. PID controller algorithm, discusses the optimization of the proposed longitudinal controller. The effects of cost function modification on controller tuning will be explained in the sixth section, i.e. cost function modifications. Finally, the last section concludes and presents future works.

2.0 LONGITUDINAL CONTROLLER DESIGN

2.1 Controller Algorithm

The longitudinal controller acts as speed control for an autonomous vehicle. The longitudinal controller proposed in this work has a feed-forward and feedback term. In control systems, the feed-forward term is a technique where the control action is determined based on a predicted or anticipated disturbance or input. On the other hand, feedback control is a technique where the control action is determined based on the difference between the desired output and the actual output of the system. The proposed controller with a feed-forward and feedback term is illustrated in Figure 1. In this work, a PID controller acts as the feedback term that consists of proportional, integral, and derivative parameters, where the feedback term needs to be tuned to obtain the best results from the controller. The throttle value at a steady-state given set-point, r, is selected as the feed-forward term that acts as a performance tracker in the system. Both controller terms are optimized by using several algorithms and the performance of the controllers are measured with cost functions.

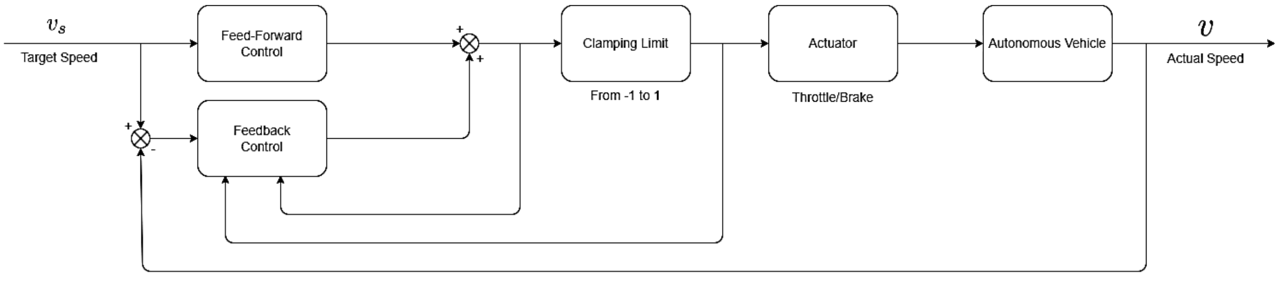


Figure 1. Proposed feed-forward and feedback control

A standard PID controller normally experiences an integral-windup [13]. An integral-windup happens due to an overshoot of accumulated error caused by the nonlinearity of the actuators and the inability to follow the control

[1] Track and simulation map obtained from Introduction to Self-Driving Cars course provided by Coursera

commands perfectly. Generally, the integral-windup phenomenon is often caused by actuator saturation or constraints [14]. Integral-windup can occur in autonomous vehicles when the actuator signals, specifically the throttle and brake, are limited to a range of 0 to 1. This phenomenon arises when the integral part of the controller continues to accumulate an error that exceeds the saturation limit. As a result, the error accumulates beyond the maximum allowed value, leading to integral-windup. The accumulated error can cause overshoot, instability, longer settling times, control saturation, degraded control performance, and increased steady-state error. To compensate for the integral-windup, an anti-windup integrator is applied in the controller, which is also known as integral-clamping [13]. Integral-clamping can help maintain the error accumulation under a saturation limit by terminating the accumulation. This means that the clamping process stops accumulating the error only, not turning off the whole integrator. There are two terms that need to be fulfilled to clamp the integrator:

i. Compare the PID controller output before and after the saturation check. If both values are the same, then the saturation limit has not been surpassed yet, but if the values are different, then the error accumulation has surpassed the saturation limit.
ii. Compare the controller output and error signs. If the signs are both positives or negatives, then the error will increase or decrease from the saturation limit, but if the signs are different, then there is an indication that the error is still located in a safe range from the saturation limit.

From both terms, it can be concluded that an integral-clamping is active when the error accumulation has surpassed the saturation limit and the signs of both the controller output and error are the same. The block diagram of the clamping mechanism is shown in Figure 2.

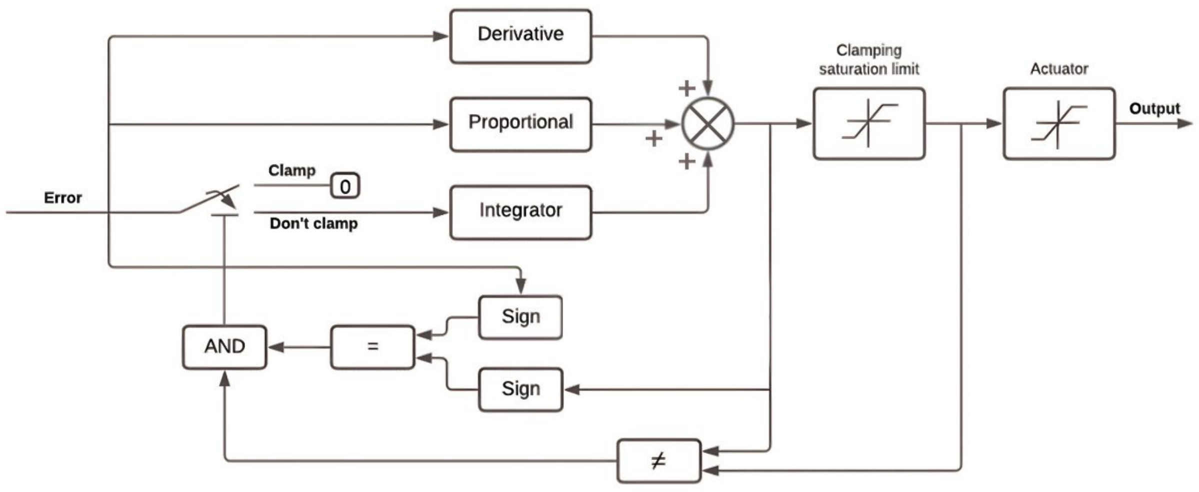


Figure 2. Schematic of integral-clamping strategy

In order to tune the proposed controller, a longitudinal vehicle model should be identified based on simulation data obtained from the CARLA simulator. The feed-forward term utilizes desired speed as input and changes it to the throttle value; hence the steady-state response of the system also needs to be identified. Both models are optimization problems that are discussed in the next section of this paper.

2.2 Steady-State Response

Generally, an autonomous vehicle is represented as a non-linear model [15]. Another aspect that should be recognized from an autonomous vehicle is the throttle-to-speed conversion. If the throttle of the vehicle stays the same for a certain amount of time, the vehicle speed will only reach a saturation point. Therefore there must be a model to define the throttle-to-speed relation of an autonomous vehicle. This work proposes a steady-state response model to represent the throttle-to-speed relation and to create a more concrete longitudinal model of the vehicle. The relation is considered non-linear; hence an exponential function is used to create the model.

The simulated autonomous vehicle in the CARLA simulator accepts an input of throttle that ranges from 0 to 1. There are multiple properties that can be measured from an autonomous vehicle. However, the simplest physical property that can be measured is the speed of the vehicle. To ensure the controller can give an appropriate throttle value given a set-point, a steady-state response model of the system is needed as a converter. The equation used to determine the steady-state response is

$$s_r = \beta_1\left(1 - e^{\beta_2(v_s+\beta_3)}\right) \quad (1)$$

where s_r is the throttle value, v_s is steady-state speed, β_1 is a positive constant, β_2 and β_3 are negative constants. The boundary for each constant is defined as $\beta_1 > 0$, $\beta_2 < 0$, and $\beta_3 < 0$. The β_1 must be positive because it is a scaling factor for the throttle-to-speed relation. The β_2 must be negative in order to create an exponential value of 0-1, whereas β_3 must be negative to represent the minimum throttle threshold needed for a vehicle to start moving.

2.3 System Identification

There are multiple physical properties that need to be defined for the algorithms, such as friction and drag force. Friction happens in a vehicle due to the interaction between the wheels and a road surface that resists vehicle movement. Friction force does not affect the contact surface, but it is proportional to the normal force, N, and affected by the contact surface coefficient of friction (COF), μ.

COF is the ratio between friction force and normal force. It also represents the surface roughness. A stationary vehicle is affected by static friction force, $\mathrm{F} \leq \mu_s \mathrm{N}$, where F is the resulting force of the vehicle and μ_s is the static COF. A moving vehicle is affected by kinetic friction force that is formulated as,

$$\mathrm{F} = \mu_k \mathrm{N} \tag{2}$$

where μ_k is the kinetic COF. The μ_s and μ_k are dimensionless, valued between 0 to 1, where $\mu_s > \mu_k$. The F and N are Newton unit forces.

The drag force that affects a vehicle is related to the aerodynamic properties of the vehicle. There are five types of drag which are, form, lift, surface friction, interference, and internal flow. Form drag is affected by the shape of the vehicle and is represented by the easiness of airflow through the contours of the vehicle. Lift drag is the difference between the upper and lower pressure of the vehicle. Surface friction drag is affected by the viscosity of the air around the vehicle, and it is the number of friction that happens through the layers of air that surround the vehicle. Interference drag is generated by the mixture of airflow around the vehicle, and internal drag happens due to the air that flows into the vehicle. A more detailed description of drag forces can be found in [16]. The percentage of drag forces that are generated from various sources, based on [16], are 55% form, 16% interference, 12% internal, 10% surface friction, and 7% lift.

Based on the physical properties mentioned above, the dynamics equation proposed for the system is

$$\dot{v} = a_1 + a_2 v + a_3 v^2 + b_1 u_{11} + b_2 \exp(b_3 + b_4 u_{12})\, u_{13} + c_1 u_{21} + c_2 \exp(c_3 v + c_4 u_{22}) u_{23} \tag{3}$$

where $\dot{v}$ is the vehicle acceleration, u_{1p} is the throttle with time delay d_{1p}, u_{2p} is the brake with time delay d_{2p}, and a_1, a_2, a_3, b_1, b_2, b_3, b_4, c_1, c_2, c_3, and c_4 are constants. Equation (3) is designed based on the physical properties of the system, hence there are constrains for each parameters which are:

i. a_1, a_2 are constants which represent friction that resists the vehicle movement, so the value must be negative ($a_1, a_2 < 0$). The value is set to 0 when the vehicle is not moving.
ii. a_3 is a constant which represents drag force that resists the vehicle movement, so the value must be negative ($a_3 < 0$).
iii. b_1, b_2 are constants which represent the throttle that moves the vehicle, so the value must be positive ($b_1, b_2 > 0$).
iv. c_1, c_2 are constants which represents brake that resists vehicle movement, so the value must be negative ($c_1, c_2 < 0$).
v. $d_{11}, d_{12}, d_{13}, d_{21}, d_{22}$, and d_{23} are time delay constants that must be positive.

3.0 OPTIMIZATION ALGORITHMS

3.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was developed by Eberhart and Kennedy in 1995 [17] who were inspired by the behavior of bird flocks where social sharing of information occurs and each bird benefits from the discoveries and previous experiences of all other companions while searching for food [18]. The companions, called particles, will move randomly and find the best position individually and the group's best position which is also known as the global best position. Those information are used for updating the value that is searched. The position and velocity update vector is formulated as

$$\mathrm{x}_{k+1}^{i} = \mathrm{x}_{k}^{i} + \mathrm{v}_{k+1}^{i}\Delta \mathrm{t} \tag{4}$$

$$\mathrm{v}_{k+1}^{i} = \mathrm{w}\mathrm{v}_{k}^{i} + \mathrm{c}_1\mathrm{r}_1\left(\mathrm{p}_{k}^{i} - \mathrm{x}_{k}^{i}\right) + \mathrm{c}_2\mathrm{r}_2\left(\mathrm{p}_{k}^{g} - \mathrm{x}_{k}^{i}\right) \tag{5}$$

where x_k^i is the position vector of the i-th particle at the k-th iteration, p_k^i is the individual best position, p_k^g is the global best position, w, c_1, and c_2 are positive constants, r_1 and r_2 are random values that ranges from 0 to 1, and $\Delta\mathrm{t}$ is the time difference.

The value of w, also known as the inertia factor, affects the search speed of the algorithm, whereas c_1 and c_2 are called cognitive learning rates. The c_1 and c_2 affect the individual learning rate and the social or swarm learning rate, respectively. The value of $\Delta\mathrm{t}$ can be considered as 1 that represents the number of iteration done until the new position is updated.

The procedure of the PSO Algorithm:

1. Generate n-particles, n position vector and velocity vector
2. Set the initial cost value for local and global cost as high as possible
3. Set the initial value of $p_k = x_k$
4. Create best global position vector
5. Set the value of w, $\mathrm{c_1}$, and $\mathrm{c_2}$
6. Iterate for each particle and find the local cost value
    a) If the cost has a lower value than the previous local cost, update the local cost and local best position
    b) If the cost has a lower value than the previous global cost, update the global cost and global best position
7. Iterate until max iteration
    a) Update the velocity vector by using Eq. 5
    b) Update the position vector by using Eq. 4
    c) Iterate for each particle and find the local cost value
        i. If the cost has a lower value than the previous local cost, update the local cost and local best position
        ii. If the cost has a lower value than the previous global cost, update the global cost and global best position
8. Use the last known global best position as the output of the algorithm

For a simple case, the inertia factor w can be considered constant, but for a more complex case, the value of w should be decreased linearly. The value of w is usually between 0.4 to 0.9 suggested by Yuhui Shi [19], whereas for a linearly decreasing w, the desired value at the start of an iteration should be $\mathrm{w_{max}}$ and at the end of the iteration, the value decreases to $\mathrm{w_{min}}$. The linearly decreasing value helps in converging the particles. The value of w is determined by

$$\mathrm{w_k = w_{max} - k(w_{max} - w_{min})/T_{max}} \tag{6}$$

where $\mathrm{T_{max}}$ is the maximum iteration. The function above is known as the linearly decreasing weight (LDW), which is introduced by Yuhui Shi [19]. Briefly, the PSO's parameters used in all simulations of this work are $w_{min} = 0.4$, $w_{max} = 0.9$, and $\Delta t = 1$, while T_{max} varies. Besides, c_1 and c_2 are selected based on [18] but $c_1 = 0.7$ and $c_2 = 0.8$ works better for this work.

3.2 Accelerated Particle Swarm Optimization

Accelerated Particle Swarm Optimization (APSO) is an improved method of the original PSO algorithm that reduces the computational time, and it is considerably simple compared to the PSO. APSO only calculates the updated value for the position vector of the global best only. As suggested by Yang in 2008 [20], the APSO only uses random values to increase the diversity instead of individual best as in PSO. The individual best is used when the searched function is non-linear and multimodal [21]. The velocity vector is not present in the APSO algorithm because it is substituted due to an improvement in the algorithm. The position vector update function is changed to

$$\mathrm{x_{k+1}^i = (1-\beta)x_k^i + \alpha\epsilon + \beta p_k^g} \tag{7}$$

where $\mathrm{x_k^i}$ is the position vector of the i-th particle at the k-th iteration, and the $\mathrm{p_k^g}$ is the global best position. The recommended value of $\beta = 0.1 \sim 0.7$ whereas the value of α will decrease with the number of iterations done, and the value of ϵ is a random number from a normal distribution $\mathcal{N}(0,1)$. The value of α is determined by

$$\mathrm{\alpha_k = \alpha_0 L} \tag{8}$$

where α_0 is a constant and L is the range of each variable.

The procedure of the APSO differs from the PSO in the sixth step, where the velocity update is not needed, and the position is updated with Eq.(7). It should be noted that in this work, APSO only updates the next x^i if it generates a better result. Also, the range L is set to the range value of each variable on x_k, so L will vary. Briefly, the APSO's parameters used in all simulations on this work are $\beta = 0.15$, and $\alpha_0 = 0.8$. These parameters are chosen based on [21] but have been tuned by performing several tests to get the best results.

3.3 Flower Pollination Algorithm

Flower Pollination Algorithm (FPA) is an algorithm that is inspired by the pollination behavior of plants; this algorithm was introduced by Yang in 2012 [22]. There are four rules to search for global minima by using the FPA, and it has been explained in [22]. The procedures of the FPA are switch probability, global search, and local search [23]. The first operation, 'switch probability' uses a probability p to decide whether a pollen will be updated using the local or global pollination.

In the second operation, 'global search', the value of $\mathrm{x_k^i}$ that represents the i-th pollen for the k-th iteration will be updated using the following function

$$\mathrm{x_{k+1}^i = x_k^i + \gamma L\big(g_k - x_k^i\big)} \tag{9}$$

where g_k is the global best value in the k-th iteration, L is the strength of pollination or Levy Flights based step size. In the Levy distribution, the value of s is stated as a random value [24] that can be written as

$$s = X/|Y|^{\alpha} \tag{10}$$

here α is a constant, $X \sim \mathcal{N}(0, \sigma^2)$, and $Y \sim \mathcal{N}(0,1)$. The σ^2 is represented by the function as follows

$$\sigma^2 = \left[\frac{\Gamma(1+\alpha)}{\alpha\Gamma(0.5+\alpha/2)} \frac{\sin(\pi\alpha/2)}{2^{(\alpha/2-0.5)}}\right]^{1/\alpha} \tag{11}$$

where Γ represents the standard gamma function. The recommended value for α is 1.5 .

The third operation, 'local search', updates the value based on the function

$$x_{k+1}^i = x_k^i + \epsilon\left(x_k^j - x_k^l\right) \tag{12}$$

where x_k^j and x_k^l are pollen that originated from different flowers but are the same types and ϵ is a value of uniform distribution that ranges from 0 to 1.

Briefly, the FPA parameters used in all simulations on this work are $p = 0.8$, $\alpha = 1.5$, $\gamma = 0.1$, and $\sigma^2 = 0.697$. Parameters p and α based on [22], while γ and σ^2 based on [24].

3.4 Modified Flower Pollination Algorithm

The modified Flower Pollination Algorithm (MFPA) is a variation of the original FPA. MFPA works similarly to the original FPA, but the step size is limited. In this paper, the step size s is limited as in $s \geq s_0$ to increase the convergence rate [25], where s_0 is a positive constant. The rate increases due to the value of $s > 0$, which changes the step vector always towards the global minima. It should be noted that, based on Eq.(9), if the value of $s < 0$, the parameters will tend to avoid the global minima. In this work, the s_0 is set to 0.1 across all simulations based on [25].

4.0 MODEL OPTIMIZATION

4.1 Steady-State Response

To find the steady-state response, throttle versus speed data is obtained from the CARLA simulator. The optimization algorithms are utilized to find the best values for β_1, β_2, and β_3 based on Eq.(1). The cost function used to find the best solution is

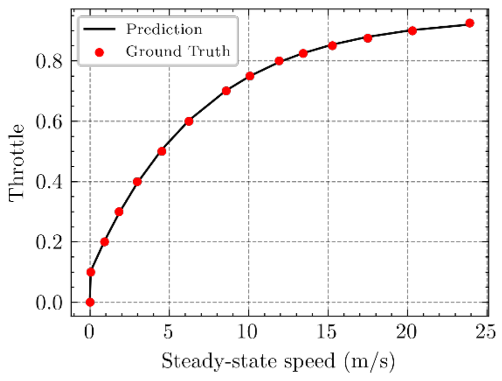
$$J = \frac{1}{M}\sum_{k=1}^{M}(y_k - \hat{y}_k)^2 \tag{13}$$

where J is the cost function, y is the true value, $\hat{y}$ is the predicted value, and M is the maximum simulation step. Equation (13) is also known as the mean squared error (MSE) function. Commonly, the MSE function is utilized in order to check the accuracy of a prediction in comparison to the true value. In this work, by utilizing the MSE function, the algorithms can find the best function parameter that results in the best-predicted data. The hyperparameter setting for the algorithms are 25 population and 5000 iterations. The computation process is performed using Ryzen 5 3600, running at 3.6 GHz. The results of the optimization from each algorithm are shown in Figure 3 (a) and in Table 1.

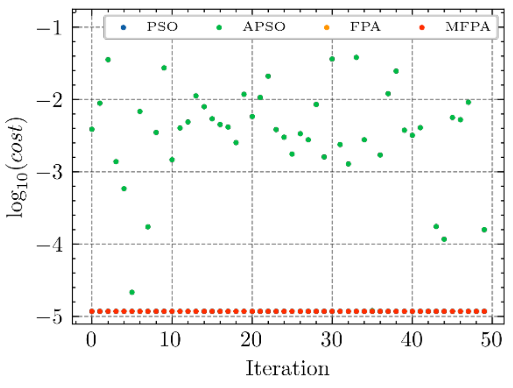
From Table 1, it can be seen that the algorithms minimum loss value are the same. It can be observed from Figure 3(a) that the algorithms can fit the simulation data accurately. Another parameter that is compared is the stability by running each algorithm 50 times with the same hyperparameters. The resulting losses obtained from the iteration is shown in Figure 3(b) and Table 2. The algorithms are defined as stable when the standard deviation (STD) of the loss obtained is near or equal to 0.

Table 1. Steady-state optimization comparison

Algorithm	Minimum Cost	β_1	β_2	β_3
PSO	0.000012	0.8501	-0.1450	0.0962
APSO	0.000012	0.8501	-0.1450	0.0962
FPA	0.000012	0.8501	-0.1450	0.0962
MFPA	0.000012	0.8501	-0.1450	0.0962



(a)



(b)

Figure 3. (a) Steady-state optimization result, and (b) steady-state stability test for PSO, APSO, FPA, and MFPA

Table 2. Steady-state stability comparison

Algorithm	Min.	Max.	Mean	STD
PSO	0.000012	0.000012	0.000012	3.4844×10^{-20}
APSO	0.000012	0.038092	0.007186	9.4475×10^{-3}
FPA	0.000012	0.000012	0.000012	2.4271×10^{-20}
MFPA	0.000012	0.000012	0.000012	2.5949×10^{-20}

4.2 System Identification

Based on Eq.(3), the algorithms finds the best model to represent the dynamics of the simulated vehicle in the CARLA simulator. The ground-truth data for the optimization is sampled at a rate of 50 Hz. The algorithm optimizes the MSE of the ground truth data compared with Eq.(3) so the resulted model will be able to represent the actual system. There are a total of 24 dataset used to identify the system. The optimization is carried out by using the same hyperparameters as the steady-state optimization. The algorithms use 50 particles and 10000 iterations. The results are then tested to find the accuracy value by using the following relation.

$$\text{acc} = 100\left(1 - \frac{\|\text{v} - \hat{\text{v}}\|}{\|\text{v} - \bar{\text{v}}\|}\right)\% \tag{14}$$

where v is the ground truth speed, $\hat{v}$ is predicted speed, and $\bar{v}$ is the mean of the ground truth speed.

Table 3. System identification stability comparison

Algorithm	Min.	Max.	Mean	STD
PSO	0.2162	39.8521	5.8126	9.9319
APSO	0.5765	31.4719	5.4365	8.0996
FPA	0.0872	0.8551	0.2338	0.1659
MFPA	0.0656	0.1201	0.0701	0.0119

Stability tests are also done where each algorithm is run 20 times. The results of the stability test are shown on Figure 4(a). The minimum cost value for the ground truth data prediction by each algorithm is shown in table Table 3. Based on the loss value, it can be seen that the APSO is the least accurate algorithm, and the MFPA is the most accurate algorithm. The prediction results for the MFPA-optimized longitudinal model are shown in Table 4. The Maximum Absolute Error (MAER) is also considered in the table.

Table 4. MFPA training results

Data index	Accuracy (%)	MSE (m^2/s^2)	MAER (m/s)	Data index	Accuracy (%)	MSE (m^2/s^2)	MAER (m/s)
1	90.41	0.0104	0.4944	13	76.63	0.0091	0.1724
2	93.42	0.0520	0.9492	14	86.19	0.0071	0.2261
3	94.89	0.0583	0.6929	15	77.77	0.0466	0.3375
4	89.84	0.0343	0.3446	16	84.36	0.0417	0.4246
5	87.88	0.1085	0.8189	17	91.54	0.0218	0.3554
6	94.75	0.0723	1.0087	18	91.73	0.0296	0.2586
7	96.04	0.0491	1.0384	19	83.79	0.1360	0.4555
8	85.61	0.0824	0.7002	20	90.45	0.0501	0.2700
9	77.48	0.0386	0.2530	21	96.70	0.0107	0.2291
10	90.56	0.0308	0.4325	22	91.70	0.0823	0.3437
11	95.36	0.0233	0.7653	23	88.83	0.1980	0.5112
12	95.85	0.0350	1.0363	24	92.87	0.1375	0.6962

Table 5. Test data results

Result	Data set 1	Data set 2
Accuracy	0.9390	0.9368
MSE	0.0311	0.0620
MAER	0.4127	0.4707

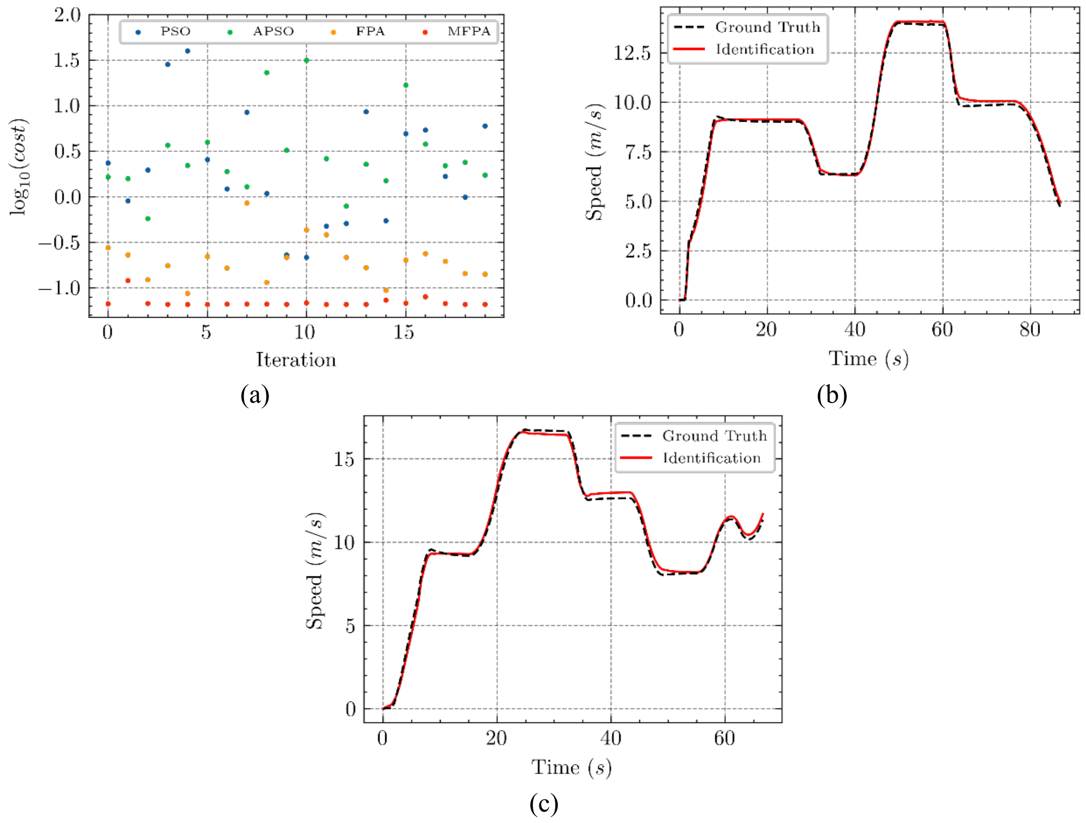


Figure 4. (a) System identification stability test for PSO, APSO, FPA, and MFPA, (b) first data set test result, and (c) second data set test result

Based on the results of optimization performed by MFPA, the model is then tested by using two sets of data. The test data set has a different throttle profile than the training data. The first data set is throttle only data, whereas the second data set is throttle data based on a PID controller output. The results are shown in Figure 4(b), Figure 4(c), and Table 5. It can be seen that the model does not overfit the training data.

4.3 Analysis

The two systems that are optimized have their own characteristics based on the optimization results. The difference is based on the number of parameters that each system needs to be searched. It can be seen that based on Table 2 and Table 3, the mean and STD results differ greatly for both systems. The system identification has a high mean and STD, thus showing that the search space is complex and makes it harder for the algorithms to find the global minima. On the other hand, the steady-state response has a much lower mean and STD. This shows that the search space of the steady-state response is simpler compared to the system identification.

Considering the search space complexity, it can be represented as a performance measure for the algorithms. The PSO, APSO, FPA and MFPA can represent the data well, although the PSO and APSO sometimes are stuck in best local values, also known as local minima. Subsequently, for the system identification, there are 24 sets of training data. The model can be represented by each algorithm very well and with high accuracy. However, only the MFPA can find the global minima with stable output, it can be seen by its low-cost value and low STD as well. There are two sets of test data, where the throttle input for the test and training data are different. The results acquired from the test data show that the model does not overfit the training data as it still can perform well on inputs having different characteristics.

5.0 PID CONTROLLER ALGORITHM

5.1 Controller Tuning

The PID controller applied in the CARLA simulator is equipped with a feed-forward and feedback term with an added integral-clamping mechanism. In discrete-time, the longitudinal controller can be represented as follows

$$e_k = r_k - v_k \tag{15}$$

$$u_k = s_{r,k} + k_p e_k + k_i E_k + k_d \frac{e_k - e_{k-1}}{\Delta T} \tag{16}$$

where $(\cdot)_k$ is a variable at time-step k, e is the difference between the set-point r and actual speed v, u is the control signal, ΔT is the sampling time, s_r is the value throttle at set-point in steady-state condition, E is the integral result by using the integral-clamping method, k_p is the proportional gain, k_i is the integral gain, and k_d is the derivative gain.

The optimum value for k_p, k_i, and k_d are acquired by minimizing the following function

$$\begin{aligned} \underset{k_p,\,k_i,\,k_d}{\text{minimize}} \quad & J = \frac{1}{M}\sum_{k=1}^{M}(e_k)^2 \\ \text{subject to} \quad & k_p \geq 0, \\ & k_i \geq 0, \\ & k_d \geq 0. \end{aligned} \tag{17}$$

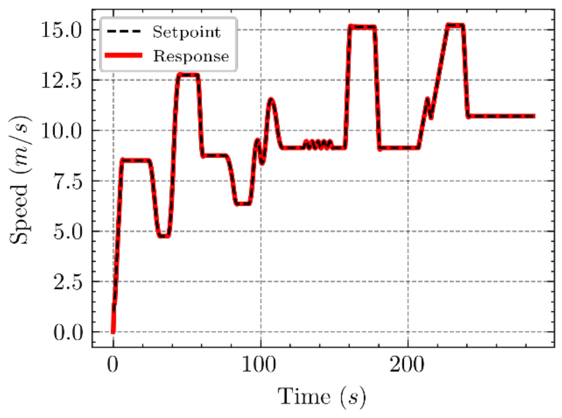
where J is the cost function, and M is the value of the number of simulation step. Equation (17) is utilized so that the algorithms can find the best combination of controller gains that generates the least value of mean squared of error.

The hyperparameter settings for the algorithms are 50 population and 5000 iteration. The loss obtained is shown in Table 7, whereas the lowest-loss parameters are $k_p = 0.9120$, $k_i = 1.5813$, and $k_d = 0.0329$. The simulation is then carried out by using the PID parameters obtained. The speed of the vehicle is used to determine the success rate of the optimization. The result is shown in Figure 5 (a) and Table 6.

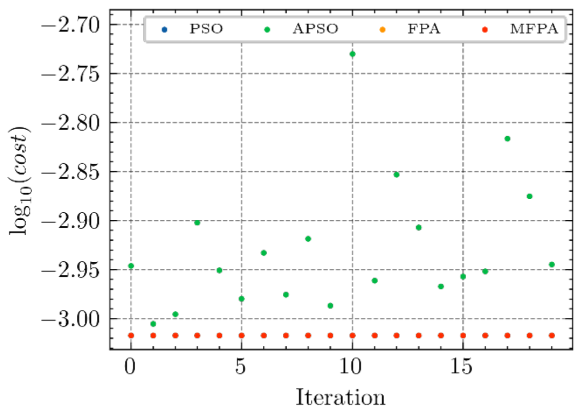
Table 6. PID tuning stability comparison

Algorithm	Min.	Max.	Mean	STD
PSO	0.000961	0.000961	0.000961	2.0807×10^{-18}
APSO	0.000988	0.001861	0.001195	2.0215×10^{-4}
FPA	0.000961	0.000961	0.000961	1.6656×10^{-18}
MFPA	0.000961	0.000961	0.000961	1.4834×10^{-18}

Stability tests are also performed on the PID tuning. The algorithms are run 20 times each. The result is shown in Figure 5(b). The controller is then implemented in the CARLA simulator, the data are sampled at 50 Hz, and the results are demonstrated in Figure 6.

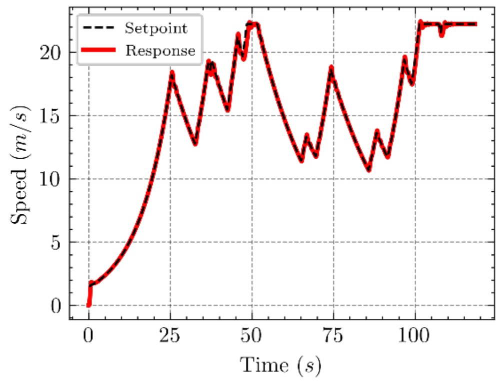


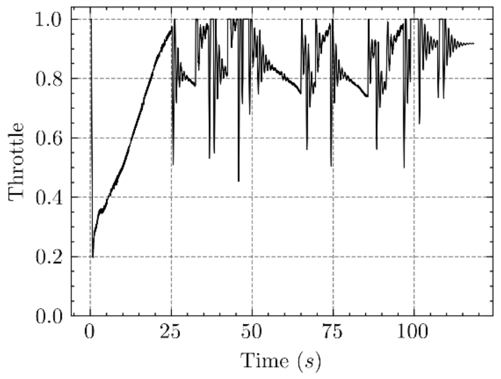
(a)



(b)

Figure 5. (a) Controller response of the lowest MSE, and (b) PID tuning stability test for PSO, APSO, FPA, and MFPA





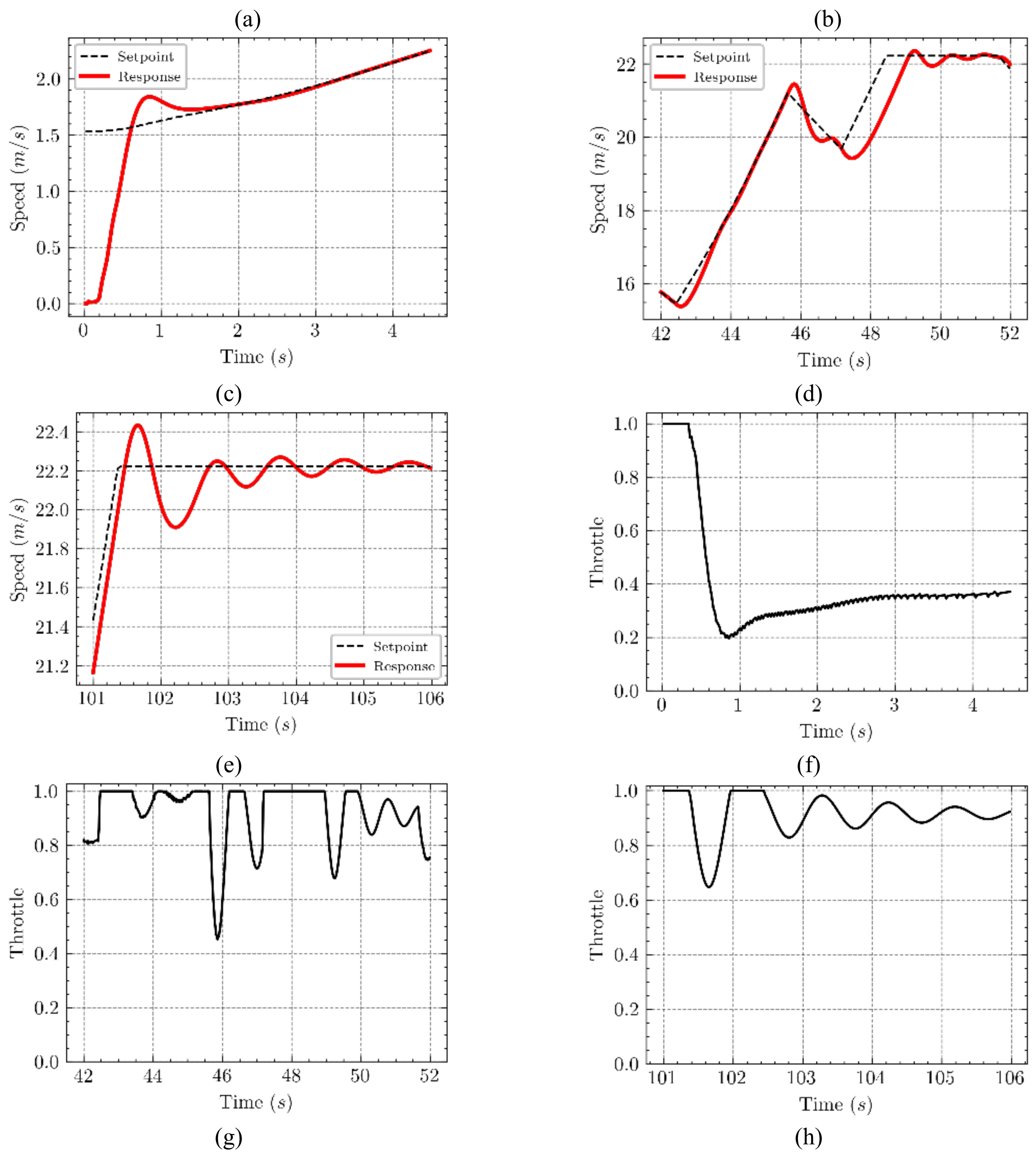


Figure 6. CARLA simulation results using the optimized controller

5.2 Analysis

The proposed PID controller uses the integral-clamping method. The cost function utilized is only the MSE. The set-point for the PID tuning has a different value than the set-point for the CARLA simulation in order to check whether the controller can adapt to varying conditions with different set-points or not. Simulation shows that the optimized controller is able to perform well in the simulation. Based on Figure 6, it can be seen that the results are oscillatory; hence there need to be improvements in the optimization process which is discussed in the next section of this paper.

6.0 COST FUNCTION MODIFICATIONS

6.1 Regularization Strategy

The optimized controller in the previous section produces highly oscillatory outputs with jittery control signals. This profile may lead to inconvenience to passengers. The profile of the control signal is also oscillatory so that the actuators could break quickly. This phenomenon shows that only using MSE in the objective function is insufficient for generating an appropriate controller. Therefore, to improve the overall performance of the controller, modifications of the cost function are investigated. The cost function is modified by giving an additional term as a regularization factor. There are three regularizations investigated in this work, namely:

i. Sum-squared inputs regularization
ii. Sum-squared derivative of inputs regularization
iii. Sum-absolute derivative of inputs regularization

The first regularization is inspired by the cost utilized in the Linear-Quadratic Regulator (LQR) method. In contrast, the second and third regularization is designed to create a smoother and lower-oscillatory control signal output. The mathematical notation of the modification is given as follow.

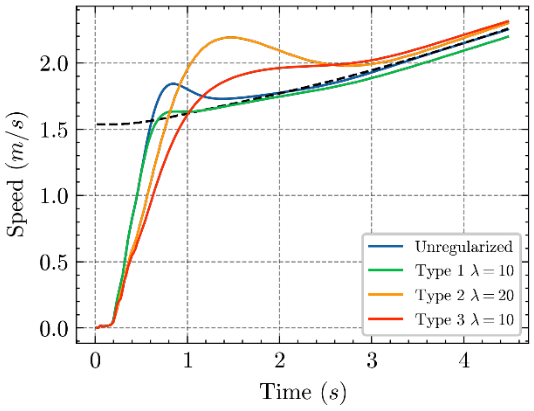
$$J = \frac{1}{M}\sum_{k=1}^{M}\left[(e_k)^2 + \lambda\, g(s_k, \dot{s}_k)\right] \tag{18}$$

where λ is a trade-off constant and g(·) is a regularization function.

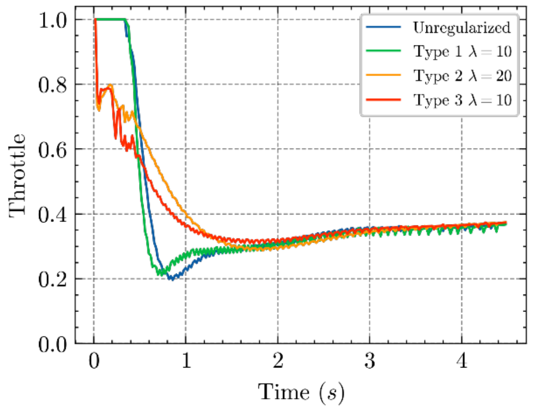
The optimization algorithm utilized in this case is only the algorithm with the best steady-state-response, system identification, and PID results, that is, the MFPA. The results of optimized controllers using the regularized cost functions are shown in Table 7 and Figure 7.

Table 7. Cost function comparison

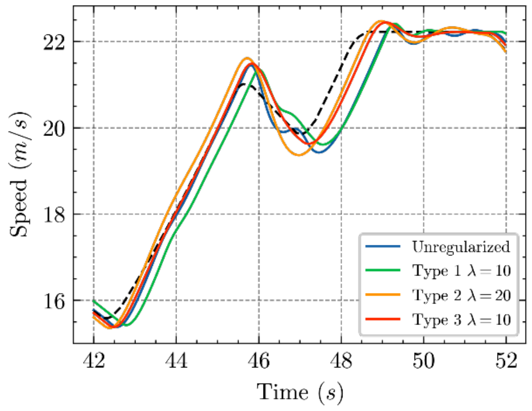
Type	λ	k_p	k_i	k_d	MSE	
					Training	Test
Unregularized	-	0.9120	1.5813	0.0329	0.000961	0.0434
1	1	0.8683	1.3099	0.0349	0.000986	0.0409
	10	1.1481	0.0036	0.0570	0.057035	0.0268
	30	0.4715	0.0005	0.0583	0.423748	0.0358
2	1	0.3975	0.7738	0.0273	0.001931	0.0479
	20	0.2962	0.3864	0.0224	0.004343	0.0512
	40	0.2795	0.2858	0.0208	0.006198	0.0546
3	1	0.4113	0.6298	0.0378	0.002165	0.0377
	10	0.3258	0.1593	0.0331	0.009162	0.0434
	20	0.2944	0.0872	0.0315	0.016824	0.0481



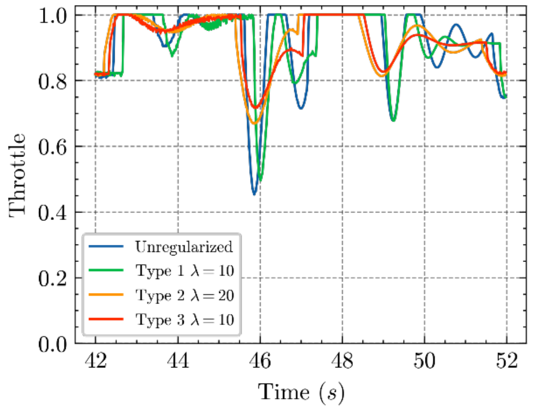
(a)



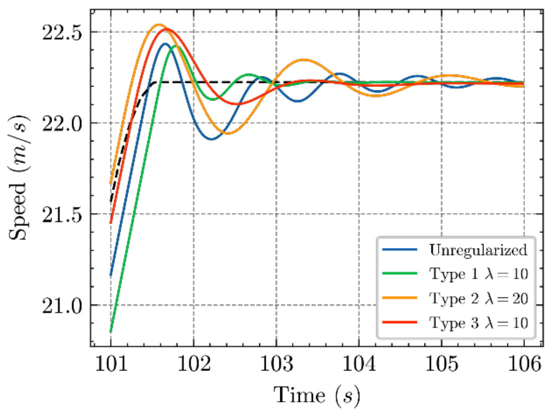
(b)



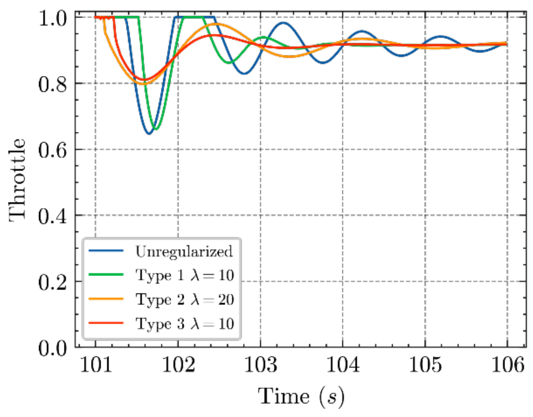
(c)



(d)



(e)



(f)

Figure 7. Comparison of the optimized controllers performance. Black-dotted line denotes the speed set-point. All of the proposed regularization factors in cost function effectively lower the oscillation of the resulting vehicle speed

6.2 Analysis

In terms of smoothness and oscillatoriness, the regularized cost functions give better responses than the unregularized ones. In addition, there are certain behaviors that can be observed from each cost modification which are:

i. The 1st cost function: When the λ increases, the value of $\mathrm{k_p}$ and $\mathrm{k_i}$ decrease whereas $\mathrm{k_d}$ increases. It is also worth noting that the k_i decreases extremely toward zero.
ii. The 2nd cost function: When λ increases, the value of $\mathrm{k_p}$, $\mathrm{k_i}$, and $\mathrm{k_d}$ decrease with $\mathrm{k_i} > \mathrm{k_p}$.
iii. The 3rd cost function: When λ increases, the value of $\mathrm{k_p}$, $\mathrm{k_i}$, and $\mathrm{k_d}$ decreases. Here, the value of $\mathrm{k_i}$ is relatively smaller than the value resulted from the 2nd cost function.

When implemented in the CARLA simulator, it can be seen that based on Table 7, the lowest MSE is obtained from the controller optimized using the 1st cost function with $\lambda = 10$. However, the overall performance is rather not good because the value of $\mathrm{k_p}$ is too large, hence creating an extremely sensitive system. Moreover, the value of k_i is too small, creating a bad tracking performance, especially if the feed-forward term is inaccurate. On the other hand, using the 2nd cost function, controllers produce more oscillation on the output compared to controllers optimized using the 3rd cost function because of the more considerable $\mathrm{k_i}$ value. Considering the trade-off between the MSE, control signal profile, and output profile, it can be concluded that the best PID parameter is obtained by the third cost function with a λ value of 10. The reason is that because the $\mathrm{k_p}$ is not too small, so the response is sensitive enough, and ki is not too large; hence the system does not oscillate and still has a good tracking ability.

Note that in this work, the feedback controller is coupled with the feed-forward term. Consequently, it may give a different result and conclusion for different kinds of use cases. Nevertheless, the most exciting thing is that the regularized cost function can produce controllers performing lower MSE values in the test set where the MSE value of the system with a controller optimized using unregularized cost function is $0.0434\ \mathrm{m/s}$.

7.0 CONCLUSIONS

The proposed longitudinal controller in this paper is able to adapt to varying set-points, and the proposed vehicle model can nicely represent the vehicle longitudinal dynamics by considering physical phenomena affecting vehicles. Also, the combination of the feed-forward and feedback term in the controller is able to give a smooth control response. Furthermore, the comparison explored in this paper shows the capability and limitation of the PSO, APSO, FPA, and MFPA optimization algorithms. The algorithms work well for steady-state response optimization, so it can be concluded that they can optimize simple search spaces. For a complex search space such as system identification, MFPA is the only algorithm that can generate stable performance and find an optimal solution. It can be clearly seen based on the cost value and the STD of the MFPA. In the case of PID tuning, all algorithms are able to find an optimal solution except for the APSO. Moreover, this work shows that controllers tuned with the regularized cost functions can perform better than the unregularized ones.

There are various possibilities of hyperparameters that can be tuned to improve the performance of the algorithms. Additional tests can be done for the algorithms such as changing the particle number or other optimization parameters to get a more definite comparison. There are more efficient forms of the algorithms that are tested in this paper that can be used to obtain better results. Another factor that can be used to improve the performance of an optimization algorithm is by modifying the cost function used.

8.0 ACKNOWLEDGEMENT

This research is partially funded by the Indonesian Ministry of Education, Culture, Research, and Technology under WCU Program managed by Institut Teknologi Bandung and in part by Institut Teknologi Bandung Research Program.

9.0 REFERENCES

[1] G.M. Hoffmann, C.J. Tomlin, M. Montemerlo, and S.Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," In Proceedings of the American Control Conference, 2007, pp.2296-2301.

[2] C.M. Filho, D.F. Wolf, V. Grassi, and F.S. Osorio, "Longitudinal and lateral control for autonomous ground vehicles," *IEEE Intelligent Vehicles Symposium,* 2014, pp.588-593.

[3] R. Attia, R. Orjuela, and M. Basset, "Combined longitudinal and lateral control for automated vehicle guidance," *Vehicle System Dynamics,* vol. 52, pp. 261-279, 2014

[4] P. Zhao, J. Chen, Y. Song, X. Tao, T. Xu, and T. Mei, "Design of a control system for an autonomous vehicle based on adaptive-PID," *International Journal of Advanced Robotic Systems*, vol. 9, p. 44, 2012.

[5] M. Marcano, JA. Matute, R. Lattarulo, E. Martí, and J. Pérez, "Low speed longitudinal control algorithms for automated vehicles in simulation and real platform," *Complexity*, vol. 2018, pp. 1-12, 2018.

[6] H.M. Asifa, and S.R. Vaishnav, "Particle swarm optimization algorithm based PID controller," In: 3rd International Conference on Emerging Trends in Engineering and Technology, 2010, pp. 628–631.

[7] M.I. Solihin, L.F. Tack, and M.L. Kean, "Tuning of PID controller using particle swarm optimization (PSO)," In Proceeding of the International Conference on Advanced Science, Engineering and Information Technology, 2011, pp. 458-461.

[8] P.B. de Moura Oliveira, J.D. Hedengren, and E.J. Solteiro Pires, "Swarm-based design of proportional integral and derivative controllers using a compromise cost function: An arduino temperature laboratory case study," *Algorithms*, vol. 13, pp. 315-332, 2020.

[9] A.M. Jadhav, and K. Vadirajacharya, "Performance verification of PID controller in an interconnected power system using particle swarm optimization," *Energy Procedia*, vol. 14, pp. 2075-2080, 2012.

[10] F. Ahmad, SA. Mazlan, H. Zamzuri, H. Jamaluddin, K. Hudha, and M. Short, "Modelling and validation of the vehicle longitudinal model," *International Journal of Automotive and Mechanical Engineering*, vol. 10, pp. 2042-2056, 2014

[11] A.R. Conn, K. Scheinberg, and L.N. Vicente, *Introduction to derivative-free optimization.* vol. 8, Philadelphia, USA: SIAM, 2009.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, "CARLA: An open urban driving simulator," In Proceedings of the 1st Annual Conference on Robot Learning, 2017, pp. 1–16.

[13] S. Kumar, and R. Negi, "A comparative study of PID tuning methods using anti-windup controller," In ICPCES 2012 - 2012 2nd International Conference on Power, Control and Embedded Systems, 2012, pp. 1-4.

[14] L. Angel, J. Viola, and M. Paez, "Evaluation of the windup effect in a practical PID controller for the speed control of a DC-motor system," presented in 2019 IEEE 4th Colombian Conference on Automatic Control (CCAC), Medellin, Colombia, 2019.

[15] Automated Highway System (AHS) System Objectives and Characteristics, National Automated Highway System Consortium, Troy, MI, 1995.

[16] B. Parker, *The Isaac Newton school of driving: physics and your car*. Baltimore, USA: Johns Hopkins University Press, 2003.

[17] J. Kennedy, and R. Eberhart, "Particle swarm optimization," In IEEE International Conference on Neural Networks - Conference Proceeding, 1995, pp.1942-1948.

[18] K.E. Parsopóulos, and M.N. Vrahatis, "Particle swarm optimization method in multiobjective problems," In Proceedings of the ACM Symposium on Applied Computing, 2002, pp.603-607.

[19] Y. He, W.J. Ma, and J.P. Zhang, "The parameters selection of PSO algorithm influencing on performance of fault diagnosis," In MATEC Web of Conferences, 2016, vol. 63.

[20] A.H. Gandomi, G.J. Yun, X.S. Yang, and S. Talatahari, "Chaos-enhanced accelerated particle swarm optimization," *Communications in Non-linear Science and Numerical Simulation*, vol. 18, pp. 327-340, 2013.

[21] X.S. Yang, S. Deb, and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," *Communications in Computer and Information Science*, vol.136, pp.53-66, 2011.

[22] X.S. Yang, "Flower pollination algorithm for global optimization," Lecture Notes in Computer Science (including including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics), vol. 7495, pp. 240-249, 2012.

[23] Z.A.A. Alyasseri, A.T. Khader, M.A. Al-Betar, M.A. Awadallah, and X.S. Yang, "Variants of the flower pollination algorithm: A review," *Studies in Computational Intelligence,* vol. 749, pp. 91-118, 2018.

[24] R.N. Mantegna, "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes," *Physical Review E*, vol. 49, pp. 4677-4683, 1994.

[25] F.A. Ma'ani, and Y.Y. Nazaruddin, Optimization of longitudinal control of an autonomous vehicle using flower pollination algorithm based on data-driven approach," *International Journal of Sustainable Transportation Technology,* vol. 3, pp. 58-65, 2020.