# Performance of a Wall-Following Robot Controlled by a PID-BA using Bat Algorithm Approach

Heru Suwoyo
Department of Electrical Engineering, Universitas Mercu Buana, Indonesia

Ferryawan Harris Kristanto
Department of Electrical Engineering, Universitas Mercu Buana, Indonesia

Abstract: A wall-following robot needs a controller that applies the closed-loop concept to move actively without hindrance. Some controllers with good capabilities can act as controllers for wall follower robots, such as PID controllers. Conceptually, this controller's good performance depends on tuning the three gains before use. Instead of giving the expected and appropriate output, wrong settings will provide inaccuracies for the controller, so applying the manual method at the tuning stage is not recommended. For this reason, PID controllers are often implemented in a system supported by appropriate optimization methods, such as Genetic Algorithm or Particle Swarm Optimization. Furthermore, different from this, in this study, the Bath Algorithm is used as an alternative optimization algorithm. Its application begins with a realistic simulation of a wall-following robot. This is done to provide the possibility to implement online PID controllers and BAs. In the end, several methods are compared to find out the performance of this type of approach. Moreover, based on the observed comparative results, the proposed method gives a better value in accumulative error and convergence speed in the PID optimization process.

Keywords: Wall Following Robot, PID Controller, Bat Algorithm

## Introduction

Wall-following robot (WFR) is a robot that has become popular in many fields of applications such as research, development, and competition. The main errand of WFR is to move by following the boundary of its determined arena. Based on the wall following principle, it follows an object, namely a wall (Iqbal & Aji, 2021). A WFR has to be capable of preserving proper distances. Thus, the robot will be close to the wall in the migration state. Maintaining a safe distance from the wall can be carried out by using an installed distance-ranging sensor. The distance information acquired by the sensor will then be processed to produce the desired output with the involvement of a specific controller (Suwoyo et al., 2018). The standard controller used in robotics is the PID controller. With the deployment of PID control, WFR can gain stability in its movement (Lee et al., 2018; Tzafestas, 2018). System stability carried by PID controller relies on its three key parameters (Suwoyo et al., 2018). However, setting the value of these three fundamental parameters might take much work.

Key parameters of PID can be obtained through various ways such as trial and error, rule-based (classical), and optimization. The trial and error method obviously will not be able to give the optimal values since it works only through the user's approximation. The most known classical method is Ziegler-Nichols (ZN), followed by CC, Kappa-Tau, and Lambda, which have a disadvantage in that these methods need further fine-tuning (Fišer & Z'itek, 2019; Mazlan et al., 2020; Sekarsari & Tata, 2021; Suksawat & Kaewpradit, 2021). Nevertheless, meta-heuristic optimization methods can provide a solution to this issue. Meta-heuristic approaches such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) were introduced and implemented in PID-controller optimization and could produce optimum outcomes (Adriansyah et al., 2019; M Zahir et al., 2018; Raza et al., 2018).

However, GA has an issue that could not deliver convergence in its computation process and PID-PSO (Adriansyah et al., 2019). Many algorithms were invented, and a new algorithm could outperform the previous one. Later it was discovered that a meta-heuristic algorithm inspired by Bat's echolocation behaviour could deliver local optimums respectively with fast convergence rates. Although the PSO algorithm has the power to find a global minimum, its society has a slow rate of convergence in finding from optimal solution (Gagnon et al., 2020).

This paper proposed to analyse the Bat Algorithm (BA) implementation, inspired by the echolocation behaviour of bats, to find the optimum values for the PID controller. At first, manual tuning will be performed to find the upper and lower bounds of Ki, Kp, and Kd. Then the acquired values range will be used for the optimization process. Hypothetically, PID-BA will deliver better convergence and local optimum performance than the PID-PSO approach. Therefore, the performance of a wall-following robot based on PID control can be enhanced using Bat Algorithm.

The hypothesis above can be justified through the dynamic error values given by the system. RMSE is a standard metric used in model evaluation. As its name implies, the RMSE is the square root of the mean squared error. RMSE has been used to ass's model performance for many years (Hodson, 2022). The fitness of the parameter's value given by Bat Algorithm optimization will be calculated according to this dynamic error approach. If the RMSE values given are lower than the PSO approach, then the hypothesis, as explained earlier, is proven to be valid.

## Research Method

PID is an essential approach for controller application and has been widely used because it is convenient and has high reliability to regulate process variables (Sekarsari & Tata, 2021). PID is the abbreviation of its three constituent components, P as proportional, I as integral, and D as derivative. In this research, the design of the system WFR-PID controller is simulated through MATLAB due to the possession of the actual hardware. The simulation system consists of a modelling sensor and PID control; thus, the robot sim will be capable of doing the wall-following task.

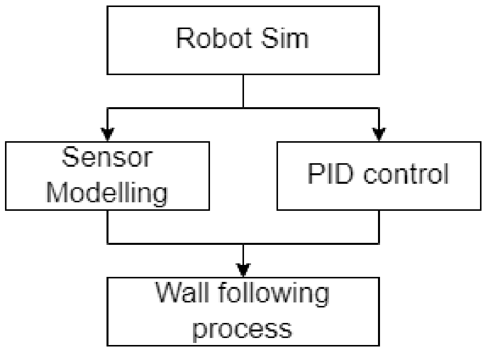

**Figure 1 System Schematic**

## Sensor Modelling

Sensor modelling is applied to create an accurate sensor simulation. In this case, the wall-following robot requires a measuring instrument that can be used to calculate the distance between itself and the arena wall. The sensor has a duty as a unit of distance measurement. To realize this task function, sensor modelling is designed based on arithmetic modelling for the intersection of two lines.

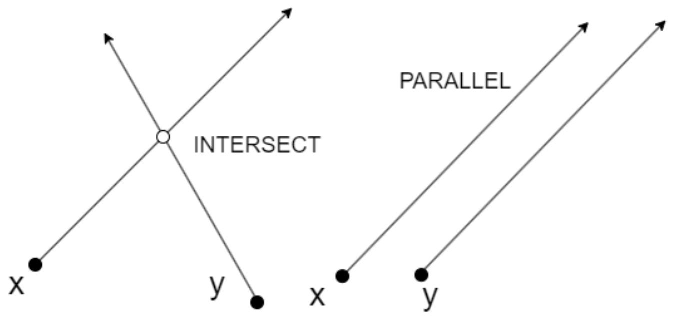

**Figure 2 Lines Segment**

In order to comprehend the measurement model, we use an analogy if the sensor location point is the initial measured distance/range sensor observed. Therefore, logically when the sensor's beam emits towards its bearing and does not detect anything, then the vector formed is just a line vector with a magnitude of max range in the direction according to the bearings.

On the other hand, if the sensor detects an obstacle in a specific bearing, the sensor vector line will be cut off and end at the point of intersection. Hence, a function is created in MATLAB to realize this model, which helps the sensor detect obstacles. According to vector cross products applied to build this function. First, we define the 2-dimensional vector cross product, Equation 1:

$$v \times w = v_x w_y - v_y w_x \tag{1}$$

Suppose the two-line segments run from $x$ to $x + r$ and $y$ to $y + s$. Then any point on the first line is representable as $x + t\,r$ (for a scalar parameter $t$) and any point on the second line as $y + us$ (for a scalar parameter $u$).

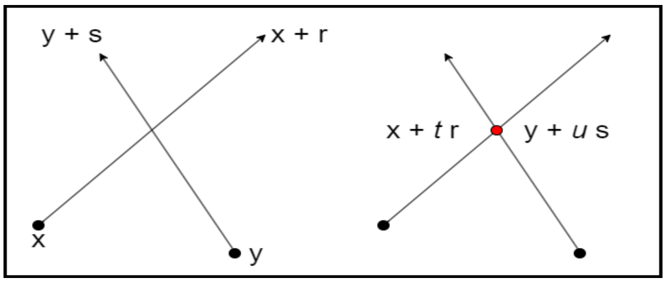

**Figure 3 Vector Cross Product**

The intersection of those two lines can be determined by the equation 2 below:

$$x + tr = y + us \tag{2}$$

Cross both sides with $s$, getting Equation 2:

$$(x + tr)xs = (y + us)xs \tag{3}$$

And since $s \times s = 0$, this means Equation 3:

$$t(r \times s) = (y - x) \times s \tag{4}$$

Furthermore, therefore, solving for $t$ Equation 4:

$$t = (y - x) \times s/(r \times s) \tag{5}$$

In the same way, $u$ can be solved through Equation 5:

$$\begin{gathered}(x + tr) \times r = (y + us) \times r \\ u(s \times r) = (x - y) \times r \\ u = (x - y) \times r/(s \times r)\end{gathered} \tag{6}$$

To reduce the number of computation steps, it is convenient to rewrite this as follows (remembering that $s \times r = -r \times s$), Equation 6:

$$u = (y - x) \times r / (r \times s) \tag{7}$$

Based on the created function, there will be three reference cases that occur in the sensor measurement model such as:

1. If $r \times s = 0$ and $(y - x) \times r \neq 0$, the two lines are parallel and non-intersecting.
2. If $r \times s \neq 0$ and $0 \leq t \leq 1$ and $0 \leq u \leq 1$, the two-line segments meet at the point $x + tr = y + us$.
3. Otherwise, the two-line segments are not parallel but do not intersect.

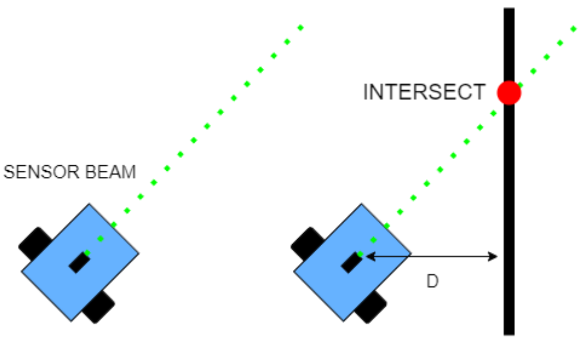

**Figure 4 Measurement Modelling**

By knowing the point of intersection when the laser scanner detects an obstacle, we can implement the measurements model (Goldman, 1990). Thus, sensor modelling as the measurement approach can be achieved through these sequential steps. The robot is assigned for exploration by following the arena wall, and the robot can measure its distance from the obstacles yielded by sensor modelling. The next phase, which has yet to be described, is how to make the wall follow the robot is maintained distance from the wall.

## PID Control

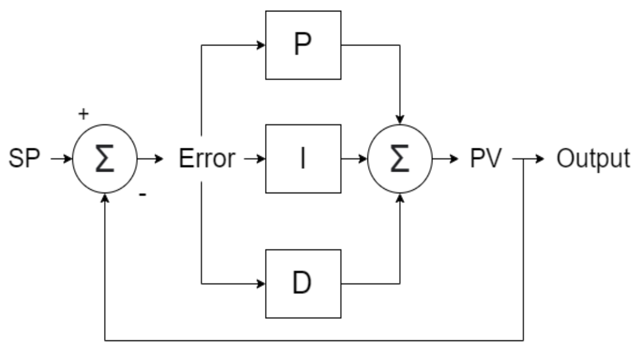

**Figure 5 PID Control**

Process controls are necessary for designing a safe automation environment. A variety of process controls are used to manipulate processes; however, the most simple and often most effective is the PID controller. The controller attempts to correct the error between a measured process variable and desired set point by calculating the difference and then performing a corrective action to adjust the process accordingly. A PID controller controls a process through three parameters: Proportional (P), Integral (I), and Derivative (D). These parameters can be weighted, or tuned, to adjust their effect on the process. PID controllers allow for much better adjustments to be made in the system. Most robotic applications use a PID controller scheme to allow for much better control and fine-tuning adjustments, Eq 8, Eq 9, Eq 10, Eq 11 see below:

$$u(t)=Kce(t)+\frac{Kc}{\tau_i}\int_0^t e(\tau)d\tau+Kc\,\tau_D\,\frac{de(t)}{dt} \tag{8}$$

$$u(t)=Kc\,e(t) \tag{9}$$

$$u(t)=\frac{1}{\tau_i}\int e(\tau)d\tau \tag{10}$$

$$u(t)=\tau_D\,\frac{de}{dt} \tag{11}$$

In an ideal form, a PID controller's output u(t) is the sum of the three terms, as illustrated in Equation 8, where $e(t)$ is the feedback error signal between the reference and the output. Proportional control is the first component of PID. From Equation 9, P-control linearly correlates the controller output to the error or provides a linear relationship between a system's error and the system's controller output. As provided in Equation 10, Integral control is a second constructor of the system. It is often used because it can remove any offset that may exist. Thus, the system returns will be able to return to its original setting. A negative error will cause the signal to the system to decrease, while a positive error will cause the signal to increase. The rest constructor is the derivative control. Unlike P-only and I-only controls, D-control is a form of feed-forward control. It predicts the outcome of the process conditions by analyzing the change in error. It has the task of keeping the system in a consistent setting. The primary benefit of adding D controllers is to resist change in the system, the most important being oscillations. The control output is calculated based on the rate of change of the error with time. The larger the error change rate, the more pronounced the controller response will be (Willis, 1999).

## PID-WFR Concept

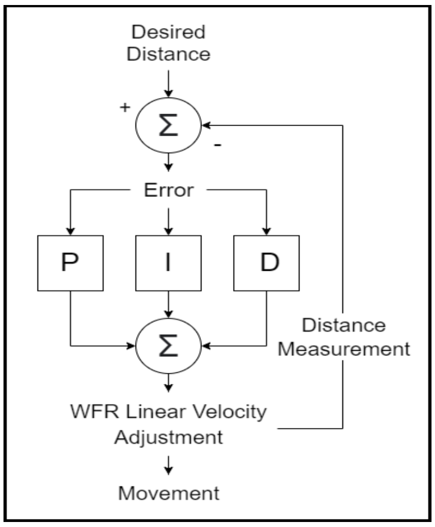

**Figure 6 PID-WFR Concept**

As illustrated in Figure 6, the PID controller is induced in the velocity adjustment process of the wall following the robot. The mobile robot has the assignment to follow the desired path by maintaining the rotational velocity of each wheel, as shown. Therefore, this concept's objective can only be defined if the designed system can produce several outputs or conditions. Three conditions should be fulfilled according to this analogy.

1. If the robot is too close to the wall, the controller adjusts the rotational velocity; thus, the robot moves away from the wall
2. If the robot is in the ideal position, the controller proceeds the robot to move forward
3. If the robot is too far from the wall, the controller adjusts the rotational velocity; thus, the robot moves towards the wall

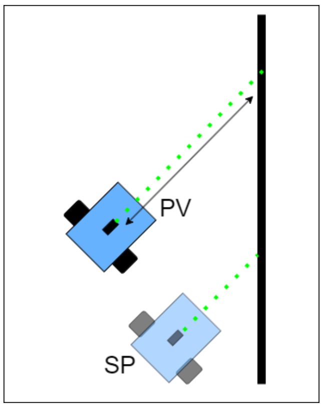

**Figure 7 Set Point and Process Value**

However, the user will need help determining the exact rotational speed values. Hence, the movement behaviour of the wall-following robot is mainly generated by PID. The output of the PID is primarily determined by the values of 3 variables, namely Kp, Ki, Kd. These variables are derived from PID ideal output form described in Equation 8. These parameter values influence the process and refer to the error value. Kp affects the output with process current errors directly. Ki affects the output with refers to the previous error value, and Kd predicts the error value with a specific scale. These three parameters can be pre-determined for their implementation.

| | |
|---|---|
| Kp | : Proportional Gain |
| Ki | : Integral Gain |
| Kd | : Derivative Gain |

In order to utilize the general equation of PID, initially, the error was defined as follows Equation 7:

$$e(t + 1) = SP - PV(t) \tag{12}$$

We refer set point (SP) as the desired robot distance when moving alongside the wall. As well as process value (PV) as the distance obtained through measurement in each iteration. For instance, when the wall is located on the robot's right side, the measured distance of the robot is 15 cm, where the SP is 10 cm. Thus the current error is $-5\ cm$, which means the robot has to move closer to the wall by 5 cm. This can be achieved by reducing the right motor speed by $U_r = -5cm$.

## Tuning Challenge

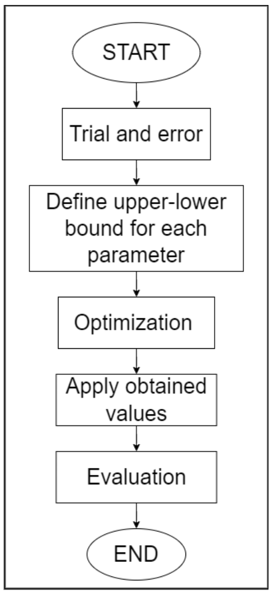

**Figure 8 PID Tuning**

Both the measurement model and PID control have been set. Still, there is an upcoming issue related to how to find the optimum values for each parameter $Kp$, $Ki$, and $Kd$. As represented in the Introduction, these values take work to find. Nevertheless, this issue will be solved by using an optimization technique. As shown above, in Figure 8, the parameters are initially set by trial and error. In order to ease this trial and error stage, it is recommended to adjust the Kp first, followed by Ki and Kd, respectively. The robot's behaviour can be observed by the values adjustment, then defining the upper and lower bound for each parameter. Then proceed to the optimization process after both upper and lower bounds are obtained. New parameters value acquired through the optimization process is applied and evaluated.

## Propose Method

Metaheuristic algorithm in recent times is one of the critical research and development areas since it has significant findings. Their capacities to address and provide near-optimal solutions to problems without providing extensive details of the problem concepts have given them an advantage over many traditional techniques. Metaheuristic algorithms are motivated mainly by some real-world phenomena. They are usually a natural optimization technique (Oladipo et al., 2020). The optimization of PID parameters will be approached by heuristic methods such as Bat Algorithm (BA).

**Table 1 Bat Algorithm Pseudocode**

```
Pseudocode of BAT Algorithm

1:   Define the objective function f(x)
2:   Initialize the bat population $x_i$ and $v_i$ (i=1,2,...,n)
3:   Initialize frequencies $f_i$, pulse rates $r_i$, and loudness $A_i$
4:   Set the iteration counter t = 0
5:   while ($t<t_{max}$)
6:           vary $r_i$ and $A_i$
7:           Generate new solutions by adjusting frequencies
8:           Update velocities and solutions
9:           if rand > $r_i$
10:              Select a solution among the best solutions
11:              Generate a local solution around the best-selected solution
12:          end if
13:          Generate a new solution by flying randomly
14:          if rand > $A_i$ and $f(x_i) < f(x_j)$
15:              Accept new solution
16:          end if
17:          Rank the bats and find the current best solution $x^*$
18:  end while
```

The bat Algorithm (BA) was created by Xin-She Yang in 2010, and this algorithm attempts to mimic the main characteristics of microbats' echolocation behaviour. Bats, especially microbats, use echolocation for navigation, emitting short, ultrasonic pulses that typically last a few milliseconds with frequencies ranging from 25 kHz to about 150 kHz. The loudness of such bursts can be up to 110 dB. When homing for prey, microbats typically increase their pulse emission rates and frequencies, which is also accompanied by a variation in their loudness. The primary purpose of such frequency tuning and echolocation is for navigation and hunting to increase the detection accuracy and success rate of capturing the prey. Such characteristics are simulated in the bat algorithm (Slowik & Kwasnicka, 2020).

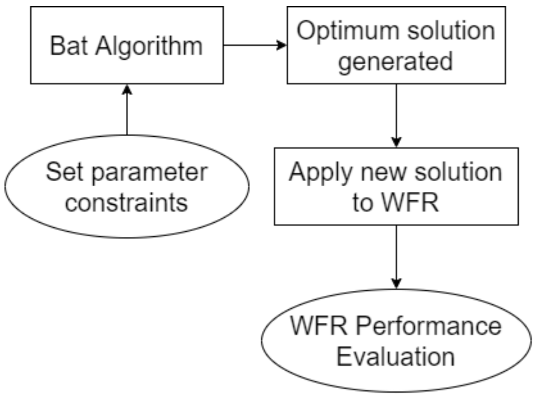
Bat Algorithm
Optimum solution generated
Set parameter constraints
Apply new solution to WFR
WFR Performance Evaluation

**Figure 9 Bat Algorithm Optimization**

The main steps of the BA consist of a single loop with some probabilistic switching during the iteration. As an overview of the proposed method, the optimization problem is finding PID parameters: $Kp$, $Ki$, $Kd$ minimizing Root Mean Square Error (RMSE) as an objective function (fitness function). Based on Table 1 and Figure 9, we see the flow process. The bat algorithm executes the computation based on the upper-lower bound initial parameter set. Refers to the function objective; a solution is initialized. While the loop is started, the bats will update frequency and loudness, and later the selection criteria are raised. When the stopping criteria are reached, the final optimum values are generated according to its function objective. The new PID gain value is obtained and later evaluated through mobile robot behaviour.

## Result and Discussion

### Robot Arena

Considering that this research is carried out using MATLAB simulations, no real arenas were used. The robot arena was built using MATLAB modelling. There were two types of arenas which were built, spirals and squares. The size of the robot modelling will be adjusted proportionally to the modelled maps; thus, the robot steps will be manageable. The spiral arena is modelled using a scale of 200 x 200 units, while the square arena uses a scale of 150 x 150.

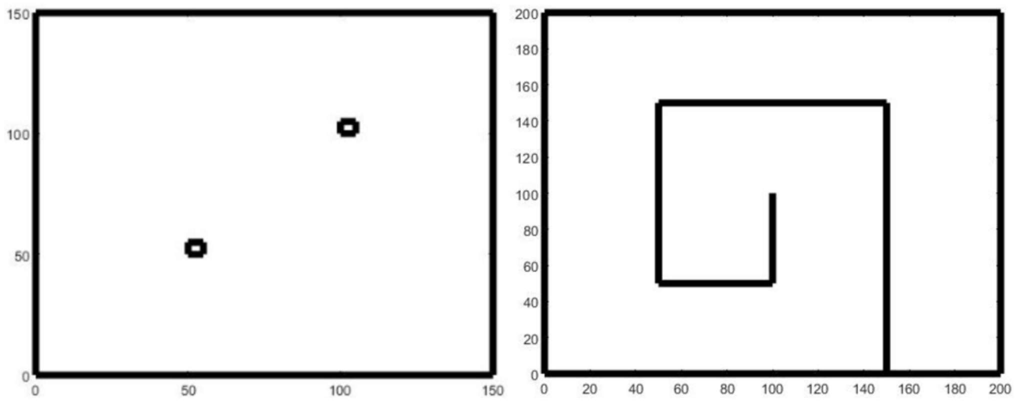

Figure 10 Map Models

Through arena and robot modelling, the performance of the wall-following robot can be observed. On the square map, the robot was placed in the orientation [30,30, deg2rad(5)]. These coordinates indicate when the robot is placed at the lower left end of the arena with a 5° front. On the spiral map, the robot was placed in the orientation [180,30, deg2rad(45)]. Both visualizations can be seen in the figure below.

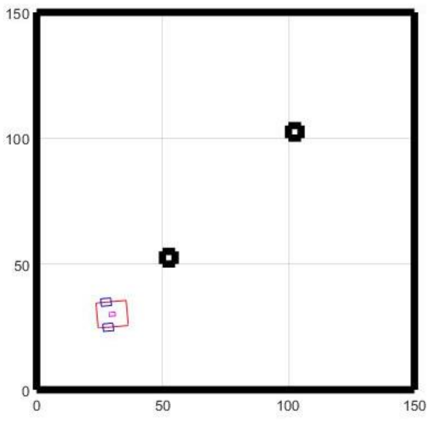

Figure 11 Orientation Map-1

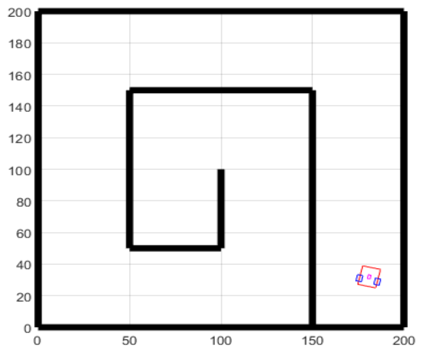

Figure 12 Orientation Map-2

## Determine Constraints

Under the methodology illustrated in figure 8, the pre-optimization step is to set the PID parameter on a trial basis. The first step is to give a Kp value and then observe the robot's performance in the arena. If the value of Kp given is sufficient to provide good performance, then these parameters' upper and lower limits are determined. Whether or not the given Kp value can be observed based on the robot's behaviour in the arena. Then proceed with Ki and Kd using the same method. From the trials that were carried out, the data for the upper and lower ranges for each parameter was obtained as follows:

**Table 2 Parameter Constraints**

| | **Parameters** | **Lower Bound** | **Upper Bound** |
|---|---|---|---|
| **Map-1** | Kp | 0.300 | 0.600 |
| | Ki | -0.011 | -0.010 |
| | Kd | 0.00 | 0.05 |
| **Map-2** | Kp | 0.350 | 0.500 |
| | Ki | -0.01 | 0 |
| | Kd | 0 | 0.003 |

## Optimization

Getting the upper and lower bounds for each parameter is the first step for optimization. Following the proposed method, the optimization approach used is Bat Algorithm. This method is built by determining the parameters of the algorithm itself. The parameters in question are the number of bat populations, the maximum number of iterations, dimensions, objective functions, acoustic vibration ratio, frequency, and amplitude (Robandi, 2021). Dimensions are reflections of the axes of the lines used. In this program, the dimensions are three because the PID position is determined from 3 parameters, namely Kp, Ki, and Kd. The BA parameter values used are listed in the table below:

**Table 3 BAT Algorithm Parameter**

| **Parameter** | **Value** |
|---|---|
| Maximum iterations | 30 |
| Population | 10 |
| Dimensions | 3 |
| Max Frequency | 1 |
| Min Frequency | 0 |
| Amplitude | 1 |
| Pulse emission rate | 1 |
| Alpha | 0.097 |
| Gamma | 0.001 |
| Objective Function | RMSE |

The optimization produced optimal PID parameter value, and the performance was observed through the RMSE value. The objective function used is the minimum RMSE. The optimization process from iteration to iteration produced a fitness value plotted using a convergence curve graph to determine the level of convergence of the optimization algorithm used. By comparing the level of convergence between the Bat algorithm and PSO. Particle Swarm Optimization was given equal parameter values for iterations and population while the user randomly set the rest. The next convergence curves for each method applied were obtained as follows.

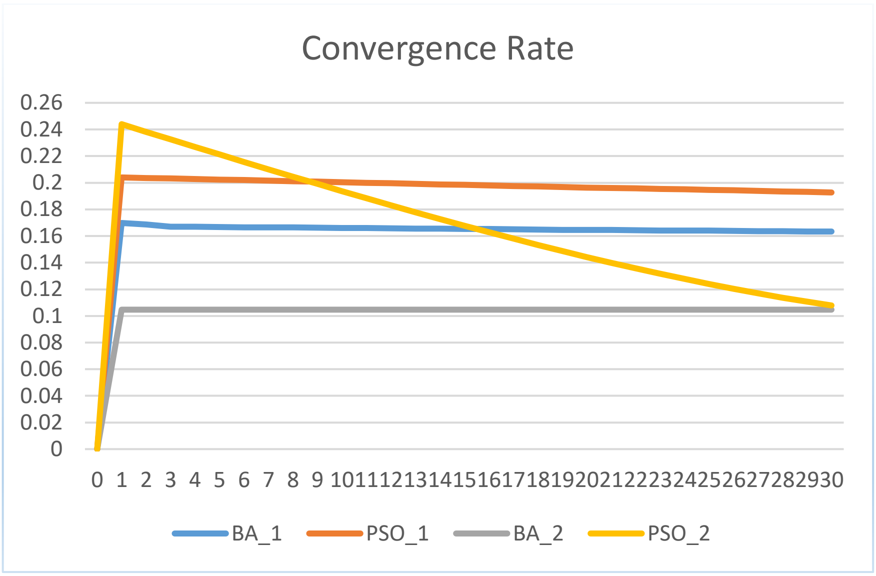

**Figure 13 Convergence Curve**

## Performance Analysis

The optimum values of the parameters Kp, Ki, and Kd obtained from the optimization process were applied to the wall-following robot. Performance evaluation can be done based on the movement behaviour of the robot in the arena and the RMSE value or dynamic error generated in the entire iteration of the movement. However, when referring to the robot's movement alone, the evaluation cannot be accurate and is subjective. Therefore, the movement behaviour of the robot will be a minor factor that is not considered. Instead, the evaluation is based on the resulting dynamic error value. The dynamic error generated by the experimental set is recorded based on the table below.

**Table 4 Dynamic Error**

| | **Dynamic Error** | | | **Improvement (%)** |
|---|---|---|---|---|
| | **PID only** | **PID-PSO** | **PID-BA** | |
| Map-1 | 0.43541 | 0.1939 | 0.16278 | 3.112 |
| Map-2 | 0.01495 | 0.01072 | 0.01046 | 0.026 |

Following the data presented in the table above, PID parameterization will improve performance when assisted by optimization. PSO is an optimization method used to compare with the proposed method, namely BA. Without needing to discuss parameterization on a trial basis, it can be seen that PID control with PSO produces a dynamic error of 0.1939 in arena-1 and 0.01072 in arena-2. Furthermore, the resulting dynamic error in PID-BA is 0.16278 in arena 2 and 0.01046 in arena 2. Performance improvements can be seen in the "improvement" column. Implementing PID parameter optimization using BA resulted in improvisation, although the difference was insignificant. When adjusted to the convergence curve in figure 13, it can be seen that the optimization for arena-1 does not change much. However, in arena-2, BA appears to be faster in producing fmin values, while PSO requires longer iterations to achieve convergence.

## Conclusions

This paper has shown the performance of a wall-following robot controlled by a PID-BA. The role of the BA was to produce proper values of three unknown parameters of the PID controller in the wall-following robot application. The PID controller performance was evaluated through the dynamic error generated by the PID controller. Different approaches were simulated and compared. Based on the comparative results, we can conclude that Bat Algorithm, inspired by bat echolocation behaviour, produces better performance in both terms of dynamic error produced and convergence rate.

## References

Adriansyah, A., Suwoyo, H., Tian, Y., & Deng, C. (2019). Improving the Wall-Following Robot Performance using PID-PSO Controller. *Jurnal Teknologi*, *81*(3).

Fišer, J., & Z\'\itek, P. (2019). PID controller tuning via dominant pole placement in comparison with Ziegler-Nichols tuning. *IFAC-PapersOnLine*, *52*(18), 43–48. https://doi.org/10.1016/j.ifacol.2019.12.204

Gagnon, I., April, A., & Abran, A. (2020). A critical analysis of the bat algorithm. *Engineering Reports*, *2*(8), e12212. https://doi.org/10.1002/eng2.12212

Goldman, R. (1990). The intersection of two lines in three-space. In *Graphics gems* (p. 304).

Hodson, T. O. (2022). Root mean square error (RMSE) or mean absolute error (MAE): when to use them. *Geoscientific Model Development Discussions*, pp. 1–10.

Iqbal, M. H., & Aji, W. S. (2021). Wall Following Control System with PID Control and Ultrasonic Sensor for KRAI 2018 Robot. *International Journal of Robotics and Control Systems*, *1*(1), 1–14. https://doi.org/10.31763/ijrcs.v1i1.206

Lee, K., I am, D.-Y., Kwak, B., Ryoo, Y.-J., & others. (2018). Design of fuzzy-PID controller for path tracking of mobile robot with differential drive. *International Journal of Fuzzy Logic and Intelligent Systems*, *18*(3), 220–228. https://doi.org/10.5391/IJFIS.2018.18.3.220

M Zahir, A. A., Already, S. S. N., Othman, W., & Ahmad, M. F. (2018). Genetic algorithm optimization of PID controller for brushed DC motor. In *Intelligent manufacturing \& mechatronics* (pp. 427–437). Springer. https://doi.org/10.1007/978-981-10-8788-2_38

Mazlan, N. N. B. M., Thamrin, N. M., & Razak, N. A. (2020). Comparison Between Ziegler-Nichols and AMIGO Tuning Techniques in Automated Steering Control System for Autonomous Vehicle. *2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 7–12. https://doi.org/10.1109/I2CACIS49202.2020.9140089

Oladipo, S., Sun, Y., & Wang, Z. (2020). Optimization of PID controller with metaheuristic algorithms for dc motor drives. *Int. Rev. Electr. Eng.*, *15*(5), 352–381. https://doi.org/10.15866/iree.v15i5.18688

Raza, Y., Ahmed, S. F., Ali, A., Joyo, M. K., & Kadir, K. A. (2018). Optimization of PID using PSO for upper limb rehabilitation robot. *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 1–4. https://doi.org/10.1109/ICETAS.2018.8629100

Robandi, I. (2021). *Artificial Intelligence: Mengupas Rekayasa Kecerdasan Tiruan*. Penerbit Andi.

Sekarsari, K., & Tata, T. (2021). Performance analysis of PID control in DC Brushless motor using trial and error method. *IOP Conference Series: Materials Science and Engineering*, *1098*(4), 42027. https://doi.org/10.1088/1757-899x/1098/4/042027

Slowik, A., & Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, *32*(16), 12363–12379.

Suksawat, T., & Kaewpradit, P. (2021). Comparison of Ziegler-Nichols and Cohen-Coon Tuning Methods: Implementation to Water Level Control Based MATLAB and Arduino. *Engineering Journal Chiang Mai University*, *28*(1), 153–168.

Suwoyo, H., Tian, Y., Deng, C., & Adriansyah, A. (2018). We are improving a wall-following robot performance with a PID-genetic algorithm controller. *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 314–318. https://doi.org/10.1109/EECSI.2018.8752907

Tzafestas, S. G. (2018). Mobile robot control and navigation: A global overview. *Journal of Intelligent & Robotic Systems*, *91*(1), 35–58. https://doi.org/10.1007/s10846-018-0805-9

Willis, M. J. (1999). Proportional-integral-derivative control. *Dept. of Chemical and Process Engineering University of Newcastle*.