International Journal on Advanced Science Engineering Information Technology

Vol.9 (2019) No. 4
ISSN: 2088-5334

Modified Particle Swarm Optimization Based PID for Movement Control of Two-Wheeled Balancing Robot

Nurul Hasanah[#1], Alrijadjis[#2], Bambang Sumantri[#3]

[#]*Electronic Engineering, Politeknik Elektronika Negeri Surabaya, Indonesia*
E-mail: [1]nurul.hasanah024@gmail.com; [2]alrijadjis@pens.ac.id; [3]bambang@pens.ac.id

***Abstract*— Two-wheeled balancing robot is a mobile robot that has helped various human's jobs such as the transportations. To control stability is still be the challenges for researchers. Three equations are obtained by analyzing the dynamics of the robot with the Newton approach. To control three degrees of freedom (DOF) of the robot, PIDs is tuned automatically and optimized by multivariable Modified Particle Swarm Optimization (MPSO). Some parameters of the PSO process are modified to be a nonlinear function. The inertia weight and learning factor variable on PSO are modified to decreasing exponentially and increasing exponentially, respectively. The Integral Absolute Error (IAE) and Integral Square Error (ISE) evaluate the error values. The performances of MPSO and PSO classic are tested by several Benchmark functions. The results of the Benchmark Function show that Modified PSO proposed to produce less error and overshoot. Therefore, the MPSO purposed are implemented to the plant of balancing robot to control the angle, the position, and the heading of the robot. The result of the simulation built shows that the MPSO – PID can make the robot moves to the desired positions and maintain the stability of the angle of the robot. The input of distance and angle of the robot are coupling so MPSO needs six variable to optimize the PID parameters of balancing and distance control.**

***Keywords*— modified PSO; balancing robot; PID; IAE; ISE; benchmark function.**

I. INTRODUCTION

Nowadays, researchers have developed various kind of robots to help human's jobs, from wheeled robot to humanoid [1]. Many controllers and methods have been made to make a robot dynamically stable and robust solution. One kind of robots is a two-wheeled balancing robot. This robot still become a popular topic because of movement control development [1]. Because the number of actuators are less than the number of degree of freedom, the robot is categorized as an underactuated system.

One of the most widely applied methods for balancing robot is the PID method [1]–[3]. However, how to tune the PID parameter is still a major problem for researchers. That is because usually the system has a nonlinear system and there are unknown disturbances, such as friction, slip, and external force. The performance of the motor also has a lot of effects on the PID tuning value due to the nonlinearity of the motor itself [4]. One of the Artificial Intelligence tuning methods is Particle Swarm Optimization (PSO).

PSO was introduced by Kennedy and Eberhart in 1995 [5] and became one of the modern heuristic algorithms [6]. This algorithm is inspired by the behavior of birds flocking, such as sharing internal and global information about food between individuals. PSO has been implemented in various fields because of high-speed computation [7] and simple operations [8]. However, the classic PSO algorithm has a big problem, namely premature convergence. This problem causes a rapid loss of diversity during evolutionary processing [9]. The classic PSO is easy to be trapped into local optimum in high dimensional space [7]. To improve the convergence characteristics of the PSO algorithm, the modifications are made in this research.

One of the most effect on the evolution process by PSO is inertia weight value *w*. Inertia weight in PSO is introduced by [10]. Inertia weight value has to be tuned in order to the process of exploration and exploitation to be able to achieve the optimal value. Many studies have described the best method how to choose inertia weight value, and one of the most method used is based on time-varying, such as linear decreasing law [11], sigmoid [12], logarithm decreasing law [13] and function [14]. In this research, the inertia weight will be modified to another function.

The plant used in this research is a two-wheeled balancing robot. After the dynamic equation is obtained, the PID controls are designed to control the balancing, the heading, and the position of the robot to keep the robot stands upright and move to the desired position. PID parameters values are obtained by tuning them using the Modified PSO that is designed later.

A. Dynamic Model of Balancing Robot

To obtain the dynamic system from balancing robot modeling, this research use force analysis [15], [16] and the dynamic mathematical model on this system. Two-wheeled balancing robot structure is consist of two main parts; they are wheels (Fig.1) and body robot (Fig.2). Each wheel is actuated by separating motors, with the assumption that parameter of quality of inertia moment and the radius of wheels are the same.

1) Force Analysis on Wheels of Robot

This robot balancing has two wheels with each force analysis, as shown in Fig. 1.

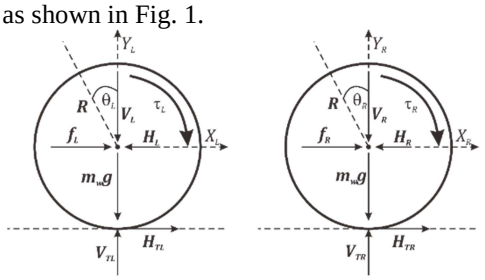

Fig. 1 Force Analysis of the Robot Wheels

According to the force analysis, the dynamic equations are obtained based on Newton's law and the torque formula for the right and the left wheel. Assuming that there is no slippage between the wheels and the ground, the balancing force and moment acting on the right wheel produce the following equations:

$$m_w \ddot{x}_R = H_{TR} - H_R + f_R \tag{1}$$

$$I_w \ddot{\theta}_R = \tau_R - H_{TR}.R \tag{2}$$

Similarly, for the left wheel:

$$m_w \ddot{x}_L = H_{TL} - H_L + f_L \tag{3}$$

$$I_w \ddot{\theta}_L = \tau_L - H_{TL}.R \tag{4}$$

Considering Eq. (1)–(4) we obtained the Eq. (5).

$$2\left(M_w + \frac{I_w}{R^2}\right)\ddot{x} = \frac{\tau_R + \tau_L}{R} - (H_L + H_R) + (f_L + f_R) \tag{5}$$

2) Force Analysis on Body of Robot

Body of the robot is modeled as an inverted pendulum. The body of the robot's force analysis is shown as Fig. 2.

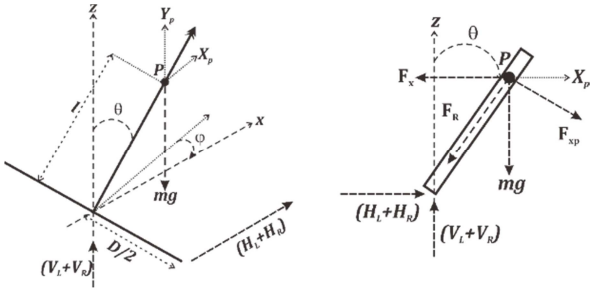

Fig. 2 Force Analysis of Robot Body

The movement on x-axis:

$$\ddot{x}_p = \ddot{x} + l\cos(\theta)\,\ddot{\theta} - l\sin(\theta)\,\dot{\theta}^2 \tag{6}$$

Balancing force acting on the pendulum of the robot on x-axis:

$$\begin{aligned} F_p &= F_x - F_{xp} + F_R \\ &= -m\ddot{x} - ml\cos(\theta)\ddot{\theta} + ml\dot{\theta}^2\sin(\theta) \end{aligned} \tag{7}$$

Balancing forces acting on the platform of the robot along the x-axis:

$$m_p\ddot{x} = H_L + H_R + F_P \tag{8}$$

Substitute the Eq. (7) with Eq. (8) we obtained:

$$H_L + H_R = (m_p + m)\ddot{x} + ml\,cos(\theta)\ddot{\theta} - ml\dot{\theta}^2\,sin(\theta) \tag{9}$$

By substituting Eq. (9) into Eq. (5) we obtain the first dynamic equation:

$$\ddot{x}\left(m_p + m + 2m_w + 2\frac{I_w}{R^2}\right) + ml\left(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right) = \frac{1}{R}(\tau_R + \tau_L) + (f_L + f_R) \tag{10}$$

The total of torque from the center of mass of the robot body is expressed as Eq. (11).

$$\tau_T = ml^2\ddot{\theta} + ml\cos(\theta)\ddot{x} - mgl\sin(\theta) \tag{11}$$

Moment of robot about the z-axis is:

$$I_M\ddot{\theta} = -\tau_T \tag{12}$$

By substituting Eq. (11) into Eq. (12) we obtain the second dynamic equation:

$$(I_M + ml^2)\ddot{\theta} = -ml\cos(\theta)\ddot{x} + mgl\sin(\theta) \tag{13}$$

3) Heading of Robot Analysis

A moment acting on pendulum and platform in the z-axis is:

$$I_p\ddot{\varphi} = D(H_L - H_R) \tag{14}$$

Considering Eq. (1) – (4) and by substituting Eq. (14) into them we obtain the third dynamic equation:

$$I_\varphi\ddot{\varphi} = \frac{D}{R}(\tau_L - \tau_R) + D(d_L - d_R) \tag{15}$$

where $I_\varphi = I_p + D^2(m_w + \frac{I_w}{R^2})$

From the three analyzes, we obtained three dynamic equations for 3 movements (3 DOF) on Eq. (10), (13) and (15). By specifying $x_1 = x;\ x_2 = \dot{x}; x_3 = \theta; x_4 = \dot{\theta}; x_5 = \varphi$; and $x_6 = \dot{\varphi}$ so the state space equation can be written as:

$\dot{x}_1 = x_2$

$$\dot{x}_2 = \frac{g\tan x_3\left({A_2}^2\cos^2 x_3-(A_1)(A_3)\right)-A_2A_3\,x_4^2\sin(x_3)+A_1A_3g\tan x_3}{{A_2}^2\cos^2 x_3-(A_1)(A_3)} - \frac{\frac{A_3}{R\left({A_2}^2\cos^2 x_3-(A_1)(A_3)\right)}u_1 - \frac{A_3}{\left({A_2}^2\cos^2 x_3-(A_1)(A_3)\right)}(f_L+f_R)}{}$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = \frac{(A_2)^2\,x_4^2\sin(2x_3)-(2A_1A_4\,g\sin x_3)}{2\left(A_2^2\cos^2 x_3-(A_1)(A_3)\right)} + \frac{A_2\cos x_3}{R\left({A_2}^2\cos^2 x_3-(A_1)(A_3)\right)}u_1 + \frac{A_2\cos x_3}{\left({A_2}^2\cos^2 x_3-(A_1)(A_3)\right)}(f_L+f_R)$$

$$\dot{x}_5 = x_6$$

$$\dot{x}_6 = \frac{D}{RI_\varphi}u_2 + \frac{D}{I_\varphi}(f_L-f_R)$$

where :

$A_1 = m_p + m + 2m_w + \frac{2I_w}{R^2}$
$A_2 = ml$
$A_3 = ml^2 + I_M$
$A_4 = mgl$
$u_1 = \tau_L + \tau_R$
$u_2 = \tau_L - \tau_R$

Before linear controllers are designed, the linearization models have to be obtained. For the linearization of system, this research uses Taylor series about equilibrium point [15]. The state-space of balancing robot equation can be written as:

$$\dot{x} = f(x,u)$$
$$y = Cx$$

where $x \in \mathbb{R}^6, u \in \mathbb{R}^2$, and $y \in \mathbb{R}^6$

Equilibrium point is defined (x_0,u_0) = (0,0). Taylor series expansion about equilibrium point is written as:

$$\begin{aligned}\dot{x} &= f(x,u)\\ &= f(x_0,u_0) + \left[\frac{\partial f}{\partial x_1}(x_1-x_{01}) + \cdots + \frac{\partial f}{\partial x_5}(x_5-x_{05}) + \frac{\partial f}{\partial u_1}(u_1-u_0) + \frac{\partial f}{\partial u_2}(u_2-u_{02})\right] + \frac{1}{2!}\left[\frac{\partial^2 f}{\partial x_1^2}(x_1-x_{01})^2 + \cdots + \frac{\partial^2 f}{\partial x_5^2}(x_5-x_{05})^2 + \frac{\partial^2 f}{\partial u_1^2}(u_1-u_0)^2 + \frac{\partial^2 f}{\partial u_2^2}(u_2-u_{02})^2\right] + \cdots\end{aligned}$$

The partial derivative is evaluated about the equilibrium point by neglecting the high order terms of its expansion:

$$\dot{x} - f(x_0,u_0) = \left[\frac{\partial f}{\partial x_1}(x_1-x_{01}) + \cdots + \frac{\partial f}{\partial x_5}(x_5-x_{05}) + \frac{\partial f}{\partial u_1}(u_1-u_0) + \frac{\partial f}{\partial u_2}(u_2-u_{02})\right]$$

Linearization form can be written as:

$$\mathbf{\dot{x} = Ax + Bu}$$
$$\mathbf{y = Cx}$$

Where **A** and **B** are the constant matrix that are obtained by using the Jacobean formula.

$$\mathbf{A} = \begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_6}{\partial x_1} & \cdots & \frac{\partial f_6}{\partial x_6}\end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix}\frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2}\\ \vdots & \vdots\\ \frac{\partial f_6}{\partial u_1} & \frac{\partial f_6}{\partial u_2}\end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}$$

So, the linearization model's state space form can be written in form:

$$\begin{bmatrix}\dot{x}_1\\\dot{x}_2\\\dot{x}_3\\\dot{x}_4\\\dot{x}_5\\\dot{x}_6\end{bmatrix} = \begin{bmatrix}0&1&0&0&0&0\\0&0&\psi&0&0&0\\0&0&0&1&0&0\\0&0&\vartheta&0&0&0\\0&0&0&0&0&1\\0&0&0&0&0&0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\\x_6\end{bmatrix} + \begin{bmatrix}0&0\\G_2&0\\0&0\\G_4&0\\0&0\\0&G_6\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}$$

where :

$$\psi = -\frac{m^2gl^2}{(4m_w+m_p+m)I_M+(4m_w+m_p)ml^2} \quad (16)$$

$$\vartheta = \frac{(4m_w+m_p+m)mgl}{(4m_w+m_p+m)I_M+(4m_w+m_p)ml^2} \quad (17)$$

$$G_2 = \frac{1}{R}\frac{(I_M+ml^2)}{(4m_w+m_p+m)I_M+(4m_w+m_p)ml^2} \quad (18)$$

$$G_4 = -\frac{1}{R}\frac{ml}{(4m_w+m_p+m)I_M+(4m_w+m_p)ml^2} \quad (19)$$

$$G_6 = \frac{1}{R}\frac{D}{(2D^2m_w+I_p)} \quad (20)$$

TABLE I
THE PARAMETERS OF BALANCING ROBOT

τ_R, τ_L	Motor torque	
f_R, f_L	External force to wheels	
θ_R, θ_L	Rotational angle of wheels	
x_R, x_L	Displacement of wheels on x-axis	
θ	Tilt angle of robot	
φ	Heading angle of robot	
m_w	Mass of the wheels	0.12 kg
R	Radius of the wheels	0.06 m
m	Mass of pendulum	0.55 kg
g	Gravitation acceleration	9.8 m/s^2
l	Distance of COG	0.2 m
D	Distance between two wheels	0.3 m
m_p	Mass of platform	0.01 kg
I_M	Moment of inertia of platform about Y-axis	0.03 kg.m^2
I_p	Moment of inertia of platform and pendulum about z-axis	0.004 kg.m^2

4) Controllability and Observability

A system is controllable if input **u** can control the system from x(0) into x(T) in finite time. A linear system is entirely controllable if the controllable value $\mathbf{C_M}$ has full rank [15]. The controllability matrix **M** is defined as

$$\mathbf{M} = [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A^2B} \quad \ldots \quad \mathbf{A^{n-1}B}] \tag{21}$$

where *n* is order of the system.

By using Eq.(21), the controllability matrix $\mathbf{M}$ and controllable value $\mathbf{C_M}$ of balancing robot can be obtained as Eq. (22) and (23).

$$\mathbf{M} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} & \mathbf{A^3B} & \mathbf{A^4B} & \mathbf{A^5B} \end{bmatrix} \tag{22}$$

$$\mathbf{C_M} = \text{rank}(\mathbf{M}) \tag{23}$$

From Eq. (23), the rank of the controllable matrix $\mathbf{C_M}$ is 6 which is equal to the order of the system, n = 6, that means the linearized model of balancing robot is completely controllable.

A system is observable if the observable value $\mathbf{O_M}$ has full rank [15]. The observable matrix $\mathbf{O}$ is defined:

$$\mathbf{O} = [\mathbf{C} \quad \mathbf{AC} \quad \mathbf{A^2C} \quad \ldots \quad \mathbf{A^{n-1}C}]^{\mathrm{T}} \tag{24}$$

By using Eq.(24), the observability matrix $\mathbf{O}$ and observable value $\mathbf{O_M}$ of balancing robot can be obtained as Eq. (25) and (26).

$$\mathbf{O} = \begin{bmatrix} \mathbf{C} & \mathbf{AC} & \mathbf{A^2C} & \mathbf{A^3C} & \mathbf{A^4C} & \mathbf{A^5C} \end{bmatrix}^{\mathrm{T}} \tag{25}$$

$$\mathbf{O_M} = \text{rank}(\mathbf{O}) \tag{26}$$

From Eq. (26), the rank of an observable matrix $\mathbf{O_M}$ is 6 which is equal to the order of the system, n = 6, that means the linearized model of balancing robot is completely observable.

5) Stability

The stability of the system uses Lyapunov's equation to test stability. The equivalent characterization of stability is obtained using Eq.(27).

$$\mathbf{A^TP + PA = -Q} \tag{27}$$

where $\mathbf{Q}$ matrix is defined as the identity matrix. By verifying that $V(x) = \mathbf{x^TPx}$ is positive definite, this system can be asymptotically stable if and only if $\mathbf{P}$ matrix is positive definite.

The open-loop of the system is analyzed by using this method and the results show that the system is unstable because P matrix is not positive definite. So the controllers are required to make the system stable on the desired positions.

6) PSO

Particle Swarm Optimization (PSO) is one of evolutionary algorithm which every potential solution called 'particle' can change their position and velocity [17]. During looping, every particle can manage their position to the best position, which is obtained from the group of particles.

Neighbor particle association and the history of their experience establish the directions of particles during exploring the best position. The position of n*th* particle X_n and the velocity of n*th* particle V_n change based on i*th* iteration. The previous best positions will be stored as Pb_n and the best particle among the group is represented as Gb. The velocity and position of particles are expressed as:

$$V_n{}^{(i+1)} = w * V_n{}^{i} + c_1 r_1 * \left(Pb_n^i - X_n{}^{i}\right) + c_2 r_2 * (Gb - X_n{}^{i}) \tag{28}$$

$$X_n{}^{(i+1)} = X_n{}^{i} + V_n{}^{(i+1)} \tag{29}$$

where:

$V_n{}^{i}$: Velocity of n*th* particle on i*th* iteration
w : Inertia weight
$c_{1,2}$: learning acceleration factor
$r_{1,2}$: Random value [0,1]
$X_n{}^{i}$: Position of n*th* particle on i*th* iteration
Pb_n^i : The best position of n*th* particle on i*th* iteration
Gb : The best position of the particle

II. Materials and Method

Control system in this system is designed such as Fig. 3. This system uses three main controllers to control three DOF of the robot movement: MPSO-PID control for balancing control itself, for heading control, and for forwarding movement control.

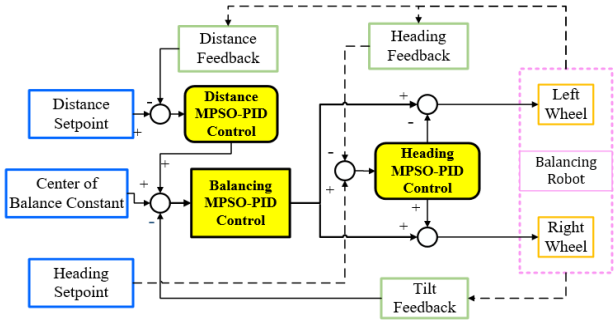

Fig. 3 Overall Control Design of Balancing Robot

Fig. 3 shows that an angle error for balancing control input is obtained by combining the tilt feedback and the output from distance control [2], [3], [18]. The output from balancing MPSO-PID would be combined with the output of heading MPSO-PID control and given to the both of motor on the right and left-side of the robot.

Distance control is used to make the robot moves to the desired position. As stated by [2] the equation of combination errors feedback are Eq.(30) and Eq.(32).

$$\delta(k) = C_B - \theta(k) \tag{30}$$

$$\alpha(k) = \delta(k) + \gamma(k) \tag{31}$$

where:

$\delta(k)$: Angle error of robot's center balance
C_B : Constant value of robot's center balance
$\theta(k)$: Actual angle of the robot's body
$\alpha(k)$: Overall angle error of the system
$\gamma(k)$: Angle; output from distance PID controller

A. Balancing MPSO-PID Control

Balancing control is the main control for a two-wheeled balancing robot. This control is used to maintain the

standing position of the robot by controlling the tilt feedback from IMU sensor.

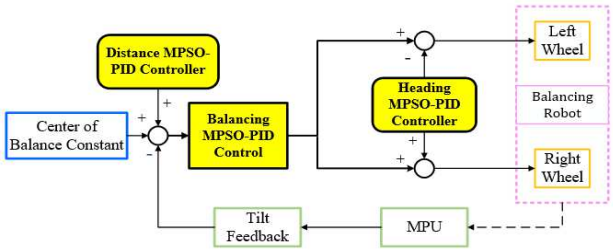

Fig. 4 Balancing Control Design

The input of balancing control is the sum of angle error from IMU (here using MPU) and the result of distance control, namely overall angle error ($\alpha(k)$). Balancing control can be calculated as :

$$\upsilon_B(k) = Kp_B * \alpha(k) + Ki_B * \sum_{n=1}^{m} \alpha(k) + Kd_B * (\alpha(k) - \alpha(k)_{k-1}) \quad (32)$$

where:

$\upsilon_B(k)$: velocity of balance control
Kp_B : Proportional constanta for Balancing PID
Ki_B : Integral constanta for Balancing PID
Kd_B : Derivative Constanta for Balancing PID
$\alpha(k)_{k-1}$: Previous overall angle error

The PID parameters value are obtained based on simulation using modified PSO to maintain the robot balance.

B. *Distance MPSO-PID Control*

As mentioned before, this control is used for the movement of robot to go to the desired position. The result of distance control can affect the angle setpoint of robot. To make robot moves forward, the angle setpoint is set not equal to zero [19]. The input for distance control is distance feedback from rotary encoder.

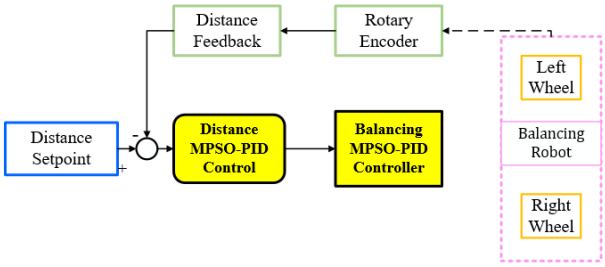

Fig. 5 Distance Control Design

The distance control can be calculated as in the following formula[2]:

$$Ed = x_S - x(k) \quad (33)$$

$$\gamma(k) = Kp_D * Ed + Ki_D * \sum_{n=1}^{m} Ed + Kd_D * (Ed - Ed_{k-1}) \quad (34)$$

where:

x_S : Distance setpoint
$x(k)$: Distance from the rotary encoder
$\gamma(k)$: Angle offset, distance PID control output
Kp_D : Proportional constanta for Distance PID
Ki_D : Integral constanta for Distance PID
Kd_D : Derivative Constanta for Distance PID
Ed : Distance error
Ed_{k-1}: Previous distance error

The PID parameters value are obtained based on simulation using modified PSO to make the robot moves to the desired position.

C. *Heading MPSO-PID Control*

Orientation of robot is obtained from IMU sensor. Diagram block for this control is shown in Fig.6

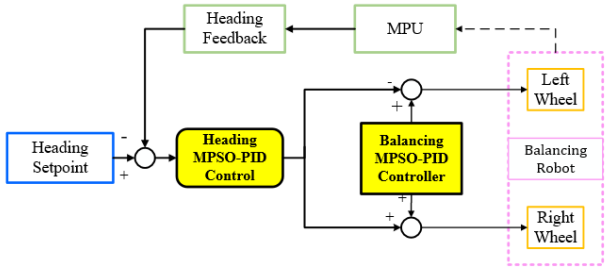

Fig. 6 Heading Control Design

Heading error and the input control can be calculated as:

$$Eh = \varphi_s - \varphi(k) \quad (35)$$

$$\upsilon_H(k) = Kp_H * Eh + Ki_H * \sum_{n=1}^{m} Eh + Kd_H * (Eh - Eh_{k-1}) \quad (36)$$

where:

φ_s : Heading setpoint
$\varphi(k)$: Heading value from sensor
$\upsilon_H(k)$: Velocity for steering
Kp_H : Proportional constanta for Heading PID
Ki_H : Integral constanta for Heading PID
Kd_H : Derivative constanta for Heading PID
Eh : Heading error now
Eh_{k-1}: Previous Heading error

Result from this control υ_H will be combined with the result of balancing control υ_B for steering the orientation of robot. The velocity of left and right motor can be calculated using equation:

$$V_L = \upsilon_B - \upsilon_H \quad (37)$$

$$V_R = \upsilon_B + \upsilon_H \quad (38)$$

The PID parameters value are obtained based on simulation using modified PSO to steer the orientation of robot.

D. *Modified PSO* (MPSO)

Modified PSO is used to set the PID parameters value on a two-wheeled balancing robot. The error value is controlled by MPSO–PID to obtain the speed that will be sent to both motors on the left and right side of the robot. The error values in MPSO use Integral Absolute Error (IAE) or

Integral Square Error (ISE) to get the value of the fitness function.

The modification is done on PSO by changing the value of the parameter on the weight formula (w_i), value of c_1 and c_2. The value of weight (w_i) of each generation are modified to Eq. (39) and Eq. (40). This parameter modification aims to widen the particles range in the beginning generation (global optimum) and to reduce the area when they are in the last generation (local optimum), so the PID parameters value obtained are more mature.

$$w_i = w_{max}\left(\sqrt[N-1]{\left(\frac{w_{max}}{w_{min}}\right)\left(\frac{w_{min}}{w_{max}}\right)^{k}}\right) \quad (39)$$

$$c_{1,2} = e^{(\frac{k}{N})^2} \quad (40)$$

The Eq. (39) shows that the w_i value is changed exponentially from w_{max} to w_{min}, while in PSO classic the w_i value is constant. The Eq. (40) also shows that the $c_{1,2}$ value is changed exponentially, while in PSO classic the $c_{1,2}$ value is constant. The classic algorithm process keeps exploring the best position but the convergence is delayed if the w_i value is large while the algorithm needs exploitation in the last iterations [20].

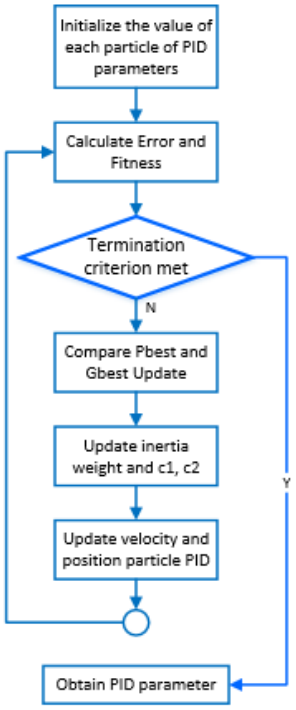

Fig. 7 Flowchart of Modified PSO

The steps for implementing the Modified PSO are presented in this study. *Firstly*, set the parameters of the learning factors $c_{1,2}$, the maximum inertia weight w_{max}, the minimum inertia weight w_{min}, the number of iterations N, the number of particles m, and the dimension of particle dim. *Secondly*, initialize the positions and velocities of particles of PID parameters in a given range. *Thirdly*, calculate the inertia weight w_i with the Eq. (39) and the learning factor $c_{1,2}$ with the Eq. (40). *Fourthly*, evaluate errors and the finesses of every particle of PID parameters and compare them with the previous optimal fitness to get the best local position *Pb* and best global position *Gb*. *Fifthly*, update the velocities and positions of particles of PID parameters.

III. Results and Discussion

This section will give the results of modified PSO tests using the benchmark functions and test the modified PSO proposed at the robot balancing plant. The Benchmark functions are used to test the performances of the optimization.

A. Benchmark Function Testing

The testing of optimization use several Benchmark Function, they are the Multivariable Sphere Function, the Rosenbrock Function, the Griwank Function, the Beale Function, and the Booth Function. Those functions use two variables (x, y) with function equations respectively are expressed as follow:

$$1.\ f_1(x,y) = (10-x)^2 + (15-y)^2; \quad -30 \le x,y \le 30 \quad (41)$$

$$2.\ f_2(x,y) = 10\cdot(x^2-y)^2 + (x-1)^2\,; \quad -10 \le x,y \le 10 \quad (42)$$

$$3.\ f_3(x,y) = 1 + \frac{x^2+y^2}{100} - \cos(x+y)\,; \quad -20 \le x,y \le 20 \quad (43)$$

$$4.\ f_4(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2; \quad -20 \le x,y \le 20 \quad (44)$$

$$5.\ f_5(x,y) = (\mathrm{x} + 2\mathrm{y} - 7)^2 + (2\mathrm{x} + \mathrm{y} - 5)^2\,; \quad -20 \le x,y \le 20 \quad (45)$$

The number of the population used are 20 with a total of iterations are 100 times. The number of population is not too big because it considers the computation speed of the microcontroller used to do the optimization. Based on 100 attempts, the results of the Benchmark Function tests with some modifications to the PSO are presented. The *SR* values show the Success Rate of the 100 attempts conducted.

TABLE II
Benchmark Function Result of PSO Classic

PSO Classic			
Function f	Mean of *f*	Minimum of *f*	SR (%)
f_1	2.31E-16	3.57E-21	100
f_2	4.48E-08	2.00E-14	100
f_3	2.00E-03	2.22E-16	99
f_4	3.00E-03	6.55E-16	98
f_5	2.00E-15	4.98E-19	100

The results of testing with the Benchmark Function on the Classic PSO method are shown in Table II. The inertia

weight and learning factor used respectively are 0.6 and 1.5. The testing using the Griwank function f_3 and the Beale Function f_4 show that the *SR* are not 100% success. From all Benchmark function, the smallest error is the Sphere Function f_1 wih the value is 3.57E-21.

TABLE III
BENCHMARK FUNCTION RESULT OF SIGMOID FUNC.

PSO (w = Sigmoid Decreasing Function)			
Function f	Mean of *f*	Minimum of *f*	SR (%)
f_1	5.27E-22	0	100
f_2	7.62E-07	5.93E-15	100
f_3	5.09E-08	0	100
f_4	2.00E-03	5.08E-17	99
f_5	3.00E-13	1.85E-26	100

The results of testing with the Benchmark Function on the PSO using the sigmoid function when updating the inertia weight are shown in Table III. The testing using Beale Function f_4 shows that the *SR* value is not 100% success. From all Benchmark function, the smallest error are the Sphere Function f_1 and the Griwank Function f_3 wih the value are 0.

TABLE IV
BENCHMARK FUNCTION RESULT OF LDPSO

PSO (w = Linear Decreasing Function)			
Function f	Mean of *f*	Minimum of *f*	SR (%)
f_1	1.6E-12	6.7E-17	100
f_2	4.69E-08	4.82E-13	100
f_3	3.51E-10	6.22E-15	100
f_4	1.E-03	5.03E-14	99
f_5	1.52E-12	2.93E-17	100

The results of testing with the Benchmark Function on the PSO using the linear decreasing function when updating the inertia weight are shown in Table IV. The testing using Beale Function f_4 shows that the SR value is not 100% success. From all Benchmark function, the smallest error is the Booth Function f_5 wih the value is 2.93E-17.

TABLE V
BENCHMARK FUNCTION RESULT OF PSO PROPOSED

Modified PSO Proposed			
Function f	Mean of *f*	Minimum of *f*	SR (%)
f_1	0	0	100
f_2	7.57E-06	6.97E-18	100
f_3	2.54E-12	0	100
f_4	7.64E-05	5.87E-19	100
f_5	1.65E-23	0	100

The results of testing with the Benchmark Function on the Modified PSO purposed are shown in Table V. All of the testing using Benchmark Function shows that the *SR* are completely 100% success. From all Benchmark function, the smallest error are the Sphere Function f_1 , the Griwank Function f_3, and the Booth Function f_5 wih the value are 0. The Modified PSO proposed show the improvement of the error and success rate produced.

B. *Results of Simulation*

Simulations are built in the Matlab Simulink program by describing the balancing robots dynamic. The experiments are done by implementing the PSO and MPSO proposed to three PIDs that will control the angle, the distance, and the heading of balancing the robot.

1) *Simulation with IAE*

For the simulation, the desired position is set 10 meters with the balancing setpoint is 0 degree and the heating setpoint is 30 degree. The errors would be calculated by Integral Absolute Error (IAE) to evaluate the fitness function. The results of optimization using PSO–PID and MPSO–PID are shown as Fig.8.

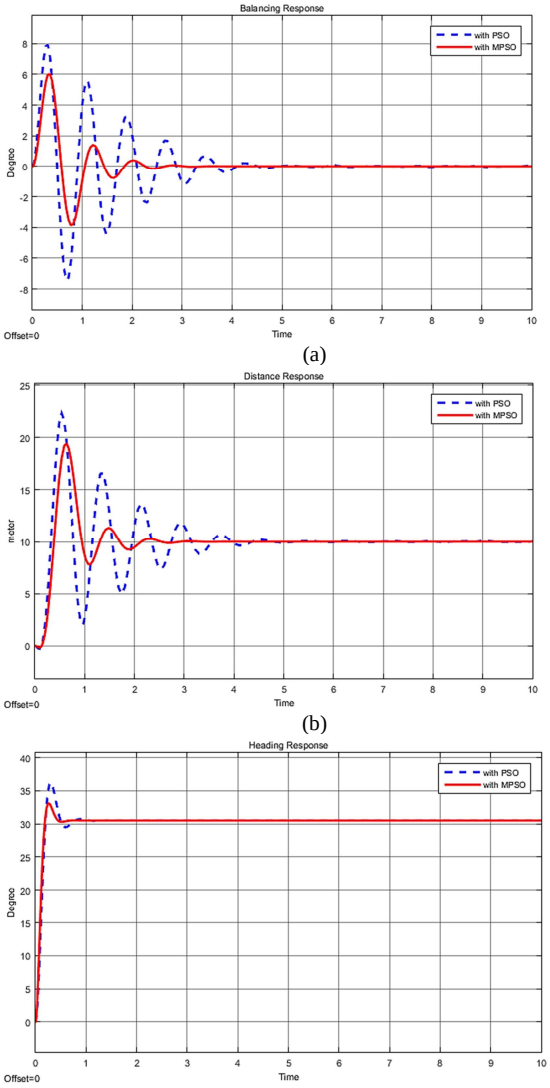

Fig. 8 Angle Response (a), Distance Response (b), and Heading response (c) of Balancing Robot with PSO and MPSO – PID – IAE

Fig. 8 shows the comparison between the responses using PID with PSO and MPSO. The results using both MPSO-PID and PSO-PID are the system of balancing robot can move to the desired position (x=10) and be able to be stably balanced. Both of them also be able to steer the plant to the desired heading (φ=30). The value of PID parameters are optimized by PSO (line: blue) and MPSO proposed (line: red). The responses using MPSO–PID have less overshoot and oscillation and reach stability faster than using PSO – PID.

2) Simulation with ISE

For comparison, the errors would be calculated by Integral Square Error (ISE) to evaluate the fitness function. For the second simulation, The desired position also is set 10 meters with the balancing setpoint is 0 degree, and the heading setpoint is 30 degree. The errors would be calculated by Integral Square Error (ISE) to evaluate the fitness function. The results of optimization using PSO–PID, and MPSO–PID are shown as Fig.9.

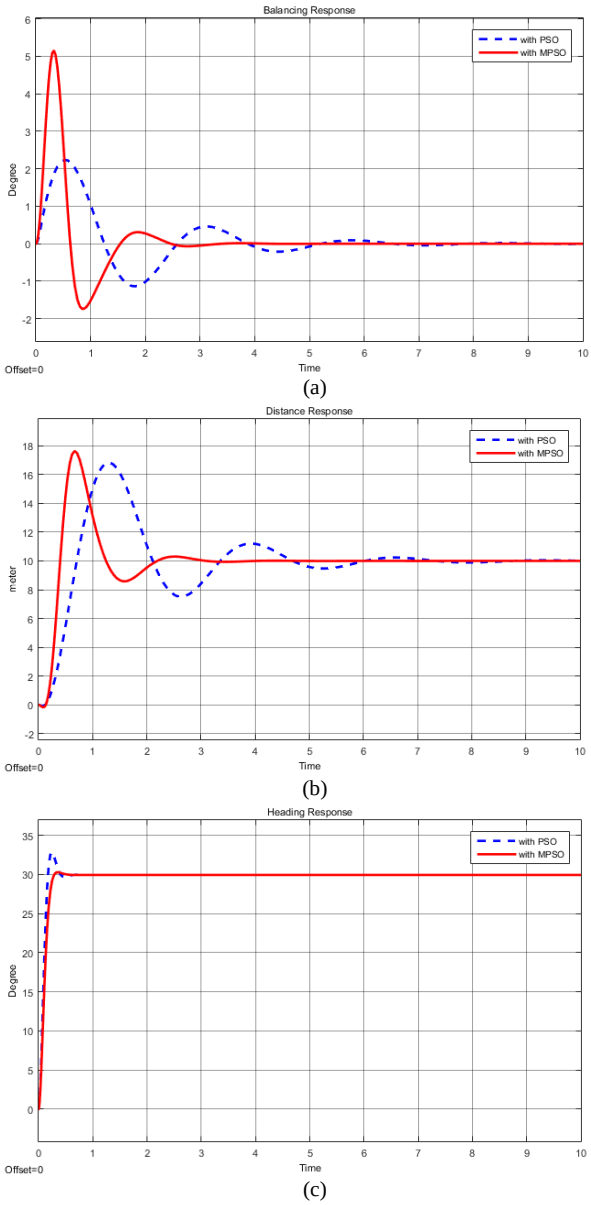

Fig. 9 Angle Response (a), Distance Response (b), and Heading response (c) of Balancing Robot with PSO and MPSO – PID – ISE

Fig. 9 shows the comparison between the responses using PID with PSO and MPSO. The results using both MPSO - PID and PSO – PID using ISE are similar to the results using IAE, but the responses using ISE have better performances than using IAE. These happen because if the system uses ISE, the error of system will be squared, so the value of error which is bigger than 1 then it will be bigger, and the values which is less than 1 then it will be smaller.

IV. Conclusions

The conclusion from this research is that Modified PSO proposed is capable enough to optimize multivariable function (verified by Benchmark Function test). For simulation on balancing robot, the MPSO – PID be able to control 3 DOF movement of the robot (balancing, distance, and heading) with less oscillation and faster responses (verified by simulations) than PSO – PID. By using the Integral Square Error (ISE) for evaluating the error, the results are better than using the Integral Absolute Error (IAE).

References

[1] D. Pratama, F. Ardilla, E. H. Binugroho, and D. Pramadihanto, "Tilt set-point correction system for balancing robot using PID controller," in *ICCEREC 2015 - International Conference on Control, Electronics, Renewable Energy and Communications*, 2015, pp. 129–135.

[2] D. Pratama, E. H. Binugroho, and F. Ardilla, "Movement control of two wheels balancing robot using cascaded PID controller," in *Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015*, 2016, pp. 94–99.

[3] E. H. Binugroho, D. Pratama, A. Z. R. Syahputra, and D. Pramadihanto, "Control for balancing line follower robot using discrete cascaded PID algorithm on ADROIT V1 education robot," in *Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015*, 2016, pp. 245–250.

[4] E. Sariyildiz, H. Yu, and K. Ohnishi, "A Practical Tuning Method for the Robust PID Controller with Velocity Feed-Back," *Machines*, vol. 3, no. 3, pp. 208–222, 2015.

[5] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[6] N. Jain, R. Gupta, and G. Parmar, "Intelligent Controlling of an Inverted Pendulum Using PSO-PID Controller," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 3712–3716, 2013.

[7] D. Yang, J. Chen, and N. Matsumoto, "Particle Swarm Optimization with Adaptive Parameters," in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 2007, pp. 616–621.

[8] M. Li, W. Du, and F. Nian, *An Adaptive Particle Swarm Optimization Algorithm Based on Directed Weighted Complex network*, vol. 2014. 2014.

[9] H. Mehdi and O. Boubaker, "Stabilization and tracking control of the inverted pendulum on a cart via a modified PSO fractional order PID controller," in *The inverted pendulum in control theory and robotics*, 2017, pp. 93–115.

[10] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE Congress on Evolutionary Computation*, 1998, pp. 69–73.

[11] Y. Shi and R. Eberhart, "Empirical Study of Particle Swarm Optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, 1999, pp. 1945–1950.

[12] R. F. Malik, T. A. Rahman, S. Z. M. Hashim, and R. Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight," *Int. J. Comput. Sci. Secur.*, vol. 1, pp. 35–44, 2007.

[13] Y. Gao, X. An, and J. Liu, "A Particle Swarm Optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation," *Int. Conf. Comput. Intell. Secur.*, vol. 1, pp. 61–65, 2008.

[14] K. Lei, Y. Qiu, and Y. He, "A New Adaptive Well-chosen Inertia Weight strategy to Automatically Harmonize Global and Local

Search Ability in Particle Swarm Optimization," in *First International Symposium on Systems and Control in Aerospace and Astronautics, Harbin*, 2006, pp. 977–980.
[15] Z. Li, C. Yang, and L. Fan, *Advanced control of wheeled inverted pendulum systems*, vol. 9781447129. 2013.
[16] J. Wu, X. Wang, and H. Wang, "Research on Two-Wheeled Self-Balancing Robot Control Strategy Based on LQR-Fuzzy Algorithm," *Int. J. Control Autom.*, vol. 9, pp. 31–40, 2016.
[17] K. Singh, "Modified PSO based PID Sliding Mode Control using Improved Reaching Law for Nonlinear systems," 2017, no. April.
[18] N. Hasanah, A. H. Alasiry, and B. Sumantri, "Two Wheels Line Following Balancing Robot Control using Fuzzy Logic and PID on Sloping Surface," in *2018 International Electronics Symposium on Engineering Technology and Applications, IES-ETA 2018 - Proceedings*, 2019, pp. 210–215.
[19] I. Dwisaputra, I. Sulistijono, and M. Nugraha, "Two Wheels Balancing Line Tracer Robot Using Fuzzy Logic Control," in *The 13th Industrial Electronics Seminar (IES)*, 2011, pp. 188–194.
[20] S. Kessentini and D. Barchiesi, "Particle Swarm Optimization with Adaptive Inertia Weight," *Int. J. Mach. Learn. Comput.*, vol. 5, no. 5, pp. 368–373, 2015.