Delta Robot Control Using Single Neuron PID Algorithms Based on Recurrent Fuzzy Neural Network Identifiers

Le Minh Thanh[1], Luong Hoai Thuong[1], Phan Thanh Loc[2], Chi-Ngon Nguyen[2]
[1] Vinh Long University of Technical Education, Vietnam
[2] Can Tho University, Vietnam
Email: thanhlm@vlute.edu.vn, thuonglh@vlute.edu.vn, locm3517004@gstudent.ctu.edu.vn, ncngon@ctu.edu.vn

***Abstract*—Parallel robot control is a topic that many researchers are still developing. This paper presents an application of single neuron PID controllers based on recurrent fuzzy neural network identifiers, to control the trajectory tracking for a 3-DOF Delta robot. Each robot arm needs a controller and an identifier. The proposed controller is the PID organized as a linear neuron, that the neuron's weights corresponding to Kp, Kd and Ki of the PID can be updated online during control process. That training algorithm needs an information on the object's sensitivity, called Jacobian information. The proposed identifier is a recurrent fuzzy neural network using to estimate the Jacobian information for updating the weights of the PID neuron. Simulation results on MATLAB/ Simulink show that the response of the proposed algorithm is better than using traditional PID controllers, with the setting time is about 0.3 ± 0.1 (s) and the steady-state error is eliminated.**

***Index Terms*—delta robot, single neural PID, recurrent fuzzy neural network, trajectory tracking**

Symbol	Unit	Meaning
$\theta_1, \theta_2, \theta_3$	Degrees	Angle of the upper leg of robot
R	mm	upper disc radius
R	mm	lower disc radius
L_1	mm	upper arm length
L_2	mm	lower arm length

Abbreviation

DOF	Degrees of freedom
PID	Proportional Integral Derivative
DC	Direct current
RFNN	Recurrent fuzzy neural network

I. Introduction

With flexible mechanisms, advantages of speed and force, and precision delta robots have become popular and widely used in industry [1]. The complex structure of this robot makes them an interesting in research focus. Delta robots were proposed in 1939, when Pollard built a robot to control the position of a spray gun [2]. In this context, other robots with the same structure have been implemented. For example, a robot proposed by Stewart with two platforms ensures fixed stability in a stationary facility [3]. In 1985, a delta robot was developed and built in the Ecole Polytechnique Federale de Lausanne (EPFL) called delta robots focused on industrial work [4]. Based on this robot, the new architecture is implemented according to the necessary characteristics in industry and school. For example, it is a robot with high accuracy but slow motion is widely used in 3D printers [5]. In the industrial sector the need to optimize production has been a major challenge for robotics companies since the 1980.

Delta robots have been successfully researched and manufactured in many countries. However, the high cost and operational control of delta robots has always been an interesting topic for many innovative studies.

So that, the neural networks and fuzzy logic are applied for improving adaptive PID controllers for Delta robot [6]. A non-parametric identifier of each robot arm using recurrent fuzzy neural network is built and trained online during control to estimate the object's sensitivity to the input signal, also known as Jacobian information. Based on the Jacobian information, a single-neuron-adaptive PID controller will be updated online with three connected weighs respectively three parameters K_p, K_i and K_d of the controller. Thus, with this principle, the PID controller will be automatically adapted due to the variation of the robot that classical control solutions can not achieve.

In the process of making efforts for manufacturing delta robots to meet the industrial needs, this paper aims to control and conduct analysis, comparison, and evaluation of different algorithms. The comparison and evaluation of efficiencies between traditional PID controllers and fuzzy-neural based PID controllers are implemented and tested in MATLAB/Simulink which the absolute error values are used to evaluate the performances of the closed loop system.

II. Delta Robot

A. Parallel Robot Model

The model of delta robot is presented in Fig. 1 [7]-[9]. In this model, B_iD_i stitching is modeled into two material points located at B_i and D_i. Each of them has a weight m_b

Manuscript received January 16, 2020; revised August 15, 2020.
Corresponding author: Chi-Ngon Nguyen

doi: 10.18178/ijmerr.9.10.1411-1418

and is connected by rigid, weightless rods. Thus, the dynamic model of this model consists of 4 solid objects, in which the A_iB_i (i = 1, 2, 3) stitches move around the axes perpendicular to the OA_iB_i plane. At A_i with mass m_1 and remaining solid objects (including three points mounted at D_i) with mass (m_p+3m_b), can be considered as a moving table of linear motion. Three points located at B_i having mass m_p, where m_p is the operation mass.

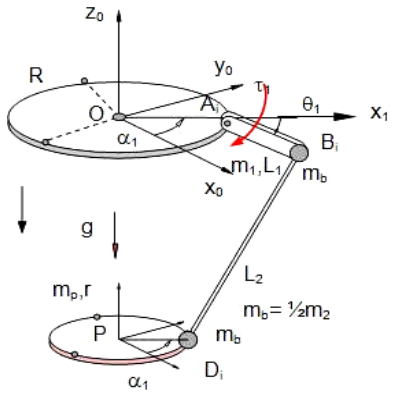

Figure 1. Parallel robot model 3RUS [9].

The set of robot actuating includes:

$$q=\left[\theta_1\theta_2\theta_3 x_P y_P z_P\right].$$

B. Establishing the Linking Equations

From Fig. 1, we have the linking equation for points B_1 and D_1 as follows:

$$l^2-\left(r_{D_1}-r_{B_1}\right)^T\left(r_{D_1}-r_{B_1}\right)=0. \quad (1)$$

where r_{D_1}, r_{B_1} are the positioning vectors of points B_1 and D_1 in the Oxyz coordinate system, calculated according to the vector equation:

$$r_{B_1}=r_{A_1}+U_{A_1B_1}. \quad (2)$$

In which, the coordinates vector r_{A_1}, $U_{A_1B_1}$ in the coordinate system $Ox_1y_1z_1$ have the forms:

$$\begin{aligned} r_{A_1}&=[R00]^T\\ U_{A_1B_1}&=[l_1\cos\theta_1 0-l_1\sin\theta_1] \end{aligned} \quad (3)$$

Replace (3) with (2) and get the r_{B1} coordinate in $Ox_1y_1z_1$ coordinate system

$$r_{B1}=\left[R+l_1\cos\theta_1 0-l_1\sin\theta_1\right] \quad (4)$$

The cosine matrix indicates the direction of the $Ox_1y_1z_1$ coordinate system with the Oxyz coordinate system as:

$$A_z(\alpha_1)=\begin{bmatrix}\cos\alpha_1 & -\sin\alpha_1 & 0\\ \sin\alpha_1 & \cos\alpha_1 & 0\\ 0 & 0 & 1\end{bmatrix} \quad (5)$$

Infer the coordinates of the r_{B1} vector in the Oxyz coordinate system:

$$r_{B_1}=\begin{bmatrix}\cos\alpha_1 & -\sin\alpha_1 & 0\\ \sin\alpha_1 & \cos\alpha_1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}R+l_1\cos\theta_1\\ 0\\ -l_1\sin\theta_1\end{bmatrix}= \begin{bmatrix}R\cos\alpha_1+l_1\cos\alpha_1\cos\theta_1\\ R\sin\alpha_1+l_1\sin\alpha_1\cos\theta_1\\ -l_1\sin\theta_1\end{bmatrix} \quad (6)$$

The coordinates of vector r_{D1} are calculated as follows:

$$r_{D_1}=r_P+u_{PD_1}= \begin{bmatrix}x_P\\ y_P\\ z_P\end{bmatrix}+\begin{bmatrix}cos\alpha_1 & -sin\alpha_1 & 0\\ sin\alpha_1 & cos\alpha_1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}r\\ 0\\ 0\end{bmatrix}= \begin{bmatrix}x_P+cos\alpha_1 r\\ y_P+sin\alpha_1 r\\ z_P\end{bmatrix} \quad (7)$$

Combining equations (6) and (7) we have:

$$r_{D_1}-r_{B_1}=\begin{bmatrix}cos\alpha_1(R-r)+l_1cos\alpha_1cos\theta_1-x_P\\ sin\alpha_1(R-r)+l_1sin\alpha_1cos\theta_1-y_P\\ -l_1sin\theta_1-z_P\end{bmatrix} \quad (8)$$

Substituting (8) into (1), we get the equation linked to the first pin. Similarly, with the second and third legs, we get the link equation of the robot as follows:

$$\begin{aligned} f_1&=l_2^2-\left(\cos\alpha_1(R-r)+l_1\cos\alpha_1\cos\theta_1-x_P\right)^2\\ &-\left(\sin\alpha_1(R-r)+l_1\sin\alpha_1\cos\theta_1-y_P\right)^2-\left(l_1\sin\theta_1+z_P\right)^2=0\\ f_2&=l_2^2-\left(\cos\alpha_2(R-r)+l_1\cos\alpha_2\cos\theta_2-x_P\right)^2\\ &-\left(\sin\alpha_2(R-r)+l_1\sin\alpha_2\cos\theta_2-y_P\right)^2-\left(l_1\sin\theta_2+z_P\right)^2=0\\ f_3&=l_2^2-\left(\cos\alpha_3(R-r)+l_1\cos\alpha_3\cos\theta_3-x_P\right)^2\\ &-\left(\sin\alpha_3(R-r)+l_1\sin\alpha_3\cos\theta_3-y_P\right)^2-\left(l_1\sin\theta_3+z_P\right)^2=0 \end{aligned} \quad (9)$$

C. The Kinetic Energy and Potential of the Robot

The kinetic energy of the A_iB_i stages is calculated as

$$T_{A_iB_i}=\frac{1}{2}I_{1y}\omega_{A_iB_i}^2=\frac{1}{2}I_{1y}\dot{\theta}_i^2 \quad (10)$$

The kinetic energy of mass m_b is set at B_1 as follow

$$T_{m_b}=\frac{1}{2}m_b v_{B_i}^2=\frac{1}{2}m_b l_1^2\dot{\theta}_i^2 \quad (11)$$

The kinetic energy of a moving table and m_b masses is

$$T_3=\frac{1}{2}(m_P+3m_b)v_P^2=\frac{1}{2}(m_P+3m_b)\left(\dot{x}_P^2+\dot{y}_P^2+\dot{z}_P^2\right) \quad (12)$$

Combining the kinetic expressions above, we get the model kinetic expression of the robot:

$$\begin{aligned} T&=\frac{1}{2}\left(I_{1y}+m_b l_1^2\right)\left(\dot{\theta}_1^2+\dot{\theta}_2^2+\dot{\theta}_3^2\right)\\ &+\frac{1}{2}(m_P+3m_b)\left(\dot{x}_P^2+\dot{y}_P^2+\dot{z}_P^2\right) \end{aligned} \quad (13)$$

The potential energy of a robot is calculated as follows:

$$\pi = -gl_1\left(\frac{1}{2}m_1 + m_b\right)\left(sin\theta_1 + sin\theta_2 + sin\theta_3\right) + g\left(3m_b + m_P\right)z_P \tag{14}$$

D. Set the Differential Equation of Motion of the Delta Robot

We use the Lagrange factor to set the differential equation of motion of this model with helium links with the following form:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_k}\right) - \frac{\partial T}{\partial q_k} = Q_k - \sum_{i=1}^{r}\lambda_i\frac{\partial f_i}{\partial q_k}\ \left(k = 1, 2, \ldots, m\right) \tag{15}$$

where q_k is the extrapolation coordinates of the robot, f_i is the linking equations, Q_k is the extrapolation force, λ_i is the Lagrange factor. With this model, the vector of extrapolation coordinates $\theta \in R^6$ and the number of associated equations is three, so m=6, r= 3. We divide the forces acting on the robot into potential forces and forces without potential energy, the extrapolation force Q_k is calculated as follows

$$Q_k = -\frac{\partial \pi}{\partial q_k} + Q_k^{np} \tag{16}$$

where Q_k^{np} are extrapolation forces corresponding to forces without potential energy. Sum of extrapolation forces is

$$\delta A = \tau_1\delta\theta_1 + \tau_2\delta\theta_2 + \tau_3\delta\theta_3\,. \tag{17}$$

So, we have

$Q_1^{np} = \tau_1$, $Q_2^{np} = \tau_2$, $Q_3^{np} = \tau_3$ and

$Q_k^{np} = 0$ with k = 4, 5, 6.

Substituting kinetic and potential energies into (15), we get the motion equations of the robot as follows:

$$\left(I_{Iy} + m_b l_1^2\right)\ddot{\theta}_1 = gl_1\left(\frac{1}{2}m_1 + m_b\right)cos\theta_1 + \tau_1 - 2\lambda_1 l_1\begin{pmatrix} sin\theta_1\left(R-r\right) - cos\alpha_1 sin\theta_1 x_p \\ -sin\alpha_1 sin\theta_1 y_p - cos\theta_1 z_p \end{pmatrix} \tag{18}$$

$$\left(I_{Iy} + m_b l_1^2\right)\ddot{\theta}_2 = gl_1\left(\frac{1}{2}m_1 + m_b\right)cos\theta_2 + \tau_2 - 2\lambda_1 l_1\begin{pmatrix} sin\theta_2\left(R-r\right) - cos\alpha_2 sin\theta_2 x_p \\ -sin\alpha_2 sin\theta_2 y_p - cos\theta_2 z_p \end{pmatrix} \tag{19}$$

$$\left(I_{Iy} + m_b l_1^2\right)\ddot{\theta}_3 = gl_1\left(\frac{1}{2}m_1 + m_b\right)cos\theta_3 + \tau_3 - 2\lambda_3 l_1\begin{pmatrix} sin\theta_3\left(R-r\right) - cos\alpha_3 sin\theta_3 x_p \\ -sin\alpha_3 sin\theta_3 y_p - cos\theta_3 z_p \end{pmatrix} \tag{20}$$

$$\left(m_p + 3m_b\right)\ddot{x}_p = -2\lambda_1\left(cos\alpha_1\left(R-r\right) + l_1 cos\alpha_1 cos\theta_1 - x_p\right) - 2\lambda_2\left(cos\alpha_2\left(R-r\right) + l_1 cos\alpha_2 cos\theta_2 - x_p\right) - 2\lambda_3\left(cos\alpha_3\left(R-r\right) + l_1 cos\alpha_3 cos\theta_3 - x_p\right) \tag{21}$$

$$\left(m_p + 3m_b\right)\ddot{y}_p = -2\lambda_1\left(\sin\alpha_1\left(R-r\right) + l_1\sin\alpha_1 cos\theta_1 - y_p\right) - 2\lambda_2\left(\sin\alpha_2\left(R-r\right) + l_1\sin\alpha_2 cos\theta_2 - y_p\right) - 2\lambda_3\left(\sin\alpha_3\left(R-r\right) + l_1\sin\alpha_3 cos\theta_3 - y_p\right) \tag{22}$$

$$\left(m_p + 3m_b\right)\ddot{z}_p = -\left(3m_b + m_p\right)g + 2\lambda_1\left(z_p + l_1 sin\theta_1\right) + 2\lambda_2\left(z_p + l_1 sin\theta_2\right) + 2\lambda_3\left(z_p + l_1 sin\theta_3\right) \tag{23}$$

$$l_1^2 - \left(cos\alpha_1\left(R-r\right) + l_1 cos\alpha_1 cos\theta_1 - x_p\right)^2 - \left(sin\alpha_1\left(R-r\right) + l_1 sin\alpha_1 cos\theta_1 - y_p\right)^2 - \left(l_1 sin\theta_1 + z_p\right)^2 = 0 \tag{24}$$

$$l_2^2 - \left(cos\alpha_2\left(R-r\right) + l_1 cos\alpha_2 cos\theta_2 - x_p\right)^2 - \left(sin\alpha_2\left(R-r\right) + l_1 sin\alpha_2 cos\theta_2 - y_p\right)^2 - \left(l_1 sin\theta_2 + z_p\right)^2 = 0 \tag{25}$$

$$l_3^2 - \left(cos\alpha_3\left(R-r\right) + l_1 cos\alpha_3 cos\theta_3 - x_p\right)^2 - \left(sin\alpha_3\left(R-r\right) + l_1 sin\alpha_3 cos\theta_3 - y_p\right)^2 - \left(l_1 sin\theta_3 + z_p\right)^2 = 0 \tag{26}$$

From the parallel motion equations of robots, we get the model of robot as shown in Fig. 2 and Fig. 3.

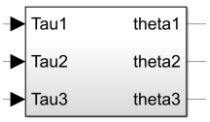

Figure 2. Model of Delta robot in Simulink.

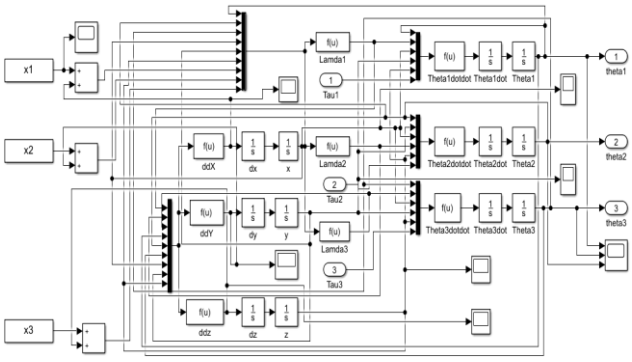

Figure 3. Inside of Parallel robot model.

III. CONTROLLER DESIGN

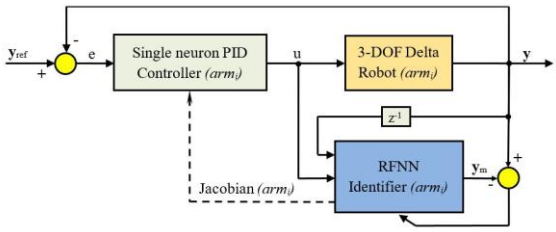

Figure 4. Controller structure.

The structure of the control system for each robot arm is presented in Fig. 4.

A. Single Neural Adaptive PID Controller

1) Controller structure

The PID controller is built by a linear neuron with 03 inputs and zero trigger threshold [10] as shown in Fig. 5. The input of neuron receives 03 corresponding values as proportional, integral and differential components of the difference between response and reference signal as (27).

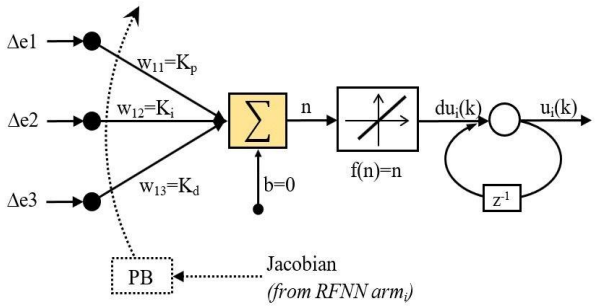

Figure 5. Structure of a PID controller adapted to a neuron.

$$\Delta e1 = e(k); \Delta e2 = \int_0^{\infty} e(k)dk; \Delta e3 = {de(k)}/{dk} \quad (27)$$

The PID controller is set as follows:

$$u(k) = u(k-1) + K_p \Delta e1 + K_i \Delta e2 + K_d \Delta e3 \quad (28)$$

where, *e(k)* is the difference between the reference signal and the system response.

With the structure of a single neuron PID controller in Fig. 5, the output of the neuron is also the output of the PID controller, as shown in (29).

$$\begin{aligned} n &= (w_{11}\Delta e1 + w_{12}\Delta e2 + w_{13}\Delta e3) \\ du(k) &= f(n) = n \\ u(k) &= u(k-1) + du(k) \end{aligned} \quad (29)$$

In which, $w_{1i \mid I = 1,2,3}$ are the weights of the neurons, which are the parameter sets (K_P, K_i, K_d) of the PID controller and they are updated online during the control process.

2) Controller training

The goal of training the single neuron PID controller is to adjust the network's weight set $w_{1i\ (i=1,2,3)}$ to minimize the cost function (30).

$$E(k) = \frac{1}{2}e^2(k) = \frac{1}{2}\left[y_{ref}(k) - y(k)\right]^2 \quad (30)$$

where $y_{ref}(k)$ is the reference signal and y(k) is the system response.

To adjust the weight set $w_{1i \mid i = 1,2,3,}$ the gradient descent method was applied:

$$\begin{aligned} K_p &= w_{11}(k+1) = w_{11}(k) + \Delta w_{11}(k) \\ K_i &= w_{12}(k+1) = w_{12}(k) + \Delta w_{12}(k) \\ K_d &= w_{13}(k+1) = w_{13}(k) + \Delta w_{13}(k) \end{aligned} \quad (31)$$

where $\Delta w_{1,i}(k)_{\ |i=1,2,3}$ are gradients defined by (32-34), successfully verified by Zhang *et al.* [11]:

$$\begin{aligned} \Delta w_{11}(k) &= \eta^{Kp}\left(-\frac{\partial E(k)}{\partial w_{11}(k)}\right) \\ &= -\eta^{Kp}\frac{\partial E(k)}{\partial y(k)}\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial w_{11}(k)} = \eta^{Kp} e(k)\frac{\partial y(k)}{\partial u(k)}\Delta e1 \end{aligned} \quad (32)$$

$$\begin{aligned} \Delta w_{12}(k) &= \eta^{Ki}\left(-\frac{\partial E(k)}{\partial w_{12}(k)}\right) \\ &= -\eta^{Ki}\frac{\partial E(k)}{\partial y(k)}\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial w_{12}(k)} = \eta^{Ki} e(k)\frac{\partial y(k)}{\partial u(k)}\Delta e2 \end{aligned} \quad (33)$$

$$\begin{aligned} \Delta w_{13}(k) &= \eta^{Kd}\left(-\frac{\partial E(k)}{\partial w_{13}(k)}\right) \\ &= -\eta^{Kd}\frac{\partial E(k)}{\partial y(k)}\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial w_{13}(k)} = \eta^{Kd} e(k)\frac{\partial y(k)}{\partial u(k)}\Delta e3 \end{aligned} \quad (34)$$

With $\eta^{k}_{\ |k=Kp,Ki,Kd}$ are the learning rate constants; *Δe1*, *Δe1* and *Δe3* determined by (27); ${\partial y(k)}/{\partial u(k)}$ is the response sensitivity to the control signal, also known as Jacobian information, determined through recurrent fuzzy neural network identifiers, which will presented in next section.

Three neuron-adaptive PID controllers are set up by MATLAB's three S-function, fully compatible with Simulink's standard library, a single-neuron-adapted PID controller depicted in Fig. 6 and two sets The remaining is built in the same way as the SingleNeural_PID1.

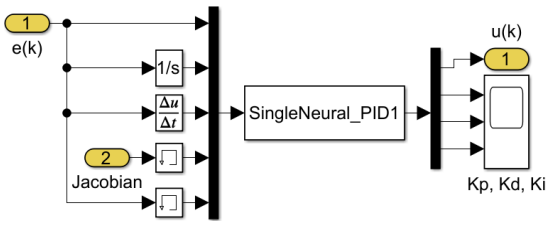

Figure 6. Single neural PID controller.

B. Recurrent Fuzzy Neural Newtwork Identifier

1) Identifier structure

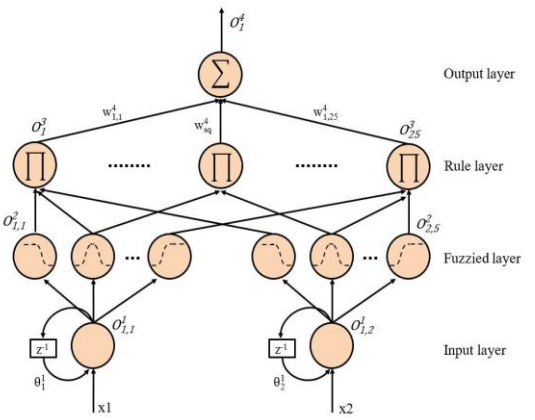

Figure 7. Diagram of the RFNN identifier.

The identifier is a recurrent fuzzy neural network (RFNN - Fig. 7). The RFNN identifier has 4 layers, with an input layer of 2 nodes, a fuzzy layer of 10 nodes, a fuzzy rule class of 25 nodes, and an output layer with 1 node. The structure of the RFNN identifier can be described as follows [12]:

Layer 1 - Input layer: This layer consists of 2 nodes that convey the input values to the next layer. Here feedback links are added to increase the responsiveness of the network. The output of layer 1 is described as (35):

$$O_i^1(k) = x_i^1(k) + \theta_i^1 O_i^k(k-1), \quad i = 1,2 \tag{35}$$

With θ_i^1 is the connection weight at the current time *k*. The input of the corresponding RFNN identifier is the current control signal and the past output of the response:

$$\begin{aligned} x_1^1(k) &= u(k) \\ x_2^1(k) &= y(k-1) \end{aligned} \tag{36}$$

Layer 2 - Fuzzy layer: This layer consists of (2x5) nodes, each node representing a related function of the Gaussian form with mean value m_{ij} and standard deviation σ_{ij}, and defined as (37).

$$O_{ij}^2(k) = \exp\left\{-\frac{\left(O_i^1(k) - m_{ij}\right)^2}{\left(\sigma_{ij}\right)^2}\right\}, i = 1,2;\ j = 1,2,...,5 \tag{37}$$

At each node on the fuzzy layer, there are 2 parameters that are automatically adjusted during the online training of the RFNN identifier, that is m_{ij} and σ_{ij}.

Layer 3 - Law class: This layer consists of (5x5) nodes. The output of the *q* node in this class is determined as follows:

$$O_q^3(k) = \prod_i O_{iq_i}^2(k),\ i = 1,2,...,5; q_i = 1,2,...,5 \tag{38}$$

Layer 4 - Output layer: This layer includes 1 linear neuron with the defined output as follows:

$$O_i^4(k) = \sum_j w_{ij}^4 O_j^3(k),\ i = 1;\ j = 1,2,...,25 \tag{39}$$

where w_{ij}^4 is the connecting weights from 3rd layer to 4th layer. The output of this layer is also the output of the RFNN identifier:

$$\begin{aligned} O_1^4(k) = y_m(k) &= \hat{f}\left[x_1(k), x_2(k)\right] \\ &= \hat{f}\left[u(k), y(k-1)\right] \end{aligned} \tag{40}$$

2) Training the RFNN identitifiers

The goal of an online training algorithm for RFNN identifier is to adjust the weighting sets of the network and the parameters of the fuzzy class dependent functions to achieve the minimum value of cost function (41):

$$E(k) = \frac{1}{2}\left[y(k) - y_m(k)\right]^2 = \frac{1}{2}\left[y(k) - O_1^4(k)\right]^2 \tag{41}$$

Using back-propagation technique, RFNN's set of connection weights will be adjusted according to the following:

$$W(k+1) = W(k) + \Delta W(k) = W(k) + \eta\left(-\frac{\partial E(k)}{\partial W}\right) \tag{42}$$

In which, η is the learning rate constant and *W* is the parameter to be adjusted during the training of the RFNN identifier.

Given $e(k)=y(k)-y_m(k)$ and $W=[\theta,\ m,\ \sigma,\ w]^T$ are the error and the connection weight vector of the RFNN identifier, then the gradient of *E(.)* in (41) according to *W* is determined as follows:

$$\frac{\partial E(k)}{\partial W} = -e(k)\frac{\partial y_m(k)}{\partial W} = -e(k)\frac{\partial O_1^4(k)}{\partial W} \tag{43}$$

With this principle, the weight of each RFNN network layer is updated as follows [13]:

$$w_{ij}^4(k+1) = w_{ij}^4(k) + \eta^w\left(-\frac{\partial E(k)}{\partial w_{ij}^4}\right) = w_{ij}^4(k) + \eta^w e(k) O_i^3 \tag{44}$$

$$\begin{aligned} m_{ij}(k+1) &= m_{ij}(k) + \eta^m\left(-\frac{\partial E(k)}{\partial m_{ij}}\right) \\ &= m_{ij}(k) + \eta^m \sum_k e(k) w_{ik}^4 O_k^3 \frac{2\left[O_{ij}^1(k) - m_{ij}\right]}{\left(\sigma_{ij}\right)^2} \end{aligned} \tag{45}$$

$$\begin{aligned} \sigma_{ij}(k+1) &= \sigma_{ij}(k) + \eta^\sigma\left(-\frac{\partial E(k)}{\partial \sigma_{ij}}\right) \\ &= \sigma_{ij}(k) + \eta^\sigma \sum_k e(k) w_{ik}^4 O_k^3 \frac{2\left[O_{ij}^1(k) - m_{ij}\right]^2}{\left(\sigma_{ij}\right)^3} \end{aligned} \tag{46}$$

$$\begin{aligned} \theta_i^1(k+1) &= \theta_i^1(k) + \eta^\theta\left(-\frac{\partial E(k)}{\partial \theta_i^1}\right) \\ &= \theta_i^1(k) + \eta^\theta \sum_k e(k) w_{ik}^4 O_k^3 \frac{(-2)\left[O_{ij}^1(k) - m_{ij}\right] O_{ij}^1(k-1)}{\left(\sigma_{ij}\right)^2} \end{aligned} \tag{47}$$

where $\eta^s{}_{|s=w,m,\sigma,\theta}$ are the corresponding learning rate constants. In addition, to estimate the output of the object model $y_m(k)$, the RFNN identifier must also estimate Jacobian information $\partial y(k) \big/ \partial u(k)$ for online training of the single neuron PID controller. Jacobian information is determined as follows [14]-[19]:

$$\begin{aligned} \frac{\partial y(k)}{\partial u(k)} &= \frac{\partial O_1^4}{\partial u} = \sum_{q=1}^{25}\left\{\frac{\partial O^4}{\partial O_q^3} \cdot \frac{\partial O_q^3}{\partial u}\right\} = \sum_{q=1}^{25} w_{ij}^4 \cdot \left\{\frac{\partial O_q^3}{\partial u}\right\} \\ &= \sum_q w_{ij}^4 \cdot \left\{\sum_s \frac{\partial O_q^3}{\partial O_{qs}^2} \cdot \frac{\partial O_{qs}^2}{\partial u}\right\} \\ &= \sum_q w_{ij}^4 \cdot \left\{\sum_s \frac{\partial O_q^3}{\partial O_{qs}^2} \cdot \frac{(-2)\left[O_{ij}^1(k) - m_{ij}\right]}{\left(\sigma_{ij}\right)^2}\right\} \end{aligned} \tag{48}$$

The RFNN identifier for each robot arm is built in MATLAB's S-function as Fig. 8.

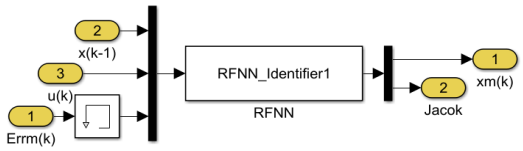

Figure 8. The RFNN identifier in Simulink.

$$\hat{y}(k) = y_m(k) = x_m(k) = f\left[x(k-1), u(k)\right] \tag{49}$$

IV. Specifications and Simulation Results

A. Simulation Parameters

The specifications of the robot are shown in Table I [6]. Parameters of the single neuron PID controller are given in Table II. And the parameters of RFNN identifiers are presented in Table III.

TABLE I. Delta Robot Mechanical Specifications

Symbol	Value	Unit
L_1	0.3	m
L_2	0.8	m
R	0.26	m
r	0.04	m
α_1	0	rad/s
α_2	$2\pi/3$	rad/s
α_3	$4\pi/3$	rad/s
m_1	0.42	kg
m_b	0.2	kg
m_p	0.75	kg

TABLE II. Parameters of Adaptive Neural PID Controller

Symbol	Meaning	Value
$\left[w_{K_p}\ w_{K_I}\ w_{K_D}\right]^T$	Initial parameters of the PID neuron	$[800\ 100\ 150]^T$
$\left[\eta_{K_p}\ \eta_{K_I}\ \eta_{K_D}\right]^T$	Learning rates of the PID neuron	$[0.2\ 0.1\ 0.1]^T$

TABLE III. Parameters of Three RFNN Identifiers

Symbol	Meaning	Value
$[\eta_1;\eta_2;\eta_3]$	The learning factor of the neuron network	$[0.01\ 0.01\ 0.01]^T$
c_{i1}	RFNN1 network center vector	$\begin{bmatrix}-1.5 & -0.75 & 0 & 0.75 & 1.5\\ -1.5 & -0.75 & 0 & 0.75 & 1.5\end{bmatrix}$
c_{i2}	RFNN2 network center vector	$\begin{bmatrix}-1.35 & -0.675 & 0 & 0.675 & 1.35\\ -1.35 & -0.675 & 0 & 0.675 & 1.35\end{bmatrix}$
c_{i3}	RFNN3 network center vector	$\begin{bmatrix}-3.05 & -1.525 & 0 & 1.525 & 3.05\\ -3.05 & -1.525 & 0 & 1.525 & 3.05\end{bmatrix}$
b_{i1}	Activation threshold 1	$\begin{bmatrix}0.5 & 0.5 & 0 & 0.5 & 0.5\\ 0.5 & 0.5 & 0 & 0.5 & 0.5\end{bmatrix}$
b_{i2}	Activation threshold 2	$\begin{bmatrix}0.5 & 0.5 & 0 & 0.5 & 0.5\\ 0.5 & 0.5 & 0 & 0.5 & 0.5\end{bmatrix}$ $[0.1\ 0.1\ \cdots\ ]^T$
b_{i3}	Activation threshold 3	$\begin{bmatrix}0.5 & 0.5 & 0 & 0.5 & 0.5\\ 0.5 & 0.5 & 0 & 0.5 & 0.5\end{bmatrix}$
$[w_1]^T$	Weighted vector 1 (25×1)	$[0.1\ 0.1\ \cdots\ 0.1\ 0.1]^T$
$[w_2]^T$	Weighted vector 2 (25×1)	$[0.15\ 0.15\ \cdots\ 0.15\ 0.15]^T$
$[w_3]^T$	Weighted vector 3(25×1)	$[0.1\ 0.1\ \cdots\ 0.1\ 0.1]^T$
w_{i1}	Weighted vector RFNN1	$\begin{bmatrix}0.1 & 0.1 & 0 & 0.1 & 0.1\\ 0.1 & 0.1 & 0 & 0.1 & 0.1\end{bmatrix}$
w_{i2}	Weighted vector RFNN2	$\begin{bmatrix}0.125 & 0.125 & 0.125 & 0.125 & 0.125\\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125\end{bmatrix}$
w_{i3}	Weighted vector RFNN3	$\begin{bmatrix}0.15 & 0.15 & 0.15 & 0.15 & 0.15\\ 0.15 & 0.15 & 0.15 & 0.15 & 0.15\end{bmatrix}$
i	Network inputs	2

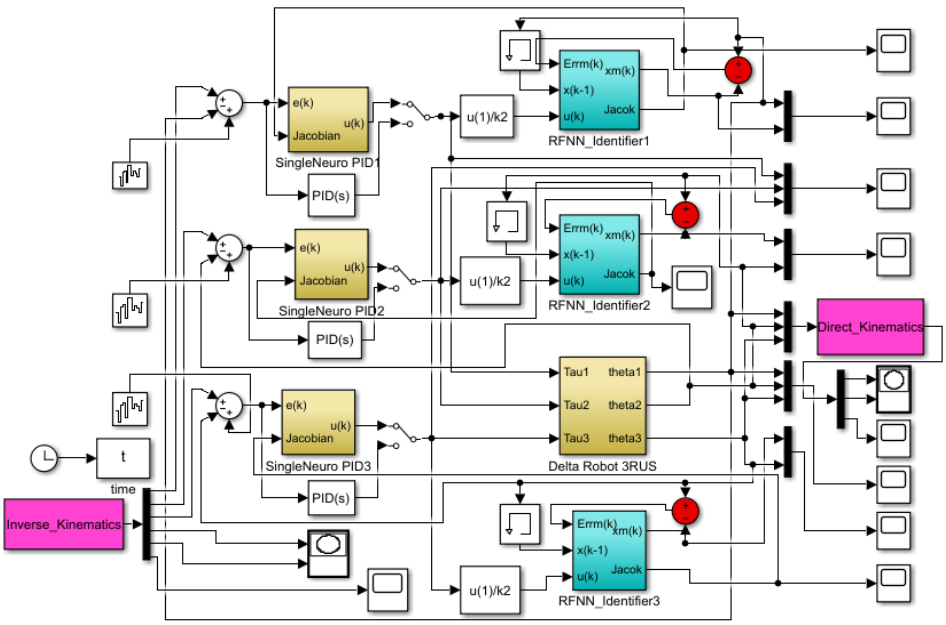

Figure 9. Delta parallel robot controller diagram in MATLAB / Simulink.

B. Simulation Results

The control system for the Delta robot is illustrated in Fig. 9 with the desired trajectory as (50).

$$\begin{aligned} x_d &= 0.17\sin(2\pi t)+0.3 \\ y_d &= 0.17\cos(2\pi t)+0.2 \\ z_d &= -0.7(m) \end{aligned} \tag{50}$$

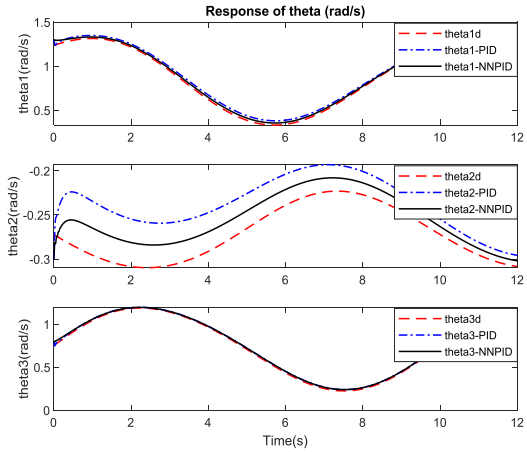

Figure 10. Comparison of traditional PID and single neuron PID controllers.

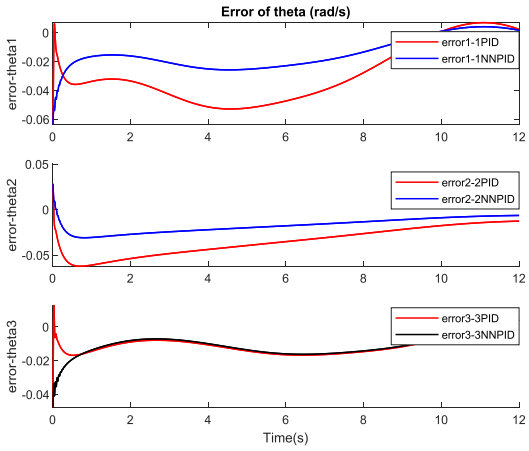

Figure 11. Errors of responses.

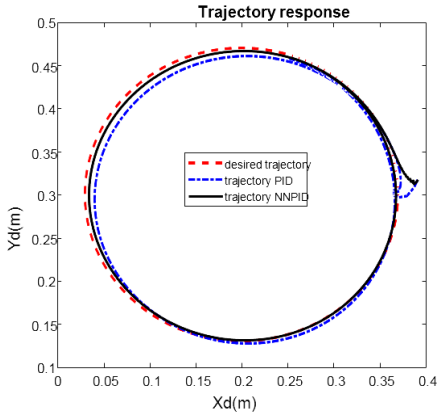

Figure 12. Trajectory tracking of traditional PID and single neural PID controller.

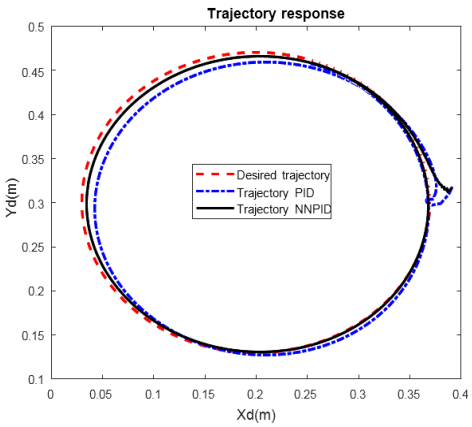

Figure 13. Responses when changing load from 3.21 Kg to 4.11 Kg.

TABLE IV. SYSTEM QUALITY STANDARDS

Quality indicators	PID	Neuron PID
Settling time	0.4±0.001(s)	0.3±0.001(s)
Overshoot	1.991 (%)	1.97 (%)
Rising time	2.535 s	2.688 s

V. CONCLUSION

This paper has applied single neural PID controllers based on RFNN identifiers to control a 3-DOF delta robot, a nonlinear MIMO system. Each robot arm is controlled by a single neural PID controller that can be online training with the Jacobian information of that arm given by a RFNN identifier. Simulation results show that the controllers and identifiers can be updated online during control action. The quality standards of improved system responses better than using traditional PID controllers (Table IV). The proposed algorithm is stable and fast response when simulating on Delta robot. In next step, the controllers will be tested on a real robot system.

The response of the traditional PID and the single neuron PID to the reference trajectory are presented in Fig. 10, Fig. 11, Fig. 12, and Fig. 13 with changing load. Simulation results show that single neural PID controllers are better than traditional PID controllers with the setting times are about 0.3±0.1 seconds, the steady-state errors are eliminated.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Mr. Le Minh Thanh, first author, is a PhD candidate under supervising of Prof. Chi-Ngon Nguyen (last and corresponding author), who has prepared the manuscript. Mr. Luong Hoai Thuong, 2nd author, has helped on correcting the paper. Mr. Phan Thanh Loc, 3rd author, is a Master's degree student, also supervised by Prof. Nguyen, has built the simulation model. And Prof. Nguyen is chef

of research group, who has supervised for this study and finalized this paper.

REFERENCES

[1] J. Merlet, *Parallel Robots*. P.O. Box 17, 3300 AA Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000.

[2] P. W. L. V, *Position-Controlling Apparatus*, Patent US2 286 571 A, June, 16, 1942.

[3] D. Stewart, *A platform with Six Degrees of Freedom*, 1991.

[4] R. Clavel, *Conception D'Un Robot Parallele Rapide a 4 Degres de Liberte*, Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, Lausanne.

[5] M. Bouri and R. Clavel (2010), *The Linear Delta: Developments and Applications*, *in* ISR 2010 (41st Inter. Symposium on Robotics and ROBOTIK 2010 (6th German Conference on Robotics), Munich, Germany, *2010*, pp. 1-8.

[6] W. Widhiada, T. G. T. Nindhia, and N. Budiarsa, "Robust control for the motion five fingered robot gripper," *International Journal of Mechanical Engineering and Robotics Research*, vol. 4, no. 3, pp. 226-232, 2015.

[7] O. T. Abdelaziz, S. A. Maged, M. I. Awad, "Towards dynamic task/posture control of a 4dof humanoid robotic arm," *International Journal of Mechanical Engineering and Robotics Research*, vol. 9, no. 1, pp. 99-105, 2020.

[8] D. Elayaraja, R. Ramakrishnan, M. Udhayakumar, and S. Ramabalan, *Intelligent Control of Mobile Robot Using C++*, *International Journal of Mechanical Engineering and Robotics Research*, vol. 3, no. 2, pp. 429-434, 2014.

[9] N. D. Dung, *Reverse Dynamics of Parallel Delta Space Robots*, LATS Mechanical and mechanical engineering: 9.52.01.01, Vietnam National Library, code: 629.892 / Đ455L, 2018.

[10] S. Slama, A. Errachdi, and M. Benrejeb, *Neural Adaptive PID and Neural Indirect Adaptive Control Switch Controller for Nonlinear MIMO Systems*, Mathematical Problems in Engineering, vol. 2019, 2019.

[11] Zhang, M. X. Wang, M. Liu, *Adaptive PID Control Based on RBF Neural Network Identification*, in *Proc. 17th IEEE Inter. Conf. on Tools with Artificial Intell, ICTAI'05*, 2005, pp. 681

[12] C. H. Lee and C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Transaction on Fuzzy Systems*, vol. 8, no.4, pp. 349-366, 2000.

[13] S. Wei, Z. Lujin, Z. Jinhai, and M. Siyi, *Adaptive Control Based On Neural Network. Adaptive Control*, Kwanho You (Ed.), InTech, 2009.

[14] J. Fabian, C. Monterrey, "Students member and ruth canahuire, member," *Trajectory Tracking Control of a 3 DOF Delta Robot: a PD and LQR Comparison*, Department of Electrical Engineering, Universidad de Ingenieriay Tecnologia UTEC, Lima Peru, 2016 IEEE XXIII International Congress on Electronics, Electrical Engineering and Computing, 2-5 Aug 2016.

[15] M. Guang, Z. Xing-gui Wang, Man-qiang Liu, *Adaptive PID Control Based on RBF Neural Network Identification*, in *Proc. the 17th IEEE International Conference on Tools with Artificial Intelligence*, 2015.

[16] R. Jafari and R. Dhaouadi, *Adaptive PID Control of a Nonlinear Servomechanism Using Recurrent Neural Networks*, Advances in Reinforcement Learning, 2014.

[17] R. Jafari and R. Dhaouadi, *Adaptive PID Control of a Nonlinear Servomechanism Using Recurrent Neural Networks*, Advances in Reinforcement Learning, Prof. Abdelhamid Mellouk, InTech, 2011.

[18] A. Errachdi and M. Benrejeb, "Performance Comparison of Neural Network Training Approaches in Indirect Adaptive Control," *International Journal of Control, Automation and Systems*, vol. 16, no. 3, pp. 1448–1458, 2018.

[19] X. Y. Zhou, C. Yang, and T. Cai, "A model reference adaptive control/PID compound scheme on disturbance rejection for an aerial inertially stabilized platform," *Journal of Sensors*, vol. 2016, 2016.

Le Minh Thanh received a Bachelor of Engineering degree in Electrical and Electronic Engineering at Cuu Long University in 2006, a Master's degree in Automation at the University of Transport in Ho Chi Minh City in 2011. He is now a lecturer of the Faculty Electrical – Electronics, Vinh Long University of Technical Education.

Luong Hoai Thuong received a Bachelor of Engineering in Control Engineering from Can Tho University in 2009, a Master's degree in Electronic Engineering at Ho Chi Minh City University of Technical Education in 2015. He is currently a lecturer in the Faculty of Electrical - Electronics, Vinh Long University of Technical Education.

Phan Thanh Loc received an engineering degree in Control and Automation Engineering 2016, and is currently pursuing a master's degree in control and automation engineering program 2017 up to now at Can Tho University.

Chi-Ngon Nguyen received his B.S. and M.S. degree in Electrical Engineering from Can Tho University and Ho Chi Minh City University of Technology, in 1996 and 2001, respectively. The degree of Ph.D. was award by the University of Rostock, Germany, in 2007.

Since 1996, he has worked at the Can Tho University. Currently, he is an associate professor in automation of the Department of Automation Technology. He is working as a position of Dean of the College of Engineering Technology at the Can Tho University. His research interests are intelligent control, medical control, pattern recognition, classifications, speech recognition and computer vision.