ERURJ **2025**, *4*, 2, 2534-2559

(Review)

A Review of Artificial Neural Network-Based Controllers for Mobile Robot Dynamics Control

Youssef F. Hanna[1*], Ahmad M. El-Nagar[2], Mohammad El-Bardini[2], A. Aziz Khater[2]

[1]*Department of Mechatronics and Robotics, Faculty of Engineering, Egyptian Russian University, Badr City, Egypt.*

[2]*Department of Industrial Electronics and Control Engineering, Faculty of Electronic Engineering, Menoufia University, Menof, 32852, Egypt.*

*Corresponding author: Youssef F. Hanna, E-mail: Youssef-Fawzy@eru.edu.eg, Tel:+201096328224

Received: 2nd November 2024, Revised 7th February 2025, Accepted 25th February 2025
DOI: 10.21608/erurj.2025.333049.1199

ABSTRACT

Artificial neural networks (ANNs) are adaptive systems employed in many applications such as solving complex nonlinear mathematical functions, identification, data classification, control, and others. The utilization of ANNs in the field of control systems has significantly enhanced the performance of controllers, surpassing that of traditional controllers such as proportional-integral-derivative (PID) controllers. This advancement has transformed these controllers into adaptive control. These adaptive controllers are utilized to control the mobile robot and handle its nonlinearity property and friction uncertainty in its mobility area. Therefore, this paper aims to introduce the comparison of various types of ANNs-based controllers used to practically control the dynamics of the mobile robot. Moreover, multiple experimental tasks are applied to the mobile robot to investigate the performance of each adaptive controller through two performance indices namely; integral absolute error (IAE) and mean absolute error (MAE). The experimental tasks included the set-point change, parameter uncertainty, and measurement error disturbance.

Keywords: Artificial neural networks; Quantum processing; Learning algorithm; Lyapunov stability criterion; Mobile robot.

1. Introduction

ANNs are computing systems that are constructed based on biological neural networks and human brain emulation [1]. The infrastructure of ANNs is the artificial neuron, which is a processing node. In other words, it is a microcomputer. Each artificial neuron simulates the behavior of the biological neuron cell, so it can be considered a digitized model of a biological neuron [2]. Several artificial neurons can be interconnected in the neural structure of ANNs to emulate the human brain processing way. The order that gives ANNs the advantage of power processing. Each neural network can be distinguished by three major issues namely; the activation function, the learning rule of the weights, and the way of connection of the neurons to each other [3]. The capability of ANNs to learn, adapt, and control has rendered them a primary choice for a myriad of applications such as signal classification, function approximation, nonlinear system identification, and control of nonlinear systems [4 and 5]. ANNs are broadly categorized into two main types [6]: conventional neural networks, such as radial basis function neural networks (RBFNNs), and advanced neural networks, which include recurrent neural networks [7], polynomial recurrent neural networks (PRNNs) [8], deep learning neural networks (DNNs) [9], convolutional neural networks (CNNs) [10], long short-term memory neural networks (LSTMNNs) [11], and quantum neural networks (QNNs) [12].

From a control engineering perspective, the two-wheel differential mobile robot falls under the category of nonlinear dynamical systems and belongs to the family of nonholonomic wheeled mobile robots (NHWMRs). In addition, mobile robots are widely used in many applications such as autonomous vehicles, military operations, planetary exploration, and factory automation [13 and 14]. Furthermore, their application in logistics services such as library services [15], and hospital logistics [16, and 17]. These systems are crucial in industrial processes requiring robust controllers capable of addressing the challenges posed by nonlinearity, parameter uncertainty, and load disturbances [18-20]. Historically, conventional controllers such as PID controllers were employed to manage these systems; however, their fixed and non-adaptive parameters rendered them insufficient in the face of plant uncertainties and disturbances [21].

Several previously published researches proposed ANN-based controllers for controlling the mobile robot, for example, in [14], a radial basis function neural network (RBFNN)-based estimator was implemented to estimate lumped uncertainties in a small-wheeled mobile robot, using feedback linearization control for trajectory management. In [22], the model predictive

control (MPC) method was integrated with a primal-dual neural network (PDNN) to control the trajectory of a two-wheel-driven mobile service robot. In [23], a neural network-based kinematic controller (NNKC) was combined with a model reference adaptive controller (MRAC) to control the trajectory of a two-wheel differential mobile robot. The NNKC used one input layer with 6 neurons, two hidden layers with 25 neurons, and one output layer with 2 neurons, i.e., (6-25-25-2) neural network (NN) structure, with a sigmoid activation function. In [24], A neural network controller was designed to identify the output of an improved conventional PID controller for trajectory tracking of a two-wheel differential mobile robot. This controller utilized a (3-4-4-2) neural structure with a tan-sigmoid activation function, employing the backpropagation method for weight learning.

In this paper, a comparison of six adaptive and learnable controllers that are previously published among of them two previously published controllers for the authors. The reason behind that is that the two previously published controllers for the authors used advanced neural networks and stable learning algorithms. The six adaptive controllers are applied practically to the mobile robot and multiple experimental tasks are performed in this work. The six adaptive controllers are explained as follows:

In [25], a PID controller with neural network error identification (PIDC-NNEI) was introduced using a (15-15-1) neural structure and hyperbolic tangent activation function, employing gradient descent with a momentum term for parameters update. Also, in [26], an RBFNN-PID controller with a (3-5-3) neural structure was presented, with all parameters learned using gradient descent with a momentum term. In addition, in [27], a neural network PID controller based on LM (NNPID-LM) with plant identifier was introduced, with a (2-5-1) NN structure using a log-sigmoid activation function, and the gradient descent method is employed for network parameters update. In [28], a smart optimized PID controller based on a neural network (SOPIDNN) was proposed to control the dynamics of a two-wheel differential mobile robot. This method employed a (4-18-3) neural structure with a tan-sigmoid activation function, updating adjustable weights using the gradient descent algorithm. In addition to the two proposed controllers for the authors that were published in the last two years and can be explained as follows:

In [29], a polynomial weighted output recurrent NN-based adaptive PID controller (APIDC-PWORNN) with a 3-2-3 NN structure using a stable learning algorithm based on Lyapunov

stability criterion and adaptive learning rate. The APIDC-PWORNN was used to control two nonlinear dynamical systems.

In [30], an adaptive PID controller based on quantum neural network (APIDC-QNN) with a 3-6-3 NN structure and stable learning algorithm. In addition to Wiener type identifier is proposed to identify the wheel's velocity and generate the sensitivity function. The APIDC-QNN is proposed to control the NHWMR and nonlinear numerical system.

Hence, the novelty and contributions of this paper can be concluded as follows:

1. Reviewing the recent adaptive controllers for mobile robot
2. Comparing various learning algorithms
3. The comparisons in this paper included advanced neural networks like a quantum neural network and a polynomial neural network for controlling the mobile robot.
4. Implementing the six adaptive controllers to the Arduino Mega 2560 microcontroller to control the mobile robot
5. Performing multiple experimental tasks on the mobile robot to distinguish the performance of each controller

The rest of this article is structured as follows: Section 2 introduces the NN structure and the learning rule used with each controller. Section 3 presents the position-speed control system scheme. Section 4 presents the experimental setup of the mobile robot. Followed that the experimental results and comparisons are introduced in section 5. Finally, the conclusions are presented in section 6.

2. Neural network structure and learning rule comparisons

This section introduces the NN structure and the learning rule of each controller from the six controllers mentioned above. In addition, a comprehensive comparison is presented in Table 1.

2.1 PIDC-NNEI [25]

This controller used a 5-5-1 NN structure as shown in Figure 1.

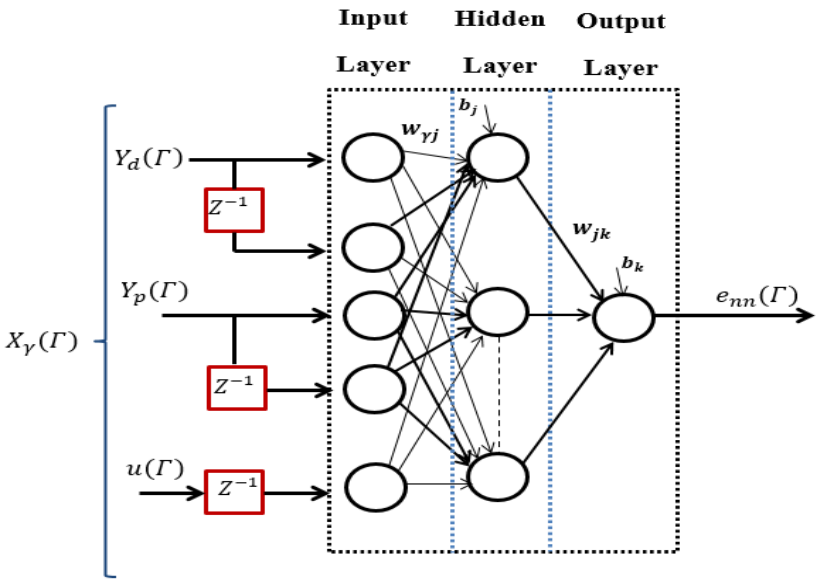


Figure 1. PIDC-NNEI NN structure [25].

As indicated in Figure 1, five input neurons are employed for the input layer, five hidden neurons are employed for the hidden layer, and one output neuron is utilized for the output layer. Where, Γ is the sample number, $X_\gamma(\Gamma) = [Y_d(\Gamma)\ \ Y_d(\Gamma-1)\ \ Y_p(\Gamma)\ \ Y_p(\Gamma-1)\ u(\Gamma-1)]$ is the online input data vector that is entered into the neural network at every sample, $Y_d(\Gamma)$ is the desired output, $Y_p(\Gamma)$ is the actual plant output measured at every sample, $u(\Gamma-1)$ is the past value of the control signal, $w_{\gamma j}$ is the connecting weight between the γ^{th} input neuron and j^{th} hidden neuron, w_{jk} is the connecting weight between the j^{th} hidden neuron and k^{th} output neuron, b_j is the bias parameter of the j^{th} hidden neuron, b_k is the bias parameter of the k^{th} output neuron. In addition, the utilized activation function for the hidden neurons is a tan-sigmoid function. The output of this network is the identified error signal (e_{nn}) between the desired output and the plant output. This NN structure yields 39 adjustable weights. In this paper, this NN structure is utilized for each wheel controller separately which yields 78 adjustable weights total for the two wheels.

Next, the learning rule will be explained to the proportional gain of the PIDC-NNEI as an example and can be generalized to the integral and derivative gains as follows [25]:

$$\frac{de\ (\Gamma)}{dK_P(\Gamma)} = \frac{de_{nn}(\Gamma)}{du(\Gamma)} \frac{du(\Gamma)}{dK_P(\Gamma)} \tag{1}$$

where, $e\ (\Gamma)$ is the error signal between the desired wheels' velocity ($\dot{\theta}_{Rd}$, $\dot{\theta}_{Ld}$) and the actual wheels' velocity ($\dot{\theta}_{R}$, $\dot{\theta}_{L}$), $e_{nn}(\Gamma)$ is the identified error signal, $K_P(\Gamma)$ is the proportional gain, and $u(\Gamma)$ is the control signal.

$$\Delta K_P(\Gamma) = \rho \left| \frac{de_{nn}(\Gamma)}{du(\Gamma)} \frac{du(\Gamma)}{dK_P(\Gamma)} \right| + \delta \Delta K_P(\Gamma - 1) \tag{2}$$

where, ρ is the fixed learning rate, and δ is the momentum coefficient.

$$K_P(\Gamma) = K_P(\Gamma - 1) - \Delta K_P(\Gamma) \tag{3}$$

2.2 RBFNN-PID [26]

This controller utilized a 3-5-3 NN structure as depicted in Figure 2.

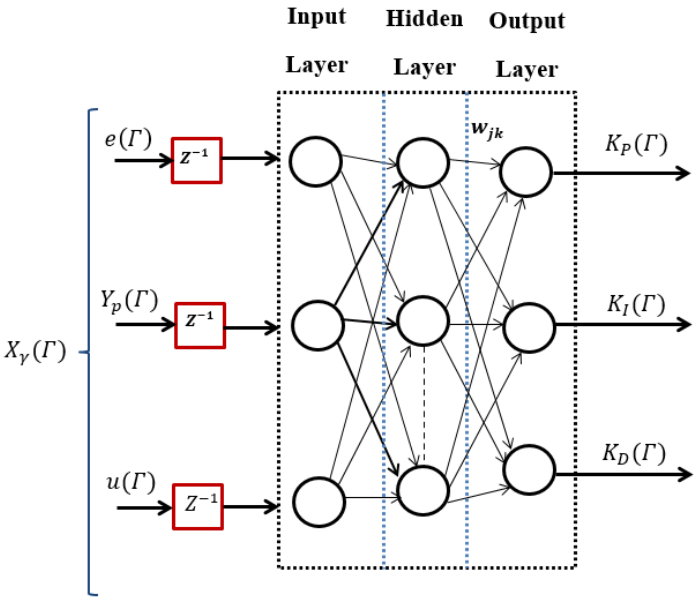


Figure 2. NN Structure of the RBFNN-PID controller [26].

This NN structure uses three input neurons, five hidden neurons, and three output neurons as shown in Figure 2. In addition, two activation functions are employed the 1st one is the RBF activation function with the hidden neurons and the 2nd activation function is the sigmoid function utilized with the output neurons. Where, $X_\gamma(\Gamma) = [e(\Gamma - 1)\ Y_p(\Gamma - 1)\ u(\Gamma - 1)]$ is the online input data vector. The outputs of this network are the adaptive PID gains namely; the proportional gain ($K_P(\Gamma)$), the integral gain ($K_I(\Gamma)$), and the derivative gain ($K_D(\Gamma)$). This NN structure yields 25 adjustable weights. In this work, this NN structure is utilized for each wheel separately which yields 50 tunable parameters total for the two wheels.

Moreover, the learning rule of this controller was described as follows [26]:

$$E = \frac{1}{2}e^2 \tag{4}$$

where, E is the cost function. The network weights; $w(\Gamma)$ can be updated using the following gradient descent formula:

$$\Delta w(\Gamma) = -\rho \frac{dE(\Gamma)}{dw(\Gamma)} \tag{5}$$

$$w(\Gamma) = w(\Gamma - 1) + \Delta w(\Gamma) - \alpha(w(\Gamma - 1) - w(\Gamma - 2)) \tag{6}$$

where, α is the momentum factor.

2.3 NNPID-LM [27]

A 2-5-1 NN structure is used to identify the plant output for each controller as shown in Figure 3.

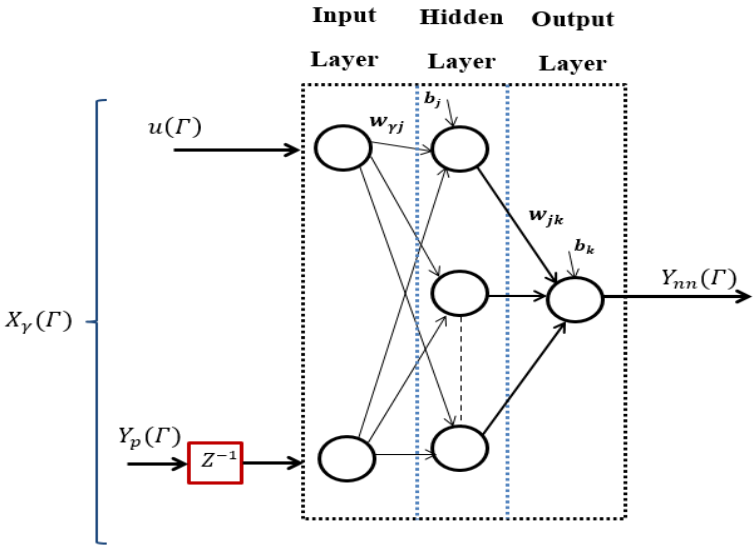


Figure 3. NN structure of the plant identifier [27].

In Figure 3, tow input neurons, five hidden neurons, and one output neuron are employed for the plant identifier. Where, $X_\gamma(\Gamma) = [\, u(\Gamma)\, Y_p(\Gamma - 1)]$ is the utilized training data vector, $Y_p(\Gamma - 1)$ is the past value of the plant output, and $Y_{nn}(\Gamma)$ is the identified plant output, i.e., the network output. A log-sigmoid activation function is used with the hidden neurons. This structure yields 24 tunable parameters for one wheel controller and 48 tunable parameters for the two wheels' controllers.

Next, the LM update rule is used for this controller as follows [27]:

$$X_w(\Gamma + 1) = X_w(\Gamma) - \left[J_{co}\left(X_w(\Gamma)\right)^T J_{co}(X_w(\Gamma)\,) + \lambda_d\, I\right]^{-1} J_{co}\left(X_w(\Gamma)\right) e(\Gamma) \tag{7}$$

where, $X_w(\Gamma)$ is the weights vector of the network, λ_d is the damping factor, and $J_{co}\left(X_w(\Gamma)\right)$ is the Jacobian matrix of all weights.

2.4 SOPIDNN [28]

A 2-18-3 NN structure is used for each wheel controller. The NN structure of this controller is depicted in Figure 4.

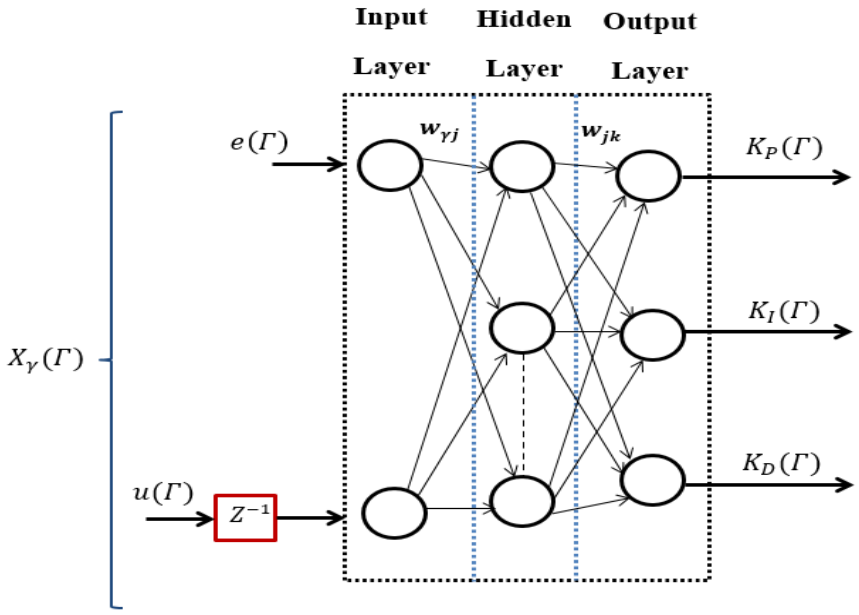


Figure 4. NN Structure of the SOPIDNN controller [28].

In Figure 4, two input neurons, eighteen hidden neurons, and three output neurons are employed for the SOPIDNN controller. In this structure, a tan-sigmoid activation function was utilized in the hidden layer and a pure line function at the output layer. Where, $X_\gamma(\Gamma) = [\, e(\Gamma)\; u(\Gamma - 1)]$ is the online input data vector. In addition, the outputs of this network are the PID adaptive gains. Consequently, this structure results 90 adjustable weights for each wheel controller and 180 adjustable for the robot two wheels. Moreover, this controller used the gradient descent learning method as follows:

$$w(\Gamma + 1) = w(\Gamma) + \rho \frac{dE(\Gamma)}{dw(\Gamma)} \tag{8}$$

where, $E(\Gamma)$ is the cost function defined as in Eq. (4).

2.5 APIDC-PWORNN [29]

In previously published paper, the authors used a 3-2-3 NN structure for each wheel controller. This NN structure is indicated in Figure 5.

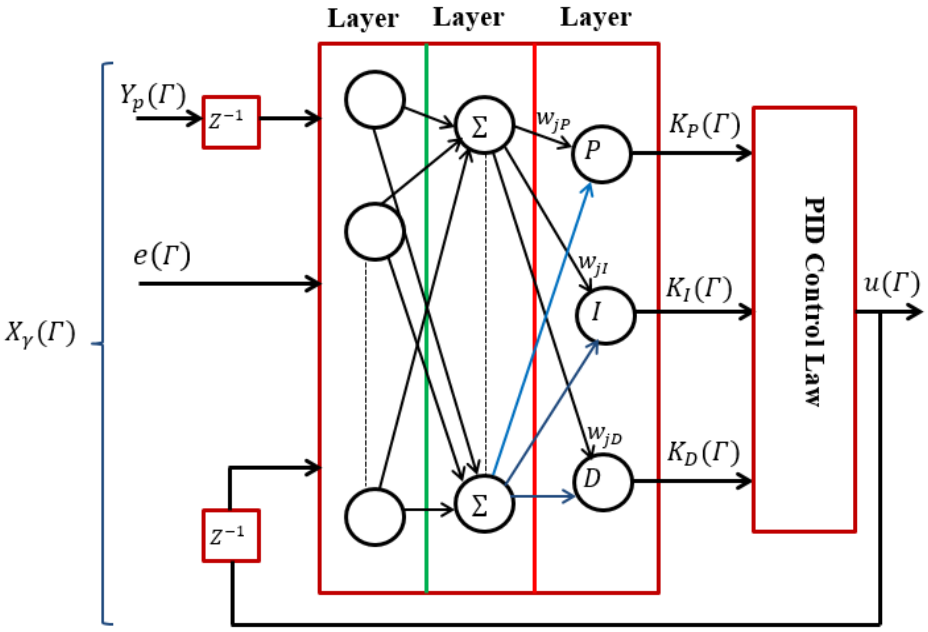


Figure 5. NN Structure of the APIDC-PWORNN [29].

As shown in Figure 5, three input neurons are employed for the input layer, two hidden neurons are used for the hidden layer, and three output neurons are employed for the output layer. Where, $X_\gamma(\Gamma) = [Y_p(\Gamma - 1)\ e(\Gamma)\ u(\Gamma - 1)\ ]$ is the online training input data vector, $w_{jP}(\Gamma)$ is the connecting weigh between the j^{th} hidden neuron and the proportional node (P), $w_{jI}(\Gamma)$ is the connecting weight between the j^{th} hidden neuron and the integral node (I), and , $w_{jD}(\Gamma)$ is the connecting weight between the j^{th} hidden neuron and the derivative node (D). Moreover, the hidden neurons process the unweighted input data by mathematical summation. Following that, the output neurons process the weighted outputs delivered from the hidden neurons by multiplying them. Therefore, the activation function of this network was called a polynomial function that results from the product of sum process. This polynomial weighted output recurrent NN (PWORNN) uses only six adjustable weights for each controller and twelve adjustable weights for the right and left controllers.

Next, this controller used a stable learning rule derived based on the Lyapunov stability criterion as follows [29]:

$$w(\Gamma) = w(\Gamma - 1) + \rho(\Gamma)\, \Delta w(\Gamma) \tag{9}$$

$$\Delta w(\Gamma) = -\frac{1}{c}[L_1 + \frac{\partial e(\Gamma)}{\partial w(\Gamma)} L_2] \tag{10}$$

where, $L_1 = \big(2b\Delta e(\Gamma) + 2be(\Gamma) + 2cw(\Gamma)\big), L_2 = \ (2a\Delta e(\Gamma) + 2ae(\Gamma) + 2bw(\Gamma))$, *a*, *b*, *c* are the coefficients of the Lyapunov function [29], $w(\Gamma)$ is the weights vector, $\Delta e(\Gamma) = e(\Gamma) - e(\Gamma - 1)$ is the difference of the error signal, and $\rho(\Gamma)$ is the adaptive learning rate derived based on the Lyapunov theory as follows [29]:

$$0 \leq \rho(\Gamma) \leq \left\| \frac{ce(\Gamma)}{\left[\left[\frac{\partial Y_P(\Gamma)}{\partial w(\Gamma)}\right]^2 \big(bw(\Gamma) - ae(\Gamma)\big) + \frac{\partial Y_P(\Gamma)}{\partial w(\Gamma)} (be(\Gamma) - cw(\Gamma)\right]} \right\| \tag{11}$$

where, $Y_P(\Gamma)$ is the actual plant output i.e., the actual wheel velocity, and $e(\Gamma)$ is the error signal.

2.6 APIDC-QNN [30]

In previously published work, the authors used a 3-6-3 NN structure with quantum neurons in the input and hidden layers as shown in Figure 6.

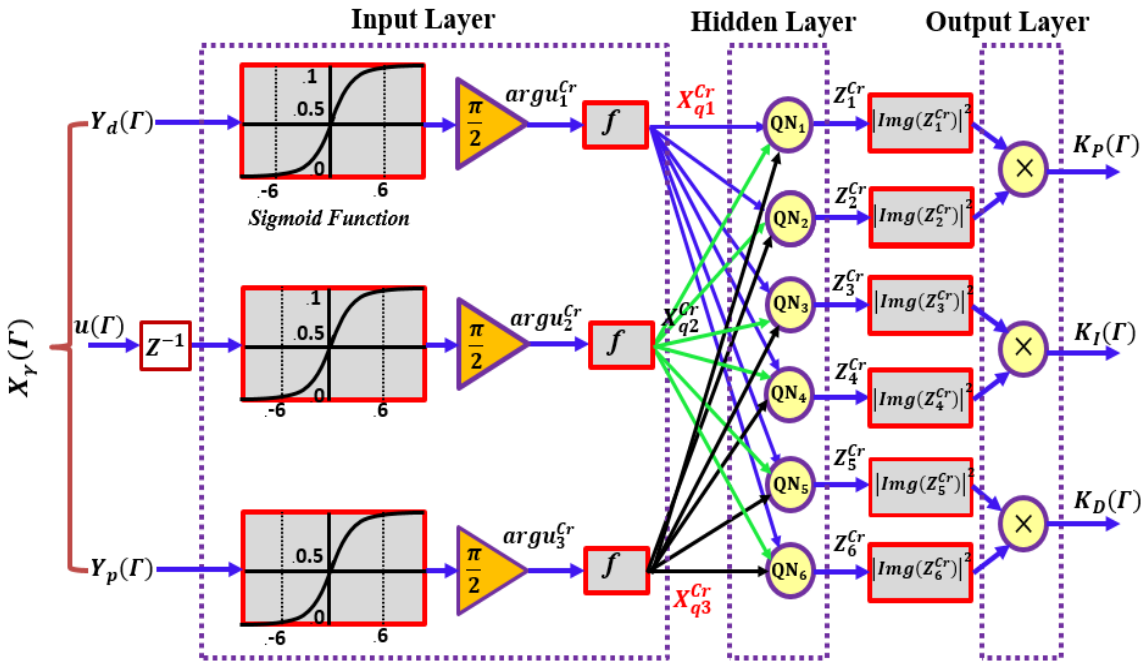


Figure 6. NN Structure of the APIDC-QNN [30].

In Figure 6, three input quantum neurons are employed for the input layer, six hidden quantum neurons namely; QN_1, QN_2, …., QN_6 are used for the hidden layer, and three neurons are employed for the output layer. Where, $X_\gamma(\Gamma) = [Y_d(\Gamma)\ u(\Gamma-1)\ Y_p(\Gamma)]$ is the online training input data vector, $Y_d(\Gamma)$ is the desired plant output. These input data are transformed into quantum states namely; ${X_{q1}}^{Cr}(\Gamma)$, ${X_{q2}}^{Cr}(\Gamma)$, and ${X_{q3}}^{Cr}(\Gamma)$ using quantum transformation function explained in [30]. Then, a quantum processing technique is utilized for each hidden quantum neuron in the hidden layer. The quantum processing in each hidden quantum neuron is performed through five stages namely; the input stage, phase rotation stage, aggregation stage, reverse rotation stage, and output stage. These stages are described in detail in [30]. The output neurons process the delivered quantum states from the hidden layer by mathematical multiplication. Then, these network outputs are considered the PID adaptive gains.

Moreover, the learning rule of this controller is derived using the Lyapunov stability theory as follows [30]:

$$w(\Gamma) = w(\Gamma-1) + \rho\,\Delta w(\Gamma) \tag{12}$$

$$\Delta w(\Gamma) = \frac{-\left[w(\Gamma) + e(\Gamma)\frac{\partial e(\Gamma)}{\partial w(\Gamma)}\right]}{\left[1 + \left\|\frac{\partial e(\Gamma)}{\partial w(\Gamma)}\right\|^2\right]} \tag{13}$$

where, ρ is a fixed learning rate in this work.

Next, a systematic comparison that summarizes the key attributes of each controller, for example, the NN structure, number of tunable parameters, activation function, learning method, and learning rate, are listed in Table 1.

Table 1. Systematic comparison of the utilized six adaptive controllers.

Controller	Year	NN structure	No of parameters	Activation function	Learning method	Learning rate
PID-NNEI [25]	2018	5-5-1	78	hyperbolic tangent	Gradient descent plus, momentum term	Fixed
RBFNN-PID [26]	2020	3-5-3	50	RBF	Gradient descent plus, momentum term	Fixed
NNPID-LM [27]	2020	2-5-1	48	log-sigmoid	LM	Fixed
SOPIDNN [28]	2021	2-18-3	180	tan sigmoid	Gradient descent	Fixed
APIDC-PWORNN [29]	2023	3-2-3	12	Polynomial function	Based on Lyapunov	Adaptive
APIDC-QNN [30]	2023	3-6-3	86	Quantum processing	Based on Lyapunov	Fixed

In Table 1, the NN structure of each controller is calculated only for one wheel, while the number of tunable parameters is calculated for the two wheels. Obviously from Table 1, the APIDC-PWORNN used the minimum number of tunable parameters (only 12), while the SOPIDNN used the largest number of tunable parameters (180). Furthermore, the controllers that are learned using the gradient descent method such as PID-NNEI, RBFNN-PID, and SOPIDNN didn't guarantee learning stability and could fall in the local minima. On the contrary, the

controllers learned based on the Lyapunov stability criterion guarantee learning stability such as APIDC-PWORNN and APIDC-QNN.

3. Position-Speed Control Scheme

In this paper, each controller from the six controllers is employed as the dynamics controller for the mobile robot, to control the right and left wheel velocity. In addition, the position-speed control block diagram of the mobile robot is represented as shown in Figure 7.

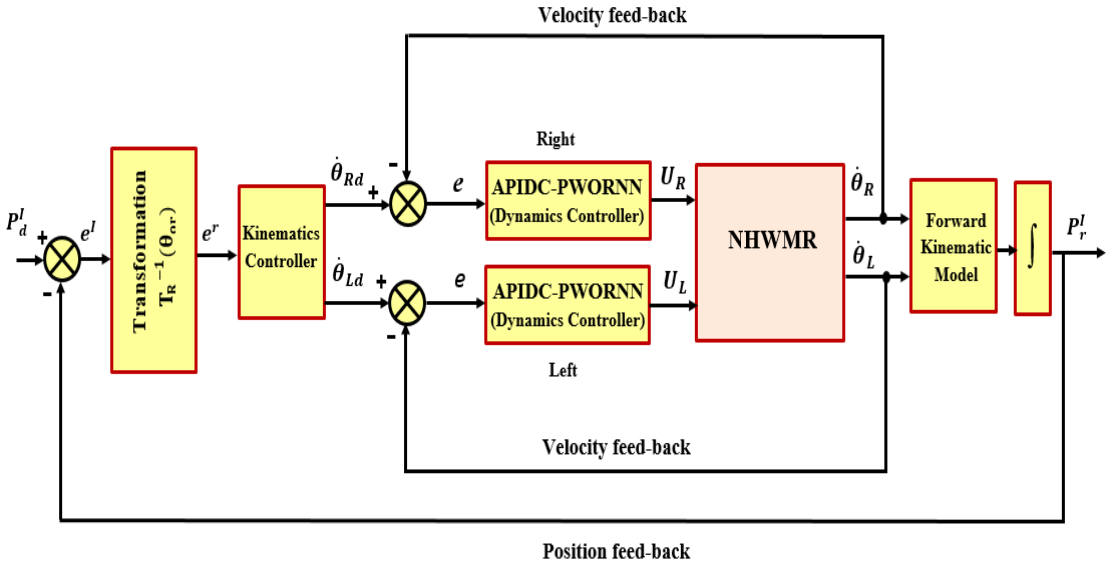


Figure 7. Position-Speed control block diagram.

In Figure 7, $\boldsymbol{P_d^I} = [X_d\ Y_d\ \theta_d\ ]$ is the desired positions vector of the desired trajectory in the inertial frame, $\boldsymbol{P_r^I} = [X_I\ Y_I\ \theta_{or}]$ is the actual positions vector of the controlled mobile robot in the inertial frame. Also, Figure 7 shows two main controllers. The first controller is the kinematics controller utilized in [31-33] and the second controller is the dynamics controller where the APIDC-PWORNN in Figure 7 is an example that can be replaced by each controller from the six controllers used for comparisons.

.

4. Experimental Setup

In this section, the experimental setup of the NHWMR is presented. The mobile robot is powered by two differential wheels (i.e., left wheel and right wheel) through two actuators (i.e., two 9 v DC motors with gearbox). So, it is considered a multi-input-multi-output (MIMO) system as shown in Figure 8. The caster wheel, which is shown in Figure 8, is an unpowered wheel is utilized to achieve the mobile robot balance. Moreover, a low-speed microcontroller (Arduino mega 2560) is utilized with a16 MHz clock speed and a small flash memory of 2560 KB to execute all experimental tasks. In addition, an L298N dual channel H-bridge driver for driving the two DC motors, two optical sensors for speed measuring are employed in this work, an HMC5883 electronic compass is used to measure the robot heading angle, and a micro-SD module is used to record all sensors data. The physical parameters of the mobile robot used are listed in Table 2.

Table 2. Physical parameters of the mobile robot.

Symbol	Description	Parameter Value
m_t	The total mass of the mobile robot	0.700 kg
m_w	Actuator with driving wheel mass	0.060 kg
r	Driving wheel radius	0.031 m
L	Mid-distance between the two driving wheels	0.0565 m
N_g	Gear ratio	48
R_a	Electrical resistance of armature coil	5.21 Ω
L_a	Electrical inductance of armature coil	1.804×10^{-3} H

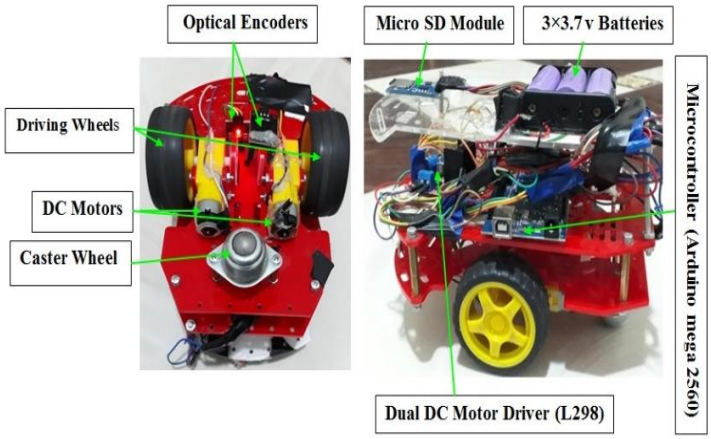


Figure 8. Mobile robot.

5. Experimental results and discussion

In this section, three practical tasks are applied to the mobile robot such as the desired trajectory change, mass uncertainty, and measurement error disturbance. All data are collected at every sample using optical sensors for measuring wheels' velocity, a digital compass for measuring the robot position, and a micro-SD card module is employed to record all sensor readings. The experimental results are listed in two tables considering the integral absolute error (IAE) and the mean absolute error (MAE) for the robot position error.

5.1 Task 1: Desired trajectory Change

In this task, the desired trajectory starts with a 2 m diameter circle and then changes into a 3 m diameter circle to examine the controllers' performance under the set-point change. Based on the desired trajectory position, the kinematics controller determines the desired wheels' velocities ($\dot{\theta}_{Rd}$, $\dot{\theta}_{Ld}$) automatically. Then the speed sensors measure the actual wheels' velocities ($\dot{\theta}_{R}$, $\dot{\theta}_{L}$) at every sample and feedback to the dynamic's controller. Also, the electronic compass measures the actual robot position and feedback to the kinematics controller. The desired and actual trajectories are shown in Figure 9, and the control signal is depicted in Figure 10. From Figure 9, each controller's performance can be observed by minimizing the trajectory deviation.

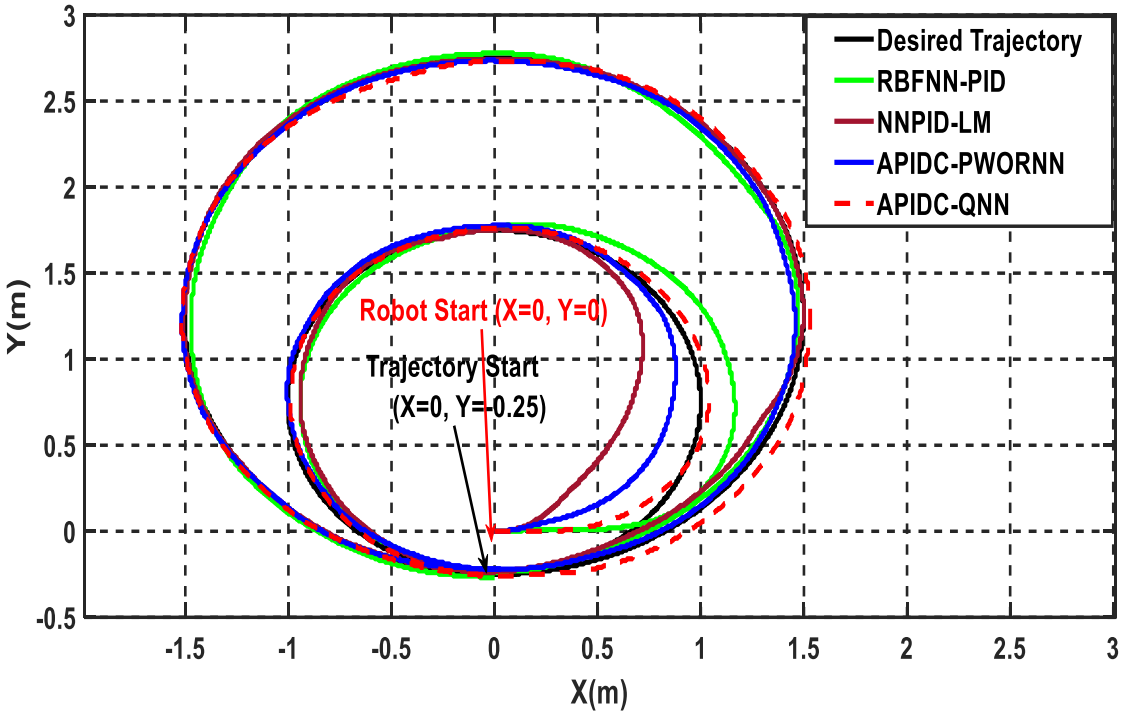
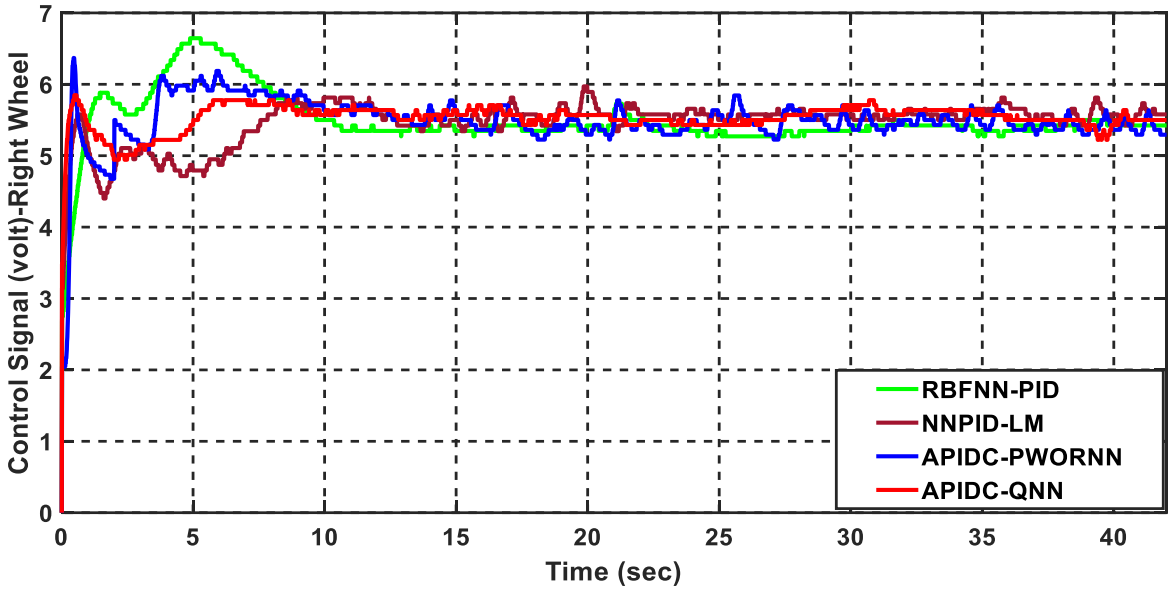
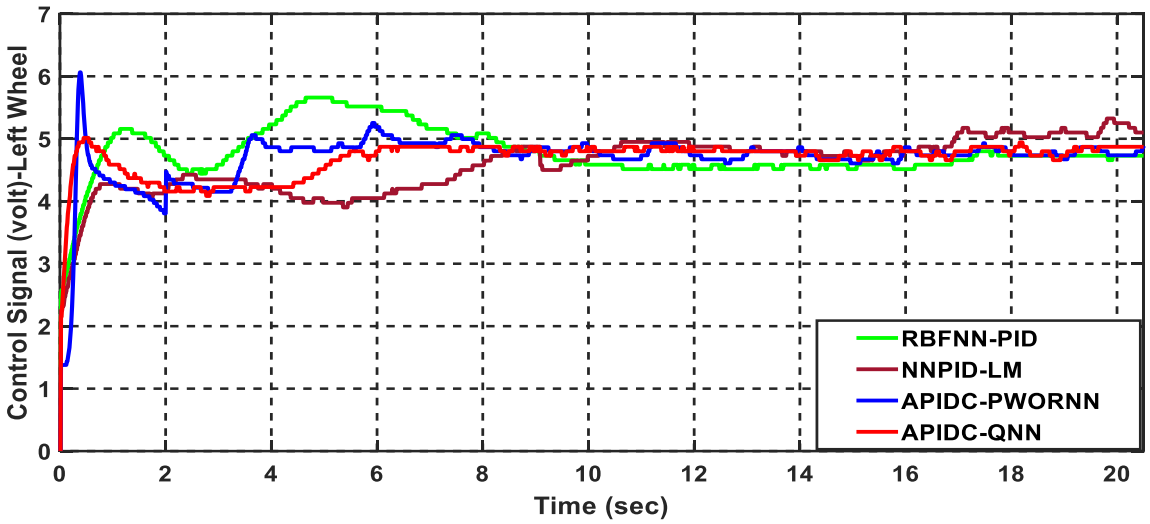


Figure 9. Circle trajectory in the X-Y plane (Task 1).



(a)



(b)

Figure 10. Control signal (Task 1) a) Right wheel b) Left wheel.

5.2 Task 2: Mass Uncertainty

This task examines the processing uncertainty for each controller by adding an additional mass to the mobile robot during the motion suddenly at the half of the circle trajectory at $t = 8.5\ sec$. Figure 11 shows the circle trajectory in the X-Y plane and Figure 12 shows the control signal (motor voltage). Still the two previously proposed controllers APIDC-PWORNN, and APIDC-QNN are the superior controllers to other controllers for handling the mass uncertainty.

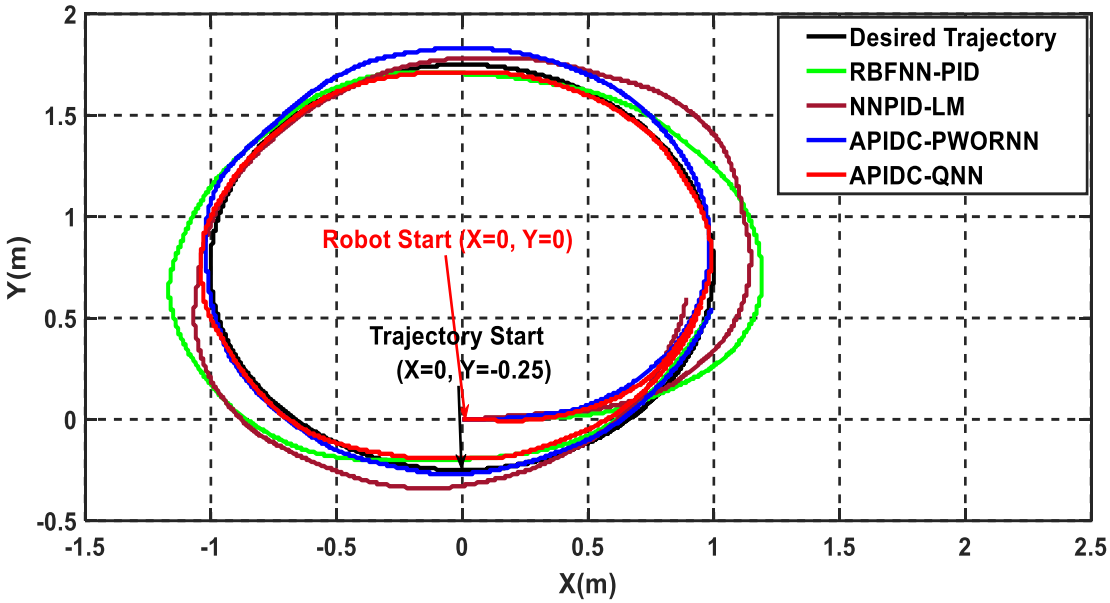


Figure 11. Circle trajectory in the X-Y plane (Task 2).

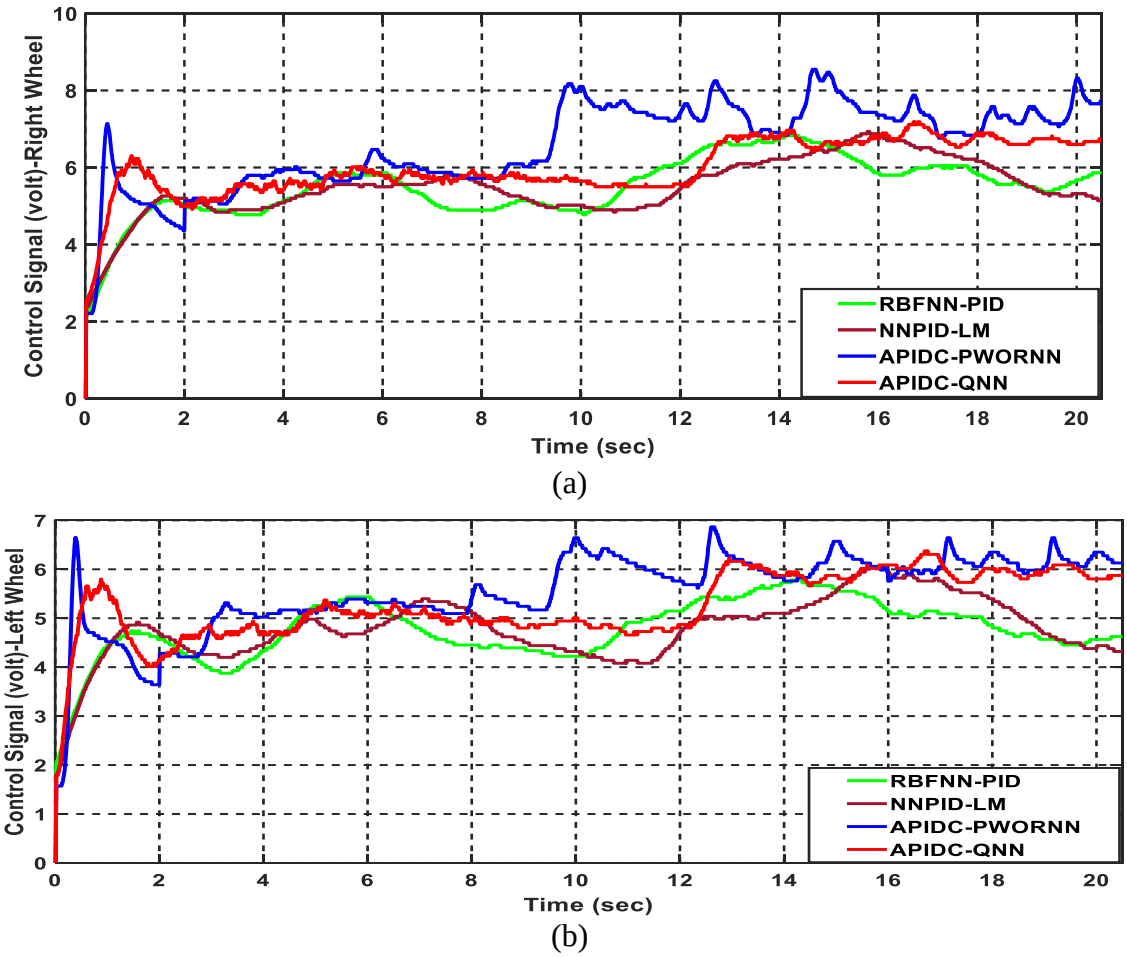


Figure 12. Control signal (Task 2) a) Right wheel b) Left wheel.

5.3 Task 3: Measurement Error Disturbance

Sensor measurement error can be expressed by adding -15% of the measured velocity to the measured velocity of each wheel to examine the ability of each controller to handle this disturbance. The circular trajectories under this disturbance are depicted in Figure 13, and the control signal is shown in Figure 14. The two previously proposed controllers (APIDC-PWORNN, and APIDC-QNN) show the best tracking and have the minimum deviation.

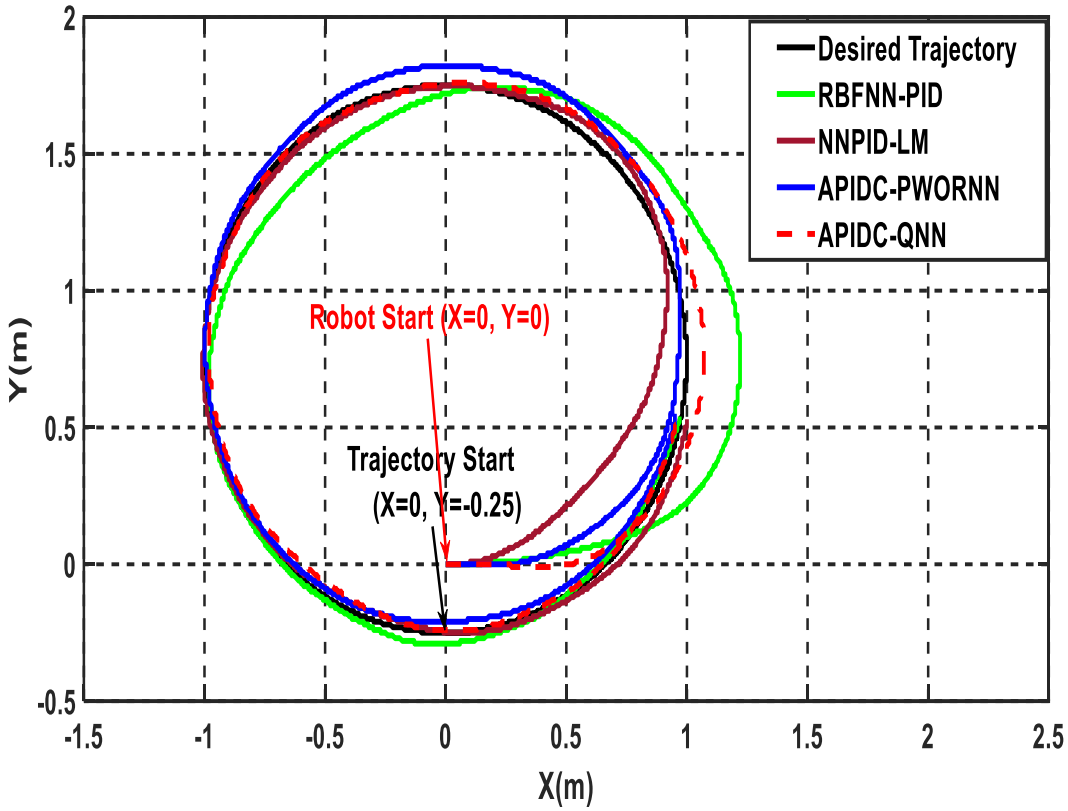
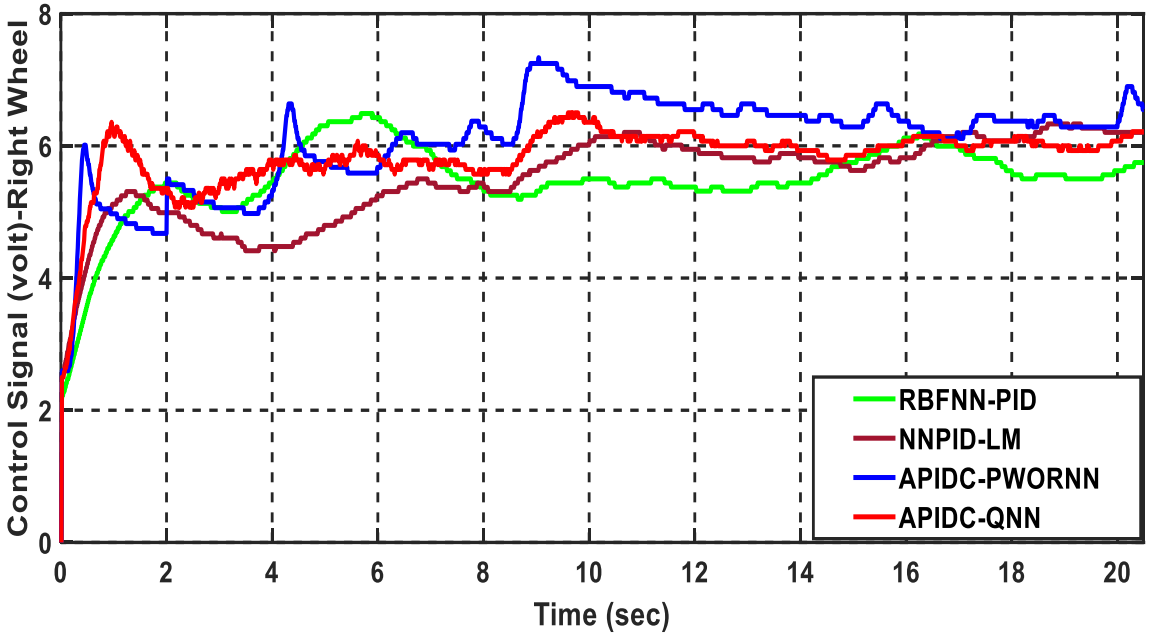
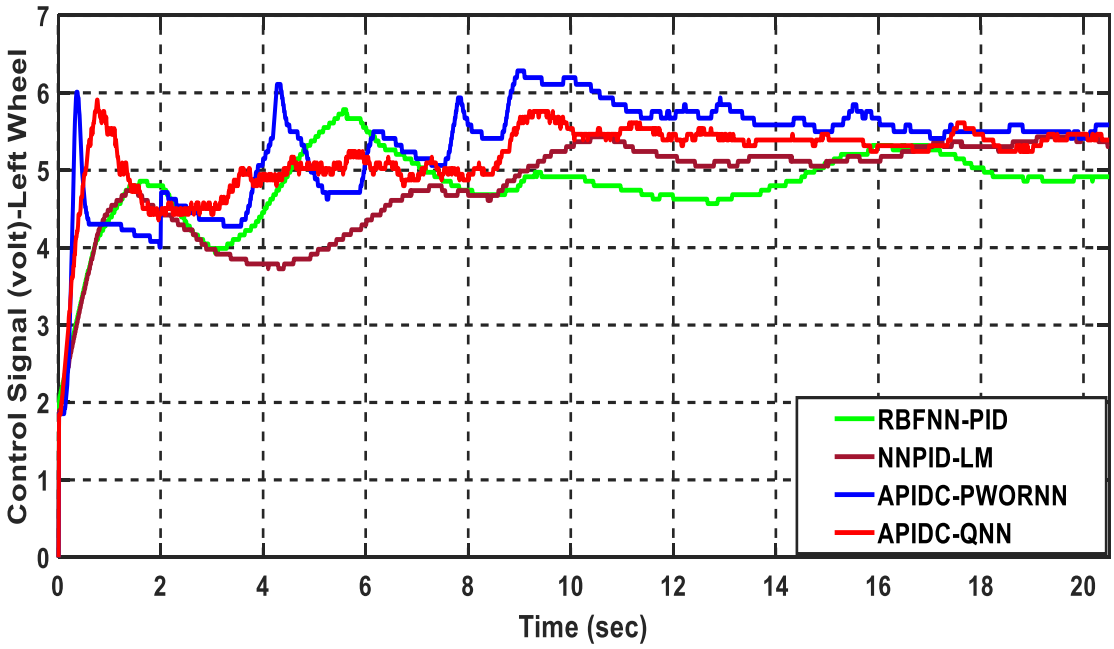


Figure 13. Circle trajectory in X-Y plane (Task 3).



(a)



(b)

Figure 14. Control signal (Task 3) a) Right wheel b) Left wheel.

Moreover, the values of performance indices (IAE and MAE) of the position error are calculated as in [23] for all experimental tasks of the six controllers. And then listed in Tables (3, and 4).

Table 3. IAE and MAE Values (Task 1, and Task 2).

Algorithms	NN structure	Task 1		Task 2	
		IAE	MAE	IAE	MAE
PIDC-NNEI [25]	5-5-1	3.0959	0.0734	1.9811	0.0966
RBFNN-PID [26]	3-5-3	2.9761	0.0706	2.6431	0.1289
NNPID-LM [27]	2-5-1	3.6726	0.0871	2.7187	0.1326
SOPIDNN [28]	2-18-3	3.4215	0.0812	1.8789	0.0917
APIDC-PWORNN [29]	**3-2-3**	**2.6365**	**0.0626**	**1.5210**	**0.0742**
APIDC-QNN [30]	**3-6-3**	**2.0459**	**0.0485**	**1.1957**	**0.0583**

Table 4. IAE and MAE Values (Task 3).

Algorithms	NN structure	Task 3	
		IAE	MAE
PIDC-NNEI [25]	5-5-1	2.2617	0.1103
RBFNN-PID [26]	3-5-3	2.5154	0.1227
NNPID-LM [27]	2-5-1	1.8699	0.0912
SOPIDNN [28]	2-18-3	2.0927	0.1021
APIDC-PWORNN [29]	**3-2-3**	**1.5474**	**0.0755**
APIDC-QNN [30]	**3-6-3**	**1.1825**	**0.0577**

From the obtained experimental results, the APIDC-QNN recorded the minimum values of IAE and MAE for all tasks as indicated in Tables (3, and 4) which indicates the superiority and the powerful processing of this controller over other controllers. The reason behind that is the advanced neural network used with this controller (quantum neural network) and the stable learning rule derived based on the Lyapunov stability criterion. Moreover, the computation time of every controller is computed and the average of one thousand readings is considered and listed in Table 5.

Table 5. Computation Time.

Controller	NN Structure	No of parameters	Computation time (ms)
PIDC-NNEI [25]	5-5-1	78	97.5840
RBFNN-PID [26]	3-5-3	50	98.5970
NNPID-LM [27]	2-5-1	48	99.2510
SOPIDNN [28]	2-18-3	180	102.2560
APIDC-PWORNN [29]	**3-2-3**	**12**	**89.3290**
APIDC-QNN [30]	**3-6-3**	**86**	**136.5840**

From Table 5, the APIDC-PWORNN controller has the minimum computation time (89.3290 ms). Although the APIDC-QNN controller based on the quantum NN has no complex neural structure and no large number of tunable parameters compared with other controllers, it scored a large computation time. This is because mathematical trigonometric functions used with quantum processing take a long time to process.

6.Conclusions

This article introduced a review of the ANN-based adaptive controllers for controlling the two-wheel differential mobile robot. The comparisons included six ANN-based adaptive controllers two of them are previously published for the authors. All controllers are implemented practically on the microcontroller Arduino Mega 2560 to control the mobile robot. Three experimental tasks are applied to examine the controller performance and two performance indices (IAE, and MAE) are considered. The experimental results show that the APIDC-QNN is superior to other controllers in that it records the minimum values of IAE and MAE as a result of using Quantum as an activation function and using a learning rule for updating weights based on Lyapunov theory. In future work, these controllers can be developed to control an autonomous vehicle and mobile robot manipulators.

• **Acknowledgment**

The authors are thankful for the support from the Egyptian Russian University to execute the research work.

• **Conflict of Interest**

The authors declare that there is no conflict of interest related to the article.

7.References

[1] Abdulridha, Hayder Mahdi, and Zainab Abdullah Hassoun. "Control design of robotic manipulator based on quantum neural network." Journal of Dynamic Systems, Measurement, and Control 140, no. 6 (2018): 061002.

[2] Agatonovic-Kustrin S, Beresford R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. Journal of pharmaceutical and biomedical analysis. 2000 Jun 1;22(5):717-27.

[3] Asghar R, Fulginei FR, Quercio M, Mahrouch A. Artificial neural networks for photovoltaic power forecasting: a review of five promising models. IEEE Access. 2024 Jun 28.

[4] Linsker R. Neural network learning of optimal Kalman prediction and control. Neural Networks. 2008 Nov 1;21(9):1328-43.

[5] Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H. State-of-the-art in artificial neural network applications: A survey. Heliyon. 2018 Nov 1;4(11).

[6] Kommadi B. Quantum Computing Solutions. Springer Books. 2020.

[7] Chang WD. Recurrent neural network modeling combined with bilinear model structure. Neural Computing and Applications. 2014 Mar;24:765-73.

[8] Khan D, Alshahrani A, Almjally A, Al Mudawi N, Algarni A, Al Nowaiser K, Jalal A. Advanced IoT-based human activity recognition and localization using Deep Polynomial neural network. Ieee Access. 2024 Jun 28.

[9] Abd Elaziz M, Dahou A, Abualigah L, Yu L, Alshinwan M, Khasawneh AM, Lu S. Advanced metaheuristic optimization techniques in applications of deep neural networks: a review. Neural Computing and Applications. 2021 Nov 1:1-21.

[10] Wiatowski T, Bölcskei H. A mathematical theory of deep convolutional neural networks for feature extraction. IEEE Transactions on Information Theory. 2017 Nov 21;64(3):1845-66.

[11] Skrobek D, Krzywanski J, Sosnowski M, Kulakowska A, Zylka A, Grabowska K, Ciesielska K, Nowak W. Prediction of sorption processes using the deep learning methods (long short-term memory). Energies. 2020 Dec 14;13(24):6601.

[12] Takahashi K, Kurokawa M, Hashimoto M. Controller application of a multi-layer quantum neural network with qubit neurons. Journal of Advanced Mechanical Design, Systems, and

Manufacturing. 2012;6(4):526-40.

[13] Khnissi K, Seddik C, Seddik H. Smart navigation of mobile robot using neural network controller. In2018 International Conference on Smart Communications in Network Technologies (SaCoNeT) 2018 Oct 27 (pp. 205-210). IEEE.

[14] Arab A, Yi J, Fateh MM, Arabshahi S. Robust control of a low-cost mobile robot using a neural network uncertainty compensator. InDynamic Systems and Control Conference 2014 Oct 22 (Vol. 46186, p. V001T17A005). American Society of Mechanical Engineers.

[15] Mehta R, Sahu A. Autonomous robot for inventory management in libraries. In2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) 2020 Feb 22 (pp. 1-6). IEEE.

[16] Tagliavini L, Baglieri L, Colucci G, Botta A, Visconte C, Quaglia G. DOT PAQUITOP, an Autonomous Mobile Manipulator for Hospital Assistance. Electronics. 2023 Jan 4;12(2):268.

[17] He W, Cao Z, Ye H. Path Planning Algorithms for Mobile Robots in Hospital Environment during Covid-19. InProceedings of the 3rd International Symposium on Artificial Intelligence for Medicine Sciences 2022 Oct 13 (pp. 522-530).

[18] Kumar R, Srivastava S. Externally Recurrent Neural Network based identification of dynamic systems using Lyapunov stability analysis. ISA transactions. 2020 Mar 1;98:292-308.

[19] Khater AA, El-Nagar AM, El-Bardini M, El-Rabaie NM. Improving the performance of a class of adaptive fuzzy controller based on stable and fast on-line learning algorithm. European Journal of Control. 2020 Jan 1;51:39-52.

[20] Zheng K, Hu Y, Wu B. Intelligent fuzzy sliding mode control for complex robot system with disturbances. European Journal of Control. 2020 Jan 1;51:95-109.

[21] Somefun OA, Akingbade K, Dahunsi F. The dilemma of PID tuning. Annual Reviews in Control. 2021 Jan 1;52:65-74.

[22] Li Z, Deng J, Lu R, Xu Y, Bai J, Su CY. Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2015 Aug 31;46(6):740-9.

[23] Hassan N, Saleem A. Neural network-based adaptive controller for trajectory tracking of wheeled mobile robots. IEEE Access. 2022 Jan 27;10:13582-97.

[24] Ly TT, Thai NH, Phong LT. Design of neural network-PID controller for trajectory tracking of differential drive mobile robot. Vietnam Journal of Science and Technology. 2024 Mar 1;62(2):374-86.

[25] Cho CN, Song YH, Lee CH, Kim HJ. Neural network-based real time PID gain update algorithm for contour error reduction. International Journal of Precision Engineering and Manufacturing. 2018 Nov;19:1619-25.

[26] Pu Q, Zhu X, Zhang R, Liu J, Cai D, Fu G. Speed profile tracking by an adaptive controller for subway train based on neural network and PID algorithm. IEEE Transactions on Vehicular Technology. 2020 Aug 26;69(10):10656-67.

[27] Hao J, Zhang G, Liu W, Zheng Y, Ren L. Data-driven tracking control based on LM and PID neural network with relay feedback for discrete nonlinear systems. IEEE Transactions on Industrial Electronics. 2020 Nov 2;68(11):11587-97.

[28] Ben Jabeur C, Seddik H. Design of a PID optimized neural networks and PD fuzzy logic controllers for a two-wheeled mobile robot. Asian Journal of Control. 2021 Jan;23(1):23-41.

[29] Hanna YF, Khater AA, El-Nagar AM, El-Bardini M. Polynomial recurrent neural network-based adaptive pid controller with stable learning algorithm. Neural Processing Letters. 2023 Jun;55(3):2885-910.

[30] Hanna YF, Khater AA, El-Bardini M, El-Nagar AM. Real time adaptive PID controller based on quantum neural network for nonlinear systems. Engineering Applications of Artificial Intelligence. 2023 Nov 1; 126:106952.

[31] Tzafestas SG. Introduction to mobile robot control. Elsevier; 2013 Oct 3.

[32] Martins FN, Celeste WC, Carelli R, Sarcinelli-Filho M, Bastos-Filho TF. An adaptive dynamic controller for autonomous mobile robot trajectory tracking. Control engineering practice. 2008 Nov 1;16(11):1354-63.

[33] Martins FN, Sarcinelli-Filho M, Carelli R. A velocity-based dynamic model and its properties for differential drive mobile robots. Journal of intelligent & robotic systems. 2017 Feb;85:277-92.