Robotica (2023), **41**, pp. 530–547
doi:10.1017/S0263574722001400

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Regulation of cost function weighting matrices in control of WMR using MLP neural networks

Moharam Habibnejad Korayem*, Hamidreza Rezaei Adriani and Naeim Yousefi Lademakhi

Robotics Research Laboratory, Center of Excellence in Experimental Solid Mechanics and Dynamics, School of Mechanical Engineering, Iran University of Science and Technology, Tehran, Iran
*Corresponding author. E-mail: hkorayem@iust.ac.ir

Received: 19 May 2022; **Revised:** 19 July 2022; **Accepted:** 5 August 2022; **First published online:** 28 October 2022

Keywords: intelligent regulation gains, optimal control, multi-layer perceptron neural networks (MLP-NN), wheeled mobile robot (WMR)

Abstract

In this paper, a method based on neural networks for intelligently extracting weighting matrices of the optimal controllers' cost function is presented. Despite the optimal and robust performance of controllers with the cost function, adjusting their gains which are the weighting matrices for the system state variables vector and the system inputs vector, is a challenging and time-consuming task that is usually selected by trial and error method for each specific application; and even little changes in the weighting matrices significantly impact problem-solving and system optimization. Therefore, it is necessary to select these gains automatically to improve controller performance and delete human energy to find the best gains. As a linear controller, linear quadratic regulator, and as a nonlinear controller, nonlinear model predictive control have been employed with trained networks to track the path of a wheeled mobile robot. The simulation and experimental results have been extracted and compared to validate the proposed method. These results have been demonstrated that the intelligent controller's operation has lower error than the conventional method, which works up to 7% optimal in tracking and up to 19% better in angle state error; furthermore, as the most important aim, the required time and effort to find the weighting matrices in various situations has been omitted.

1. Introduction

As science and technology developed, robots and their improvement have been needed to be increased in various fields. Mobile robots have been successfully employed in different areas, such as medical, planetary exploration, agricultural machinery, and object moving [1–4]. So far, many control methods have been used for mobile robots, and for choosing one of them, systems' linearity and nonlinearity, the design purpose, environment conditions, and the issue's constraints should be considered [5]. The proportional – integral – derivative (PID) controller's simple structure is superior to other linear controller types. In order to increase the performance accuracy of this controller, Elsisi *et al.* have presented a method based on modified neural networks to extract the optimal gains of this controller, which performed better than the conventional way for robotic manipulators [6]. However, controllers with more optimal capabilities have been studied and employed as control methods develop and the more profound need for more accuracy, speed, and flexibility. The linear quadratic regulator (LQR) is another type of linear controller known as a common and widely used type due to its optimal solution. Xin *et al.* have implemented the system's controller with reverse dynamic control to track the path of a four-legged mobile robot reliably; in this way, the current status of the robot is rapidly updated, and the desired values have been sent to the actuators by minimizing the cost function of LQR controller [7]. In another study, they have used the LQR controller with the Extended Kalman Filter Estimator method for underwater mobile robot positioning problems and proposed a method to determine it accurately [8]. Trimp *et al.* have suggested a method for extracting

better coefficient from the LQR controller's cost function to reduce inverted pendulum vibrations. This method's problem is its dependence on SPSA optimization parameters, and its optimization is limited to the last step data [9]. Big bang-big crunch (BB-BC) optimization algorithm has been used for a pendulum on a cart to find the cost function's weighting matrices [10]. Kayacan and Chowdhary have proposed the Tracking-Error Learning Control algorithm to improve off-road mobile robots' tracking. In this algorithm, the feedforward control actions are updated using tracking error dynamics, and the problem of the model is eliminated. The results showed improved performance in the control system [11]. Dönmez *et al.* have proposed a method called Bi-RRT based on vision for path planning a wheeled mobile robot (WMR) and obstacle avoidance [12].

Since most systems are generally nonlinear, there is a considerable difference between the actual and the modeled system when the system is linearized, which reduces the accuracy of the controller's performance in practical applications. In this regard, using a nonlinear model that needs a nonlinear controller to solve makes a better outlook of the system's actual behavior. State-Dependent Riccati Equation, which was studied for the first time in robotics by Innocenti *et al.*, can be described as one of the useful and optimal nonlinear controllers [13]. Korayem *et al.* have employed this controller to determine the maximum load of mechanical manipulators with flexible joints. R and Q matrices' effects on the path have been considered, and the results compared with the LQR controller [14]. A finite-time feedback linearization controller has been implemented on the manipulators of a mobile robot. The optimal gain is obtained by solving the state-dependent differential Riccati equation in a time-varying dynamic system to achieve a finite time constraint [15]. Hun Lee *et al.* have suggested a method for generating and regulating joint torque at each time step in legged robots. In this way, actual torque or force has been directly used in control loops instead of feedforward force, which results in better performance [16].

Korayem *et al.* have introduced an approach to improve the sliding mode control (SMC) in the robotic manipulators called nonsingular terminal SMC, simulated and implemented for the Scout mobile robot manipulators. The results indicate that this method's performance error was less than the conventional method [17].

The model predictive control (MPC) is a perfect controller for mobile robots in different environments. This controller predicts some of the future steps to make better decisions, especially in unknown environments, so its performance is desirable and reliable. This controller's accuracy and solution speed for a differential mobile robot has been studied and compared in the nonlinear and stable mode [18]. In another study, the problem of path tracking control has been solved by combining MPC for the kinematic part and adaptive control for the dynamic part [19]. A Hierarchical Decomposed-Objective based on Model Predictive Control (HiDO-MPC) has been proposed to improve the performance of this controller used in rescue robots to approach the casualty, which has demonstrated results in enhancing accuracy and speed [20].

Extracting a model close to the existing system is necessary for good control performance. Fierro and Lewis have provided the kinematic and dynamic model of WMRs with n-dimensional configuration space S. Also, a robust-adaptive controller based on neural networks for trajectory tracking has been proposed [21]. Another dynamic model of a WMR with manipulators has been extracted and presented by recursive Gibbs–Appell formulation. This method's benefit is omitting computing the Lagrange multipliers associated with the nonholonomic system constraints [22]. Dönmez *et al.* have presented a visual-based technique to increase the accuracy of WMRs' controller act. The kinematic model has been extracted using the general Gaussian function, and the system control works in real-time. In this method, they need to calculate just one function to adjust system velocity parameters, which is a faster method for real time [23].

The development of machine learning has solved the absence of an exact mathematical system model and causes no more trial and error to achieve the controlling desire. It is reasonable to say that combining classic and modern controllers with machine learning methods can be helpful and achieve excellent accuracy. Chen *et al.* have presented an adaptive neural networks control scheme for a WMR to compensate system's uncertainties by approximating its uncertain parameters, which improved system robustness [24]. Recently, Peng *et al.* have developed neural networks based on adaptive control for optimal path

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

tracking when environmental information is unavailable [25]. A computational method called QRnet based on machine learning has been proposed to design optimal gains for high-dimensional nonlinear systems using an LQR controller. However, this method has been limited to the "is it possible to approximate the original problem to a linear-quadratic system" question [26]. Yan *et al.* have presented an approach based on robust MPC influenced by finite uncertainties for discrete-time nonlinear systems. At first, the system has been linearized, and then the MPC signal has been optimized with a recursive neural network [27]. Also, an nonlinear model predictive control (NMPC) technique based on Gaussian Particle Swarm Optimization has been employed, that a linear equivalent is extracted and solved for the system in each solution's step; however, it causes a reduction in the method solution's speed [28]. Linearizing the model in the two last methods increases practical errors.

There are some gains in each controller, which must be optimally extracted and regulated for better performance. In controllers with cost function, there are two weighting matrices, Q and R, to optimize and obtain control output state and system input variables, which changing them affects the tracking of the path significantly. Therefore, it is necessary to extract these gains for each situation by trial and error method to improve the controller performance, and it requires a lot of time and energy. This paper proposes an intelligent controller based on neural networks, with the following four innovations: 1. Extracting the cost function's weighting matrices for every path without wasting operator time and effort. 2. Improving the controller's accuracy by intelligently selecting the gains separately in each solution step. 3. Presenting a predictive training method based on MLP-NN to extract gains with an outlook on future steps. 4. Better accommodating the networks with new paths by training them based on system state variables. The structure of the paper is illustrated as follows. Part II will discuss the MLP-NN structure and the suggested method for network training to extract control gains. There is a review of two controllers with the cost functions, LQR and NMPC, in Section III. Section IV deals with the kinematic modeling of a wheeled mobile robot (WMR) in both linear and nonlinear formulations and introduces the experimental robot. Then, in Section V, in addition to simulation results and their comparison, actual test results for the four-WMR are presented and reviewed. Discussing the results is followed in Section VI, and finally, the conclusions are provided in Section VII.

2. Controller's weighting matrices regulation

2.1. Neural network training method considering the future steps

The method starts with data collecting and network training. The most critical question is: what inputs are suitable for training the network? In this case, the first method for tracking by a mobile robot seems to be putting the path type as the network's input and the best Q and R for each of them as the network's output. There are some problems with this assumption; first, the amount of data is limited by path type, and a meager amount of data is extracted; so the network has a low accuracy for new paths. Second, the network is not trained in the actual control concept (the system's state variables) and causes little efficiency in new situations. Therefore, the best solution is to collect and use data based on the system's state variables at each time step; In this way, the differences between the variables of the current and the reference states are given to the neural network at every time step, and its weighting matrices are extracted and used. So, not only can much more data be extracted to train the network, but also it will work much better for new paths since states' errors are more adaptable than the path type.

However, the best way to train has been discovered, but the network decisions are still limited to the current particular step. The proposed method can perform better along the path by entering the system's future state variables in network decisions. In this case, the network trained with this method will extract control gains by looking at the future path changes. So, according to Fig. 1, the network's input is extracted in the form of Eq. (1).

$$\psi(i) = \left[q_{1_i}(k) \quad q_{2_i}(k) \quad \dots \quad q_{n_i}(k) \quad q_{1_i}(k+1) \quad q_{2_i}(k+1) \quad \dots \quad q_{n_i}(k+P)\right]^T \tag{1}$$

Where n is the number of state variables of the system, $q_{1_i}(k)$ is the difference between the first state variable and its reference state variable at the k moment, $q_{n_i}(k)$ is the difference between the n state

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

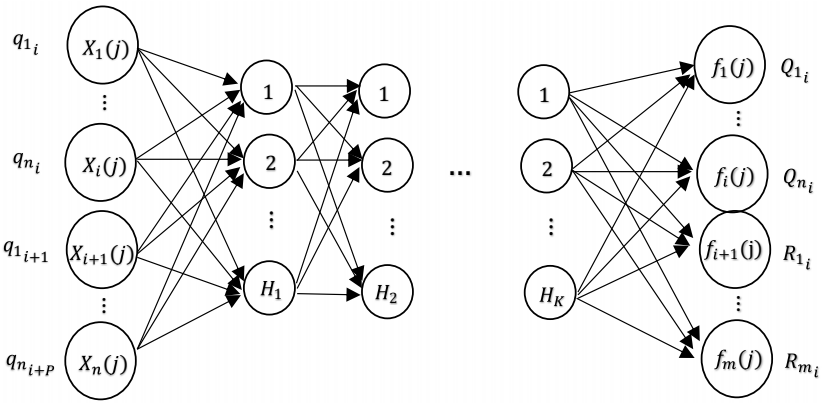

Figure 1. *Architecture of a neural network with k hidden layer(s).*

variable and its reference at the exact moment, $q_{1_i}(k+1)$ is the difference between the first reference state at the k and $k+1$ moments, and finally $q_{n_i}(k+P)$ is the difference between the n reference state at the $k+P$-1 and $k+P$ moments. The data extraction has been done by Algorithm 1 to train the network.

Algorithm 1: Predictive learning of neural networks

Require: Sampling time $\boldsymbol{t_s}$, Number of step variables $\boldsymbol{N_t}$, Number of Future steps for learning P, Weighting matrix for system inputs R, Weighting matrix for system states Q.

1: **while** $\boldsymbol{i}$ is less than $\boldsymbol{N_t}$ **do**
2: **while** finding the best of Q and R matrices, **do**
3: choose a Q matrix and an R matrix
4: calculate errors between desired states and states of ith step
5: compare the error with previous steps
6: **end while**
7: **for** $\boldsymbol{j = i : i + P}$
8: calculate the errors between states of $(\boldsymbol{j+1})$ and $\boldsymbol{j}$
9: **end for**
10: organize the input vectors for learning by line 4 and line 8
11: organize the output vectors for learning by the best of Q and R matrices
12: **while end**

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

3. Structure of controllers with cost function

In this paper, NMPC as a nonlinear controller and LQR as a linear controller have been used to validate the content. The primary reason for choosing them is that these two controllers have the cost function for optimization and require appropriate Q and R matrices. In addition, the NMPC is a perfect option for path tracking with a mobile robot since it can predict some of the future steps. The LQR controller has the flexibility to regulate better system inputs and inherent resistance to noise and disturbance.

3.1. LQR controller

Suppose the system equation in discrete linear form as Eq. (2) is extracted:

$$x(k+1) = A(k)x(k) + B(k)u(k)$$
$$y(k) = C(k)x(k) + D(k) \tag{2}$$

The matrices A(k) and B(k) are the coefficients of the control state variables and the system inputs, respectively. Also, C(k) is defined as the state coefficient matrix in the system output equation, and external uncertainties and disturbances define D(k). The system's cost function as Eq. (3) is minimized in this controller method:

$$J_{LQR} = \sum_{k=0}^{\infty} \left(X(k)^T Q X(k) + u(k)^T R u(k) \right) \tag{3}$$

Q and R values must be selected based on design requirements. After finding the coefficient matrix F(k), the system input is extracted as Eq. (4) by solving the controller at each step.

$$u(k) = -F(k)x(k) \tag{4}$$

3.2. NMPC

A discrete nonlinear system as (5) is assumed:

$$x(k+1) = f(x(k), u(k))$$
$$y(k) = h(x(k)) \tag{5}$$

Which includes the constraints $u_{min} \le u(k) \le u_{max}$and $y_{min} \le y(k) \le y_{max}$. $x(k) \in R^n$ is the state variables vector and $u(k) \in R^m$ is the control input vector and also $y(k) \in R^p$ is the output vector. According to [29], the cost function of this controller has been extracted as Eq. (6):

$$J_{NMPC} = \varnothing(y(k+N|k)) + \sum_{i=0}^{N-1} L(y(k+i|k), u(k+i|k), \Delta u(k+i|k)) \tag{6}$$

In this equation, $u(k+i|k)$ input and $u(k+i)$ are calculated from the available data at the k step, and also $y(k+N|k)$ output and $y(k+N)$ comes from the available data at the k step. C is the control horizon, N is the prediction horizon, and as Eqs. (7) and (8), φ and L are two nonlinear functions that form the cost function.

$$\begin{aligned} L = {} & (y(k+i|k) - y_s(k))^T Q (y(k+i|k) - y_s(k)) \\ & + (u(k+i|k) - u_s(k))^T R (u(k+i|k) - u_s(k)) + \Delta u^T (k+i|k) S \Delta u(k+i|k) \end{aligned} \tag{7}$$

$$\varphi = (y(k+N|k) - y_s(k))^T Q (y(k+N|k) - y_s(k)) \tag{8}$$

Where the targets of the steady states y and u are y_s and u_s.

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

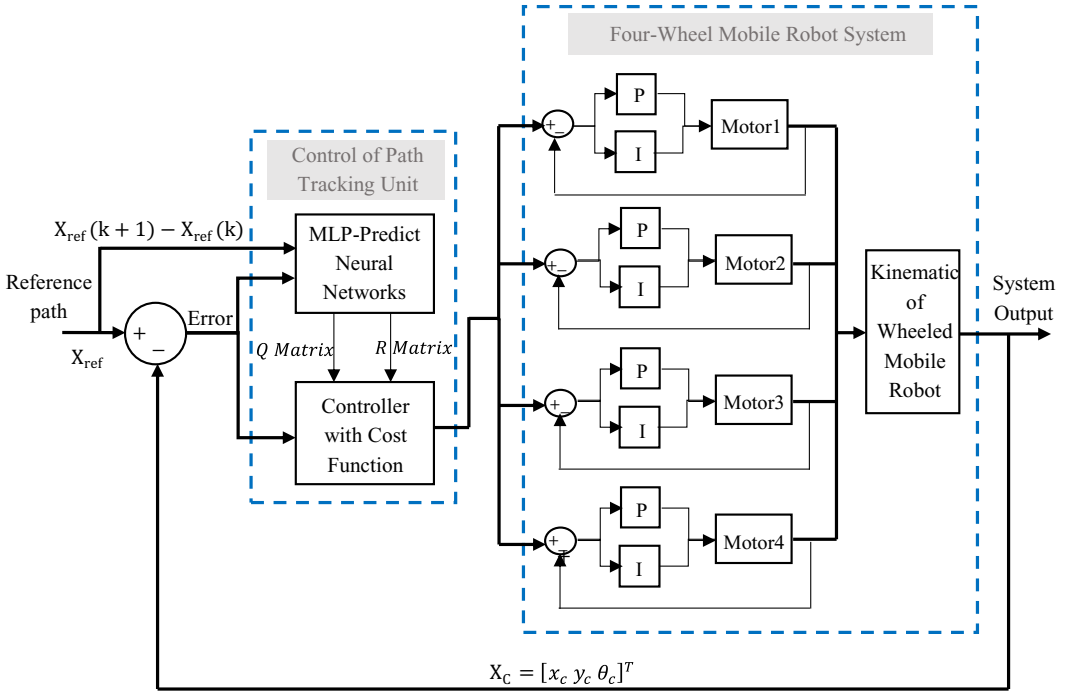

Figure 2. *Intelligent path tracking block diagram for WMRs with adjustment of the cost function.*

Also, Q and R are, respectively, n × n and m × m, where n is the number of the system's state variables and m is the number of control inputs. The difference between system inputs at the current step and the previous step is defined as $\Delta u(k+i|k) = u(k+i|k) - u(k+i-1|k)$, and S has been used as the weighting matrix of the Δu. Finally, the block diagram of the intelligent controller for tracking the path by the WMR is shown in Fig. 2. As can be seen, a PI controller is used to control the speed of the Dynamixel motors for their better performance, and the motors use this controller by default. Relative to the reference coordinates, $X_c(k)$ is the states value of the robot's center of mass (COM) at the k step, and $X_{ref}(k)$ is the state value of reference at the same step.

4. WMR modeling and introduction

4.1. Kinematic of a WMR

The schematic of the system, the mechanical motion structure, and the distribution of velocity vectors have been shown in Fig. 3, where the robot's center of mass is marked with COM. As can be seen, the robot on the screen moves in the x and y directions and has a rotation angle θ. An ideal reference robot has been considered to follow the path and move according to the desired values, so the real robot must always follow it.

The state variables of the system are assumed to be $q = \begin{bmatrix} x\ y\ \theta \end{bmatrix}^T$ which are related to the general coordinates. It can be considered that there is no slip in the system, so the friction between the wheels and the floor has been neglected in this study. The linear velocity of each wheel is equal to the angular velocity of the wheel multiplied by its radius. Thus the kinematic of the WMR is extracted as Eq. (9) [30].

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

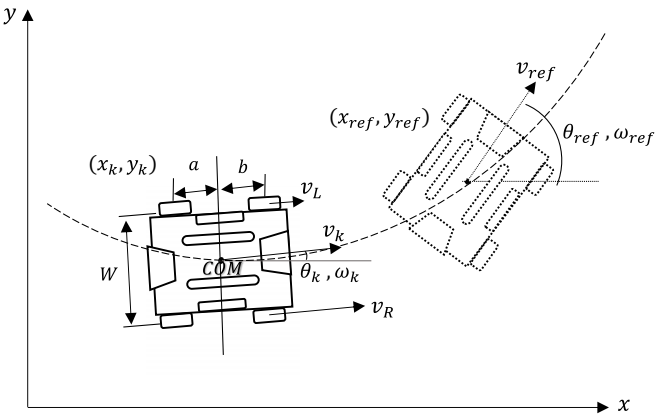

Figure 3. *The schematic of the robotic motion structure and the distribution of velocity vectors.*

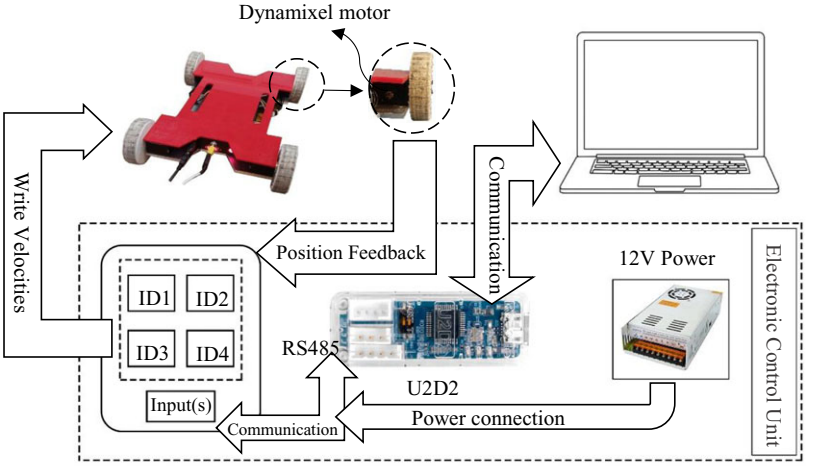

Figure 4. *Schematic of the experimental setup to track the path of a four-WMR.*

$$\begin{bmatrix} \dot{x}_c(t) \\ \dot{y}_c(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} cos\theta(t) & 0 \\ sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{COM}(t) \\ \omega_{COM}(t) \end{bmatrix} \tag{9}$$

Where $v_{COM}(t)$ is the linear velocity of the robot's COM in the direction of the longitudinal axis, and $\omega_{COM}(t)$ is the angular velocity of the robot's COM around its vertical axis. Physically, two wheels on the same side of the robot must move at the same speed. In this case, the conversion matrix of the COM's linear and angular velocities to the right and left velocities has been expressed as Eq. (10), that W is the width of the robot.

$$\begin{bmatrix} v_{COM}(t) \\ \omega_{COM}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{W} & -\frac{1}{W} \end{bmatrix} \begin{bmatrix} v_L(t) \\ v_R(t) \end{bmatrix} \tag{10}$$

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

Table I. *The dimensional and mass characteristics of the robot.*

Unit	Value	Parameter
mm	380	Length of robot
mm	270	Width of robot
mm	50	Radius of wheels
rpm	53	The maximum value of motors speed
deg	0.08	Accuracy of motors for reading angles
kg	2.7	Mass of robot
kgm^2	0.062	Moment of inertia of base about Z-axis
kgm^2	0.00028	Moment of inertia of wheels about the rotation axis

The discretized kinematic model of the WMR corresponding to Fig. 3 has been extracted as Eq. (11).

$$\begin{aligned} X(k+1) &= f(X(k), u(k)) \\ x(k+1) &= x(k) + (v(k)\cos\theta(k))t_s \\ y(k+1) &= y(k) + (v(k)\sin\theta(k))t_s \\ \theta(k+1) &= \theta(k) + \omega(k)t_s \end{aligned} \tag{11}$$

Where k is the current step and t_s is sampling time. The system equations are linearized using the Taylor method's linear controller in WMR. According to Ref. [31], the system's discrete equations can be extracted as Eq. (12), and the matrices $A(k)$ and $B(k)$ can be seen as Eqs. (13) and (14), respectively:

$$\tilde{q}(k+1) = A(k)\tilde{q}(K) + B(K)\tilde{u}(k) \tag{12}$$

$$A(k) = \begin{bmatrix} 1 & 0 & -v(k)\, sin\theta_r(k)\, t_s \\ 0 & 1 & v_r(k)\, cos\theta_r(k)\, t_s \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$B(k) = \begin{bmatrix} cos\theta_r(k)\, t_s & 0 \\ sin\theta_r(k)\, t_s & 0 \\ 0 & t_s \end{bmatrix} \tag{14}$$

4.2. Experimental setup of the robot

A four-WMR has been used to validate the results, shown in Fig. 4. The motors of this robot are Dynamixel type and XM540-W150-r model, which can be networked with each other, and various information such as position, speed, motor load, and input voltage can be extracted while receiving commands from the central controller. The robot's main body is made of aluminum, which all motors, the network of motors, wires, and connections are placed on it. Finally, they are covered with a plexiglass surface, and A 12(V) - 30(A) power supply is used for the motors. The dimensional and mass charcteristics of the robot are shown in Table I.

To get feedback from the rotation angle of the robot body, the output of the motors' encoders, or in other words, the amount of the wheels' rotation, have been used; Thus, the rotation ratio of the right and left wheels determines the value of the angle.

For example, when the wheels' rotation on both sides is equal, the body's rotation value is zero, and when the rotations of the two sides are opposite, the robot rotates around itself, and the angle value is

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

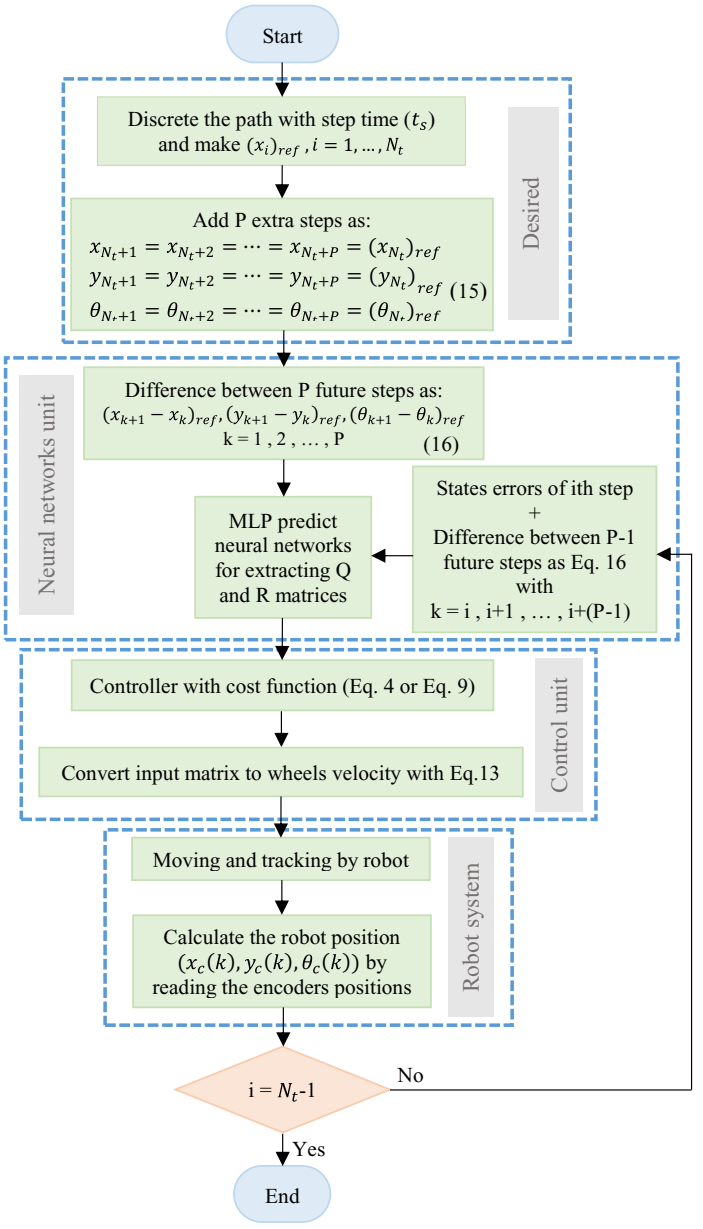

Figure 5. *Executive flowchart of intelligent control to track the path by a WMR.*

determined depending on the different rotation between two sides at each step time. Then, according to the distance between the robot's COM and the wheels, the x and y positions of the robot are calculated, and the system state variables' errors continue the loop.

Also, for better expression, the schematic of the experimental setup has been presented in Fig. 4. A U2D2 interface in the electrical box takes commands from the relevant software and sends them to the motors' board via the RS485 protocol; also, the power supply is connected to the motors' board. Finally, motors send requested information as feedback to the software, and the loop is completed.

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

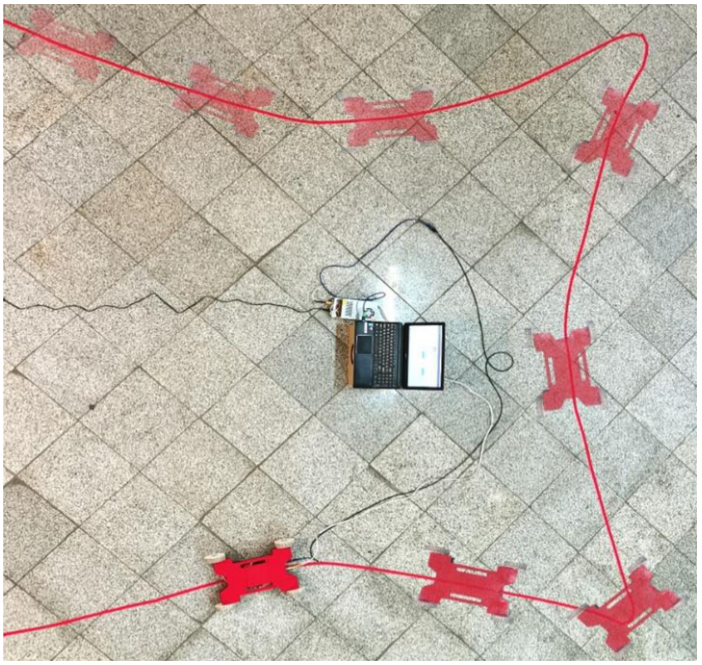

Figure 6. *Four-WMR during the experimental test and tracking the path.*

5. Simulation and experimental tests

5.1. Network training to track the path

The number of future steps entered in computation has been considered as $P = 29$ for the WMR path tracking with $n = 3$ state variables, x, y, and θ; So there is 90 datum in each data set as the network input. The number of output data in each data set is 5, which includes three matrix elements of $Q_i = diag[Q_{1i} \;\; Q_{2i} \;\; Q_{3i}]$ and two matrix elements of $R_i = diag[R_{1i} \;\; R_{2i}]$. Therefore, with this operation, the output vector which is the best weighting matrices of the specific state, has consisted, and network training of MLP has been performed by the input and output vectors. At this step, the number of network layers and the number of nodes of each hidden layer are required. In this regard, the best numbers of the final network have been selected by testing different numbers and their performance results. The experiment test by the 4-WMR has been shown in Fig. 5.

5.2. NN-LQR controller

At first, training data were collected for network learning, and 16,000 data sets were extracted by solving several different paths. According to Fig. 1, the best number of layers has been chosen after consecutive training as $K = 4$, and the number of hidden layers nodes as $H_1 = H_2 = H_3 = H_4 = 5$. This network has been trained in MATLAB software, and its function has been extracted; the least-squares error for this network training was 0.38×10^{-5}. An algorithm is needed to find the minimum function to implement this network with the MLP method.

In this paper, the Levenberg-Marquardt Algorithm (LMA) has been used to find the minimum of multivariate nonlinear functions. In many cases, this method is more resistant than the Gauss-Newton method and gives the desired answer, even when the starting point is far from the final minimum. Fig. 6 shows the four-WMR tracking selected path during the experimental test.

In Fig. 7, the desired path is sigma-shaped, with various curves along the way. Test results in the desired path for simulation test with the LQR controller (LQR), with the neural network (NN LQR),

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

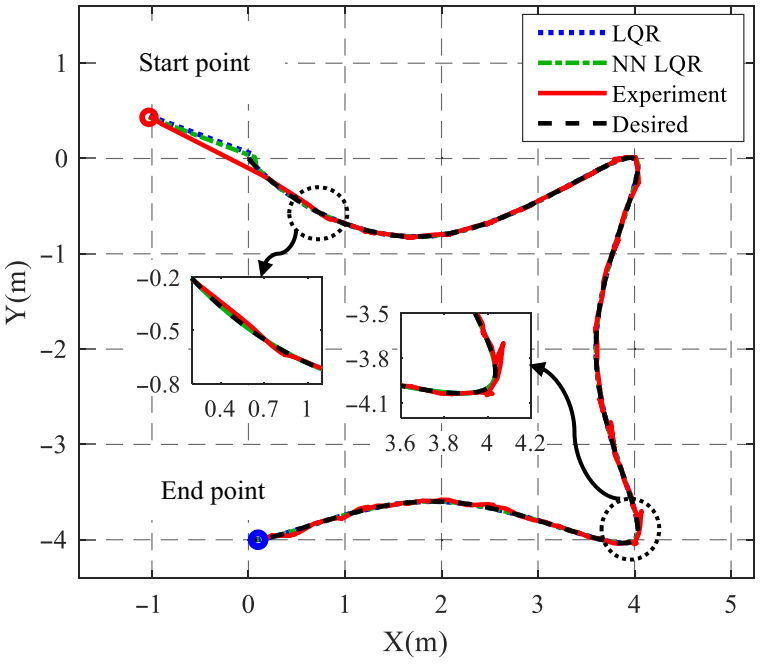

Figure 7. *Path tracking in simulation and experimental test of LQR controller.*

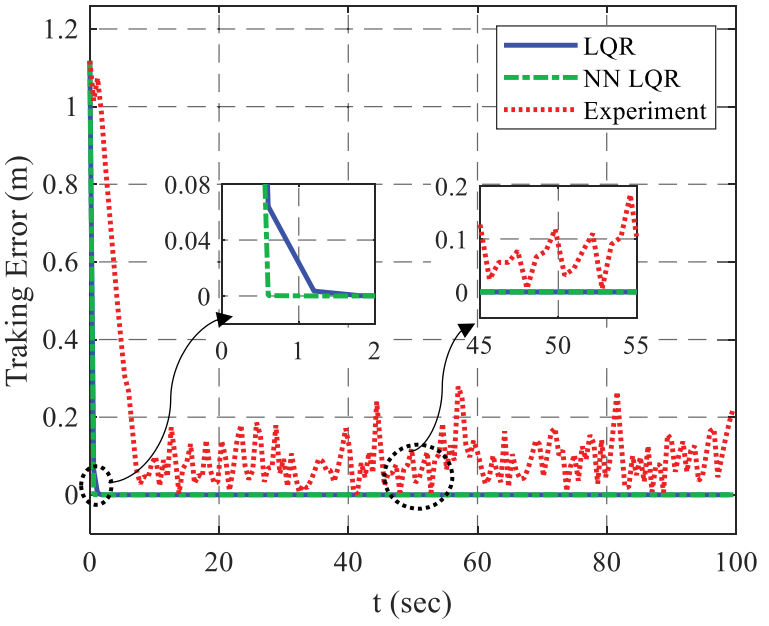

Figure 8. *Tracking error relative to the reference path at each step.*

and the path taken by the real four4-WMR with the intelligent controller (Experiment) has been shown. The best Q and R were extracted by trial and error method for the LQR test. However, the path's information was unfamiliar to the network, but the NN LQR trained network has optimally obtained the best gains, and the robot tracked the path very well. As can be seen, in the experimental test, the path taken by the robot has slight oscillations around the desired path, which is part of the actual system reality.

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

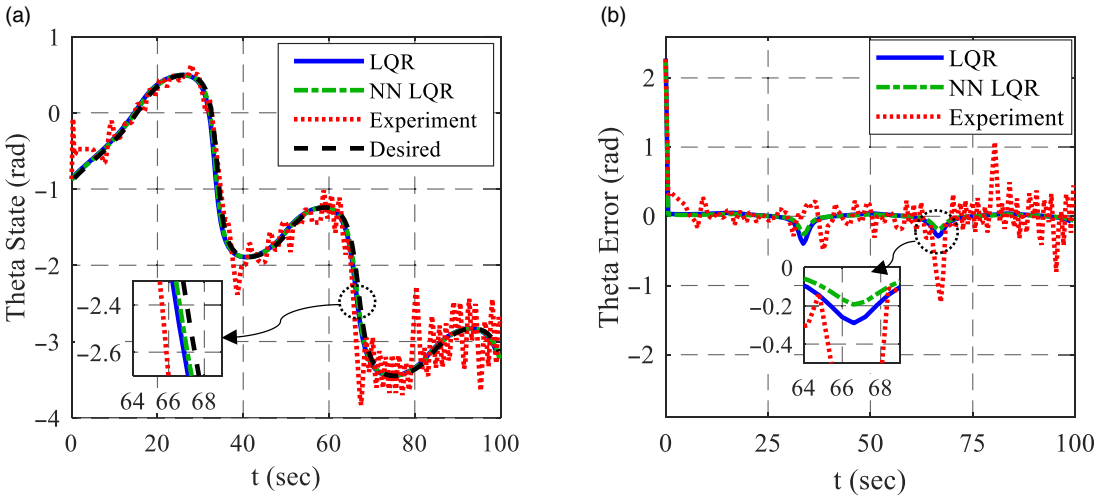

Figure 9. *(a) Theta state rate in simulation and experimental test. (b). Theta state error in simulation and experimental test.*

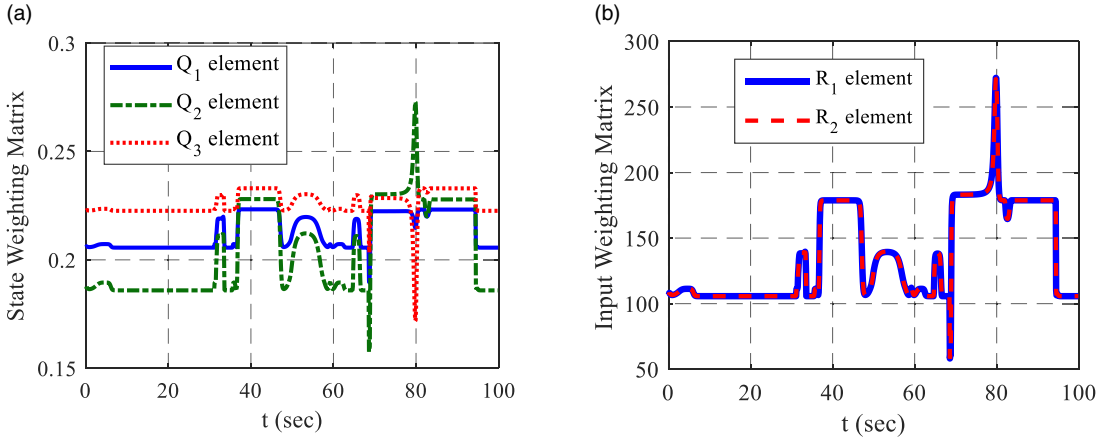

Figure 10. *(a) Three diagonal elements of the Q matrix extracted from the neural network. (b) Two diagonal elements of the R matrix extracted from the neural network.*

Figure 8 shows the path tracking error, obtained as the least-squares error method of two states x and y as Eq. (15).

$$Tracking\ Error(k) = \sqrt{\left(x_c(k) - x_{ref}(k)\right)^2 + \left(y_c(k) - y_{ref}(k)\right)^2} \tag{15}$$

The path error in the NN LQR mode is less than the LQR in the simulation test. However, the path error is generally low in the simulation test due to the parametric solution. To have a better outlook of the difference between the two simulation tests in the linear controller, Fig. 9(a) shows the angle state variable in the considered path, and the difference between the value of this state variable and its reference in each step is referred in Fig. 9(b). As can be seen, this state's error in the controller with the proposed method is lower, especially in the corners of the path.

Data obtained graph related to the weighting matrices from the NNLQR controller's network along the path in form (10a) is related to diagonal elements of the Q matrix, which have been optimally

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

Table II. *Compared matrices elements between LQR controllers with the same condition.*

Parameters configuration	Step time (sec)	Desired path starting point	Robot starting point	Q Matrix	R Matrix
LQR	0.1	[0 0 −1]	[−1 0.5 −0.9]	Constant [0.2 0.19 0.22]	Constant [110 110]
NN-LQR	0.1	[0 0 −1]	[−1 0.5 −0.9]	Variable [(0.18–0.23) (0.16–0.27) (0.17– 0.24)]	Variable [(55–272) (55–272)]

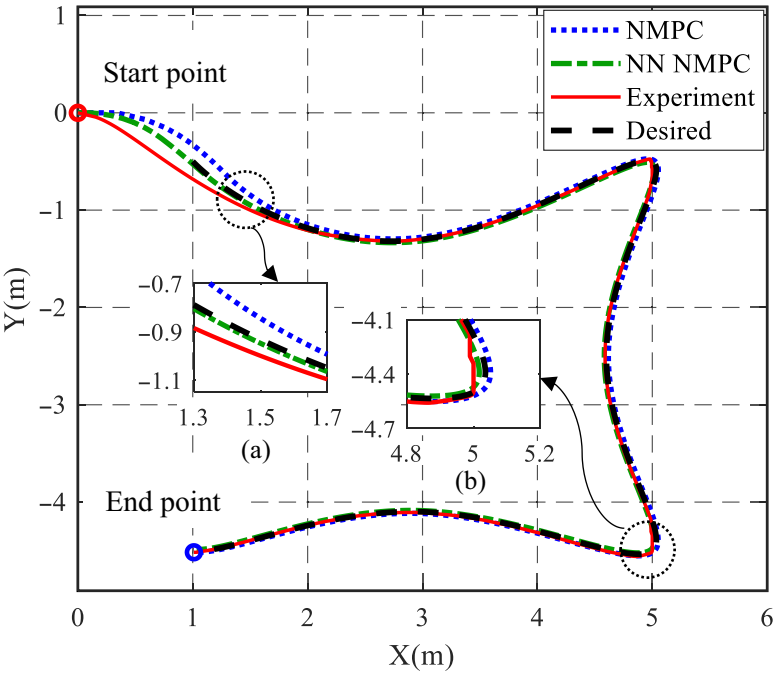

Figure 11. *Path tracking in simulation and experimental test of NMPC controller.*

extracted in each step of the path. All three elements are separately extracted in different amounts; however, their changes are in the same rhythm based on network training. Figure 10(b) shows two diagonal elements of the R matrix, which are approximately the same values based on the training.

One of the challenges in using the intelligent control method is the solution time. Mentioning that the proposed method has several advantages, such as reducing tracking error, increasing path convergence speed, and saving time and human energy to select control gains; the intelligent controller solution time under the same conditions has been increased by less than 3% compared to the conventional method, which is not comparable with its advantages.

Finally, Table II shows the Q and R matrix elements for two LQR and NNLQR controllers in the same conditions. As can be seen, the amount elements related to the LQR controller are fixed and were the most optimal amount for these conditions.

5.3. NN-NMPC controller

The NMPC has been used as a nonlinear controller for network training, and more than 18,000 data sets have been collected by solving different paths. Since the number of the network's inputs, outputs, and

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

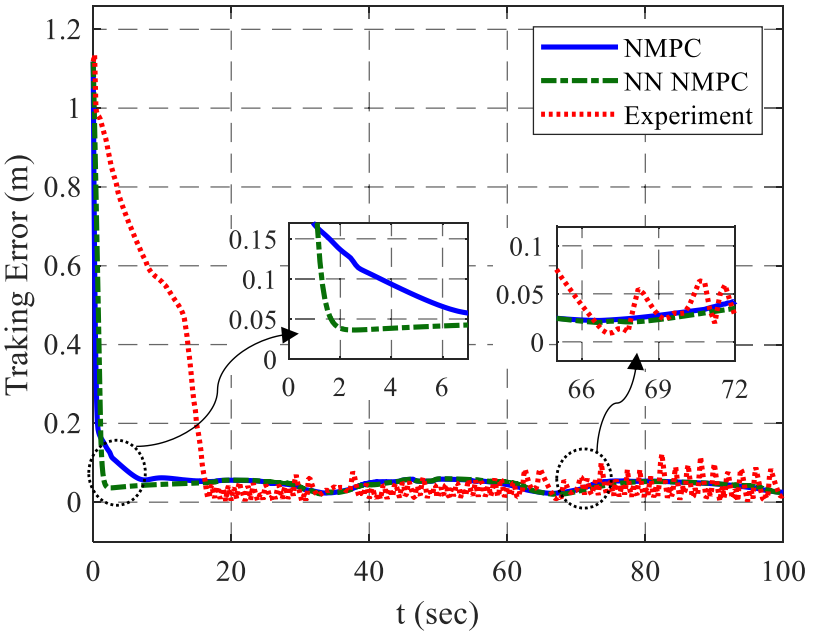

Figure 12. *Tracking error relative to the reference path at each step.*

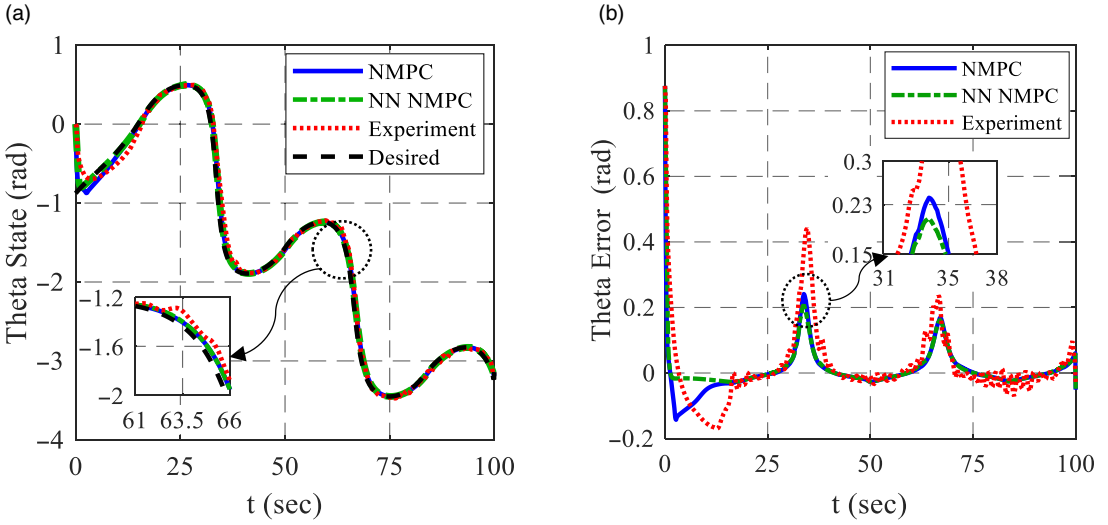

Figure 13. *(a) Theta state rate in simulation and experimental test. (b) Theta state error in simulation and experimental test.*

data types is the same as the previous network, the number of layers and nodes used in the previous network training has also been applied in this network training. The least-squares error for this network training was 0.64×10^{-5}. The simulation code has been prepared in MATLAB, and the sigma-shaped path figures are extracted as Fig. 11. This figure shows the test results in the desired path for the simulation with the NMPC controller (NMPC), with the neural network (NN NMPC), and the path taken by the actual robot with the intelligent controller (Experiment). As can be seen, in the simulation, by applying the neural network and determining the appropriate weighting matrices in each step, the robot got closed to the path with more convergence speed and had lower error than the conventional method, especially

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

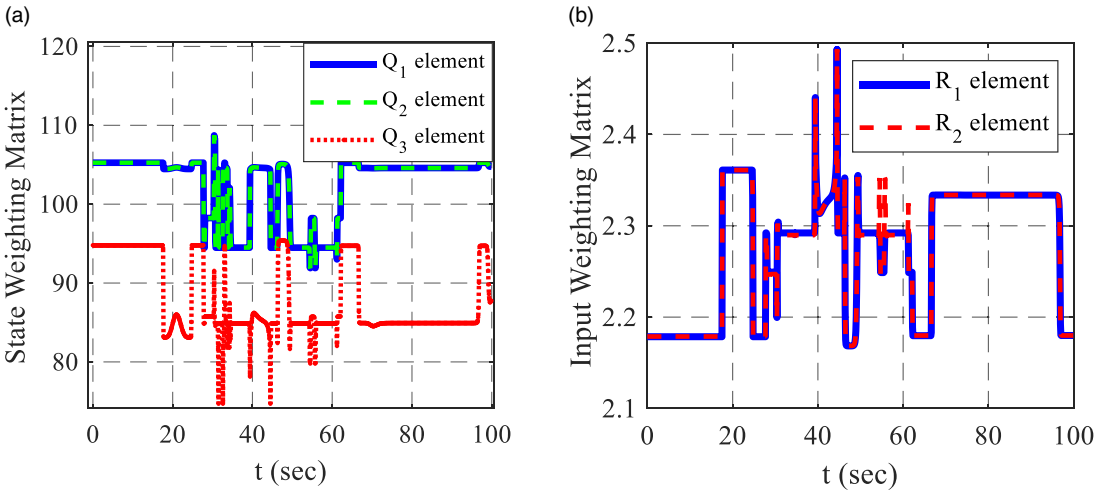

Figure 14. *(a) Three diagonal elements of the Q matrix extracted from the neural network. (b) Two diagonal elements of the R matrix extracted from the neural network.*

in the path's corners, as shown in Fig. 11(b). In the experimental test, the robot has successfully reached the path and has followed it optimally.

Sampling error and the controller's endeavor to improve tracking affected the data and caused small oscillations in the experimental test, presented in Fig. 12. However, it is normal and insignificant in the experimental.

Figure 13(a) shows change rates in the robot body's rotation angle during solution time. The intelligent controller obtained a better performance with a smaller error than the conventional controller, as displayed in Fig. 13(b).

Figure 14(a) shows the diagonal elements of the Q matrix along the considered path, and as can be seen, the optimal gains are extracted at each path step. According to the network training, the two elements Q_1 and Q_2, which are the coefficient of the two states x and y, have been extracted almost identically, and Q_3, which is related to the angle state, was different from them but with the same rate of change. Figure 14(b) relates to two elements of the R matrix which R_1 is the coefficient of the robot linear velocity, and R_2 is the coefficient of the robot angular velocity.

Like the previous controller, this controller's solving time in the proposed method has been measured and compared to the conventional method. Under the same conditions, the intelligent controller solution time has been increased by less than 4%, which is not comparable with its advantages. Also, Table III displays the Q and R matrix elements for two NMPC and NN-NMPC controllers in the same conditions.

6. Discussion of results

Generally, with adding the trained networks to controllers, there is no need to extract the weighting matrices by the trial and error method anymore, and the networks have extracted these gains. In addition, as can be seen in the figures, intelligent controllers' performance is better and optimizer than the usual methods, which work up to 7% optimal in tracking (Fig. 11) and up to 19% better in angle state error (Figs. 9(b) and 13(b)). Controlling the system gets easier and simpler by linearizing it, so the system's error in tracking the path by a linear controller is less than a nonlinear one in the simulation test that can be found by looking closely at the diagrams. Nevertheless, a nonlinear controller performance in simulation is closer to reality. The actual system operates better and more efficiently with a nonlinear controller due to its nonlinear nature, but the linear controller has more oscillations around the path and

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

Table III. *Compared matrices elements between NMPC controllers with the same condition.*

Parameters configuration	Step time (sec)	Desired path starting point	Robot starting point	Q Matrix	R Matrix
NMPC	0.1	[1 −0.5 −0.9]	[0 0 0]	Constant [103 103 92]	Constant [2.25 2.25]
NN-NMPC	0.1	[1 −0.5 −0.9]	[0 0 0]	Variable [(92 – 109) (92 – 109) (75 – 95)]	Variable [(2.18–2.5) (2.18–2.5)]

less accuracy in following the path. This point can be followed by comparing the experimental tests of two linear and nonlinear models.

7. Conclusion

This paper proposes an intelligence approach based on MLP-NN for the controllers with the cost function. The controller's performance has been improved, and the path tracking error has been reduced. In addition, the need to spend time and human energy to adjust the weighting matrices has been eliminated. The proposed method can be implemented for all linear and nonlinear controllers with the cost function. A suggested method also causes better performance by entering the calculations of future steps in the extraction of weighting matrices. In order to validate the performance of NN LQR and NN NMPC controllers, the results have been compared with their conventional methods in simulation mode. Also, the experimental test results obtained from the 4-WMR show the excellent performance of the presented method in a real environment. In this case, the trained network obtains the best weighting matrices in each step for any path or starting point selected for each robot, and the controller optimally tracks the path. The intelligent controllers performed up to 7% better than the conventional method in path tracking and up to 19% better in angle state error in corners of the path.

Authors' contributions. M. H. Korayem and N. Y. Lademakhi conceived of the original idea. This was also discussed with H. R. Adriani, and all authors agreed with this paper's main focus and idea. The test's plan was designed by H. R. Adriani and N. Y. Lademakhi, and H. R. Adriani carried out the simulations and the experimental tests under the supervision of M. H. Korayem. The first version of the paper was written by H. R. Adriani and M. H. Korayem, and N. Y. Lademakhi provided critical feedback and helped shape the research, analysis, and manuscript.

Financial support. Professor Korayem funded this study at Iran University of Science and Technology.

Conflicts of interest. The authors declare that they have no conflicts of interest.

Ethical considerations. None.

References

[1] Y. Hu, H. Su, J. Fu, H. R. Karimi, G. Ferrigno, E. De Momi and A. Knoll, "Nonlinear model predictive control for mobile medical robot using neural optimization," *IEEE Trans. Ind. Electron.* **68**(12), 12636–12645 (2020).

[2] D. St-Onge, M. Kaufmann, J. Panerati, B. Ramtoula, Y. Cao, E. B. Coffey and G. Beltrame, "Planetary exploration with robot teams: implementing higher autonomy with swarm intelligence," *IEEE Robot. Autom. Mag.* **27**(2), 159–168 (2019).

[3] J. P. Vasconez, G. A. Kantor and F. A. A. Cheein, "Human-robot interaction in agriculture: a survey and current challenges," *Biosyst. Eng.* **179**, 35–48 (2019).

[4] E. Dönmez and A. F. Kocamaz, "Design of mobile robot control infrastructure based on decision trees and adaptive potential area methods," *Iran. J. Sci. Technol. Trans. Electr. Eng.* **44**(1), 431–448 (2020).

[5] H. Lee and H. J. Kim, "Trajectory tracking control of multirotors from modelling to experiments: a survey," *Int. J. Control Autom. Syst.* **15**(1), 281–292 (2017).

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

[6] M. Elsisi, K. Mahmoud, M. Lehtonen and M. M. Darwish, "An improved neural network algorithm to efficiently track various trajectories of robot manipulator arms," *IEEE Access* **9**, 11911–11920 (2021).
[7] G. Xin, S. Xin, O. Cebe, M. J. Pollayil, F. Angelini, M. Garabini, S. Vijayakumar and M. Mistry, "Robust footstep planning and LQR control for dynamic quadrupedal locomotion," *IEEE Robot. Autom. Lett.* **6**(3), 4488–4495 (2021).
[8] A. Alalwan, T. Mohamed, M. Chakir and T. M. Laleg, "Extended Kalman filter based linear quadratic regulator control for optical wireless communication alignment," *IEEE Photonics J.* **12**(6), 1–12 (2020).
[9] S. Trimpe, A. Millane, S. Doessegger and R. D'Andrea, "A Self-Tuning LQR Approach Demonstrated on an Inverted Pendulum," **In:** *IFAC Proceedings*, vol. **47** (2014) pp. 11281–11287.
[10] M. Almobaied, I. Eksin and M. Guzelkaya, "Design of LQR Controller With Big Bang-Big Crunch Optimization Algorithm Based on Time Domain Criteria," **In:** *24th Mediterranean Conference on Control and Automation (MED)* (IEEE, 2016) pp. 1192–1197.
[11] E. Kayacan and G. Chowdhary, "Tracking error learning control for precise mobile robot path tracking in outdoor environment," *J. Intell. Robot. Syst.* **95**(3), 975–986 (2019).
[12] E. Dönmez, A. F. Kocamaz and M. Dirik, "Bi-RRT Path Extraction and Curve Fitting Smooth with Visual Based Configuration Space Mapping," **In:** *International Artificial Intelligence and Data Processing Symposium (IDAP)* (IEEE, 2017) pp. 1–5.
[13] M. Innocenti, F. Baralli, F. Salotti and A. Caiti, "Manipulator Path Control using SDRE," **In:** *Proceedings of the American Control Conference, ACC, IEEE Cat. No. 00CH36334* (2000), pp. 3348–3352.
[14] M. H. Korayem, M. Irani and S. R. Nekoo, "Load maximization of flexible joint mechanical manipulator using nonlinear optimal controller," *Acta Astronaut.* **69**(7–8), 458–469 (2011).
[15] M. H. Korayem, S. R. Nekoo and S. Kazemi, "Finite-time feedback linearization (FTFL) controller considering optimal gains on mobile mechanical manipulators," *J. Intell. Robot. Syst.* **94**(3), 727–744 (2019).
[16] Y. H. Lee, Y. H. Lee, H. Lee, H. Kang, J. H. Lee, J. M. Park, Y. B. Kim, H. Moon, J. C. Koo, H. R. Choi, "Whole-body control and angular momentum regulation using torque sensors for quadrupedal robots," *J. Intell. Robot. Syst.* **102**(3), 1–15 (2021).
[17] M. H. Korayem, R. Shiri, S. R. Nekoo and Z. Fazilati, "Non-singular terminal sliding mode control design for wheeled mobile manipulator," *Ind. Robot. Int. J.* **44**(4), 501–511 (2017).
[18] H. R. Adriani, N. Y. Lademakhi and A. H. Korayem, "Superiority of Nonlinear and Stable MPC in a Differential Mobile Robot: Accuracy and Solving Speed," **In:** *9th RSI International Conference on Robotics and Mechatronics (ICRoM)* (IEEE, 2021) pp. 13–17.
[19] Y. Chen, Z. Li, H. Kong and F. Ke, "Model predictive tracking control of nonholonomic mobile robots with coupled input constraints and unknown dynamics," *IEEE Trans. Ind. Inform.* **15**(6), 3196–3205 (2018).
[20] R. P. Saputra, N. Rakicevic, D. Chappell, K. Wang and P. Kormushev, "Hierarchical decomposed-objective model predictive control for autonomous casualty extraction," *IEEE Access* **9**, 39656–39679 (2021).
[21] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot: backstepping kinematics into dynamics," *J. Robotic Syst.* **14**(3), 149–163 (1997).
[22] M. H. Korayem and A. M. Shafei, "Motion equation of nonholonomic wheeled mobile robotic manipulator with revolute-prismatic joints using recursive Gibbs-Appell Formulation," *Appl. Math. Model.* **39**(5–6), 1701–1716 (2015).
[23] E. Dönmez, A. F. Kocamaz and M. Dirik, "A vision-based real-time mobile robot controller design based on gaussian function for indoor environment," *Arab. J. Sci. Eng.* **43**(12), 7127–7142 (2018).
[24] Z. Chen, Y. Liu, W. He, H. Qiao and H. Ji, "Adaptive-neural-network-based trajectory tracking control for a nonholonomic wheeled mobile robot with velocity constraints," *IEEE Trans. Ind. Electron.* **68**(6), 5057–5067 (2020).
[25] G. Peng, C. P. Chen and C. Yang, "Neural networks enhanced optimal admittance control of robot-environment interaction using reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.* **33**(9), 1–11 (2021).
[26] T. Nakamura-Zimmerer, Q. Gong and W. Kang, "QRnet: optimal regulator design with LQR-augmented neural networks," *IEEE Cont. Syst. Lett.* **5**(4), 1303–1308 (2020).
[27] Z. Yan, X. Le and J. Wang, "Tube-based robust model predictive control of nonlinear systems via collective neurodynamic optimization," *IEEE Trans. Ind. Electron.* **63**(7), 4377–4386 (2016).
[28] J. Fan and M. Han. *Nonlinear Model Predictive Control of Ball-Plate System Based on Gaussian Particle Swarm Optimization, Congress on Evolutionary Computation* (IEEE, 2012) pp. 1–6.
[29] E. S. Meadows and J. B. Rawlings, "Model Predictive Control," **In:** *Nonlinear Process Control*, M. A. Henson and D. E. Seborg, Chapter 5 (Prentice-Hall, Englewood Cliffs, NJ, 1997) pp. 233–310.
[30] X. Wu, M. Xu and L. Wang, "Differential speed steering control for four-wheel independent driving electric vehicle," *IEEE Int. Symp. Ind. Electron.* **14**, 1–6 (2013).
[31] K. R. Sharma, F. Dušek and D. Honc, "Comparitive Study of Predictive Controllers for Trajectory Tracking of Non-Holonomic Mobile Robot," **In:** *21st International Conference on Process Control (PC)* (IEEE, 2017) pp. 197–203.

Appendix A

Theory of MLP-NN

According to the structure of neural networks, the output vector of the whole network is defined as Eq. (A1), assuming that the number of layers of the whole network is N.

$$f_N = \varphi_N(W_N f_{N-1} + b_N) \tag{A1}$$

https://doi.org/10.1017/S0263574722001400 Published online by Cambridge University Press

Where the input vector is considered as $X = [X_1 X_2 \ldots X_n]^T$ which n is the number of network inputs and the vector $f_N = \left[f_1 f_2 \ldots f_m\right]^T$ is the output vector of the whole network, that m is the number of the network's outputs. φ_N is the last layer activation function, W_N is the matrix connecting the N-1 hidden layer with the n hidden layer and b_N is the bias vector for the last hidden layer. In addition, f_{N-1} is assumed to be the output of the N-1 hidden layer. in the other word, the output of the neural network with K hidden layer can be extracted in the form of Eq. (A2):

$$f = W_K \max\left(0, W_{K-1} \max\left(\ldots max\left(0, W_1 X\right)\right)\right) \tag{A2}$$

Cite this article: M. Habibnejad Korayem, H. Rezaei Adriani and N. Yousefi Lademakhi (2023). "Regulation of cost function weighting matrices in control of WMR using MLP neural networks", Robotica **41**, 530–547. https://doi.org/10.1017/S0263574722001400