RESEARCH ARTICLE OPEN ACCESS

Comparative Analysis for NN-Based Adaptive Back-stepping Controller and Back-stepping Controller

Margarita Gjonaj*, BetimÇiço**, ArnisaMyrtellari***
**(Department of Automation, Polytechnic University of Tirana, Albania*
*** (Faculty of Contemporary Sciences and Technologies,South East European University,Macedonia)*
**** (Department of Automation, Polytechnic University of Tirana, Albania*

ABSTRACT
This work primarily addresses the design and implementation of a neural network based controller for the trajectory tracking of a differential drive mobile robot. The proposed control algorithm is an NN-based adaptive controller which tunes the gains of the back-stepping controller online according to the robot reference trajectory and its initial posture. In this method, a neural network is needed to learn the characteristics of the plant dynamics and make use of it to determine the future inputs that will minimize error performance index so as to compensate the back-stepping controller gains. The advantages and disadvantages of theproposed control algorithms will be discussed in each section with illustrations.Comprehensive system modeling including robot kinematics and dynamics modeling has been done. The dynamic modeling is done using Newtonian and Lagrangian methodologies for nonholonomic systems and the results are compared to verify the accuracy of each method. Simulation of the robot model and different controllers has been done using Matlab and Matlab Simulink.

***Keywords*:** Back-stepping controller, Dynamic Model

I. INTRODUCTION

In recent years a plethora of research has been carried out on the control problem of the autonomous mobile robotic systems. This is mainly due to the growing application of these systems both in industrial and service environments. Some typical applications of such systems are for instance order-pick robots in automated warehouses, post-delivery robots in office buildings and deep see exploration robots. Different kinds of robots can be used in these applications [1]. In rough terrains, walking robots are usually preferred. On smoother surfaces, wheeled mobile robots have the advantage because they are much faster and more agile. Other kinds of systems such as unmanned aerial vehicles or unmanned under water vehicles are used when we need to maneuver in three dimensional spaces[2,6,7]. Wheeled mobile robots are the type of mobile robotic systems that we are considering in this thesis because they are most widely used among the class of mobile robots. This is due to their fast maneuvering, simple controllers and energy saving characteristics [2,3,4,5].The work includes the development and construction of neural network based trajectory tracking controllers. The control algorithm is a NN-based adaptive controller which tunes the gains of the back-stepping controller online according to the robot reference trajectory and its initial posture. In this method, a neural network is needed to learn the characteristics of the plant dynamics and make use of it to determine the future inputs that will minimize error performance index so as to compensate the back-stepping controller gains. The controller will be applied to the trajectory tracking problem and the results of its implementation on a mobile robot platform will be presented. This thesis in general contributes to the growing field of mobile robot systems. Specific contributions of this thesis are mainly the design and implementation of novel trajectory tracking controllers for nonholonomic mobile robots. Listed below are the detailed contributions:

Designing a novel NN-based adaptive back-stepping controller using the neural network direct model approximation of the robot which highly improves the traditional back-stepping controller tracking performance. Designing a novel neural network computed torquecontroller using the neural network inverse model approximation of the robot which improves the tracking performance of the back-stepping controller in presence of bounded disturbance and uncertainty in the model.

Detailed comparison analysis between two traditional tracking controllers and the proposed controller and pointing out the advantages and disadvantages of each one of them.

II. FORWARD KINEMATIC MODEL OF MOBILE ROBOT

The goal of the robot kinematic modeling is to find the robot speed in the inertial frame as a function of the wheels speeds and the geometric parameters of the robot (configuration coordinates). In other words we want to establish the robot speed$\dot{q} = [\dot{x}\dot{y}\dot{\theta}]^T$ as a function of the wheel speeds $\dot{\varphi}_R$ and $\dot{\varphi}_L$ and the robot geometric parameters or we want to find the relationship between control parameters ($\dot{\varphi}_R$ and $\dot{\varphi}_L$) and the behavior of the system in the state space. The robot kinematics generally has two main analyses, one Forward kinematics and one Inverse kinematics:[5,6,8,9]
Forward kinematics:

$$\dot{q} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = f(\dot{\varphi}_R, \dot{\varphi}_L, gemometric\ parameters) \quad (1)$$

Inverse kinematics:

$$\begin{bmatrix}\dot{\varphi}_R\\ \dot{\varphi}_L\end{bmatrix} = f(\dot{x}, \dot{y}, \dot{\theta}) \quad (2)$$

Assume a differential drive mobile robot setup which has two wheels with the radius of R_a placed with a distance L from the robot center as shown in Fig. 1:The following notations will be used in this work:A: The intersection of the axis of symmetry with the driving wheels axis. C: The center of mass of the platform. a: The distance between the center of mass and driving wheels axis in x-direction. L: The distance between each driving wheel and the robot axis of symmetry in y-direction. R_a: The radius of each driving wheel. $\dot{\varphi}_R$: The rotational velocity of the right wheel. $\dot{\varphi}_L$: The rotational velocity of the left wheel. v: The translational velocity of the platform in the local frame. ω: The rotational velocity of the platform in the local and global frames.
The forward kinematic problem can be described as the problem of finding the following function:

$$\dot{q} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = f(\dot{\varphi}_R, \dot{\varphi}_L, L, R_a, \theta) \quad (3)$$

The speed of each wheel in the robot frame is $R_a\dot{\varphi}$, therefore the translational speed in the robot frame is the average velocity:

$$v = R_a \frac{\dot{\varphi}_R + \dot{\varphi}_L}{2} \quad (4)$$

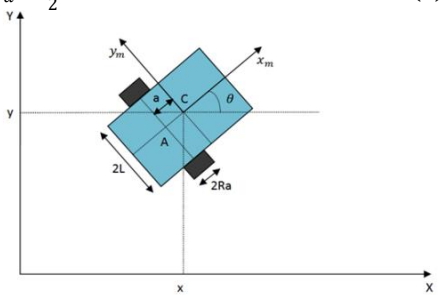

Figure 1: The differential drive mobile robot model

And the rotational velocity is:

$$\omega = \frac{R_a}{2L}(\dot{\varphi}_R - \dot{\varphi}_L) \quad (5)$$

The robot position in the inertial and robot frame can be defined as follows:

$$q_I = [x_I \quad y_I \quad \theta_I]^T \quad (6)$$

$$q_R = [x_R \quad y_R \quad \theta_R]^T \quad (7)$$

The mapping between these two frames is through the standard orthogonal rotation transformation:

$$\dot{q}_R = R(\theta)\dot{q}_I \quad (8)$$

$$R(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0\\ -sin\theta & cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} \quad (9)$$

Therefore the robot velocity in the global or inertial frame is:

$$\dot{q}_I = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\\ \sin(\theta) & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix} \frac{R_a}{2} \begin{bmatrix}\dot{\varphi}_R + \dot{\varphi}_L\\ 0\\ \frac{\dot{\varphi}_R - \dot{\varphi}_L}{L}\end{bmatrix} = \begin{bmatrix} R_a \frac{\dot{\varphi}_R + \dot{\varphi}_L}{2}\cos(\theta)\\ R_a \frac{\dot{\varphi}_R + \dot{\varphi}_L}{2}\sin(\theta)\\ \frac{R_a}{2L}(\dot{\varphi}_R - \dot{\varphi}_L)\end{bmatrix} \quad (10)$$

The above equation is the general forward kinematic equation for a differential drive mobile robot.

III. THE NN-BASED ADAPTIVE BACK-STEPPING CONTROLLER

The kinematic based controller or the so called back-stepping controller proposed by Kanayama in 1992 is a stable tracking control rule for a nonholonomicmobile robot and is explained thoroughly in section forward kinematic model of mobile robot. This controller's structure is shown in Fig. 2:[5,12,13,15]

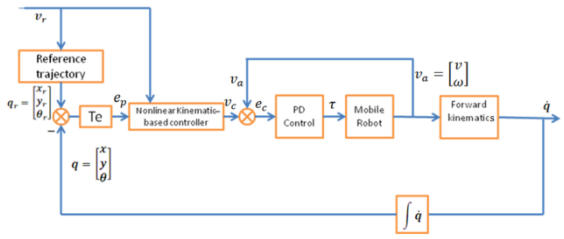

Figure 2: The back-stepping controller structure

The input error to this controller is defined as follows:

$$e_p = \begin{bmatrix}e_x\\ e_y\\ e_\theta\end{bmatrix} = T_e(q_r - q) = T_e \times e_r \quad (11)$$

$$\begin{bmatrix}e_x\\ e_y\\ e_\theta\end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta & 0\\ -sin\theta & cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} \begin{bmatrix}x_r - x\\ y_r - y\\ \theta_r - \theta\end{bmatrix} \quad (12)$$

The control algorithm that calculates the robot velocity input vector v_c is shown in the following equation:

$$v_c = \begin{bmatrix} v_r cose_\theta + K_x e_x\\ \omega_r + K_y v_r e_y + K_\theta v_r sine_\theta\end{bmatrix} \quad (13)$$

$$v_c = f(e_p, v_r, K) \quad (14)$$

$$K = (K_x, K_y, K_\theta) \quad (15)$$

The additional PD controller block is to make sure that the robot velocities follow the input reference velocities and has the following equation:

$$\tau = K_P e_c + K_D \frac{de_c}{dt} \quad (16)$$

In which:

$$K_P = \begin{bmatrix} K_{P_R} & 0 \\ 0 & K_{P_L} \end{bmatrix} \quad (17)$$

As it can be seen in the above equation, the controller has three gains which are mentioned to be positive constant values in the other references. The disadvantage of the above controller with constant gains is that it needs a careful gain tuning for each different reference trajectory and it will not give zero trajectory error with a smooth tracking. The proposed control algorithm provides the back-stepping controller with adaptive gains which are variable and change according to the reference trajectory[10,11,18,16,20,21]. The NN-based adaptive back-stepping controller structure is shown in Fig. 3:

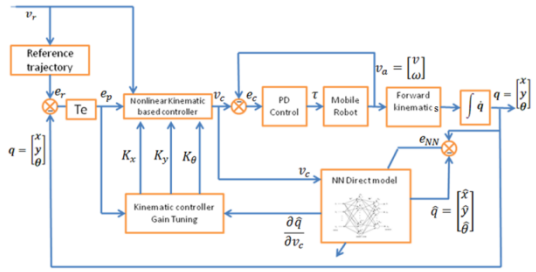

Figure 3: The NN-based adaptive back-stepping controller structure

The above control structure adapts the kinematic based controller gains to minimize the following cost function:

$$J = \frac{1}{2}\sum \gamma_x {e_x}^2 + \gamma_y {e_y}^2 + \gamma_\theta {e_\theta}^2 \quad (18)$$

The kinematic model based controller gains are considered part of the above cost function and are optimized and updated according to the gradient descent method [12,16,17]. The kinematic controller gains are represented by the set $\alpha = [K_x \quad K_y \quad K_\theta]$. The partial derivative of the cost function with respect to α is:

$$\frac{\partial J}{\partial \alpha} = \gamma_x e_x \frac{\partial e_x}{\partial \alpha} + \gamma_y e_y \frac{\partial e_y}{\partial \alpha} + \gamma_\theta e_\theta \frac{\partial e_\theta}{\partial \alpha} = \gamma {e_p}^T \frac{\partial e_p}{\partial \alpha} \quad (19)$$

$$\gamma = \begin{bmatrix} \gamma_x & 0 & 0 \\ 0 & \gamma_y & 0 \\ 0 & 0 & \gamma_\theta \end{bmatrix} \quad (20)$$

Substituting equations (11, 12) in the above equation, we have:

$$e_p = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = T_e(q_r - q) \quad (21)$$

$$\frac{\partial J}{\partial \alpha} = \gamma {e_p}^T \frac{\partial (T_e(q_r - q))}{\partial \alpha} = -\gamma \times {e_p}^T \times T_e \times \frac{\partial q}{\partial \alpha} \quad (22)$$

Using the chain rule, the derivative $\frac{\partial q}{\partial \alpha}$ can be written as:

$$\frac{\partial q}{\partial \alpha} = \frac{\partial q}{\partial v_c} \times \frac{\partial v_c}{\partial \alpha} \quad (23)$$

The first derivative in the above equation $\frac{\partial q}{\partial v_c}$ can be defined as the Jacobian matrix with the system velocity inputs, as follows:

$$\frac{\partial q}{\partial v_c} = \mathrm{Jac}_v = \begin{bmatrix} \frac{\partial x}{\partial v_c} & \frac{\partial x}{\partial \omega_c} \\ \frac{\partial y}{\partial v_c} & \frac{\partial y}{\partial \omega_c} \\ \frac{\partial \theta}{\partial v_c} & \frac{\partial \theta}{\partial \omega_c} \end{bmatrix} = \begin{bmatrix} \mathrm{Jac}_{v11} & \mathrm{Jac}_{v12} \\ \mathrm{Jac}_{v21} & \mathrm{Jac}_{v22} \\ \mathrm{Jac}_{v31} & \mathrm{Jac}_{v32} \end{bmatrix} \quad (24)$$

The second derivative in equation (23) $\frac{\partial v_c}{\partial \alpha}$ is calculated according to equation (13,14,15) as follows:

$$\frac{\partial v_c}{\partial \alpha} = \begin{bmatrix} \frac{\partial v_c}{\partial K_x} & \frac{\partial v_c}{\partial K_y} & \frac{\partial v_c}{\partial K_\theta} \\ \frac{\partial \omega_c}{\partial K_x} & \frac{\partial \omega_c}{\partial K_y} & \frac{\partial \omega_c}{\partial K_\theta} \end{bmatrix} = \begin{bmatrix} e_x & 0 & 0 \\ 0 & v_r e_y & v_r \sin e_\theta \end{bmatrix} \quad (25)$$

Therefore, the required derivative in equation (23) can be found by substituting equations (24) and (25) in (23):

$$\frac{\partial q}{\partial \alpha} = \mathrm{Jac}_v \times \begin{bmatrix} e_x & 0 & 0 \\ 0 & v_r e_y & v_r \sin e_\theta \end{bmatrix} \quad (26)$$

The derivative of the cost function with respect to the controller gains $\frac{\partial J}{\partial \alpha}$ is:

$$\frac{\partial J}{\partial \alpha} = \begin{bmatrix} \frac{\partial J}{\partial K_x} & \frac{\partial J}{\partial K_y} & \frac{\partial J}{\partial K_\theta} \end{bmatrix} = -\gamma \times {e_p}^T \times T_e \times \mathrm{Jac}_v \times \begin{bmatrix} e_x & 0 & 0 \\ 0 & v_r e_y & v_r \sin e_\theta \end{bmatrix} \quad (27)$$

In which:

$$\gamma = \begin{bmatrix} \gamma_x & 0 & 0 \\ 0 & \gamma_y & 0 \\ 0 & 0 & \gamma_\theta \end{bmatrix} \quad (28)$$

$${e_p}^T = [e_x \quad e_y \quad e_\theta] \quad (29)$$

$$T_e = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Jac}_v = \begin{bmatrix} \frac{\partial x}{\partial v_c} & \frac{\partial x}{\partial \omega_c} \\ \frac{\partial y}{\partial v_c} & \frac{\partial y}{\partial \omega_c} \\ \frac{\partial \theta}{\partial v_c} & \frac{\partial \theta}{\partial \omega_c} \end{bmatrix} = \begin{bmatrix} \mathrm{Jac}_{v11} & \mathrm{Jac}_{v12} \\ \mathrm{Jac}_{v21} & \mathrm{Jac}_{v22} \\ \mathrm{Jac}_{v31} & \mathrm{Jac}_{v32} \end{bmatrix} \quad (30)$$

Therefore, the kinematic controller gains will change and adapt to make the cost function zero according to the gradient descent method as follows:

$$K_x = K_x + \Delta K_x \quad (31)$$

$$K_y = K_y + \Delta K_y \quad (32)$$

$$K_\theta = K_\theta + \Delta K_\theta \quad (33)$$

In which the change in the gains is computed according to the following equations:

$$\Delta K_x = -\eta_{K_x} \frac{\partial J}{\partial K_x} \quad (34)$$

$$\Delta K_y = -\eta_{K_y} \frac{\partial J}{\partial K_y} \quad (35)$$

$$\Delta K_\theta = -\eta_{K_\theta} \frac{\partial J}{\partial K_\theta} \quad (36)$$

The parameters η_{K_x} , η_{K_y} and η_{K_θ} are the learning rates of the gradient descent algorithm. The important part of calculating the derivatives of the equation (26) is how to compute the Jacobian matrix of the system. The Jacobian matrix can be calculated using the exact equations of the system or it can be provided by the neural network direct model. Calculating the Jacobian matrix based on the above two methods and their advantages and restrictions will be explained in the following two sections.

IV. JACOBIAN CALCULATION USING THE SYSTEM'S EXACT EQUATIONS

The Jacobian matrix can be calculated using the robots governing kinematic equations. The derivative $\frac{\partial q}{\partial v_c}$ can be expanded as follows:[4,12,13]

$$Jac_v = \frac{\partial q}{\partial v_c} = \frac{\partial q}{\partial v_a} \times \frac{\partial v_a}{\partial \tau} \times \frac{\partial \tau}{\partial e_c} \times \frac{\partial e_c}{\partial v_c} \quad (37)$$

To calculate the first derivative in the above equation, we use the robots kinematic equation as follows:

$$\dot{q} = Sv_a = \begin{bmatrix} \cos\theta & -a\sin\theta \\ \sin\theta & a\cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (38)$$

Integrating the above equation over time, we have:

$$q = \int \dot{q} dt = \int Sv_a dt \quad (39)$$

The required Jacobian matrix can be found

$$Jac_v = \int S dt \times Jac_\tau \times \begin{bmatrix} K_{P_R} & 0 \\ 0 & K_{P_L} \end{bmatrix} \quad (40)$$

Calculation of the Jacobian matrix using the neural network direct model approach solves this problem because the neural network will learn the real robot's model online and gives us a better approximation of the system.

V. JACOBIAN CALCULATION USING THE NEURAL NETWORK DIRECT MODEL

The neural network which is used to approximate the robot platform or so called the direct model neural network is shown in Fig. 4:[12,16,17]

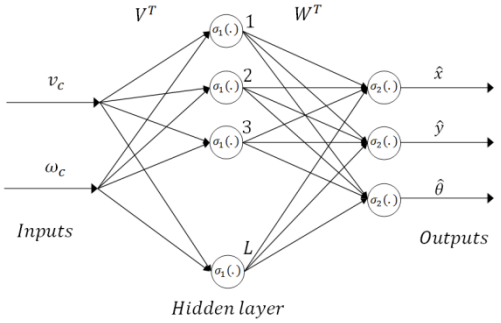

Figure 4: The direct model neural network

The Jacobian matrix that can be found from the above neural network is:

$$\frac{\partial \hat{q}}{\partial v_c} = Jac_v = \begin{bmatrix} \frac{\partial \hat{x}}{\partial v_c} & \frac{\partial \hat{x}}{\partial \omega_c} \\ \frac{\partial \hat{y}}{\partial v_c} & \frac{\partial \hat{y}}{\partial \omega_c} \\ \frac{\partial \hat{\theta}}{\partial v_c} & \frac{\partial \hat{\theta}}{\partial \omega_c} \end{bmatrix} = \begin{bmatrix} Jac_{v11} & Jac_{v12} \\ Jac_{v21} & Jac_{v22} \\ Jac_{v31} & Jac_{v32} \end{bmatrix} (41)$$

As it is mentioned in the neural network introduction section, the equation relating the inputs and outputs of this network is:

$$y_i = \sigma\left(\sum_{l=1}^{L} W_{il}\, \sigma\left(\sum_{j=1}^{n} V_{lj} x_j + v_{l0}\right) + w_{i0}\right) \quad i = 1,2,\dots,m \quad (42)$$

The input to layer one is x_j. Define the input to layer two as:

$$z_l = \sigma\left(\sum_{j=1}^{n} V_{lj} x_j + v_{l0}\right) \quad l = 1,2,\dots,L \quad (43)$$

The thresholds can more easily be dealt with by defining $x_0 = 1$ and $z_0 = 1$. Then one can say:

$$y_i = \sigma\left(\sum_{l=1}^{L} W_{il}\, z_l\right) \quad (44)$$

$$z_l = \sigma\left(\sum_{j=1}^{n} V_{lj} x_j\right) \quad (45)$$

Using the chain rule, the required derivative for the Jacobian matrix is

$$\frac{\partial y_i}{\partial x_j} = \frac{\partial y_i}{\partial z_l} \times \frac{\partial z_l}{\partial x_j} \quad (46)$$

Therefore, the derivative of the equation (46) is:

$$\frac{\partial y_i}{\partial x_j} = W_{il} \times \acute{o}\left(\sum_{l=1}^{L} W_{il}\, z_l\right) \times V_{lj} \times \acute{o}\left(\sum_{j=1}^{n} V_{lj} x_j\right) (47)$$

Equation (4.204) is the equation that can be used to compute the Jacobian matrix using the direct model neural network. The neural network to be used in this approach should be well trained so it can perform a precise approximation of the system. The details of the neural network training and implementation will be explained in the simulation and results section. The simulation results of the above control algorithm are shown in detail in the next section.

VI. SIMULATION AND RESULTS

Comparison analysis between back-stepping controller and the NN-based adaptive back-stepping controller is done to show the advantages of the proposed controller. The following features of the controllers should be discussed to have a perfect comparison:The trajectory tracking error
Smoothness of the robot trajectories

Trajectory reach timeMagnitude of the control action Comparison between the performances of the controllers in the above categories results in a comprehensive analysis and shows the advantages and restrictions of each controller. The simulation results of the back-stepping controller with fixed controller gains and adaptive gain tuning controller in response to different reference trajectories and robot initial positions will be shown in the remaining of this section and conclusion will be made about the performance of each controller.

Linear reference trajectory
Robot initial location: (0, 1, 0)
Back-steppingcontrollergains
$K_x = 1, K_y = 55, K_{th} = 15$
Gains of the adaptive controller cost function:

$$J = \frac{1}{2}\sum \gamma_x {e_x}^2 + \gamma_y {e_y}^2 + \gamma_\theta {e_\theta}^2$$
$$\gamma_x = 1, \gamma_y = 50, \gamma_\theta = 1$$

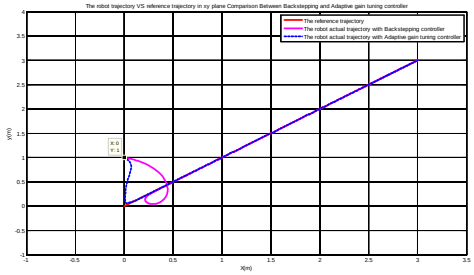

Figure 5: The robot trajectory in x-y plane, Comparison between the Back-stepping controller and adaptive gain tuning controller (Linear reference trajectory)

The time response of the robot output states $x(t), y(t)$ and $\theta(t)$, the trajectory errors $e_x(t), e_y(t)$ and $e_\theta(t)$, the robot velocities $v(t)$ and $\omega(t)$, the robot velocity errors $e_v(t)$ and $e_\omega(t)$, the controller output torques to the right and left robot wheels $T_r(t)$ and $T_l(t)$ and the adaptive gain tuning controller gains$K_x(t), K_y(t)$ and $K_\theta(t)$ are shown in the following figures

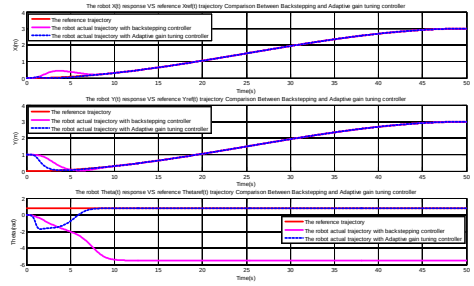

Figure 6: The robot output states time response vs. reference trajectories, Comparison between the Back-stepping controller and adaptive gain tuning controller (Linear reference trajectory)

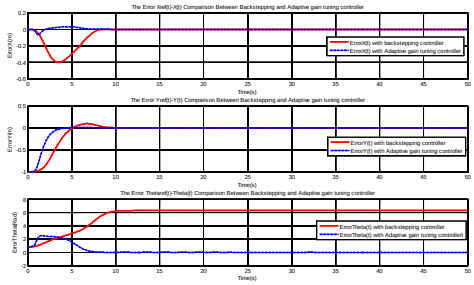

Figure 7: The robot output states errors Ex, Ey and Eth, Comparison between the Back-stepping controller and adaptive gain tuning controller (Linear reference trajectory)

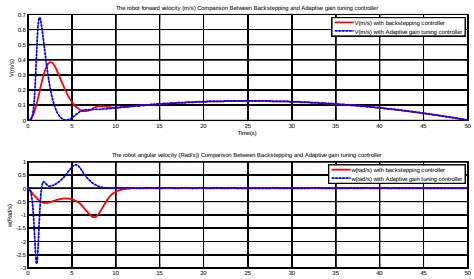

Figure 8: The robot linear and angular velocities, Comparison between the Back-stepping controller and adaptive gain tuning controller (Linear reference trajectory)

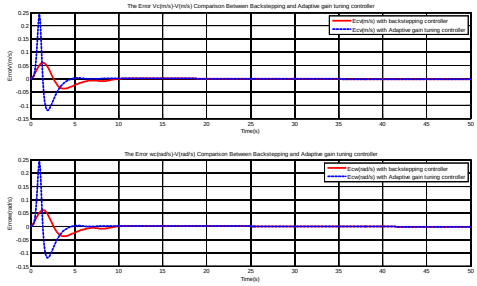

Figure 9: The robot velocity errors, Comparison between the Back-stepping controller and adaptive gain tuning controller (Linear reference trajectory)

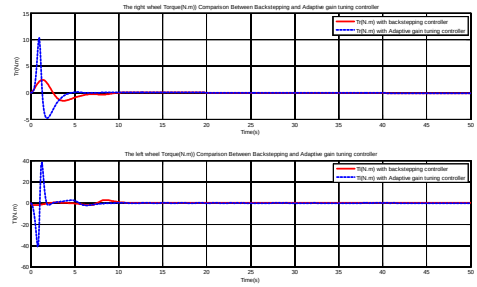

Figure 10: The rights and left wheel torques, Comparison between the Back-stepping controller and adaptive gain tuning controller (Linear reference trajectory)

The controller gain K_x is made to be fixed because it will increase the controller output torque and make the robot unstable. The change in the other controller gains $K_y(t)$ and $K_\theta(t)$ will adapt the robot to each different reference trajectory and force the trajectory error to zero. As it can be seen from the above figure, the controller gains will become constant when the trajectory error becomes zero. This shows the convergence of the gradient decent learning algorithm which drives the cost function to zero.

Sinusoidal reference trajectory
Robot initial location: (0, 0, 0)
Back-stepping controller gains:

$$K_x = 1, K_y = 55, K_{th} = 15$$

Gains of the adaptive controller cost function:

$$J = \frac{1}{2}\sum \gamma_x {e_x}^2 + \gamma_y {e_y}^2 + \gamma_\theta {e_\theta}^2$$

$$\gamma_x = 1, \gamma_y = 50, \gamma_\theta = 1$$

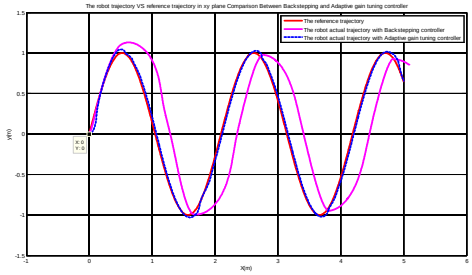

Figure 11: The robot trajectory in x-y plane, Comparison between the Back-stepping controller and adaptive gain tuning controller (Sinusoidal reference trajectory)

The sinusoidal reference trajectory is considered a challenging reference trajectory for a nonholonomic mobile robot to track because of its peaks which involve a big change in the heading angle. The perfect tracking performance of the adaptive gain tuning controller in comparison with the back-stepping controller can be seen in the above figure.

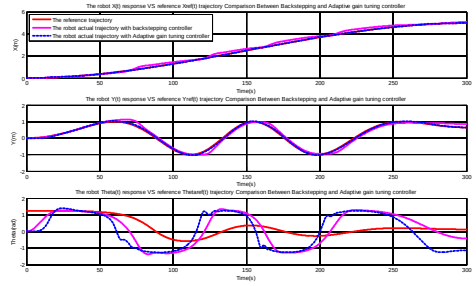

Figure 12: The robot output states time response vs. reference trajectories, Comparison between the Back-stepping controller and adaptive gain tuning controller (Sinusoidal reference trajectory)

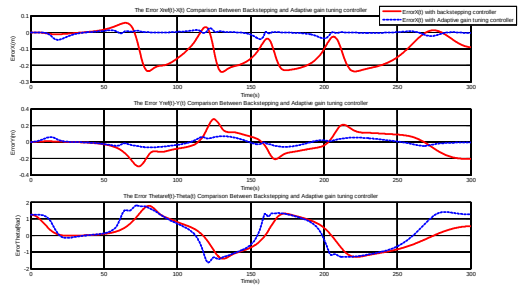

Figure 13: The robot output states errors Ex,Ey and Eth, Comparison between the Back-stepping controller and adaptive gain tuning controller (Sinusoidal reference trajectory)

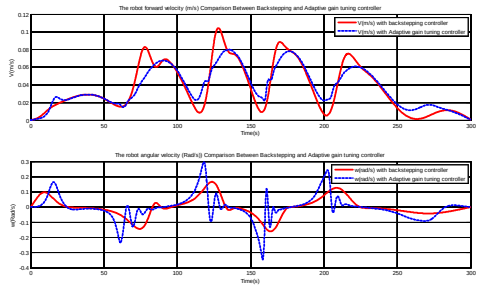

Figure 14: The robot linear and angular velocities, Comparison between the Back-stepping controller and adaptive gain tuning controller (Sinusoidal reference trajectory)

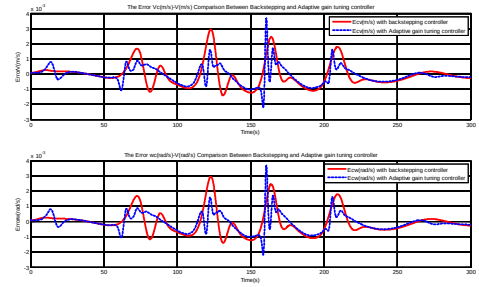

Figure 15: The robot velocity errors, Comparison between the Back-stepping controller and adaptive gain tuning controller (Sinusoidal reference trajectory)

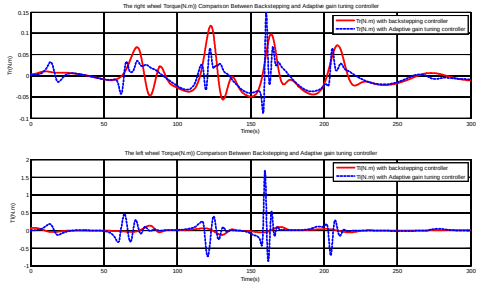

Figure 16: The rights and left wheel torques, Comparison between the Backstepping controller and adaptive gain tuning controller (Sinusoidal reference trajectory)

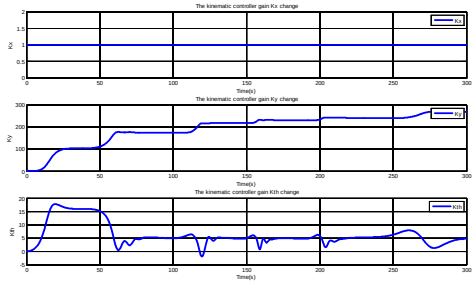

Figure 17: The adaptive gain tuning controller gains time response (Sinusoidal reference trajectory)

The square reference trajectory is one of the famous trajectories which have been tested in literature for different trajectory tracking control algorithms.
Rectangular reference trajectory
Robot initial location: (0, 1, 0)
Back-stepping controller gains:

$$K_x = 1, K_y = 65, K_{th} = 15$$

Gains of the adaptive controller cost function:

$$J = \frac{1}{2}\sum \gamma_x e_x{}^2 + \gamma_y e_y{}^2 + \gamma_\theta e_\theta{}^2$$

$$\gamma_x = 1, \gamma_y = 50, \gamma_\theta = 1$$

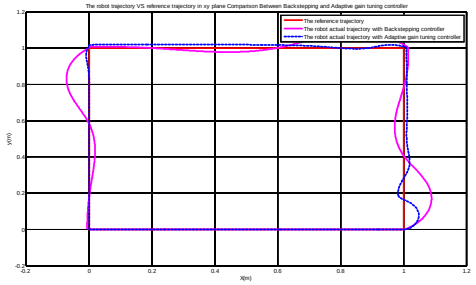

Figure 18: The robot trajectory in x-y plane, Comparison between the Back-stepping controller and adaptive gain tuning controller (Rectangular reference trajectory)

The perfect tracking performance of the adaptive gain tuning controller in comparison to the back-stepping controller is shown in the above figure. Note that the back-stepping controller with the fixed gains can be tuned for each of the reference trajectories but it a very time consuming process. Making the controller gains adaptive, solves this problem and makes the robot track any trajectory regardless of its initial position.

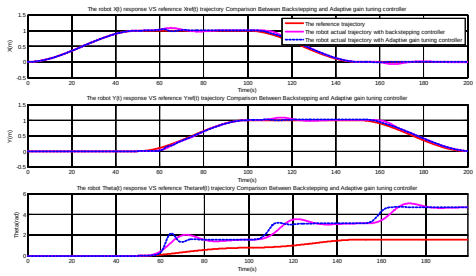

Figure 19: The robot output states time response vs. reference trajectories, Comparison between the Back-stepping controller and adaptive gain tuning controller (Rectangular reference trajectory)

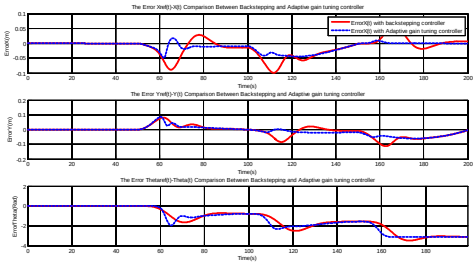

Figure 20: The robot output states errors Ex,Ey and Eth, Comparison between the Back-stepping controller and adaptive gain tuning controller (Rectangular reference trajectory)

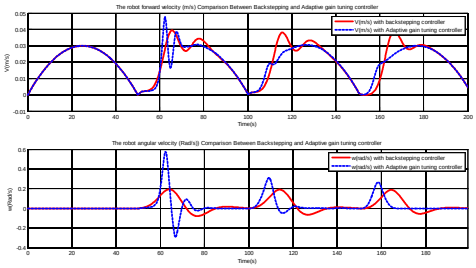

Figure 21: The robot linear and angular velocities, Comparison between the Back-stepping controller and adaptive gain tuning controller (Rectangular reference trajectory)

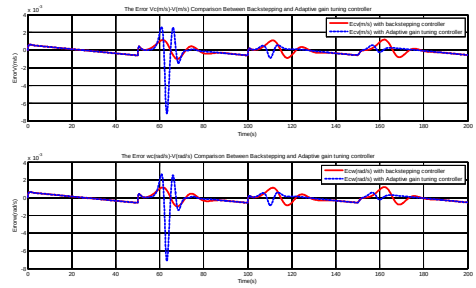

Figure 22: The robot velocity errors, Comparison between the Back-stepping controller and adaptive gain tuning controller (Rectangular reference trajectory)

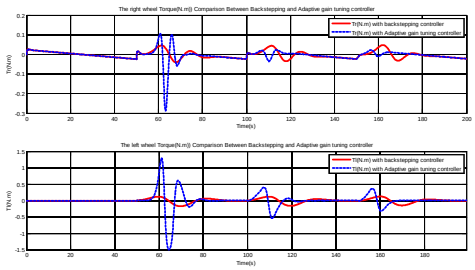

Figure 23: The rights and left wheel torques, Comparison between the Back-stepping controller and adaptive gain tuning controller (Rectangular reference trajectory)

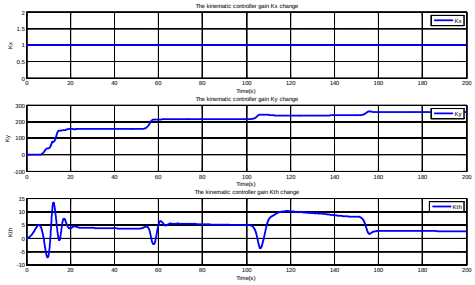

Figure 24: The adaptive gain tuning controller gains time response (Rectangular reference trajectory)

The edges of the each trajectory, which are the places that the robot heading angle has a big change are the most challenging parts of the trajectories. The change in the controller gains will happen mostly at these edge points as can be seen from the above figure.

Circular reference trajectory
Robot initial location: (0, 0, 0)
Back-stepping controller gains:

$$K_x = 1, K_y = 55, K_{th} = 15$$

Gains of the adaptive controller cost function:

$$J = \frac{1}{2}\sum \gamma_x {e_x}^2 + \gamma_y {e_y}^2 + \gamma_\theta {e_\theta}^2$$

$$\gamma_x = 1, \gamma_y = 50, \gamma_\theta = 1$$

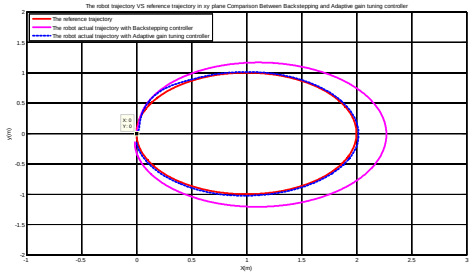

Figure 25: The robot trajectory in x-y plane, Comparison between the Back-stepping controller and adaptive gain tuning controller (Circular reference trajectory)

The circular reference trajectory is another famous reference trajectories tested by different researched in this field. The great performance of the proposed adaptive gain tuning controller in comparison with the back-stepping controller can be seen from the above figure.

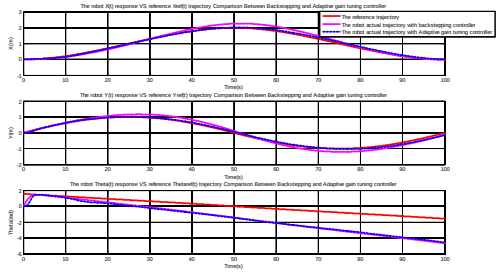

Figure 26: The robot output states time response vs. reference trajectories, Comparison between the Back-stepping controller and adaptive gain tuning controller (Circular reference trajectory)

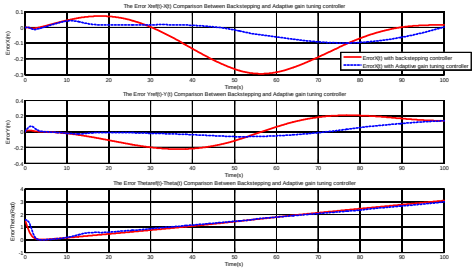

Figure 27: The robot output states errors Ex,Ey and Eth, Comparison between the Back-stepping controller and adaptive gain tuning controller (Circular reference trajectory)

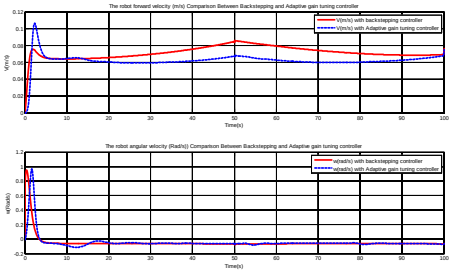

Figure 28: The robot linear and angular velocities, Comparison between the Back-stepping controller and adaptive gain tuning controller (Circular reference trajectory)

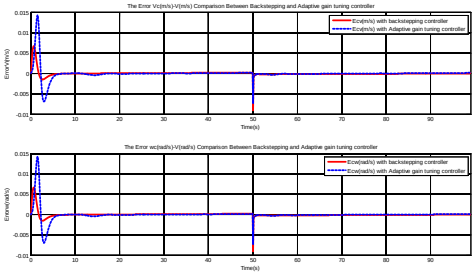

Figure 29: The robot velocity errors, Comparison between the Back-stepping controller and adaptive gain tuning controller (Circular reference trajectory)

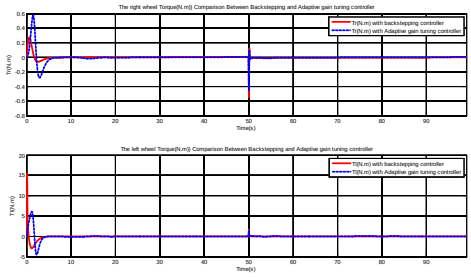

Figure 30: The rights and left wheel torques, Comparison between the Back-stepping controller and adaptive gain tuning controller (Circular reference trajectory)

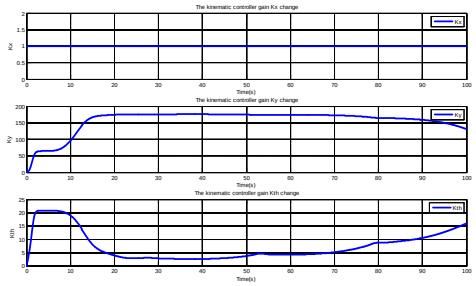

Figure 31: The adaptive gain tuning controller gains time response (Circular reference trajectory)

VII. CONCLUSION

The above comprehensive comparison analysis between the NN-based adaptive back-stepping controller and the back-stepping controller shows that the NN-based adaptive back-stepping controller is a perfect trajectory tracking controller according to the following advantages that it has over the back-stepping controller:

1. No knowledge of the system dynamics, robot initial position and the reference trajectory is needed before using this controller. The adaptive feature of this algorithm deals with the changes in the robot dynamics, reference trajectories and other uncertainties.
2. The zero tracking error will be achieved regardless of the shape of the reference trajectory and the robot initial position with respect to the reference trajectory.
3. A respectively small trajectory reach time can be achieved when there is a big distance between the robot and the reference trajectory.
4. A smooth robot trajectory will be produced using this controller which makes it easier to implement in real life applications.
5. The controller output torques are within the range of the robot actuators which makes this controller applicable to the experimental robot platform.

REFERENCES

[1] G. Artus, P. Morin, and C. Samson, "Tracking of an omnidirectional target with a nonholonomic mobile robot," *in Proc. IEEE Conf. Adv. Robot.(ICAR), 2003*, pp. 1468–1473.

[2] R. Siegwart and I. R. Nourbakhah, *Introduction to Autonomous Mobile Robots.* Cambridge, MA: MIT Press, 2004.

[3] A. Filipescu, V. Minzu, B. Dumitrascu and E. Minca, Trajectory Tracking and Discrete-Time Sliding-Mode Control of Wheeled Mobile Robot. *Proceedings of the IEEE International Conference on Information and Automation, Shenzhen, China*, 2011, pp. 27-32.

B. Krose, Environment Representations for Cognitive Robot Companions. *ERCM News*, No. 53, 2003, pp. 29-30.

[4] R. Barzamini and A. Afshar, "Dynamic adaptive tracking control for wheeled mobile robots," AmirKabir University of Technology, Tehran, Iran, 2006.

[5] R. W. Brockett, "*Asymptotic stability and feedback stabilization," in Differential Geometric Control Theory*, R. W. Brockett, R. S. Millman, and H. J. Sussmann, Eds. Boston, MA: Birkauser, 1983.

[6] J. M. Yang and J. H. Kim, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots *," in IEEE Transactions on Robotics and Automation* , vol. 15, pp. 578-587, June 1999.

[7] Y. Kanayama et al., "A stable tracking control method for an autonomous mobile robot," in *Proc. IEEE Conf. Robot. Autom.* 1990, pp. 384–389.

[8] G. Campion, G. Bastin, and B. d'Andrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots*," in IEEE Trans. Robot. Automat*., vol. 12, pp. 47–62, Jan. 1996.

[9] G. Campion and G. Bastin, "*On adaptive linearizing control of omni-directional mobile robots,*" MTNS, pp. 531-538, Amsterdam, Holland, 1989.

[10] G. Klancar and I. Skrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *in Robotics and autonomous systems* , vol. 25, pp. 460-469, Sep. 2007.

[11] J. Ye, "Tracking control for nonholonomic mobile robots: Integrating the analog neural network into the backstepping technique," in Neurocomputing, vol. 71, pp. 3373-3378, Dec. 2007.

[12] J. Velagic, N. Osmic and B. Lacevic, " Neural Network controller for mobile robot motion control," in International journal of intelligent systems and technologies, vol. 21, pp. 470-479, Oct. 2008.

[13] R. Fierro and F. L. Lewis, "Control of a Nonholonomic Mobile Robot Using Neural Networks," in IEEE Transactions on Neural Networks, vol. 9, pp. 589-600, July 1998.

[14] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot: backstepping kinematics into dynamics," in Proc. Of 34th IEEE Conf. on Decision and Control, Piscataway, NJ, IEEE press, USA, 1995, pp. 3805-3810.

[15] A. Bloch and S. Drakunov, " Stabilization of a nonholonomic system via sliding modes," in *Proc. Of IEEE international conference on Decision Control, Madrid, Spain*, 1994, pp. 2961-2963.

[16] K. N. Faress, M. T. El hargy and A. A. El kosy, " Trajectory tracking control for a wheeled mobile robot using fuzzy logic controller," in *Proceedings of the 9th WSEAS International Conference on Systems, Athens, Greece,*2005, pp. 1261-1269.

[17] P. Morin and C. Samson, "Control of Nonholonomic Mobile Robots Based on the Transverse Function Approach*," in IEEE Trans. on Robotics*, vol. 25, pp. 1058-1073, Oct. 2009.

[18] D. Buccieri, D. Perritaz, P. Mullhaupt, Z. P. Jiang and D. Bonvin, "*Velocity-Scheduling Control for a Unicycle Mobile Robot*:

[19] L. Ge, R. Li, D. Yu and L. Zhao, " A Nonlinear Adaptive Variable Structure Trajectory Tracking Algorithm for Mobile Robots," *in Intelligent robotics and applications*, vol. 15, pp. 892-900, Nov. 2008.

[20] G. Oriolo, A. D. Luca and M. Vendittelli, "WMR Control via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation," *in IEEE Trans. on Control Systems Thechnology* , vol. 10, pp. 835-852, Nov. 2002.

[21] F. Dong and W. L. Xu, "Adaptive Tracking Control of Uncertain Nonholonomic Dynamic System," *in IEEE Transactions on Automatic Control*, vol. 46, pp. 450-454, March 2001.