Active Disturbance Rejection Controller for Tracking Control of a Differential Drive Robot

—
Department of Computer Science
University of Sharjah
P.O.Box. 27272 Sharjah, UAE

——
Department of Computer Science
University of Sharjah
P.O.Box. 27272 Sharjah, UAE

——
Department of Computer Science
University of Sharjah
P.O.Box. 27272 Sharjah

***Abstract*—In recent years, application of robotics has greatly advanced many fields, provide robust performance and efficiency in difficult tasks without the need for human intervention. Robot control is one area of robotics that has received a lot of interest and technological development. Active Disturbance Rejection Controller (ADRC) controllers are widely adopted due to their robust performance in disturbance rejection and state estimation. The manual tuning process of controller parameters is a lot time time-consuming and prompting the need of automation. This paper presents an innovative approach to tune the parameters of ADRC using Reinforcement Learning (RL) based on Deep Deterministic Policy Gradient (DDPG) for a Differential Drive Mobile Robot (DDMR). The RL agent learn the ideal parameters of ADRC controller through iterative interactions with a simulated environment, improving ADRC's performance without the need of manual tuning. Simulation results presents the effectiveness of the proposed idea that improves the performance of the trajectory tracking, By combining RL and ADRC, a promising automated controller tuning solution is provided, opening the door to more intelligent and adaptive robotic systems.**

***Index Terms*—Artificial Intelligence (AI), Reinforcement Learning (RL), Active Disturbance Rejection Control (ADRC), Differential Drive Mobile Robot (DDMR), Deep Deterministic Policy Gradient (DDPG)**

I. Introduction

The addition of Artificial Intelligence (AI) into control systems has overcome tradition control techniques and can handle more complex and nonlinear environments. Comparing different AI approaches, the Reinforcement Learning (RL) proved its significant potential in improving the control strategy by interacting to the environment. RL is especially helpful in scenarios where the system dynamics are hard to accurately model or are subject to change, in contrast to classical control techniques that depend on exact mathematical models. RL allows data-driven learning, which is especially helpful in scenarios where the system dynamics are hard to accurately model or are subject to change, in contrast to classical control techniques that depend on exact mathematical models [1].

RL is a type of machine learning where an agent learns to make decisions by receiving feedback in the form of rewards or penalties [2]. In control systems, RL helps by autonomously tuning controller parameters, learning optimal control policies, and adapting to varying conditions without human intervention. This capability is particularly valuable for robots that need to operate in dynamic, uncertain, or complex environments. Particularly, Deep Reinforcement Learning without taking precise models of the system into account is transforming control systems especially by handling non-linear dynamics. The authors in [3] explained that RL model is based on interactions with the system, feedback and reward optimizing the control policy, which is applicable to the real world problem especially where system contains uncertainty.

Jardine et al. [4] proposed a technique of an adaptive predictive control for a differential drive robot tuned with RL. The approach is used to tune the predictive controller parameters which results in an improved robot performance in simulation. Also, real-world performance is measured, using model predictive control based on turtle-bot which reflects a better trajectory tracking and reduced error. The authors in [5] implemented Twin Delayed Deep Deterministic Policy Gradient (TD3) on a PyBullet Racecar model in the OpenAI Gym environment. They used a reward function that stabilizes the policy convergence for differential drive robot. The system is trained in such a way that it maintains a distance from walls and move towards the goal by avoiding collisions. In another work, the ability of RL-based control was to identify the best solution and retrieve the best control policy, as one of its qualities that makes it compatible with conventional control methods. The best course of action is followed by interacting with the environment, getting feedback in the form of rewards or penalties, and learning how to act in a way that maximizes cumulative rewards over time [6]. In [7], an optimized nonlinear tracking control method using an Actor-Critic Reinforcement Learning Strategy is proposed. This approach involves a neural network for both actor and critic to handle non-linear dynamic systems, where convergence is guaranteed using Lyapunov stability theory.

Jingqing Han [8] developed Active Disturbance Rejection Control (ADRC) in the 1990s, and it has since been investigated and applied in a variety of real-world fields, such as robotics. The ADRC is designed to dynamically reject internal and external disturbances to accurately control the system. Unlike classical PID controllers, ADRC adapts disturbances in real-time rather than relying on fixed gains and visualizing a model of the system. ADRC consist of three components: Tracking Differentiator (TD) to smooth the desired input signal, Extended State Observer (ESO) to estimate the system states and handle unknown disturbances and the Nonlinear State Error Feedback (NLSEF) which generate control signals

that stabilize the system [9]. PID was also used to control mobile robots [10], they doesn't require a mathematical model of the system and are sensitive to the environmental variations and it cannot handle uncertainties and disturbance like ADRC. but it is based on fixed parameters and struggle with changing dynamics, ADRC overcomes PID by not requiring a perfect mathematical model of the system and handle disturbances better [11].

Authors in [12] proposed a framework for mobile manipulator system and achieves separate trajectory tracking for both. The appropriate ADRC controller is designed, and the nonlinear Extended State Observer (ESO) disturbance is applied to estimate and compensate for disturbance. The ADRC approach's resilience in managing MIMO nonlinear systems with time-varying dynamics shown in [12] by the simulation results. In [13] [14], controller parameters has been scaled so that the tuning parameters can be reduced to only one, by developing ADRC block library in MATLAB/Simulink, enabling easy construction and customization of ADRC systems using graphical modeling.

The PID parameters and as well as ESO bandwidth require manual tuning. A system with complex dynamics and high non-linearity is hard to tune because its time consuming process, manually tuned parameters can never provide the optimal result especially when external disturbances are varying because such disturbances and non-linearity require dynamic adaption [15]. One promising way to get around this limitation is to use RL, where these parameters can be dynamically modified in response to real-time system feedback by utilizing RL network, this combined technique could improve control systems performance and robustness.

The contribution of this paper is to employ a Deep Deterministic Policy Gradient (DDPG) algorithm in conjunction with RL to design such a system using an ADRC controller. To guarantee accurate estimation and tracking, the RL agent automatically adjusts the ADRC controller's parameters and trains to converge to the ideal values. This research builds on this that the developed controller is able to provide better trajectory tracking and can be extended and applied in various application.

The remainder of the paper is organized as follows, section 2 provides system description and modeling, followed by section 3 that briefs the control design, section 4 presents results and discussion, section 5 concludes the paper.

II. System Modeling and Description

A differential drive robot is a widely used mobile robot configuration, especially in indoor applications and research, due to its simplicity and control efficiency. By altering the angular velocities of the two independently driven wheels on either side of the robot, it enables omnidirectional maneuverability. Rotating around its central axis and moving forward and backward are made possible by varying the speeds of the left and right wheels.

Figure 1 illustrates the geometric properties related to DDMR, (x_c, y_c) are the coordinates of the robot's center point

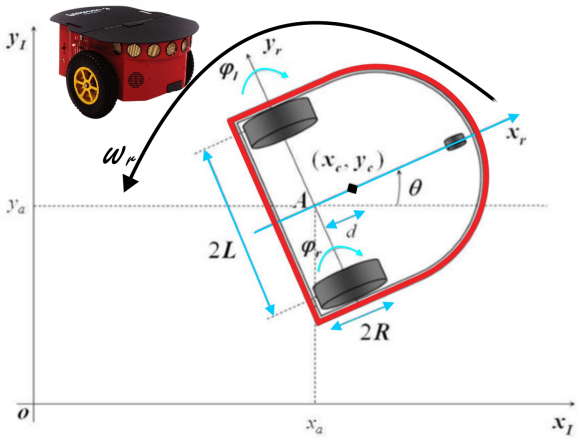

Fig. 1: DDMR Platform

in the reference frame, $2R$ is the distance between the center of the wheels and the point A on the arc path, $2L$ represents the distance between the two wheels, ϕ_r the rotation angle of the right wheel, ϕ_l is the rotation angle of the left wheel, d the distance between the center point (x_c, y_c) and the point A along the longitudinal axis, (x_r, y_r) are the coordinates of the target/reference point, (x_a, y_a) are the axes representing the global reference frame.

A. Kinematics Model

At first DDMR's forward kinematics for reference trajectory is to convert reference linear velocity (v_r) and angular velocity (w_r) into x and y axis (x_r, y_r) reference trajectory and orientation angle (θ_r) [7]. The equations used in forward kinematics for reference trajectory are shown below:

$$\dot{x}_r = v_r \cos(\theta_r) \tag{1}$$

$$\dot{y}_r = v_r \sin(\theta_r) \tag{2}$$

$$\dot{\theta}_r = w_r \tag{3}$$

After integrating equations 1, 2 and 3, (x_r, y_r, θ_r) is obtained. The forward kinematics for robot's actual trajectory, is to convert measured linear velocity (v_m) and angular velocity (w_m) obtained from dynamic model into x and y axis (x_m, y_m) real trajectory, while the orientation angle (θ_m) is extracted from the dynamic model of the robot [7]. The equations used in forward kinematics for actual trajectory are shown below:

$$\dot{x} = v_m \cos(\theta_m) \tag{4}$$

$$\dot{y} = v_m \sin(\theta_m) \tag{5}$$

After integrating equations 4 and 5, (x_m, y_m) is obtained. The θ_m in equation (4) and (5) is calculated in dynamics model by integrating (w_m). Figure 2 provides an overall architecture of the system.

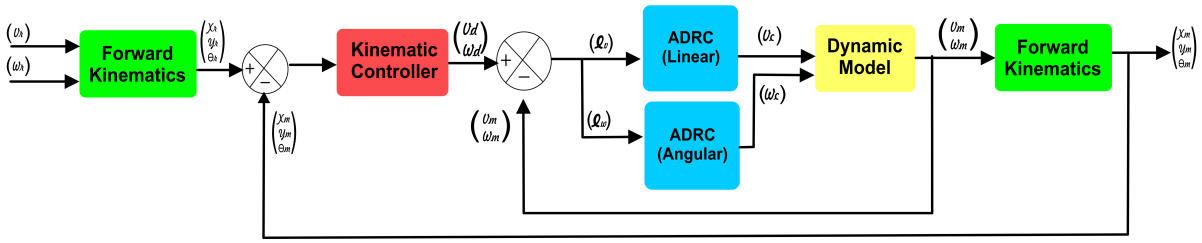

Fig. 2: ADRC-FPD block diagram.

B. Dynamics Model

The purpose of the dynamic model of the system is to show how the commanded linear (v_c) and angular (w_c) velocity affects the output of the robot. The input of the dynamic model is (v_c, w_c), and the output is the derivative of the linear velocity ($\dot{v_m}$) and angular velocity ($\dot{w_m}$) [8], which are passed to the integrator to obtain (v_m, w_m). The actual angular velocity (w_m) is further integrated to get actual orientation angle θ_m of the robot. In a simplified form, the dynamics model equations are as follows:

$$\dot{v}_m = \left(\frac{k_3}{k_1}\right) w_m^2 - \left(\frac{k_4}{k_1}\right) v_m + \frac{v_c}{k_1} \tag{6}$$

$$\dot{w}_m = -\left(\frac{k_5}{k_2}\right) (v_m w_m) - \left(\frac{k_6}{k_2}\right) w_m + \frac{w_c}{k_2} \tag{7}$$

The above dynamics model is specific for Pioneer (3-DX) mobile robot [8]. k_1, k_2, k_3, k_4, k_5 and k_6 are fixed constants as system parameters, which are defined as [8]:

$$k_1 = 0.24089, \quad k_2 = 0.2424, \quad k_3 = -0.00093603, \tag{8}$$

$$k_4 = 0.99629, \quad k_5 = -0.0057256, \quad k_6 = 1. \tag{9}$$

III. Control Design

Figure 2 demonstrates the block diagram of the system. Figure 3 explains the structure of ADRC-FPD block present in Figure 2. As the block diagram explains, reference trajectory (x_r, y_r, θ_r) and actual trajectory(x_m, y_m, θ_m) errors are fed to the kinematic controller which provides desired velocity commands (v_d, w_d). Further, velocity errors (e_v, e_w) are calculated and passed to FPD controller. The equations for errorS calculation are as follows:

$$e_v = v_d - v_m \tag{10}$$

$$e_w = w_d - w_m \tag{11}$$

There are two FPD controllers in the system, one for linear velocity control and the other for angular velocity control. which results in velocity commands (v_c, w_c) as an input to dynamics model, this model provides actual linear and angular velocity (v_m, w_m), further, the forward kinematics block converts them into the actual robot trajectory (x_m, y_m). The control design consists of three parts: kinematic control, ADRC and FPD.

A. Kinematic Control

The kinematic controller is responsible for generating the required control inputs (v_d, w_d) to drive a mobile robot from its current position towards a reference trajectory (x_r, y_r, θ_r). The control strategy is based on the error states, which define the deviation between the mobile robot's actual position and the desired reference position. The position errors are computed in the robot's local coordinate frame using the following equations:

$$e_x = (x_r - x_m)\cos(\theta_m) + (y_r - y_m)\sin(\theta_m) \tag{12}$$

$$e_y = -(x_r - x_m)\sin(\theta_m) + (y_r - y_m)\cos(\theta_m) \tag{13}$$

$$e_\theta = \theta_r - \theta_m \tag{14}$$

where (e_x) represents the longitudinal error, which measures the deviation along the robot's heading direction, (e_y) represents the lateral error, which measures the deviation perpendicular to the robot's heading direction and (e_θ) is the orientation error, defined as the difference between the reference and actual heading angles. A back-stepping control law is applied to correct these errors, where gains (k_x, k_y, k_θ) determine the convergence behavior and influence the tracking performance. The velocity control inputs are computed as:

$$v_d = k_x e_x + v_r \cos(e_\theta) \tag{15}$$

$$w_d = k_y v_r e_y + w_r + k_\theta \sin(e_\theta) \tag{16}$$

Here, linear velocity (v_d) is adjusted based on the longitudinal error and the reference velocity. while, angular velocity

(w_d) incorporates the lateral error and orientation correction to align the robot with the reference trajectory. This controller ensures that the mobile robot follows the desired trajectory with minimal deviation while maintaining stability and smooth motion.

B. Active Disturbance Rejection Control

Consider the below state-space model of the system represented as a first-order as follows:

$$\dot{x} = f(x(t), u(t), t) \tag{17}$$

This equation represents the system's dynamic model and captures how the states evolve under the influence of control inputs $u(t)$ and system dynamics [9]. Figure 3 represents the ADRC structure, two ADRC's are used one for linear and one for angular, where a first-order linear ADRC version is used in this work. In ADRC structure, PD controller is replaced with FPD which generates an initial signal (u) based on the rules and membership function. The ESO is used to estimate both the system output $x_1 = y$ and the disturbance $x_2 = f(y, t)$; the ESO uses these augmented states to estimate x_1 and x_2, allowing the controller to compensate for disturbances [10], which are computed as follows:

$$\dot{\hat{x}}_1 = \hat{x}_2 + b_0 u - L_1 e_{01} \tag{18}$$

$$\dot{\hat{x}}_2 = -L_2 e_{01} \tag{19}$$

where $\dot{\hat{x}}_1$, $\dot{\hat{x}}_2$ are the estimated state values for x_1 and x_2, e_{01} is the estimated error defined as $e_{01} = y(t) - \hat{x}_1$, where $y(t)$ is the output of the actual angular velocities [11]. The observer gains L_1 and L_2 are calculated using the observer bandwidth w_0 to ensure effective disturbance rejection, such that, $L_1 = 2\omega_0$ and $L_2 = \omega_0^2$. Lastly, the control signal u is subtracted from state $\hat{f}$ and fed into the last control block.

$$u_o = u \cdot (b_o)^{-1} \tag{20}$$

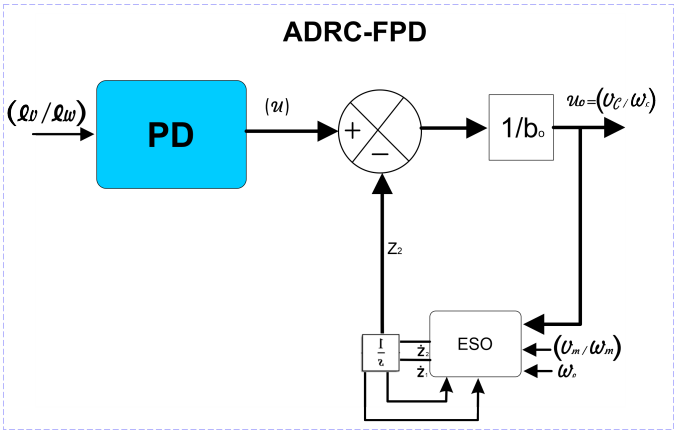

Fig. 3: ADRC-FPD Structure.

C. Reinforcement Learning

RL has shown itself to be a useful tool for automating controllers in recent years [1]. Without the use of mathematical models, RL allows controllers to learn the best control strategies through trial-and-error interactions with the system. The DDPG algorithm works well for continuous action spaces, such as PID gains. The critic network combines the state and action inputs, processes them in parallel, and then combines them in a common path.

Observation for the RL agent consist of desired right and left angular velocities ($\dot{\phi}_{dl}$, $\dot{\phi}_{dr}$), orientation angle (θ_r), errors ($e_{\dot{\phi}_l}$, $e_{\dot{\phi}_r}$) and derivative of the errors in angular velocities ($\dot{e}_{\dot{\phi}_l}$, $\dot{e}_{\dot{\phi}_r}$). Further, the reward function plays a crucial role in RL agent learning process. This function guides the agent during learning that the step taken is either positive or negative which is called rewards, a positive reward encourage the behavior that are favorable to the objective/goal, while negative are opposite. This function is a combination of parameters of the system which are related to actuators. Following is the reward function developed in this paper for DDMR.

$$R = -\alpha \cdot (e_{\dot{\varphi}_r} + e_{\dot{\varphi}_r}) - \beta \cdot \left(\tau_R^2 + \tau_L^2\right) - \gamma \cdot \text{Boundry} \tag{21}$$

Here, $\alpha = 0.001$, $\beta = 0.002$ and $\gamma = 1$, errors($e_{\dot{\phi}_l} + e_{\dot{\phi}_r}$) are angular velocities error penalty term for torque is added based on τ_L^2 and τ_R^2 and boundary term to encourage the agent to stay within desired operating conditions.The general architecture of RL block is shown in figure 3.

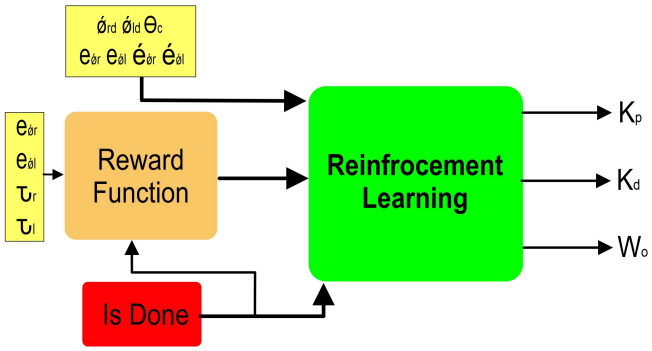

Fig. 4: RL Structure

IV. Results and Discussion

In this section, the simulation results of RL tuned ADRC controller implemented on DDMR (P3-DX) are discussed. The experiment is simulated on MATLAB/Simulink. A 30-second circular trajectory is provided to test the controller. The parameters related to dynamic model [16] used in the simulation are: $m_w = 0.5\,\text{kg}$, $m_c = 17\,\text{kg}$, $R = 0.095\,\text{m}$, $L = 0.24\,\text{m}$, $d = 0.05\,\text{m}$, $I_c = 0.537\,\text{kg} \cdot \text{m}^2$, $I_m = 0.0011\,\text{kg} \cdot \text{m}^2$, $I_w = 0.0023\,\text{kg} \cdot \text{m}^2$.

The actor network uses a fully connected structure with two layers of 400 neurons each for the state input, culminating in

the action layer with three outputs. While critic network contains two layers of 500 neurons. The selected hyper parameters for the RL agent training are shown in Table 1. Beside the learning algorithm hyper parameters also play a significant impact on the training of RL agent. The discount factor is set as 0.99 to have a stable learning, the noise has a lower decay at $1e-4$. So, that exploration should be for large period of training. Furthermore, total number of observations are 7 and actions are 3.

Discount Factor	0.99
Noise Standard Deviation	0.3
Noise Decay Rate	$1e^{-}4$
Actor Learning Rate	$1e^{-}4$
Critic Learning Rate	$3e^{-}4$
Experience Replay Buffer Size	$1e^{6}$
Mini-batch Size	128

TABLE I: Training Hyper-parameters

Figure 5 shows the x, y & θ tracking of desired input trajectory, robot's actual trajectory using ADRC and RL tuned ADRC for a circular motion. The controller operates with parameter values set as Kp= 150, Kd= 150 and wo= 50. By designing an effective reward function, the objective of the controller is achieved and proven through simulation results. It can be observed from the figures that estimation of the trajectory is almost perfect, where trajectories overlap entirely. This signifies that ESO was tell tuned. Also, RL tuned ADRC is able to follow desired trajectory . Figure 5 also contain trajectory error for x axis, y axis and θ for ADRC and RL tuned ADRC. The trajectory errors are almost negligible, which means that both controllers track the desired trajectory without much deviation in both X and Y axes.

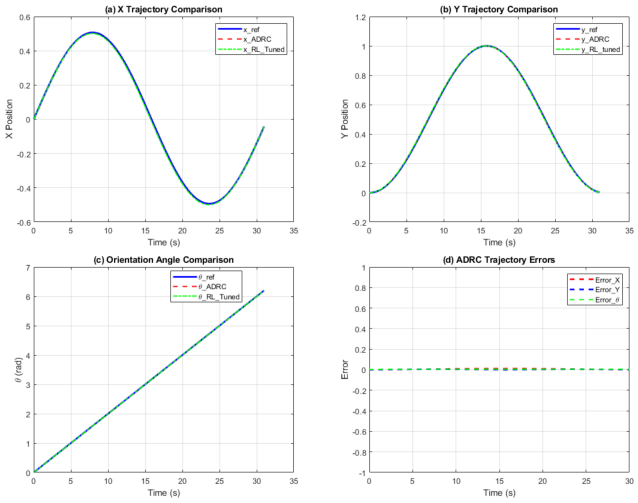

Fig. 5: Trajectory Tracking. (a) X-axis (b) Y-axis (c) θ (d) X, Y & θ trajectory errors

Figure 6 illustrates the angular velocities of the left and right wheels for the desired trajectory, the ADRC controller, and the RL-tuned ADRC controller. As shown, the signals exhibit initial oscillations before stabilizing and then closely following the desired angular velocities. The results indicate both technique follow the desired angular velocity but RL tuned ADRC has negligible oscillation as compared to manual tuned ADRC.

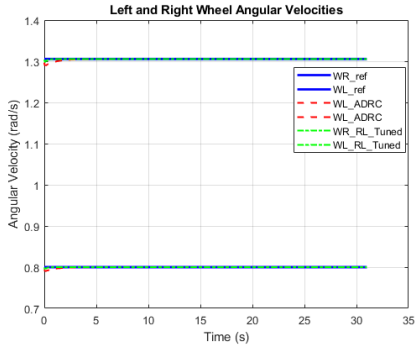

Fig. 6: Left & Right Wheel Angular Velocities

Figure 7 shows the x, y & θ trajectory tracking and errors after adding a step disturbance at t=5 seconds with the magnitude of 1. Its clearly visible in the graph as the continuous disturbance appears the robot start to deviate from the trajectory, ADRC is able to control the disturbance but RL tuned ADRC show better performances by optimizing the parameters. Also the system was tested on pulse disturbance of magnitude 0.5 at t=5 seconds and sinusoidal disturbance of magnitude 0.5 at t=5 seconds, the RL tuned controller performed well as compared to manual tuned ADRC. The trajectory errors of RL tuned ADRC are less then than the manual tuned ADRC, which indicates better performance.

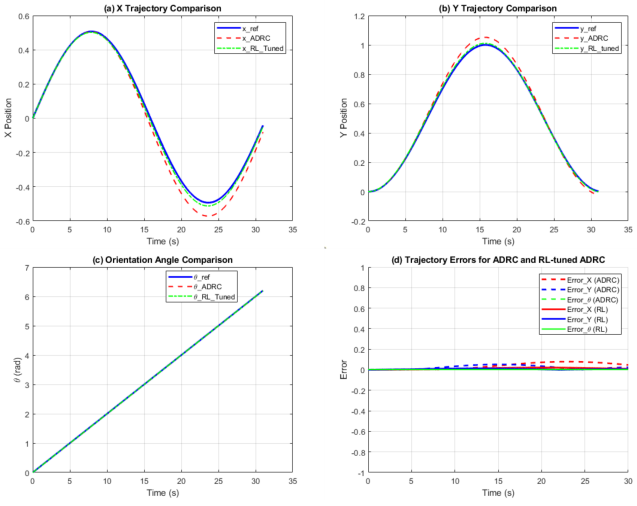

Fig. 7: Trajectory Tracking. (a) X-axis (b) Y-axis (c) θ (d) X, Y & θ trajectory errors after disturbance

Figure 8 contains left and right angular velocity errors after adding step disturbance, RL tuned ADRC slightly increasing angular velocities as compared to manual tuned ADRC to keep the robot on circular path.

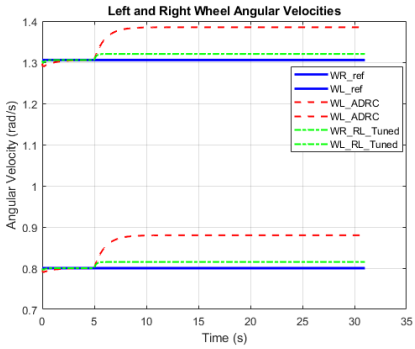

Fig. 8: Left & Right Wheel Angular Velocities after Disturbance

V. Conclusion

The goal of this research is to employ a DDPG-based RL agent to automatically adjust the parameters of an ADRC controller. In order to achieve our main goals, we investigate this hybrid control solution for a differential drive robot, which combines RL approaches with conventional control tactics. Although the industry holds ADRC controllers in high respect due to their strong state estimation and disturbance rejection capabilities, optimizing it's parameters is still a difficult undertaking. By using an RL agent, we can dynamically learn the best tuning parameters, which improves tracking performance. Our simulation findings show that ADRC and RL can be successfully integrated for parameter adjustment, since the system exhibits accurate trajectory tracking and near-perfect state estimation.

In the future, the simulation clearly indicates to lead the research to experiment level, also to build a controller based on neural network which overcomes the traditional ADRC, with the help of desired trajectory the RL controller result in output torque to control the system.

References

[1] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, "Reinforcement learning - overview of recent progress and implications for process control," *Computers and Chemical Engineering*, vol. 127, pp. 282–294, 2019.

[2] B. Belousov, H. Abdulsamad, P. Klink, S. Parisi, and J. Peters, Eds., *Reinforcement Learning Algorithms: Analysis and Applications*, ser. Studies in Computational Intelligence. Cham, Switzerland: Springer Nature Switzerland AG, 2021, vol. 883. [Online]. Available: https://doi.org/10.1007/978-3-030-41188-6

[3] D. T. J. K. I. P. Lucian Buşoniu, Tim de Bruin, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[4] P. T. Jardine, M. Kogan, S. N. Givigi, and S. Yousefi, "Adaptive predictive control of a differential drive robot tuned with reinforcement learning," *International Journal of Adaptive Control and Signal Processing*, vol. 33, no. 3, pp. 410–423, 2018.

[5] K. F. I. S. A. . Y. A. Mahrukh Shahid, Semab Naimat Khan, "Dynamic goal tracking for differential drive robot using deep reinforcement learning," *Neural Processing Letters*, vol. 55, p. 11559–11576, 2023.

[6] P. Abbeel, "Apprenticeship learning and reinforcement learning with application to robotic control," *Stanford University*, 2008.

[7] G. Wen, C. L. P. Chen, S. S. Ge, H. Yang, and X. Liu, "Optimized adaptive nonlinear tracking control using actor critic reinforcement learning strategy," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 4969–4979, 2019.

[8] G. Feng, L. Huang, and D. Zhu, "A nonlinear auto-disturbance rejection controller for induction motor," in *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON '98)*, vol. 3. IEEE, 1998, pp. 1509–1514.

[9] G. Herbst, "A simulative study on active disturbance rejection control (adrc) as a control tool for practitioners," *Electronics*, vol. 2, no. 3, pp. 246–279, 2013.

[10] N. H. Thai, T. T. K. Ly, H. Thien, and L. Q. Dzung, "Trajectory tracking control for differential-drive mobile robot by a variable parameter pid controller," *International Journal of Mechanical Engineering and Robotics Research*, vol. 11, no. 8, pp. 614–621, 2022.

[11] M. Drakulic and M. Stankovic, "Adrc-based trajectory tracking of unmanned tracked vehicles under high slippage disturbance," in *2024 14th International Conference on Electrical Engineering (ICEENG)*. IEEE, 2024, pp. 1–6.

[12] Ningyue, Liyan, and Liukeping, "The research of mobile manipulator trajectory tracking cooperative control based on the adrc," in *2015 International Conference on Electrical and Electronic Engineering*. IEEE, 2015, pp. 385–389.

[13] M. Kia, P. Mansouri, and A. Javdani, "Modeling and simulation of a single gain tuning adrc controller in matlab/simulink," *Datis Elevator*, 2021, available online.

[14] Y. Wen-bin and Y. Dong, "Modeling and simulation of an active disturbance rejection controller based on matlab/simulink," *International Journal of Research in Engineering and Science (IJRES)*, vol. 3, no. 7, pp. 62–69, 2015. [Online]. Available: http://www.ijres.org/

[15] C. Fu and W. Tan, "Tuning of linear adrc with known plant information," *ISA Transactions*, vol. 65, pp. 384–393, 2016.

[16] M. Nasir, "Dynamic modeling and control of wheeled mobile robot using active disturbance rejection control for trajectory tracking," University of Sharjah, Tech. Rep., 2024, accessed on 26 October 2024.