2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Enhancing 3D Trajectory Tracking of Robotic Manipulator Using Sequential Deep Reinforcement Learning with Disturbance Rejection

Saikat Majumder and Soumya Ranjan Sahoo

Abstract— **This paper addresses the problem of trajectory tracking for robotic manipulators in three-dimensional space. We use Deep Deterministic Policy Gradient (DDPG), a model-free deep reinforcement learning technique, with a sequential training to significantly expedite the training process. The reward function is a key component in reinforcement learning. Our proposed design is particularly tailored to handle external disturbances. This feature ensures robust performance and adaptability, which is crucial for real-world applications of robotic manipulators in dynamic environments. To evaluate the effectiveness and efficiency of our approach, we present comprehensive numerical simulation results. These results not only demonstrate the capability of our model to facilitate a faster training rate but also showcase a remarkable reduction in the tracking mean square error (MSE).**

I. INTRODUCTION

In recent years, Deep Reinforcement Learning (DRL) has emerged as a powerful and popular paradigm in the field of artificial intelligence and robotics. This cutting-edge approach has the remarkable ability to tackle complex tasks, making it a focal point of research and development. DRL represents the fusion of Reinforcement Learning (RL) and Artificial Neural Networks (ANNs), ushering in a new era of intelligent systems. A pivotal moment in the rise of DRL came when DeepMind, a renowned AI research organization, harnessed this technology to develop groundbreaking DRL agents [1]. These agents learned to master the Atari 2600, using only the raw pixel data from the game screen as input. Another prominent example of DRL's success is AlphaZero [2] that excels in strategic board games like chess, shogi, and go, rivaling and often surpassing human-level performance.

DRL is not confined to the virtual world of gaming; it extends its influence to real-world applications as well. One particular DRL technique, known as Deep Deterministic Policy Gradient (DDPG), has proven to be invaluable in addressing the challenges of high-dimensional control spaces [3]. This technique offers a promising avenue for devising control strategies for robotic manipulators, which play a pivotal role in various fields [4], [5], [6]. Robotic manipulators are extensively employed where they can perform laborious and often hazardous tasks that are unsuitable for humans. In each of these applications, precision in motion control is of paramount importance.

Traditional control methods, such as adaptive control [7], fuzzy control [8], and robust control [9], have been previously utilized for robotic manipulators. Traditional approaches to controlling robotic manipulators come with numerous limitations when dealing with unstructured scenarios, relying heavily on environmental models. When the environment undergoes alterations, it necessitates the reconstruction of both the manipulator's mathematical model and the environment itself [4]. To overcome these challenges and adapt to diverse and dynamic environments, researchers have turned to Reinforcement Learning (RL). This allows robotic manipulators to learn from their interactions with the environment, making them more adaptable and robust.

In recent research, several papers have addressed the challenging task of trajectory tracking in the context of robotic systems. Authors in [5] introduced an improved version of the Deep Deterministic Policy Gradients (DDPG) algorithm, known as Distributed DDPG, to enhance the trajectory tracking performance of SCARA robots. Similarly, in [10], trajectory tracking was demonstrated for a two-link robot manipulator. In [11], the researchers proposed a controller that combines traditional control laws with Deep Reinforcement Learning (DRL) techniques for a 3 degree-of-freedom (DoF) manipulator. However, it's worth noting that none of the study considered external disturbances, which are often encountered in real-world applications. This leaves an open challenge for future research in the field of reinforcement learning for robotics control, where the integration of effective disturbance rejection strategies remains a critical area of investigation in conjunction to DRL.

In this paper, the goal is to create a control system based on DRL such that an n-link robotic manipulator tracks a desired trajectory in 3-dimension even in the presence of bounded external disturbances. The following is the major summary of this paper:

- An adaptive control approach based on DDPG, which has adaptability and precise 3 dimensional trajectory tracking capacity, is proposed for manipulator system. This adaptive method takes into account the effect of nonlinearity, uncertainty and also time-varying external disturbances in the dynamic model.
- A reward function is proposed which ensures DDPG agent learns effectively and steadily.
- Networks are trained sequentially for fast learning.
- The efficacy of the proposed method is demonstrated by simulation results.

Saikat Majumder and Soumya Ranjan Sahoo are with the Department of Electrical Engineering, Indian Institute of Technology, Kanpur, India {saikatm21, srsahoo}@iitk.ac.in

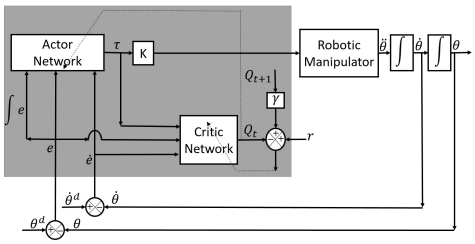


Fig. 1: Controller set up for Robotic Manipulator

II. Control Design for an n-Link Robot Manipulator

A. Dynamic Model

The dynamics of an n-link robotic manipulator is determined by the relationship between joint control input ($\tau \in \mathbb{R}^n$), external disturbance ($\tau_d \in \mathbb{R}^n$), angular position ($\theta \in \mathbb{R}^n$), angular velocity ($\dot{\theta} \in \mathbb{R}^n$), and angular acceleration ($\ddot{\theta} \in \mathbb{R}^n$) [12], as expressed in (1):

$$\mathbf{M}(\theta)\ddot{\theta} + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} + \mathbf{G}(\theta) = \tau + \tau_d. \tag{1}$$

Here, $\mathbf{M}(\theta) \in \mathbb{R}^{n\times n}$ represents the manipulator mass matrix, $\mathbf{C}(\theta, \dot{\theta}) \in \mathbb{R}^{n\times n}$ accounts for centrifugal and Coriolis effects, and $\mathbf{G}(\theta) \in \mathbb{R}^n$ denotes the gravitational force. $\mathbf{M}(\theta)$ is invertible due its symmetry and positive definiteness. So, (1) is reformulated as:

$$\ddot{\theta} = \mathbf{M}^{-1}(\theta)\left[-\mathbf{C}(\theta, \dot{\theta})\dot{\theta} - \mathbf{G}(\theta) \ + \tau + \tau_d\right] \tag{2}$$

The assumptions made for this mathematical model are as follows:

A1: The robot arms are rigid and symmetric.
A2: The center-of-gravity (CoG) of each robotic arm coincides with its centroid.
A3: External disturbances are bounded.
A4: The maximum torque exerted by the motor (actuator) at a joint is $\tau_{\max}$.

A DDPG Actor-Critic Network has been employed for the controller design of the robotic manipulator, as depicted in Figure 1. The training process uses a sequential approach. In the following sections, we elaborate on the DDPG learning law and our training method for the manipulator controller.

B. DDPG learning law

The learning process of the DDPG algorithm, described in [3], involves training the Critic network, denoted by the action-value function $Q(s, \tau|\phi^Q)$, using the Bellman equation similar to the Deep Q-Network (DQN) [1]. Let s_k represent the state vector at time step k. Utilizing s_k and the control input $\tau_k = \pi(s_k|\phi^\pi)$ at step k, the subsequent state (s_{k+1}) and the reward r_k are determined within each episode lasting from 0 to T seconds.
After certain number of steps, the Actor-Critic network undergoes updates leveraging rewards and Q-values at each step. Consider N_T to be the total number of steps in an episode, where each time step size is of T/N_T seconds. The Q-value at step k for $k \in \{0, 1, 2....N_T\}$, derived from the Critic-Network, is denoted as $Q(s_k, \tau_k|\phi^Q)$. The desired Q-value at step k is computed as:

$$Q_k^d = r_k + \gamma Q(s_{k+1}, \tau_{k+1}|\phi^Q) \tag{3}$$

Here, γ represents the discount factor ($0 \leq \gamma \leq 1$), $r_k = r(s_k, \tau_k)$ is the reward at step k. The Critic-Network updates occur through gradient descent, minimizing the loss function, L given in (4).

$$L = \frac{1}{N}\sum_{i=k}^{N}(Q_k^d - Q(s_k, \tau_k|\phi^Q))^2. \tag{4}$$

Simultaneously, the Actor-Network updates are based on the expected return J, where,$J = E_{r_k, s_k\sim g, \tau_k\sim\pi}[R_k]$ (g signifying the environment) and $R_k = \sum_{i=k}^{N}\gamma^{(i-k)}r(s_k, \tau_k)$. The gradient for the update of the Actor-Network is given by

$$\nabla_{\phi^\pi} J \approx E[\nabla_{\phi^\pi} Q(s, \tau|\phi^Q)|_{s=s_k, \tau=\pi(s_k|\phi^\pi)}] \tag{5}$$
$$= E[\nabla_\tau Q(s, \tau|\phi^Q)|_{s=s_k, \tau=\pi(s_k)} \nabla_{\phi^\pi}\pi(s|\phi^\pi)|_{s=s_k}]$$
$$\approx \frac{1}{N}\sum_{i=k}^{N}\nabla_\tau Q(s, \tau|\phi^Q)|_{s=s_k, \tau=\pi(s_k)} \nabla_{\phi^\pi}\pi(s|\phi^\pi)|_{s=s_k} \tag{6}$$

Furthermore, in [3] DDPG employs a target network similar to DQN, with the network parameters updated through exponential smoothing as follows:

$$\phi^{Q'} \leftarrow \rho\phi^Q + (1-\rho)\phi^{Q'}$$
$$\phi^{\pi'} \leftarrow \rho\phi^\pi + (1-\rho)\phi^{\pi'}$$

To ensure gradual and stable changes in the target network, the hyperparameter ρ is set such that $0 < \rho << 1$.

C. Proposed Training approach

In this paper, we adopt a dual Actor-Critic network strategy to enhance the training process. Instead of relying on a single Actor-Critic network for direct 3D space trajectory training, we employ two separate Actor-Critic networks. The primary actor-critic network focuses on generating control inputs for the joints required to track a 2D trajectory, while the secondary actor-critic network is responsible for extending the manipulator tracking into 3D space.

Initially, we train the Actor-Critic network for the 2D trajectory up to a certain level of proficiency. Subsequently, both networks undergo joint training for 3D trajectory.

Algorithm 1 Pseudo-code of our proposed algorithm

Hyperparameters: soft update factor ρ, reward discount factor γ, actor learning rate η^a and critic learning rate η^c.

Input: empty replay buffer D_1, D_2, initialize parameters σ_1, σ_2 of critic networks $Q_1(s_1, \tau_1|\sigma_1), Q_2(s_2, \tau_2|\sigma_2)$ and parameters ϕ_1, ϕ_2 of actor networks $\pi_1(s_1|\phi_1)$, $\pi_2(s|\phi_2)$. Initialize target network Q_1', Q_2', and π_1', π_2' with weights $\sigma_1' \leftarrow \sigma_1, \sigma_2' \leftarrow \sigma_2$ and $\phi_1' \leftarrow \phi_1, \phi_2' \leftarrow \phi_2$, Initialize a random process N_1 and N_2 for action exploration.

Procedure:

Step 1: 2D trajectory

Receive initial observation state s_1
do
  for t = 1 to T **do**
    $action_1 \tau_1^t = \pi_1(s_1^t|\phi_1^t) + N_1^t$
    Execute τ_1^t, observe reward r_1^t, calculate absolute error and observe new state s_1^{t+1}
    Store transition $(s_1^t, \tau_1^t, r_1^t, p^t, s_1^{t+1})$ in D_1
    Set $Y_1^i = r_1^t + \gamma(1-p^t)Q_1'(s_1^{t+1}, \pi_1'(s_1^{t+1}|\phi_1')|\sigma_1')$
    Critic network loss function:
$$L_1^c = \frac{1}{N}\sum_i (Y_1^i - Q(s_1^i, \tau_1^i|\sigma_1))^2$$
    Update critic network with learning rate η^c by minimizing the loss with gradient descent:
$$\nabla_{\sigma_1} L_1^c = \nabla_{\sigma^1}[\frac{1}{N}\sum_i (Y^i - Q(s_1^i, \tau_1^i|\sigma_1))^2]$$
    Update the actor policy with learning rate η^a using the sampled policy gradient and minimizing actor loss functions with gradient descent:
$$\nabla_{\phi_1} L_1^a = -\nabla_{\phi_1} J_1$$
    where, $\nabla_{\phi_1} J_1 \approx$
$$\frac{1}{N}\sum_{i=k}^{N} \nabla_\tau Q(s, \tau|\sigma_1)|_{s=s_1^i, \tau=\pi_1(s_1^i)} \nabla_{\phi_1} \pi_1(s|\phi_1)|_{s=s_1^i}$$
    Update the target networks:
$$\phi_1' \leftarrow \rho\phi_1 + (1-\rho)\phi_1'$$
$$\sigma_1' \leftarrow \rho\sigma_1 + (1-\rho)\sigma_1'$$
  end for
  calculate absolute mean square error
while absolute mean square error $<$ 1 cm

The trained policies π_1, value functions Q_1, and their updated target networks π_1' and Q_1' from Algorithm 1 will be utilized as the initial configuration for step 2.

Step 2: 3D trajectory

Receive initial observation state s_1, s_2
do
  for t = 1 to T **do**
    $action_1 \tau_1^t = \pi_1(s_1^t|\phi_1^t) + N_1^t$
    $action_2 \tau_2^t = \pi_1(s_2^t|\phi_2^t) + N_2^t$
    Execute τ_1^t, τ_2^t, observe reward r_1^t, r_2^t, calculate absolute error and observe new state s_1^{t+1}, s_2^{t+1}
    Store transition $(s_1^t, \tau_1^t, r_1^t, p^t, s_1^{t+1})$ in D_1
    Store transition $(s_2^t, \tau_2^t, r_2^t, p^t, s_2^{t+1})$ in D_2
    Set $Y_1^i = r_1^t + \gamma(1-p^t)Q_1'(s_1^{t+1}, \pi_1'(s_1^{t+1}|\phi_1')|\sigma_1')$
    Set $Y_2^i = r_2^t + \gamma(1-p^t)Q_2'(s_2^{t+1}, \pi_2'(s_2^{t+1}|\phi_2')|\sigma_2')$
    Critic network loss function:
$$L_1^c = \frac{1}{N}\sum_i (Y_1^i - Q(s_1^i, \tau_1^i|\sigma_1))^2$$
$$L_2^c = \frac{1}{N}\sum_i (Y_2^i - Q(s_2^i, \tau_2^i|\sigma_2))^2$$
    Update critic network with learning rate η^c by minimizing the loss with gradient descent:
$$\nabla_{\sigma_1} L_1^c = \nabla_{\sigma^1}[\frac{1}{N}\sum_i (Y_1^i - Q(s_1^i, \tau_1^i|\sigma_1))^2]$$
$$\nabla_{\sigma_2} L_2^c = \nabla_{\sigma^2}[\frac{1}{N}\sum_i (Y_2^i - Q(s_2^i, \tau_2^i|\sigma_2))^2]$$
    Update the actor policy with learning rate η^a using the sampled policy gradient and minimizing actor loss functions with gradient descent:
$$\nabla_{\phi_1} L_1^a = -\nabla_{\phi_1} J_1$$
$$\nabla_{\phi_2} L_2^a = -\nabla_{\phi_2} J_2$$
    where, $\nabla_{\phi_m} J_m \approx$
$$\frac{1}{N}\sum_{i=k}^{N} \nabla_\tau Q(s, \tau|\sigma_m)|_{s=s_m^i, \tau=\pi_m(s_m^i)} \nabla_{\phi_m} \pi_m(s|\phi_m)|_{s=s_m^i}$$
    and $m = 1, 2$
    Update the target networks:
$$\phi_1' \leftarrow \rho\phi_1 + (1-\rho)\phi_1'$$
$$\phi_2' \leftarrow \rho\phi_2 + (1-\rho)\phi_2'$$
$$\sigma_1' \leftarrow \rho\sigma_1 + (1-\rho)\sigma_1'$$
$$\sigma_2' \leftarrow \rho\sigma_2 + (1-\rho)\sigma_2'$$
  end for
  calculate absolute mean square error
while absolute mean square error $<$ 1 mm

This sequential training method results in substantial improvements in stability and accelerates the convergence of the algorithm. For a more in-depth understanding of our proposed training approach, please refer to Algorithm 1 in the paper. In this algorithm, all parameters marked with "1" pertain to the Actor-Critic network responsible for the 2D trajectory, which is trained first. Subsequently, the parameters marked with "2" correspond to the Actor-Critic network that provides control input for the joints responsible for the 3D trajectory. Eventually, both actor-critic networks are trained simultaneously for the 3D trajectory, once the actor-critic network 1 has achieved a certain level of proficiency in the 2D trajectory.

D. Design of reward function

The reward function serves as a motivational tool, guiding the agent by using a system of rewards and penalties to define correct and incorrect actions. Agents strive to optimize their cumulative rewards. In the recommended approach, the reward function at time step k computes a numerical score by considering errors in angular position (e_i^k), angular velocity ($\dot{e}_i^k$), and angular acceleration ($\ddot{e}_i^k$) where $i = 1, 2$. In this proposed approach, the reward function at time step k is formulated as follows:

$$E_i = -a||e_i^k||_1 - b||\dot{e}_i^k||_1 - c||\ddot{e}_i^k||_1$$
$$r_1^k = max(E_1, -20\alpha) + max(-d||\int e_1^k||_1, -20\alpha) \quad (7)$$
$$r_2^k = -f||e_p||_1 - max(-E_2, -20\beta) + max(-h||\int e_2^k||_1, -20\beta) \quad (8)$$

where a, b, c, d, f, h are finite positive constants and α, β are the number of joints responsible for 2D and 3D trajectory tracking respectively. Increasing the reward, decreases the e_i^k, $\dot{e}_i^k$ and $\ddot{e}_i^k$ at each time step k.

III. SIMULATION

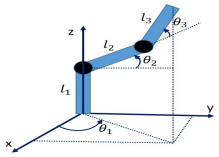


Fig. 2: Three link manipulator schematic diagram

On a simulated three-link robotic manipulator, we apply the suggested model. The dynamics of the three-link manipulator is explained in [12]. The angles of the three links in the manipulator are $\theta = [\theta_1 \quad \theta_2 \quad \theta_3]^T$, and $\tau = [\tau_1' \quad \tau_2' \quad \tau_3']^T$ are the torques applied to the three joints. In the suggested DRL model, state (s) refers to the error between desired and actual joint angle and velocity. We denote $s_1 = [(\theta_2^d - \theta_2) \quad (\theta_3^d - \theta_3) \quad (\dot{\theta}_2^d - \dot{\theta}_2) \quad (\dot{\theta}_3^d - \dot{\theta}_2) \int(\theta_2^d - \theta_2)dt \quad \int(\theta_3^d - \theta_3)dt]^T$ as joint 2 and 3 are responsible for 2D trajectory tracking in vertical plane and $s_2 = [(\theta_1^d - \theta_1) \quad (\dot{\theta}_1^d - \dot{\theta}_1) \quad \int(\theta_1^d - \theta_1)dt]^T$. Similarly, $\tau_1 = [\tau_2' \quad \tau_3']$ and $\tau_2 = [\tau_1']$

Table I shows the list of physical parameters and their corresponding values of the three-link manipulator considered for this numerical simulation.

TABLE I: Physical parameter values of three-link Robotic Manipulator

Symbol	Parameter	Value
l_1	Length of the 1^{st} link	0.04 m
l_2	Length of the 2^{nd} link	0.17 m
l_3	Length of the 3^{rd} link	0.132 m
g	Gravity acceleration	9.8 m/s^2
m_1	Mass of the 1^{st} link	0.1 kg
m_2	Mass of the 2^{nd} link	0.17 kg
m_3	Mass of the 3^{rd} link	0.1 kg
r_1	radius of the 1^{st} link	0.039 m
r_2	radius of the 2^{nd} link	0.02 m
r_3	radius of the 3^{rd} link	0.02 m

TABLE II: Parameter values for both the Actor-Critic Networks

Symbol	Parameter	Value
γ	Discount factor	0.6
η^c	Learning rate of Critic-network	10^{-4}
η^a	Learning rate of each Actor-network	10^{-6}
N_h^c	Number of hidden layers in Critic-Network	2
N_h^a	Number of hidden layers in each Actor-Network	2
N_u^c	Number of units in each hidden layer in Critic-Network	64
N_u^a	Number of units in each hidden layer in Actor-Network	64
ρ	Hyperparameter for updating target network	0.01

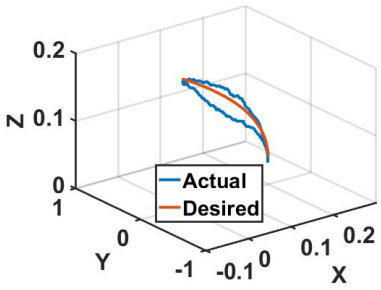
The Actor-critic networks within the simulation have no prior knowledge of the dynamics and parameters of the three-link manipulator. Table II provides details of the parameters for the Actor-Critic model. Initial weights for both the Actor and Critic networks are randomly generated. The manipulator undergoes training in episodes, with constraints imposed on angular position, angular velocity, and acceleration to ensure they stay within the physical limits throughout the simulation.

A. Analysis of 2D Trajectory Results

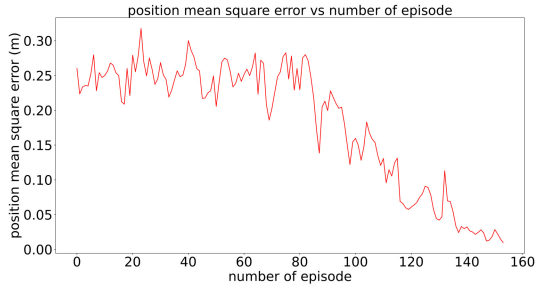
In Figure 3, we present the simulation outcomes for the initial phase, which entails tracking a 2D trajectory. During this initial phase, we train the first Actor-Critic network until it achieves a position mean square error of less than 1 cm, a milestone reached after 158 episodes. Each episode corresponds to the complete trajectory that the manipulator needs to follow in the X-Z plane, comprising 1000 points. The comparison between the desired and actual trajectories after training is illustrated in Figure 3a. Furthermore, Figures 3b and 3c show the evolution of position mean square error and episodic rewards over the course of the training episodes.

B. Analysis of 3D Trajectory Results

In Figure 4, we present the simulation results for the second phase, which involves tracking a 3D trajectory. In



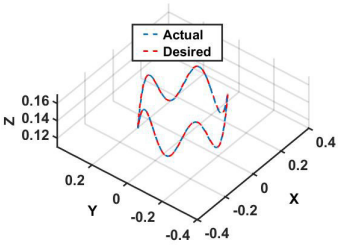
(a) Trajectory tracking in 2D plane



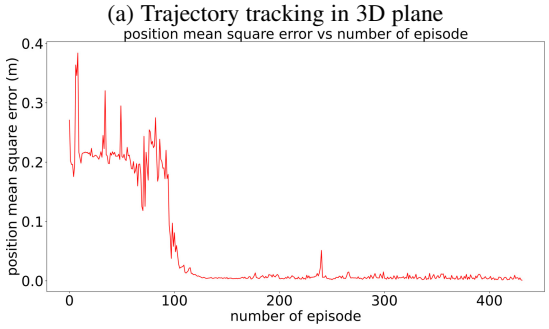
(b) Position mean square error vs no. of episode

(c) Episodic reward vs no. of episode

Fig. 3: Trajectory tracking in the vertical 2D plane



(a) Trajectory tracking in 3D plane



(b) Position mean square error vs no. of episode

Fig. 4: Trajectory tracking in the vertical 2D plan

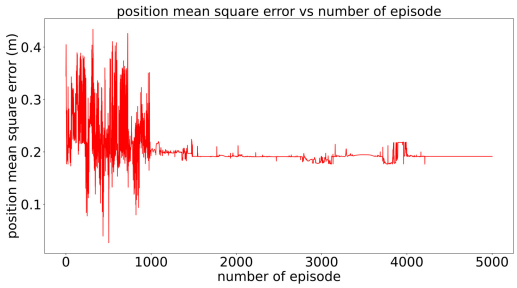
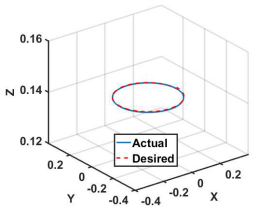
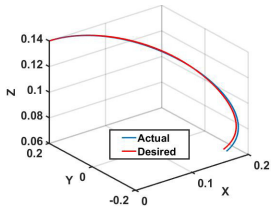


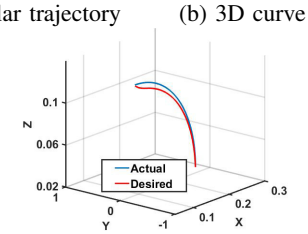
Fig. 5: position mean square error(mse) vs. no. of episodes without sequential learning



(a) Circular trajectory

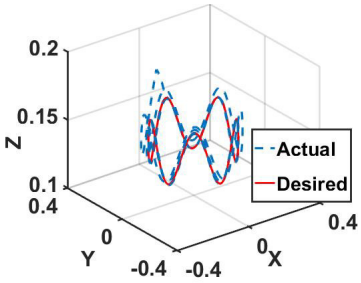


(b) 3D curve trajectory

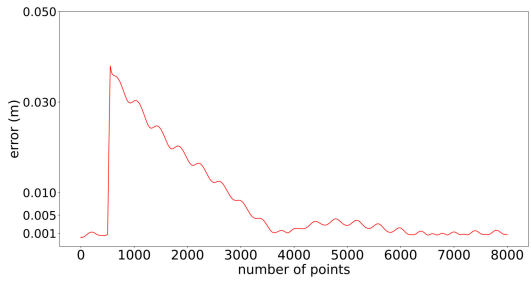


(c) 2D curve trajectory

Fig. 6: Different trajectory tracking testing

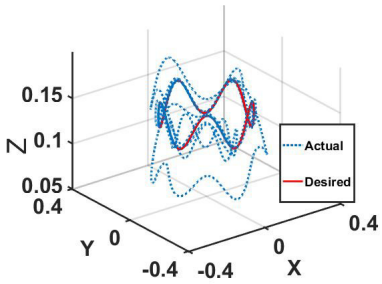


(a) Trajectory tracking after impulse disturbance

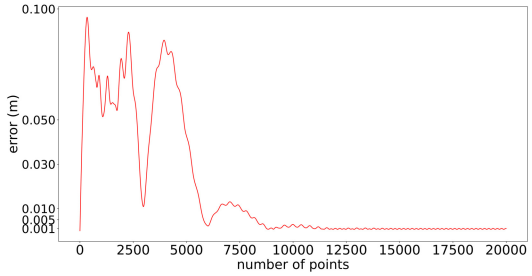


(b) Evaluation of position mse with trajectory points

Fig. 7: Trajectory tracking after impulse disturbance



(a) Trajectory tracking after constant disturbance



(b) Evaluation of position mse with trajectory points

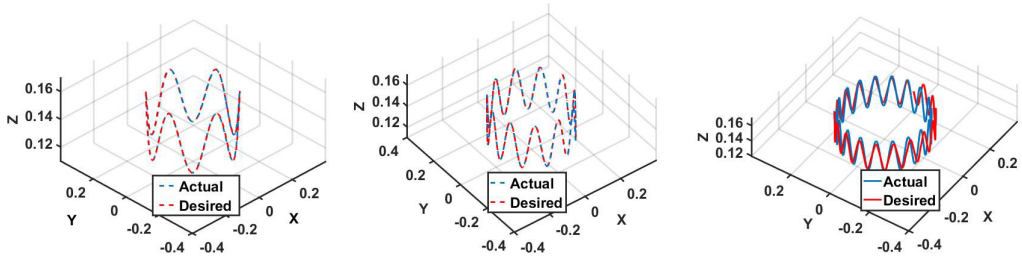
Fig. 8: Trajectory tracking after constant disturbance

this phase, we concurrently train both Actor-Critic networks until the position mean square error is reduced to less than 1 mm, a goal achieved after 458 episodes. Each episode corresponds to the complete trajectory that the manipulator needs to follow in 3D space, comprising 4000 points. Figure 4a illustrates the comparison between the desired and actual trajectories after training. Additionally, Figures 4b depict the progression of position mean square error throughout the training episodes. Following training, we assessed the performance of our Actor-Critic network controller in tracking various trajectories, both in 2D and 3D, as depicted in Figure 6. The simulation results presented in Figure 5 depict the evaluation of mse for position across varying numbers of episodes during the training phase for both Actor-Critic networks, without employing sequential training. Notably, these results indicate that the mse for position does not exhibit a reduction, even after 5000 episodes, suggesting that the networks fail to converge.

C. Analysis of 3D Trajectory Tracking Under External Disturbance

Figure 8 depicts the outcomes of a trajectory tracking simulation when subjected to an abrupt external disturbance. During the initial rotation of the manipulator, a torque of 1 Nm is exerted on the end-effector from the 500th to the 550th point along the trajectory. The impact of this disturbance is illustrated in Figure 7a, where the error shows an initial increase but gradually diminishes during the subsequent rotation. Figure 7b portrays the error assessment concerning the number of points, revealing that the mean square error falls below $1mm$ after the disturbance subsides.

Next, Figure 8a displays the results of a simulation involving a continuous external force. In this scenario, a



(a) Number of cycle 6 (b) Number of cycle 11 (c) Number of cycle 18

Fig. 9: Trajectory tracking in 3D space in various frequency

torque of $0.5Nm$ is consistently applied to the manipulator's end-effector throughout the trajectory tracking process. As depicted in Figure 8a, the error exhibits an initial increase in response to the constant disturbance but gradually decreases over time. The error evaluation in Figure 8b, concerning the number of points, indicates that the mean square error decreases to approximately $1.1mm$.

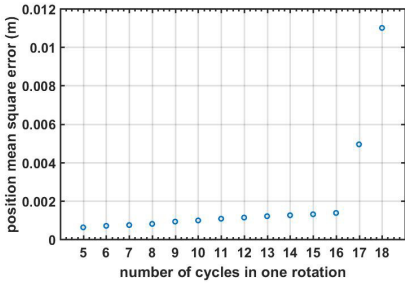


Fig. 10: Position mean square error(mse) vs. no. of cycle in one rotation

D. Analysis of 3D Trajectory Tracking at Various Frequencies

In Figure 9, we showcase simulation results illustrating the performance of 3D trajectory tracking at different frequencies. The Actor-Critic network controller undergoes training by following a 3D trajectory spanning five complete cycles. The visual representations in the figures demonstrate that the manipulator, when controlled by our designed controller, can successfully track as many as 16 cycles within a single rotation. The graph in Figure 10 displays how the error varies with different numbers of cycles per rotation.

IV. CONCLUSIONS

This paper delves into the challenge of managing robotic manipulator systems across various environments characterized by unknown parameters. It presents an optimal control strategy achieved through the application of a neural network. This network model comprises two distinct subnetworks: an actor network, responsible for acquiring the optimal control strategy by maximizing the performance index, and a critic network, which approximates the performance index. We employ a sequential training approach with two Actor-Critic models to expedite the learning process. The adaptability of the network model to the unfamiliar physical properties and dynamics of robotic manipulators in diverse environments stems from its learning capabilities. The effectiveness of this approach is demonstrated through simulations involving a three-link manipulator.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] R. Dong, J. Du, Y. Liu, A. A. Heidari, and H. Chen, "An enhanced deep deterministic policy gradient algorithm for intelligent control of robotic arms," *Frontiers in Neuroinformatics*, vol. 17, p. 1096053, 2023.

[5] Y. Hu and B. Si, "A reinforcement learning neural network for robotic manipulator control," *Neural computation*, vol. 30, no. 7, pp. 1983–2004, 2018.

[6] S. Zhang, C. Sun, Z. Feng, and G. Hu, "Trajectory-tracking control of robotic systems via deep reinforcement learning," in *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE, 2019, pp. 386–391.

[7] J. M. Martín-Sánchez, J. M. Lemos, and J. Rodellar, "Survey of industrial optimized adaptive control," *International Journal of Adaptive Control and Signal Processing*, vol. 26, no. 10, pp. 881–918, 2012.

[8] R.-E. Precup and H. Hellendoorn, "A survey on industrial applications of fuzzy control," *Computers in industry*, vol. 62, no. 3, pp. 213–226, 2011.

[9] M. W. Spong, "On the robust control of robot manipulators," *IEEE Transactions on automatic control*, vol. 37, no. 11, pp. 1782–1786, 1992.

[10] E. Bejar and A. Morán, "Predictive control of a robot manipulator with deep reinforcement learning," in *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, 2021, pp. 127–130.

[11] A. Liu, B. Zhang, W. Chen, Y. Luo, S. Fang, O. Zhang, Z. Liu, Z. Wang, and J. Liu, "Reinforcement learning based control for uncertain robotic manipulator trajectory tracking," in *2022 China Automation Congress (CAC)*, 2022, pp. 2740–2745.

[12] H. Sadegh and H. Zarabadipour, "Modeling, simulation and position control of 3dof articulated manipulator," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 2, no. 3, pp. 132–140, 2014.