Ain Shams Engineering Journal 16 (2025) 103595

Contents lists available at ScienceDirect

Ain Shams Engineering Journal

journal homepage: https://www.sciencedirect.com

Full Length Article

Physics-informed reward shaped reinforcement learning control of a robot manipulator

Raouf Fareh [a,*], Tanjulee Siddique [b], Kheireddine Choutri [c], Dmitry V. Dylov [d]

[a] Electrical Engineering Department, University of Sharjah, Sharjah, 27272, United Arab Emirates
[b] Research Institute of Sciences & Engineering (RISE), University of Sharjah, Sharjah, 27272, United Arab Emirates
[c] Aeronautical and Space Studies Institute, Blida 1 University, Blida, 0900, Algeria
[d] Skolkovo Institute of Science and Technology, Bolshoi Blvd. 30, Bld. 1, Moscow, 121205, Russia

ARTICLE INFO

Keywords:
Reinforcement learning
Reward shaping
Robot manipulator
Tracking control
Physics-informed NN

ABSTRACT

Reinforcement Learning (RL)-based control plays a pivotal role in developing adaptive robotic systems capable of optimal decision-making in dynamic and uncertain environments. Its ability to learn directly from interaction makes RL particularly effective for handling complex control tasks where traditional model-based approaches often fall short. However, conventional RL algorithms typically suffer from slow convergence rates and may lack stability guarantees, especially in continuous control applications. To address these limitations, this paper proposes a novel RL-based control framework for a 4-DOF robotic manipulator, leveraging an improved Physics-Informed Deep Deterministic Policy Gradient (PI-DDPG) agent for precise trajectory tracking. The proposed agent integrates Physics-Informed Neural Networks (PINNs) into the RL architecture, allowing it to exploit prior knowledge of the system's dynamics to significantly accelerate convergence. Furthermore, a Lyapunov-based reward shaping strategy is introduced to enhance the stability and reliability of the learning process without compromising optimality. An adaptive noise generation technique is also proposed to dynamically regulate exploration, improving the agent's ability to discover effective control actions. The effectiveness of the PI-DDPG framework is validated through both simulation and real-world experiments. Results show a 20–30% improvement in tracking accuracy and a threefold (x3) reduction in convergence time compared to baseline RL approaches, demonstrating the potential of physics-informed reinforcement learning for high-performance and robust robotic control.

1. Introduction

Robot manipulators have slowly become integral to a wide range of applications, from manufacturing [1,2] and medical surgery [3–5] to space exploration [6–8]. These robotic systems require sophisticated control strategies to perform complex tasks with precision and reliability, whereas good control strategies lead to controlled behavior and reliable results. The use of traditional controllers, however, becomes challenging in high-dimensional and dynamic environments. Reinforcement learning (RL) has emerged as a promising approach for training robots to perform complex tasks autonomously by learning from their interactions with the environment [9]. RL provides a framework for agents to learn optimal behavior by receiving feedback in the form of rewards while navigating through the environment. With higher dimensions, as is the case of robotic control, challenges stem from high-dimensional states and action spaces, such as low convergence speed and the inability to find an optimal policy. There has been remarkable work done [10–13], from designing learning algorithms that enable the use of continuous state and action spaces, to introducing asynchrony into the existing actor-critic structure to ensure faster convergence speed. Since its advent, RL has received continual advancement, reaching exponential heights with the recent boom in the interest in artificial intelligence (AI). While much work has been done to improve the efficiency of RL, which is undoubtedly significant, robust control still stands as a difficult task to deal with.

In this regard, many researchers have opted for a combination approach, merging two or more control techniques to complete tasks successfully. A combination of different intelligent approaches has been a rising application for robust tracking of robot manipulators. In [14], fuzzy Q-learning was applied to a 2-link manipulator to observe robust

* Corresponding author.
E-mail address: rfareh@sharjah.ac.ae (R. Fareh).

https://doi.org/10.1016/j.asej.2025.103595
Received 11 December 2024; Received in revised form 17 June 2025; Accepted 23 June 2025
Available online 24 July 2025

tracking in uncertain environments. In another paper [15], fuzzy Q-learning was used to achieve an adaptive position control scheme for a robotic manipulator with angular velocity and position constraints.

Yet, whether standalone or combination RL is realized, one of the most challenging aspects of either technique is often a well-designed reward function, as, without one, the training takes longer to converge or, in the worst case scenario, doesn't converge at all. In response to this, researchers have further explored methods to enhance the effectiveness of RL algorithms with the reward function in mind. One such technique garnering significant attention is the integration of Reward Shaping (RS) [16,17]. RS involves modifying the reward function provided to the RL agent to provide additional guidance and accelerate learning. The goal of integrating RL techniques with RS mechanisms is to force the robot to execute diverse tasks with adaptability and efficiency [18]. The incorporation of reward shaping allows the integration of prior knowledge about the task domain, such as kinematic constraints, safety considerations, or task-specific objectives, into the learning process. This enables the robot to more effectively generalize its learned behaviors across different scenarios and environments.

Another state-of-the-art technique for improving convergence speed and generalization of agents that is noteworthy is Physics Informed Reinforcement Learning (PIRL). PIRL has gained recognition as an effective method for integrating physical knowledge into RL systems, leading to more robust and dependable results in various fields [19,20]. In the field of robotics, PIRL has been utilized to ensure that control policies comply with physical constraints, including joint limits and energy efficiency, thereby producing more stable and realistic robotic behaviors [21].

This research paper, therefore, focuses on developing and utilizing reward-shaping techniques for controlling a 4-degree of freedom (DOF) robot, aiming to enhance the learning efficiency and effectiveness of the RL agent. The 4-DOF robot is a versatile platform for exploring various manipulation tasks. By introducing physics-informed learning into the agent architecture, it will further improve the performance of the agent. While the RS technique promotes the exploration of actions during training, avoiding local minima and improving convergence speed, therein, an increase in the convergence is further guaranteed by making the agent physics aware. The major contributions of this paper are summarized as follows:

- **Development of a physics-informed RL agent:** A modified DDPG framework is proposed, integrating prior knowledge of the manipulator's dynamics to improve convergence speed and learning efficiency.
- **Lyapunov-based reward shaping:** A stability-guaranteed reward formulation is introduced to guide the training process, ensuring both convergence and control optimality.
- **Adaptive exploration mechanism:** A novel adaptive noise modulation technique is designed to enhance the exploration-exploitation tradeoff, enabling robust training in uncertain and dynamic environments.
- **Validation on a 4-DOF robotic platform:** The effectiveness of the proposed approach is demonstrated through simulation and real-world experiments, showing a 20–30% improvement in tracking performance and up to 3× faster convergence compared to standard DDPG.
- **Demonstration of generalizability:** The trained agent successfully adapts to unforeseen conditions without explicit training in such scenarios, confirming its robustness and generalization capabilities.

The rest of the paper is structured as follows: Section 2 presents the related works for this study. Section 3 discusses the background, such as the robot model description, as well as general RL. Section 4 presents the technical background of our PIRS-DDPG control framework, where we discuss the PIRS agent, reward shaping mechanisms, and proving the stability of the system. It also discusses the adaptive noise model incorporated in our study and the reward function design itself. Section 5 delves into the details of hyperparameter tuning and network structure for our specific application, followed by Section 6, which highlights the results and discussion for the simulation and experimental cases. Finally, concluding remarks appear in Section 7.

2. Related works

Reinforcement Learning (RL) has emerged as a powerful approach for controlling manipulator robots, particularly in scenarios where traditional model-based control techniques face limitations due to complex, nonlinear dynamics or unpredictable contact interactions. Unlike classical methods, RL enables robots to learn optimal control policies through interaction with the environment, making it well-suited for tasks involving high degrees of freedom and continuous action spaces—a trend evident in recent surveys of robotic manipulation using deep RL [22,23]. RL is especially advantageous for robotic manipulation because it does not require precise system models, allowing for robust, model-free control in environments with uncertainty and variability, as demonstrated in real-robot trajectory tracking on Baxter arms via deep RL with reference corrections [24]. Moreover, recent advances in deep RL have enabled learning end-to-end control policies from raw sensory inputs, including motion planning for serial manipulators using DDPG with hindsight experience replay [25]. Additionally, RL facilitates generalization and skill transfer, highlighted in surveys addressing real-world deployment challenges and achievements [23]. These methodologies have been successfully applied across tasks ranging from pick-and-place with 6-DOF industrial arms [26] to complex, contact-rich manipulation and deformable object handling, underscoring RL's potential to revolutionize robot control in dynamic, uncertain, and interactive contexts.

Despite the remarkable progress RL algorithms in robotic manipulation, these methods often face significant challenges when applied to real-world environments. Algorithms like DDPG, SAC, and PPO require extensive exploration and suffer from sample inefficiency, particularly in scenarios with sparse, delayed, or binary rewards—common in manipulation tasks where success is only indicated upon task completion. To address these limitations, reward shaping has emerged as a powerful strategy for guiding the learning process. Recent research has shown that carefully designed shaping functions, in [27], RS was applied to mobile robot navigation to mitigate the risk of the agent getting trapped in local minima. The results indicated that RS improved the agent's decision-making capabilities by providing more informative feedback during learning. Similarly, [28] proposed a dynamic reward shaping technique for Unmanned Surface Vehicle (USV) tracking, where a dynamic threshold was defined to adapt the reward function in real-time. This approach accelerated training while preserving control accuracy, although its effectiveness was only validated on low-dimensional systems, raising concerns about scalability to more complex tasks. In [29], RS was integrated into a DDPG-based motion planning framework, where the reward function was dynamically updated using online knowledge of the environment's map. This approach not only improved convergence speed but also enhanced the agent's adaptability to previously unseen states. Likewise, [30] employed RS to satisfy specific control constraints, highlighting the flexibility of shaping functions in incorporating domain-specific requirements. However, these studies also emphasize a critical challenge: manual reward design, while seemingly straightforward, can inadvertently lead to undesirable behaviors if the shaped rewards misalign with the true objective. As discussed in [31], constructing effective multi-level reward functions requires deep domain expertise and meticulous tuning, which limits their general applicability and transferability. Work has also been done using reward-shaping in the scope of tracking control. In [32], the authors used a Lyapunov-based reward shaping technique for the tracking control and vibration suppression of a rotary flexible robot. Other than this work and one presented in [30], there is very little literature published in the field of control and robotics utilizing reward shaping. With continuous control, policy convergence becomes a significantly difficult task; hence, by

Table 1
Summary of recent advances and challenges in Reinforcement Learning for Robotic Manipulation.

Theme	Description	Representative Work(s)	Key Takeaways
General Advantages of RL	Model-free control suitable for complex, high-DOF systems	[22–24]	Effective in uncertain and nonlinear environments; handles continuous actions; enables learning from raw sensory data
End-to-End Learning	Direct policy learning from sensors to actions	[25]	Supports motion planning and real-time adaptation in manipulation tasks
Skill Transfer & Generalization	Reusability of learned behaviors across tasks	[23]	Critical for real-world deployment and multi-task learning
Industrial & Complex Tasks	Application in diverse manipulation settings	[26]	Applied to pick-and-place, deformable objects, and contact-rich environments
Reward Shaping (RS)	Guides learning by refining reward signals	[27–30]	Improves convergence and task performance, but depends on expert design
RS in Control Applications	RS for constraint satisfaction and stability	[30,32]	Limited research; useful in vibration suppression and trajectory tracking
RS Limitations	Manual tuning can mislead learning	[31]	Designing effective reward functions is domain-specific and error-prone
Adaptive Noise for Exploration	Dynamic exploration strategies	[33–35]	Balances exploration/exploitation; improves sample efficiency and behavior diversity
Learning Noise Schedules	Exploration based on agent confidence	[36,37]	Enables uncertainty-driven exploration; enhances performance in real-world tasks
Physics-Informed RL (PI-RL)	Embedding physical models into RL	[38–41]	Improves safety, learning speed, and generalization by enforcing physical consistency

employing RS techniques, convergence speed and accuracy of desired behavior can be improved.

While standard RL algorithms often rely on fixed stochastic policies or manually tuned Gaussian noise for exploration, these methods may either under-explore or introduce excessive randomness that hinders policy convergence. To address this, adaptive noise mechanisms have been proposed, enabling the exploration strategy to evolve in response to the agent's learning progress. For instance, [33] initially proposed parameter space noise for exploration, laying the groundwork for adaptive approaches. More recent works have extended this idea. In [34], the authors introduce CEM-RL, combining evolutionary strategies with actor-critic methods where exploration is guided by adaptive noise sampled from a population distribution, enabling more efficient policy search. Similarly, [35] demonstrates the benefits of task-agnostic, unsupervised exploration using structured noise to discover diverse behaviors before fine-tuning on a downstream task, improving sample efficiency and generalization. Another promising direction is learning noise schedules conditioned on agent performance. In [36], the authors propose an adaptive exploration strategy that dynamically adjusts noise based on epistemic uncertainty estimates derived from a Bayesian ensemble, allowing the agent to explore more when uncertainty is high and exploit when confident. Likewise, in robotic manipulation, [37] incorporates adaptive action noise into their RL framework to better balance exploration and exploitation during training, showing that this improves both sample efficiency and final task performance on real-world manipulation benchmarks.

Physics-Informed Reinforcement Learning (PI-RL) is an emerging paradigm that integrates prior physical knowledge into RL algorithms to enhance learning efficiency, improve generalization, and ensure safer behavior, especially in robotics and control applications. Traditional RL methods often rely on purely data-driven interactions, which can be sample-inefficient and risky, particularly in real-world systems where safety and stability are critical. By embedding physical models, constraints, or principles such as conservation laws, dynamics equations, or energy bounds into the RL framework, PI-RL helps guide the learning process, reducing exploration in unsafe or physically implausible regions of the state space. Recent work has demonstrated the effectiveness of PI-RL in various robotic domains. For example, in [38], the authors provide a comprehensive survey of model-based and physics-informed RL techniques, highlighting how integrating dynamics priors reduces sample complexity. In [39], a model-based PI-RL framework is developed that incorporates physical laws into the transition model to improve policy learning in dynamic systems. Similarly, [40] proposes an adaptive PI-RL approach for robotic manipulation that adjusts the influence of physics-based priors during training, showing improved robustness and adaptability in dynamic environments. Another notable application is presented in [41], where PI-RL is used to guide bipedal robot locomotion by encoding biomechanical constraints and symmetries, which significantly improve stability and gait generalization.

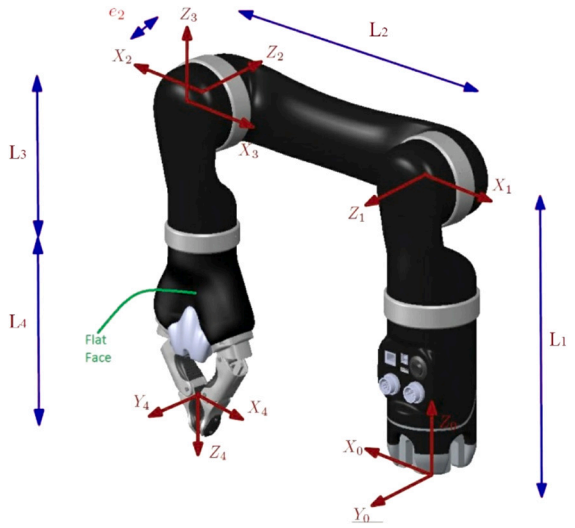


Fig. 1. Schematic of the 4-DOF robot manipulator.

Table 1 provides a structured overview of recent research in reinforcement learning for robotic manipulation, grouped by key themes such as end-to-end learning, skill generalization, reward shaping, and physics-informed approaches. It highlights representative works and summarizes their main contributions and insights.

3. Background

3.1. Robot model description

For the validation of the control objective, a 4 DOF robot manipulator is taken into consideration. This manipulator robot was used to test a number of research control systems. The robot manipulator consists of four rotary joints connected in series via links and an end-effector tool. Fig. 1 illustrates the structure of the 4-DOF robot manipulator. To effectively operate the robot, first, the forward kinematics (FK) and inverse kinematics (IK) models are computed using the Denavitt-Hartenberg (DH) parameters. The desired reference path is determined within the

workspace and is then used in the IK model to calculate the desired joint angles. The controller aims to guarantee precise and seamless tracking control of the desired trajectory, minimizing tracking errors. Finally, the FK model converts the joint space positions to determine the robot trajectory in the workspace. The robot's homogeneous transformation matrix must be known in order to calculate the end-effector position $P = [xyz]^T$ with respect to the base's coordinates, as shown in Eqn. (1).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (-c_1c_2s_3 - c_1c_3s_2)L_{34} + c_1c_2L_2 \\ (-s_1c_2s_3 - s_1c_3s_2)L_{34} + s_1c_2L_2 \\ (-c_2c_3 - s_2s_3)L_{34} - s_2L_2 + L_1 \end{bmatrix} \tag{1}$$

where $c_i = \cos(\theta_i), s_i = \sin(\theta_i)$, and L_i is the link length.

The IK model, solved using geometric equations, provides multiple solutions of the possible joint angles that the robot can be in to attain the desired workspace position determined previously. A potential solution for the desired position obtained by applying the IK model is given by Eqn. (2)

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{y}{x}\right) \\ -(\beta + \eta) \\ \sin^{-1}\left(\frac{L_2^2+(L_3+L_4)^2-(x^2+y^2+(z-L_1)^2)}{2L_2(L_{34})}\right) \end{bmatrix} \tag{2}$$

where:

$$\beta = tan^{-1}\left(\frac{z - L_1}{\sqrt{x^2 + y^2}}\right); \tag{3}$$

$$\eta = cos^{-1}\left(\frac{R^2 + L_2^2 - L_{34}^2}{2L_2R}\right) \tag{4}$$

with $R = \sqrt{r^2 + (z - L_1)^2}$, and $r = L_2c_2 - L_{34}s_{23}$.

Since the joint q_4 affects only the orientation of the end effector and not its position, it is not included in the solution of the IK model.

While the kinematics of the robot is important to understand its positional movement, the dynamics are the main focus when dealing with the input and output of the robot for control applications. In this case, the robot's dynamic model is computed using the Lagrangian approach. The dynamic model is illustrated in Eqn. (5).

$$u = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta) \tag{5}$$

where $M(\theta) \in \mathbb{R}^{n\times n}$ is the mass and inertia matrix of the robot, $V(\theta, \dot{\theta}) \in \mathbb{R}^{n\times 1}$ is the Coriolis and centripetal vector, and $G(\theta) \in \mathbb{R}^{n\times 1}$ is the gravity vector. $u \in \mathbb{R}^{n\times 1}$ is the input torque vector, $\theta \in \mathbb{R}^{n\times 1}$ represents the position vector, $\dot{\theta} \in \mathbb{R}^{n\times 1}$ is the velocity vector, and $\ddot{\theta} \in \mathbb{R}^{n\times 1}$ is the acceleration vector at time t. Similarly, $\theta_d \in \mathbb{R}^{n\times 1}$ represents the desired position vector, $\dot{\theta}_d \in \mathbb{R}^{n\times 1}$ is the desired velocity vector, and $\ddot{\theta}_d \in \mathbb{R}^{n\times 1}$ is the desired acceleration vector. For 4-DOF robots, $n = 4$ is the degree of freedom of the robot.

3.2. Reinforcement learning

The tracking control problem can be framed as an infinite-horizon discounted Markov Decision Process (MDP), assuming the environment adheres to the Markov property, where future rewards and states depend solely on the present state. The MDP is represented as the 5-tuple $< S, A, P, R, \gamma >$, where S denotes the set of states $s \in S$, A represents the set of actions $a \in A$, T symbolizes the transition probability function $T : S \times A \times S \rightarrow [0, 1]$, determining the probability of transitioning to a successor state given an action a, R signifies the reward function, and $\gamma \in [0,1]$ indicates the discount factor. RL aims to optimize a policy $\pi : O \rightarrow A$ maximizing the expected return R to solve the MDP. This policy defines the agent's behavior, mapping the suitable action to a given state. To iteratively learn the optimal policy, we determine the action-value function $Q(s, a)$. This function quantifies the value of a state given an action, facilitating the derivation of the optimal policy. Q-learning embodies this process, as depicted in Eqn. (6).

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) max_{a'} Q(s', a') \tag{6}$$

Here, s denotes the current state, a represents the action taken to reach that state, and Q signifies the action-value, which is the sum of the immediate reward R and the discounted expected reward, where a' and s' denote the subsequent action and state.

Despite its foundational importance, RL has garnered significant interest in control engineering, particularly for dealing with continuous action spaces. Q-learning employs a tabular database to store previously discussed Q-values. However, for complex applications, using algorithms like Q-learning becomes computationally burdensome. Newer algorithms, such as Deep Deterministic Policy Gradient (DDPG), have been developed to handle large datasets and continuous tasks.

This paper predominantly utilizes the DDPG algorithm, an off-policy actor-critic-based algorithm. It leverages the Q-function to evaluate the policy, known as the critic, and policy gradient, along with other techniques to determine the optimal policy, also known as the actor. The optimal Q-function is determined by minimizing the Mean-Squared Bellman Error (MSBE) loss function. After learning the Q-function, assuming its differentiability with respect to time, gradient ascent is performed to solve:

$$\max_{\theta} E_{s\ D}[Q_{\phi}(s, \mu_{\theta}(s))] \tag{7}$$

Here, $\mu_{\theta}(s)$ represents the deterministic policy, and ϕ denotes the parameter of the Q-function, treated as a constant in the gradient ascent process.

4. PIRS-DDPG control framework

The generalized diagram of the control scheme proposed for the training phase is depicted in Fig. 2. The PIRS agent acts as the decision maker, taking in different inputs in each episode iteration, the observations of the robot, along with samples from the replay buffer $\mathcal{D}$, the Lyapunov-shaped reward received based on the actions taken by the learning agent, and finally, the $isDone$ signal that indicates either that the robot is out of bounds from the safety constraints set, or that the robot has reached the end of the tracking trajectory signifying the end of the episode.

4.1. Physics informed RL

Physics Informed Machine Learning (PIML) has quickly become a prominent field of ML research. It is a novel technique that encourages inclusion of mathematical physics models of the systems into the observational or domain space of the ML models. Not only can PIML provide fast convergence, it also uses physical laws specific to the task to improve the interpretability of the ML model, sample efficiency, as well as generalization for better predictions [19,42]. Since the learning mechanism in traditional ML approaches is often highly dependent on large amounts of data, which may not often be readily available, sample efficiency is achieved by incorporating guiding equations and physical laws. In the same sense, generalization is made easier due to the inclusion of physical information [20].

In order to apply the principles of PIML, several techniques have been established that work by incorporating various types of bias into the learning process. One prominent technique, known as Physics-Informed Neural Networks (PINN), includes guiding equations and soft physical laws directly into the neural network model's loss function, using the learning bias [43]. This type of bias helps align the model with the underlying physics of the problem, reducing the reliance on large datasets and, therefore, improving generalization capabilities. Another prominent technique of PIML includes the use of observational

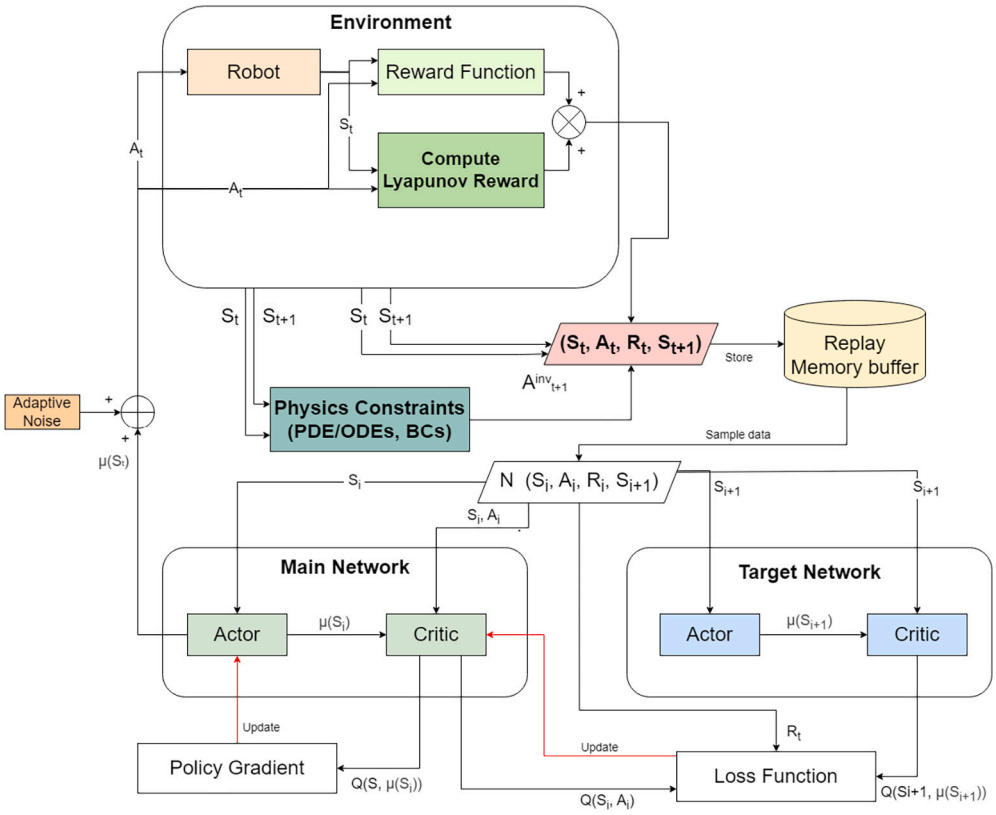


Fig. 2. PIRS-agent based learning architecture.

bias. This method uses data from various sources that reflect the physical laws guiding its generation and trains the NN models based on these data [44,45]. The data may be sourced from a simulation environment, physics-based simulations, mappings, or just real-world observations. Inductive bias is introduced in methods like Hamiltonian Neural Networks (HNNs) and Lagrangian Neural Networks (LNNs), where the model architecture is designed to reflect the underlying physical laws, such as the conservation of energy or momentum, directly within the neural network structure [46,47].

RL is particularly well suited for the PIML strategy because it excels in learning optimal policies through interaction with an environment. Through this integration, the model will be improved in terms of generality and optimality. For example, the RL agent can learn policies by incorporating physical constraints or dynamics into its reward function or transition dynamics. One well-known method that uses this strategy is observational bias-based action regulation. In action regulation, the observational bias is fed into the system to incorporate knowledge of the model and, hence, yield an action that is better suited for the system.

The way this works is that the current state s_t, the next state s_{t+1}, and the resulting episode reward r_t along with the physics-informed structure generated action, a_t is stored in the synthesized replay buffer $\mathcal{D}$ as the tuple $< s_t, a_t^s, s_{t+1}, r_t^s >$ for the agent to access in batches throughout the training procedure. This way, the generated physics-informed actions are injected into the agent's training as samples, guiding the agent to learn from these good action-state pairs. In our proposed application, we leverage the availability of the dynamic model of the robot and perform inverse-dynamics techniques to attain the synthesized reference torque that controls the system. The replay buffer $\mathcal{D}$ is updated with these synthetic data, which are later accessed by the agent during training, allowing us to increase the sample efficiency and improve the convergence speed drastically.

4.2. Reward shaping

The premise of shaping, which comes from behavioral psychology, is to provide the learning agent with a sequence of relatively simple challenges that lead to the more difficult problem of the final interest. The term originates from psychology, wherein this behavior is observed in training animals to perform tricks, etc., and human education is also built up as a process of molding if the behavior is regarded to include "understanding." To apply shaping in reality, one needs to understand more about the problem than merely the criteria that result in an absolute good or poor state. Generally, a shaped reward may take up the following form:

$$R' = R + F \tag{8}$$

with $R : S \times A \times S \to \mathbb{R}$ the expected reward function of the original MDP, which we augment by the expected reward shaping function $F : S \times A \times S \to \mathbb{R}$, resulting in the modified expected reward function $R_\delta : S \times A \times S \to \mathbb{R}$. We now take R_δ into account and optimize the policy that maximizes the discounted sum of the adjusted rewards R_δ, rather than learning a policy based on R.

4.2.1. Lyapunov function-based reward shaping

Potential-based reward shaping [48] is a technique used in reinforcement learning to accelerate learning by adding additional reward signals to guide the agent toward the desired behavior. It works by defining a potential function that quantifies how "good" or "bad" a state is perceived to be. The potential function is then used to create shaping rewards, which are added to the original reward signal provided by the environment [49]. Eqn. (9) highlights the PBRS technique.

$$R'(s, a, s') = R(s, a, s') + \rho\phi(s') - \phi(s) \tag{9}$$

Following the general format of the reward shaping equation, where ϕ is the to-be-defined potential function, which leads to optimality and an invariant optimal policy as verified using Bellman's equations [32]. The convergence speed may be reduced further by initializing the potential function to determine the optimal policy. However, in the case of real applications with high dimensions due to complex dynamics, obtaining an optimal policy in itself becomes challenging. Furthermore, the potential function being a function of states only rather than a state-action pair imposes a limitation on the shaping mechanism. Addressing this pitfall, [50] further widened the scope of PBRS by making the reward shaping function a function of both state and action pairs shown in Eqn. (10). Although optimality is preserved and validated through Bellman's equations, the resulting policy is no longer equal to the previous greedy policy because it is biased $\phi(s,a)$.

$$R'(s,a,s') = R(s,a,s',a') + \rho\phi(s',a') - \phi(s,a) \tag{10}$$

Following the successes of the PBRS method, [51] presented a Lyapunov-based reward shaping method that follows suit with the aforementioned methods by guaranteeing policy convergence and preserving optimality while ensuring that the optimal greedy policy obtained remains unbiased. The proposed shaping equation can be seen in Eqn. (11).

$$R^L(s',a') = R(s,a) + \lambda(\gamma R(s',a') - R(s,a)) \tag{11}$$

where λ is the tuning factor deciding the weight of the shaping term, and γ is the discount factor. Other methods of tuning the weight for shaping the reward have also been proposed [52].

4.2.2. Stability analysis

Given that the nature of RL is to seek out the optimal policy for a given task, MDPs can, therefore, be considered optimal control tasks. The Lyapunov-based RS method may be used to ensure the stability of the said control task and facilitate the convergence of the optimal policy. By equating the Lyapunov function $V(s,a)$ to be proportional to the negative of the reward function, i.e., $V(s,a) = -R(s,a)$, stability can be proved.

Theorem 1. *Consider (s^*,a^*) as the state-action pair with the highest return of rewards and the Lyapunov function $V(s,a)$. If the following condition is satisfied:*

$$V(s',a') - V(s,a) \leq 0, \tag{12}$$

then the state-action pair (s,a) will converge to the optimal state-action pair (s^*,a^*).

Proof. By analyzing the above condition and Eqn. (11), we observe that the weighted reward term in the Lyapunov-based reward shaping method guarantees that this condition holds. Specifically, minimizing $V(s',a') - V(s,a)$ is equivalent to maximizing the reward difference $R(s',a') - R(s,a)$. Furthermore, the monotonically decreasing property of the Lyapunov function ensures that the state-action pair (s,a) converges to the point of maximum reward (s^*,a^*), which facilitates the agent's learning of the optimal policy. □

Theorem 1 establishes the framework for guiding an agent in an MDP by ensuring that the condition in Eqn. (12) is satisfied through a suitable reward shaping strategy. We introduce a shaped reward term, as defined in Eqn. (11), that encourages the agent to maximize the difference $R(s',a') - R(s,a)$. This maximization is equivalent to minimizing $V(s',a') - V(s,a)$, which helps to ensure that the condition in Eqn. (12) is satisfied as often as possible. This alignment of reward shaping with the minimization of the Lyapunov function difference provides a robust foundation for proving the convergence of the Q-learning procedure when employing Lyapunov function-based RS.

Furthermore, given the Lyapunov-shaped reward R^L as defined in Eqn. (11), it follows that reward shaping maintains optimality. For the original MDP M with $\gamma = 1$, the Bellman equation can be expressed as:

$$Q^*_M(s,a) = \mathbb{E}_{s\sim T(s|s,a)}\left[R(s,a) + V^*_M(s')\right],$$

where $Q^*_M(s,a)$ and $V^*_M(s')$ represent the optimal Q value function and the value function, respectively, under MDP M. We can derive the following relationship:

$$Q^*_M(s,a) = Q^*_M(s,a) - \lambda R(s,a),$$

indicating that $Q^*_M(s,a)$ also satisfies the Bellman equation.

4.3. Adaptive noise for exploration

One way to realize reward space noise is by incorporating reward shaping. It can be done simultaneously within the actor-critic architecture to include stochasticity in the action and network parameter space. Standard Ornstein-Uhlenbeck (OU) noise is introduced in the action space to induce the first exploration stage. The OU process U_t, vastly used to introduce environmental disturbance in machine learning schemes, can be described as a continuous-time stochastic process with the following stochastic differential equation (SDE):

$$dU_t = \alpha(\mu - U_t)dt + \sigma dW_t \tag{13}$$

where μ is the long-term mean, $\alpha \geq 0$ is the rate of mean reversion. W_t is the Weiner process based on Brownian motion where $\sigma > 0$ denotes the diffusion coefficient or the scale of noise of the process.

The second stage of exploration is done by integrating adaptive noise into the actor-critic network parameters themselves. Given that the OU noise model is used to update the network parameters θ of the policy ϕ, the update at each step will be as follows:

$$\theta^{\hat{\phi}} = \theta^{\phi} + \eta \tag{14}$$

where

$$\eta_{t+1} = \eta_t + \alpha(\mu - \eta_t) + \sigma \cdot N(0,I) \tag{15}$$

η_t is the current noise value, and as previously mentioned, α is the mean reversion rate, σ is the scale of noise, μ is the mean, and $N(0,I)$ is a standard Gaussian noise vector with a mean of 0 and variance of 1. As neural networks are updated iteratively, the agent's learning impact is more responsive to variations σ. Therefore, σ it is designed to be a time-varying value correlated with the variance of the action space it produces to fully utilize the performance of the noise of the parameter space. The update is, therefore defined as follows:

$$\sigma_{i+1} = \begin{cases} 1.02 \cdot \sigma_i, & d < \delta \\ \sigma_i / 1.02, & d \geq \delta \end{cases}$$

d denotes the distance between the strategy that includes parameter noise and the one that does not. It can be noted that the action space of the agent may be limited in cases where d it falls below the threshold δ. Therefore, to expand the action space and hence promote better exploration, the value σ should be increased. On the other hand, if d exceeds the threshold δ, it suggests that the agent's action space is extensive, and the neural network's learning may become unstable. To ensure the exploration process remains stable, the value σ should be decreased. The value of d can be adjusted by adjusting the difference in rewards as shown in Eqn. (16):

$$d = r(s',a') - r(s,a) \tag{16}$$

The threshold δ on the other hand is a constant value.

4.4. Reward function design

The reward function was designed through domain knowledge as well as trial and error. A major part of the contribution of this paper lies in the design and manipulation of the reward function. The reward function was primarily designed to be a function of tracking errors.

$$r_{pos} = x + y + z * t_{sim} \tag{17}$$

The values of x are defined as

$$x = \begin{cases} x_{pos} & \text{if } e_{deg}^2 < e_{th1} \\ 0 & \text{otherwise} \end{cases}$$

and the value of y is defined as

$$y = \begin{cases} y_{pos} & \text{if } |e_{rad}| \leq e_{th2} \\ 0 & \text{otherwise} \end{cases}$$

where

$$e = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}^T$$

are the tracking errors for joints 1, 2, and 3. z is a constant gain equal to 0.05 that is multiplied with the simulation time t_{sim} to ensure prolonged episodes and avoid early termination. The value of x is defined based on e_{th1} which is the threshold value equal to 4°, if the threshold is met x equals x_{pos}, which is 1.2, and 0 otherwise. y is defined based on the threshold value e_{th2}, and is equal to $y_{pos} = 0.1$, once the threshold is met, and 0 otherwise. The reward at each episode consists of the positive reward r_{pos} (17) described earlier as well as the negative counterpart r_{neg} (18).

$$r_{neg} = \zeta \sum (e_{rad})^2 \tag{18}$$

where ζ is a constant gain set -4 to penalize the agent for selecting actions that result in high error values. The total reward for each timestep is (19).

$$R = r_{pos} + r_{neg} \tag{19}$$

The Lyapunov-based reward shaping is then formulated. The immediate reward is subtracted from the reward received from the new states, presenting the shaping component. The future state reward is cascaded with a constant K, which is the discount factor, equal to 0.99. The shaping component is then combined with the immediate reward to produce the Lyapunov-based shaped reward R^L. It follows the same architecture as discussed in Eqn. (11) and the graphical illustration of it is also noted in Fig. 2.

The proposed control technique uses the continuous-state and action-space-based DDPG as its off-policy RL agent. This algorithm uses an experience replay buffer, which allows the agent to reuse state action and reward pairs from earlier episodes, allowing it to make use of ideal actions regardless of the current policy.

5. Tracking control using reward-shaped RL

The state space consists of 12 states, and the action space consists of 3 actions, the torque efforts for the 3 joints; as previously mentioned, joint 4 does not affect the position of the end-effector and therefore the joint is input with no control effort. The observation states for the training of the RL environment were the 3 joint angles $[\theta_1, \theta_2, \theta_3]$, their derivatives $[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}]$, the tracking errors $[e_1, e_2, e_3]$, and their derivatives $[\dot{e_1}, \dot{e_2}, \dot{e_3}]$. Consequently, the action space consisted of the control efforts $[u_1, u_2, u_3]$.

This approach integrates a physics-informed model with a Deep Deterministic Policy Gradient (DDPG) algorithm to improve robotic manipulation tasks. By incorporating an inverse dynamic model, which calculates the control actions based on a robot's current and next states, we ensure that the actions generated by the agent are physically feasible and adhere to the system's dynamics. This allows the DDPG algorithm to operate within the physical constraints of the robot manipulator, resulting in a more accurate and reliable action selection during training.

Algorithm 1 DDPG with Physics-Informed Inverse Dynamic Model and Lyapunov Function-Based Reward Reshaping.

Initialize critic network $Q(s, a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ
Initialize target networks Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathcal{D}$
Define the physics-informed inverse dynamic model $f_{\text{inv}}(s_t, s_{t+1})$ based on the manipulator's dynamic equations
Define Lyapunov function $V(s_t)$ for reward reshaping, ensuring system stability
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$
    Execute action a_t, observe reward $R(s_t, a_t)$ and next state s_{t+1}
    Use physics-informed inverse dynamic model $f_{\text{inv}}(s_t, s_{t+1})$ to compute the required action a_{t+1}^{inv}
    Compute the Lyapunov function $V(s_t)$ and the difference: $\Delta V = V(s_t) - V(s_{t+1})$
    Compute the Lyapunov-based reshaped reward:

$$R_{\text{lyap}}(s_t, a_t) = \text{sign}(\Delta V) \cdot |\Delta V|$$

    Compute the total reshaped reward:

$$R_{\text{total}}(s_t, a_t) = R(s_t, a_t) + \lambda R_{\text{lyap}}(s_t, a_t)$$

    Store transition $(s_t, a_{t+1}^{\text{inv}}, R_{\text{total}}(s_t, a_t), s_{t+1})$ in the replay buffer $\mathcal{D}$
    Sample a mini-batch of N transitions $(s_i, a_i^{\text{inv}}, R_{\text{total}}(s_i, a_i), s_{i+1})$ from $\mathcal{D}$
    Set the target: $y_i = R_{\text{total}}(s_i, a_i) + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update the critic by minimizing the loss: $L(\theta^Q) = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i^{\text{inv}}|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
  end for
end for

In addition, we reshape the reward function using a Lyapunov function-based approach to enforce stability throughout the learning process. A Lyapunov function measures the stability of the system, and by incorporating this into the reward structure, we ensure that the robot's movements remain stable while maximizing task performance. This reshaped reward combines the original task objective with a penalty or bonus term related to the system's stability, promoting safer and more stable control policies.

By combining the inverse dynamic model with Lyapunov-based reward reshaping, the algorithm ensures that actions taken are both optimal for task performance and compliant with physical and stability constraints. This hybrid approach results in a more robust learning framework that is particularly well-suited for controlling complex robotic systems like manipulators. The training was conducted using the robot dynamic model in the simulation environment. The proposed DDPG-based control mechanism was realized by designing a suitable actor and critic network for the training environment. The architecture of the actor and critic network can be seen in Fig. 3, where the actor-network takes the states in the input layer and gives out four action values, i.e., the torque values of each joint.

The input states are passed through two fully connected layers, each followed by ReLU activation function layers. The fully connected layers have 200 and 100 neurons, respectively. On the other hand, the critic

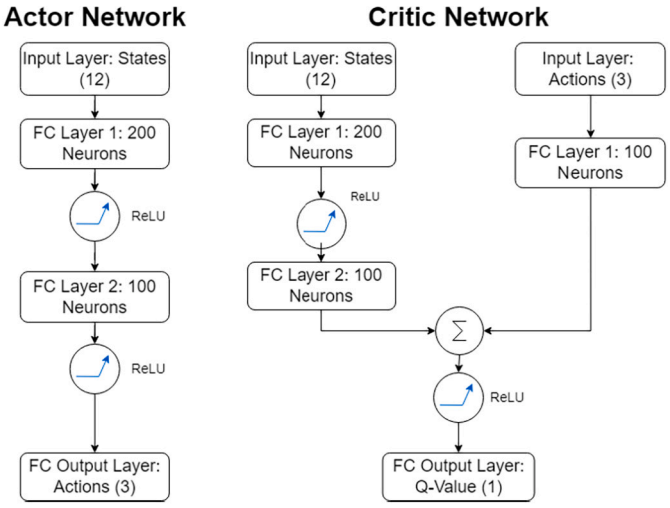


Fig. 3. The actor and critic network structure.

Table 2
Training Hyperparameters.

Discount Factor	0.99
OU Noise Standard Deviation	0.3
OU Noise Decay Rate	$1e^{-5}$
Actor Learning Rate	$1e^{-5}$
Critic Learning Rate	$1e^{-4}$
Experience Replay Buffer Size	$1e^{6}$
Minibatch Size	128

layer takes in both the states and actions, processing each of them individually before combining them. The state path of the critic network consists of the input layer followed by two fully connected layers of 200 and 100 neurons, with a ReLU activation layer connecting them. The action path starts with the actions as input layer, followed by a fully connected layer of 100 neurons. The two paths are then combined, followed by a ReLU activation layer, and finally, the fully connected output layer computes the Q-value. Hyperparameter tuning is critical in machine learning applications as it strongly influences algorithm convergence. An advanced learning algorithm combined with a well-structured reward function may not guarantee the convergence of a policy if the hyperparameters are poorly configured. Among many other parameters, the most influential are listed in Table 2.

The discount factor in continuous control tasks can often be set at a lower value to ensure optimal immediate state. However, this can lead to unstable learning and abrupt actions of the agent. To provide stable learning by prioritizing long-term rewards, the discount factor is set as 0.99. The noise model used was the Ornstein-Uhlenbeck (OU) noise model, with a standard deviation of 0.3, and decay rate of $1e^{-5}$. This causes constant low-level exploration throughout the training due to the low decay rate. The actor and critic learning rates were set with reference to working models and the minibatch size was set to 128, which is on the higher side to allow the agent to sample a larger batch of state-action-reward data from the experience replay buffer. To evaluate the training and sample efficiency of our proposed PIRS DDPG-based control agent, training was performed on a standalone DDPG agent for the same tracking task, as well as on an RS-DDPG agent. The training plot of the three, shown in Fig. 4, illustrates our proposed method's fast learning and converging ability. The episode rewards were normalized to the maximum attainable reward value, and as evident from the figure, our proposed agent reaches higher value rewards at a considerably earlier stage compared with the other two agents. It also managed to achieve the highest magnitude of reward and reach a steadier learning stage more quickly than the others.

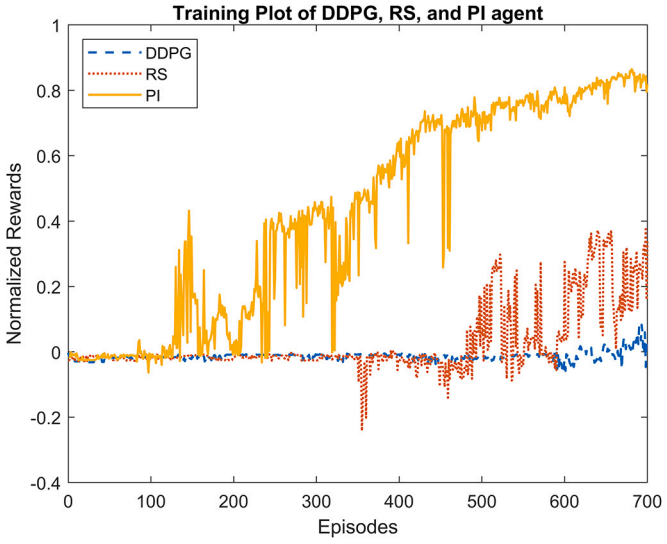


Fig. 4. Training plot of the Proposed technique PIRS-DDPG, RS-DDPG and DDPG.

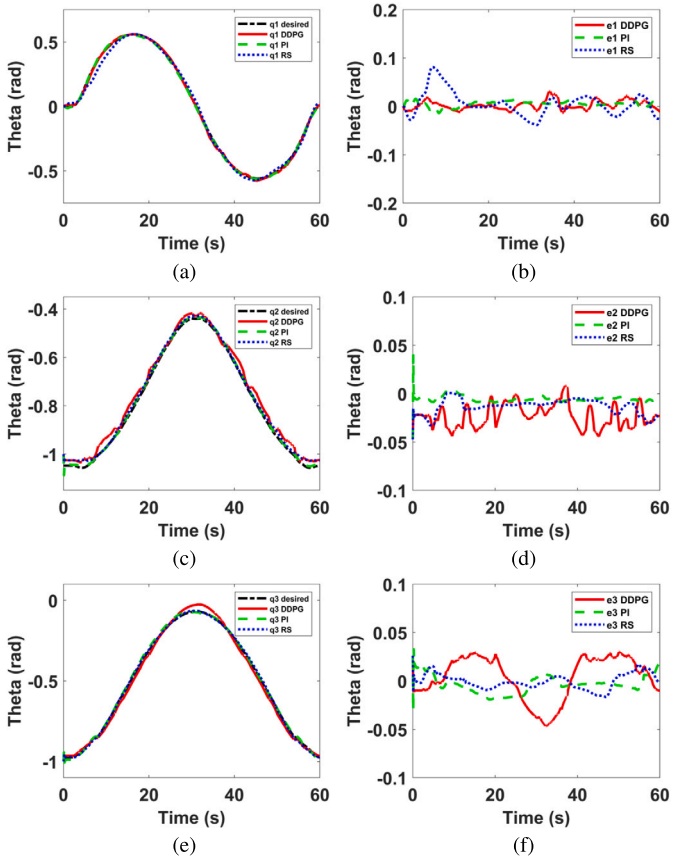


Fig. 5. Performance of DDPG-based tracking control in simulation. Subplots illustrate the tracking results for: (a) joint 1 (q_1), (b) tracking error for joint 1 ($error_1$), (c) joint 2 (q_2), (d) tracking error for joint 2 ($error_2$), (e) joint 3 (q_3), and (f) tracking error for joint 3 ($error_3$). Results are compared between the desired trajectory, DDPG, PI, and RS approaches.

6. Results & discussion

In this section, the performance of the proposed control strategy for the 4-DOF robot manipulator using an RL-based controller is evaluated through simulation and experimental results. The sim-to-real transfer of the controller has been proven to be effective as the controller train-

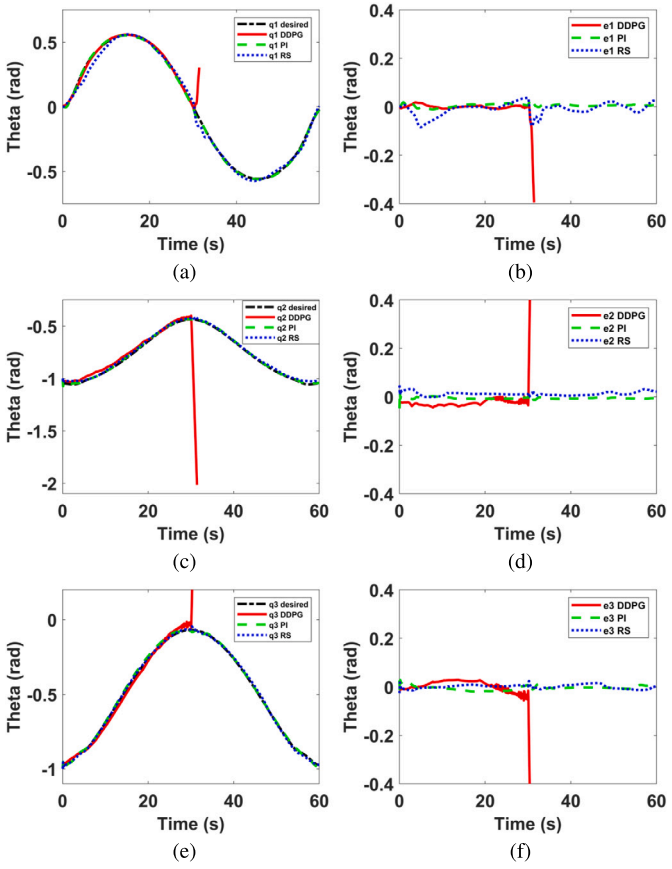


Fig. 6. Tracking and tracking errors in simulation with impulse disturbance at t = 30 s. (a) joint 1 tracking (b) joint 1 tracking error (c) joint 2 tracking (d) joint 2 tracking error (e) joint 3 tracking (f) joint 3 tracking error.

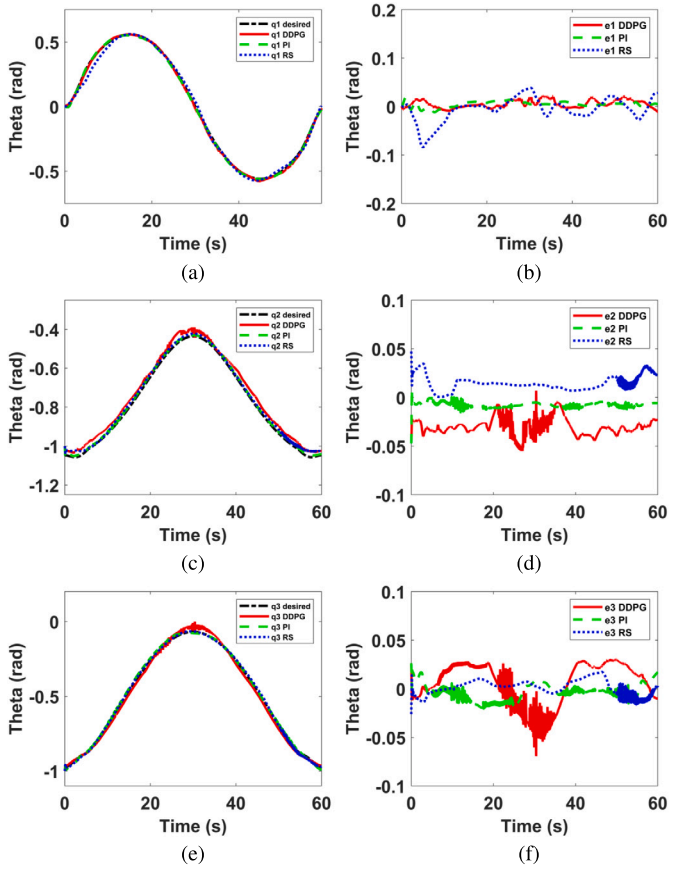


Fig. 7. Tracking and tracking errors in simulation with model uncertainties. (a) joint 1 tracking (b) joint 1 tracking error (c) joint 2 tracking (d) joint 2 tracking error (e) joint 3 tracking (f) joint 3 tracking error.

ing primarily took place in the simulation environment, and the control policy was transferred to the experimental environment. The training trajectory in the workspace was designed to be a simple circle.

6.1. Simulation results

After successful training, the control policy was extracted and evaluated first through simulation results to validate the efficiency and accuracy of the RL-based controllers. The results can be seen in Fig. 5, where the desired trajectory, the tracking trajectory, and the errors are presented.

From Fig. 5, it is evident that our proposed PIRS-DDPG agent surpasses the other two agents in terms of tracking. The errors seen for the PI agent mostly remain in the ± 0.02 range and depict a more steady change throughout the tracking of the entire trajectory in the joint space. For the RS agent, although joints 2 and 3 are comparable to our control agent, joint 1 sees a higher error value. As for the simple DDPG agent, the error value range is comparable to our agent, but the real trajectory depicts a higher fluctuation around the desired trajectory as seen in the errors presented in Figs. 5b, 5d and 5f.

Simulations were performed with a disturbance in the input signal and model uncertainties to further test the reliability of the trained agents. The root mean square errors (RMSE) for the 3 agents are presented in Table 3 where q1, q2, and q3 are joints 1, 2, and 3, respectively. We have four cases, stated below:

- case 1 - RMSE in simulation
- case 2 - RMSE in simulation with an impulse disturbance
- case 3 - RMSE in simulation with model uncertainties
- case 4 - RMSE in experiment

Table 3
Tracking Error Evaluation in Simulation.

		DDPG	PI	RS
Case 1	q1	0.0094	**0.0075**	0.0252
	q2	0.0252	**0.0065**	0.0152
	q3	0.0224	0.0098	**0.0077**
Case 2	q1	0.0379	**0.0083**	0.0274
	q2	0.1903	**0.0065**	0.0146
	q3	0.1984	0.0099	**0.0079**
Case 3	q1	0.0095	**0.0073**	0.0250
	q2	0.0316	**0.0083**	0.0160
	q3	0.0220	0.0093	**0.0078**
Case 4	q1	**0.0468**	0.0481	0.0690
	q2	0.0491	0.0386	**0.0380**
	q3	0.0595	0.0493	**0.0436**

From the table, it is clear that the performance of the simple DDPG-trained agent is easily outperformed by the PI and RS agents. For the simulation section, the PI-trained agent outperforms the other agents in 6 out of 9 cases. It is also noticeable that the magnitude of the RMSE resulting from the PI-trained agent is significantly smaller than that of the rest. In the cases where agents trained with RS perform better, RMSE values are still in a low range and are comparable to the best-performing value. The same can be seen in the experimental section, where the RMSE values for the PIRS-trained agent perform well overall.

The resulting performance of the three agents is also presented. Fig. 6 illustrates the simulation results with an impulse disturbance of an amplitude 100% of the input signal at time $t = 30$ s. As seen from the figure, the PIRS agent as well as the RS agent were able to withstand the distur-

Fig. 8. The Experimental Setup of the Robot.

bance well, however the DDPG agent becomes unstable. Moreover, from the error plots, it is clear that the tracking performed by the PI agent is superior to the rest for joints 1 and 2. As for joint 3, it is comparable to the RS agent. Fig. 7 illustrates the simulation results of the three agents with model uncertainty 15%. It is evident from the figure that the PIRS agent outperforms both agents for joints 1 and 2. While the RS agent follows second, leaving the DDPG agent to be the worst-performing agent. For joint 3, visually the performance for both RS and PIRS agents are similar, by looking into the RMSE we can note that RS agent has the edge, however, PIRS still exhibits excellent behavior. Additionally, the PI and RS-trained agents could withstand a higher amount of model uncertainty and higher disturbance before becoming unstable, which was not the case for the DDPG-trained agent. This shows the effectiveness of reward shaping and physics-informed techniques in terms of optimal policy convergence.

6.2. Experimental results

After observing that the agent was well-trained in the simulation environment, we moved on to the experimental validation. Physical experiments were performed on the 4 DOF robot manipulator, which was also used for training the policy. The experimental environment can be seen in Fig. 8.

To test sim-to-real transfer performance and evaluate the reality gap between simulation results and experimental results, the agent trained in the simulation was used without further retraining to gather the experimental results. This shows the efficiency of the controller in the perfect transfer of learned behavior, despite some differences that may exist between simulation and experiment. The experimental results obtained are presented in Fig. 9. As seen in the figure, the tracking observed in all three of the joints is remarkable, as the steady-state error settles to close to 0 radians for joints 2 and 3 presented in Fig. 9b and 9c. With joint 1, the error peaks up to around 0.12 radians at the initial stage attempting to reach the desired trajectory, however, once the transient region passes, the error in joint one also settles to around 0 radians shown in Fig. 9a. In all the cases, due to the nature of the robot, there exists a windup phenomenon at the beginning stage of the trajectory response, causing the tracking error to reach around 0.4 radian to reach the initial position in the desired trajectory.

The experimental validation was performed taking this into account and attempting to diminish this windup phenomenon as best as possible. The RMSE values of the tracking for the experimental scenario can be seen in Table 3, and it is evident from the results that the PIRS technique performs well in all the three joints. Additionally, the three-dimensional tracking plots are illustrated in Fig. 10d, 10e and 10f, which depict the circular trajectory in the workspace that was tracked by the agents for simulation and experimental results. It is evident from the figure that the tracking done by all agents was feasible, while the proposed technique achieved the same results within a considerably shorter training period. Torque efforts are also shown in Fig. 10. The torque signal can be seen to be stable and illustrates low chattering overall.

7. Conclusion

This study presents an advanced reinforcement learning (RL) framework for trajectory tracking control of a 4-DOF robotic manipulator, leveraging a Physics-Informed Deep Deterministic Policy Gradient (PI-DDPG) agent. The proposed approach combines three key innovations: (1) a physics-informed agent architecture that incorporates prior knowledge of system dynamics to enhance convergence speed; (2) a Lyapunov-based reward shaping (RS) strategy to ensure stability while preserving policy optimality; and (3) an adaptive noise generation model to promote robust exploration and improved performance under uncertainty. Simulation and experimental results validate the effectiveness of the proposed method, demonstrating improved tracking accuracy and sig-

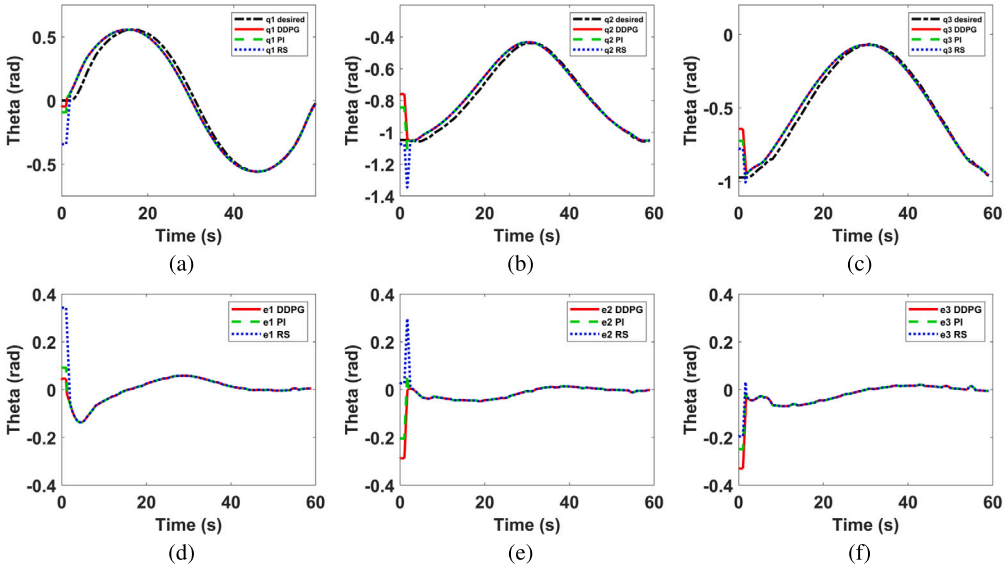


Fig. 9. Experimental Results of a tracking control task using PIRS-DDPG agent (a) joint 1 tracking (b) joint 1 tracking error (c) joint 2 tracking (d) joint 2 tracking error (e) joint 3 tracking (f) joint 3 tracking error.

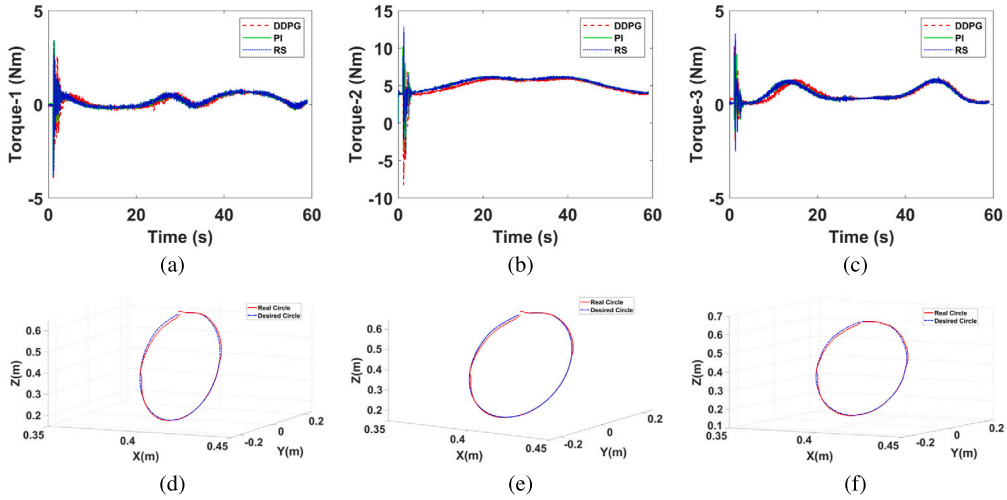


Fig. 10. Torque values of the 3 joints by the PI Agent and the three-dimensional reference tracking in the workspace (a) torque in joint 1 (b) torque in joint 2 (c) torque in joint 3 (d) 3D plot of PI-agent (e) 3D plot of RS-agent (f) 3D plot of DDPG-agent.

nificantly faster convergence compared to standard RL techniques. The agent also shows resilience to unexpected behaviors and environmental disturbances, even without direct exposure to such conditions during training. Despite these promising results, several limitations remain. The current approach relies on accurate modeling of system dynamics for the physics-informed component, which may not generalize well to highly complex or poorly understood systems. Additionally, real-time deployment in more dynamic and unstructured environments remains an open challenge. Future work will focus on extending this framework to high-DOF manipulators, incorporating online adaptation, and validating its effectiveness in real-world, human-in-the-loop robotic scenarios.

CRediT authorship contribution statement

Raouf Fareh: Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **Tanjulee Siddique:** Writing – original draft, Validation, Software, Data curation. **Kheireddine Choutri:** Writing – review & editing, Writing – original draft, Resources, Conceptualization. **Dmitry V. Dylov:** Writing – review & editing, Supervision, Methodology, Formal analysis.

Declaration of competing interest

The authors have declared no conflict of interest.

References

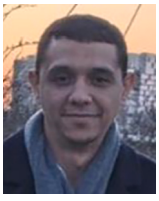
[1] Balanji HM, Turgut AE, Tunc LT. A novel vision-based calibration framework for industrial robotic manipulators. Robot Comput-Integr Manuf 2022;73:102248.
[2] Truong TN, Vo AT, Kang H-J. A backstepping global fast terminal sliding mode control for trajectory tracking control of industrial robotic manipulators. IEEE Access 2021;9:31921–31.
[3] Shin C, Ferguson PW, Pedram SA, Ma J, Dutson EP, Rosen J. Autonomous tissue manipulation via surgical robot using learning based model predictive control. In: 2019 international conference on robotics and automation. IEEE; 2019. p. 3875–81.
[4] Zhu J, Lyu L, Xu Y, Liang H, Zhang X, Ding H, et al. Intelligent soft surgical robots for next-generation minimally invasive surgery. Adv Intell Syst 2021;3(5):2100011.
[5] Verma V, Gupta A, Gupta M, Chauhan P. Performance estimation of computed torque control for surgical robot application. J Mech Eng Sci 2020;14(3):7017–28.
[6] Sreekanth N, Dinesan A, Nair AR, Udupa G, Tirumaladass V. Design of robotic manipulator for space applications. Mater Today Proc 2021;46:4962–70.
[7] Buckner C, Lampariello R. Tube-based model predictive control for the approach maneuver of a spacecraft to a free-tumbling target satellite. In: 2018 annual American control conference. IEEE; 2018. p. 5690–7.
[8] Mishra H, De Stefano M, Giordano AM, Lampariello R, Ott C. A geometric controller for fully-actuated robotic capture of a tumbling target. In: 2020 American control conference. IEEE; 2020. p. 2150–7.
[9] Kober J, Bagnell JA, Peters J. Reinforcement learning in robotics: a survey. Int J Robot Res 2013;32(11):1238–74.
[10] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR; 2018. p. 1861–70.
[11] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR; 2016. p. 1928–37.
[12] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint. arXiv:1707.06347, 2017.
[13] Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic policy gradient algorithms. In: International conference on machine learning. PMLR; 2014. p. 387–95.
[14] Al-Dahhan MRH, Ali MM. Path tracking control of a mobile robot using fuzzy logic. In: 2016 13th international multi-conference on systems, signals & devices. IEEE; 2016. p. 82–8.
[15] Xie Z, Lin Q. Reinforcement learning-based adaptive position control scheme for uncertain robotic manipulators with constrained angular position and angular velocity. Appl Sci 2023;13(3):1275.
[16] Huang B, Jin Y. Reward shaping in multiagent reinforcement learning for self-organizing systems in assembly tasks. Adv Eng Inform 2022;54:101800.
[17] Devlin S, Kudenko D, Grześ M. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. Adv Complex Syst 2011;14(02):251–78.
[18] Levine S, Finn C, Darrell T, Abbeel P. End-to-end training of deep visuomotor policies. J Mach Learn Res 2016;17(39):1–40.
[19] Kashinath K, Mustafa M, Albert A, Wu J, Jiang C, Esmaeilzadeh S, et al. Physics-informed machine learning: case studies for weather and climate modelling. Philos Trans R Soc A 2021;379(2194):20200093.
[20] Banerjee C, Nguyen K, Fookes C, Raissi M. A survey on physics informed reinforcement learning: review and open problems. arXiv preprint. arXiv:2309.01909, 2023.
[21] Polydoros AS, Nalpantidis L. Survey of model-based reinforcement learning: applications on robotics. J Intell Robot Syst 2017;86(2):153–73.
[22] Han Y, Liang X, Zhang K. Deep reinforcement learning for robotic manipulation: a comprehensive survey. IEEE Trans Neural Netw Learn Syst 2023.
[23] Tang C, Liu W, Zhao M. Real-world deployment of deep reinforcement learning in robotics: challenges and advances. Robot Auton Syst 2024.
[24] Pavlichenko N, Behnke S. Imitation learning and deep reinforcement learning for trajectory tracking on Baxter robot. In: European conference on mobile robots; 2022.
[25] Andrychowicz M, Baker B, Chociej M. Efficient motion planning for robotic manipulators using ddpg and hindsight experience replay. In: Autonomous intelligent robotics; 2023.
[26] Stoica R, Popescu A. Pick-and-place task learning for industrial manipulators using ddpg and lqr. Sustainability 2025.
[27] Miranda VR, Neto AA, Freitas GM, Mozelli LA. Generalization in deep reinforcement learning for robotic navigation by reward shaping. IEEE Trans Ind Electron 2023.
[28] Zhong W, Li H, Meng Y, Yang X, Feng Y, Ye H, et al. Usv path following controller based on ddpg with composite state-space and dynamic reward function. Ocean Eng 2022;266:112449.
[29] Botteghi N, Sirmacek B, Mustafa KA, Poel M, Stramigioli S. On reward shaping for mobile robot navigation: a reinforcement learning and slam based approach. arXiv preprint. arXiv:2002.04109, 2020.
[30] De Lellis F, Coraggio M, Russo G, Musolesi M, di Bernardo M. Guaranteeing control requirements via reward shaping in reinforcement learning. IEEE Trans Control Syst Technol 2024.
[31] Bai H, Cheng R, Jin Y. Evolutionary reinforcement learning: a survey. Intell Comput 2023;2:0025.

[32] Viswanadhapalli JK, Elumalai VK, Shivram S, Shah S, Mahajan D. Deep reinforcement learning with reward shaping for tracking control and vibration suppression of flexible link manipulator. Appl Soft Comput 2024;152:110756.
[33] Plappert M, Houthooft R, Dhariwal P, Sriram S, Schulman J, Radford A, et al. Parameter space noise for exploration. In: International conference on learning representations; 2018.
[34] Pourchot A, Sigaud O. Cem-rl: combining evolutionary and gradient-based methods for policy search. In: International joint conference on artificial intelligence; 2021. p. 3208–14.
[35] Mendonça R, Racanière S, Weber T, Heess N, Toyama D, Guez A, et al. Discovering and achieving goals via world models. Adv Neural Inf Process Syst 2021.
[36] Fortuin V, Osband I, Ritter H, Rauber PE, Gal Y. Deep exploration via bootstrapped ensemble uncertainty. Adv Neural Inf Process Syst 2022;35.
[37] Zhan W, Gupta A, Pinto L. Sense and adapt: leveraging adaptive noise in reinforcement learning for real-world robotic manipulation. In: Conference on robot learning; 2022.
[38] Wang J, Bao W, Yang W, Yang K, Qiao H. A survey on physics-informed reinforcement learning. IEEE Trans Neural Netw Learn Syst 2022;34(5):2231–49. https://doi.org/10.1109/TNNLS.2022.3142942.
[39] Du Y, Zhang K, Gao Y, Lin Y, Sadigh D, Finn C. Model-based physics-informed policy learning for robotics manipulation. In: Proceedings of robotics: science and systems; 2021.
[40] Zhu H, Liang H, Wang Z, Zhang F, Sun L. Adaptive physics-informed reinforcement learning for generalizable manipulation policies. IEEE Robot Autom Lett 2023;8(2):975–82. https://doi.org/10.1109/LRA.2023.3234695.
[41] Pfrommer B, Robuffo Giordano P, Trinkle J, Schaal S. Physics-informed reinforcement learning for bipedal locomotion. In: IEEE/RSJ international conference on intelligent robots and systems; 2022. p. 10864–71.
[42] Meng C, Seo S, Cao D, Griesemer S, Liu Y. When physics meets machine learning: a survey of physics-informed machine learning. arXiv preprint. arXiv:2203.16797, 2022.
[43] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[44] Kashefi A, Rempe D, Guibas LJ. A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. Phys Fluids 2021;33(2).
[45] Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nat Mach Intell 2021;3(3):218–29.
[46] Greydanus S, Dzamba M, Yosinski J. Hamiltonian neural networks. Adv Neural Inf Process Syst 2019;32.
[47] Cranmer M, Greydanus S, Hoyer S, Battaglia P, Spergel D, Ho S. Lagrangian neural networks. arXiv preprint. arXiv:2003.04630, 2020.
[48] Ng AY, Harada D, Russell S. Policy invariance under reward transformations: theory and application to reward shaping. In: Icml, vol. 99. 1999. p. 278–87.
[49] De Moor BJ, Gijsbrechts J, Boute RN. Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management. Eur J Oper Res 2022;301(2):535–45.
[50] Wiewiora E, Cottrell GW, Elkan C. Principled methods for advising reinforcement learning agents. In: Proceedings of the 20th international conference on machine learning; 2003. p. 792–9.
[51] Dong Y, Tang X, Yuan Y. Principled reward shaping for reinforcement learning via Lyapunov stability theory. Neurocomputing 2020;393:83–90.
[52] Li X, Xiao J, Cheng Y, Liu H. An actor-critic learning framework based on Lyapunov stability for automatic assembly. Appl Intell 2023;53(4):4801–12.

Dr. Raouf Fareh received a Ph.D. and a master's degree in electrical engineering from the University of Quebec (Ecole de technologie superieure) Canada in 2008 and 2013. He then joined Ottawa University in 2013 as a postdoctoral researcher. He joined the University of Sharjah, UAE in 2014.

Currently, Dr. Fareh is an Associate Professor at the University of Sharjah in 2014, where he is teaching courses related to robotics and control systems. His research interests include control and path planning for various types of robotic systems such as Manipulator, Mobile Robot Manipulator, Parallel robots, Exoskeletons Robot, Drones, etc. In control theory, his research interests include mathematical modeling, Intelligent control such as reinforcement learning control, Linear and Nonlinear control such as Active Disturbances Rejection Control, distributed control, Fractional Control, Sliding mode control, Synergetic control, etc. In addition, He is working on path planning for robotic systems using vision and image processing. He is the coordinator of Autonomous Robotics and Active Vision Research Group at the University of Sharjah and collaborates with Power Electronics and Industrial Control Research Group (GRÉPCI) in ETS, Canada; Computational Imaging Group at Skoltech, Moscow, Russia; and BioRobotics Lab at University of Wisconsin-Milwaukee, USA.

Tanjulee Siddique received her B.Sc. degree in Electrical and Electronics Engineering from the University of Sharjah in 2022. She briefly worked as a Robotics Software Engineer before joining the Autonomous Robotics and Active Vision Research Group at the University of Sharjah as a Research Assistant. Her recent research consists of intelligent control on robotic systems with special focus on tracking control, humanoid robots for upper-extremity rehabilitation, fuzzy logic-based control and reinforcement learning.



Dr. Kheireddine Choutri is a researcher and lecturer at the Aeronautics and Space Studies Institute of Blida 1 University. In 2019 he earned his Ph.D. in Aeronautics for his work on distributed artificial intelligence applied to a swarm of drones. Kheireddine is involved in several national and international research projects. He is also a reviewer for several indexed journals and conferences. His research focuses on robotics, artificial intelligence, computer vision, as well as guidance, navigation, and control of drones.

Dmitry V. Dylov (Member, IEEE) received the M.Sc. degree in applied physics and mathematics from Moscow Institute of Physics and Technology, Moscow, Russia, in 2006, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2010. He is currently an Associate Professor and the Head of the Computational Imaging Group at Skoltech, Moscow. His group specializes in computational imaging, computer/medical vision, and fundamental aspects of image formation.