# Evaluating Control Techniques On Robotic Arm

Ibrahim ELshenhapy Muhamed Ali Radwa Mahmoud Shady Osama

***Abstract*—The use of robots has grown significantly in recent years, in different applications spatially industrial applications. Efficiently controlling robots is crucial to ensure their optimal performance and productivity. This research paper presents a comparative analysis of two prominent control methods for controlling the joints of a welding manipulator. The traditional Proportional-Integral-Derivative PID control, analyzing its stability and precision. AND and Reinforcement Learning (RL), a modern control approach using machine learning algorithms. The main objective was to assess the effectiveness of each controller in controlling the Robot. The PID controller proved robust in predictable environments with known dynamics but not adaptive and requires manual tuning of their parameters, The RL controller showed adaptability and excelled at real-time learning for complex tasks. Welding processes have varying conditions and uncertainties, This comparative study provides that RL can adapt to these changes more effectively.**

***Index Terms*—Reinforcement learning, PID, Robot arm control, Model-free control**

## I. Introduction

In recent years, the utilization of robots in various industries and applications has witnessed a significant upsurge [1], leading to a growing need for efficient and precise robot control methods. The ability to control robots effectively is vital to achieving optimal performance, enhance productivity and ensure safety in diverse tasks, ranging from industrial automation to autonomous vehicles. A robot manipulator is a type of robotic system specifically designed for the purpose of manipulating objects in its environment. It consists of a series of connected links and joints, mimicking the human arm's structure, which allows it to reach different positions and orientations in three-dimensional space. The end of the manipulator is equipped with an end-effector, which can be a gripper, a tool, or any device capable of interacting with the environment. Robot manipulators find a wide range of usage and applications across various industries due to their versatility, precision, and ability to perform repetitive tasks with consistency. Some of the common applications include:(Manufacturing, Warehouse and Logistics, Medical and Healthcare, Agriculture, Construction, Space Exploration, Entertainment and Gaming, Household and Personal Use). Robot manipulators are extensively used in manufacturing processes to automate tasks such as material handling, welding, assembly, painting, and quality inspection. They can work in hazardous environments, perform repetitive tasks without fatigue, and improve production efficiency.

To control the joints of the robot, there are a lot of controllers ways :

**1. Task-Level Control:** Task-level control involves specifying the desired end-effector position and orientation rather than controlling individual joint parameters. This type of control is more intuitive for operators and simplifies programming for complex tasks. It is commonly used in applications like painting, welding, or assembly, where the end-effector's position and orientation are critical.

**2. Joint-Level Control:** This type of control directly controls the individual joints of the robot manipulator. It allows for precise control of each joint's position, velocity, or torque. Joint-level control is often used in applications where specific joint movements are essential, such as industrial robot arms performing pick-and-place tasks.

**3. Adaptive Control:** Adaptive control adjusts the robot's control parameters in real time based on changes in the environment or the robot's behaviour. It can improve performance and stability in dynamic or uncertain conditions.

**4. based Control:** With advancements in machine learning, controllers can be trained using data-driven techniques. Reinforcement learning, deep learning, and neural networks are increasingly used to optimize robot control policies, making them more adaptable and efficient.

This research paper aims to address the critical issue of selecting the appropriate control methods for joints of welding robots, specifically comparing the traditional Proportional-Integral-Derivative (PID) controller with the emerging Reinforcement Learning (RL)controller, some researchers discuss a continuous robot manipulator system with unknown function and dead-zone input by using reinforcement learning [3]. The robot control directly influences the ability of the robot to achieve desired tasks in accurate and reliable way. Traditional control methods like PID controllers have been widely used due to their simplicity and ease of implementation. Some researcher proved that a special linear PID controller is valid for tracking control of A compliant robotic manipulator theoretically [2] However, with the increasing complexity of tasks and the demand for adaptability in dynamic and uncertain environments, there is a growing interest in exploring alternative control techniques, such as RL controllers. To achieve these objectives, we will implement both the PID controller and RL controller on a Robot The performance of each controller will be evaluated and compared based on their ability to maintain set points, handle disturbances, and adapt to changing environments. In preparation for this research, relevant literature

and previous studies on PID controllers, RL controllers, and robot control methods have been extensively reviewed and analyzed. In this research, we will utilize MATLAB Simulink as the primary tool to simulate and analyze the results of our experiments comparing the PID controller and Reinforcement Learning (RL) controller for robot control.

The paper's organization is as follows:

1. Introduction that presents the background and significance of the problem, clearly states the research objectives, a brief overview of the tools and techniques used and the The related Work section delves into a comprehensive literature review on PID controllers, RL controllers, and previous studies on robot control methods. 2. the problem formulation

3. The solution method section describes the experimental setup in detail, including the description of the robotic platform and the implementation of both the PID and RL controllers.

4. The simulation setup section further elaborates on the scenario design and description, simulated environment configuration.

5. The Results and Analysis section presents the performance evaluation of both the PID and RL controllers. It analyzes tracking accuracy, steady-state error, control effort, and adaptability for each controller. The section also includes a comparative analysis of the strengths, weaknesses, and performance trade-offs between the two controllers, providing valuable insights into their suitability for various robotic applications.

6. The Conclusion section succinctly summarizes the key findings and result field of the robot control.

## II. Problem Formulation

We noticed a disturbance signal comes to the system when the robot arm grabs an object. Because of the new load added to the robot arm body. The load changes the environment states which causes an effect on the accelerations and velocities of the robot joint.

### A. Mathematical and Modeling Aspects

We treat robot control as a problem where the robot interacts with its environment, and we use a type of learning called "reinforcement learning." The robot's movement can be described using equations that show how its position and motion change over time when forces or torques are applied. The main goal is to train the robot to make decisions (actions) based on what it sees (its state) to achieve a specific task or goal while considering rewards for good actions and penalties for bad ones.

### B. Key Variables, Parameters, and Constraints

State (s): The robot's "state" means all the information about the robot's current condition, like its joint angles, how fast it's moving, and where its "hand" is located. We use "S" to represent the state space, which includes all possible states. Action (a): The "action" is what the robot does to control itself, like applying forces to its joints or sending commands to its motors. "A" represents the action space, which includes all possible actions. Reward (r): The "reward function" tells the robot how good or bad its actions are. It depends on the state and the action taken. We use "R(s, a)" to represent this function. Policy ($\pi$): The "policy" is like the robot's strategy. It's a rule that maps states to actions, helping the robot decide what action to take based on what it observes. We use "$\pi$(a—s)" to represent this mapping. Constraints: There may be certain limits or rules that the robot must follow, like not moving its joints beyond certain angles or staying within a specific workspace.

### C. Relevant Equations and Mathematical Models

Bellman Equation: This equation is crucial in reinforcement learning as given in equation 1. It helps the robot estimate the value of being in a certain state by considering the expected sum of rewards it can get from that state onwards.

$$Q_K(X_K, \mu_K^{\pi}) = U_K(X_K, \mu_K) + Q_{K+1}(X_{K+1}, \mu_{K+1}) \quad (1)$$

Q-Value Function: This function helps the robot evaluate how good it is to take a specific action in a specific state. It also considers the expected sum of rewards it can get from that stage onwards.

### D. Assumptions or simplifications

The Markovian Assumption in RL assumes that the agent's future actions depend only on its current state and not on the complete history of states and actions. This simplifies learning and reduces computational complexity, as the agent can focus on the immediate state without considering the entire past.

## III. Solution Method

To control a 6 DOF robotic arm using different control techniques, we need to first define the mathematical equations that describe the arm's dynamics. defining the following variables:

- Joint angles: $\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$ (radians)
- Joint angular velocities: $\omega = [\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6]$ (radians per second)
- Joint torques (control inputs): $\tau = [\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6]$ (Nm)

The dynamics of the robotic arm can be represented using the Lagrange-Euler equation:

M($\theta$)$\dot{\omega}$+ C($\theta, \omega$) $\omega$ = $\tau$ where:

- M($\theta$) is the inertia matrix
- C($\theta, \omega$) is the Coriolis and centrifugal forces matrix
- $\dot{\omega}$ is the joint angular acceleration

### A. Control Techniques

We will use this equation to apply the control techniques:

*1) PI Controller:* A PI controller can be applied to each joint independently to control its position.

*2) Model-free Q-learning:* Q-learning is a reinforcement learning algorithm where the robotic arm learns a policy by updating a Q-function. The Q-function Q(s, a) represents the expected cumulative reward when taking action in states. For the robotic arm, the states could be the joint angles and velocities, and action a could be the joint torques.

*3) Value iteration:* Value iteration is an algorithm for solving Markov Decision Processes (MDPs). The idea is to iteratively update the value function V(s), which represents the expected cumulative reward starting from state s and following the optimal policy thereafter. Again, state s could be the joint angles and velocities, and the value function V(s) would represent the expected cumulative reward from that state.

*4) Policy iteration:* Policy iteration is another algorithm for solving MDPs. It alternates between policy evaluation and policy improvement steps. In the evaluation step, the value function is computed for the current policy, and in the improvement step, a new policy is obtained by acting greedily concerning the current value function.

*5) Least squares:* Least squares is a numerical method used to estimate the parameters of a model that best fit a given set of data. In the context of robotics, it could be used for system identification or trajectory planning.

*6) Recursive Least Squares (RLS):* Recursive Least Squares is an online algorithm used for parameter estimation in a dynamic system. It updates the model parameters sequentially as new data becomes available, making it suitable for real-time applications.

### *B. Mathematical Formulation*

*1) PI Controller:* The control signal for each joint $\tau_i$ is given by:

$$\tau_i = K_p \cdot (\theta_{d_i} - \theta_i) + K_i \cdot \int (\theta_{d_i} - \theta_i) dt \tag{2}$$

where:
- Kp is the proportional gain and Ki the integral gain.
- $\theta_{d_i}$ is the desired joint angle for joint I
- $\omega_{d_i}$ is the desired joint angular velocity for joint i

*2) Reinforcement Learning (Model Free Value Iteration Approach):*

- Objective Function

$$U_K(X_K, \mu_K^\pi) = \frac{1}{2} X_K^T S X_K + \mu_K^{\pi^T} R \mu_K^\pi \tag{3}$$

Where: $\pi$= Policy, $X_K$= State, $\mu_K$= Action,
S,R= Weighting matrices

- Performance Index

$$J = \sum_{i=k}^{\infty} U_i(X_i, \mu^\pi) \tag{4}$$

- Bellman Equation as it was introduced in equation 1
- Optimal Policy

$$\mu^{\pi^*} = \arg\min \left( \frac{1}{2} \left[ X_K^T \ \mu_K^{\pi^T} \right] \ H \ \left[ (\mu_K^\pi)^T \ X_K \right] \right) \tag{5}$$

- Value Iteration Approach
  Initialize: $X_1$, $\pi^{(1)}$ and $\mu_1$
- Calculate Value

$$Q_{(K)}^{(r+1)}(X_{(K)}, \mu_K) = U_K(X_K, \mu_K^{(r)}) + Q_{K+1}^{(r)}(X_{K+1}, \mu_{K+1}^{(r)}) \tag{6}$$

- Extract Policy

$$\mu^{(r+1)} = -\left[ H_{\mu\mu}^{(-1)} \ H_{\mu X} \right]^{(r+1)} X \tag{7}$$

- Terminate Upon Convergence of $H^{(r)} \to H^*$

*3) Model free adaptive critics implementation:*

- Actor Tuning
  Solving Value Function:

$$V_k(\boldsymbol{X}_k) = \frac{1}{2} \boldsymbol{X}_k^T \boldsymbol{H} \boldsymbol{X}_k \tag{8}$$

Critic Approximation:

$$\hat{V}(\boldsymbol{X}_k) = \frac{1}{2} \boldsymbol{X}_k^T \boldsymbol{W}_{(c)} \boldsymbol{X}_k \tag{9}$$

$$\boldsymbol{\mu}^{\pi*} = -\boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{H} \boldsymbol{X} \tag{10}$$

Actor Approximation:

$$\hat{\boldsymbol{\mu}}_k = \boldsymbol{W}_{(a)} \boldsymbol{X}_k \tag{11}$$

Tuning Error:

$$\varepsilon_k^{\text{critic}} = \frac{1}{2} \left( \hat{V}(\boldsymbol{X}_k) - \tilde{V}_k^{\text{desired}} \right)^2 \tag{12}$$

Tuning Using Gradient Descent:

$$\boldsymbol{W}_{(c)}^{(r+1)} = \boldsymbol{W}_{(c)}^{(r)} - \alpha_c \left( \frac{\partial \varepsilon_k^{critic}}{\partial W_{(c)}} \right)^{(r)} \tag{13}$$

Tuning law:

$$\boldsymbol{W}_{(c)}^{(r+1)} = \boldsymbol{W}_{(c)}^{(r)} - \alpha_c \left( \left( \hat{V}(\boldsymbol{X}_k) - \tilde{V}_k^{\text{desired}} \right) \boldsymbol{X}_k \boldsymbol{X}_k^T \right)^{(r)} \tag{14}$$

- Critic Tuning
  Approximation:

$$\hat{\mu}_k = \boldsymbol{W}_{(a)} \boldsymbol{X}_k \boldsymbol{V} \tag{15}$$

Desired Value:

$$\tilde{\boldsymbol{\mu}}_k^{\text{desired}} = -\boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{W}_{(c)} \boldsymbol{X}_k \tag{16}$$

Tuning Error

$$\varepsilon_k^{\text{actor}} = \frac{1}{2} \left( \hat{\boldsymbol{\mu}}_k - \tilde{\boldsymbol{\mu}}_k^{\text{desired}} \right)^T \left( \hat{\boldsymbol{\mu}}_k - \tilde{\boldsymbol{\mu}}_k^{\text{desired}} \right) \tag{17}$$

Tuning Using Gradient Descent

$$\boldsymbol{W}_{(a)}^{(r+1)} = \boldsymbol{W}_{(a)}^{(r)} - \alpha_a \left( \frac{\partial \varepsilon_k^{actor}}{\partial W_{(a)}} \right)^{(r)} \tag{18}$$

Tuning Law:

$$\boldsymbol{W}_{(a)}^{(r+1)} = \boldsymbol{W}_{(a)}^{(r)} - \alpha_a \left( \left( \hat{\boldsymbol{\mu}}_k - \tilde{\boldsymbol{\mu}}_k^{\text{desired}} \right) \boldsymbol{X}_k^T \right)^{(r)} \tag{19}$$

## IV. Simulation

Consider a step input with an initial value of 0 and a final value of 1 radian, occurring at t=5 seconds. This input is then applied to our plant, comprising a Braccio robot arm with 5 degrees of freedom and a gripper. Prior to this, the system is governed by a PI controller, as expressed in Eq. (2). This approach along with a model-free value iteration reinforcement learning (RL) algorithm, employing the actor-critic architecture where the actor is represented as Wa (refer to Eq. (19)), and Wc as depicted in Eq. (14)). Both strategies utilize negative feedback to compute the error and enact precise control actions. The parameters essential for the PI controller, namely the proportional gain $K_P$ and integral gain $K_I$, are specified in Table I.

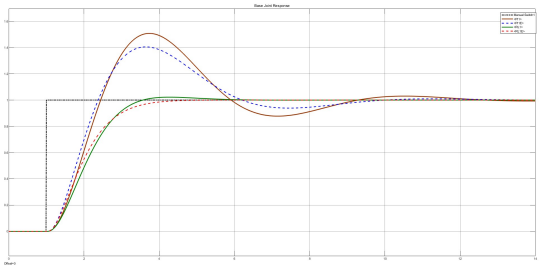
TABLE I: PI controller parameters for Each Joint

| Joint Number | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| P | 0.026 | 0.073 | 0.031 | 0.061 | 0.056 | 0.043 |
| I | 0.017 | 0.038 | 0.017 | 0.021 | 0.031 | 0.022 |

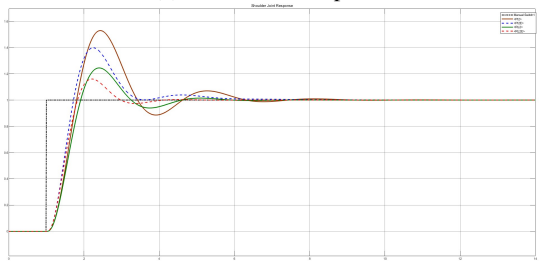
A model-free actor-critic architecture based on reinforcement learning, from Eq.(3) to Eq.(19), is simulated for comparison within a controlled parameter environment. In this reinforcement learning scenario, various parameters need to be adjusted. However, all parameters in the RL case can be set to the same values across all joints. This is exactly what I am doing in my case, except for the initial values of 'Wa'. For 'Wa', I assign only the initial value, which is a (1*1) matrix equal to the value of the 'P' parameter for the corresponding joint and the initial value for 'Wc', which is a (2*2) matrix. The matrix has (1, -P) for the first row and (-P, 1) for the second row. As for the other parameters, I set $\alpha_a$ to 0.006, $\alpha_c$ to 0.0004, 'R' (action weighting matrix) to 0.08, and 'S' (state weighting matrix) to 0.4. In my particular case, both 'R' and 'S' are (1*1) matrices since in my approach I'm utilizing only the current error.

The simulation results are presented in Figure 1. Sub-figures 1a through 1f depict the outcomes for individual joints. In the first scenario, the PI results are illustrated with a blue dashed line, while the RL outcomes are represented by a red dashed line. The input signal, indicated by the step function, is depicted as the black line. It's clear that the RL approach outperforms the PI controller in terms of both overshooting and reaching the steady state. The RL method exhibits a lower overshoot and converges more rapidly to the steady state. However, this scenario isn't our only focal point.

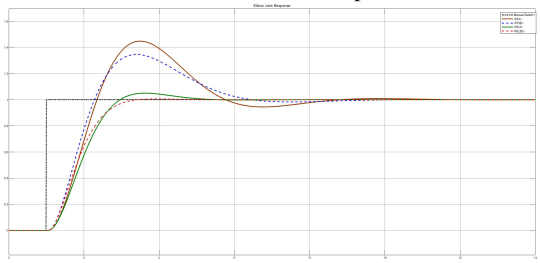
To provide a clearer demonstration of the superiority of the RL algorithm, we introduced a second scenario using the same parameters as a previous case but now we changed the inertia parameter in the motor controlling each joint to increase by 1.5 times its initial value. The corresponding results are depicted in Figure 1, where in this scenario the PI response is represented by the brown solid line, and the RL response is represented by the green solid line. Observing the results, it becomes evident that the RL algorithm continues to outperform the PI controller. Furthermore, it's important to note that the



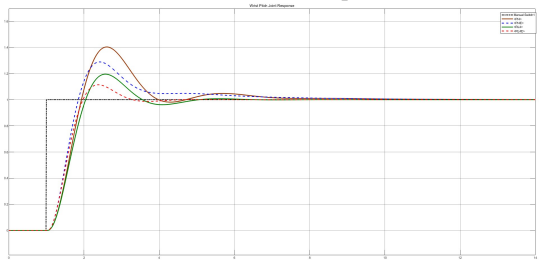
(a) Base Joint Response



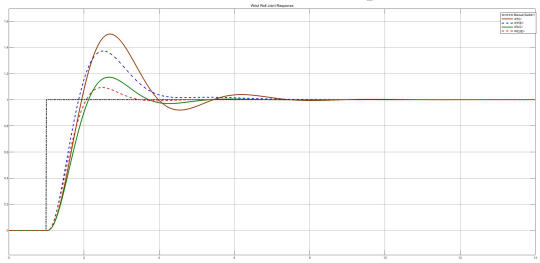
(b) Shoulder Joint Response



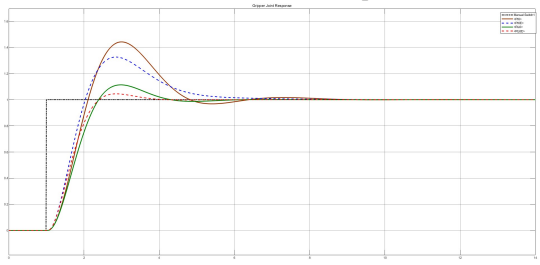
(c) Elbow Joint Response



(d) Wrist Pitch Joint Response



(e) Wrist Roll Joint Response



(f) Gripper (End Effector) Joint Response

Fig. 1: Joint Responses for Braccio Robot Arm

PI controller exhibits a much worst performance when the motor parameters are altered. Noticing, oscillations appear, suggesting that if the motor parameters were to undergo further changes, the PI controller could become unstable. However, the RL algorithm remains stable even with the slight parameter adjustment, and it maintains convergence to the steady state without introducing any oscillations.

## V. Conclusion

With the goal of having a better control technique for the robot arm the comparative analysis of the PI controller and the model-free RL algorithm revealed the latter's superiority in terms of control accuracy, speed of convergence, and robustness to parameter changes. Where the RL algorithm's capability appeared to enhance control accuracy and efficiency in a typical operational setting. It also shows maintained stability and convergence even with the parameter change, showcasing its ability to handle variations and maintain desirable control behaviour. These findings underscore the potential of RL-based approaches in advancing control strategies for complex robotic systems, as exploring hybrid control strategies that combine the strengths of traditional methods with the adaptability of RL. These could further elevate the performance and adaptability of control mechanisms in complex robotic systems

## References


[1] P. Rocco,"Stability of PID control for industrial robot arms",IEEE Transactions on Robotics and Automation,vol.12,(1996), DOI:10.1109/70.508444

[2] Pan.Yongping, Li.Xiang and Yu.Haoyong, "Efficient PID Tracking Control of Robotic Manipulators Driven by Compliant Actuators",IEEE Transactions on Control Systems Technology,Vol.27, (2018), DOI:10.1109/TCST.2017.2783339.

[3] Li Tang,Yan-Jun Liu and Shaocheng Tong"Adaptive neural control using reinforcement learning for a class of robot manipulator",Neural Computing and Applications vol.25,(2014).