International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Robotic Arm Drift Compensation Using Adaptive Computed Torque Control Based on Reinforcement Learning

Pride Mashiyani[1], Didymus Tanyaradzwa Makusha[2]

[1]Student, Electronic Engineering Department, Harare Institute of Technology
[2]Supervisor, Electronic Engineering Department, Harare Institute of Technology

Abstract
The merging of machine learning (ML) into robotics has significantly enhanced the management of drift in industrial applications. Robotic arms often experience drift over time due to various factors such as temperature fluctuations, gearbox backlash, wear and tear, sensor inaccuracies, and changes in load. This drift poses challenges to maintaining accuracy and precision in industrial robotics. While model-based methods have addressed drift compensation, they come with considerable limitations compared to data-driven approaches. This paper will review the PID, Computed Torque Control and Adaptive Computed Torque Control based on Reinforcement Learning, soft actor critic (SAC) model, examining their strengths, weaknesses, and potential areas for improvement to enhance the accuracy and precision of robotic systems in industrial settings.

Keywords: Computed Torque Control (CTC), Machine Learning (ML), Proportional Integral Derivative (PID), Reinforcement Learning (RL), Soft Actor Critic (SAC).

1. Introduction

Model-based control systems for robotic arms are prone to drift over time and lack the ability to adapt to new operating conditions, primarily due to discrepancies between kinematic and nominal values stemming from manufacturing and assembly tolerances, known as kinematic errors [1]. Traditional drift compensation methods often fail to adjust Denavit-Hartenberg (DH) parameters as these kinematic errors accumulate, necessitating periodic manual calibration to mitigate drift. However, the introduction of neural networks enhances adaptability by enabling these models to learn complex operational dynamics from real-time data, making them effective for drift compensation. Advances in artificial intelligence, particularly through ML and deep learning algorithms, offer improved performance in addressing the complexities and nonlinearities of robotic operations compared to classical models [2]. Drift can also impact sensor readings, which in turn affects the accuracy, precision, and reliability of robotic force control systems. To ensure effective force control, it is critical to develop methods for real-time drift prediction and compensation [3]. In complex robotic operations, hard coding and modeling safe trajectories can be cumbersome and time-consuming. Machine learning models, particularly through imitation learning, can quickly adapt and learn dynamics based on expert demonstrations, allowing them to generalize to new operating environments [4] Overall, AI has emerged as a valuable tool for addressing drift in robotic arms,

outperforming traditional models and presenting promising solutions for various challenges in industrial robotics.

PID along with fuzzy logic controllers, represent traditional control models grounded in mathematical concepts and logical analysis. They work well in applications involving a single input and output. While they are relatively easy to implement, they can become unstable when managing complex and nonlinear systems [5].

2. Methodology

This study will focus on a 2DOF robotic arm dynamics. Matlab SimScape model will be developed and a trapezoidal trajectory profile will be used to analyse the control system response to path dynamics. Random number generator will be used during the training of the RL agent. This will model the random nature of drift generators in real world applications.

DRL algorithms were selected for their capability to learn from experience and engage with the environment in a step-by-step manner. These algorithms can also develop dynamic policies that adjust to environmental changes and effectively manage continuous action spaces [6].

3. Robotic Arm System Dynamics

Robotic arm system dynamics refers to the study of motion and forces of a robotic arm, comprising of joints, links and end effector. It is crucial in studying the dynamics in order to understand how the arm moves, interacts with the environment and performs tasks.

3.1 Mathematical modelling of robotic system

The dynamics of a robotic arm can be modelled using various mathematics models such as Newton-Euler equations, Lagrange's equations and Hamilton's equations. In this paper, The Newton-Euler equations will be examined for a 2 DOF robotic arm manipulator and its controller design. Forward dynamics focuses on determining the motion of the robotic arm based on the applied forces and torques. Conversely, inverse dynamics is concerned with calculating the forces and torques needed to produce a desired motion of the robotic arm.

Neuton's second law of motion

$$\ddot{a} = F/m \quad (1)$$

Where,

$\ddot{a}$ is the acceleration

F is the force

m is the mass

Equation of a motion of a robotic arm,

$$\tau = M(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + g(q) \quad (2)$$

Where,

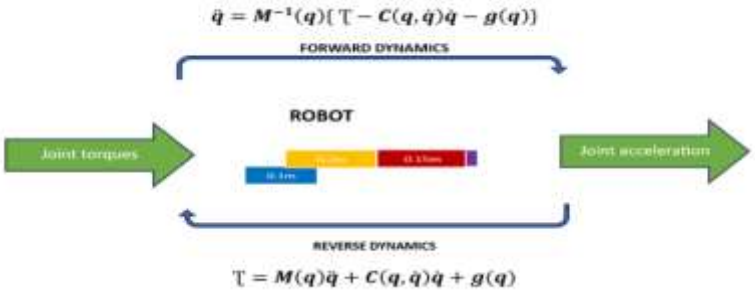
τ is the total torque required to drive the manipulator

$M(q)\,\ddot{q}$ is the amount of torque required to move a mass of the arm body.

$C(q,\dot{q})\,\dot{q}$ is the torque required to overcome Coriolis's and centrifugal forces.

$g(q)$ is the torque required to overcome the gravitational pull

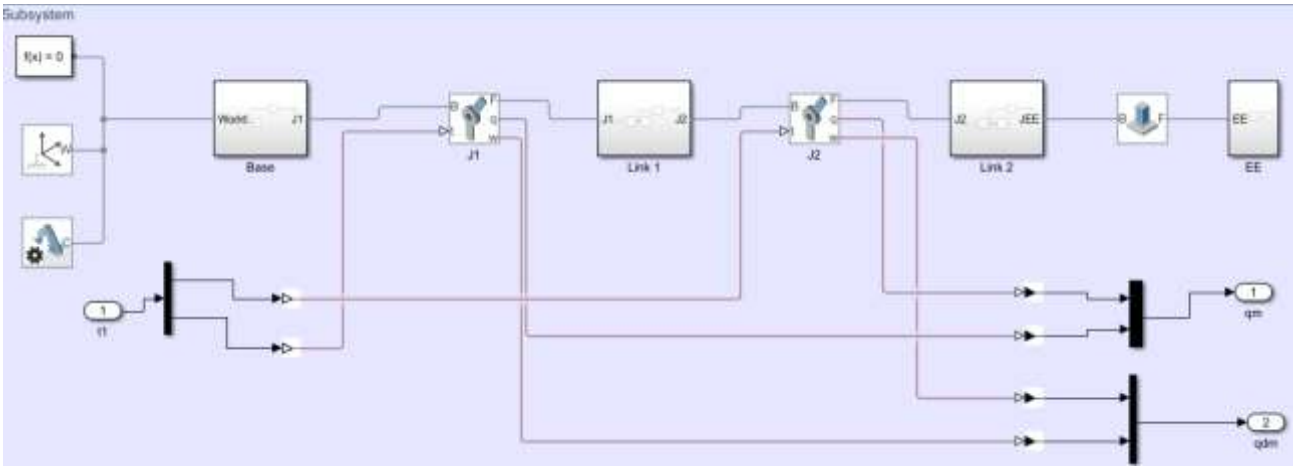
Figure 1: Forward and Reverse dynamics of a robotic arm.



4. Physical Modelling of Robotic Arm Using SimScape Model

The 2DOF robotic arm with the dimensions shown in figure 1 was modelled in SimScape with material density of 1000kg/m3. Position and velocity sensors are used to provide measured data from the 2 joints.

Figure 2: SimScape model of a 2 joint robotic arm.



4.1. The experiment Setup

Robotic arm drift of the SimScape model will be measured against the modelled forward dynamics for each joint.

Figure 3: Set up to determine drift.

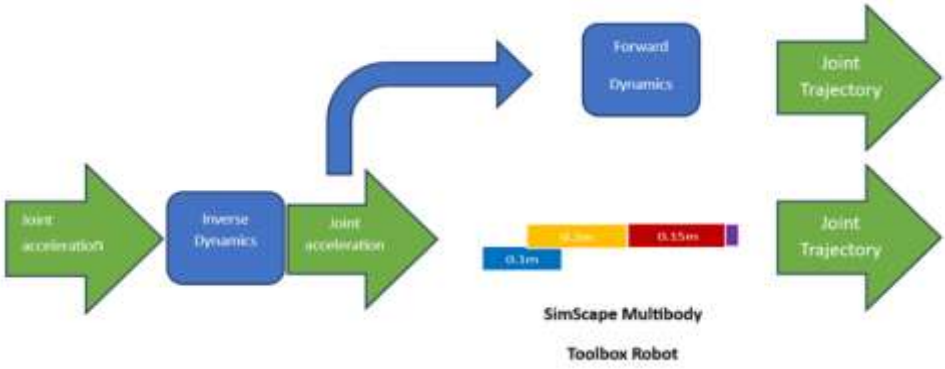
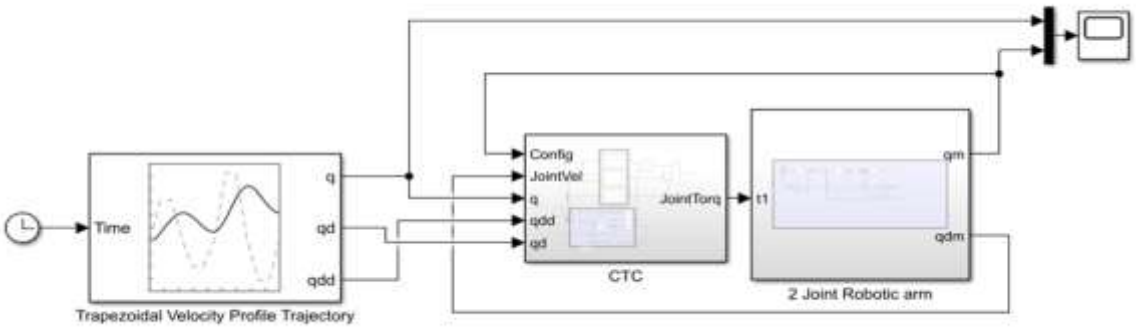


Figure 4: Set up to determine drift in Matlab Simulink.



5. Robotic Arm Basic Controller Design

5.1. Independent Joint Control

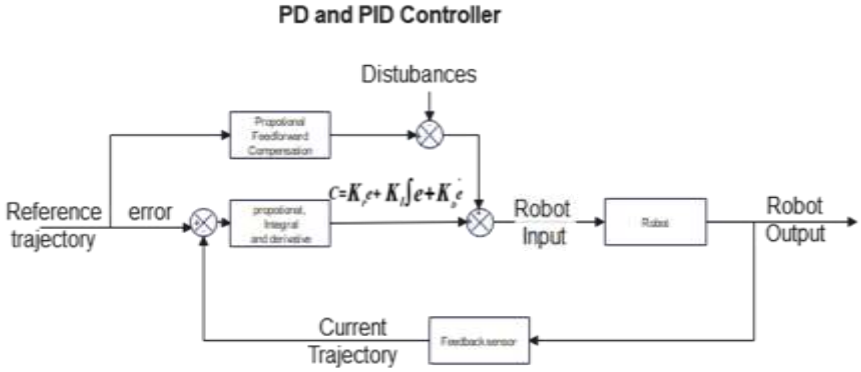
PID controller is one of the classical models used in the design of many control systems because of a number of reasons. Settling time of a controlled process is reduced significantly. Controllers can be used in a wide range of application. Response time is shorter allowing faster correction of error. The PID controller can be modelled using the following equation

$$C = K_P e + K_I \int e + K_D \dot{e} \qquad (3)$$

Where,

C is the controller gain
K_P is the proportional gain coefficient
K_I is the integral gain coefficient
K_D is the derivative gain coefficient.

Figure 5: PID controller design for single joint control.



5.2 . Feed Forward Compensator

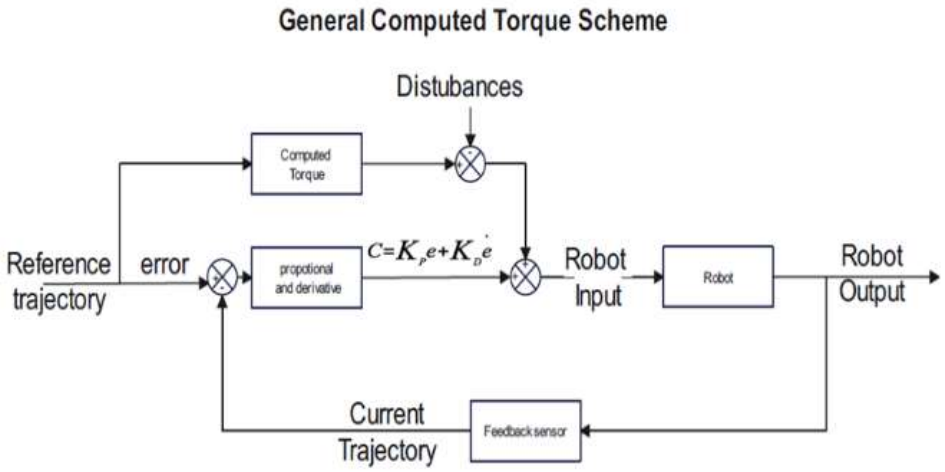
It will predict the disturbance that is to come and generate a suitable signal to minimize its effect. Proportional feed forward compensator multiplies reference signal with a constant. PID is good at setpoint tracking but fails to track the constantly changing reference perfectly.

5.3. General Computed Torque Control Scheme.

The computed torque control (CTC) strategy applies the equations of robotic dynamics to anticipate dis

turbances. This method relies on the controller's commands, which utilize prior knowledge encapsulated in a dynamic model of the system. With the aim of tracking trajectories in joint space, the target trajectories consist of joint angles, velocities, and accelerations. By using feedback linearisation, it effectively mitigates the manipulator's nonlinearities, as illustrated in Figure 6.

Figure 6: General CTC Scheme.



CTC scheme accepts joint position, velocity and acceleration as inputs. Figure 7 below shows the scheme with a linearised controller. Position and velocity error are multiplied with K_p and K_D coefficients respectively. The results are added to the acceleration and a mass matrix is produced. The mass matrix is added to the Coriolis and gravitational force models. The controller output is then fed to the robot and the input torque required.

The outer loop controller is most suitable and PD controller have been proposed as outer loop controller [7]. It is important to highlight that the use of the feedback linearizing transformation results in the tracking error dynamics being described by a linear state equation with constant coefficients.

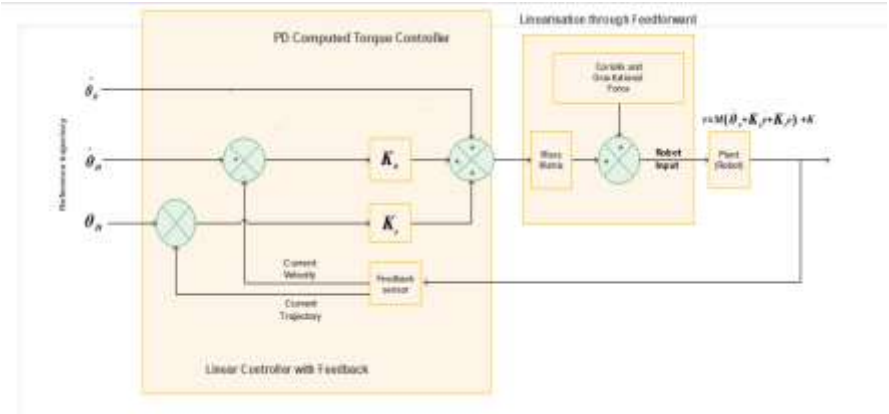
The CTC law is given as follows:

$$\tau = M\left(\ddot{\theta} + K_D(\dot{\theta}_D - \dot{\theta}_m) + K_p(\theta_D - \theta_m)\right) + N \qquad (4)$$

Where,
τ is the torque required
M is the mass matrix
$\ddot{\theta}$ is the desired acceleration
$\dot{\theta}_D$ is the desired velocity
$\dot{\theta}_m$ is the measured velocity
θ_D is the desired angle
θ_m is the measured angle
N is the torque required to overcome Coriolis, gravitational and centrifugal force
K_D is the derivative gain coefficient

K_p is the proportional gain coefficient

Figure 7: CTC Scheme with linearised controller.



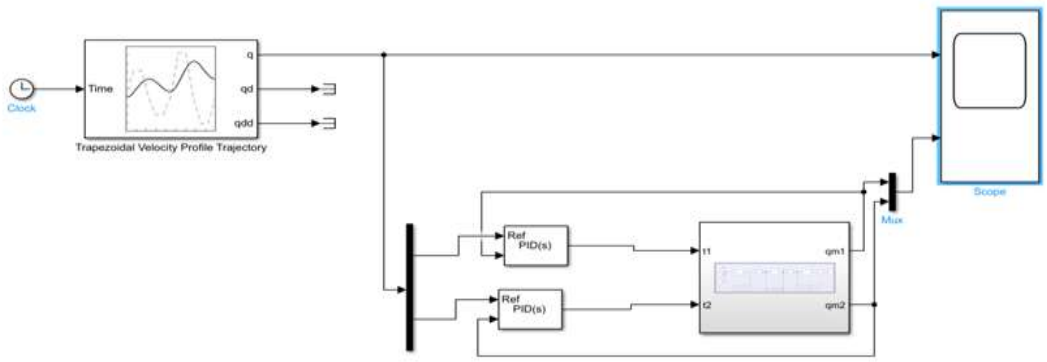
CTC scheme has three inputs,θ_D, $\dot{\theta}_D$, and $\ddot{\theta}_D$ representing desired trajectory, desired velocity and desired acceleration respectively. Error signal generated by comparing the desired trajectory and current trajectory is multiplied by a factor K_p and is added. Error from the desired velocity and the measured velocity is multiplied by a factor K_D and added. The adder output is multiplied with the mass matrix. The result is added to coriolis and gravitational force models. The result is the controller output torque appropriate to control the joints position, velocity and acceleration.

6. Matlab Implementation and Simulations

6.1. The PID implementation.

The 2DOF manipulator has 2 controllers for joint position tracking.

Figure 8: 2 PID controllers



The two PID controllers each controlling a single joint are used. Reference signal is generated from the trajectory generator. PID control tracks the trajectory and ensure the error between the reference and the measured is reduced timeously and as lowest as possible. To ensure the controllers are tuned properly, Matlab PID tuner is used.

6.2. PID Single Joint Tunning Using PID Tuner In Matlab.

Figure 9: PID tuner results J1.

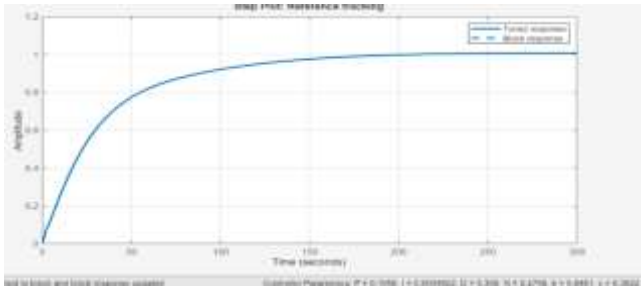
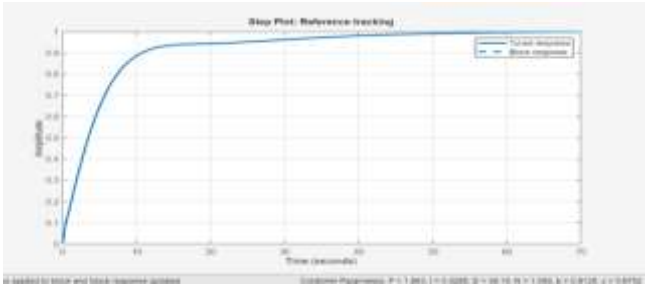


Figure 9 shows the J1 tunning results indicating the J1 response characteristics indicating a rise time of 180 seconds, and the controller coefficients, P = 0.1, I = 0.00048, and D = 5.3.

Figure 10: PID tuner results J2.

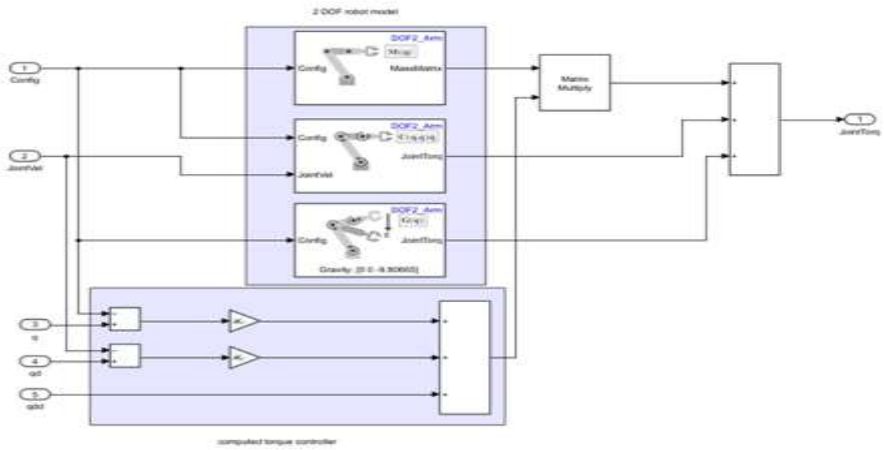


On figure 10, Tunning results corresponding to J2 response characteristics indicating a rise time of 15 seconds and the controller coefficients, P = 1.88, I = 0.028, and D = 30.18.

6.3. The CTC implementation and results.

Joint space mass matrix, velocity product torque and gravity torque Simulink blocks are used to model the robotic arm dynamics. Only the K_p and K_D parameters are adjusted.

Figure 11: CTC Simulink.



Adjusting the 2 parameters is all what it required to tune the controller because the model has been line-arised and easier to adjust. Below is table of tunning results.

Table 1: Controller gain after adjustingK_p and K_D parameters

$K_p \backslash K_D$	1	100	1000	2000	3000
1	14	250	348	352	354
100	46	270	349	350	362
1000	98	230	317	350	353
2000	127	276	358	353	354
3000	266	282	362	351	352

Controller gain was obtained by determining the peak to peak output signal for J2. The maximum output of 362 corresponds to K_p = 3000 and K_D = 1000 or K_p = 100 and K_D= 3000.

Figure 12: K_p = 1, K_D = 1, Gain= 14

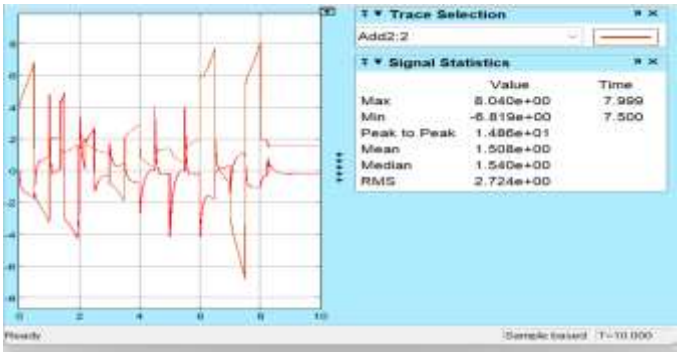


Figure 13: K_p=1, K_D =1, trajectory for J1 and J2

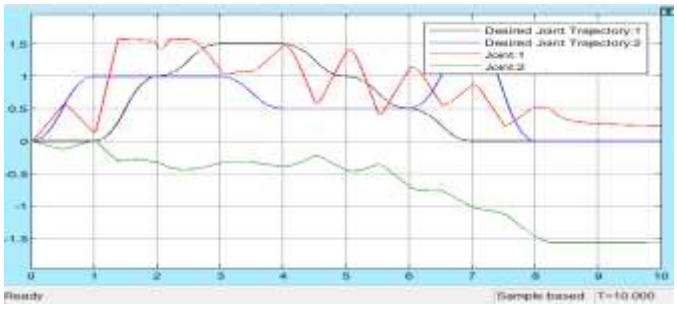


Figure 14: K_p=2000, K_D =1, Gain =127

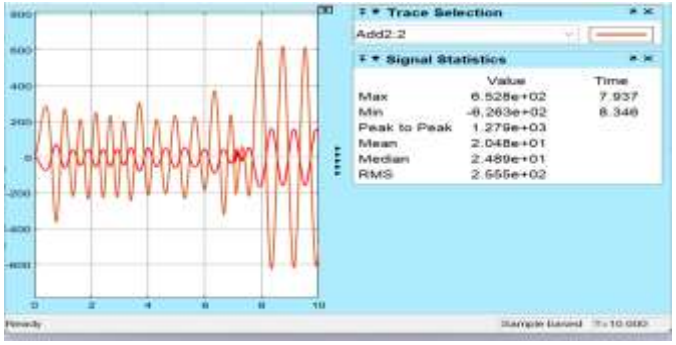


Figure 15: K_p=1, K_D =1000, Trajectory for J1 and J2

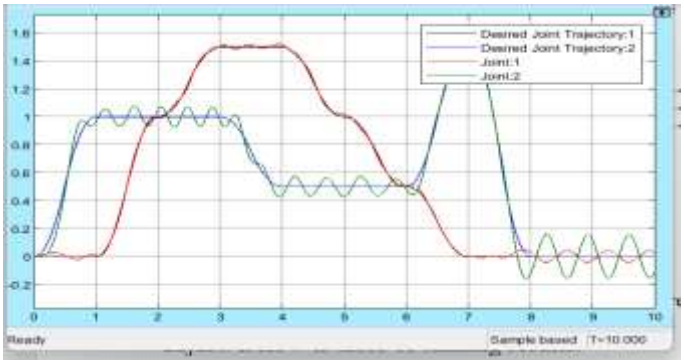


Figure 16: K_p=3000, K_D =1000, Gain = 362

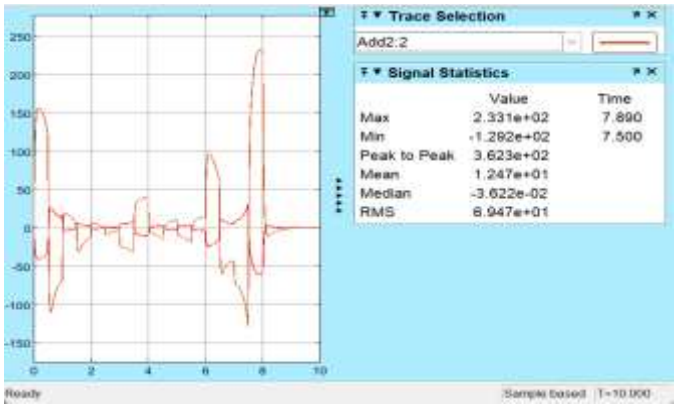


Figure 17: K_p=3000, K_D =1000, Trajectory for J1 and J2

CTC provides numerous benefits; however, it necessitates a deep understanding of the system's analytical model to calculate the feedforward torques necessary for trajectory execution. The dynamics of robotic manipulators are often coupled and exhibit significant nonlinearities, with uncertainties arising from friction and unmodeled dynamics. This complexity makes it challenging to achieve an accurate model, prompting extensive research into alternative control strategies. Notably, feedforward nonlinear control and inverse dynamics control are distinct approaches that can be employed, with the latter typically demonstrating superior tracking performance [8].

7. The SAC Reinforcement Learning model

A key feature of reinforcement learning (RL) is its interactive trial-and-error methodology. In this process, the agent learns by observing the results of its actions and modifying its behaviour according to the rewards it receives, which are represented as scalar values indicating the effectiveness of those actions. In RL terminology, the state refers to a collection of all the relevant information necessary to characterize the environment.

Figure 18: K_p=3000, K_D =1000, Trajectory for J1 and J2

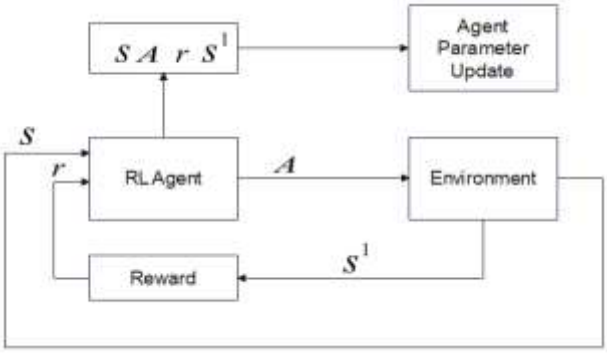


Figure. 18 depicts the typical interaction between an agent and its environment in a RL algorithm, the agent observes state S from the environment and interacts with it by choosing an action A based on the current policy π. This action A leads to a change in the environment, resulting in a new state $S'$and producing scalar feedback in the form of a reward r. The transition data $\{S, A, r, S'\}$ is known as an experience, and this iterative process continues until the training is finished. The agent's primary objective is to learn an optimal policy $\pi*$ that maximizes the expected return gt, represented as follows

$$g_t = \sum_{k=0}^{\infty} \Upsilon^k r_{t+k+1} \quad (5)$$

where the discount factor $\gamma \in [0, 1]$, the importance of future rewards.

A key challenge in RL lies in balancing exploration and exploitation: determining whether to try random actions to explore the environment or to utilize existing knowledge to maximize returns. The ϵ- greedy strategy is frequently used in this context. It randomly selects an action with a probability of $\epsilon \in [0, 1]$, and opts for the greedy action otherwise. Finding an appropriate ϵ value is crucial for effectively balancing exploration and exploitation, with this value typically decreasing over time to prioritize exploitation.

In this study, Soft Actor-Critic (SAC) Algorithm is a favorable among three major properties. An actor-critic model is used which has different nets for policy and value functions, it is also described as off-policy method which uses past experiences efficiently and maximizing entropy. The latter is a characteristic property of SAC, which drives the policy to trade off expected return and entropy (a regularization term that measures the randomness of the policy). This equilibrium adds considerable balance and exploration.

The SAC agent is trained off-policy with a behaviour and target critic networks. The distinction between exploration and target policy offers more meta parameter control as to whether online data is collected with one strategy while the learned value function is updated according to another, as in the DDQN

The SAC algorithm was selected for multiple reasons. As a model-free and off-policy approach, it excels in scenarios with continuous state and action spaces, making it ideal for practical applications. Furthermore, SAC demonstrates high sample efficiency and functions within a maximum entropy framework,

which benefits environments involving robotic arms. Additionally, the SAC algorithm generates smoother actions, which are crucial for achieving precise control of robotic arms [9].

As a model-free and off-policy approach, the SAC algorithm is ideal for tasks involving continuous state and action spaces, making it a popular choice for addressing real-world challenges. The SAC algorithm provides excellent sample efficiency and operates within a maximum entropy framework, which is particularly advantageous for robotic arm applications. Additionally, SAC produces smoother actions, which is essential for achieving precise control of a robotic arm [10].

SAC employs a stochastic approach that yields several benefits over deterministic ones. For example, increasing the randomness can prevent premature convergence to a local optimum and encourage exploring the solution space more widely to guide an agent well enough to approach the global optimum.

SAC is an off-policy RL algorithm in which three types of functions are represented by five networks. These are the policy, the action value, and the state value function. A randomized action network approximates the policy function, a V critic network estimates the state value function, and Q-networks are used to approximate the action value function. The algorithm consists of a number of steps, including strategy evaluation, strategy improvement, updates to the state value network and tuning the entropy weight [11].

SAC is a popular DRL algorithm that is hailed for its impressive sample efficiency and robust-ness under continuous action domains. Method Standard policy gradient methods such as Trust Region Policy Optmisation (TRPO) and proximal policy Optimisation (PPO) considerably improved DRL via facilitating stable on-policy learning. These methods however tend to require more samples, being unable to efficiently exploit the knowledge learned from the past. In contrast, SAC works as an off-policy algorithm which can leverage a replay buffer for re-using past experience and helps a lot for sample efficiency. SAC then, through adopting the maximum entropy framework, reasonably balances exploration and exploitation and is therefore particularly attractive in complex industrial applications where such exploration costs are prohibitive. Its ability to handle high-dimensional state and action has enabled SAC to develop sophisticated control policies, a feature crucial for accurate robotic-arm control [12].

8. Adaptive CTC Based on SAC RL Model Design.

The CTC model has shown that it can adress drift problems by modeling the Newton-Euler equation. The challenge is the need to adjust the K_p and K_D parameters which is time consuming as it is a try and error approach. In order to address this problem, an adaptive SAC RL model is proposed where the agent will learn the model by itself and updates its neuron's weights accordingly. The RL agent has replaced the K_p parameter. K_D parameter is preset to 1000. During training, the agent constantly adjusts the K_p parameter until the position error for J2 is equal or less than 0.1 as shown in figure 19. At this error margin, the agent is rewarded with a value 10. If the error is greater than 0.1 as shown in figure 18, the penult is of value -1 is given to the agent for failing to control the error within the margin.

Figure 19: K_p=1, K_D =1 error margin for J2 greater than 0.1

Figure 20: K_p=3000, K_D =1000 error margin for J2 greater than 0.1

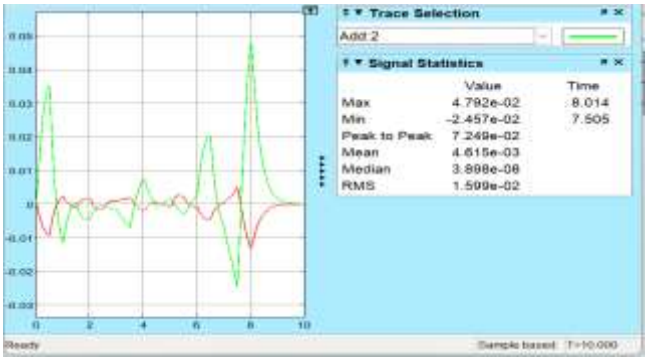
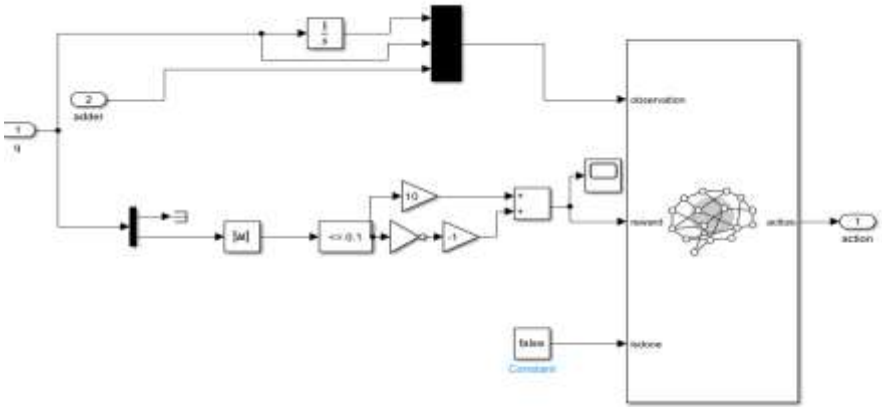


Table 2: Controller gain after adjustingK_p and K_D parameters

Joint position		Max error K_p=1,K_D =1	Min error K_p=1,K_D =1
	J1	2.5	0.01
	J2	1.3	0.073

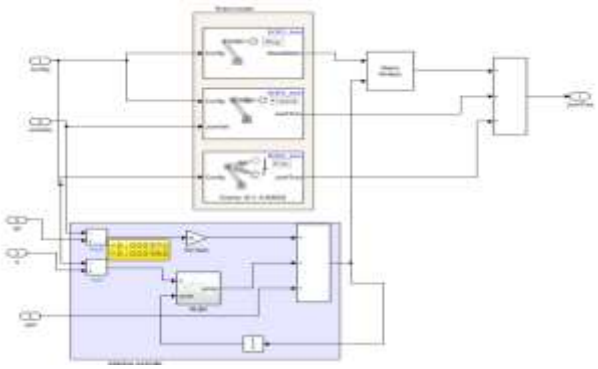
Figure 21: The SAC RL argent.



8.1. The Adaptive CTC Design.

The K_p parameter adjustment is carried out by ML model. RL model learns how to reduce the position error by adjusting the K_p parameter. The goal is to reduce the J1 and J2 joint position errors magnitude to 0.01and 0.073 respectively. This error margin corresponds to the desired trajectory tracking tolerance. Anything outside is considered as drift.

Figure 22: RL agent replaced the K_p gain block



The RL agent will explore the possible actions and exploit the actions which maximises the rewards. The reward is maximum if the position error is minimum. Consequently, tracking the trajectory with maximum precision.

Figure 23: Before training and K_D parameter set to 1000.

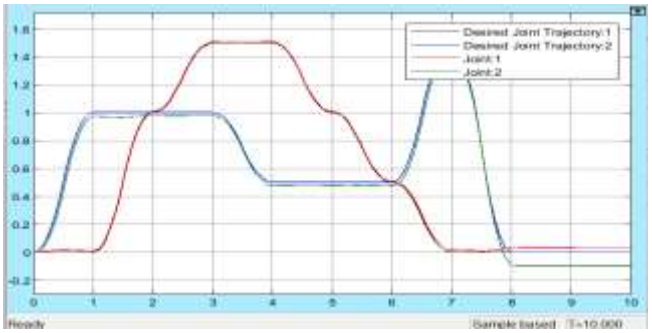
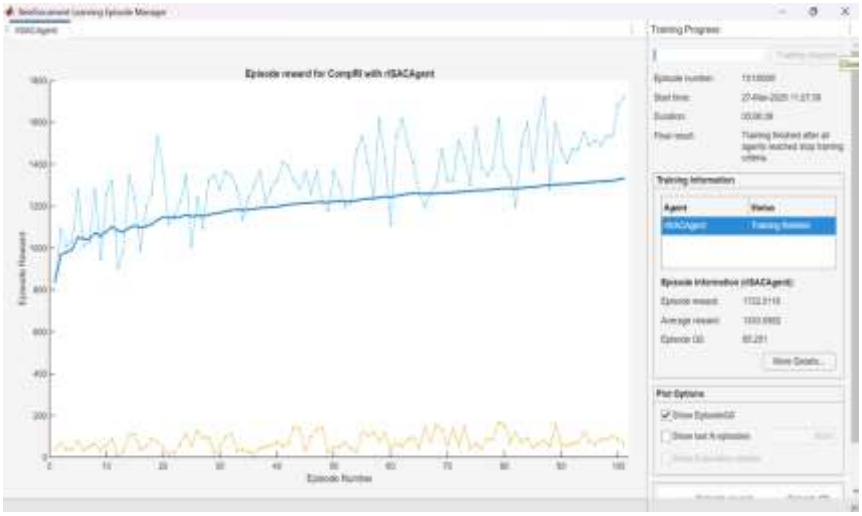


Figure 22 shows trajectory tracking before training of the RL agent. It is clear that the tracking accuracy is poor and exhibiting drift. Hence a need to train the model.

8.2. The SAC RL Training.

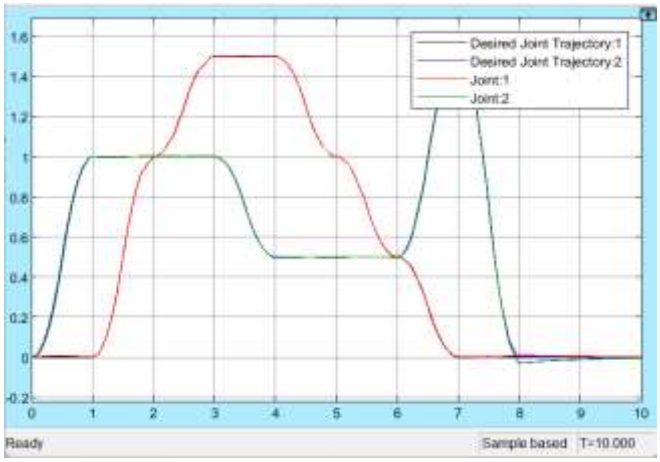
The initial joint angles are randomised to +/- 5 degrees. This will enable the argent to be able to compensate the drift in the range of +/-5 degrees. Argent gain range limit is +/- 100. This was set so that the amount of time required for training is reduced and precision control is not compromised.

Figure 24: Training for the RL agent.



Training terminated after 100 episodes, average reward reached 1333 and reward value at 1722.

Figure 25: After Training the RL agent.



Trajectory tracking by the RL agent has achieved the accuracy expected. Both J1 and J2 are being controlled accurately with no drift.

9. Results and Conclusion

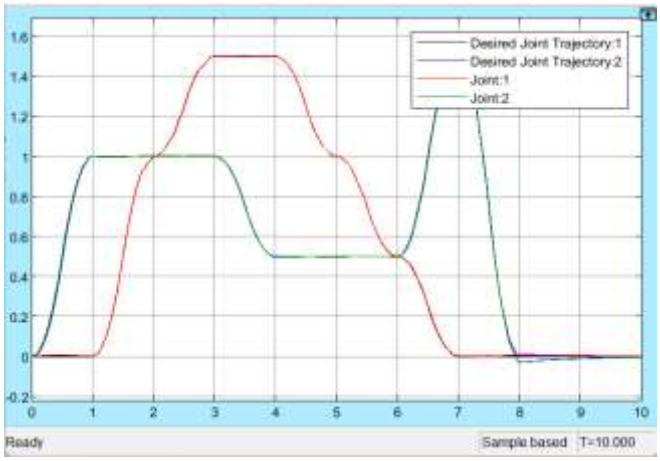
Figure 26: The PID Controller results.

Figure 24 is J1 and J2 position with respect to the desired joint trajectory. Because there are various factors that contributes to a complex dynamic of a robotic arm, PID control fails to compensate for these factors hence a poor response to trajectory tracking.

Figure 27: The Computed Torque Controller results.

CTC model has proved to be more effective as compared to PID model. The model takes into account the factors contributing to the dynamics of the robotic arm, hence a better performance on tracking position trajectory. Figure 27 shows how the model achieved trajectory tracking more precisely. Nevertheless, the model fails to adapt online to drift over time because it requires manually readjustment of the K_p and K_D parameters.

Figure 28: The Adaptive CTC based on SAC RL model.



The adaptive CTC controller has achieved the expected performance by precisely tracking the joint position trajectory. It also has the aspect of online self-calibration as it learns from the environment and not from a model during training phase. The environment during training was meant to randomly change its initial joint positions and the agent would learn how to compensate for the error. During normal operation the agent would be in a position to compensate for drift without any human intervention.

10. Acknowledgment

I wish to acknowledge My Supervisor, Eng T.D Makusha, HOD, Electronic department, Eng B. Musiiwa and members of the staff for providing the guidelines and directions throughout the research work.

References

1. Y. Kong et al., "Online kinematic calibration of robot manipulator based on neural network," Measurement, vol. 238, p. 115281, Oct. 2024, doi: 10.1016/j.measurement.2024.115281.
2. A. Mystkowski, A. Wolniakowski, N. Kadri, M. Sewiolo, and L. Scalera, "Neural Network Learning Algorithms for High-Precision Position Control and Drift Attenuation in Robotic Manipulators," Appl. Sci., vol. 13, no. 19, p. 10854, Sep. 2023, doi: 10.3390/app131910854.
3. X. Qiao, C. Xu, Y. Wang, and G. Ma, "SDI: A sparse drift identification approach for force/torque sensor calibration in industrial robots," Neurocomputing, vol. 620, p. 129292, Mar. 2025, doi: 10.1016/j.neucom.2024.129292.
4. S. Nahavandi, R. Alizadehsani, D. Nahavandi, C. P. Lim, K. Kelly, and F. Bello, "Machine learning meets advanced robotic manipulation," Inf. Fusion, vol. 105, p. 102221, May 2024, doi: 10.1016/j.inffus.2023.102221.
5. J. R. Torraca, B. D. O. Capron, and A. R. Secchi, "A robust deep reinforcement learning approach for the control of crystallization processes," Comput. Chem. Eng., vol. 199, p. 109114, Aug. 2025, doi: 10.1016/j.compchemeng.2025.109114.
6. A. Chatterjee and D. Khovalyg, "Dynamic indoor thermal environment using Reinforcement Learning-based controls: Opportunities and challenges," Build. Environ., vol. 244, p. 110766, Oct. 2023, doi: 10.1016/j.buildenv.2023.110766.
7. V. M. Becerra, S. Cook, and J. Deng, "PREDICTIVE COMPUTED-TORQUE CONTROL OF A PUMA 560 MANIPULATOR ROBOT," 16th IFAC World Congr., vol. 38, no. 1, pp. 229–234, Jan. 2005, doi: 10.3182/20050703-6-CZ-1902.01308.
8. F. Abdessemed and Y. Bazi, "SVM Based Computed Torque for Robot Manipulator Control," 14th IFAC Conf. Methods Models Autom. Robot., vol. 42, no. 13, pp. 593–598, Jan. 2009, doi: 10.3182/20090819-3-PL-3002.00103.
9. L. Tresca, L. Pulvirenti, and L. Rolando, "A Cutting-Edge Energy Management System for a Hybrid Electric Vehicle relying on Soft Actor–Critic Deep Reinforcement Learning," Transp. Eng., vol. 19, p. 100308, Mar. 2025, doi: 10.1016/j.treng.2025.100308.
10. Y. Weng et al., "Reinforcement learning-based tracking control of autonomous underwater vehicles for seafloor platform data collection," Ocean Eng., vol. 328, p. 121047, Jun. 2025, doi: 10.1016/j.oceaneng.2025.121047.
11. Z. Wang et al., "BoilerNet: Deep reinforcement learning-based combustion optimization network for pulverized coal boiler," Energy, vol. 318, p. 134804, Mar. 2025, doi: 10.1016/j.energy.2025.134804.
12. E. Mohammadi et al., "Application of Soft Actor-Critic algorithms in optimizing wastewater treatment with time delays integration," Expert Syst. Appl., vol. 277, p. 127180, Jun. 2025, doi: 10.1016/j.eswa.2025.127180.