113 學年度「機電實作專題成果展」構想書

以多種非線性方法精準控制六軸機械手臂
Precision Control of 6-DOF Robot Arm with Multiple Nonlinear Methods

國立中山大學/機械與機電工程學系/李冠緯/B103025026

國立中山大學/機械與機電工程學系/陳正偉/B103025007

國立中山大學/機械與機電工程學系/蘇廷育/B103025016

國立中山大學/機械與機電工程學系/蔡鎧蔚/B103025019

指導教授 ： 程啟正

主辦單位 ： 國立中山大學機械與機電工程學系

Outline

I. Abstract

This paper presents a comprehensive approach to optimizing the system of a 6-degree-of-freedom (6-DOF) robot arm built by low-cost servo motors using six different control strategies, including internal adaptive PI control, external adaptive PI control, model reference adaptive control (MRAC), sliding mode control (SMC), fuzzy logic PI control, and reinforcement learning (RL). The goal of using these nonlinear methods is to enhance system precision, stability, and dynamic response. Some series of precise movements (e.g. coffee art) controlled by the robot arm will be represented on the day of project exhibition.

Key word: Fuzzy Logic Control, Sliding Mode Control, MRAC, RL, 6-DOF robot arm

II. Motivation and Objective

The 6-DOF robot arm used today are driven by high-precision servo motors, equipped with optical encoders or magnetic encoders, efficient closed-loop control systems, precision gear structure and reducers, digital signal processors (DSP), as well as noise-resistant design and filter – all features lacking in low-cost servo motors like famous MG996R. This low-cost motor is only equipped with a potentiometer and a PI controller to detect position and adjust speed. Additionally, it uses metal gears to increase durability, but these gears' backlash affects control accuracy. Backlash causes slight spatial movement when changing direction, further reducing precision. Ergo, this paper, based on the low precision characteristics of low-cost servo motors, applies six nonlinear algorithms in an attempt to enhance the precision and achieve quick response.

III. Research Method

1. System and experiment

The original system is composed of a robot arm built by six servo motors equipped with PI controller only. There are six joints in the robot arm, including turntable, bicep, forearm, wrist, and gripper. For acquiring actual angle, we use medium precision rotary encoder connecting to the shaft of the servo motor to measure. With every step pass, we record the difference between actual angle and reference angle to calculate the embedded P and I values. Note that the following six different nonlinear algorithms are all tested when the internal potentiometer and PI controller in the MG996R are removed when MCU is connected to Simulink.

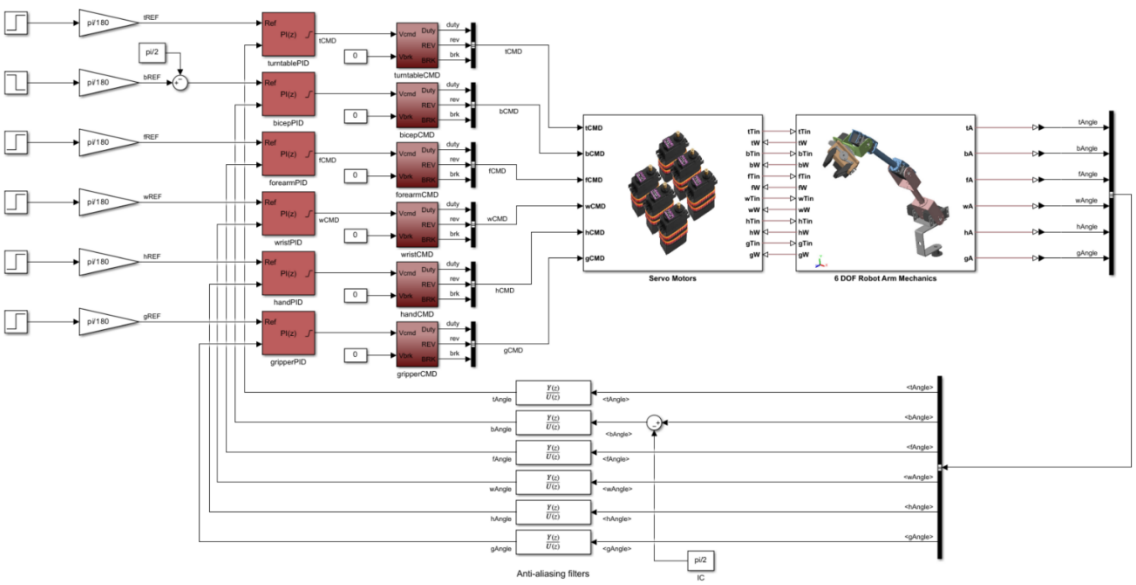

Figure 1.Simulink model for 6-DOF robot arm (with PI controller only)

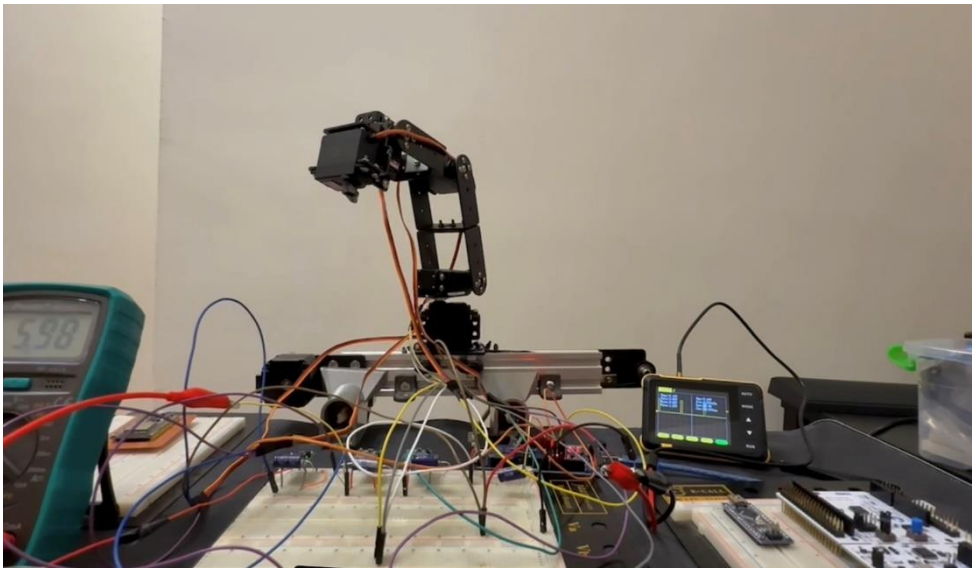

Figure 2.Robot arm with six servo motors

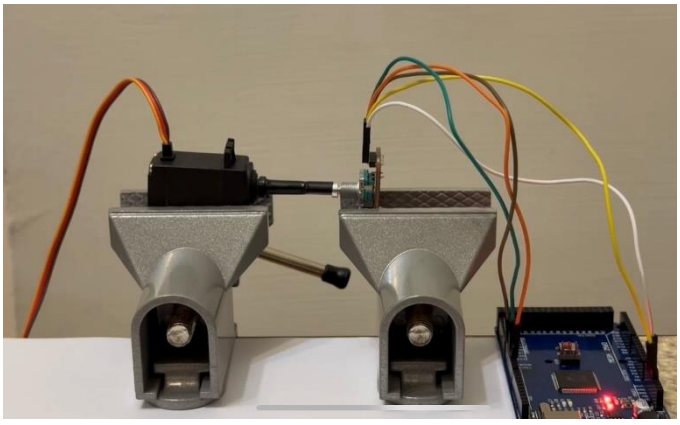

Figure 3.Use rotary encoder to get actual signal

2. Internal Adaptive PI Control

The PI (Proportional-Integral) controller, in its discrete-time form, can be expressed as:

$$u[k] = P \cdot (b \cdot r[k] - y[k]) + I \cdot T_s \sum_{i=0}^{k} e[i] \qquad -(1)$$

where

$u[k]$: control input at step k

P: proportional gain

I: integral gain

b: setpoint weight

$e[k]$: error at step k, defined as $e[k] = r[k] - y[k]$, where $r[k]$ is reference output and $y[k]$ is the system output

T_s: sample time (controller is updated every T_s second)

This internal adaptive PI control method, however, is used when knowing the value of P, I, and b which, for instance, are often stored in non-volatile memory like EEPROM or flash memory within the microcontroller. By this new method, we can dynamically adjust the value of P, I, and b to reduce the difference between actual angle and reference angle resulting in quicker response. Note that the value of b is set as 1 for simplicity.

2.1 Mathematical model

(a) Error definition

Define the tracking error $e[k]$ as:

$$e[k] = r[k] - y[k] \qquad -(2)$$

This error term is used to compute the adjustment required for the gains.

(b) Update of integral error e_{int}

The integral error is defined as:

$$e_{int}[k] = e_{int}[k-1] + e[k] \cdot T_s \qquad -(3)$$

This integral term aims at reducing the steady-state error.

(c) MIT rule

MIT rule is a popular adaptive control method generally used in model reference adaptive control (MRAC) systems. The MIT rule [1] is designed to adjust the parameters of a controller to minimize the error between the system output and reference model output

It is defined as:

$$J = \frac{1}{2}e^2 \qquad -(4)$$

where

J: cost function or performance index

e: error and it is defined as $e = y_m - y$ which is the error between reference model output (y_m) and the system output (y)

To minimize the performance index, the parameter θ (in this method, the proportional gain P) is adjusted in the direction of the negative gradient of J with respect to θ:

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} \qquad -(5)$$

where

γ: adaptation gain, which determines how fast the adaptation occurs

Then with (4), we have

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma e \frac{\partial e}{\partial \theta} \qquad -(6)$$

(d) Adaptive law for proportional gain P

The adaptive law for the proportional gain P follows the MIT rule described in (c):

$$\Delta P[k] = \gamma_P \cdot e[k] \cdot (r[k] - y[k]) \qquad -(7)$$

where

$\Delta P[k]$: change in the proportional gain at step k

γ_P: adaptation rate of P

Note that the product $e[k] \cdot (r[k] - y[k])$ aims ate minimizing the tracking error.

Ergo, the updated proportional gain P is:

$$P_{new} = P_{old} + \gamma_P \cdot e[k] \cdot (r[k] - y[k]) \qquad -(8)$$

(e) Adaptive law for integral gain I

The adaptive law for integral gain I is derived as:

$$\Delta I[k] = \gamma_I \cdot e_{int}[k] \qquad -(9)$$

where

$\Delta I[k]$: change in the integral gain I at step k

γ_I: adaptation rate of I

The product $\gamma_I \cdot e_{int}[k]$ aims at adapting the integral term to reduce cumulative error.

Thus, the updated integral gain is:

$$I_{new} = I_{old} + \gamma_I \cdot e_{int}[k] \qquad -(10)$$

(f) New control command $u[k]$

By using the newly tuning gains P_{new} and I_{new}, the new control command is recalculated:

$$u[k] = P_{new} \cdot e[k] + I_{new} \cdot e_{int}[k] \qquad -(11)$$

This is a term dynamically adjusting the command output of PI controller, which drives the system closer to the desired response.

(g) Anti-windup mechanism

To avoid issues like integral windup which the integral term becomes excessively large, we set a conditional statement:

If the error $e[k]$ is small, the integral term is reset as

$$e_{int}[k] = 0 \text{ if } |e[k]| < 10^{-3} \qquad -(12)$$

This term helps prevent instability due to the accumulation of error when the system is close to the reference.

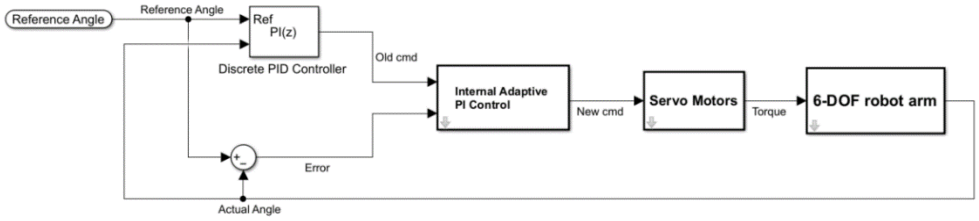

Figure 4.Block diagram of Internal Adaptive PI control

3. External Adaptive PI Control

The governing equation of PI controller is the same as (1) and we set b as 1. But in this method, it is used when the value of P and I are unknown. By guessing them and with proper adaptation rate, er can still reduce the error (which is the difference between reference angle and actual angle) resulting in quicker response and location precision. The reason it is called external is because the PI controller block in Simulink is set as external.

3.1 Mathematical model

(a) Error definition

Ther error calculation is the same as (2), which is the difference between reference output and system output.

(b) Adaptive update [2]

- Proportional gain P update

$$P_{new}[k] = P_{old}[k] + \gamma_P \cdot e[k] \qquad -(13)$$

where

γ_P: adaptation rate for proportional gain

- Integral gain I update

$$I_{new}[k] = I_{old}[k] + \gamma_I \cdot e[k] \qquad -(14)$$

where

γ_I: adaptation rate for integral gain

(c) Restriction

If the error turns out to be zero, the gains remain unchanged.

$$\begin{cases} P_{new} = P_{old} \\ I_{new} = I_{old} \end{cases} \text{ if } e = 0 \qquad -(15)$$

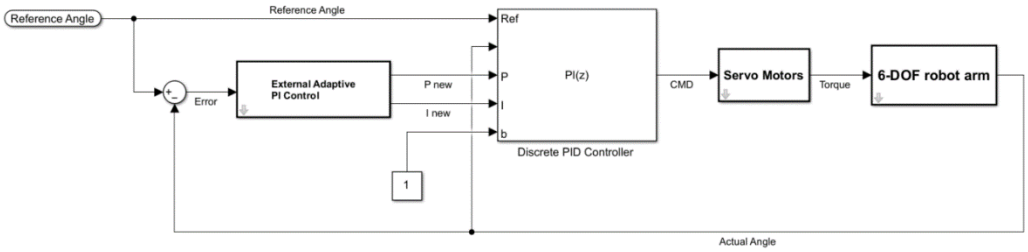

Figure 5.Block diagram of External Adaptive PI Control

4. Model Reference Adaptive Control (MRAC) [1]-[3], [5]

The governing equation of PI controller is still the same as (1) and we set b as 1. By using the MIT rule mentioned in internal adaptive PI control, we get the reference (which is the value of previous error) and be able to reduce the error contributing to quicker response.

4.1 Mathematical model

(a) Reference model dynamics

In MRAC, the reference model provides the desired trajectory for the system output.

Denote:

$r[k]$: reference signal at time step k

$y_m[k]$: output of the reference model

e_m: error of reference model and is denoted $e_m[k] = y_m[k] - y[k]$

And in out setup, we assume $y_m[k] = r[k]$, which implies that we want the system output to match the reference output.

(b) Adaptive laws for P and I

Using the MIT rule, the cost function is defined in (4). The goal is to adapt P and I to minimize J.

Therefore, the adaptive laws for P and I are defined using the adaptation rates:

$$P_{new}[k] = P_{old}[k] + \gamma_P \cdot e_m[k] \cdot \frac{\partial e_m}{\partial P} \qquad -(16)$$

$$I_{new}[k] = I_{old}[k] + \gamma_I \cdot e_m[k] \cdot \frac{\partial e_m}{\partial I} \quad -(17)$$

where

γ_P: adaptation rate of proportional gain P

γ_I: adaptation rate of integral gain I

(c) Partial derivative calculations

The adaptation law requires calculating the partial derivative of the error with respect to each gain.

For proportional gain P,

$$\frac{\partial e_m}{\partial P} = -e[k] \quad -(18)$$

where $e[k] = r[k] - y[k]$

For integral gain I,

$$\frac{\partial e_m}{\partial I} = -\sum_{i=0}^{k} e[i] \cdot T_s \quad -(19)$$

(d) Adaptive update

By incorporating partial derivatives (18) and (19) into (16) and (17), we have

$$P_{new}[k] = P_{old}[k] + \gamma_P \cdot e_m[k] \cdot e[k] \quad -(20)$$

$$I_{new}[k] = I_{old}[k] + \gamma_I \cdot e_m[k] \cdot \sum_{i=0}^{k} e[i] \cdot T_s \quad -(21)$$

(e) Constraints on gains

To ensure the gains do not become negative, we add a condition to keep them non-negative

$$\begin{cases} P_{new}[k] = \max(0, P_{new}[k]) \\ I_{new}[k] = \max(0, I_{new}[k]) \end{cases} \quad -(22)$$

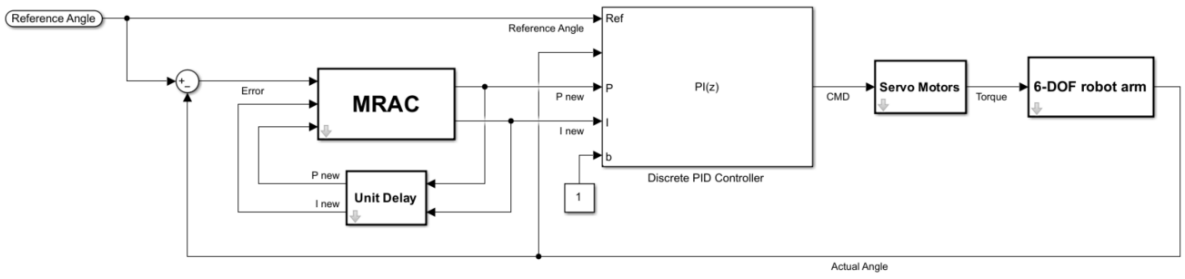

Figure 6.Block diagram of MRAC

5. Sliding Mode Control (SMC)

Sliding mode control is a variable structure control strategy that drives the system to follow a pre-defined sliding surface. It achieves by altering the control gains depending on the state of the system and its deviation from the desired trajectory

The governing equation of PI controller is still the same as (1) and we set b as 1. The goal here is to use the SMC algorithm to adjust the proportional gain P and integral gain I in real-time, ensuring robust performance and fast convergence.

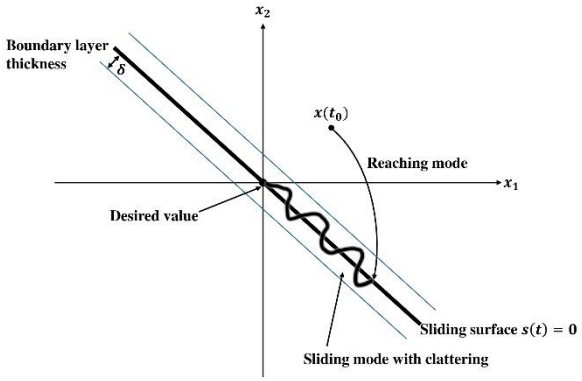

Figure 7.Phase portrait of Sliding Mode Control

5.1 Mathematical model

(a) Discrete sliding surface [2]

$$\mathrm{s}[k] = e[k] + \lambda e_{int}[k] \qquad -(23)$$

where

$e[k] = r[k] - y[k]$: error between the reference output and system output at step k

$e_{int}[k] = e_{int}[k-1] + e[k] \cdot T_s$: integral of error over time, which represents accumulated error

λ: positive constant determining the weight of the integral term in the sliding surface

(b) Control law [2], [4], [6]

- Equivalent control

  This is the control output that would keep the system keep the system on the sliding surface if there were no disturbances.

- Switching control

  This component compensates for disturbances and uncertainties and ensures the system reaches and stays on the sliding surface. It is often defined as a sign function.

$$u_{sw}(t) = -k \cdot sgn(s(t)) \qquad -(24)$$

where

k: positive gain that help mitigate the abrupt changes, thus reducing the clattering

$sgn(s)$: sign of the sliding surface

(c) Adjust gains using SMC control law

By using components in (b), we have

$$P_{new} = P_{old} + \gamma_P \cdot sgn(s[k]) + \eta_P \cdot s \quad -(25)$$

$$I_{new} = I_{old} + \gamma_I \cdot sgn(s[k]) + \eta_I \cdot s \quad -(26)$$

where

γ_P, γ_I: adaptation rates for the gains, which determine how quickly the gains adapt based on the sliding surface

η_P, η_I: additional gain term to ensure that the adjustment responds to the magnitude of the sliding surface

(d) Anti-windup condition

To prevent adaptive gains become excessively large, we set

$$\begin{cases} P_{new}[k] = \max(0, \min(P_{new}[k], 100)) \\ I_{new}[k] = \max(0, \min(I_{new}[k], 100)) \end{cases} \quad -(27)$$

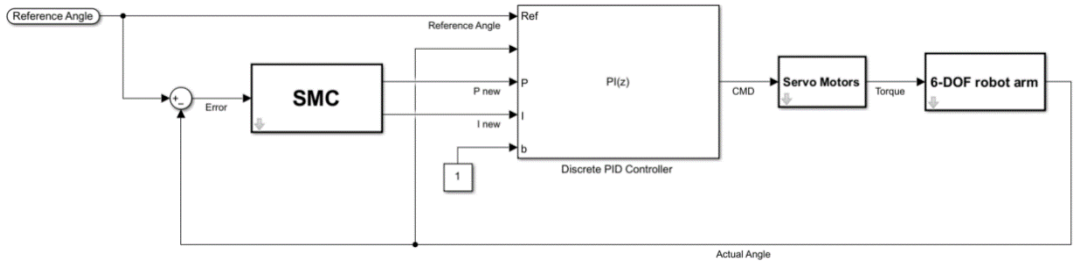

Figure 8.Block diagram of SMC

6. Fuzzy Logic PI Control

Fuzzy logic PI control is a method involving mathematical derivation that includes defining membership functions, applying fuzzy inference rules, and defuzzification. The controller we used is Mamdani Type 1 fuzzy inference with triangular membership functions (MFs), and there are 25 rules in total for inference [4].

For the algorithm, I define two inputs and two outputs.

Table 1.Information of inputs and outputs

	Input		Output	
Symbol	e	Δe	P_{new}	I_{new}

Name	Error	Delta_e	P_new	I_new
Range	[-10, 10]	[-5, 5]	[0, 50]	[0, 10]
Number of MFs	5			
Type of MFs	Triangle			

6.1 Mathematical model

(a) Membership functions

For each variable (inputs and outputs), triangular membership functions are used. And it is defined as:

$$\mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \qquad -(28)$$

where

a, b, c: points defining the triangle

x: input to the membership function

$\mu(x)$: the degree of membership

For each variable (inputs and outputs), we define five MFs as in the Table.2

Table 2.Settings of MFs in inputs and outputs

Input MFs	Meaning	Output MFs	Meaning
NL	Negative Large	VL	Very Low
NS	Negative Small	L	Low
ZE	Zero	M	Medium
PS	Positive Small	H	High
PL	Positive Large	VH	Very High

For simplicity, we evenly distribute each MFs for their range as in Figure.8

(b) Fuzzification

This step involves determining the degree to which the inputs (e and Δe) belong to each of the MFs. For each input, calculate the membership value for each of the five triangular MFs.

(c) Rule evaluation

The fuzzy inference system uses 25 rules to map the inputs to the outputs as in Table.3

Table 3.Rules for evaluation

	Error (e)	Delta_e (Δe)	P_new (P_{new})	I_new (I_{new})
Rule 1	NL	NL	VH	L
Rule 2	NL	NS	H	M
Rule 3	NL	ZE	M	M
Rule 4	NL	PS	M	H
Rule 5	NL	PL	L	VH
Rule 6	NS	NL	VH	L
Rule 7	NS	NS	H	M
Rule 8	NS	ZE	M	M
Rule 9	NS	PS	L	H
Rule 10	NS	PL	VL	VH
Rule 11	ZE	NL	M	L
Rule 12	ZE	NS	M	M
Rule 13	ZE	ZE	M	M
Rule 14	ZE	PS	M	M
Rule 15	ZE	PL	M	H
Rule 16	PS	NL	L	VH
Rule 17	PS	NS	L	H
Rule 18	PS	ZE	M	M
Rule 19	PS	PS	H	M
Rule 20	PS	PL	VH	L
Rule 21	PL	NL	L	VH
Rule 22	PL	NS	L	H
Rule 23	PL	ZE	M	M
Rule 24	PL	PS	H	M
Rule 25	PL	PL	VH	L

The calculation would include determining the firing strength and then apply the rule. The output fuzzy set is scaled by the firing strength.

(d) Aggregation

After evaluating all 25 rules, the next step is to aggregate the outputs from all rules. The aggregation is done using the maximum operator for all rules contributing to each output (P_{new}, I_{new}).

For each output, combine the contributions from all rules using maximum operator.

$$\mu_{aggregated}(x) = \max\left(\mu_{rule1}(x), \mu_{rule2}(x), \ldots\ldots, \mu_{rule25}(x)\right) \quad -(29)$$

where

$\mu_{aggregated}(x)$: aggregated MF for the output

(e) Defuzzification

This step converts the aggregated fuzzy into a crisp value. We use the centroid defuzzification method.

For each output (P_{new} and I_{new}) in its own range,

$$x_{crisp} = \frac{\int x \cdot \mu_{aggregated}(x)dx}{\int \mu_{aggregated}(x)dx} \quad -(30)$$

where

x_{crisp}: defuzzified output value, which is the value of P_{new} or I_{new}

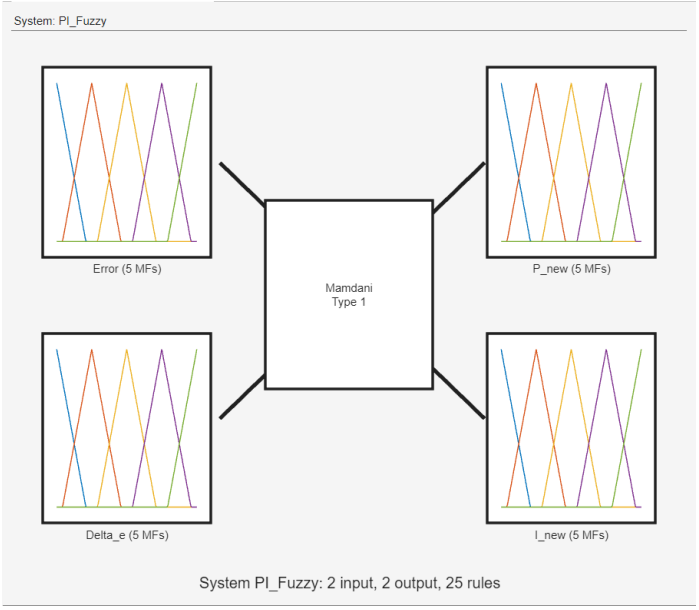

Figure 9.Mamdani method with two inputs and two outputs

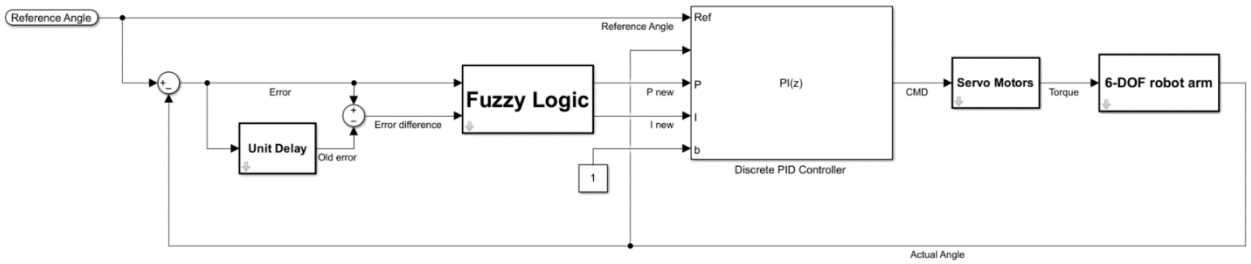

Figure 10.Block diagram of Fuzzy Logic PI Control

7. Reinforcement Learning (RL)

This method is a type of machine learning in which an agent learns how to make decisions by interacting with an environment to achieve a specific goal. It focuses on learning from the consequences of actions, with the goal of maximizing a reward signal over time. We choose deep deterministic policy agent (DDPG) to tune the PI controller gains [6]. Note that the governing equation of PI controller is still the same as (1) and we set b as 1.

7.1 Reinforcement learning process

(a) State, action, and reward

- State (Observation)

  We define observation as the system error at time step k

$$e[k] = r[k] - y[k] \quad -(31)$$

- Action

  The action taken by the RL agent is the adjustment of the PI controller gains.

$$a[k] = [\Delta P, \Delta I] \quad -(32)$$

  where ΔP and ΔI represent the incremental adjustments to the proportional and integral gain, respectively

- Reward

  The reward is defined to penalize the squared error, promoting error minimization.

$$R[k] = -e[k]^2 \quad -(33)$$

(b) Update equations for gains

The updated gains at each time step k are given by

$$P_{new} = P_{old} + \Delta P \quad -(34)$$

$$I_{new} = I_{old} + \Delta I \quad -(35)$$

(c) Actor-Critic Framework

- Actor network

  It is responsible for determining the action. Given the current state (error), it outputs the adjustments to the gains.

- Critic network

  It evaluates the quality of the action taken by the actor. It estimates the value function $Q(s, a)$ representing the expected cumulative future reward from taking action a in state s.

(d) Exploration

The RL agent uses Ornstein-Uhlenbeck noise for exploration, allowing it to explore different gain values to improve learning.

7.2 Mathematical model

(a) Bellman equation for policy evaluation

The critic network aims ate minimizing the temporal difference (TD) error given by the Bellman equation:

$$\delta = R[k] + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k) \qquad -(36)$$

where

δ: TD error

$R[k]$: immediate reward

γ: discount factor

$Q(s_k, a_k)$: current estimate of the value function for state s_k and action a_k

The critic network is trained to minimize δ^2.

(b) Policy update using the actor network

The actor network aims at finding the optimal policy by maximizing the expected return. The gradient of the policy is defined as:

$$\nabla_\theta J \approx \mathbb{E}[\nabla_a Q(s,a) \nabla_\theta \mu(s)] \qquad -(37)$$

where

$\mu(s)$: the policy (actor network) that determines the action a

$\nabla_\theta \mu(s)$: the gradient of the policy with respect to the actor parameter θ

The actor network parameters are updated in the direction of the gradient to improve the expected return.

(c) Target network

To stabilize training, target networks are used for both the actor and critic. These target networks are slowly updated to track the learned networks:

$$\theta_{target} \leftarrow \tau\theta + (1-\tau)\,\theta_{target} \qquad -(38)$$

where

τ: small value that ensures a slow update, reducing oscillations and instability in training

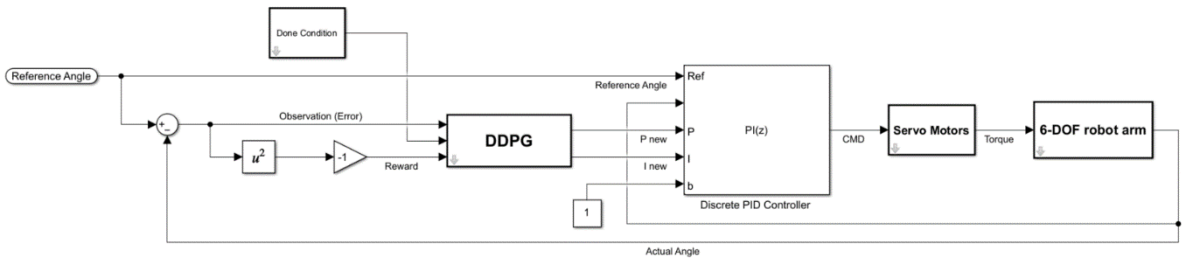

Figure 11.Block diagram of RL DDPG tuning PI controller

IV. Result and Discussion

1. Step response

With the usage of six nonlinear algorithms, it can be observed that the system has a faster response, reaches the specified position more quickly, and becomes more stable.

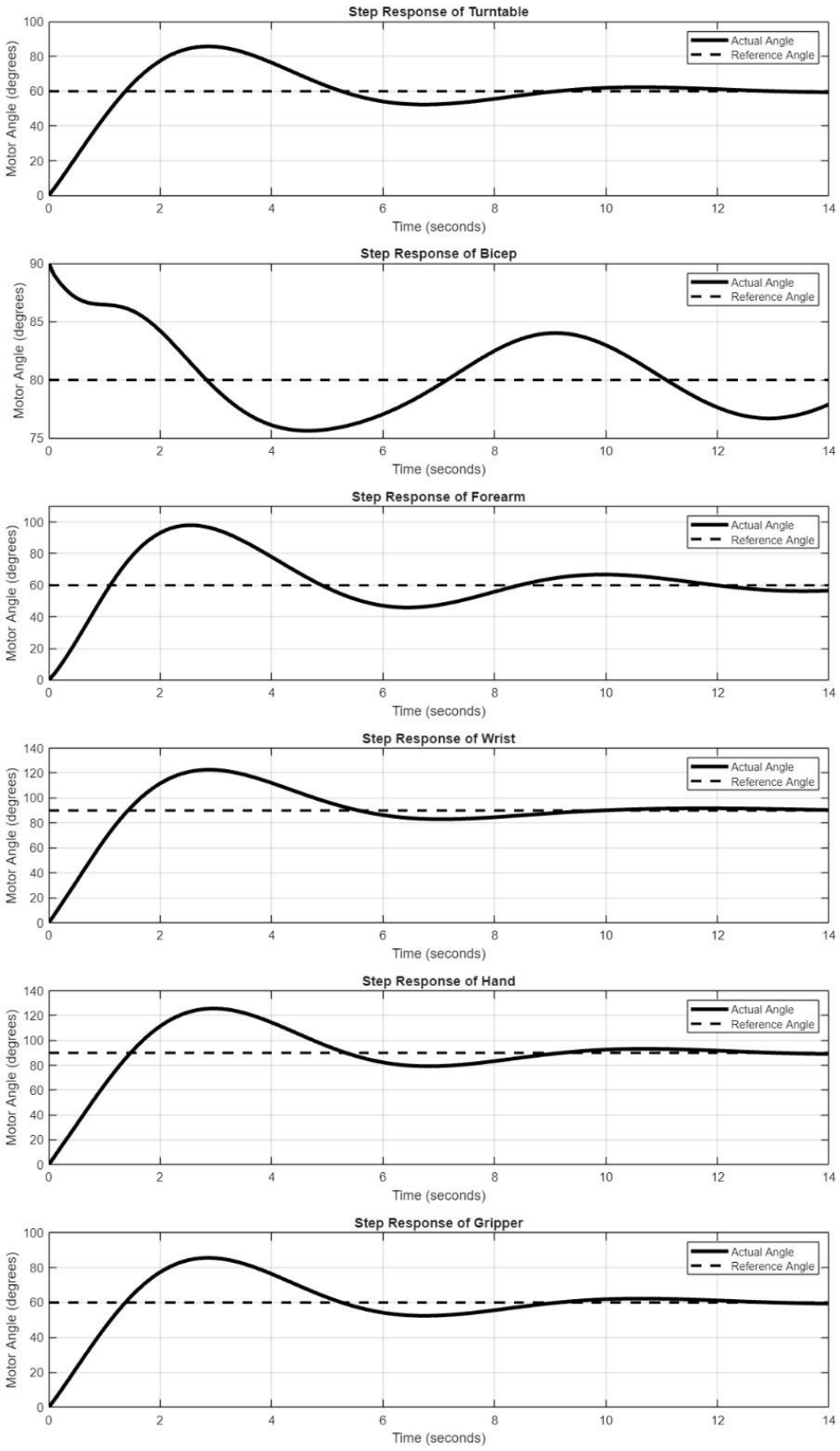

Figure 12.Step response with PI controller only

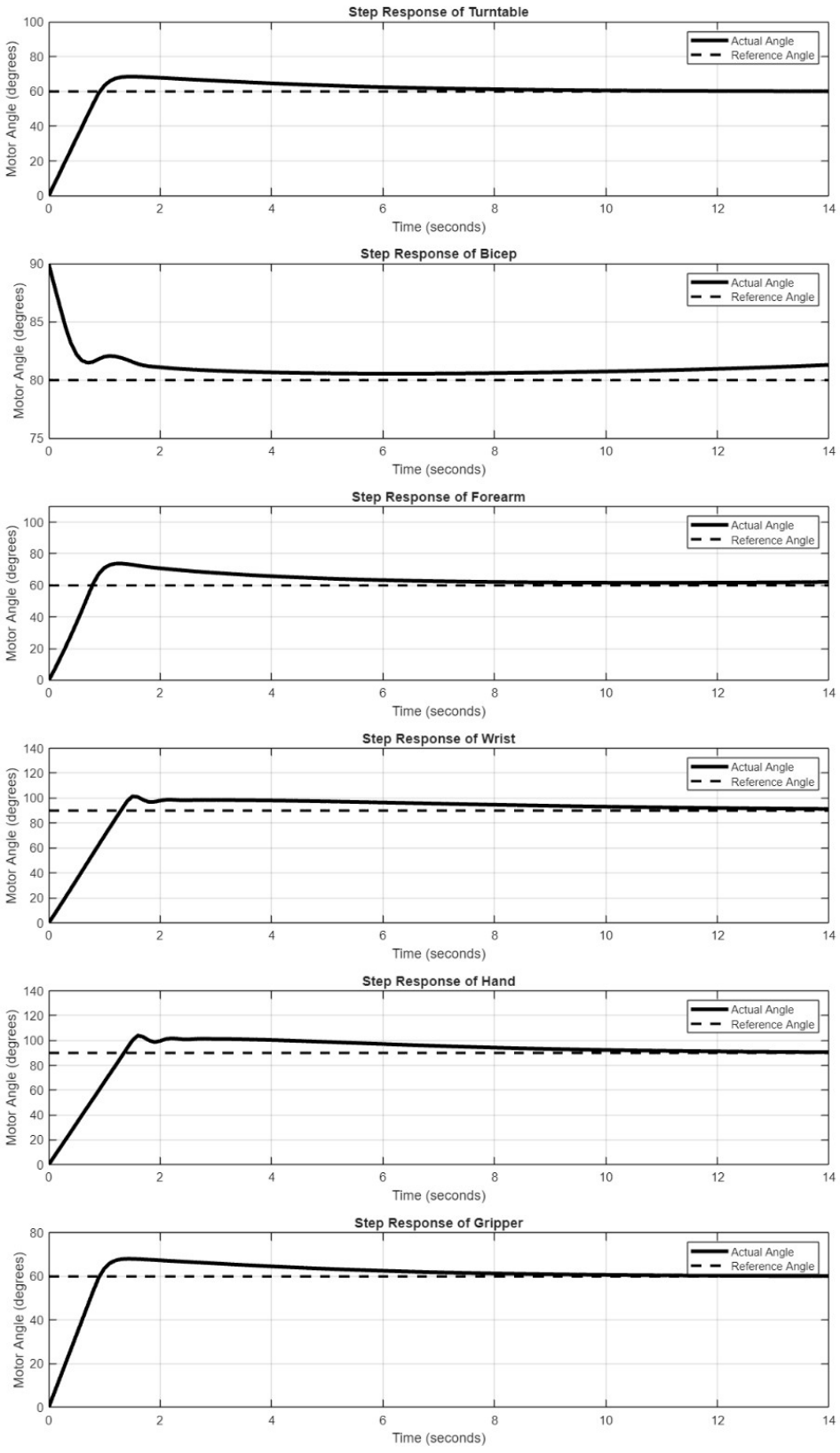

Figure 13.Step response with Internal Adaptive PI Control

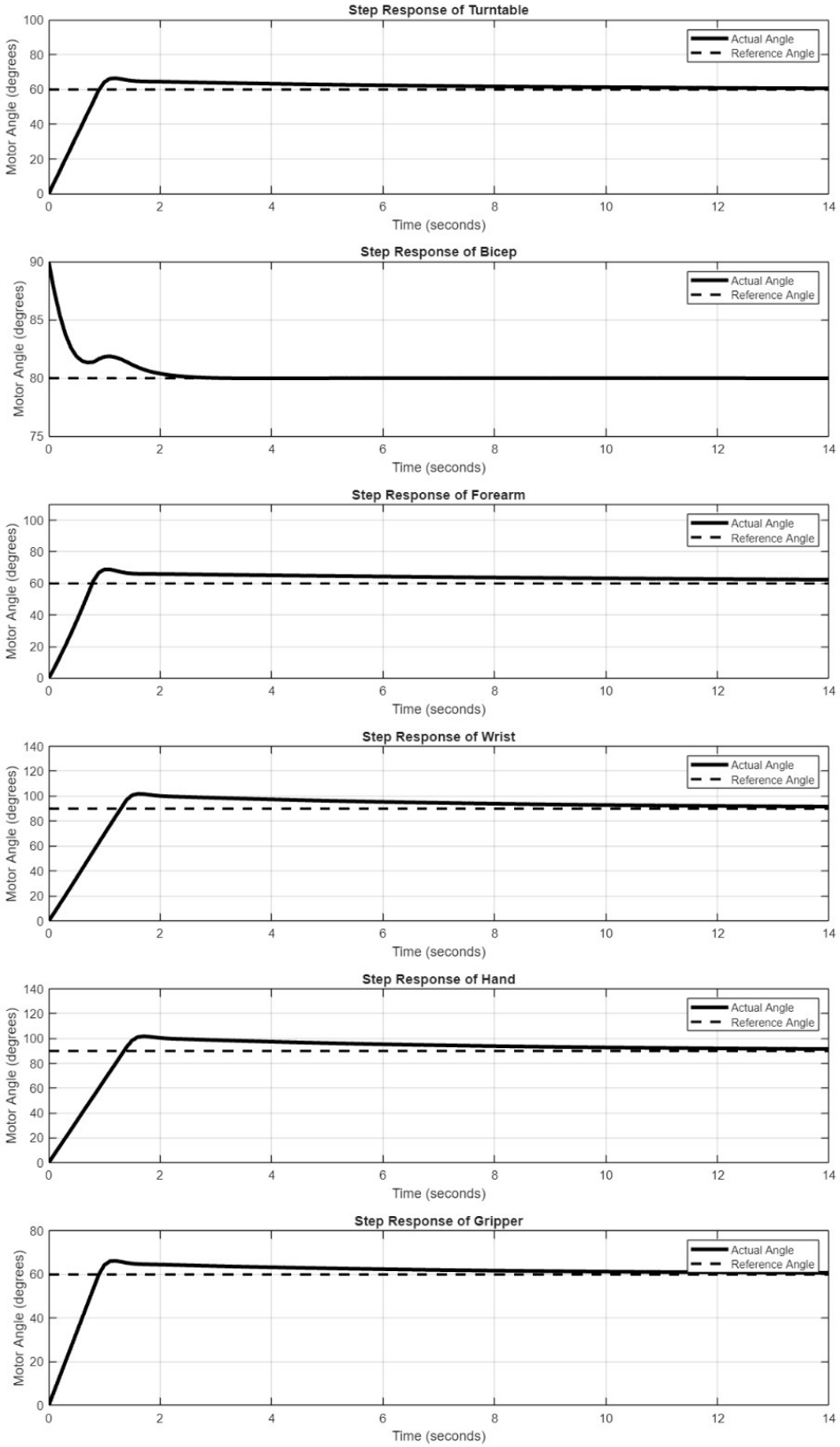

Figure 14.Step response with External Adaptive PI Control

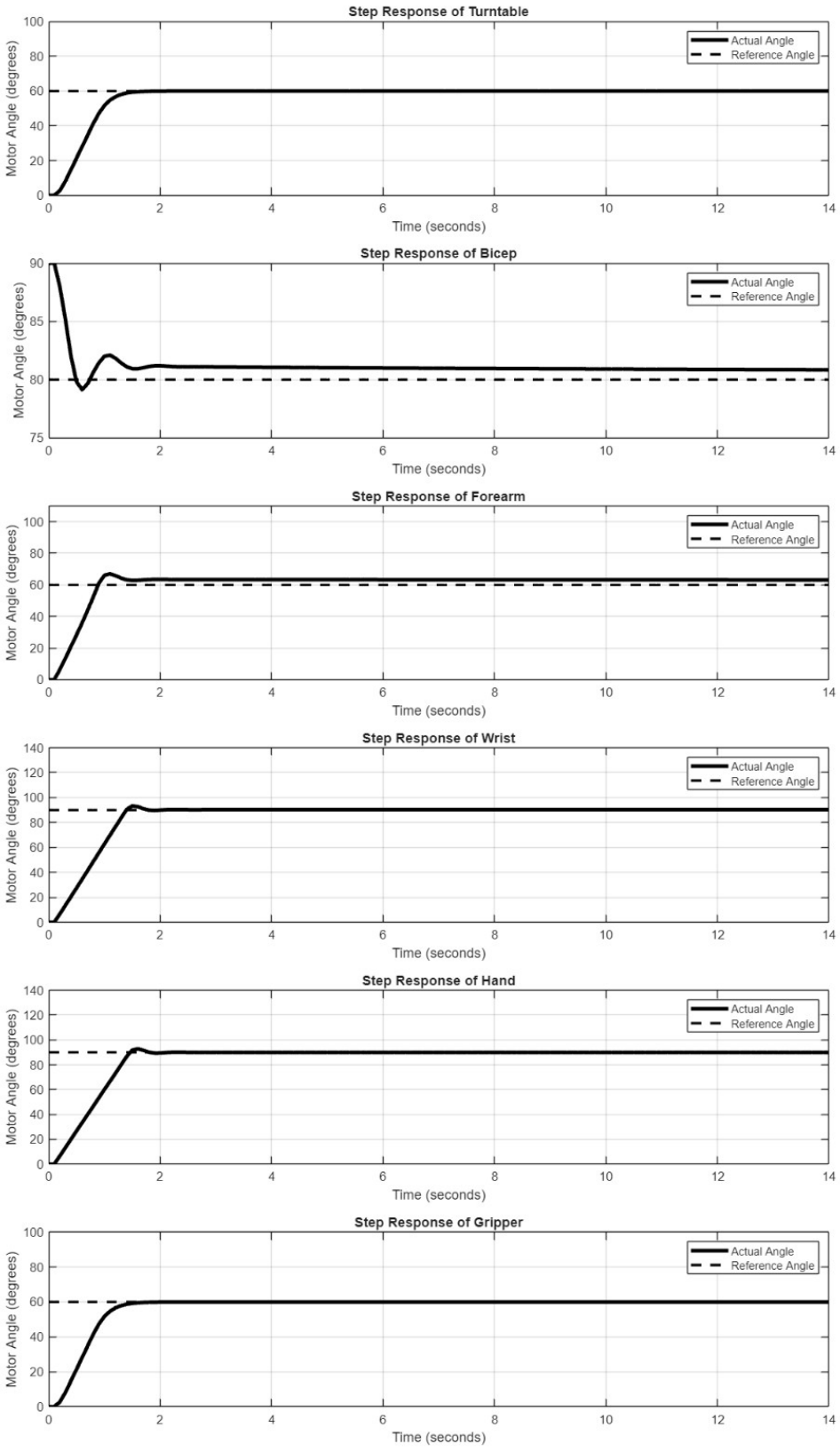

Figure 15.Step response with MRAC

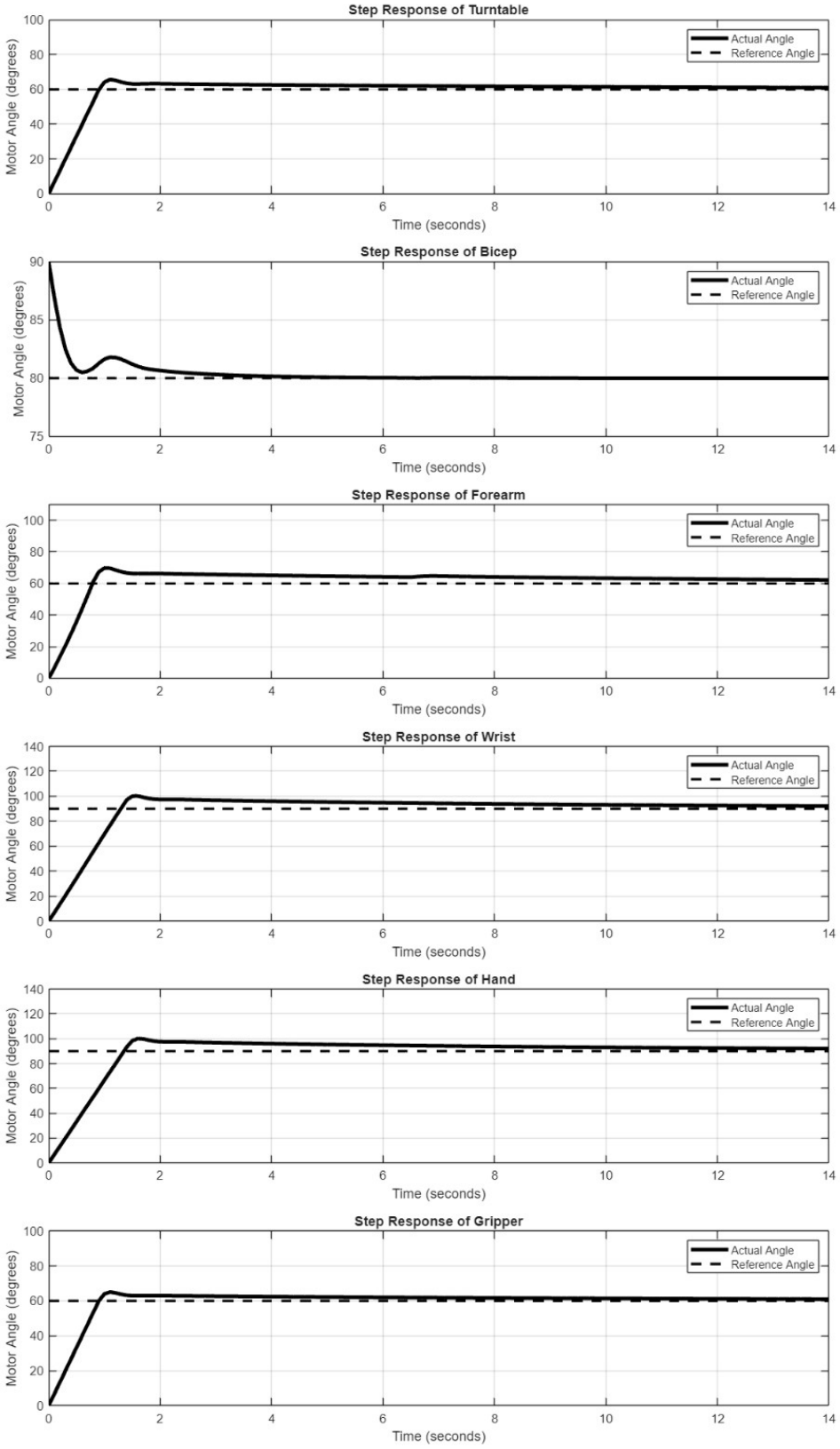

Figure 16.Step response with SMC

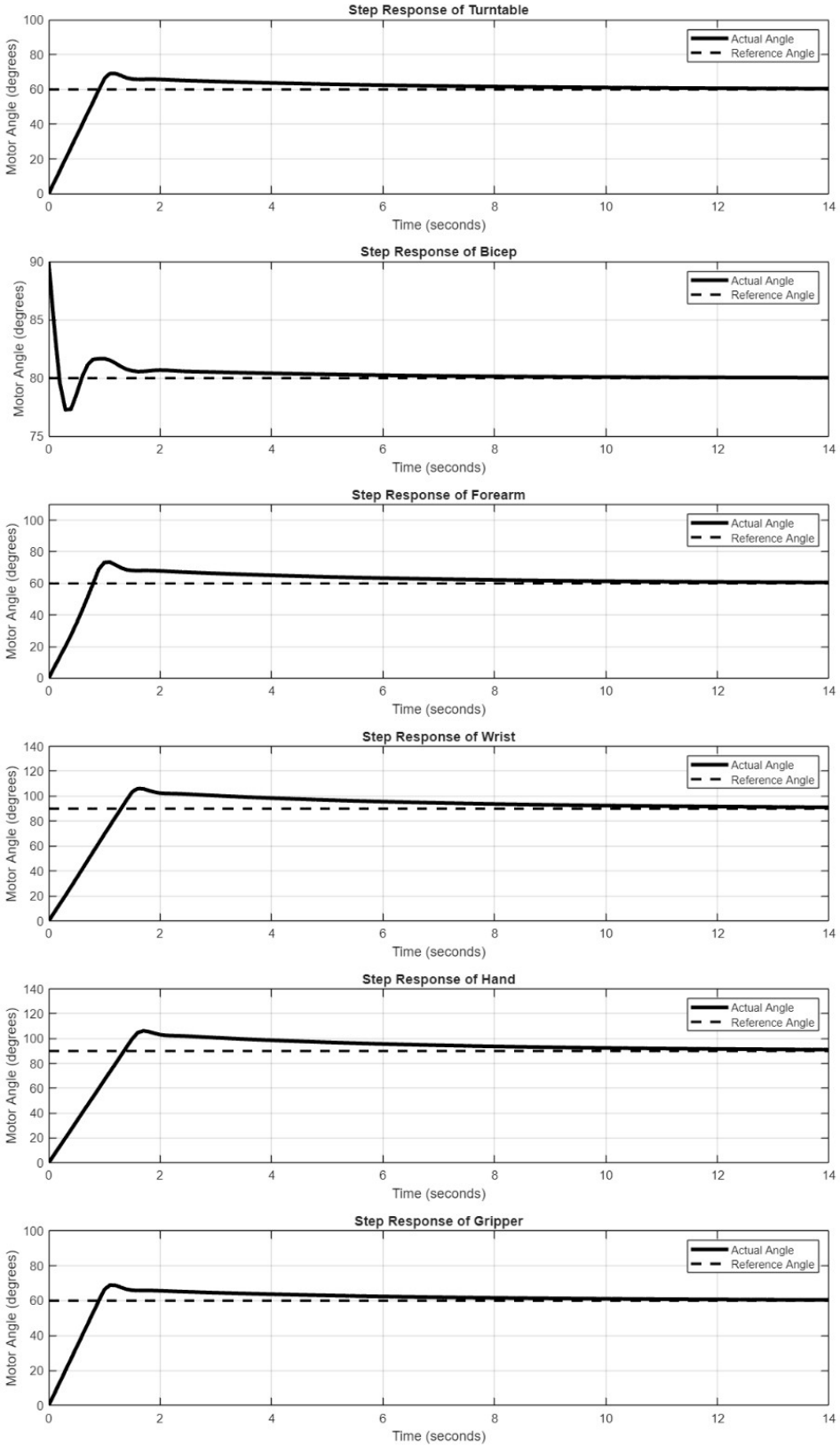

Figure 17.Step response with Fuzzy Logic PI Control

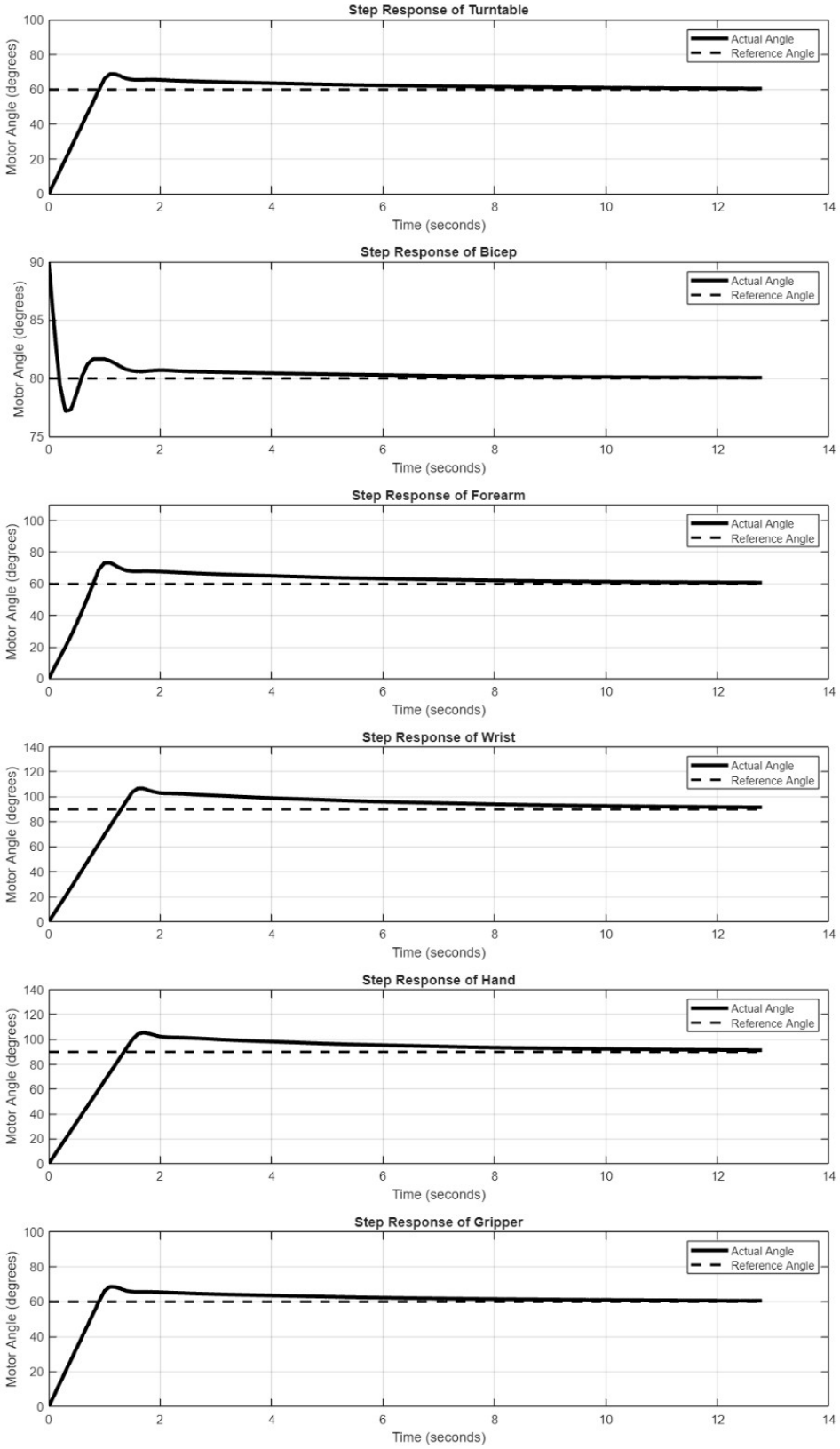

Figure 18.Step response with RL DDPG

2. Settling time

It is known that a lower settling time is preferred as it indicates the system reaches steady state more quickly. The following table shows the data of settling time (second).

Table 4.Settling time in every joints by applying different methods

Method / Joint	Turntable	Bicep	Forearm	Wrist	Hand	Gripper
Original PI	11.96	NaN	NaN	9.17	11.97	11.97
Internal Adaptive PI	8.06	1.49	NaN	12.53	10.84	8.28
External Adaptive PI	10.39	1.31	NaN	13.10	12.94	10.37
MRAC	1.40	1.22	NaN	1.65	1.68	1.90
SMC	11.50	1.31	NaN	NaN	NaN	11.55
Fuzzy Logic PI	9.38	1.05	10.84	11.54	11.50	9.37
RL	9.29	1.04	10.77	11.98	11.22	9.18

- Observations

(a) The original PI controller has a higher settling time for almost all joints, which shows it is not as optimized as the tuned methods. And actually, it didn't reach steady state, which means that the settling time of original PI controller is equivalent to transient time.

(b) The internal and external adaptive PI control method show significant reduction in settling time compared to the original PI. This may indicate that adaptation helps in faster stabilization.

(c) MRAC generally has the lowest settling time across the joints, which suggests a highly adaptive and well-tuned system.

(d) SMC shows better result than the original PI but not as effective as MRAC for certain joints.

(e) Fuzzy logic PI and reinforcement learning also reduce settling time effectively.

(f) The NaN values for the joint forearm and other joints may imply that the system was not able to stabilize within the measured criteria, possibly indicating instability or persistence oscillations.

3. Rise time

A lower rise time indicates the system reaches near its steady state faster. The following table shows rise time (second) of all joints applied by different methods.

Table 5.Rise time in every joints by applying different methods

Method / Joint	Turntable	Bicep	Forearm	Wrist	Hand	Gripper
Original PI	1.0757	0.00025	0.8512	1.1079	1.1533	1.0741
Internal Adaptive PI	0.7251	0.0035	0.6174	1.0293	1.806	0.7207
External Adaptive PI	0.7239	0.00221	0.6207	1.0288	1.0807	0.72
MRAC	0.8163	0.0045	0.6254	1.0286	1.0807	1.0205
SMC	0.72388	0.00235	0.62561	1.0296	1.0807	0.71995
Fuzzy Logic PI	0.72441	0.00415	0.64334	1.0335	1.0808	0.71995
RL	0.72442	0.0015	0.64356	1.0336	1.0808	0.71995

- Observations

(a) The original PI has longer rise time, especially for the turntable and wrist joints, showing that it us slower in reacting to changes.

(b) Internal and external adaptive PI controller show a reduction in rise time for almost all joints, which indicates a quicker initial response.

(c) MRAC shows a slightly higher rise time for the turntable, but overall, it is within acceptable limits. This balance may help to minimize overshoot and ensuring stability.

(d) Fuzzy logic PI and RL controllers perform well, with very similar rise time across joints, showing their effectiveness in reducing delay.

(e) SMC has the lowest rise time for bicep, but for other joints, it is similar to MRAC, indicating that it might be a better solution for some specific applications.

4. Percent overshoot

Lower overshoot is better as it means that the system is less inclined to exceed desired value. Percent overshoot (P.O) is defined as:

$$P.O = \frac{Peak\ value - Final\ value}{Final\ value} \times 100\% \qquad -(39)$$

The following table shows percent overshoot (%) of six joints applied by different methods.

Table 6.Percent overshoot in every joints by applying different methods

Method / Joint	Turntable	Bicep	Forearm	Wrist	Hand	Gripper
Original PI	44.37	15.53	73.01	35.69	41.00	44.06
Internal Adaptive PI	13.95	10.69	18.84	11.09	14.72	13.07
External Adaptive PI	9.51	12.51	11.06	11.24	11.26	9.10
MRAC	2.30E-4	11.31	6.77	3.34	3.25	0.0023

SMC	7.61	12.52	12.20	8.78	8.84	6.74
Fuzzy Logic PI	14.43	12.45	21.27	16.55	16.72	14.06
RL	13.87	12.40	20.74	16.65	15.64	13.42

- Observations

(a) The original PI controller has very high overshoot, particularly for the forearm and gripper, indicating poor tuning and potential for instability.

(b) Internal and external adaptive PI controllers significantly reduce overshoot for all joints, implying better control.

(c) MRAC shows extremely low overshoot, with values close to zero for most joints, indicating very well-behaved transient response.

(d) SMC and fuzzy logic PI controller also manage to maintain overshoot at a lower level compared to the original PI controller, though SMC is particularly effective for joints like hand and forearm

(e) RL show varying levels of overshoot across joints, with some joints being slightly higher than other adaptive methods, may indicate the need for further training or adjustments.

5. Characteristics of each method

(a) Best overall method – MRAC

It shows the lowest settling time, low rise time, and minimal overshoot, making it a good choice for overall performance improvement.

(b) Most balanced performance

Fuzzy logic PI and RL also achieve balanced performance, offering a good reduction in all data.

(c) SMC performance

It performs quite well in terms of reducing overshoot and rise time but does not always reduce the settling time compared to other adaptive methods.

6. Prospects

(a) Combination of controllers

Try to investigate the potential benefits of combining control methods to utilize the strength of each approach. It could provide enhanced robustness and adaptability.

(b) Improvement of RL training

By enhancing the training episodes, the result may be more accurate. Also, we can implement advanced RL algorithms to improve learning efficiency and robustness. Simulation with different reward functions may minimize overshoot, rise time, and settling time.

(c) Integrate with optimization algorithms

We can enhance the performance by integrating with optimization algorithms like Genetic Algorithms (GA) or Particle Swarm Optimization (PSO) to automate the process of tuning the PI controller parameters.

(d) IC design for newer low-cost internal controller for RC servo motor

While the new IC designing with six different methods may cost lots of effort, however, MG996R act as a popular student-used servo motors for projects across the globe, it is valuable when the control is improved.

V. Acknowledgements

Special thanks to Dr. Chi-Cheng Cheng for taking the time over the past year to guide us in relevant knowledge, helping us gain a deeper understanding in our professional field, which will be beneficial for our future career development. We hereby express our sincere gratitude.

VI. References

[1] Ehsani, M. S., 09-12 September 2007 "Adaptive Control of Servo Motor by MRAC Method," 2007 IEEE Vehicle Power and Propulsion Conference, IEEE, Arlington, TX, USA

[2] J.-J. E. Slotine, W. L., 1991, Applied Nonlinear Control, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632.

[3] Khalil, H. K., 2002, 1996, Nonlinear Systems Third Edition, Prentice Hall, Upper Saddle River, NJ 07458.

[4] Reza Shahnazi, H. M. S., Naser Pariz, March 2008, "Position Control of Induction and DC Servomotors: A Novel Adaptive Fuzzy PI Sliding Mode Control," IEEE Transactions on Energy Conversion, 23(1), pp. 138-147.

[5] S. Maiti, C. C., Y. Hori, Minh C. Ta, February 2008, "Model Reference Adaptive Controller-Based Rotor Resistance and Speed Estimation Techniques for Vector Controlled Induction Motor Drive Utilizing Reactive Power," IEEE Transactions on Industrial Electronics, 55(2), pp. 594-601.

[6] Teng Long, E. L., Yunqing Hu, Lei Yang, Junfeng Fan, Zize Liang, February 2021, "A Vibration Control Method for Hybrid-Structured Flexible Manipulator Based on Sliding Mode Control and Reinforcement Learning," IEEE Transactions on Neural Networks and Learning Systems, 32(2), pp. 841-852.