

# Investigating Adaptive Tuning of Assistive Exoskeletons using Offline Reinforcement Learning: Challenges and Insights

Yasin Findik, Christopher Coco, and Reza Azadeh

**Abstract**— Assistive exoskeletons have shown great potential in enhancing mobility for individuals with motor impairments, yet their effectiveness relies on precise parameter tuning for personalized assistance. In this study, we investigate the potential of offline reinforcement learning for optimizing effort thresholds in upper-limb assistive exoskeletons, aiming to reduce reliance on manual calibration. Specifically, we frame the problem as a multi-agent system where separate agents optimize biceps and triceps effort thresholds, enabling an adaptive and data-driven approach to exoskeleton control. Mixed Q-Functionals (MQF) is employed to efficiently handle continuous action spaces while leveraging pre-collected data, thereby mitigating the risks associated with real-time exploration. Experiments were conducted using the MyoPro 2 exoskeleton across two distinct tasks involving horizontal and vertical arm movements. Our results indicate that the proposed approach can dynamically adjust threshold values based on learned patterns, potentially improving user interaction and control, though performance evaluation remains challenging due to dataset limitations.

## I. INTRODUCTION

Assistive robotics, particularly powered exoskeletons, have emerged as a promising technology for enhancing human mobility, whether by helping individuals with disabilities, supporting the elderly in daily activities, or improving physical performance in demanding tasks [1], [2], [3]. Effective control in these systems depends on the ability to interpret user intentions and adapt to user learning and changes in physical conditions (e.g., fatigue) [4]. However, most exoskeleton controllers are pre-configured and remain static after initial calibration, limiting their responsiveness over time [5], [6]. While low-level controllers, such as PID controllers, effectively regulate movement, their high-level parameters often require manual tuning to align with natural human movement patterns [7].

This tuning process often requires expert intervention, involving repeated testing, fine-tuning, and validation to ensure the exoskeleton responds naturally to the user's movements [8]. Since user needs can change over time due to factors like fatigue, motor learning, or varying environments, frequent recalibration may be necessary, making the process time-consuming, resource-intensive, and difficult to implement on a larger scale [9]. Addressing these challenges requires advanced modeling approaches capable of capturing the dynamic interplay between users and robotic systems. Unlike traditional rigid-body models, assistive robots function in complex, human-in-the-loop environments, requiring

Persistent Autonomy and Robot Learning (PeARL) Lab, University of Massachusetts Lowell, Lowell, MA 01854, USA. {yasin.findik, christopher.coco}@student.uml.edu, reza.azadeh@uml.edu



Fig. 1: Overview of the MyoPro 2 device, highlighting the placement of sEMG sensors.

data-driven techniques for more adaptive and responsive control [10]. Consequently, research has increasingly focused on developing automated hyper-parameter tuning methods to enhance the adaptability and usability of these systems [11].

Building on these advancements, we explore a reinforcement learning-based approach to automate parameter tuning and reduce the reliance on manual recalibration. To investigate this, we use the MyoPro 2 [12] device, a 2-DoF exoskeleton, as our experimental platform. Designed to restore functionality in individuals with paralyzed or weakened upper limbs, this device assists elbow and hand movements, as shown in Fig. 1. To enhance user experience, it is essential to properly tune the robot's hyper-parameters, particularly the biceps and triceps effort thresholds, in a way that adapts to individual users and task variations. We approach this problem through a multi-agent framework, where each agent is responsible for adjusting a specific threshold across two tasks. Specifically, we investigate the potential of offline multi-agent reinforcement learning that leverages Mixed Q-Functionals (MQF) [13], enabling automated, task-specific parameter adaptation without requiring manual intervention. This approach aims to enhance the system's adaptability and responsiveness, ultimately improving user experience and control precision.

## II. EXOSKELETON OVERVIEW

### A. Device Description

The MyoPro 2 (Fig. 1) is a motorized upper-limb exoskeleton designed to assist users in performing daily activities. It features two motor-controlled degrees of freedom (DoF): the

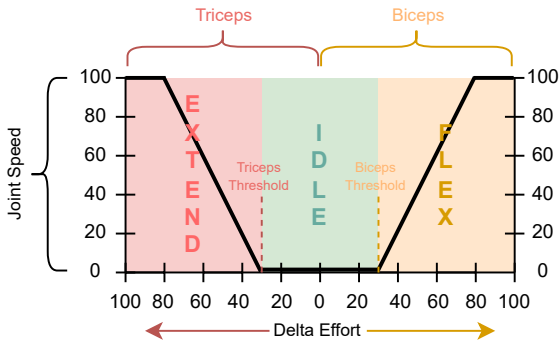


Fig. 2: Illustration of joint speed as a function of delta effort, highlighting the relationship between biceps and triceps activation. The graph is divided into three regions: *flex*, where triceps activation exceeds the threshold, *extend*, where biceps activation surpasses the threshold, and *idle*, where neither muscle reaches activation. The placement of triceps and biceps thresholds is marked, demonstrating their influence on movement transitions.

elbow joint, responsible for flexion and extension, and the wrist joint, enabling hand opening and closing. The device translates muscle activation patterns into motor commands via two surface electromyography (sEMG) sensors. It is important to emphasize that this study focuses exclusively on continuous arm movements controlled by the elbow joint.

### B. Control Modes

The MyoPro 2 operates in four distinct control modes:

- *Standby Mode*: Both motors remain inactive, ignoring sEMG sensors inputs.
- *Biceps Mode*: The device responds solely to the biceps sEMG sensor, allowing only flexion movement.
- *Triceps Mode*: The device responds only to the triceps sEMG sensor, allowing only extension movement.
- *Dual Mode*: The device receives input from both sensors, enabling both flexion and extension movements.

This study focuses on the *Dual Mode*, as it aligns with the natural bidirectional control required for most daily tasks. Within this mode, three movement types can be selected: constant, proportional, and ramp (i.e., exponential). We specifically examine proportional mode, where movement speed scales proportionally to the user’s muscle effort.

In proportional mode, the joint speed,  $S_{\text{joint}}$ , is determined by the difference in efforts between the dominant and opposing muscles:

$$S_{\text{joint}} = k_p \Delta E,$$

where  $k_p$  is proportional gain value specified by the manufacturer, and  $\Delta E$  is the difference between the dominant and opposing muscle efforts,  $E_d$  and  $E_o$ , respectively. The muscle exhibiting the higher activation level is considered dominant. Once the delta effort,  $\Delta E = E_d - E_o$ , exceeds the predefined threshold for the dominant muscle,  $S_{\text{joint}}$  increases proportionally to  $\Delta E$  in the corresponding direction, up to a maximum speed of 100 (dimensionless), as illustrated in Fig 2. For example, if a user intends to flex their arm, the

biceps effort ( $E_d$ ) exceeds the triceps effort ( $E_o$ ), generating a positive  $\Delta E$ . If this difference surpasses the threshold, the exoskeleton initiates flexion at a speed proportional to  $\Delta E$ . Conversely, if the triceps effort becomes dominant, the device facilitates extension. The possible muscle activations and their corresponding actions are illustrated in Fig. 3.

### C. Effort Threshold Optimization

To summarize, beyond selecting a movement type, users can adjust two key parameters: (a) the effort thresholds, which define the biceps/triceps activation levels required to initiate movement, and (b) the gain values, which amplify the sEMG signal. In this study, we focus on optimizing effort thresholds rather than gain values. While gain values affect sensor sensitivity, effort thresholds play a more direct role in determining when and how the device responds to muscle activation. Properly tuning these thresholds enhances control precision, ensuring smoother and more intuitive user-device interaction.

## III. PROPOSED METHOD

Offline Reinforcement Learning (RL) [14] is a powerful data-driven approach for optimizing sequential decision-making. Unlike traditional Deep Reinforcement Learning (DRL), where an agent interacts with the environment to collect experience, offline RL relies solely on a fixed dataset collected through an arbitrary process, eliminating the need for online exploration [15]. This characteristic makes offline RL particularly well-suited for scenarios where real-time interaction is impractical, costly, or unsafe. In the context of exoskeleton control, where continuous online experimentation could lead to user discomfort or even injury, offline RL provides a viable alternative by leveraging pre-collected data to optimize control policies. This enables adaptive and personalized assistance while minimizing risks associated with direct user interaction during training.

Muscle Activation	Effort Bars	Resulting Action
Bicep Contraction		Pure Flexion
Tricep Contraction		Pure Extension
Co-Contraction		No Movement
Imbalanced Muscle		Extension
Imbalanced Muscle		Flexion

Fig. 3: Visualization of muscle activation patterns, corresponding effort levels, and resulting actions. The left column categorizes different activation scenarios, including biceps contraction, triceps contraction, co-contraction, and imbalanced muscle activation. The middle column represents effort levels using effort bars, where biceps and triceps contributions are indicated. The right column shows the resulting action, such as pure flexion, pure extension, no movement, or imbalanced extensions and flexions, based on the activation pattern.

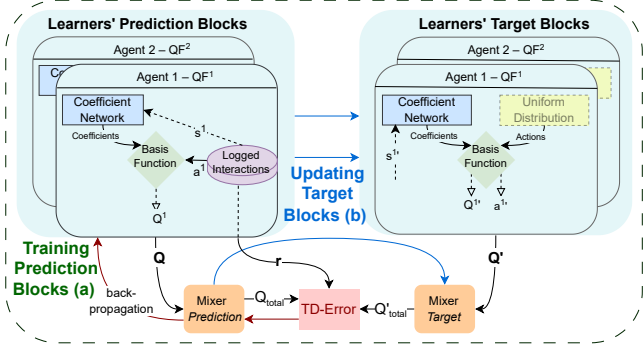


Fig. 4: Overview of the proposed offline MARL architecture with MQF: (a) training prediction blocks, (b) updating target blocks. The red arrows indicate the direction of the backpropagation, while the blue arrows depict the target network updates for mixer and learners' prediction blocks.

The applications of exoskeletons vary depending on the user's condition, often necessitating complex control strategies that could benefit from multi-agent systems. Although our study focuses on a single upper-limb exoskeleton, we frame the problem as a multi-agent system by assigning one agent to optimize the biceps effort threshold and another to optimize the triceps effort threshold. This formulation not only allows for more flexible and adaptive control but also provides a scalable framework that can be extended to scenarios where multiple exoskeletons or assistive devices operate collaboratively to enhance user mobility.

Since the threshold values serve as the action space for the agents and take continuous values, specifically ranging between  $[20, 50]$  for the MyoPro device, we employ Mixed Q-Functionals (MQF) [13] as the learning algorithm for our multi-agent system. MQF has demonstrated superior performance compared to policy-based algorithms, particularly in continuous settings, due to its increased sample efficiency, an essential factor in offline learning where data collection is limited and costly. By leveraging MQF, our approach ensures more effective learning from pre-collected data, leading to improved threshold optimization for adaptive exoskeleton control.

MQF transforms each state into a set of parameters that define a function over the action space. This approach represents states through learned coefficients of basis functions, enabling efficient evaluation of multiple actions via matrix operations. A key advantage of MQF over other value-based multi-agent reinforcement learning methods is its ability to handle continuous action spaces while maintaining the sample efficiency inherent in value-based approaches, resulting in improved agent performance compared to policy-based methods. In this work, we adapt MQF for offline settings by modifying its structure. As illustrated in Fig. 4, our proposed framework consists of two main components: (i) training learners' prediction blocks, and (ii) updating learners' target blocks.

**(i) Training Learners' Prediction Blocks:** The training process for the MQF begins with the selection of a batch

of transition tuples  $\langle s^i, a^i, r^i, s'^i \rangle$ , each of size  $b$ , from the logged interactions  $L^i$  of each agent. These tuples are utilized to calculate the agents' individual  $Q$  and  $Q'$  values. Concisely, the process starts with the agent's coefficient neural network predicting the basis function coefficients for the current state. In simple terms, each state is mapped to a function within the action domain, with the function's coefficients predicted by the network,  $C(s)$ . These coefficients are then used to calculate action values by multiplying them with the corresponding action representations. The total number of coefficients,  $k$ , is determined using the combinatorial formula  $\binom{\mathcal{O}+D}{D}$ , where  $D$  represents the action dimension and  $\mathcal{O}$  the order of the state function. Thus, the  $Q$  function can be expressed as follows:

$$Q^F(s, a) = C(s)^\top \Phi(a),$$

where  $C(s)$  returns a vector of coefficients, calculated by the neural network and  $\Phi(a)$  returns the action representations as a vector, computed according to the predefined function type.

For simplicity, assume the basis function is a *polynomial* of order 2, and the action space is 2-dimensional  $a = [a_1, a_2]$ . Under these conditions, the returned coefficient vector can be represented as:

$$C(s) = [c_0 \ c_1 \ c_2 \ c_3 \ c_4 \ c_5]^\top.$$

The second-order *polynomial* basis consists of all polynomials of the form  $a_1^i a_2^j$  where  $i + j \leq 2$ . Therefore, the action representation is expressed as follows:

$$\Phi(a) = [1 \ a_1 \ a_1^2 \ a_2 \ a_2^2 \ a_1 a_2]^\top.$$

And,  $Q$  value for given state-action pair becomes as follows:

$$Q^F(s, a) = c_0 + c_1 a_1 + c_2 a_1^2 + c_3 a_2 + c_4 a_2^2 + c_5 a_1 a_2.$$

By leveraging matrix multiplication, the evaluation of uniformly sampled actions is significantly accelerated, allowing the action with the highest  $Q$ -value, in (1), to be selected more efficiently.

We employ another two networks: one for mixing the agents' current  $Q$  values to derive  $Q_{\text{total}}$ , and another for mixing the next-state  $Q'$  values to obtain  $Q'_{\text{total}}$ . Fig. 4(a) illustrates these networks as the mixer prediction and the mixer target, respectively. This procedure is followed by the calculation of the temporal difference error between the prediction and target values, which can be encapsulated by the following formulation:

$$e_{\text{TD}} = \sum_{s, a, r, s'}^{b \sim B} [(Q_{\text{total}}(s, a; \mathbf{w}_{\text{PN}}) - y(r, s'))^2], \quad (1)$$

where  $y(r, s')$  is defined as:

$$y(r, s') = \sum_{i=1}^N r^i + \gamma \max_{a'} (Q_{\text{total}}(s', a'; \mathbf{w}_{\text{TN}}) |_{a' \sim \mathcal{U}}). \quad (2)$$

The bold symbols, in (1) and (2), are vectors denoting the set of corresponding values for all agents and  $\mathcal{U}$  signifies a

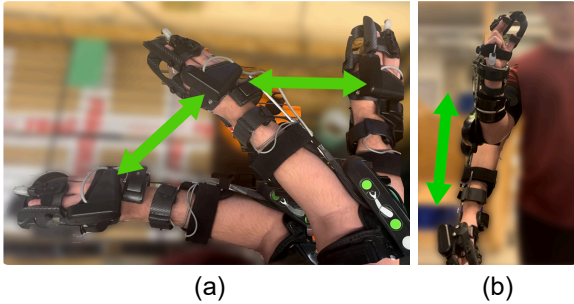


Fig. 5: Demonstration of task movements for data collection using the MyoPro exoskeleton. (a) Depicts the *horizontal* task, where the user moves their arm laterally. (b) Shows the *vertical* task, involving an upward and downward arm motion.

uniform distribution. The function  $Q_{\text{total}}$  serves as a mixing mechanism for the agents’  $Q$ -values. It is also important to note that we view the function for mixing agents’ action values as a meta-function that can vary from additive forms, as in Value Decomposition Networks (VDN) [16], to monotonic functions, as in QMIX [17], or more complex methods, similar to those described in Qtran [18] and weighted QMIX [19]. A more complex mixer model requires increased training effort to handle task complexities. However, for the tasks presented in this study, we observed that simpler mixer functions, as in VDN, work well. The optimization of the agents’  $Q$ -functionals aims to minimize the temporal difference error,  $e_{\text{TD}}$  in (1), thus facilitating the coordination of agent actions to maximize collective rewards. This training is conducted centrally using  $Q_{\text{total}}$ , while the determination of individual agent actions is guided by their  $Q_{\text{F}}^i$ .

**(ii) Updating Learners’ Target Blocks:** Our framework, designed for multi-agent environments with continuous action spaces, diverges from standard value-based methods in updating the target networks. It employs a *soft* update mechanism at each time-step, proven more effective in continuous action domains instead of *periodic* updates [20]. This method

is characterized by incremental adjustments, formalized as:

$$w_{\text{TN}} = \tau w_{\text{PN}} + (1 - \tau)w_{\text{TN}},$$

where  $\tau$  is a small factor ( $\tau \ll 1$ ), ensuring improved stability and effectiveness in such environments.

#### IV. EXPERIMENTS

For our experiments, we utilize the MyoPro [12], a lightweight upper-limb exoskeleton with two degrees of freedom. This wearable device is equipped with four surface electromyography (sEMG) sensors, two placed on the upper arm and two on the forearm, which detect muscle activity associated with biceps and triceps engagement. The exoskeleton’s motor activates when predefined thresholds are exceeded, providing assistance accordingly. Our approach focuses on optimizing these threshold values to enhance ease of use and improve user experience. To evaluate its effectiveness, we train three models: one for each task individually and a third designed to generalize across both tasks.

##### A. Task Design & Data Collection

We designed two tasks, vertical and horizontal, to capture two ubiquitous and essential primitive movement patterns. In the horizontal task, a participant transported an empty can between two fixed locations on a table, emphasizing a controlled lateral arm movement. The vertical task required rotating the arm around a shoulder-originating axis, mimicking a curling motion. These tasks were selected for their distinct characteristics: the horizontal task consists of movements along the horizontal plane, whereas the vertical task requires movements along the vertical axis, engaging different muscle groups and movement dynamics. Demonstration of both tasks are shown in Fig. 5.

##### B. Data Collection

To collect data, each task was repeated multiple times with systematically varying the biceps and triceps effort thresh-

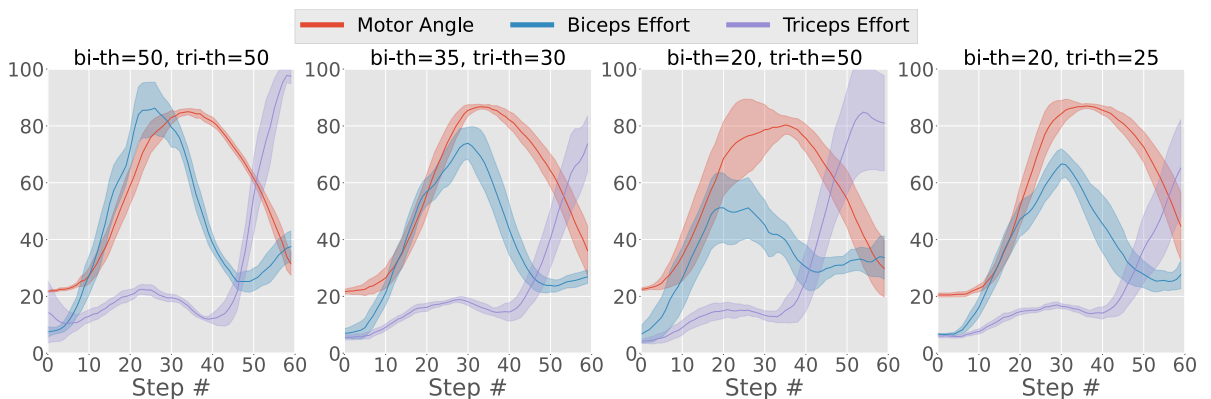


Fig. 6: Visualization of data collected from the vertical task (Fig. 5(b)), showing the relationship between *motor angle*, *biceps* effort, and *triceps* effort across different step numbers. Each subplot represents a different combination of biceps (bi-th) and triceps (tri-th) effort thresholds, as indicated in the titles. The red, blue and purple lines represents motor angle, biceps effort, triceps effort, respectively, with shaded regions indicating the 95% confidence interval. As this task primarily relies on biceps activation, lower biceps threshold values (e.g., 20) appear to facilitate more efficient movement by reducing the required effort while preserving stability and control.

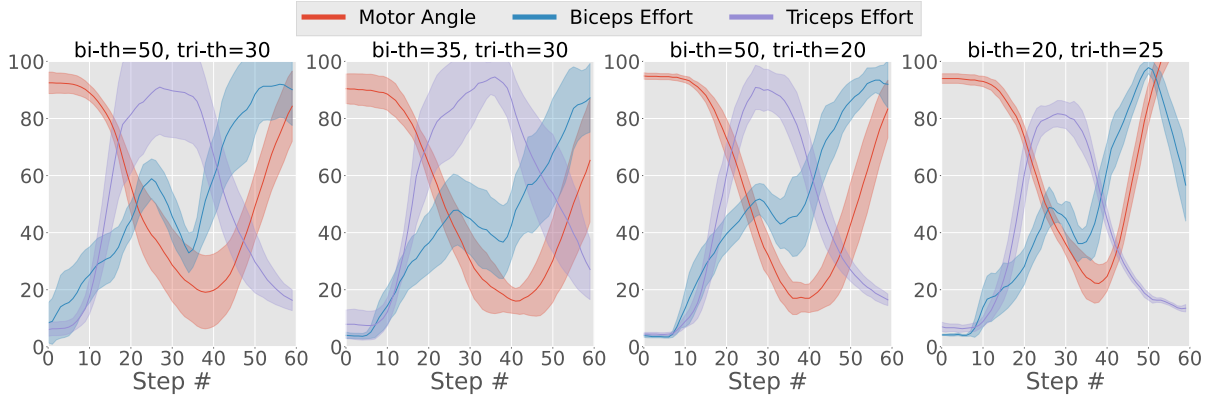


Fig. 7: Analysis of data collected from the horizontal task (Fig. 5(a)), illustrating *motor angle*, *biceps* and *triceps* efforts over time for different biceps (bi-th) and triceps (tri-th) effort thresholds (indicated in the titles). The red line depicts motor angle, while blue and purple lines correspond to biceps and triceps effort, respectively. Shaded regions indicate the 95% confidence interval, highlighting variability in the recorded data. Since this task primarily relies on triceps activation, biceps and triceps threshold values (20, 25), respectively, appear to be the most effective, as they minimize user triceps effort while maintaining smooth movement.

olds. To limit the number of demonstrations and avoid fatigue, these thresholds were manually adjusted in increments of 5, ranging from 20 to 50. For each threshold combination, data was collected over 10 episodes, with each episode lasting approximately 40 seconds. To minimize the effects of fatigue, the user also took brief rest periods between episodes. Fig. 6 and Fig. 7 illustrate the data distribution for selected thresholds in the vertical and horizontal tasks, respectively, where solid lines represent the average values, and the shaded area indicates the 95% confidence interval.

### C. Results, Insights & Challenges

**Preprocessing** - We begin by preprocessing the collected data to ensure it is suitable for training a reinforcement learning model. The state representation consists of three variables:  $\langle p, E_{\text{biceps}}, E_{\text{triceps}} \rangle$ , where  $p$  represents the motor angle, and  $E_{\text{biceps}}$  and  $E_{\text{triceps}}$  correspond to the effort levels for the biceps and triceps, respectively. The action space consists of adjusting either the biceps threshold,  $th_{\text{biceps}}$ , or the triceps threshold,  $th_{\text{triceps}}$ , depending on the assigned agent. Specifically, the agent responsible for biceps muscle adjusts  $th_{\text{biceps}}$ , and the triceps agent modifies  $th_{\text{biceps}}$ .

**Reward function design** - To guide learning, we define a reward function for each state-action pair, formulated as:

$$r = e^{-d/c},$$

where  $d$  represents the difference between  $\Delta E$  and the threshold value of the dominant muscle, and  $c$  is a constant. The objective is to minimize this difference, ensuring that the exoskeleton provides continuous and adaptive assistance throughout task execution. By incorporating the exponential function, we constrain the reward values within the range  $[0, 1]$ , maintaining numerical stability and smooth optimization during training.

**Training** - We have trained three models: one for the horizontal task, one for the vertical task, and one for both tasks combined. During training, batches were drawn from

the preprocessed dataset, which captures how actions (i.e., threshold values) influence new states, such as pose (i.e., motor angle) and effort values. Because the dataset already contains the outcomes of specific state-action pairs, reward calculation during training requires no additional estimation. The models are trained to maximize reward values that can be directly computed using the described reward function, as all relevant parameters are known, ensuring efficient learning within the dataset’s constraints.

**Findings** - Our results highlight the task-dependent nature of threshold optimization in assistive exoskeletons. Fig. 6 illustrates the findings from the vertical task (Fig. 5(b)), where movement primarily relies on biceps activation. The data suggests that lower biceps threshold values (e.g., 20) facilitate smoother movement by reducing the required effort while maintaining control. Conversely, Fig. 7 presents results from the horizontal task (Fig. 5(b)), which depends more on triceps activation. Here, biceps and triceps thresholds of (20, 25), respectively, appear to be the most effective, as they minimize user effort while ensuring smooth and stable motion. These insights emphasize the importance of adaptive tuning to accommodate varying task demands. Even though the models can dynamically set thresholds for the muscles, evaluating their performance and comparing them to static threshold settings presents a significant challenge. When the models generate new actions (i.e., threshold values), they may lead to states that are not present in the dataset, making their outcomes unknown. This issue is further compounded by the dataset’s limitations—it was collected from a single participant with threshold increments of only 5, restricting the ability to assess overall performance comprehensively. One approach to mitigate this limitation is expanding the dataset to ensure that generated actions during testing lead to states already represented in the data. Alternatively, a transition model could be trained to predict state changes resulting from new threshold values, as suggested in [21]. Ultimately, the most reliable way to assess the impact of

dynamic threshold adjustments on user performance in daily tasks is through real-world testing. Future work should involve multiple participants and iterative training cycles incorporating human feedback to refine the models based on real-world interactions.

## V. CONCLUSIONS & FUTURE WORK

This study explored the application of offline reinforcement learning for optimizing effort threshold parameters in a upper-limb assistive exoskeleton. By leveraging a multi-agent learning framework with Mixed Q-Functionals (MQF), we aimed to enable adaptive and data-driven parameter tuning, reducing the need for manual calibration. Our experimental results indicate that the proposed approach can learn from pre-collected data, enabling dynamic adjustments to effort thresholds that may improve user interaction and control. However, challenges remain in evaluating model performance due to the limitations of offline datasets and the absence of real-time interaction. Future work will focus on expanding the dataset with a more diverse participant pool, integrating transition models to better estimate unseen states, and conducting real-world user studies to assess the practical impact of dynamic threshold adjustments. Additionally, further investigations into task complexity and alternative adaptive learning techniques will be essential to improving assistive exoskeleton control in broader applications.

## ACKNOWLEDGEMENT

This work was supported by the National Science Foundation (CMMI-2110214).

## REFERENCES

- [1] A. Esquenazi, M. Talaty, and A. Jayaraman, "Powered exoskeletons for walking assistance in persons with central nervous system injuries: a narrative review," *PM&R*, vol. 9, no. 1, pp. 46–62, 2017.
- [2] R. Baud, A. R. Manzoori, A. Ijspeert, and M. Bouri, "Review of control strategies for lower-limb exoskeletons to assist gait," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, pp. 1–34, 2021.
- [3] U. Martinez-Hernandez, B. Metcalfe, T. Assaf, L. Jabban, J. Male, and D. Zhang, "Wearable assistive robotics: A perspective on current challenges and future trends," *Sensors*, vol. 21, no. 20, p. 6751, 2021.
- [4] K. L. Poggensee and S. H. Collins, "How adaptation, training, and customization contribute to benefits from exoskeleton assistance," *Science Robotics*, vol. 6, no. 58, p. eabf1078, 2021.
- [5] R. Gopura, D. Bandara, K. Kiguchi, and G. K. Mann, "Developments in hardware systems of active upper-limb exoskeleton robots: A review," *Robotics and Autonomous Systems*, vol. 75, pp. 203–220, 2016.
- [6] J. de Miguel-Fernández, J. Lobo-Prat, E. Prinsen, J. M. Font-Llagunes, and L. Marchal-Crespo, "Control strategies used in lower limb exoskeletons for gait rehabilitation after brain injury: a systematic review and analysis of clinical effectiveness," *Journal of neuroengineering and rehabilitation*, vol. 20, no. 1, p. 23, 2023.
- [7] R. Nasiri, M. Shushtari, and A. Arami, "An adaptive assistance controller to optimize the exoskeleton contribution in rehabilitation," *Robotics*, vol. 10, no. 3, p. 95, 2021.
- [8] P. M. Pilarski, M. R. Dawson, T. Degris, J. P. Carey, and R. S. Sutton, "Dynamic switching and real-time machine learning for improved human control of assistive biomedical robots," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechanics (BioRob)*. IEEE, 2012, pp. 296–302.
- [9] P. Slade, M. J. Kochenderfer, S. L. Delp, and S. H. Collins, "Personalizing exoskeleton assistance while walking in the real world," *Nature*, vol. 610, no. 7931, pp. 277–282, 2022.
- [10] D. Gopinath, S. Jain, and B. D. Argall, "Human-in-the-loop optimization of shared autonomy in assistive robotics," *IEEE robotics and automation letters*, vol. 2, no. 1, pp. 247–254, 2016.
- [11] C. Coco Jr, J. Spanos, H. Osooli, and R. Azadeh, "Design of fuzzy logic parameter tuners for upper-limb assistive robots," *WIP paper at 21st International Conference on Ubiquitous Robots (UR), New York, USA, pp. 386–389, June 24–27, 2024*.
- [12] M. Myomo Inc., Cambridge, "Myomo – The MyoPro Brace for Stroke Survivor's Paralyzed Arm." [Online]. Available: <https://myomo.com>
- [13] Y. Findik and S. R. Ahmadzadeh, "Mixed q-functionals: Advancing value-based methods in cooperative marl with continuous action domains," *arXiv preprint arXiv:2402.07752*, 2024.
- [14] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.
- [15] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning: State-of-the-art*. Springer, 2012, pp. 45–73.
- [16] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.
- [17] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.
- [18] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 5887–5896.
- [19] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 10 199–10 210, 2020.
- [20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [21] H. Osooli, C. Coco, J. Spanos, A. Majdi, and R. Azadeh, "Investigating the generalizability of assistive robots models over various tasks," in *2024 21st International Conference on Ubiquitous Robots (UR)*. IEEE, 2024, pp. 227–232.