2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Controlling FES of arm movements using physics-informed reinforcement learning via co-kriging adjustment

Nat Wannawas[1], Clara Diaz-Pintado[1], Jyotindra Narayan[1,2], A. Aldo Faisal[1,2,3]

Abstract— **Upper limb paralysis affects the quality of life. Functional Electrical Stimulation (FES) offers a solution to restore lost motor functions. Yet, there remain challenges in controlling FES to induce arbitrary arm movements. Reinforcement learning (RL) emerges as a promising method for controlling arm movement with success in simulation. However, challenges remain in translating the successes into real-world settings. One dominant challenge is the sample efficiency of RL. This study presents a practical RL setup to control FES for arm movements. We also present a flexible method, called co-kriging adjustment (CKA), which combines a biomechanical simulator and real data to build an accurate model of the real system. We demonstrate our RL-based control on a 2-DoF planar setting where the subject's arm, placed on a frictionless supporter, is stimulated to perform point-to-point reaching. By using 90 seconds of real interaction data, our RL-based control can perform the reaching with the average error over the workspace of 5.5 cm. Beyond the application of FES, our method can be extended to other control systems, propelling RL towards general uses in the real world.**

I. INTRODUCTION

Every year, numerous individuals across the globe suffer from spinal cord injuries and strokes, losing their motor functions [1]. One commonly lost motor skill is arm movement, which severely restricts the ability of daily activities. Functional Electrical Stimulation (FES) can facilitate the restoration of the lost motor functions [2]. Advances have been made in lower-limb cycling where a certain pattern of FES is applied periodically to induce cycling motions [3]–[5]. However, controlling FES for arbitrary arm movements is relatively difficult as reaching towards an end-point requires non-linear muscle coordination and complex control schemes. Additionally, the fact that different individuals' muscles respond differently to FES complicates the controller design and parameter tuning. The advancement in this area is therefore relatively limited.

Multiple control approaches have been explored. The approaches that have been investigated in simulation studies include feedback-feedforward controls [6], [7], open-loop control [8], adaptive control [9], and trajectory planning [10], and reinforcement learning [11]–[15]. Several approaches have successes in real-world settings; some of them require hybrid setups where FES is used together with assistive devices such as exoskeletons. Those include iterative learning control [16], [17], feedback control [18], invert dynamics [19], [20], reinforcement learning [21], and model predictive control [22]. One drawback of hybrid systems is that they pose challenges for practical integration into daily tasks [23]. In addition, rigorous calibration and adjustment are necessary to guarantee optimum performance. The heuristic tuning to find the appropriate control parameters and thresholds can be computationally expensive, and they might need to be changed periodically between sessions [24].

The approach that has been rigorously investigated in simulation recently but has limited studies in real-world settings is Reinforcement Learning (RL). This is partly due to the data efficiency issue of RL, where the required amount of interactions is prohibitive in human-in-the-loop systems. One dominant solution is to use a simulated environment, which is generally referred to as model-based RL.

Simulated environments are often created by fitting function approximators to real transition data. Recent trends involve merging RL with various function approximators, such as neural networks. For example, Deisenroth et al. [25] and Andersson et al. [26] use the Gaussian Process as a sample-efficient and uncertainty-aware function approximator to model cart-pole, cart-double pendulum, and extended cart-pole systems. Some machine learning approaches integrate data and physical dynamics, like physics-informed neural networks (PINNs) [27]. PINNs, recently applied in modelling human musculoskeletal systems and artificial muscle actuators [28], use physics knowledge for derivative loss in parameter updates, requiring grey box modelling [29]. However, the effectiveness hinges on the physics model's complexity—too intricate is hard to optimize, and too simplified may not capture system dynamics accurately.

In real-world problems, we may have simulators or intuitive ideas about their behaviours. In a robot arm manipulation task, for example, we know that if we apply positive torque to a joint, the joint angle should change in a positive direction even though we may be unable to predict the exact amount of change. Leveraging such basic knowledge in modelling could improve the sample efficiency of the model, give sensible predictions when the data is scarce, and provide flexibility in using the inaccurate model. This leads us to multi-fidelity modelling, which leverages low-fidelity data (computationally cheap but inaccurate) and high-fidelity data (computationally expensive but accurate) to maximise the accuracy of model estimates. Specifically, we are interested in the co-kriging technique [30], [31], also known as the multi-fidelity Gaussian Process ($\mathcal{GP}$) [32]. This technique uses two $\mathcal{GP}$s to fuse the data from two fidelity levels to deliver high-fidelity predictions.

[1]Brain & Behaviour Lab, Imperial College London, London SW7 2AZ, United Kingdom. (email:nat.wannawas18@imperial.ac.uk). [2]Brain & Behaviour Lab, Institute for Artificial and Human Intelligence, Universität Bayreuth, Bayreuth, Germany 95445. [3]Chair of Digital Health, Universität Bayreuth, Germany 95445. (aldo.faisal@imperial.ac.uk).

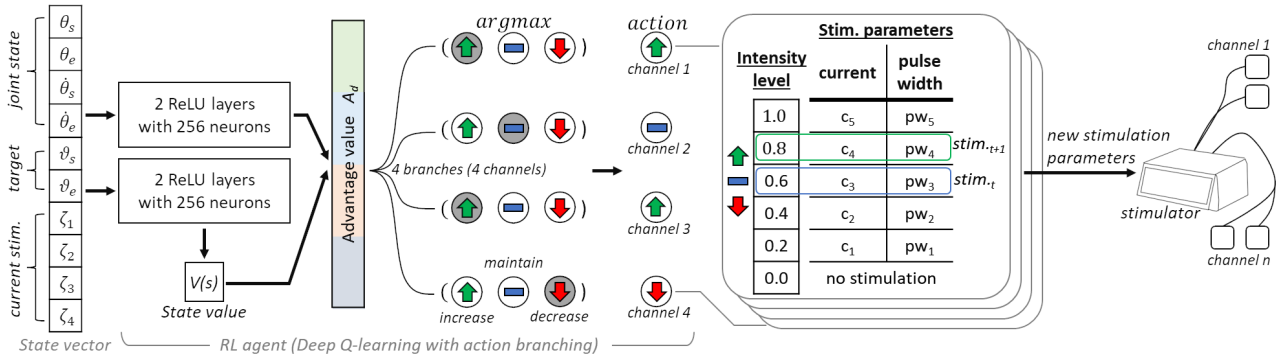

Fig. 1. RL-based Control diagram. The RL agent has a duelling network architecture where both state and advantage value streams are parameterised by separated neural networks. The agent receives a 10-element state vector and outputs Q values which are split into 4 branches; each branch has 3 sub-actions.

Our contributions here are as follows. Firstly, we present an RL setup to learn FES controls for arbitrary arm-reaching movements that can be applied to real-world settings. Secondly, we present a sample-efficient method that combines a physics-derived model and real-world data to form a probabilistic model for model-based RL. This method enables the sim-to-real transfer of RL with a handful amount of data. Thirdly, we present a real-world experimental setup for performing FES-induced planar reaching movements. We demonstrate our RL-based FES control framework on an able-bodied subject performing the reaching tasks.

II. Method

A. Control problem statement

The overview of our control task is to move the arm from an initial position to a target position by using FES. Here, we focus on a planar (horizontal) setting with two degrees of freedom: shoulder and elbow joints. The position is defined in terms of shoulder and elbow joint angles, θ_s and θ_e. The stimulation is applied via four channels of surface electrodes targeting *Pectoralis*, *Deltoid Posterior*, *Biceps*, and *Triceps*. The stimulation of each channel is defined in terms of stimulation intensity levels $\zeta \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, with 0.0 and 1.0 representing no stimulation and maximum intensity, respectively. The stimulation level can only change to an adjacent level to prevent dangerous abrupt movements.

B. Reinforcement Learning (RL)

We use RL to learn stimulation strategies for moving the arm to an assigned target. The overview of RL algorithms is briefly described as follows. RL learns a task through reward signals collected from the interaction with an environment. The interactions occur in a discrete-time fashion, starting with the agent observing the environment's state $\mathbf{s}_t$ and selecting an action $\mathbf{a}_t$ based on its policy π. The action causes the environment to be in a new state $\mathbf{s}_{t+1}$. The agent then receives an immediate reward r_t and observes the new state. This interaction experience is collected as a tuple $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ which is stored in a replay buffer $\mathcal{D}$. This tuple is used to learn an optimal policy π^* that maximises a return R–the sum of discounted immediate rewards.

In this task, the state vector comprises angles and angular velocities of shoulder and elbow, target shoulder and elbow angles (ϑ_s and ϑ_e), and the current stimulation intensities of 4 channels (ζ_1, ζ_2, ζ_3, ζ_4). This is formally expressed as $\mathbf{s} = [\theta_s, \theta_e, \dot{\theta}_s, \dot{\theta}_e, \vartheta_s, \vartheta_e, \zeta_1, \zeta_2, \zeta_3, \zeta_4]$, where $\dot{\theta}_s$ and $\dot{\theta}_e$ are the angular velocities of shoulder and elbow angles.

The action here is to manipulate the stimulation intensity of each individual channel through 3 action options: increase, maintain, and decrease. As the action space is discrete, the RL algorithm of choice centres around a family of Q-learning [33]. However, applying typical Q-learning to this setup results in an action dimension of 81 which requires a large amount of interaction data to learn a good policy. To deal with the high action dimension problem, we adopt a variant architecture of Q-learning called action branching [34], which was originally designed to make Q-learning work in an environment with continuous action space. The action branching architecture is similar to a typical Q-learning architecture except that the action space is divided into N branches. For an action branch $d \in \{1, ..., N\}$ with $|A_d| = n$ discrete sub-actions, the individual branch's Q-value at state $\mathbf{s}$ and sub-action $a_d \in A_d$ is expressed as

$$Q_d(\mathbf{s}, a_d) = V(\mathbf{s}) + (A_d(\mathbf{s}, a_d) - \frac{1}{n} \sum_{a'_d \in A_d} A_d(\mathbf{s}, a'_d)), \quad (1)$$

where $V(\mathbf{s})$ and $A_d(\mathbf{s}, a_d)$ are state and advantage values as in typical Q-learning. The state and advantage value functions are parameterised by fully-connected neural networks comprising two hidden layers with 256 neurons and ReLU activation functions. The networks are optimised in a typical Q-learning fashion, minimising the temporal difference loss:

$$L = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}[\frac{1}{N} \sum_d (y_{\text{TD}} - Q_d(\boldsymbol{s}, a_d))^2], \quad (2)$$

where y_{TD} is a temporal difference target computed as

$$y_{\text{TD}} = r + \gamma \frac{1}{N} \sum_d Q_d(\boldsymbol{s}', \underset{a'_d \in A_d}{\arg\max}\, Q_d(\boldsymbol{s}', a'_d)), \quad (3)$$

where $\gamma \in [0, 1)$ is a discount factor.

In this task, specifically, the action space has 4 branches, one for each channel. Each branch has 3 sub-actions: *increase*, *maintain*, and *decrease* the stimulation intensity level (Fig.1). The output action vector, therefore, has 4 elements that contain selected sub-actions for 4 channels.

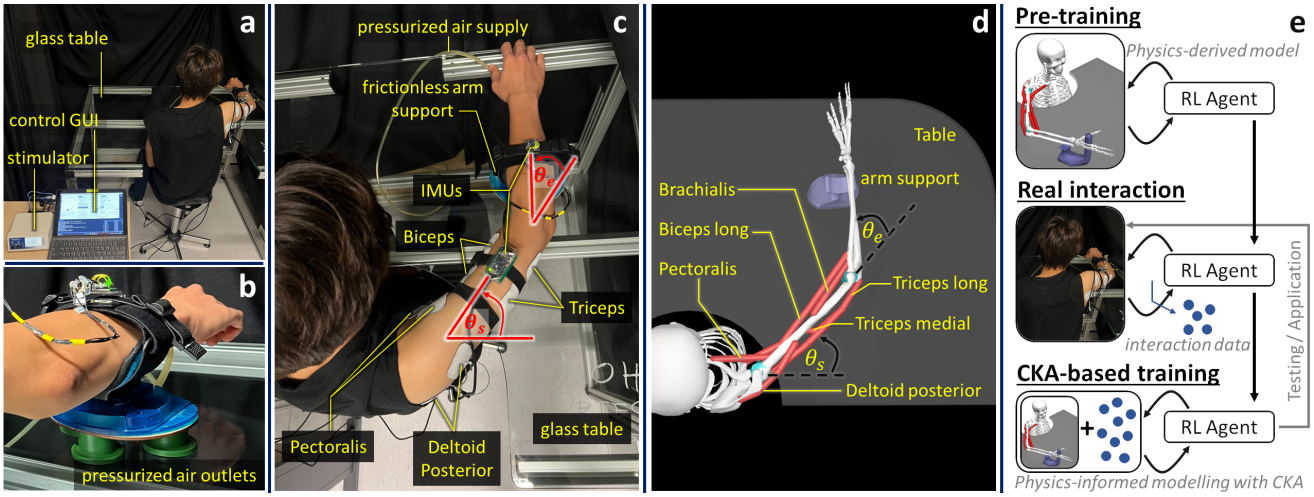

Fig. 2. (a) The overview of the experimental setup. (b) Frictionless arm support. (c) Detail setup at the subject's arm placed on the arm support that moves on a glass table. (d) Human arm model built in OpenSim. (d) Process diagram of our framework comprising two training phases.

The RL policy here is ϵ-greedy where, for each branch, the probability of selecting the sub-action with the highest Q value is $\epsilon \in [0, 1]$. ϵ is set to 0.25 and 0 during the training and performance evaluations, respectively. The immediate reward r_t is simply computed using the square error and action penalty as

$$r_t = -(\| [\theta_s, \theta_e]_{t+1}, [\vartheta_s, \vartheta_e]_t \|) - \frac{1}{4}\Sigma_{i=1}^{m}\zeta_{i,t}, \quad (4)$$

where m is the number of stimulated muscles ($m = 4$).

C. RL pre-training

We pre-trained an RL agent before transferring it to the real-world setup. In the pre-training, an RL agent interacts with a biomechanical model of a human arm (Fig.2d) which was introduced in our previous work [12]. The model, built using a biomechanics simulator called OpenSim, has the right arm connected to the upper body. The elbow and shoulder joints are modelled as pin joints rotating about vertical axes. The model has 4 Hill-type muscles crossing elbows: *Triceps medial*, *Triceps long*, *Brachialis*, and *Biceps short*. The other muscles crossing the shoulder are *Pectoralis* and *Deltoid posterior*. The arm is placed on an arm support that moves on a table with low friction.

RL agent interacts with the arm model in an episodic fashion. Each episode comprises 18 time steps with a size of 200 ms and has random initial and target arm poses. Along the training, the agent's performance is evaluated on a 100-target reaching task where the targets are the Cartesian product of ϑ_s and ϑ_e discretized values. The performance is measured in terms of the hand location error in the Cartesian coordinates computed from the last second of each reaching.

D. Co-kriging adjustment (CKA)

The overview of physics-informed modelling here is described as follows. Suppose we have a physics-derived function or a simulator that predicts the transition of a system, $\mathbf{s}_{t+1} = f_p(\mathbf{x}_t) = f_p(\mathbf{s}_t, \mathbf{c}_t)$, where $\mathbf{s}_t$ and $\mathbf{c}_t$ is the state and control applied to the system, respectively. This function, which may not fully capture the system's dynamics, can be viewed as the crude function of the system's transition. The modelling task is to learn a function $f_a(x)$ that adjusts the outputs of $f_p(x)$ to predict the real system's transitions using the transition data collected from the real system. Formally, this is expressed as $\mathbf{s}_{t+1} = f_a(f_p(\mathbf{x}_t), \mathbf{x}_t)$.

Our base idea is to build a probabilistic model that relies on f_p and is aware of the uncertainty if the data are absent. When the data are present, the predictive means become close to the data, and the predictive variances are small. These behaviours are intuitive and essential for preventing overconfident issues in reinforcement learning.

E. Co-kriging adjustment (CKA): Formulation

The mathematical formulation of the base idea is as follows. We design $f_p(x)$ to be adjusted by a scaling function $\rho(x)$ and a bias term $\delta(x)$, expressed as:

$$f_a(f_p(x), x) = \rho(x) f_p(x) + \delta(x). \quad (5)$$

We make $f_a(x)$ a probabilistic function by putting Gaussian Process priors over $\rho(x)$ and $\delta(x)$. As we want the predictive mean to be close to $f_p(x)$ in the locations where the data are not available, we set the prior means of $\rho(x)$ and $\delta(x)$ to be 1 and 0, respectively, as:

$$\rho(x) \sim \mathcal{GP}(1, k_\rho(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}_\rho)),$$
$$\delta(x) \sim \mathcal{GP}(0, k_\delta(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}_\delta)),$$

where k_ρ and k_δ are covariance functions with $\boldsymbol{\theta}_\rho$ and $\boldsymbol{\theta}_\delta$ parameters. Hence, the prior of f_a becomes

$$f_a \sim \mathcal{GP}(f_p(\boldsymbol{x}), F_p k_\rho(\boldsymbol{x}, \boldsymbol{x'}) F_p^T + k_\delta(\boldsymbol{x}, \boldsymbol{x'})), \quad (6)$$

where F_p is the diagonal matrix of $f_p(x)$.

F. Co-kriging adjustment (CKA): Learning and Prediction

The learning of CKA is the optimisation of the covariance functions' parameters, $\boldsymbol{\theta}_\rho$ and $\boldsymbol{\theta}_\delta$. The optimisation follows a standard procedure for Gaussian Processes. We model the observations $\boldsymbol{y}$ at input locations $\boldsymbol{x}$ as the outputs of the real system with Gaussian noises:

$$\boldsymbol{y} = f_a(\boldsymbol{x}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(0, \sigma^2 \boldsymbol{I}), \quad (7)$$

where σ is a Gaussian noise parameter. The distribution of $\boldsymbol{y}$ can then be expressed as:

$$\boldsymbol{y} \sim N(f_p(\boldsymbol{x}), \underbrace{F_p k_\rho(\boldsymbol{x}, \boldsymbol{x'}) F_p^T + k_\delta(\boldsymbol{x}, \boldsymbol{x'}) + \sigma^2 \boldsymbol{I}}_{\boldsymbol{K}}), \quad (8)$$

The parameters $\boldsymbol{\theta}_\rho$, $\boldsymbol{\theta}_\delta$, σ can be optimised by performing gradient descent to minimise the negative log marginal likelihood (NLML), computed as

$$\begin{aligned} NLML =& \frac{1}{2}(\boldsymbol{y} - f_p(\boldsymbol{x}))^T \boldsymbol{K}^{-1}(\boldsymbol{y} - f_p(\boldsymbol{x})) \\ &+ \frac{1}{2} log|\boldsymbol{K}| + \frac{N}{2} log(2\pi), \end{aligned} \quad (9)$$

where N is the number of training data–the observations collected from the real system.

The prediction of CKA also follows the standard procedure of Gaussian Processes where the predictive means and variances at location $\boldsymbol{x}_*$ can be computed as:

$$\boldsymbol{\mu}_{\boldsymbol{x}_*} = f_p(\boldsymbol{x}_*) + \boldsymbol{q}^T \boldsymbol{K}^{-1}(\boldsymbol{y} - f_p(\boldsymbol{x})), \quad (10)$$

$$\boldsymbol{var}_{\boldsymbol{x}_*} = \boldsymbol{K}_{**} - \boldsymbol{q}^T \boldsymbol{K}^{-1} \boldsymbol{q} + \sigma^2 \boldsymbol{I}, \quad (11)$$

where $\boldsymbol{K}_{**}$ and $\boldsymbol{q}$ are computed as

$$\begin{aligned} \boldsymbol{K}_{**} &= F_p(\boldsymbol{x}_*) k_\rho(\boldsymbol{x}_*, \boldsymbol{x}'_*) F_p(\boldsymbol{x}_*)^T + k_\delta(\boldsymbol{x}_*, \boldsymbol{x}'_*), \\ \boldsymbol{q} &= F_p(\boldsymbol{x}) k_\rho(\boldsymbol{x}, \boldsymbol{x}_*) F_p(\boldsymbol{x}_*)^T + k_\delta(\boldsymbol{x}, \boldsymbol{x}_*), \end{aligned}$$

where $F_p(\boldsymbol{x}_*)$ is a diagonal matrix of $f_p(\boldsymbol{x}_*)$.

G. RL training with CKA

Here, the physics-derived function $f_p(x)$ is the OpenSim arm model which takes joint angles, angular velocities, and stimulation intensities as input and predicts joint angles and angular velocities of the next time step. The prediction is then adjusted by $f_a(x)$. The RL training with CKA is similar to the pre-training except that the trajectories are the outputs of f_a rather than those of f_p. In this phase, the agent learns only from CKA-generated trajectories, i.e., the experiences from the pre-training and real-world interactions are not used.

H. Experiment setup

The real-world setup has a human subject sitting next to a glass table (Fig.2a). The right arm of the subject is placed on an arm support that discharges pressurised air on the table to minimise contact friction (Fig.2b). The shoulder and elbow angles are measured through two inertial measurement units (IMUs) placed on the upper and lower arms (Fig.2c). The stimulation is delivered to arm muscles via self-adhesive surface electrodes. There are 4 independent pairs (4 channels) of electrodes targeting *Biceps*, *Triceps*, *Pectoralis*, and *Deltoid posterior* (Fig.2c). The stimulation pulses are generated by a Rehastim 1 stimulator (HASOMED GmbH, Magdeburg, Germany) that receives commands from a computer (Fig.2a). RL training, IMUs data stream, and control decision-making are processed locally on the computer that runs a graphic user interface (GUI) built using Python and Tkinter library.

The stimulation has three parameters: frequency, current, and pulse width. In this setup, the frequency is fixed at 40 Hz. The stimulation intensity is manipulated by varying the current and pulse width. The current and pulse width values representing different intensity levels for each channel are determined empirically. The process starts with fixing the pulse width at 200 μs and finding the minimum current (c_m) that can move the corresponding joint. After that, 5 pairs of current and pulse width (current [mA], pulse width [μs]) representing 5 intensity levels are generated as $(c_m, 200)$, $(c_m + 1, 200)$, $(c_m + 1, 225)$, $(c_m + 1, 250)$, $(c_m + 2, 225)$.

The real-world performance evaluation is similar to that in simulation but with 25 targets and smaller workspace: ϑ_s and $\vartheta_e \in \{20°, 37.5°, 55°, 72.5°, 90°\}$. Lower- ($\vartheta < 20°$) and upper-end ($\vartheta > 90°$) angles are excluded because they require high muscle contraction. Additionally, a shoulder angle above 90° can cause physical contact between electrodes due to skin deformation. Each reaching starts at shoulder and elbow angles between 5° and 10°. Each reaching lasts 3.6 seconds, comprising 18 time steps with the size of 200 ms.

III. Results

A. RL pre-training

We trained two variances of Q-learning algorithms: 1) typical Q-learning which has 81-dimensional action space (Q) and 2) Q-learning with action branching architecture (QBR). The training has 200 episodes of data collection. Fig.3 shows the learning curves in terms of rewards and reaching RMSEs tested on the 100-target task.

Both Q and QBR become more proficient at the task as the training progresses; they collect higher rewards and achieve lower reaching errors. Noticeably, QBR's learning curve is sharper; it reaches the RMSE below 10 cm and the plateau within 25 and 75 episodes, respectively. Q's learning curve reaches similar milestones within 70 and 100 episodes. In the required amount of data perspective (the upper axis of Fig.3), QBR requires roughly 70 seconds of interaction data to achieve the reaching RMSE below 10 cm. Q requires almost 3 times as much. Additionally, QBR has slightly better end performance than Q. These show how the branching architecture is effective in this application.

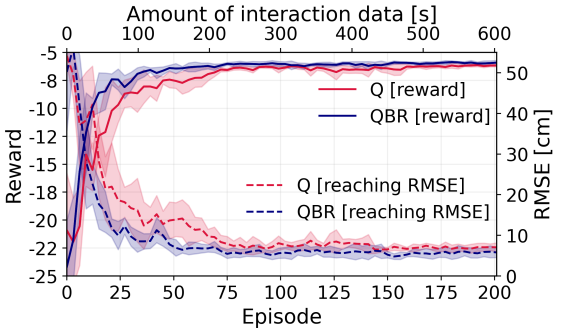

Fig. 3. Learning curves of (red) Q and (blue) QBR in the pre-training. The shade represents the standard deviation of 5 runs.

B. Simulation performance

We include a simple PID-based independent joint control in the performance evaluation. PID controller's gains are optimised by using a genetic algorithm. Fig.4a summarises

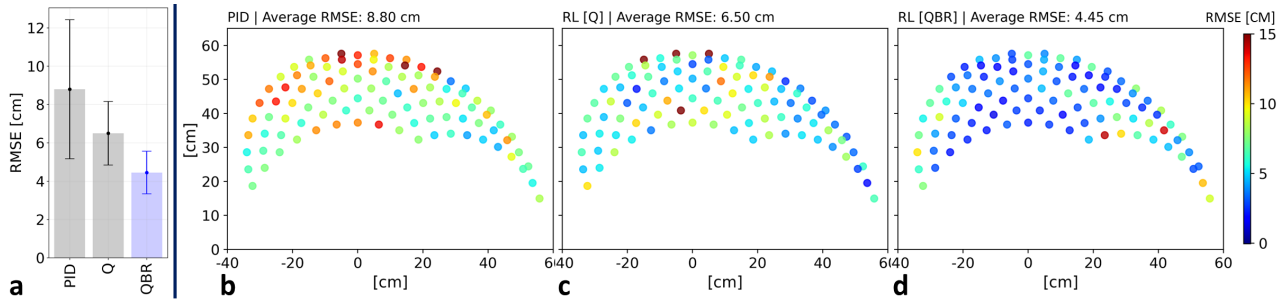

Fig. 4. Performance evaluation on the 100-target task in simulation. (a) Bar chart summarising the reaching RMSEs. 100 colour-coded dots showing RMSEs of all reaching motions controlled by (b) PID, (c) Q, and (d) QBR. The shoulder joint functions as the origin which is located at (0,0).

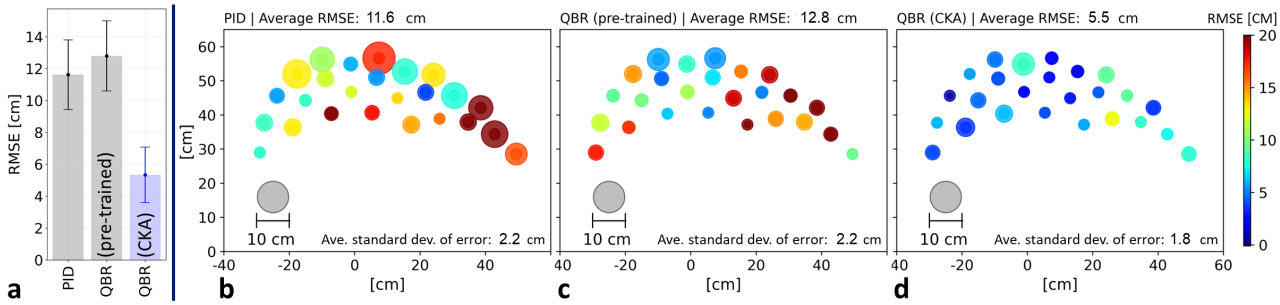

Fig. 5. Real-world performance evaluation. (a) Bar chart summarising the performance. 25 colour-coded dots showing RMSEs of all reaching motions controlled by (b) PID, (c) pre-trained QBR, and (d) CKA-QBR. The dots' sizes represent the standard deviations of RMSE in the last second of the reaching motions.

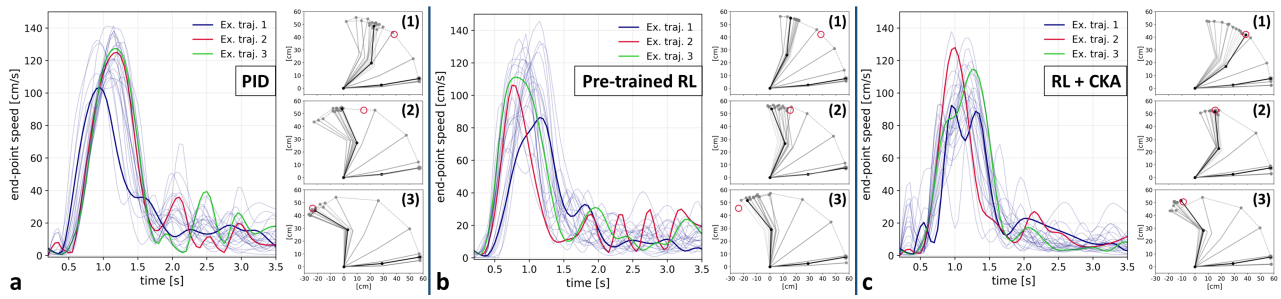

Fig. 6. End-point (hand) velocity profiles of all real-world reaching controlled by (a) PID, (b) pre-trained QBR, and (c) CKA-QBR. Sub-figures (1-3) in each case show example movements to 3 targets whose corresponding profiles are highlighted.

the performances of PID, Q, and QBR on the 100-target task in the RMSE measure. QBR is the best performer with the average RMSE of 4.45 cm, which is roughly 30% less than Q's. PID controller performs reasonably well and is capable of achieving the RMSE below 10 cm. Fig.4b-d show the RMSEs of all targets. PID controller has relatively high RMSEs at outer regions where the elbow has to be fully extended (Fig.4b). Q performs better than PID but has relatively poor performance in the central region (Fig.4c). Noticeably, QBR has better performance than the others.

C. Real-world performance

The real-world evaluation is divided into two phases. The first phase is a direct sim-to-real transfer where the agent trained on the OpenSim model (pre-trained QBR) is tested on the real-world setup. The data collected in this phase is used to train a CKA model with which the pre-trained agent interacts to improve its policy (CKA-QBR). The agent is then tested again on the same setup.

Fig.5b-d show the reaching RMSEs of 25 targets achieved by PID, pre-trained QBR, and CKA-QBR. Both PID (Fig.5b) and pre-trained RL (Fig.5c) perform worse than what they achieve in the simulation, especially in the right-half region of the workspace. QBR's performance improves significantly after further training with the CKA model, with the average RMSE decreasing to 5.5 cm (Fig.5d). The RMSEs in the right-half region are also relatively high. The dot size represents the standard deviation (SD) of error in the last second of the reaching. CKA-QBR has the smallest average SD which implies less movement during the final stage of the reaching.

Fig.6 shows velocity profiles of all reaching motions. Three examples where the targets are in the right, centre, and left regions are shown in detail. In all cases, the velocity profiles have a bell-shaped characteristic as found in natural movements. PID (Fig.6a) and pre-trained QBR (Fig.6b) have wobbling motions after the bell curves, while most of CKA-transferred QBR's profiles gradually decrease to a near-zero region. These behaviours align with the dot size in Fig.5b-d.

IV. Conclusion & Future Work

We present an RL-based framework for controlling arm movement through FES. We address the data-intensive prob-

lem of RL by introducing a sample-efficient modelling method that combines a physics-derived model with sample-efficient adjustment functions. We demonstrate our framework on a 2-DoF planar movement setup and show that, by using 90 seconds of real interaction data, our framework can learn reaching tasks with an average error of 5.5 cm.

One limitation of our framework is that the current and pulse width pairs are determined empirically. We observed that the method is less effective on large muscles such as *Deltoid posterior* which requires high current. A more systematic approach is to perform a parameter sweep, record the force, and select a set of pairs that yields linear force responses. Another limitation is that the current RL setup takes no account of muscular fatigue. Although the fatigue had moderate effects in this study because of the short experiment duration, adding a tool to deal with the fatigue, e.g. [13], can benefit the practical use. Further experiments on more subjects including paralysed individuals are also necessary for evaluating the effectiveness of this framework.

In a broader view, we believe that RL will have an important role in automatic controls. CKA, a flexible method that leverages both physics knowledge and data, is very sample-efficient and can be beneficial to general control applications. Our RL application in human-in-the-loop systems where safety and the limited amount of interaction are major issues can be viewed as a case study illustrating and propelling RL in the realm of automatic controls in the real world.

ACKNOWLEDGMENT

NW acknowledges his support by the Royal Thai Government Scholarship. AAF acknowledges his support by UKRI Turing AI Fellowship (EP/V025449/1).

REFERENCES

[1] M. Mekki *et al.*, "Robotic rehabilitation and spinal cord injury: a narrative review," *Neurotherapeutics*, vol. 15, pp. 604–617, 2018.

[2] N. Donaldson *et al.*, "Fes cycling may promote recovery of leg function after incomplete spinal cord injury," *Spinal Cord*, vol. 38, no. 11, pp. 680–682, 2000.

[3] A. P. Bo *et al.*, "Cycling with spinal cord injury: A novel system for cycling using electrical stimulation for individuals with paraplegia, and preparation for cybathlon 2016," *IEEE Robotics and Automation Magazine*, vol. 24, pp. 58–65, 2017.

[4] N. Wannawas, M. Subramanian, and A. A. Faisal, "Neuromechanics-based deep reinforcement learning of neurostimulation control in fes cycling," in *Intl IEEE/EMBS Conf Neural Eng. (NER)*, 2021.

[5] N. Wannawas and A. A. Faisal, "Towards ai-controlled movement restoration: Learning fes-cycling stimulation with reinforcement learning," in *IEEE Intl Conf Rehabil. Robotics (ICORR)*, 2023.

[6] D. Blana *et al.*, "Combined feedforward and feedback control of a redundant, nonlinear, dynamic musculoskeletal system," *Med Biol Eng Comput*, vol. 47, pp. 533–542, 2009.

[7] R. S. Razavian *et al.*, "Feedback control of functional electrical stimulation for 2-d arm reaching movements," *IEEE Trans Neural Sys. & Rehabil. Eng.*, vol. 26, pp. 2033–2043, 2018.

[8] Y.-W. Liao *et al.*, "Modeling open-loop stability of a human arm driven by a functional electrical stimulation neuroprosthesis," in *35th Annl Intl Conf of IEEE/EMBS*, 2013, pp. 3598–3601.

[9] E. Bardi *et al.*, "Adaptive cooperative control for hybrid fes-robotic upper limb devices: A simulation study," in *Annual Intl. Conf. IEEE/EMBS*, 2021, pp. 6398–6401.

[10] D. N. Wolf *et al.*, "Data-driven dynamic motion planning for practical fes-controlled reaching motions in spinal cord injury," *IEEE Trans. Neural Systems and Rehabil. Eng.*, vol. 31, pp. 2246–2256, 2023.

[11] K. M. Jagodnik *et al.*, "Human-like rewards to train a reinforcement learning controller for planar arm movement," *IEEE Trans Human-Machine Systems*, vol. 46, pp. 723–733, 10 2016.

[12] N. Wannawas and A. A. Faisal, "Towards ai-controlled fes-restoration of arm movements: neuromechanics-based reinforcement learning for 3-d reaching," in *Intl. IEEE/EMBS Conf. Neural Eng. (NER)*, 2023.

[13] ——, "Towards ai-controlled fes-restoration of arm movements: Controlling for progressive muscular fatigue with gaussian state-space models," in *Intl. IEEE/EMBS Conf. Neural Eng. (NER)*, 2023.

[14] J. Abreu *et al.*, "Deep reinforcement learning for control of time-varying musculoskeletal systems with high fatigability: a feasibility study," *IEEE Trans. on Neural Syst. & Rehabil. Eng.*, 2022.

[15] D. C. Crowder *et al.*, "Improving the learning rate, accuracy, and workspace of reinforcement learning controllers for a musculoskeletal model of the human arm," *IEEE Tran. Neural Sys. & Rehabil. Eng.*, vol. 30, pp. 30–39, 2022.

[16] C. T. Freeman, "Iterative learning control of fes applied to the upper extremity for rehabilitation," *Control Engineering Practice*, vol. 17, pp. 368–381, 2009.

[17] ——, "Upper limb electrical stimulation using input-output linearization and iterative learning control," *IEEE Trans Control Systems Technology*, vol. 23, pp. 1546–1554, 2015.

[18] R. S. Razavian *et al.*, "Feedback control of functional electrical stimulation for arbitrary upper extremity movements," in *IEEE Intl. Conf Rehabil. Robotics (ICORR)*, 2017, pp. 1451–1456.

[19] D. N. Wolf and E. M. Schearer, "Evaluating an open-loop functional electrical stimulation controller for holding the shoulder and elbow configuration of a paralyzed arm," in *IEEE Intl. Conf. Rehabil. Robotics*. IEEE Computer Society, 8 2017, pp. 789–794.

[20] D. N. Wolf, Z. A. Hall, and E. M. Schearer, "Model learning for control of a paralyzed human arm with functional electrical stimulation," in *IEEE Intl Conf on Robotics & Automation (ICRA)*, 2020.

[21] N. Wannawas, A. Shafti, and A. A. Faisal, "Neuromuscular reinforcement learning to actuate human limbs through fes," *Artificial Organs*, vol. 46, pp. 315–319, 2022.

[22] E. Trigili *et al.*, "Hybrid fes-exoskeleton control: Using mpc to distribute actuation for elbow and wrist movements," *Frontiers in Neurorobotics*, 2023.

[23] P. Sampson *et al.*, "Using functional electrical stimulation mediated by iterative learning control and robotics to improve arm movement for people with multiple sclerosis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, pp. 235–248, 2 2016.

[24] F. Grimm and A. Gharabaghi, "Closed-loop neuroprosthesis for reach-to-grasp assistance: Combining adaptive multi-channel neuromuscular stimulation with a multi-joint arm exoskeleton," *Frontiers in Neuroscience*, vol. 10, 2016.

[25] M. P. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *28th Intl. Conf. Machine Learning (ICML)*, 2011, pp. 465–472.

[26] O. Andersson *et al.*, "Model-based reinforcement learning in continuous environments using real-time constrained optimization," in *29th AAAI*, 2015.

[27] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Computational Physics*, vol. 378, pp. 686–707, 2019.

[28] K. Taneja *et al.*, "A feature-encoded physics-informed parameter identification neural network for musculoskeletal systems," *J. Biomechanical Engineering*, vol. 144, no. 12, 2022.

[29] H. J. Tulleken, "Grey-box modelling and identification using physical knowledge and bayesian techniques," *Automatica*, vol. 29, 1993.

[30] M. Kennedy and A. O'Hagan, "Predicting the output from a complex computer code when fast approximations are available," *Biometrika*, vol. 87, pp. 1–13, 2000.

[31] L. L. Gratiet and J. Garnier, "Recursive co-kriging model for design of computer experiments with multiple levels of fidelity," *Intl. J. Uncertainty Quantification*, vol. 4, pp. 365–386, 2014.

[32] L. Brevault *et al.*, "Overview of gaussian process based multi-fidelity techniques with variable relationship between fidelities, application to aerospace systems," *Aerospace Science & Technology*, vol. 107, 2020.

[33] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *32nd AAAI*, 2018, pp. 3215–3222.

[34] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *32nd AAAI*, 2018, pp. 4131–4138.