

# The Crucial Role of Problem Formulation in Real-World Reinforcement Learning

Georg Schäfer<sup>\*†‡</sup>, Tatjana Kraus<sup>§</sup>, Jakob Rehr<sup>\*†</sup>, Stefan Huber<sup>\*†</sup>, Simon Hirllaender<sup>‡</sup>

<sup>\*</sup>Salzburg University of Applied Sciences, Salzburg, Austria

<sup>†</sup>Josef Ressel Centre for Intelligent and Secure Industrial Automation, Salzburg, Austria

<sup>‡</sup>Paris Lodron University of Salzburg, Salzburg, Austria

<sup>§</sup>University of Applied Sciences Kempten, Kempten, Germany  
georg.schaefer@fh-salzburg.ac.at

**Abstract**—Reinforcement Learning (RL) offers promising solutions for control tasks in industrial cyber-physical systems (ICPSs), yet its real-world adoption remains limited. This paper demonstrates how seemingly small but well-designed modifications to the RL problem formulation can substantially improve performance, stability, and sample efficiency. We identify and investigate key elements of RL problem formulation and show that these enhance both learning speed and final policy quality. Our experiments use a one-degree-of-freedom (1-DoF) helicopter testbed, the Quanser Aero 2, which features non-linear dynamics representative of many industrial settings. In simulation, the proposed problem design principles yield more reliable and efficient training, and we further validate these results by training the agent directly on physical hardware. The encouraging real-world outcomes highlight the potential of RL for ICPS, especially when careful attention is paid to the design principles of problem formulation. Overall, our study underscores the crucial role of thoughtful problem formulation in bridging the gap between RL research and the demands of real-world industrial systems.

**Index Terms**—Reinforcement Learning, Problem Formulation, Industrial Cyber-Physical System.

## I. INTRODUCTION

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment and iteratively adjusting its policy to maximize cumulative rewards [1]. This approach has shown impressive success in domains such as robotics [2], video gaming [3], and Cyber-Physical Systems (CPSs) [4]. In RL, the agent does not require an a priori model of the environment; instead, it learns directly from data generated through interaction, making it particularly attractive for complex or uncertain scenarios.

Industrial CPSs (ICPSs) combine physical processes with computational control, forming the backbone of many modern industrial systems in manufacturing, energy management, and smart transportation [4]. Conventional control approaches for ICPSs often rely on mathematical models, which are difficult to derive when facing dynamic, high-dimensional, or stochastic processes [5]. RL reduces this challenge by providing

Financial support for this study was provided by the Christian Doppler Association (JRC ISIA), the corresponding WISS Co-project of Land Salzburg, the European Interreg Österreich-Bayern project BA0100172 AI4GREEN and by the Federal Ministry of Education and Research (BMBF) under the project iCARus.

techniques that learn controllers or decision-making policies through direct experimentation or simulation. However, while RL offers significant potential, its application in real-world ICPSs remains limited. One main reason for this is the lack of standardization in problem formulation and broader engineering workflows.

In comparison to supervised learning pipelines, where data preprocessing, model selection, training, evaluation, and deployment are well-defined steps [6], in RL there are no universally systematic frameworks for problem design and experimentation established yet. This gap frequently leads to an overemphasis on hyperparameter tuning at the expense of carefully specifying the environment, reward function, as well as state and action spaces. Yet, thoughtful problem formulation is critical for ensuring that RL solutions align with physical constraints (e.g., temperature limits, actuator torque bounds) and industrial objectives (e.g., safety, production quality). In this paper, we claim that structured problem design is essential to closing the gap between RL research and its real-world application in ICPSs.

### A. Contributions

- 1) We propose a set of structured problem-design principles for RL in ICPSs.
- 2) We systematically examine how factors such as normalization, target signal randomization, horizon lengths, initial state distributions, and action penalties influence training stability and performance.
- 3) We validate these design principles on a one-degree-of-freedom (1-DoF) helicopter testbed, highlighting their real-world applicability and efficiency.
- 4) We demonstrate that RL can be trained directly on physical hardware without relying on a priori models, underscoring its practical feasibility for industrial settings.

## II. DESIGN PRINCIPLES OF RL PROBLEM FORMULATION

Formulating the problem in RL is crucial to the success of the learning process. The interactions between the agent and the environment are often described by an infinite-horizon, discounted Markov Decision Process (MDP)

$$M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mu)$$

where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  the action space,  $\mathcal{P}$  the transition function,  $r$  the reward function,  $\gamma$  the discount factor, and  $\mu$  the initial state distribution [7].

However, in practical settings (particularly in real-world engineering tasks), one often adopts a *finite-horizon* or *episodic* approach to training and evaluation. Although the horizon length  $T$  is not strictly part of the MDP formalism, it plays a central role in how we structure each training episode, terminate interactions, and evaluate performance. Following Puterman’s classification [8], the decision epochs may be bounded by a finite horizon or treated in an infinite-horizon framework.

#### A. Components of Markov Decision Processes

1) *State Space*: The state space  $\mathcal{S}$  defines all possible configurations the agent can encounter in the environment, forming the basis for decision-making. It can be discrete or continuous, and must be designed to satisfy the Markov property, where future states depend only on the current state and action [1], influencing the choice of suitable RL algorithms. An effective state space must include all necessary information for optimal decision-making, while avoiding excessive complexity that leads to the *curse of dimensionality* [9]. When key information is missing from the state representation (partial observability), recurrent neural networks or additional sensor inputs may be required [10].

2) *Action Space*: The action space  $\mathcal{A}$  is the set of all possible actions the agent can take in any given state. It may be discrete (e.g., specific commands) or continuous (e.g., real valued control adjustments). An effective action space should reflect physical constraints of real-world systems. Hierarchical action structures [11] for complex tasks may improve scalability and efficiency. Details on action space shaping can be found in [12].

3) *Transition Function*: The transition function  $P(s'|s, a)$  describes the environment’s dynamics, determining the probability of transitioning from state  $s$  to state  $s'$  after taking action  $a$ . The transition function should align with the Markov property, ensuring that future states depend solely on the current state and action. This property is aligned with Linear Time-Invariant (LTI) state space models, where the following system state is determined solely by current inputs and the current state [13]. The shared principles of the Markov property in RL and time invariance in control theory highlight the compatibility of RL methods, such as those based on MDPs [14], with control-theoretic frameworks.

In real-world applications, the transition function is generally not explicitly available, requiring either approximation through simulation or learning from interaction data. Model-based RL approximates or uses a known transition function to simulate dynamics, enabling greater sample efficiency by reducing the interactions with the environment [15]. However, inaccuracies in the model can lead to suboptimal policies.

In contrast, model-free approaches, such as Q-Learning [16] and PPO [17], bypass this explicit modeling, relying on interaction data to learn policies or value functions directly,

thus typically require more environment interactions to achieve comparable performance. The choice between model-based and model-free methods depends on task complexity and computational constraints, with hybrid approaches emerging as promising solutions [18].

4) *Reward Function*: The reward function guides the agent by assigning rewards for transitions. It is substantial to design the reward function with the task’s goals. Sparse rewards simplify the design, but provide limited feedback, which slows learning. Dense rewards, on the other hand, accelerate learning by rewarding intermediate steps but risk introducing noise or instability if poorly designed. Proper weighting of multiple objectives (e.g., efficiency and safety) is essential to avoid unintended behavior. Reward shaping, a term formalized by Ng et al. in [19], describes a technique in RL where additional rewards are designed and added to the environment to guide the agent’s learning process. Balancing between sparse and dense rewards is a critical challenge that can significantly affect the efficiency and effectiveness of learning algorithms [20], [21].

5) *Discount Factor*: The discount factor  $\gamma \in [0, 1)$  determines the importance of future rewards in the agent’s decision-making process [7]. Higher  $\gamma$  values encourage long-term planning, whereas lower  $\gamma$  values prioritize immediate rewards [1].

6) *Initial State Distribution*: The initial state distribution  $\mu \in \Delta(\mathcal{S})$  where  $\Delta(\mathcal{S})$  is the space of probability distributions over  $\mathcal{S}$  defines the probability of the system starting in the initial state  $s_0$  [7]. It plays a critical role in shaping the agent’s initial interactions with the environment and the subsequent learning process.

- *Relevance*: The initial distribution should cover realistic scenarios to ensure the agent’s learned policy is trained on the practical applications.
- *Diversity*: A diverse initial state distribution can improve the agent’s reliability by exposing it to a wide range of starting configurations, reducing the likelihood of overfitting to a specific set of initial conditions.
- *Alignment*: The choice of  $\mu$  should align with the task objectives and the environment’s expected usage, ensuring that critical states are not underrepresented during training.

In practice, the initial state distribution can be defined deterministically (e.g., a fixed starting state) or stochastically (e.g., sampled from a predefined distribution). Tasks involving exploration or environments with complex dynamics may benefit from stochastic initializations to avoid biasing the agent’s learning toward specific regions of the state space.

### III. A REAL-WORLD RL APPLICATION

The testbed used for the experiments is the Quanser Aero 2 system, configured in its 1-DoF mode. The system is equipped with two motors that control the pitch of the beam by adjusting the voltages applied to the motors. We define the control task as achieving and maintaining desired pitch angles over time by applying appropriate voltage inputs. The motors operate

within a voltage range of  $-24\text{ V}$  to  $24\text{ V}$ . To determine the performance of the control strategy, the deviation to the target is considered. Although relatively compact, the Quanser Aero 2 features non-linear system dynamics that mirror many of the challenges found in industrial control processes, making it an effective and representative platform for evaluation RL approaches.

#### A. Simulation and Real-World Environments

The experiments were conducted in both a simulation environment and on the real-world environment. A detailed explanation of how these environments were accessed can be found in [22]. A description of the model is proposed in [23].

- 1) *Simulation Environment*: The system was first modeled in Simulink to allow for a safe and controlled training process, meaning that no actual hardware is exposed to risk (e.g., the beam cannot crash into its socket or cause damage if a suboptimal policy is attempted).
- 2) *Real-world Environment*: The most promising problem formulation identified in simulation was then deployed directly on the physical Quanser Aero 2, to evaluate the feasibility of training on directly on the real hardware.

#### B. Performance Metrics

The performance of the RL agent in this study is primarily evaluated using the average deviation to the target. This metric is defined as the mean absolute error between the actual pitch angle and the target pitch, and directly aligns with the base reward function used in the training process. Other common control performance metrics, such as steady-state deviation, overshoot and rise-time (as proposed in [23]), are excluded from this evaluation.

To evaluate the RL training performance, three additional criteria are considered. The first is training stability, measured by the standard deviation of returns across multiple training runs. A lower standard deviation indicates more consistent performance, reflecting the reliability of the training process. The second criterion is sample efficiency, measured as the number of training steps required to achieve the specified performance threshold of  $4^\circ$  average deviation to the target. This metric is particularly relevant for real-world applications where interaction time with the environment is limited. Finally, the average absolute voltage applied to the system is measured, reflecting the energy efficiency of the policy. All performance metrics were recorded on an evaluation environment identical to the environment from the baseline problem formulation.

#### C. Baseline Problem Formulation

The original problem formulation in [23] employed a time-limited setup with a fixed target profile. This choice was motivated by practicality and by the goal of maintaining a straightforward, comparable setup. Inspired by this formulation, we define the baseline problem formulation for our experiments as follows:

- *State Space*: For each time step  $t$ , the state is defined as  $(\Theta_t, \omega_t, r_t)$ , where  $\Theta_t$  represents the pitch angle,

$\omega_t = \Theta_t - \Theta_{t-1}$  denotes angular velocity, and  $r_t$  is the desired target angle. This formulation was chosen to simplify the agent’s state representation while still capturing the essential dynamics of the system, as the pitch and angular velocity naturally represent the states of the mechanical system.

- *Action Space*: The action space consists of a single continuous action, which represents the voltage applied to the motors. This action space was bounded by the physical constraints of the system ( $-24\text{ V}$  to  $24\text{ V}$ ). The motors operate in opposition, meaning one motor applies positive voltage while the other applies an equal magnitude of negative voltage.
- *Reward Signal*: The reward is defined as the negative absolute difference between the current pitch angle  $\Theta_t$  and the target pitch angle  $r_t$ , i.e.,  $R_t = -|\Theta_t - r_t|$ . If the system truncates ( $|\Theta_t| \geq \pi/2$ ), the reward is scaled by the remaining time steps, penalizing premature truncation.

The test task is structured as follows: the system aims to achieve a sequence of target angles over an 80 s time frame with a sample time of 0.1 s. Every 10 s, the target pitch is updated to a new value, following the predefined profile  $0^\circ$ ,  $5^\circ$ ,  $-5^\circ$ ,  $20^\circ$ ,  $-20^\circ$ ,  $40^\circ$ ,  $-40^\circ$ ,  $0^\circ$ . This formulation provides a controlled setting to evaluate the agent’s ability to track a series of reference angles.

### IV. CRUCIAL ASPECTS OF PROBLEM FORMULATION

Effective problem formulation is essential for successful RL applications in ICPSs. Subtle adjustments to states, actions, and rewards can significantly impact training efficiency and performance.

#### A. Improvements to the Problem Formulation

Building on the core components of MDPs, we propose five hypotheses (A–E) to enhance RL outcomes in this context:

- A) *Normalization Improves Convergence and Sample Efficiency*: Large or inconsistent input ranges can destabilize gradient updates in deep RL. Normalization of states, actions, or rewards can help constrain the domain of the learning signal, improving gradient flow and accelerating exploration.
- B) *Randomizing Target References Improves Generalization*: With a fixed target trajectory, the agent overfits rather than learning a more flexible control policy. Randomizing the target references across episodes broadens the range of scenarios encountered during training.
- C) *Longer Episodes Provide Richer Trajectories and Accelerate Learning*: Short episodes can limit the agent’s exposure to delayed rewards and reduce state-space exploration.
- D) *Randomizing Initial States Encourages Better Exploration*: Always starting from the same initial condition restricts early exploration to a narrow region of the state space.
- E) *Combining All Factors Leads to Stable and High-Performing Convergence*: Each of the above techniques

addresses weaknesses in RL problem formulation. When applied together, these techniques should create a complementary effect, leading to improved training stability, faster convergence, and higher overall performance by addressing multiple challenges simultaneously.

To test these hypotheses, we define two main configurations: (i) the original problem definition from Sec. III-C called “Baseline”, and (ii) applying all improvements simultaneously to a configuration called “New setting”. The differences of those two environments is summarized in Table I.

TABLE I  
PARAMETER COMPARISON OF “BASELINE” AND “NEW SETTING”.

	Baseline	New setting
stop time	80 s	100 000 s
target tilt	fixed	random
initial tilt	0°	random
norm obs	no	yes
norm action	no	yes
action penalty	0.0	0.25

### B. Studies on the Influence of Suggested Improvements

For each hypothesis A–D, we conduct an addition and an ablation study:

- “Baseline” with “New” Parameter: We start from the “Baseline” and apply one change from the “New setting” (e.g., just add normalization).
- “New Setting” with “Baseline” Parameter: We start from the “New setting” and change that parameter to match the “Baseline” configuration (e.g., remove normalization).

This isolates the direct effect of each parameter. We use Proximal Policy Optimization (PPO) from the Stable Baselines3 (SB3) library [24] with default hyperparameters and train for 1 million steps, repeating each experiment over 10 seeds. Every 10 000 steps, an evaluation run is triggered. This allows us to record (i) how many steps it takes to reach an average deviation of 4° or better on the evaluation profile, (ii) final performance metrics (mean/min/max/standard deviation of pitch deviation) after 1 million steps, (iii) the average absolute voltage applied, indicating how aggressively the agent acts. Additionally, we conduct a training run on the real system using the parametrization from the “New setting”, starting the training process from scratch without utilizing any pretrained agent.

### C. Implementation Details of Suggested Improvements

1) *Stop Time*: The simulation horizon was increased from 80 s to 100 000 s, allowing the agent to explore a longer sequence of states. With 1 000 000 training steps and a sample time of 0.1 s this results in a single episode per training run if no truncation occurs.

2) *Random Target Tilt*: Rather than training with a single, fixed target tilt profile, the “New setting” environment dynamically changes the target. At each time step, there is a 1% probability that the target tilt is randomly redrawn from the

range of  $-40^\circ$  to  $40^\circ$ . On average, this results in a target change every 10 s (i.e., every 100 steps), ensuring that the agent does not converge to a narrow policy specialized to a single tilt profile. Introducing this variation fulfills the Markov property, as the environment state no longer follows a static target.

3) *Random Initial Tilt*: When the environment is reset, the tilt angle is uniformly sampled from a range of  $-40^\circ$  to  $40^\circ$  in the “New setting”, instead of being fixed at  $0^\circ$ . This introduces additional variability into each trial and forces the agent to adapt from a variety of starting positions.

4) *Observation Normalization*: In the “New setting”, each component of the observation vector is normalized by its maximum expected magnitude. Specifically, the pitch and target angles are divided by  $\pi/2$ , while the velocity is scaled by a factor of 0.2441, based on empirical measurements. This normalization ensures that the inputs to the agent remain within specific ranges ( $-1$  to  $1$ ) and helps stabilize the learning process.

5) *Action Normalization*: The action space is rescaled from direct voltages values from the range of  $-24$  V to  $24$  V to a normalized range of  $-1$  to  $1$ . To apply the action in the actual system, the agent’s chosen action  $a \in [-1, 1]$  is multiplied by the maximum voltage of  $24$  V. Hence, the true control voltage becomes  $\text{voltage} = 24 \cdot a$ .

6) *Action Penalty*: Finally, an action penalty factor was added to the reward function in the “New setting” to discourage large fluctuations in the applied voltage. Over a 1-second window, the standard deviation of the normalized applied voltage is computed and then multiplied by the action penalty factor (0.25). This term is subtracted from the original reward, thus encouraging smoother voltage profiles.

By incorporating these modifications, either individually (for the hypothesis tests A–D) or collectively (in the full “New setting”) for hypotheses E, we can isolate and examine the effect of each parameter choice and observe how it impacts the agent’s learning progress.

### D. Remark on Real-World Applications

When training RL on a physical setup, additional considerations are necessary to preserve hardware integrity and ensure safe operation. In simulation, aggressive of “bang-bang” control policies may appear highly effective, as virtual environments do not exhibit real-world wear-and-tear. In contrast, physical motors, actuators, and sensors face constraints such as temperature limits, friction, and mechanical stress. Repeatedly applying large, abrupt control signals can lead to overheating, premature failures, or damage to the system’s components. To mitigate these risks, we introduce an action penalty term that motivates smoother, more gradual voltage changes. This adjustment not only improves the longevity of the hardware but also encourages the agent to learn control policies that avoid abrupt power surges.

## V. EXPERIMENTAL RESULTS AND DISCUSSION

In Table II, we summarize the performance of the different experiments based on the results of 10 repeated training runs

each. For each metric and each experiment we assume that the outcomes follow a normal distribution. To compare two experiments for a given metric, we essentially ask for the probability, when we draw a sample of each, that the value for one experiment would be less (or greater) than the other.

Say we have  $X_1 = \mathcal{N}(\mu_1, \sigma_1)$  for experiment 1 and  $X_2 = \mathcal{N}(\mu_2, \sigma_2)$  for experiment 2 then  $X_1 - X_2$  is distributed  $\mathcal{N}(\mu_1 - \mu_2, \sqrt{\sigma_1^2 + \sigma_2^2})$  and we ask for the probability  $P(x_1 < x_2)$  when  $x_1 \sim X_1, x_2 \sim X_2$ . This leads to the score

$$z(X_1, X_2) = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}$$

used as follows:  $P(x_1 < x_2)$  is equal to the probability that  $N(0, 1)$  gives a value greater than  $z(X_1, X_2)$ , i.e., the smaller  $z(X_1, X_2)$  the higher the probability for  $x_1 < x_2$ . This underpins the following notation: We call  $X_1$  is significantly better (smaller) than  $X_2$  (indicated by  $\uparrow$ ) when  $z(X_1, X_2) < -1$ ; we call  $X_1$  significantly worse (greater) than  $X_2$  (indicated by  $\downarrow$ ) when  $z(X_1, X_2) > 1$ , and otherwise there is no significant difference (indicated by  $\sim$ ).

#### A. Analysis of Hypotheses (A–E)

We summarize our findings for each hypothesis below:

- *Normalization (Hypothesis A)*: Normalizing states and actions emerges as a key factor in stabilizing and expediting training. In the “Baseline” setting, normalization alone enables the agent to reach the  $4^\circ$  threshold in every run. In the “New setting”, removing normalization significantly degrades performance. Additional experiments isolating action and state normalization confirm that action normalization exerts a greater influence in our case: the observation space already lies in a relatively low range, while action space spans  $-24$  V to  $24$  V.
- *Random Targets (Hypothesis B)*: Allowing the agent to experience a variety of target reference angles during training improves sample efficiency in the “Baseline” configuration and does not yield a significant impact by removing it from the “New setting”. To analyze the actual impact of the generalizability, additional evaluations runs must be conducted using a variety of target trajectories, which is out of scope of this work.
- *Longer Episodes (Hypothesis C)*: Increasing the horizon provides more continuous state-space exploration before resets occur. In the “Baseline”, this markedly enhances performance, whereas in the “New setting” its effect is less pronounced.
- *Random Initial Pitch (Hypothesis D)*: Introducing a stochastic range of initial pitch angles encourages better coverage of the state space during early training. Although it greatly benefits the “Baseline”, in the “New setting” the effect is smaller, as long episodes reduce reliance on any single initial condition.
- *Combining All Factors (Hypothesis E)*: Aggregating these individual improvements yields the fastest and most stable convergence. The “New setting” outperforms the

“Baseline” in both sample efficiency and final performance, underscoring the complementary nature of these design choices.

Even though the “Baseline” configuration appears to yield lower action magnitudes, this can be deceptive: the agent may simply fail to achieve the goal and therefore minimizing control effort (i.e., doing almost nothing). By contrast, the “New setting” applies notably high control signals, effectively resembling a “bang-bang” strategy. Introducing an action penalty term into the “New setting” does not degrade performance; in fact, it marginally improves the final policy while substantially reducing abrupt control actions, thereby mitigating hardware stress when deployed to the real system.

Fig. 1 illustrates the training performance, measured as the average deviation from the target, over 1 million training steps. The “New setting” demonstrates superior performance compared to the “Baseline”, excelling in sample efficiency, overall performance, and stability. Training on the real system reached the predefined goal of  $4^\circ$  within 200 000 steps and quickly converged to within  $5^\circ$  degrees. Fig. 2 provides a detailed view of the actual pitch, the target, and the applied voltages of the agent trained on the real system after 250 000 steps.

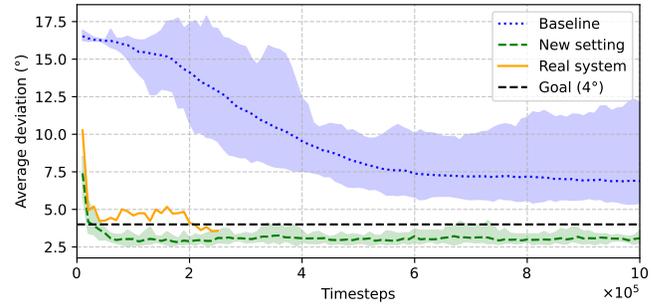


Fig. 1. Average deviation to the target for the “Baseline” and “New setting” configurations on the evaluation profile during training. The “Baseline” and “New setting” configurations were trained on the simulation model for 1 million steps. Additionally, the plot includes the results for the “New setting with action penalty” configuration trained on the real system for 250 000 steps.

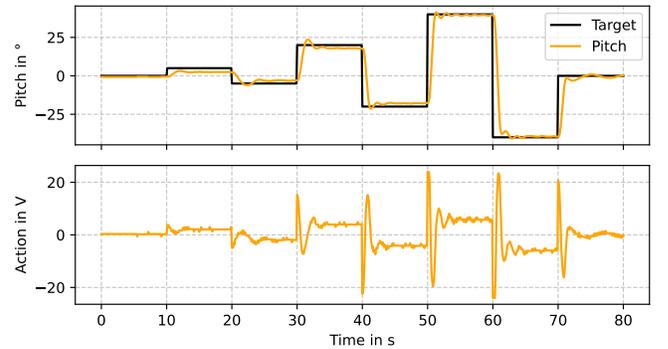


Fig. 2. Evaluation on the real system with an agent trained for 250 000 steps, showing the target pitch, actual pitch, and applied voltage.

TABLE II  
COMPARISON OF THE DIFFERENT EXPERIMENTS DERIVED FROM THE HYPOTHESES (A–E).

Method	steps to 4°	z	deviation (°)	z	voltage (V)	z
Baseline	n/a ± n/a (0%)		6.88 ± 1.94		2.81 ± 0.37	
Baseline with normalization	63 000 ± 22 825 (100%)	n/a ↑	2.91 ± 0.23	-2.03 ↑	8.78 ± 5.42	1.10 ↓
Baseline with random targets	940 000 ± 0 (10%)	n/a ↑	6.37 ± 1.66	-0.20 ~	2.84 ± 0.38	0.06 ~
Baseline with long episodes	886 667 ± 26 247 (30%)	n/a ↑	5.47 ± 2.04	-0.50 ~	3.15 ± 0.79	0.39 ~
Baseline with random initial pitch	872 500 ± 83 179 (40%)	n/a ↑	4.80 ± 1.21	-0.91 ~	3.37 ± 0.55	0.85 ~
New setting	40 000 ± 11 832 (100%)		3.05 ± 0.15		21.47 ± 2.28	
New setting without normalization	595 000 ± 123 119 (60%)	4.49 ↓	3.72 ± 1.27	0.53 ~	3.84 ± 0.63	-7.45 ↑
New setting without random targets	42 000 ± 22 271 (100%)	0.08 ~	2.88 ± 0.23	-0.63 ~	10.29 ± 6.36	-1.65 ↑
New setting without long episodes	35 000 ± 9220 (100%)	-0.33 ~	2.97 ± 0.31	-0.24 ~	13.00 ± 6.12	-1.30 ↑
New setting without random initial pitch	34 000 ± 14 967 (100%)	-0.31 ~	3.05 ± 0.16	0.01 ~	21.30 ± 2.22	-0.05 ~
New setting with action penalty	26 000 ± 9165 (100%)	-0.94 ~	3.08 ± 0.23	0.11 ~	4.29 ± 0.19	-7.51 ↑
New setting with action penalty on real system	200 000		3.58		3.85	

## VI. CONCLUSION AND FUTURE WORK

This paper showed that minor yet carefully chosen adjustments to RL problem formulation can significantly boost performance, training stability, and efficiency in ICPSs. Our experiments on a 1-DoF helicopter testbed confirm that normalization, randomizing targets and initial states, extending episodes to horizons, and reward shaping all foster more reliable and sample-efficient learning. Moreover, training directly on the real hardware without a prior model illustrates the feasibility of real-world RL applications. Future research will focus on establishing a robust RL engineering pipeline for ICPSs, integrating advanced approaches such as data-driven probabilistic Model Predictive Control (MPC) or physics-informed RL, and incorporating additional performance metrics for further optimization.

## REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [2] B. Singh, R. Kumar, and V. P. Singh, “Reinforcement learning in robotic applications: a comprehensive survey,” *Artificial Intelligence Review*, vol. 55, no. 2, pp. 945–990, 2022.
- [3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [4] X. Liu, H. Xu, W. Liao, and W. Yu, “Reinforcement learning for cyber-physical systems,” in *2019 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 2019, pp. 318–327.
- [5] P. Sánchez-Sánchez and M. A. Arteaga-Pérez, “Simplified methodology for obtaining the dynamic model of robot manipulators,” *International Journal of Advanced Robotic Systems*, vol. 9, no. 5, p. 170, 2012.
- [6] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media, Inc., 2022.
- [7] A. Agarwal, N. Jiang, S. M. Kakade, and W. Sun, “Reinforcement learning: Theory and algorithms,” *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep.*, vol. 32, p. 96, 2019.
- [8] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [9] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

- [10] X. Xiang and S. Foo, “Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (pomdp) problems: Part 1—fundamentals and applications in games, robotics and natural language processing,” *Machine Learning and Knowledge Extraction*, vol. 3, no. 3, pp. 554–581, 2021.
- [11] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, “Hierarchical reinforcement learning: A comprehensive survey,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.
- [12] A. Kanervisto, C. Scheller, and V. Hautamäki, “Action space shaping in deep reinforcement learning,” in *2020 IEEE conference on games (CoG)*. IEEE, 2020, pp. 479–486.
- [13] C.-T. Chen, *Linear system theory and design*. Oxford University Press, Inc., 1995.
- [14] S. Kamthe and M. Deisenroth, “Data-efficient reinforcement learning with probabilistic model predictive control,” in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1701–1710.
- [15] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, “Model-based reinforcement learning: A survey,” *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.
- [16] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, pp. 279–292, 1992.
- [17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [18] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, “Combining model-based and model-free updates for trajectory-centric reinforcement learning,” in *International conference on machine learning*. PMLR, 2017, pp. 703–711.
- [19] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *Icml*, vol. 99, 1999, pp. 278–287.
- [20] J. Eschmann, “Reward function design in reinforcement learning,” *Reinforcement Learning Algorithms: Analysis and Applications*, pp. 25–33, 2021.
- [21] S. Ibrahim, M. Mostafa, A. Jnadi, and P. Osinenko, “Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications,” *arXiv preprint arXiv:2408.10215*, 2024.
- [22] G. Schäfer, M. Schirl, J. Rehrl, S. Huber, and S. Hirllaender, “Python-based reinforcement learning on simulink models,” in *11th International Conference on Soft Methods in Probability and Statistics (SMPS 2024)*, Salzburg, Austria, Sep. 2024.
- [23] G. Schäfer, J. Rehrl, S. Huber, and S. Hirllaender, “Comparison of Model Predictive Control and Proximal Policy Optimization for a 1-DOF Helicopter System,” in *2024 IEEE 22nd IEEE International Conference on Industrial Informatics (INDIN)*. Beijing, China: IEEE, Aug. 2024.
- [24] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: <http://jmlr.org/papers/v22/20-1364.html>