Advancing Reinforcement Learning Control: Continuous Scheme and Curriculum Learning for Mechanical Systems

Oleg Rogov[1,2], Peter Manzl[3], Johannes Gerstmayr[3], and Grzegorz Orzechowski[2]

[1]Savonia University of Applied Sciences
[2]Lappeenranta-Lahti University of Technology LUT
[3]University of Innsbruck

September 24, 2024

Todo list

1 Introduction

Multibody System Dynamics (MSD) is a field of study that concentrates on modeling, simulating, and analyzing the dynamics of systems composed of interconnected rigid and flexible bodies. It is a fundamental approach to understanding and predicting the dynamic behavior of complex mechanical systems through sophisticated mathematical models and computational algorithms. The essence of MSD lies in its capacity to accurately describe the motion, forces, and interactions among the components of a system and with the environment, considering both translational and rotational movements and the constraints that govern these movements [17]. MSD is applicable across various engineering fields, from aerospace and automotive to biomechanics and robotics. This versatility underscores its capability to address the specific dynamics of different systems, enhancing its utility in diverse technological and scientific domains. Also, by accurately predicting system behavior, MSD supports the iterative design and optimization of mechanical systems. Engineers can use MSD to test and refine designs virtually, reducing the need for physical prototypes and accelerating the development process. One important characteristic of MSD is that the detailed models developed using this method make it an excellent foundation for implementing learning-based or model-based control strategies.

These strategies, such as Reinforcement Learning (RL) [18] and Model Predictive Control [16], can dynamically adapt to the system's behavior, leading to optimized performance even in complex and changing environments.

Reinforcement Learning, a primary control method used in our research, is a branch of Machine Learning (ML) where an agent learns to make decisions by taking actions in an environment to maximize the cumulative reward under acting according to a certain policy [18]. In RL, rewards may not be immediate. An agent may need to perform a series of actions before receiving a reward, creating a delay between the initial action and the eventual reward. The learning process involves exploring various actions and states (exploration) and exploiting the knowledge gained to make better decisions (exploitation). The exploration and exploitation trade-off is one of the crucial concepts in RL, which emphasizes the need for a balance that results in efficient agent training. Achieving this balance is essential for the agent to learn effectively and optimize its performance over time. RL can be divided into two main types: model-free methods and model-based methods. The main difference between them is that in model-free methods the agent learns directly from interactions with the environment without trying to understand its dynamics, while in model-based methods the agent learns a model of the environment's dynamics and uses this model to plan and make decisions. Model-free methods are generally simpler and more robust in highly dynamic and complex environments, though they require more interactions for an agent to learn effectively. Additionally, RL can be further categorized into traditional RL and Deep Reinforcement Learning (DRL). Traditional RL focuses on methods where the agent typically operates with simpler, lower-dimensional state spaces. These methods often rely on tabular representations or linear function approximations. DRL, however, specifically integrates deep neural networks to approximate value functions or policies, so that it can effectively handle high-dimensional state spaces and complex environments. This capability allows DRL to be applied to tasks such as playing video games, robotic control, and autonomous driving [14, 11, 20]. Examples of DRL methods are Advantage-Actor Critic (A2C), Proximal Policy Optimization (PPO) and Deep Q-Network (DQN). A2C is an advanced version of the actor-critic method that uses both policy (actor) and value (critic) functions, with the actor updating the policy based on feedback from the critic, enhancing stability and performance [10]. PPO is a popular policy gradient method known for its robustness and simplicity, utilizing a clipped objective function to ensure stable learning [15]. DQN combines Q-learning [18] with deep neural networks to handle high-dimensional state spaces, approximating the Q-value function, which represents an expected cumulative reward an agent can achieve by taking a specific action in a given state and following the optimal policy, enabling it to learn effective policies in complex environments [11]. In Reinforcement Learning, actions (control inputs) can be either discrete or continuous. Discrete actions are suitable for tasks where the set of possible actions is limited and distinct, such as choosing between different predefined options. Continuous

actions, on the other hand, are used in scenarios where actions can take any value within a range, such as adjusting the speed of a car or the angle of a robotic arm.

The application of RL techniques in the context of MSD is still an emerging field, with limited research available. Current studies have primarily focused on traditional control methods, leaving significant potential for further exploration and development of RL-based approaches to enhance the adaptability and efficiency of MSD models. A noticeable study in this area is done by Kurinov et al. [8]. Researchers have developed an autonomous excavator system utilizing RL as a control base integrated with MSD (detailed simulation with hydraulics and sensors). This study was focused on enabling the excavator to perform various tasks autonomously, including excavation and loading. The effectiveness of their approach was demonstrated in achieving autonomous operation with minimal collisions, showcasing the potential for automation in heavy machinery within the construction and mining industries. Additionally, Egli et al. [4] explored the application of RL for adaptive control in hydraulic excavators, aiming to enhance the automation of excavation tasks. Their study introduced a controller capable of adjusting to varying soil properties within a single excavation cycle. Utilizing an RL-based approach, they trained a control policy in simulation, integrating proprioceptive measurements to infer soil characteristics. This method demonstrated significant improvements in adaptability and efficiency compared to traditional excavation techniques. Notably, their controller could operate without explicit soil parameter inputs, relying instead on measurements from hydraulic pressure and kinematic sensors. The experiments conducted on a 12-ton excavator confirmed the controller's ability to maintain high performance across diverse soil conditions, emphasizing the feasibility of RL in real-world excavation scenarios. Furthermore, Osa and Aizawa [13] investigated deep reinforcement learning with adversarial training for automated excavation. They proposed a novel regularization method that employs virtual adversarial samples to mitigate the overestimation of Q-values in a Q-learning algorithm. Their approach, which integrates depth images for excavation planning, demonstrated superior sample efficiency and robustness in policy learning compared to traditional actor-critic methods [18]. This research highlights the importance of incorporating visual information and advanced regularization techniques to enhance the reliability and effectiveness of RL-based systems.

While RL has shown considerable promise as a control method for complex dynamic systems, it is not without its limitations. One of the primary challenges of RL is its high demand for extensive interactions with the environment to learn effective policies, which can result in long training times and high computational costs. Additionally, RL algorithms often struggle with the exploration-exploitation trade-off, especially in high-dimensional state spaces and non-linear dynamics typical of MSD. These factors can make RL less efficient and robust when applied directly to complex mechanical systems [3]. This is where Curriculum Learning (CL) can provide substantial benefits. CL is about structuring training content and learning

experiences in a way that mirrors the progression from simple to complex, akin to how a curriculum in educational settings is designed [1]. This method may not only accelerate the learning process but also enhance the robustness and adaptability of the control strategies. Research by Hacohen et al. [7] delves into the effectiveness of CL in the training of deep neural networks, with a focus on how CL can significantly impact the efficiency of the learning process. Experiments to analyze the impact of curriculum-based learning on the training efficiency of deep neural networks were conducted and the curriculum was structured around a series of tasks, each representing a different level of difficulty, designed to challenge the neural network progressively. This method allowed for the examination of how neural networks adapt their learning strategies in response to escalating task complexity. This work suggests that by implementing a curriculum-based approach deep RL algorithms can develop more robust and effective strategies.

The work by Narvekar et al. [12] is particularly instructive for our research as it provides a taxonomy of CL methodologies that can be directly applied to the RL challenges we could face in tandem with MSD. They categorize CL strategies into three main approaches: task sequencing, transfer learning, and multi-task learning. Task sequencing involves arranging learning tasks in a meaningful order to gradually increase complexity, thereby enhancing the learning efficiency and robustness of RL agents. Transfer learning focuses on leveraging knowledge gained from previous tasks to improve performance on new, related tasks. Multi-task learning allows simultaneous training on multiple tasks to share knowledge and improve overall learning efficiency. By structuring the learning experiences from simple to complex tasks, we can systematically improve the RL agent's ability to handle the difficulties of mechanical systems in MSD. In our research an implementation of transfer learning can be useful, since we can start training inverted pendulum on a cart as a simple one-link system and then increase the complexity of it by adding more links. This systematic enhancement of learning experiences can mitigate some of the efficiency and robustness issues traditionally associated with RL in dynamic, high-dimensional environments.

Furthermore, Weinshall's work [19] have delved into the integration of TL within CL, highlighting how accumulated knowledge can significantly boost performance in more complex systems. Their work demonstrates that using a pre-trained neural network on a different task to guide the curriculum can approximate an ideal learning sequence, leading to faster convergence and improved generalization.

The study written by Gupta et al. [6] investigates how integrating CL can enhance RL for continuous control tasks, which are highly relevant to MSD. Their research focuses on tasks such as robotic arm manipulation and locomotion, demonstrating that various CL strategies, like progressive task sequencing and parameter adaptation, significantly improve learning efficiency and performance.

A paper by Bhati et al. [2] demonstrates the effectiveness of CL in a multi-agent RL environment. A sequential task structure was implemented where each task

increased in complexity and inter-agent dependency. This approach was designed to incrementally develop and refine the cooperative strategies of individual agents within a simulated multi-agent environment, allowing researchers to observe how agents adapted and optimized their performance in progressively challenging scenarios. This finding is crucial for tasks where teamwork among agents is a key factor.

The perspectives of using Curriculum Reinforcement Learning (CRL) in mechanical system dynamics (MSD) are promising. CRL, which integrates Curriculum Learning with Reinforcement Learning, offers a structured approach to tackle the complexities of MSD by gradually increasing the difficulty of control tasks. This methodology enhances learning efficiency and improves the adaptability and robustness of RL agents in managing dynamic mechanical systems. Additionally, continuous RL plays a crucial role in this context. Unlike discrete RL, continuous RL allows for more precise and smooth control by operating within a continuous action space. This capability is an important point for the system of our study, the inverted multi-link pendulum on a cart, where maintaining stability and controlling the pendulum's oscillations demand precise and continuous adjustments to manage the complex, nonlinear dynamics of the system effectively. In our prior work by Manzl et al. [9], we evaluated RL algorithms for controlling mechanical systems using discrete control methods, effectively stabilizing single to triple link inverted pendulum systems. The RL approach demonstrated adaptability and effectiveness in managing the dynamics and complexities of these models. Our current study extends this work by applying a novel continuous control method combined with CL technique. Using the same multi-link inverted pendulum on a cart system, we can clearly demonstrate the improvements afforded by these advanced methods. By switching from discrete to continuous action space, we enable smoother and more precise adjustments in system behavior with the faster training time, which are critical when managing the complex dynamics of multibody systems. Moreover, integrating CL accelerates the training process more, allowing RL agents to quickly adapt to increasingly complex tasks. This dual approach not only streamlines control operations but also significantly enhances learning efficiency. Our empirical evidence demonstrates the effectiveness of these strategies in a controlled mechanical environment, highlighting their practical implications. This research paves the way for developing more advanced, autonomous control systems that could transform interactions with mechanical systems across various industries.

2 Methods

2.1 Mechanical model

The system under investigation is a multi-link inverted pendulum on a cart. This setup, known for its inherent instability and dynamic nature, is a cornerstone in

control theory [5]. The cart, serving as the base platform for the pendulum links, is limited to linear motion along a horizontal track, simplifying the translational dynamics to one-dimensional motion along the x-axis. The cart's movement is controlled by an externally applied force F, which is crucial for the system's stabilization. The magnitude and direction of this force significantly impact the system's dynamics, providing a means to counteract the gravitational torque imposed by the pendulum links. The details of the system modelling are described in [9].

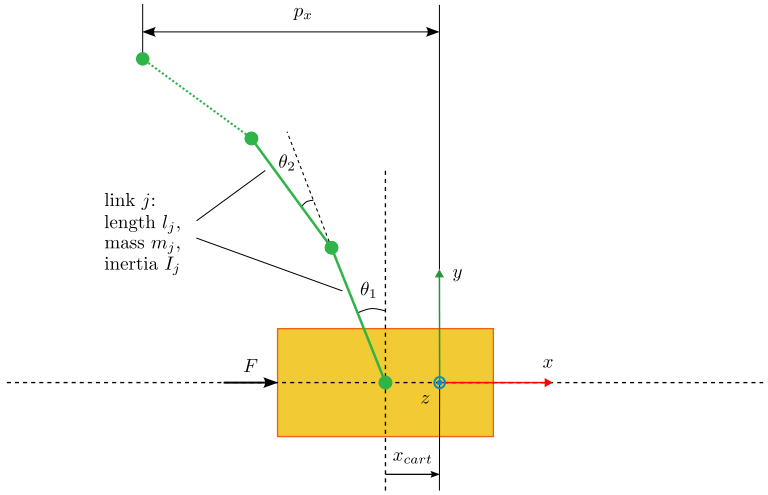


Figure 1: N-link inverted pendulum on a cart. Minimal coordinates formulation $\mathbf{q} = \mathbf{s} = [x_{\text{cart}}, \theta_1, \ldots, \theta_N]$ are used to model the system. Each link consists of length l_j, mass m_j, moment of inertia I_j. The cart is a rigid body with mass m_{cart}.

Attached to the cart is a series of n rigid links, each connected end-to-end by revolute joints. These joints allow for free rotational movement in the vertical plane, and each link j is characterized by mass m_j, length l_j and inertia I_j. This model, with its high degree of control difficulty and relevance to real-world applications, provides a valuable platform for testing sophisticated control algorithms, including those based on Reinforcement Learning.

2.2 Reinforcement Learning with continuous action space

Reinforcement Learning is a branch of machine learning where an agent learns optimal behavior through systematic interaction with a dynamic environment, aiming to maximize cumulative rewards. This learning paradigm is distinct from supervised learning; in RL, the agent is not explicitly instructed which actions to take. Instead, it must explore and discover which actions yield the highest rewards by

trial and error, a process often facilitated by a policy—a decision-making function that maps states of the environment to actions to be taken in those states [18].

In our research, RL is employed to train agents to manage the dynamics of a multi-link inverted pendulum on a cart, a challenging control problem that requires maintaining precise dynamic stability throughout the process. The reward function in RL plays a crucial role as it guides the learning process. For instance, a reward function can be designed to penalize the agent for excessive movement away from a target state or for using too much energy, while rewarding closer approximations of the desired state, such as maintaining the pendulum in an upright position. We use the reward function, which has shown one of the best performances from our previous study

$$r = 1 - w_p \frac{|p_x|}{\chi_{\text{cart}}} - (1 - w_p) \frac{\sum_{j=1}^{\text{N}} |\theta_j|}{\text{N}\chi_\theta} \tag{1}$$

It combines a tip position with the sum of pendulum link angles $\sum_{j=1}^{N} |\theta_j|$, weighted by a factor w_p. χ_{cart} and χ_θ are predefined, constant position and angle thresholds, which differ from 1-link to more link systems. N is the number of the links.

In RL, the choice between discrete and continuous action space might significantly affect the performance of learning algorithms. Discrete action space, used in our previous study, limits the agent's actions to a finite set of possibilities: it either applies a negative or a positive force of the same magnitude to regulate the behavior of the system. In a continuous action space the agent is provided with an interval of the control force

$$F = [-f_{\text{cart}}, +f_{\text{cart}}] \tag{2}$$

From this interval in Eq. (2) the agent is free to choose any suitable control force as an action for learning the stabilization task. Since it provides more possible actions then the discrete action space, it emerges in a faster training of an agent and achieves a more smooth and robust control task execution.

2.3 Model evaluation

For the model evaluation we use the same specifications as in our previous research of Manzl et. al. [9]. The performance of the agent is periodically evaluated based on the mean reward exceeding a defined threshold λ_r and the loss being below a threshold λ_l. During each evaluation, the agent undergoes n_{test} tests in an environment simulated for n_{eval} time steps, ensuring $n_{\text{eval}} > n_{\text{learn}}$. For instance, with a time step $h = 0.02$ s, $n_{\text{eval}} = 5000$ corresponds to a 100 s test duration. During the tests, the agent's initial states are perturbed within a range of $\pm x_{\text{init}}$, and the maximum norm of the state vector is used to compute the test error $e_{\text{test},i} = \|\mathbf{s}_i\|_\infty$ at each time step t_i. The total test error e_{test} is defined as the maximum error

encountered in the final quarter of the evaluation period:

$$e_{\text{test}} = \max(e_{\text{test},i}) \quad \forall\, i \geq \frac{3}{4} n_{\text{eval}}. \tag{3}$$

The training is deemed complete when e_{test} falls below a pre-determined threshold χ_{test} for all tests, providing a criterion for potential early stopping. To assess model robustness, several agents are trained and evaluated, with their performances depicted as envelopes formed by the best, worst, and mean error metrics across time steps. Notably, early evaluation results (within the first 50,000 steps) may be unreliable due to insufficient model development, necessitating the use of linear interpolation for clearer data interpretation.

2.4 Performance evaluation of PPO with continuous action space

For the evaluation of an action space benefits in our study we use the PPO RL-algorithm. PPO algorithm itself aims to improve policy gradient methods by optimizing a surrogate objective function while ensuring that updates to the policy do not deviate too far from the previous policy. It accomplishes this by using a clipped objective function that limits the size of policy updates, thereby stabilizing training and improving performance [15]. The evaluation is conducted on an N-link inverted pendulum system, specifically analyzing the performance with 1-link, 2-link, and 3-link systems. The analysis explores two different action spaces and their impact on the agent's ability to stabilize the pendulum in the upward-facing equilibrium. For all systems, we focus on the task of balancing the inverted pendulum in the unstable upward-facing equilibrium. The challenge increases with the number of links, as each additional link contributes to the overall instability of the system. The single pendulum is a well-known benchmark for RL methods, and this evaluation extends to more complex systems with additional links. The experiments are conducted using Exudyn Version 1.8.52 and stable-baselines3 Version 1.8.0. Each experiment is repeated 10 times with different random seeds to account for variations in initialization. The PPO algorithm is implemented with standard parameters, except where adjustments are made for the specific requirements of the environment.

Table 1 summarizes the hyperparameters used in the experiments, and Table 2 outlines the environment, reward, and training parameters for the different link configurations. Notably, the cart force is now continuous and is presented as an interval, ranging from $[-12, 12]$ N for the 1-link system, and similarly for the 2-link and 3-link systems.

The continuous action space, particularly in the control of the cart force, provides more nuanced control via the specified interval of the force, which is expected to influence the stabilization process, especially in the more complex multi-link systems.

Table 1: The hyperparameters used for the PPO method in the experiments. The physical parameters are provided in Table 2.

Parameter	**Value**	**Parameter**	**Value**
Reward function	r, Eq. 1,	Reward threshold	$\lambda_r = 0.9$
Step size	20 ms	Loss threshold	$\lambda_l = 0.01$
Evaluation length	5000 steps $\Rightarrow$ 100 s	PPO: n_{steps}	$n_{\text{episode,max}}$
Learning rate	$5 \cdot 10^{-4}$		

Table 2: Environmental, reward, and training parameters for the environments with link numbers 1 to 3.

Name	**Parameter**	**1 link**	**2 link**	**3 link**
Cart force	f_{cart} in N	$[-12, 12]$	$[-40, 40]$	$[-60, 60]$
Threshold cart position	χ_x in m	1.2	3.6	5.4
Threshold link angle For θ_1 to θ_N	χ_φ in rad	$\frac{\pi}{20}$	$\frac{\pi}{10}$	$\frac{3\pi}{20}$
Max test error	e_{test}	0.2	0.5	0.75
Reward position factor	w_p	0.5	0.5	0.8 to 1
Required training steps	n_{learn}	$80 \cdot 10^3$	$150 \cdot 10^3$	$350 \cdot 10^3$
Max episode length	$n_{\text{episode,max}}$	1280	1536	2048
Tests per evaluation	n_{test}	50	50	100

2.5 Curriculum learning implementation

Our approach to Curriculum Learning in the domain of MSD is rooted in the incremental introduction of complexity to the learning environment of the RL agent, which, according to the taxonomy provided in [12], is based on Transfer Learning. In our study, the concept of CL is physically represented through the implementation of mechanical spring-damper systems, which play a crucial role in regulating the dynamic behavior of the system under study. These spring-damper complexes are strategically positioned between different components of the system to modulate the interactions and stability during the learning process. A schematic view of this physical representation is illustrated in Figure 2. Here, we utilize two types of spring-damper systems: a translational spring-damper and a rotational spring-damper. The translational spring-damper is connected between the cart body and the ground. Its primary function is to restrict the translational movement of the cart along the x-axis. By controlling this movement, the translational spring-damper indirectly adjusts the speed of the pendulum system, ensuring stability during the initial learning phases. This stability is vital for the agent to safely explore the

system dynamics without being overwhelmed by excessive movement. The rotational spring-damper systems are attached to the revolute joints, regulating the angular displacement of the pendulum. These systems provide resistance to angular changes, thus preventing abrupt rotational movements that could destabilize the learning process. By controlling the angular behavior, the rotational spring-dampers ensure that the pendulum remains within a manageable range of motion during the early stages of learning.

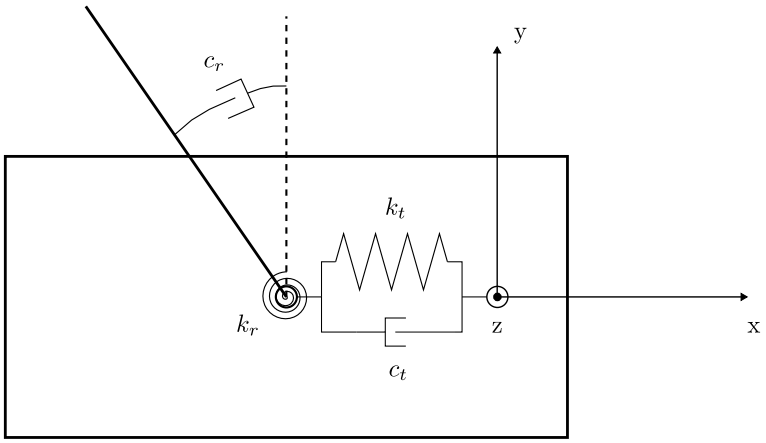


Figure 2: Scheme of the system with translational and rotational spring-dampers.

These mechanical components provide physical constraints that simplify the learning environment, enabling the agent to gradually adapt to the complex dynamics of the system. As the training progresses, the influence of these spring-damper systems is systematically reduced, allowing the agent to take more control over the system. In the later stages of training, the spring-dampers' effects are diminished, either by reducing their stiffness and damping coefficients or by removing them entirely, thus transferring full control to the learning agent. This transition is crucial for ensuring that the agent is not overly reliant on the physical constraints and is capable of managing the system dynamics independently. Following this physical setup, We have four parameters in our CL-scheme and their roles are explained in the Table 3.

Our approach involves predefined decay functions that dictate the rate and pattern of assistance withdrawal, ensuring a smooth transition of control from the spring-damper systems to the agent. The comparison of decay functions behavior, which are used in our study, is showed at the Figure 3.

In general, decay function could be any function, which will reduce the control values (stiffness and damping coefficients) via some mathematical law.

Table 3: Decay types and their descriptions

Curriculum Learning parameters	**Description**
Control values	control parameters representing spring-damper values, which influence the restriction of the system behavior
Decay steps	time steps when the transition to another set of control values occurs
Decay function	describes the law on how the control values will be changed
Decay factor	sets the speed of decay function

3 Results

In this section we examine the performance of the state-of-the-art RL algorithm PPO for a discrete and continuous action spaces and show how can CL influence the overall learning task. Most of the results for comparison of the discrete action space are taken from our previous paper [9]. First subsection presents the results achieved while comparing the trained agent with the same initial parameters using continuous and discrete control schemes. Second subsection describes the work of an agent in a modified environment, where the physical parameters of the system are changed. In the last subsection we present the role of CL in the enhancement of the overall RL agent training.

3.1 Reinforcement Learning: discrete vs continuous action space

To make a comparative analysis we present figures, which illustrate the training performance of an agent via RL-training using in both discrete and continuous action spaces across three different systems: a single-link pendulum (Figure 4 (a)), a two-link pendulum (Figure 4 (b)), and a three-link pendulum (Figure 4 (c)) The discrete action space data, represented by the blue curves, are derived from our previous study of Manzl et. al. [9], while the continuous action space results, represented by the orange curves, are from the current analysis. The vertical dashed lines in each plot indicate the training step at which the agent reaches the maximum number of evaluation tests. For the single-link system, the agent trained using a continuous action space PPO algorithm achieves optimal performance much faster than the agent trained having the same PPO algorithm but discrete action space. The continuous action space agent stabilizes around 23000 steps, while the discrete agent takes approximately 53,000 steps to achieve a similar level of performance. This indicates that the continuous action space allows for finer control and quicker con-

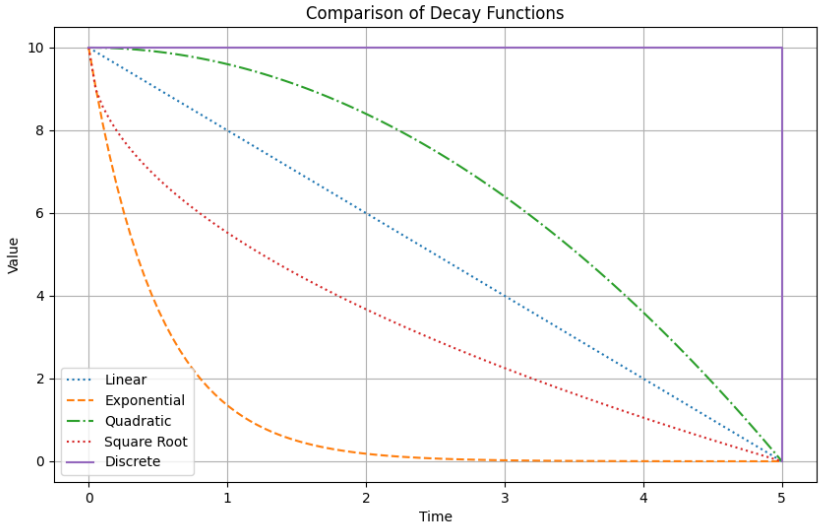
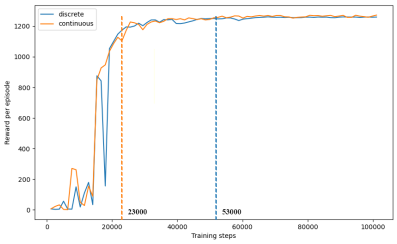


Figure 3: 5 decay functions comparison for a given set of control values [10, 0]. At the decay step equal to 5 all the functions take the next control value of 0.

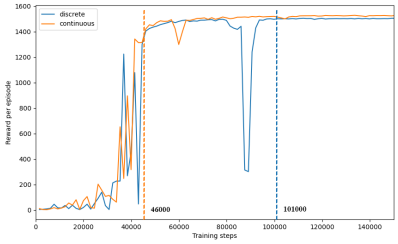
vergence in simpler systems. The two-link system reveals an even more pronounced difference. The continuous action space agent reaches stable performance at around 46000 steps, significantly ahead of the discrete action space agent, which stabilizes much later at around 101,000 steps. The continuous agent not only converges faster but also exhibits higher stability in maintaining optimal rewards throughout training. In the more complex three-link system, the advantage of the continuous action space is evident once again. The continuous agent reaches optimal performance at around 270000 steps, whereas the discrete agent takes over 500,000 steps to achieve comparable results. Notably, the discrete agent experiences significant fluctuations in performance, underscoring the challenges posed by the increased complexity when using a discrete action space.

The continuous action space consistently demonstrates superior stability across all systems, as reflected in the smoother reward curves after convergence. It also reduces training time on up to 50% for an agent to achieve a maximum number of successful tests. In contrast, the discrete action space leads to more oscillatory performance, particularly in more complex systems, which could indicate less efficient exploration and exploitation during training.

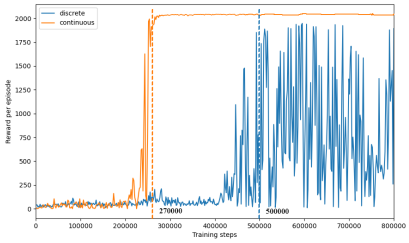
For a deeper evaluation of the continuous control scheme in comparison to the discrete one, we have created a stability zone, the development of which is described



(a) 1-link system. Agent in continuous action space reaches maximum number of successful tests in approximately of 23000 timesteps.



(b) 2-link system. At the timesteps of 46000 the agent has reached the maximum number of successful tests.



(c) 3-link system. Less then 300000 timesteps required to train the system with the usage of continuous action space.

Figure 4: Training times for the PPO agents for 1-link (a), 2-link (b) and 3-link (c) systems

in our work of Manzl et al. [9]. From an engineering standpoint, not only are the randomized tests used for evaluation important, but it is also crucial to identify an area where the agent successfully performs the stabilization task for practical, real-world applications. The stability zones are shown in Figure 5 and illustrate the performance of discrete (orange) and continuous (light yellow) action spaces. These zones represent the regions in state space where the agent successfully maintains the pendulum's stability. The axes represent different state variables depending on the system's complexity, as detailed below. For the 1-link system (Figure 5 (a)) the stability zones are depicted as elongated regions around the origin where the pendulum is successfully balanced. X-axis represent the link's angle and Y-axis its

angular velocity. The continuous action space encompasses a slightly broader area compared to the discrete action space, indicating a greater tolerance for variations in the angle and angular velocity. This suggests that the continuous action space provides more flexibility in controlling the pendulum, allowing it to stabilize from a wider range of initial conditions. For the 2-link system (Figure 5 (b)), the stability zones are narrower and more elongated, reflecting the increased complexity of the system. First and the second link angle combinations are presented in this zone. The continuous action space again covers a slightly larger area than the discrete action space, particularly along the axis. This expanded zone suggests that the continuous action space better handles the interactions between the two links, allowing for more robust stabilization strategies. In the 3-link system (Figure 5 (c) and Figure 5 (d)), the stability zones become even more restricted, particularly in the second plot, where the angles of the second and third links are plotted. The continuous action space consistently outperforms the discrete space by maintaining a larger stability region in both plots. The narrower orange zones indicate that the discrete action space struggles with the additional degrees of freedom, whereas the continuous action space adapts more effectively to the system's increased complexity.

These stability zones highlight the advantages of continuous action spaces in maintaining the stability of multi-link inverted pendulum systems. As the system's complexity increases (from 1-link to 3-link), the continuous action space's broader stability regions suggest a higher robustness to variations in state variables, making it more effective in stabilizing complex systems. The continuous action space's ability to control a wider range of initial conditions is particularly beneficial in systems with multiple degrees of freedom, where precise and adaptive control strategies are required. This is evident in the progressively more complex systems, where the continuous action space maintains larger stability zones, indicating better overall performance. These findings support the notion that continuous action spaces may be more suitable for applications requiring fine-tuned control in high-dimensional environments, where the ability to handle a broader range of state variations is crucial for success.

3.2 Agents with continuous action space tested on modified environments

In this section, we examine the impact of changing properties of the environment, namely link length, mass, and added friction, on the stability zones. The friction is implemented in the same fashion as in our previous study [9] and is modeled using Coulomb friction, characterized by a friction torque, which opposes rotation and is independent of the applied forces. The investigation is conducted using the inverted double pendulum on a cart model (same continuous agent as in the Figure 5 (b)). In each subplot of Figure 6, the stability zone is determined for the same agent in various modified environments, and the contour of the stability zone from the

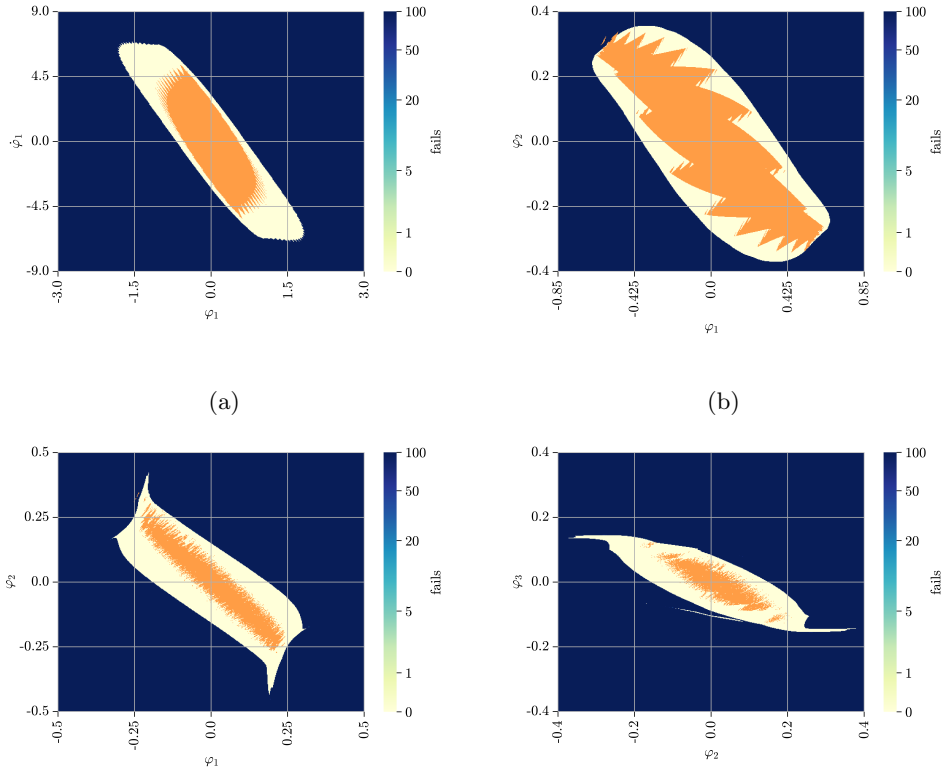


Figure 5: Stability zones comparison of the PPO agent using different control strategies. For the 1-link system (a) the zone axis are link's angle ϕ_1 and angular velocity $\dot{\phi}_1$, while for the 2-link system (b) axis are the pendulum link angles ϕ_1 and ϕ_2. Figures (b) and (c) present the stability zones based on the dependence of pendulum link angles of ϕ_1 and ϕ_2 and of ϕ_2 and ϕ_3. The discrete control stability zone is indicated in orange; continuous is in white beige. Each grid cell represents 100 randomized tests.

original environment is overlaid.

The observations shows, that increasing the link length will cause the stability zone to expand (see Figures 6 (a) and (b)), while not having the increase of the blind spots, as it was for the discrete control cases [9]. Increasing the mass of the link doesn't provide any differences from the base model (Figures 6 (c) and

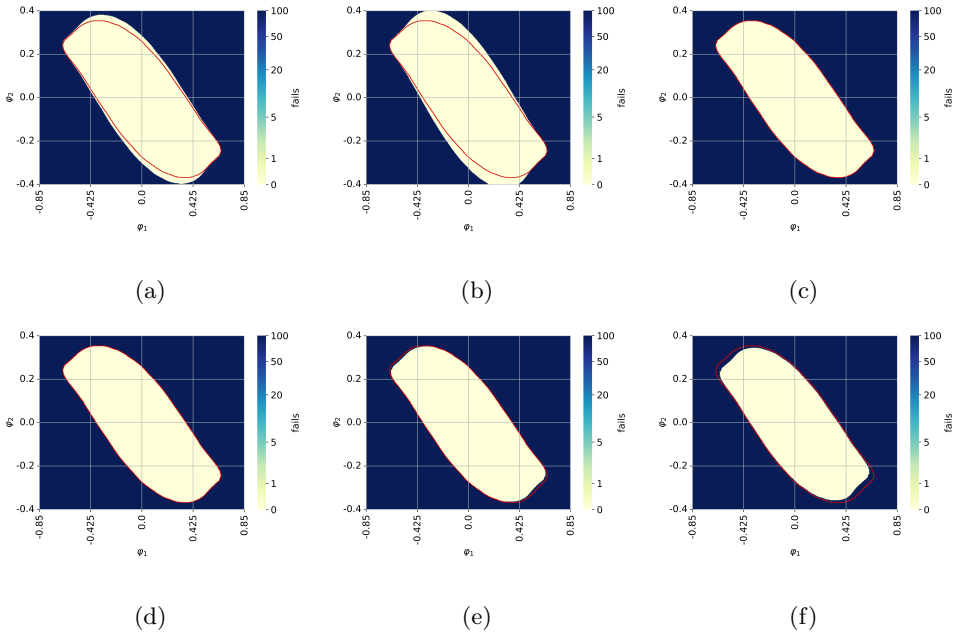


Figure 6: The stability zone for the double link system using the PPO agent, case 0. The red boundary corresponds to that depicted continuous control stability zone in Figure 5 (b). The environment parameters are modified as follows: (a) 1.1 l, (b) 1.2 l, (c) 1.1 m, (d) 1.2 m, (e) $f_{rel} = 0.01$, and (f) $f_{rel} = 0.02$.

(d)), so that we can state that up to the changes of 20% the model is independent from the link mass change, showing the same behavior as for the discrete model, where the changing of mass was also neglectable on the stability zones. Considering friction the stability zone becomes smaller within each increase of it, but the value of $f_{\text{rel}} = 0.02$ still provides us with the stable zone without causing the agent to fail in the original contour (Figures 6 (e) and (f)), as it was shown in [9]. It can be concluded that the continuous control scheme is more suitable if friction is involved in the environment model simulation.

3.3 RL training enhancement with CL

a) setting up a baseline

Before proceeding with the analysis, it is essential to establish baseline results for the continuous RL PPO method, without enhancements from Curriculum Learning. This provides a point of reference to assess how much CL can improve performance. For this we have conducted tests using the system parameters presented in the

Table 1 and 2 for systems with 1 to 3 links. For each system, 10 distinct simulation cases were executed to account for variability in control due to different initial conditions. The baseline is determined by measuring the training timesteps required for success, focusing primarily on the mean and median values, which serve as the key performance indicators. Additionally, the minimum and maximum values provide insight into the range of results across cases. These metrics allow us to assess the general performance of PPO in continuous action space. Table 4 summarizes the results for the RL training using the PPO algorithm in continuous action space. The results are presented for systems with 1, 2, and 3 links.

Table 4: Baseline results of using PPO algorithm in continuous action space for 1 to 3 link pendulum systems

System	**Successful cases**	**Mean**	**Median**	**Minimum**	**Maximum**
1-link	10	26168	25600	24320	32000
2-link	7	48055	49152	38400	55296
3-link	5	278937	288768	194560	319488

As indicated in Table 4, the results show a clear progression in training time as the system complexity increases from 1-link to 3-link. The 1-link system requires the least number of timesteps on average to reach successful training, while the 3-link system exhibits significantly higher values, with a mean training time nearly 10 times longer than the 1-link system. This demonstrates that as the system complexity increases, the training duration grows substantially, making it harder for the PPO algorithm to converge quickly. The primary goal of introducing CL enhancement is to accelerate the training process and potentially make the system more robust to various initial conditions. By progressively increasing the difficulty of the tasks during training, we aim to reduce the number of timesteps required for convergence. To further explore the impact of CL, the next step focuses on selecting the appropriate CL implementation parameters, dividing the research into well-defined steps.

b) selection of a decay type (function)

At first, we shall determine which decay type is specifically the most performing for our particular task of pendulum stabilization. For this we have tested all described decay functions in subsection 2.5 on a single pendulum system. With 150 runs per decay function of 5 initialization cases we use control values matrix of the form $\begin{bmatrix} 1 & n \\ 0 & 0 \end{bmatrix}$, where $n = 2, 4, 6, 8, 10$ and decay steps vector $\begin{bmatrix} 0 \\ k \end{bmatrix}$, where $k = 5000, 6000, \ldots, 10000$ to evaluate the performance across different control parameters. The results are presented in a form of a bar plot, which shows the number of agent successes for the whole dataset (see Figure 7).

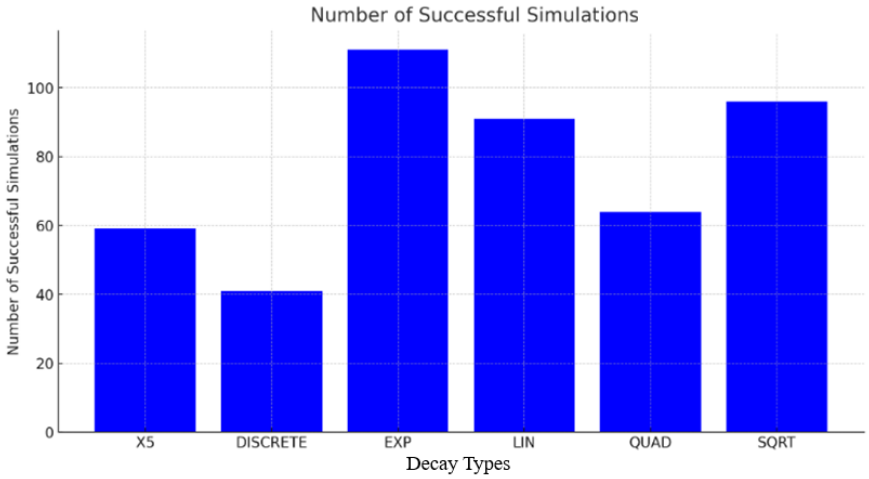


Figure 7: Simulation results for decay types analysis. Each bar plot represents the value of successfully trained cases out of 150 in total per decay function.

The exponential decay function, provides the best results of having 74% of successes across the dataset and therefore will be considered as the main function used in our next analysis.

c) selection of decay factor

For the selection of the decay factor, we performed an extensive analysis using a double pendulum system. The control values matrix was set as $\begin{bmatrix} 1 & n & n \\ 0 & 0 & 0 \end{bmatrix}$, where $n = 1, 2, \ldots, 10$. The decay steps, which represent how long the curriculum learning is conducted, were varied from 9000 to 22000 with increments of 1000. For each unique combination of control values, decay steps, and decay factors, we executed 10 cases to ensure the reliability and robustness of our results. In total, we analyzed 4 decay factors: 0.005, 0.01, 0.05, and 0.5, generating a dataset with approximately 70,000 lines of results. The objective of this analysis was to determine the optimal decay factor for maximizing the system's learning performance. The output metric used for comparison included the value of training timesteps. This metric provide insight into how efficiently the system is able to solve tasks within the corresponding timesteps.

As shown in Figure 8 decay factor 0.005 consistently results in the lowest timesteps across all combinations of control values and decay steps. This indicates that the system converges faster and more efficiently with a decay factor of 0.005 compared to the other decay factors. The smaller the timesteps, the faster the system is able to learn, making 0.005 the most optimal choice for our training

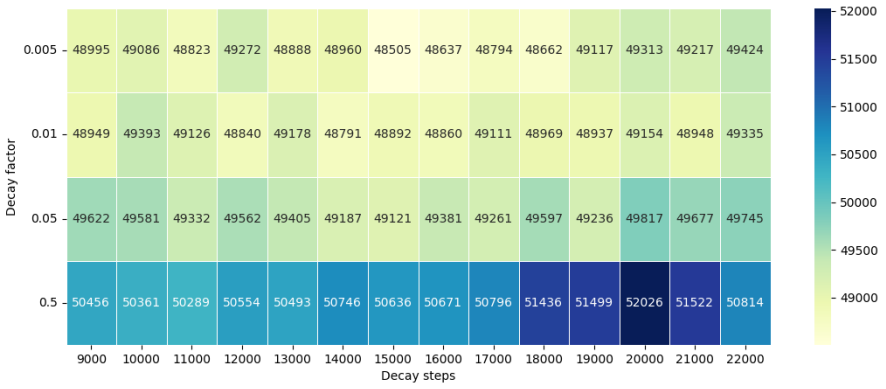


Figure 8: Heatmap showing the impact of different decay factors on the number of successful tests and timesteps. The decay factor 0.005 consistently performs best in terms of minimizing timesteps.

setup.

d) applying CL enhancement

The next step in our analysis involves the development of a control value decay scheme to assess its influence while varying decay steps. For the single pendulum system, the control values matrix is maintained as $\begin{bmatrix} 1 & n \\ 0 & 0 \end{bmatrix}$, since the simplicity of the system allows for efficient training without requiring significant constraints on movement. Experiments were conducted by varying n from 0.1 to 1 (step size of 0.1), and from 1 to 12 (step size of 1). Decay steps ranged from 1000 to 12000, with increments of 1000. For the double pendulum system, the control scheme was set as $\begin{bmatrix} 1 & c \cdot n & r \cdot n \\ 0 & 0 & 0 \end{bmatrix}$, where c and r represent extension coefficients used to observe the effects of pendulum link constraint magnitudes on the results and are varied over the set $1, 2, 4, 8$, while $n = 1, 2, \ldots, 10$. Decay steps were varied between 9000 and 22000, with an increment of 1000. For the triple link system, the control scheme used was $\begin{bmatrix} 1 & 4n & 2n & n \\ 0 & 0 & 0 & 0 \end{bmatrix}$, where $n = 1, 2, \ldots, 10$. This configuration required stronger restrictions on the first link due to its responsibility in bearing the mass of the subsequent links. The decay steps were varied between 30000 and 50000, with increments of 1000. The evaluation of results was performed across cases within each unique combination of control values and decay steps, rather than across the entire control scheme. This is due to the CL implementation, which focuses on enhancing training efficiency where the key factor is to find the optimal combination of values

to facilitate faster and more robust learning in the system. To determine the best-performing combination of control values and decay steps, we concentrated on three primary factors: number of successful cases in each combination, mean and median values of training time across cases. For the single and double pendulum systems, a successful case was defined as achieving 50 out of 50 successful tests. For the triple pendulum system, due to its higher complexity, the threshold for success was set at 70 out of 100 tests passed. This adjustment reflects the increased difficulty of achieving full success for the triple link system. The results are shown in the Table 5.

Table 5: CL enhancement: best results for 1 to 3 link pendulum systems

System	**Best performing schemes**	**Successful cases**	**Decay steps**	**Mean**	**Median**
1-link	$\begin{bmatrix} 1 & 0.8 \\ 0 & 0 \end{bmatrix}$	10	7000	17817	18432
2-link	$\begin{bmatrix} 1 & 20 & 5 \\ 0 & 0 & 0 \end{bmatrix}$	10	15000	43929	43008
3-link	$\begin{bmatrix} 1 & 16 & 8 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	10	30000	233676	216064

The results in Table 5 demonstrate the effectiveness of the best-performing control schemes across the three systems, with all systems achieving 10 out of 10 successful cases. These results highlight the robustness of the implemented Curriculum Learning in reducing training times while maintaining high success rates, even as system complexity increases. To gain further insights into how the variations in control schemes and decay steps influenced the training times, we present a series of heatmaps showed on Figure 9 (a) to (f). The percentage change in both mean and median training times for each pendulum system is visualized, offering a clear depiction of the impact of CL enhancements across different cases.

The heatmaps clearly demonstrate that, across all systems, the implementation of CL resulted in a substantial improvement in training speed. The 1-link system showed the greatest consistency, while the 2-link and 3-link systems demonstrated significant improvements in reduced training time.

The results in Table 6 indicate that the Curriculum Learning (CL) enhancement has proven effective in reducing training times and improving the robustness of reinforcement learning across all systems. While the 1-link system showed the greatest improvement, with a 32% reduction in mean training time, the double and triple pendulum systems also exhibited substantial improvements, confirming the scalability of CL to more complex multi-link systems. This success demonstrates that CL enables the RL agent to better navigate the increased complexity of the

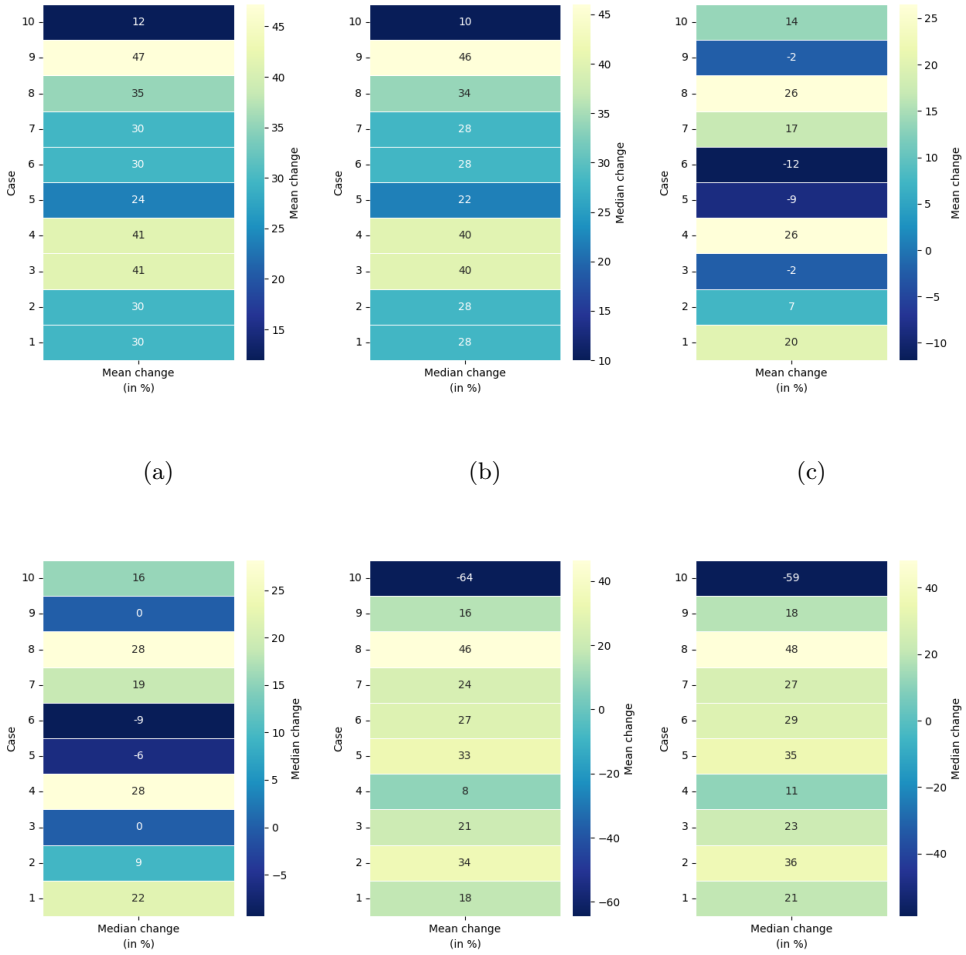


Figure 9: Mean and median training time change per cases for 1- to 3- link pendulum systems using CL enhancement.

tasks, allowing it to converge more efficiently. One of the most important findings is that all systems, including the 2-link and 3-link systems, reached 10 out of 10 successful cases. This confirms that the control schemes used, combined with CL,

Table 6: Improvement statistics by enhancing RL training of the 1 to 3 link pendulum systems with CL

System	**Mean improvement**	**Median improvement**
1-link	32%	28%
2-link	8.6%	12.5%
3-link	19%	25.2%

were able to handle even the more complex tasks. By gradually increasing the difficulty of the tasks during training, CL helps the reinforcement learning agent learn more effectively, avoiding large jumps in difficulty that could slow down training or cause failure. The heatmaps provided earlier also showed consistent improvement in training time across different control values and decay steps, meaning that the benefits of CL are not limited to specific parameters. This makes CL a powerful and flexible tool for improving reinforcement learning across a wide range of scenarios. However, it's important to recognize that finding the best parameters (such as control values and decay steps) is still key to maximizing the benefits of CL. While our results show significant improvements, further fine-tuning could make the training even faster, especially in more complicated systems. Future research could focus on dynamic approaches where the curriculum adjusts based on the agent's performance during training, potentially leading to even better results. In summary, Curriculum Learning has proven to be an effective way to speed up training and improve robustness in all systems tested. From the simpler 1-link pendulum to the more challenging 3-link system, CL has provided consistent improvements, making it a valuable strategy in reinforcement learning. These findings show the importance of a well-planned training process and suggest future possibilities for even greater gains with more advanced strategies.

4 Conclusions

References

[1] Yoshua Bengio et al. "Curriculum learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), pp. 41–48.

[2] Rupali Bhati et al. "Curriculum Learning for Cooperation in Multi-Agent Reinforcement Learning". In: *37th Conference on Neural Information Processing Systems (NeurIPS)* (2023). URL: `https://ar5iv.org/html/2312.11768`.

[3] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. "Challenges of real-world reinforcement learning". In: *arXiv* (2019).

[4] Pascal Egli et al. "Soil-Adaptive Excavation Using Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 1–8. DOI: 10.1109/LRA.2022.3189834.

[5] I. Fantoni and R. Lozano. "Non-linear Control for Underactuated Mechanical Systems". In: (2001). DOI: 10.1115/1.1483350.

[6] Kashish Gupta, Debasmita Mukherjee, and Homayoun Najjaran. "Extending the Capabilities of Reinforcement Learning Through Curriculum: A Review of Methods and Applications". In: *SN Computer Science* 3.28 (2021).

[7] Guy Hacohen and Daphna Weinshall. "On the power of curriculum learning in training deep networks". In: *Proceedings of the 36th International Conference on Machine Learning* (2019), pp. 2535–2544.

[8] Ilya Kurinov et al. "Automated Excavator Based on Reinforcement Learning and Multibody System Dynamics". In: *IEEE Access* 8 (Jan. 2020), pp. 213998–214006. DOI: 10.1109/ACCESS.2020.3040246.

[9] Peter Manzl et al. "Reliability evaluation of reinforcement learning methods for mechanical systems with increasing complexity". In: *Multibody System Dynamics* (2023).

[10] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *arXiv preprint arXiv:1602.01783* (2016).

[11] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[12] Sanmit Narvekar et al. "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey". In: *Journal of Machine Learning Research* 21.181 (2020), pp. 1–50.

[13] Takayuki Osa and Masanori Aizawa. "Deep Reinforcement Learning with Adversarial Training for Automated Excavation using Depth Images". In: *IEEE Access* 10 (2022), pp. 4523–4535. DOI: 10.1109/ACCESS.2022.3140781.

[14] Loris Roveda et al. "Model-Based Reinforcement Learning Variable Impedance Control for Human-Robot Collaboration". In: *Journal of Intelligent and Robotic Systems* (Nov. 2020).

[15] John Schulman et al. "Proximal policy optimization algorithms". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org. 2017, pp. 3191–3199.

[16] Max Schwenzer et al. "Review on model predictive control: an engineering perspective". In: *The International Journal of Advanced Manufacturing Technology* 117 (2021), pp. 1327–1349.

[17] Ahmed A. Shabana. *Dynamics of Multibody Systems*. 4th ed. Cambridge University Press, 2013.

[18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. English. second edition. Cambridge, Massachusetts: A Bradford Book, 2018. ISBN: 978-0-262-03924-6.

[19] Daphna Weinshall, Gad Cohen, and Dan Amir. "Curriculum learning by transfer learning: theory and experiments with deep networks". In: *Proceedings of the 35th International Conference on Machine Learning* (2018).

[20] Shenghan Zhu et al. "An optimization method for the inverted pendulum problem based on deep reinforcement learning". In: *Journal of Physics: Conference Series* 2296.1 (2022), p. 012008.