Multibody System Dynamics
https://doi.org/10.1007/s11044-023-09960-2

RESEARCH

Reliability evaluation of reinforcement learning methods for mechanical systems with increasing complexity

Peter Manzl[1] · Oleg Rogov[2] · Johannes Gerstmayr[1] · Aki Mikkola[2] · Grzegorz Orzechowski[2]

Received: 15 June 2023 / Accepted: 6 December 2023

Abstract
Reinforcement learning (RL) is one of the emerging fields of artificial intelligence (AI) intended for designing agents that take actions in the physical environment. RL has many vital applications, including robotics and autonomous vehicles. The key characteristic of RL is its ability to learn from experience without requiring direct programming or supervision. To learn, an agent interacts with an environment by acting and observing the resulting states and rewards. In most practical applications, an environment is implemented as a virtual system due to cost, time, and safety concerns. Simultaneously, multibody system dynamics (MSD) is a framework for efficiently and systematically developing virtual systems of arbitrary complexity. MSD is commonly used to create virtual models of robots, vehicles, machinery, and humans. The features of RL and MSD make them perfect companions in building sophisticated, automated, and autonomous mechatronic systems. The research demonstrates the use of RL in controlling multibody systems. While AI methods are used to solve some of the most challenging tasks in engineering, their proper understanding and implementation are demanding. Therefore, we introduce and detail three commonly used RL algorithms to control the inverted N-pendulum on the cart. Single-, double-, and triple-pendulum configurations are investigated, showing the capability of RL methods to handle increasingly complex dynamical systems. We show 2D state space zones where the agent succeeds or fails the stabilization. Despite passing randomized tests during training, blind spots may occur where the agent's policy fails. Results confirm that RL is a versatile, although complex, control engineering approach.

Keywords Reinforcement learning · Reliability analysis · Inverse pendulum · Machine learning · Dynamical systems

1 Introduction

Multibody system dynamics (MSD) is a research field concerned with the modeling, simulation, application, and analysis of systems comprised of interconnected rigid and flexible bodies, which undergo large translational and rotational displacements. Systems in the field of MSD range from mechanisms as pendulums to machines, robots, and vehicles [1].

Although MSD provides an effective means of systematically simulating complex systems, it requires appropriate control algorithms to act as desired. Reinforcement learning

Extended author information available on the last page of the article

(RL) offers an innovative approach to control multibody models. RL is a type of machine learning where an agent learns to make decisions by interacting with an environment to maximize a cumulative reward signal [2]. It has been successfully applied to various control applications and has the potential to learn complex behaviors in a data-driven manner without the need for explicit programming, making it a powerful tool for developing intelligent systems. The use of deep neural networks as function approximators in RL changes its notion to deep reinforcement learning (DRL)—it tackles the problem of efficiency while working with large state spaces [3]. RL and its subfield DRL have recently gained popularity and proven capable of solving complex control tasks, which MSD offers in a huge variety.

A paper by Hashemi et al. [4] provides an overview of how machine learning might be applicable to multibody system dynamics. Specifically talking about RL, the authors show that its use has recently been suggested for a wide variety of tasks: starting from improving feedback control systems and robotic applications and ending with assistance in simulating human movements. The work by Benatti et al. [5] shows how DRL and MSD could be combined in a single Python-based framework called Gym Chrono. Gym Chrono is based on PyChrono and features MSD simulation with constraints, contact interaction, and deformable bodies through finite element analysis. The authors demonstrate the effectiveness of this approach through several case studies, including the control of a hexapod and a hallway cone track, as well as autonomous driving in off-road conditions. The paper highlights the potential of using MSD and RL to develop advanced control algorithms for various complex systems.

From a robotics perspective, the paper by Sun et al. [6] proposes a fully autonomous RL approach for mobile manipulation tasks in a real-world setting. The authors use a combination of a physics-based simulator and a real robot to train an RL agent to perform tasks such as object grasping and manipulation. The study demonstrates the ability of the RL agent to generalize to different object shapes and sizes and to adapt to changing environmental conditions. Another interesting application of RL is the automatic controller weights adjustment since the adjustment is usually done manually and is often time-consuming [7]. This work presents an implementation of RL for tuning the impedance control of a robotic prosthesis to provide personalized gait assistance. The authors use an MSD model to simulate the behavior of the prosthesis and the human leg during walking. RL is employed to optimize the impedance control parameters of the prosthesis in real-time based on the user's walking behavior. The study highlights the potential of RL in controlling complex systems such as robotic prostheses and suggests that it can lead to more personalized and effective gait assistance. Song et al. [8] prepared an article that provides an overview of neuromechanical simulations and DRL in human locomotion. Neuromechanical simulations have proven helpful in evaluating control models, while DRL shows promise in developing control models for complex movements. While DRL has successfully produced coordinated body motions in physics-based simulations, there are still challenges in producing more complex motions that involve long-term planning and learning physiologically plausible models.

It is worth noting the use of RL for general control applications. A paper by Busoniu et al. [9] provides a comprehensive overview of RL in control systems. One key conclusion of the paper is that RL can effectively control complex systems, including those described by MSD models. However, the success of RL in control systems depends on several factors, such as the choice of the algorithm, the quality of the system model, and the stability of the resulting control policy. Additionally, using deep neural networks can further improve learning performance, but it also introduces new challenges related to stability and convergence. In the end, it can be stated that using RL in controlling multibody systems provides a promising avenue for solving complex control problems where traditional control techniques may be ineffective or difficult to implement.

The objective of this research paper is to demonstrate the use of RL in the control of a complex multibody system. To this end, the RL is applied to the problem of controlling N-link inverted pendulums on a cart. Control of an inverted pendulum on a cart is a classical control task extensively studied in the literature [10, 11]. However, controlling multiple inverted pendulums presents a more challenging task since every link of a pendulum is unstable by itself. Adding more links dramatically increases the difficulty of the stabilization task and leads to increased computation complexity. This work examines three systems of inverted pendulums on a cart: a single link, double link, and triple link system.

To solve considered control problems, three RL algorithms are employed: Advantage Actor-Critic (A2C) [12], Proximal Policy Optimization (PPO) [13], and Deep Q-Network (DQN) [3]. Those algorithms are selected due to their maturity, robustness, and possibility of handling discrete control tasks since our control input for a system under study consists of pushing the cart to the left or to the right. Many other algorithms are used for continuous control, where the control action space is infinite, such as Soft Actor-Critic (SAC) [14] or Twin Delayed Deep Deterministic policy gradient (TD3) [15]. A2C and PPO are actor-critic methods that use a neural network to predict both the policy and the state-value function. In contrast, DQN uses a neural network to approximate the Q-value function, representing the action quality in a given state. These algorithms performed well in various RL applications and are widely used in RL research [16–18]. The reliability of the selected RL algorithms is analyzed in handling increasingly complex multibody systems. Performance evaluation of these algorithms is based on their ability to stabilize the pendulum in an upright position—the rate of convergence and the robustness of the control policy under varying environmental conditions, such as the magnitude of applied control force and the presence of friction. The results of this study can provide valuable insights into the applicability of different RL algorithms for controlling multibody systems and pave the way for future research in this area.

2 Multibody system dynamics

In control and state observer applications, it is advantageous to express the equations of motion based on the multibody system dynamics using a minimal set of coordinates. In practice, this can be accomplished in several ways, such as coordinate partitioning when using global coordinates [1, 19]. Alternatively, methods based on the use of relative joint coordinates can be employed [20]. An often-used approach based on the relative coordinates is the family of semi-recursive multibody formulations. In semi-recursive multibody formulations, the equations of motion for an open-loop system [21] with N_b bodies can be written as

$$\mathbf{R}_\mathrm{d}^\mathrm{T}\mathbf{T}^\mathrm{T}\bar{\mathbf{M}}\mathbf{T}\mathbf{R}_\mathrm{d}\ddot{\mathbf{z}} = \mathbf{R}_\mathrm{d}^\mathrm{T}\mathbf{T}^\mathrm{T}\left(\bar{\mathbf{Q}} - \bar{\mathbf{M}}\mathbf{T}\dot{\mathbf{R}}_\mathrm{d}\dot{\mathbf{z}}\right) \quad \Rightarrow \quad \bar{\mathbf{M}}^\Sigma\ddot{\mathbf{z}} = \bar{\mathbf{Q}}^\Sigma, \tag{1}$$

where $\mathbf{z}$, $\dot{\mathbf{z}}$, and $\ddot{\mathbf{z}}$ are the relative joint coordinates, velocities, and accelerations, respectively [22], $\mathbf{R}_\mathrm{d}$ is a block diagonal matrix that describes joints. The $\mathbf{R}_\mathrm{d}$ can be expressed in a block diagonal form when making use of an intermediate body frame whose origin coincides with the origin of the global frame. In this approach, $\mathbf{T}$ is the path matrix representing the system topology. In Eq. (1), $\bar{\mathbf{M}}$ is the mass matrix and $\bar{\mathbf{Q}}$ is the external force vector. Note that

$$\bar{\mathbf{M}}^\Sigma = \left(\mathbf{R}_\mathrm{d}^\mathrm{T}\mathbf{T}^\mathrm{T}\bar{\mathbf{M}}\mathbf{T}\mathbf{R}_\mathrm{d}\right) \tag{2}$$

and

$$\bar{\mathbf{Q}}^{\Sigma} = \left[\mathbf{R}_{\mathrm{d}}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\left(\bar{\mathbf{Q}} - \bar{\mathbf{M}}\mathbf{T}\dot{\mathbf{R}}_{\mathrm{d}}\dot{\mathbf{z}}\right)\right]. \tag{3}$$

For a closed-loop system, a cut-joint technique needs to be used, and a set of N_m loop-closure constraints, $\mathbf{\Phi} = \mathbf{0}$, can be incorporated into the open-loop dynamics. Using the coordinate partitioning method [23], the relative joint velocities are mapped onto a set of independent relative joint velocities leading to the minimal set of coordinates.

3 Reinforcement learning in multibody system dynamics

Multibody system dynamics is a field of engineering that deals with the simulation and analysis of the motion and interaction of interconnected rigid or flexible bodies subjected to various external and internal forces [1]. MSD has been used extensively in designing and analyzing mechanical systems in various domains, such as automotive, aerospace, and robotics engineering. In many cases, simple control approaches, such as PID controllers, have been used to regulate the behavior of mechanical systems. However, these simple control approaches may not always be optimal, especially when dealing with complex and nonlinear systems.

To address these challenges, researchers have started exploring advanced control techniques that could be applied to MSD. One approach is to use optimal control, such as model predictive control (MPC), to optimize a control policy based on a model of the system dynamics [24]. Another approach is to use adaptive control, such as sliding mode control (SMC), to adapt the control policy based on the system response [25]. Despite the success of these advanced control techniques, they still have limitations. Optimal control requires an accurate model of the system dynamics, which can be difficult to obtain in practice. Adaptive control relies on tuning control parameters, which can be time-consuming and challenging to perform in real-time. To address those challenges, research on using RL is conducted [4].

3.1 Reinforcement learning and the classical control engineering

Classical control methods are preferable in linear dynamical systems due to well-established methods and provable stability [26]. In the case of nonlinear systems, which usually applies to multibody systems, classical control approaches are widely utilized; however, stability proofs are involved. Often, linearization or other simplifications are made, such as neglecting friction or compliance of the systems [26]. In the latter case, experimental validation, as well as the experience of control engineers, is used to achieve the necessary robustness. Closed-loop feedback control is applied to a (multibody) system to obtain a desired system response. The controller compares the actual system output (feedback signal) with the desired output and uses the difference as a means of control. This difference—the error—is reduced by the controller, *e.g.*, amplified and used for the system's actuation to achieve the desired output. In addition to basic error control, the closed-loop system should also exhibit specified behavior such as effective disturbance attenuation and rapid responsiveness to changes in operating conditions [27]. It is also important to point out that the implementation of classical control algorithms requires in-depth knowledge of the theory and practice to create a proper controller.

With RL, controllers can be developed based on mainly training neural networks using experiments with real or simulated environments. A fundamental concept is the role of an

Table 1 Reinforcement learning and control theory correspondence. Note that the agent can only act as a controller after training, although from the perspective of system architecture, the agent generally takes the controller's place

Reinforcement learning	Control theory
(trained) Agent	Controller
Environment	System / Plant
State	System state
Action	Control signal
Policy	Control law

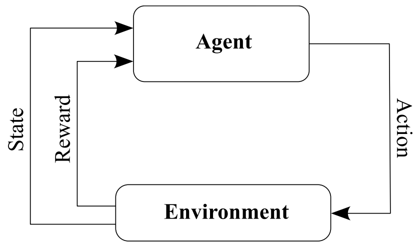


Fig. 1 Reinforcement learning scheme. An Agent takes an action in the environment and the environment outputs the state and reward to the Agent, which updates itself based on these parameters to select a better action to take

agent, a specialized entity that perceives its environment through observations, compares the state with the expected state, and accordingly executes actions to optimize its cumulative reward. The realization is alleviated due to the broad availability of RL libraries, such as Stable-Baselines3 [28] or TF-Agents [29]. RL methods circumvent the necessity of a model for the system's (nonlinear) dynamics. The controller parameters, which are the controller's neural network parameters, are determined through trial-and-error interactions with the environment, using the RL method's optimizer to find optimal parameters to minimize the loss and maximize the reward. However, inherent complexities and nonlinearities in neural networks, *e.g.*, with Rectified Linear Units (ReLUs), may generate certain (undetected) zones of a controller failure, similar to 'blind spots' [30]. This study consequently aims to illustrate these blind spots through increasingly complex environments.

The RL terminology differs significantly from the common terms used in control engineering. Therefore, relations are provided in the literature [2], see also Table 1. Other standard terms used in the RL, like the reward and value functions, are relevant to the controller creation (learning) phase and thus cannot be directly related to classical control theory concepts. In short, the reward sets the goal, and the agent's objective is to maximize the *total* reward it receives. Starting from a given state, the value is the total *expected* reward an agent can obtain in the future [2]. Value functions are defined with respect to policies, which are particular ways of acting.

It is important to notice that a state in RL is a complete specification of the environment that includes all relevant information that is necessary to make decisions about what action to take next, while observation is a subset of the state that is actually observed by the agent. After an action has been taken, an agent receives feedback in the form of rewards or penalties and achieves a new state [2]. The RL interaction scheme is shown in Fig. 1.

In the area of policy-based methods, the primary focus lies in directly learning the policy function, a mapping from states to the most appropriate corresponding actions. The objective is to discover the optimal policy, which maximizes the expected return when adhered to by the agent [31]. The training process in policy-based methods entails adjusting the policy parameters based on observed rewards and chosen actions, aiming to enhance its performance [32].

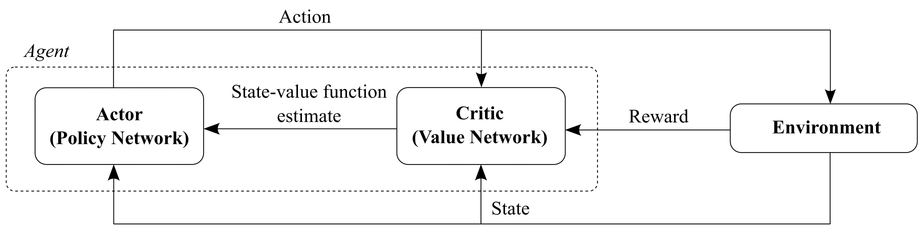


Fig. 2 A2C scheme. An actor takes action in the environment. After that, the state and reward are returned from the environment. Critic provides an actor with the state-value function estimate, and the actor updates its weights and takes a, hopefully, better action. The critic also updates itself at the same time step as the actor

In the domain of value-based methods, instead of directly acquiring the policy, the agent learns the value function, assigning a value to each state or state-action pair [31]. A value is an expected return an agent can obtain from a given state or state-action pair under a particular policy [2]. Using the value function, the agent makes decisions by opting for actions leading to states with higher expected values. Training in value-based methods involves iteratively estimating these values through methods such as temporal difference learning [15] or Q-learning [33].

In the study, we use three deep RL methods for solving our control task: Advantage Actor-Critic (combines policy-based and value-based approach), Proximal Policy Optimization (policy-based method), and Deep Q-Network (value-based method).

3.2 Advantage Actor-Critic (A2C)

The A2C method is a policy optimization algorithm based on policy gradients [12]. The scheme of the algorithm is shown in Fig. 2. The actor in A2C is the neural network to determine the taken policy. It takes the current state of the environment as an input, and then outputs a probability distribution over possible actions (policy). The critic, also typically modeled using a neural network, takes a state as an input and produces the estimate of a state-value function, representing the estimated future return from this state. The critic's objective is to minimize the discrepancy between its estimated values and the actual observed rewards. This discrepancy is quantified using a loss function, which is typically the mean squared error (MSE) between the estimated values and the discounted sum of obtained rewards. With the MSE, the critic network then updates its weights via stochastic gradient descent. On the other hand, the actor network utilizes the estimated state-value function provided by the critic in calculating the advantage function. The advantage tells us about the extra reward that could be obtained by the agent by taking a particular action: by incorporating the advantage function into the learning process, the agent can improve its policy by favoring actions with positive advantages and discouraging actions with negative advantages. This leads to more informed and effective decision-making as the agent learns to choose actions that are likely to lead to higher cumulative rewards. It also helps to reduce the variance of the policy gradient estimate, making the learning process more stable and efficient. With the computed advantage, the actor network updates its weights via gradient ascent and begins a new step of the algorithm.

Successful application of the A2C algorithm has been demonstrated, for instance, in [34], where the authors controlled an inverted pendulum on a cart based on image data.

3.3 Proximal Policy Optimization (PPO)

The PPO algorithm improves the stability of the agent's training by limiting the extent of policy updates. The policy is updated iteratively based on experiences collected from the environment, using a surrogate objective function that encourages the policy to take actions that lead to higher rewards while maintaining close proximity to the previous policy [13]. PPO addresses some of the challenges encountered in traditional policy gradient methods. The likelihood of oscillations or divergence in the learning process is reduced, and the exploration of new policies and exploitation of current policies is balanced. An example of a successful application of the PPO algorithm to control a complex multibody system is presented in the work of Kurinov et al. [35]. This research involved deep reinforcement learning, where an agent was trained to load and unload an excavator. The trained agent achieves an accuracy in the range of 67–97% of the maximum weight the bucket could hold. Although the reward function still fluctuated, the learning curves indicated that increasing the number of training episodes would likely lead to further improvements in the excavator's performance.

3.4 Deep Q-Network (DQN)

DQN is an RL algorithm that employs a neural network to learn a Q-function, which provides an estimate of the expected sum of future rewards for a given state-action pair [3]. Its operation is characterized by iteratively collecting experiences from the environment, storing them in a replay buffer, and subsequently utilizing this data to update the Q-function. To encourage exploration and prevent the agent from settling into suboptimal policies, DQN integrates a ε-greedy exploration strategy. This means that the agent chooses a random action with probability ε and chooses the action with the highest Q-value with probability $1 - \varepsilon$. Israilov et al. [36] applied the DQN algorithm to control the inverted pendulum on a cart both in an experimental set-up and in simulation for swing-up and stabilization of the pendulum in its unstable upward equilibrium without a dependency on initial conditions.

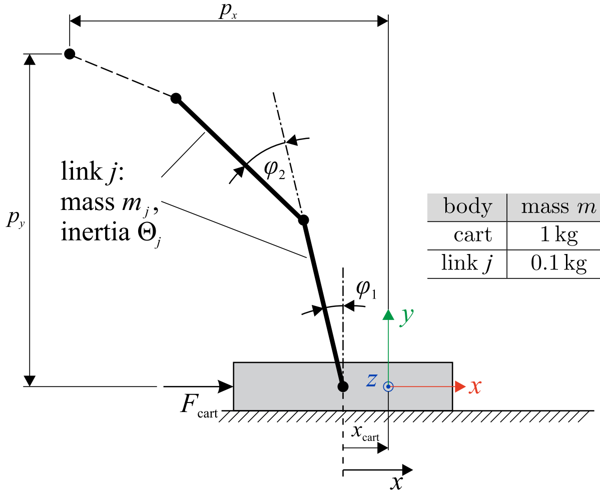
4 Reinforcement learning environment

In this section, the mechanical model, which resembles the control task or *environment*, is described. We shortly show how it is implemented and what the actions and states, which the agent uses to interface the simulation, are. Based on these quantities the rewards applied in the training and the tests to evaluate the agent are described.

4.1 Mechanical model

For learning the stabilization of the inverted N-link pendulum on a cart, shown in Fig. 3, a multibody simulation model is implemented using the multibody simulation library Exudyn [37]. The values of the model's physical and geometrical properties are based on [38] and benchmark examples from OpenAI Gym [39],[1] as shown in Fig. 3. Each link j is assumed to be rigid, has a mass m_j, length l_j, and inertia Θ_j. The inertia of the link j is specified relatively to the joint j, assuming thin rods with constant mass distribution over their length. The first body, undergoing only translational motion in the x-direction, is called

[1] https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py.



body	mass m	length l	inertia Θ
cart	1 kg	-	-
link j	0.1 kg	1m	$\frac{0.1}{3}\,\text{kg}\,\text{m}^2\,\left(=l^2\frac{m}{3}\right)$

Fig. 3 The inverted N-link pendulum is modeled using the minimal coordinates $\mathbf{q} = \mathbf{s} = [x_{\text{cart}}, \varphi_1, \ldots, \varphi_N]$. Each link consists of length l_j, mass m_j, and the moment of inertia Θ_j. The cart is a rigid body with mass m_{cart}. The values for the parameters are shown in the table

the *cart*. It has a mass of m_{cart} and is controlled with a force F_{cart}, which corresponds to the agent's *action*. The action is discrete with the force

$$F_{\text{cart}} = \begin{cases} +f_{\text{cart}} & \text{agent output } 1 \\ -f_{\text{cart}} & \text{agent output } 0\,. \end{cases} \tag{4}$$

In control theory, this is also known as *bang-bang* control [40]. While in OpenAI Gym [39] a continuous model is used for the inverted double pendulum on the cart,[2] we still use a discrete force for all models to be better able to compare the agents with each other. The links 1 to N are connected with rotational joints to their predecessor and form a so-called *kinematic tree*. The implementation of the model is done in a minimum coordinate formulation in which the cart-pole model corresponds to a 2D kinematic chain with one prismatic joint as the cart and N rotational joints for the links. The implementation follows Featherstone's algorithm for a kinematic tree, which is described in further detail in [41, 42], and the structure of the equations of motion as presented in Eq. (1). An implicit generalized-α solver [43] with constant step size $h = 0.02$ s is used to solve the equations of motion in order to guarantee stability even for large step sizes. An explicit integrator could be used as well, with the potential for even faster computation. However, the main computational expenses arise from the training and evaluation of the network. Hence, a performance increase due to additional simulation improvements would be minor. When using A2C in the training process of the single pendulum agent, only 7% of the computation time is spent on time integration and solving the equations of motion, while 39% is spent on the optimization algorithm and 37% on obtaining the policy. For the more complicated triple pendulum, 9% of the computation time is spent on time integration, 36% on optimization, and 33% on the policy respectively. Note that the agent's control frequency is chosen equal to the simulation step size. The min-

[2]https://www.gymlibrary.dev/environments/mujoco/inverted_double_pendulum/.

imum coordinates of the N-link inverted pendulum correspond to the model's position level states

$$\mathbf{s} = [s_0, s_1, \dots, s_\mathrm{N}]^T = [x_\mathrm{cart}, \varphi_1, \dots, \varphi_\mathrm{N}]^T \ , \tag{5}$$

with the cart position x_cart and the pendulum angles φ_j, with $j \in \{1, \dots, \mathrm{N}\}$. The agent also receives velocity-level states

$$\dot{\mathbf{s}} = [\dot{s}_0, \dot{s}_1, \dots, \dot{s}_\mathrm{N}]^T = [\dot{x}_\mathrm{cart}, \dot{\varphi}_1, \dots, \dot{\varphi}_\mathrm{N}]^T \ , \tag{6}$$

with the cart velocity $\dot{x}_\mathrm{cart}$ and the angular velocity of the jth link $\dot{\varphi}_j$. The position and velocity states form together the (full) state vector

$$\mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \dot{\mathbf{s}} \end{bmatrix} . \tag{7}$$

For the N-link pendulum, this leads to a total of $2\mathrm{N}+2$ states. The agent's interface consists of the action F_{cart} and full state vector $\mathbf{x}$. For general multibody systems, such as closed-loop systems, Exudyn allows the use of redundant coordinates, which are not compatible with the implemented RL methods. Therefore, the present dynamic model in the simulation uses the minimum coordinate formulation, such that no conversion between the state vector of the RL method and the simulation model coordinates is needed. The task terminates (and fails) if the allowed observation space of $|x_\mathrm{cart}| \le \chi_x$ and $|\varphi_j| \le \chi_\varphi \ \forall j \in \{1, \dots, N\}$ is left. The values of the position and angle thresholds χ_cart and χ_φ chosen for the experiments can be found in Sect. 5. The observation space for the velocities is not restricted.

4.2 Reward design

In the scope of this work, various different reward metrics are considered. In the benchmark example included in OpenAI Gym [39] the simple reward

$$r_0 = 1 \tag{8}$$

is used for each observation, while $r_0 = 0$ when the model stops and the episode ends—either because of a state exiting the allowed observation space and the environment terminates, causing the task to fail, or the maximum number of steps for the episode to be reached. It is similar to the reward from [38], where the reward is set to 0, while the pendulum is inside the desired observation space and set to -1 when the agent fails the task. Especially for inverse pendulum systems with a number of links $\mathrm{N} > 1$, the pendulum stabilizing with the cart moving into the $\pm x$ direction with constant velocity may be easier to achieve for the controller than to hold the unstable equilibrium position $\mathbf{s}_0 = [0, \dots, 0]$. In real systems, this drifting of the cart position can typically not be accepted as the space is limited. Thus to avoid this undesired behavior, for the rewards r_1 to r_3, the factor w_p weights the relative positional error and the angle error to each other. The reward

$$r_1 = 1 - w_p \frac{|x_\mathrm{cart}|}{\chi_\mathrm{cart}} - (1 - w_p) \frac{\sum_{j=1}^{\mathrm{N}} |\varphi_j|}{\mathrm{N}\chi_\varphi} \ , \tag{9}$$

combines the cart displacement x_cart with the sum of angles, such that $r_1 = 1$ is equal to the unstable equilibrium of the pendulum at $\mathbf{s}_0 = [0, \dots, 0]$. Tests on the inverted double pendulum showed that using the sum of absolute values $\sum_{j=1}^{N} |\varphi_j|$ leads to better results than the absolute value of the sum $\left|\sum_{j=1}^{N} \varphi_j\right|$. Furthermore, the reward

$$r_2 = 1 - w_p \frac{|p_x|}{\chi_{\text{cart}}} - (1 - w_p) \frac{\sum_{j=1}^{\text{N}} |\varphi_j|}{\text{N}\chi_\varphi}, \tag{10}$$

uses the tip position p_x instead of the cart position, whereas the third reward uses the tip position and the absolute value of the last link's angle only

$$r_3 = 1 - w_p \frac{|p_x|}{\chi_{\text{cart}}} - (1 - w_p) \frac{|\varphi_{\text{N}}|}{\chi_\varphi}. \tag{11}$$

The reward is calculated in each time step. In the following figures, the reward is shown accumulated over each episode.

4.3 Model evaluation (tests)

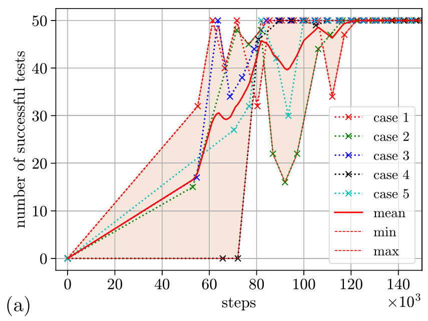
To assess the quality of the agent, it is evaluated periodically once the mean reward is greater than a reward threshold λ_r and the loss is under a threshold λ_l. For each evaluation, a number of n_{test} tests are run in the environment for $n_{\text{eval}} > n_{\text{learn}}$ steps, *e.g.*, for the single link model the number of time steps in the evaluation $n_{\text{eval}} = 5000$, which runs the test for 100 s if $h = 0.02$ s. The applied parameters are shown in Tables 2 and 3. In each test, the states are initialized with an initial perturbation, sampled for all states from a uniform random distribution in the range of $\pm\lambda_{\text{init}}$. The environment is simulated for the time steps $\mathbf{t} = \left[t_0, t_1, \ldots, t_{n_{\text{eval}}}\right]$, where $t_i = i\,h$. The test error is evaluated for each time step $i \geq \frac{3}{4} n_{\text{eval}}$, so the agent has time to reach the desired unstable equilibrium state $\mathbf{s}_0$ after the initial perturbation. The test error

$$e_{\text{test},i} = ||\mathbf{s}_i||_\infty \tag{12}$$

is then calculated for the ith time step from the maximum norm of the position states $\mathbf{s}_i = \left[s_{i,0}, \ldots, s_{i,N}\right]^T$. The total error is the largest error that occurred in the last quarter of the tested time steps,

$$e_{\text{test}} = \max(e_{\text{test},i})\, \forall\, i \geq \frac{3}{4} n_{\text{eval}}. \tag{13}$$

During the training process the current policy is tested regularly. Each time n_{test} tests are performed with varying initial conditions. The training of the agent is considered to be finished when, for each of the tests, the maximum error e_{test} is below the threshold value of χ_{test} and thereby successful. This could be used as a criterion for early stopping. As soon as the agent succeeds in a minimum number of tests, the training result is tracked, and the network parameters are saved upon improvement. To better assess the performance of the methods, in each experiment, several agents are trained and evaluated. In the later figures, the success of each method over the trained steps is shown by the best, worst, and mean performance as displayed in Fig. 4(a). The *min* and *max* curves form the envelope. As the agents are not evaluated in the exact same time steps, as seen in (b), linear interpolation and resampling are used for plotting. In the first 50,000 steps, the model is not evaluated due to its performance not being sufficiently developed yet. Consequently, the linear increase of successes observed in the plot is attributed to the interpolation applied to the data.



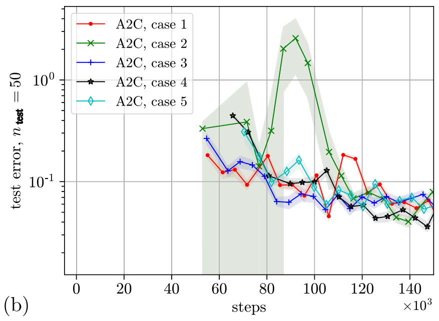


Fig. 4 The success of the agent in the inverted single pendulum environment over the training process is shown as the number of successful tests (a) and the test sum of test errors (b). Five agents are trained, and the worst/best and mean number of successful tests for the agents is shown in (a). After $140 \cdot 10^3$ steps, all models succeed in all 50 tests (Color figure online)

5 Performance evaluation of RL methods

In the following section, we evaluate the performance of the standard RL methods A2C, PPO, and DQN in the N-link environments previously described. The analysis starts with a 1-link system, progressing subsequently to a 2-link and a 3-link system. In particular, different rewards are explored, ultimately concluding with modifications of the environment and the influence of the agent's performance.

For all systems, we consider the task of balancing the inverted pendulums in the unstable upward-facing equilibrium. Note that the swing up is considered as a separate task [44], as it is in control theory. Stabilizing the inverted pendulum becomes successively more challenging by adding more links, as every link would be an unstable system by itself, even if the other links were fixed. It is known that the single pendulum can be stabilized with the discrete bang-bang type control as it is used not just as an example in many RL environments but also in classical control [45]. For the stabilization of the double and triple pendulum in the literature, more advanced control strategies such as H-∞ methods [46] are used.

In the following results, each experiment is repeated eight times. Different seeds are used for initializing random values, *e.g.*, the weights of the neural networks or the initialization of the environment in each episode and test. Not all eight trainings are shown in each plot to avoid cluttered figures. For all experiments, stable-baselines3[3] [28] is used in Version 1.7.0. Furthermore, PyTorch 2.0.0 and Exudyn Version 1.6.65 are used in the following experiments. The methods A2C, PPO, and DQN are used as implemented in stable-baselines3 with standard parameters for the agent, except the ones shown in Table 2. The inverted pendulum with one link is a classical benchmark example for RL methods and nonlinear control [45] and is available in many environments such as MuJoCo [47] and OpenAI Gym [39], which is since 2021 further developed under the name *Gymnasium*[4]. For all parameters of the RL methods that are not shown here, the standard parameters implemented in `stable-baselines3, V1.7.0` are applied.

The different methods and rewards are compared for the single, double, and triple link models (1, 2, and 3 links). For the following evaluations, if not stated differently, the parameters shown in Table 2 and Table 3 are used. The parameters from Table 2 are used inde-

[3] https://stable-baselines3.readthedocs.io/en/master/guide/algos.html.

[4] https://gymnasium.farama.org/.

Table 2 The hyperparameters used for the RL methods in the shown investigations. The physical parameters are provided in Fig. 3

Parameter	Value	Parameter	Value
Reward function	r_3, Eq. (11)	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`	$n_{\text{episode,max}}$
A2C: learning rate	$7 \cdot 10^{-4}$	learning rate PPO/DQN	$5 \cdot 10^{-4}$

Table 3 Environmental, reward, and training parameters for the environments with link numbers 1 to 3. If $x_{\text{cart}} > \chi_x$ or $\varphi_i > \chi_\varphi$, then the episode ends or the evaluation fails. To receive a reasonable model, approximately n_{learn} steps were necessary for our experiments

Name	Parameter	1 link	2 link	3 link
Cart force	f_{cart} in N	12	40	60
Threshold cart position	χ_{x} in m	1.2	3.6	5.4
Threshold link angle	χ_φ in rad	$\frac{\pi}{20}$	$\frac{\pi}{10}$	$3\frac{\pi}{20}$
For φ_1 to φ_n				
Max test error	χ_{test}	0.2	0.5	0.75
Reward position factor	w_p	0.5	0.5	0.5 to 1
Required training steps A2C/PPO	n_{learn}	$200 \cdot 10^3$	$400 \cdot 10^3$	$500 \cdot 10^3$
Max episode length	$n_{\text{episode,max}}$	1280	1536	2048
Tests per evaluation	n_{test}	50	50	100

pendently of the number of links. The physical properties of the considered environments are different: with a higher number of links N, the total mass and inertia increase. Thus the environment, reward, and training parameters are adjusted with the number of links according to Table 3. Suitable parameters have been found by parameter variations. To control the inverted triple pendulum, a higher force is needed because of the higher total inertia and mass. In the shown plots we train for more steps than the required learning steps, compare Table 3, to check the performance of all methods and to observe the training behavior.

5.1 Inverted single pendulum: 1 link

For the inverted pendulum on a cart, also called *cartpole*, all three RL methods lead to successful agents with the standard settings from stable-baselines3 without adjustment of hyperparameters.

In Fig. 5 the successes over the course of the training are shown for the different methods, whereby the solid line resembles the mean number of successful tests, and the highlighted area represents the envelope of the best and worst result as described in Sect. 5. The figure also shows that the agent passes all tests already before the minimum of the loss is reached, see Fig. 6. Although for A2C the tests are passed after $120 \cdot 10^3$ steps, the agent still learns and the occurring errors in the evaluations, described in Sect. 5, decrease. In Fig. 6 the training for the inverted single pendulum on the cart is shown for A2C, PPO, and DQN. The parameter `case`, shown from 1 to 3, is the seed set for all randomized initialized values as the weights of the network and the state of the pendulum for training. The A2C method explores the fastest, and all test cases run successfully for the full episode lengths after

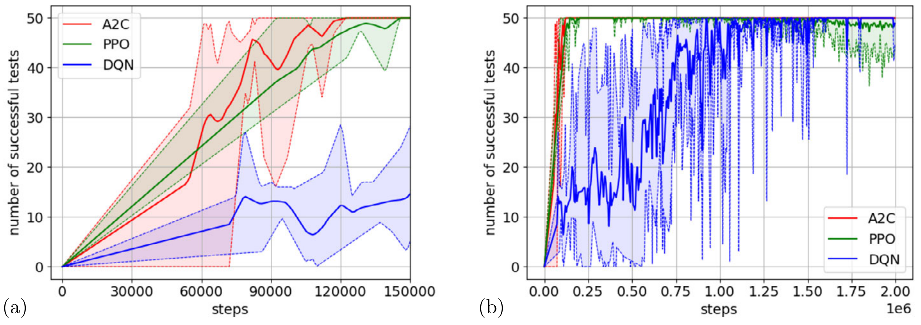


Fig. 5 The number of successful tests over the training zoomed into the first $150 \cdot 10^3$ steps (a) and over the longer training (b), which was needed for DQN to succeed. The solid line represents the interpolated mean between all tests, whereas the dashed lines represent the envelope with the worst and best results. With A2C the training succeeds the fastest and the model remains robust even when trained for longer (Color figure online)

$60 \cdot 10^3$ to $80 \cdot 10^3$ training steps, although there are drops in the reward afterward. If the reward per episode is close to its length, it indicates that the controller is at least able to keep the pendulum stable inside the thresholds χ_{cart} and χ_{φ}. With a maximum episode length of 1280, this corresponds to 25.6 seconds in the simulation. The loss decreases rapidly until $250 \cdot 10^3$ steps, whereas DQN takes over 10^6 learning steps to successfully stabilize the pendulum.

From an engineering perspective not only the randomized tests used for the evaluation are of interest, but instead for real-world application of the agent an area where the stabilization task is successful needs to be known. The question arises if this region of stability is *simply connected*. To visualize the stability regions of the agent in the state-space all states are set to 0 except the visualized states s_k and s_l. The state space is then discretized into a 2D grid array with a discretization size of

$$\mathrm{d}s_k = \frac{s_{k,\max} - s_{k,\min}}{n_k} \tag{14}$$

with the maximum value for the state $s_{k,\max}$, minimum $s_{k,\min}$, and the number of calculations in dimension k

$$\mathrm{d}s_l = \frac{s_{l,\max} - s_{l,\min}}{n_l}\,. \tag{15}$$

In each grid cell, the agent is evaluated n' times with the starting state for the test randomized in a range of $\pm\mathrm{d}s_k/2$ and $\pm\mathrm{d}s_l/2$ in the cell. With the states $s_k = \varphi_1$ and $s_l = \dot{\varphi}$, the heatmaps shown in Fig. 7 are calculated. In the bright areas, the agent succeeds with the task. Figure 7 (a) shows the complete stable region of the inverted single pendulum. In the numerical experiment the discretization is set to $n_k = n_l = 1000$ cells per direction, with $n' = 100$ tests per cell this leads to a total of 10^8 tests and a resolution of $2.4 \cdot 10^{-3}$ rad $\times$ $8 \cdot 10^{-3}$ rad s^{-1}. Figure 7 (b) shows in more detail the spiked shape of the outline of the stability zone. While the exact values of the zone differ between agents trained with different seeds, all tests show the same *ragged* structure at the ends and not a clear line which one would expect from a classical deterministic control algorithm. As shown by Mori in [48], a smooth boundary would be expected for a classical control algorithm. Besides the spikes at the outline of the

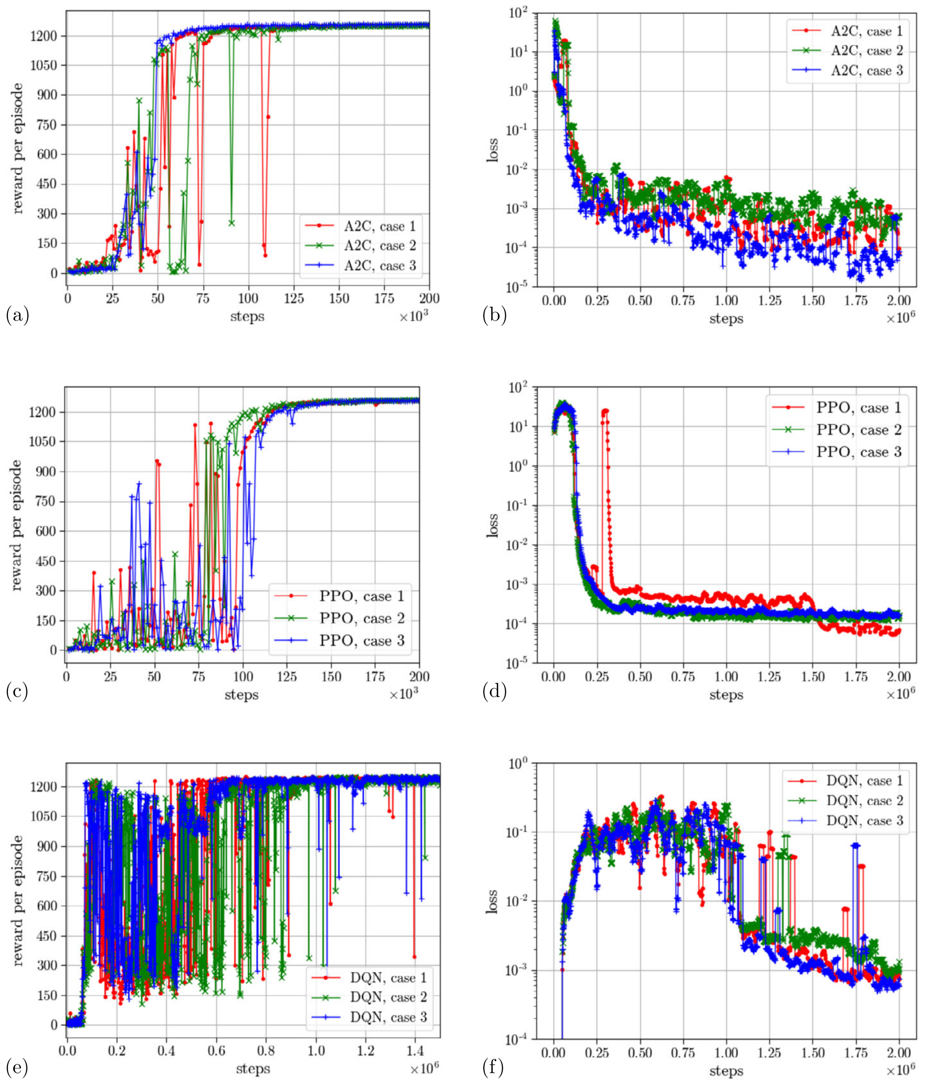


Fig. 6 Reward per episode and loss for A2C (a, b), PPO (c, d), and DQN (e, f). The maximum episode length is 1,264 steps, therefore with a maximum reward of 1 per step the reward per episode cannot surpass 1264. In the case of DQN, the model starts learning the control task after $50 \cdot 10^3$ steps, thus the loss is 0 until then (Color figure online)

stabilized zone isolated zones occur where the agent fails to stabilize the system. The region of stability shown in Fig. 7 is significantly larger than the range of states used for training and testing, shown as a rectangle.

To handle the large number of tests necessary to visualize the stability regions, the high-performance computing cluster *Leo5* is used. In the cluster, the computing power is separated into *nodes*, having their own CPUs, GPUs, and memory. Besides its simulation capabilities, the used multibody simulation library Exudyn also offers features for parameter

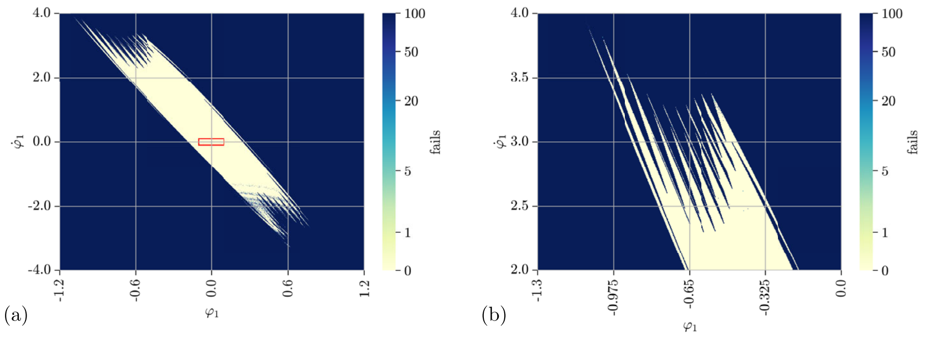


Fig. 7 The stability zones of the A2C agent trained in Fig. 6 (a), case 4. Within each grid cell, 100 tests are done with randomized starting states. The number of fails is plotted scaled by log (fails + 1) in the heatmap for better visibility of single failed tests. The environment's initialization for the agent's evaluation while training was performed in a randomization range of $x_i\,(t=0) \in [-\lambda_{\text{init}}, +\lambda_{\text{init}}]$ with $\lambda_{\text{init}} = 0.1$. This training range is shown on the left in red and is much smaller than the zone of stability (Color figure online)

variation and parallel processing, both on multi-core CPUs and on clusters using the message passing interface (MPI). With MPI the agent can be tested using shared or distributed memory, enabling parallelization of the calculations over multiple nodes of the cluster. With an average CPU time of 50.6 ms per test for the single pendulum environment, 10^8 tests can be computed in 11.7 hours on 120 parallel MPI tasks. As for the neural network, only 64 neurons are used for each of the 2 hidden layers, no speed-up was observed using the GPU. The double link environment requires 89.7 ms per test resp. The triple link environment takes on average 194.9 ms per test. As the test terminates when leaving the observation space χ_x or χ_φ respectively, the computation times can vary greatly between individual tests. Successful tests require a significantly longer computation time.

5.2 Inverted double pendulum: 2 link environment

The inverted double pendulum can be seen as an extension of the classical cartpole system with higher-order nonlinearities and is more difficult [49]. In Fig. 8 the training of all three methods is shown. After the initial fast learning for both A2C and PPO, the agents' number of successes becomes notably worse when training for a longer time, which was not the case for the single pendulum shown in Fig. 6. After $25 \cdot 10^4$ to $50 \cdot 10^4$ steps, both A2C and PPO run for the maximum number of steps $n_{episodes,\max}$ without early termination.

The agents' successes, shown in Fig. 9, using PPO are on average more stable than A2C. As PPO uses clipping to avoid too large updates for the policies, this slower but more stable learning can be explained.

In Fig. 10, the stability zones are evaluated for A2C cases 1 and 2 as well as PPO case 1 from the training shown in Fig. 8. Figures (a), (c), and (d) have a resolution of $n_l = n_k = 1000$ grid cells in φ_1 and φ_2, leading to a resolution of $1.7 \cdot 10^{-3}$ rad $\times$ $8 \cdot 10^{-4}$ rad in φ_1 and φ_2 for each grid cell, with 100 tests per grid cell this leads to $100 \cdot 10^6$ tests calculated per figure. The outline of the zone in which the controller is able to stabilize is similar for all agents. As expected, changing the environment can also change the size of the stabilized zone, *e.g.*, increasing the control frequency could lead to better agent performance. In all tests of the double pendulum, the shown *spikes* on the outline of the stabilized zone were observed.

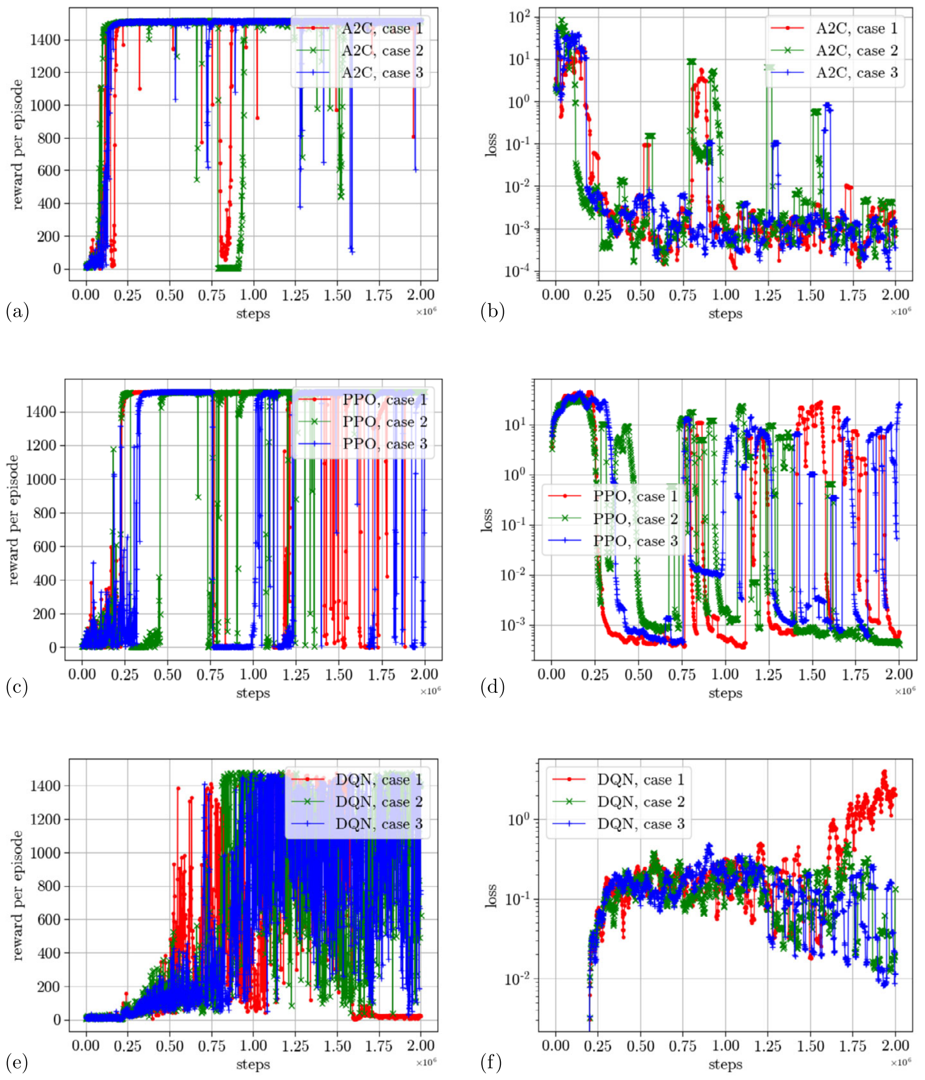


Fig. 8 The results for all three methods applied to the double link model. Equivalent to the previous training of the single pendulum, (a) and (b) show the results from A2C, (c) and (d) PPO, and (e) and (f) DQN (Color figure online)

The distorted shape of the stability region for values $\text{sign}(\varphi_1) = -\text{sign}(\varphi_2)$ can be explained as the opposing signs help to move the inverted pendulum closer to its equilibrium state and thereby increase the range of values where the system can be stabilized by the agent. Figure 10 (b) shows a detail of an unstable zone. The size of this zone is only $2.5 \cdot 10^{-3}$ rad $\times$ $2.5 \cdot 10^{-3}$ rad and could be easily missed when testing the agent. It is surrounded by regions of success, and changing the starting value for any variable by 0.01 leads the agent to succeed in the test. The shape of the local failure region on the inside is similar to the outer borders of the stability region.

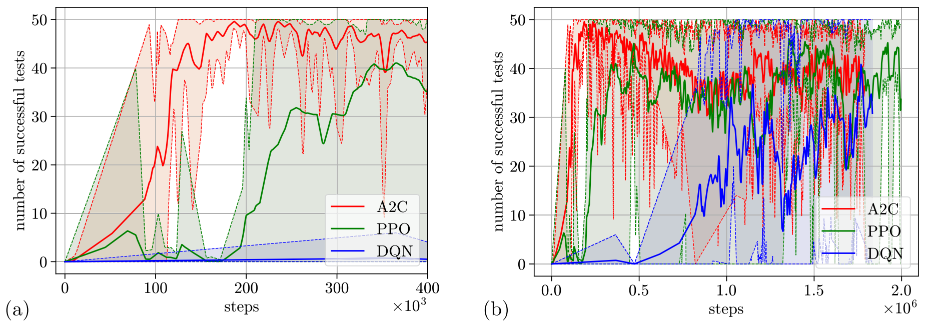


Fig. 9 The successful tests shown over learning for the double link model (Color figure online)

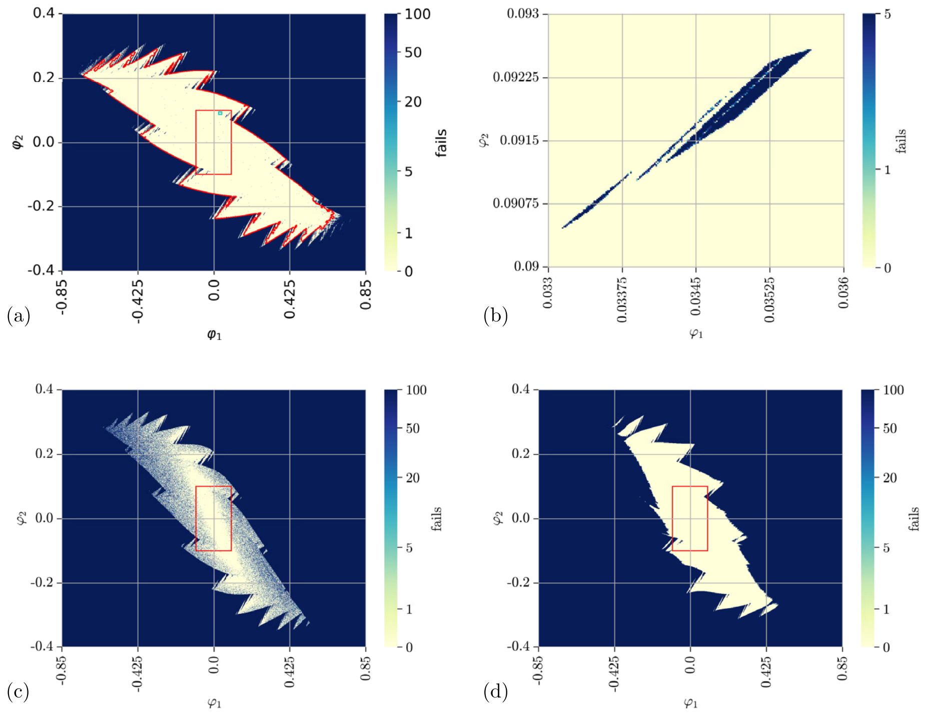


Fig. 10 The stability zones for the agents from Fig. 8. For the training, the states are sampled from the red rectangle. In (a), the agent corresponds to A2C, case 1. Inside the stabilized zone, there are small zones where the agent does not succeed in stabilizing the environment. The red line represents the smoothed outline of the main stability zone and is added to Fig. 16. The small spot on the upper right marked in cyan is shown in (b). (c) shows the A2C agent case 2, (d) PPO case 1 (Color figure online)

The high-frequency oscillations visible in Fig. 11 (b) are a result of the bang-bang control, which applies a discontinuous force of $\pm f_{cart}$ to the cart. These results are characteristic of the controller: when it manages to stabilize the inverted pendulum around the equilibrium position, it will still not reach the equilibrium perfectly and the cart will continue to oscillate.

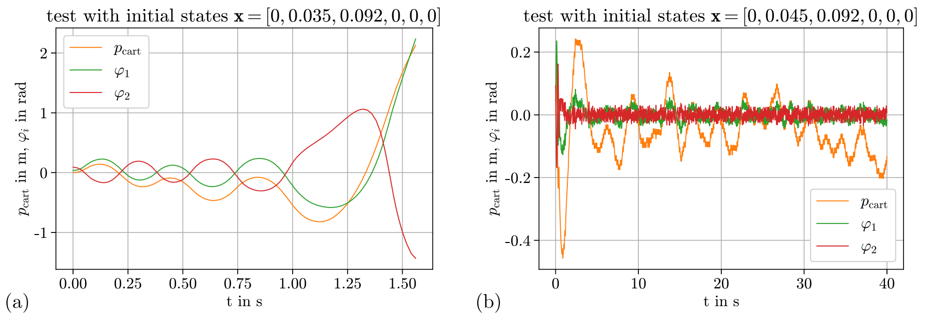


Fig. 11 Results for the double pendulum tests from Fig. 10, A2C case 1. The controller fails in case (a) by leaving the observation space $\pm\chi_{cart}$, χ_{φ}. This happens although the initial perturbations are close to the equilibrium, and the agent succeeds with marginally larger (or smaller) initial states shown in (b) (Color figure online)

5.3 Inverted triple pendulum: 3 link environment

The inverted 3-link pendulum on a cart, in the following called the triple link model, is a very challenging control problem because of its instability and nonlinearity. It is not just used for testing control strategies but is also a good model for a humanoid robot standing on one leg [46]. In Fig. 12, the training of the triple link pendulum is shown using the A2C and PPO methods. With both methods, the agent can learn to balance the system in the unstable equilibrium position although, in training, the system has problems keeping it upright for the training periods. In our experiments, the agent performed better when training with larger ($\lambda_{init,train}$, red rectangle in Fig. 14) and testing with smaller states ($\lambda_{init,test}$, green rectangle in Fig. 14) than setting the training and testing values to be equal. In contrast to the previous training results, the shown reward and loss are filtered in the postprocessing using a sliding window mean filter with size 50 for better visibility. The triple pendulum only succeeded with all tests when using the methods A2C or PPO, in the experiments using DQN did not lead to any successes, not even after $15 \cdot 10^6$ steps. As DQN already performed significantly worse for the inverted double pendulum, as shown in Fig. 9, this method was no longer considered for the triple link environment. As shown in Fig. 13 (a), the mean performance of the trained models is best after approximately $35 \cdot 10^4$ steps. The loss does not reach a minimum, and the model changes, but it is not improving anymore. In the stabilization process, the cart typically oscillates around the equilibrium position, as shown in Fig. 13 (b). The amplitude of this oscillation increases with the number of links. To reduce this oscillation, the reward position factor w_p of the reward is increased to $w_p = 0.7$, helping to keep the cart position centered.

In Fig. 14, the stable zones for the A2C agent of case 1 are shown. The difficulty of the task is not only increased by the higher number of links but also by the definition of the angles φ_j relative to their predecessor. Thus the angle φ_N measured against the vertical can become up to $N\chi_{\varphi}$, as it can be seen in Fig. 3. The red rectangle shows the training area, and the green inner rectangle is the testing area. Increasing the control frequency enlarges the stability region for the triple pendulum significantly. As the agent uses bang-bang control and chooses the action in each state only by the measured states, we found that the control frequency can also be changed for the testing without new training data.

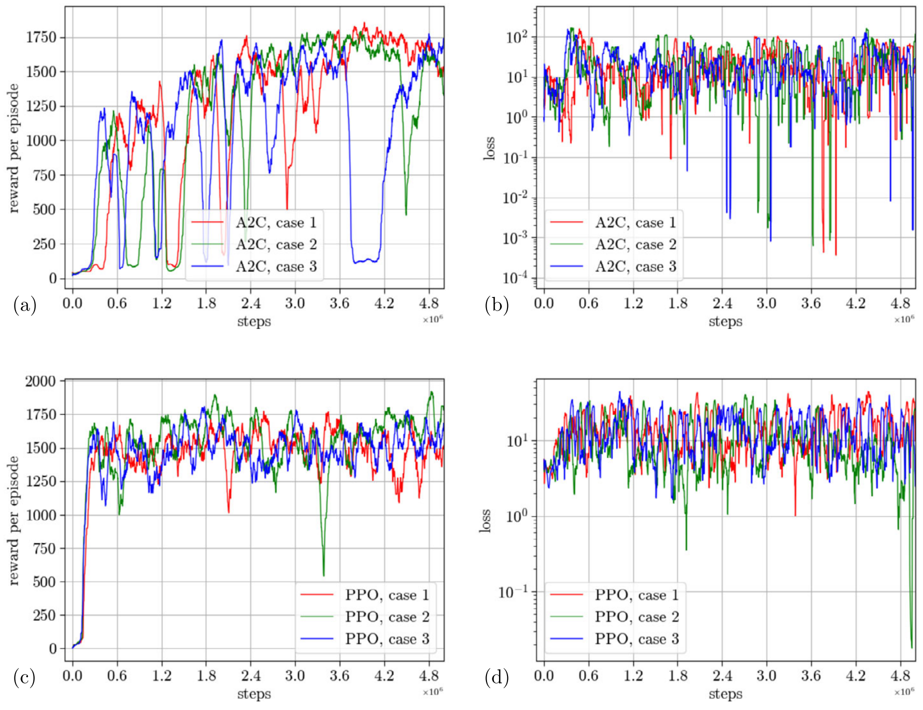


Fig. 12 The results for all three methods applied to the triple link model. (a) and (b) show the results from A2C and (c) and (d) PPO. The DQN agent failed all tests after $15 \cdot 10^6$ steps. The shown reward and loss are filtered using a sliding window mean filter with a size of 50 (Color figure online)

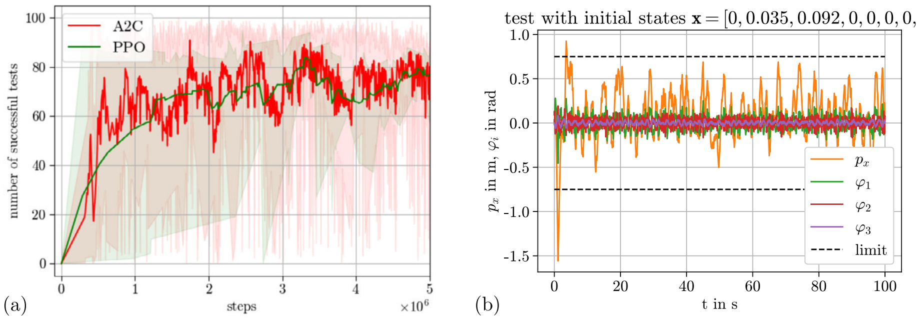


Fig. 13 (a) The successes plotted over time for A2C and PPO agents on the triple link model. (b) The results of the stabilization for initial states $\mathbf{s}_0$, showing the characteristic oscillations of the cart position p_{cart}. The shown limit corresponds to χ_{test} from Table 3 (Color figure online)

5.4 Evaluation of rewards

In Fig. 15, the different rewards described in Sect. 4.2 are shown. The reward r_0, sometimes used for the single inverted pendulum on the cart, offers a significantly worse performance when the number of steps for evaluation is larger than the test's maximum episode length

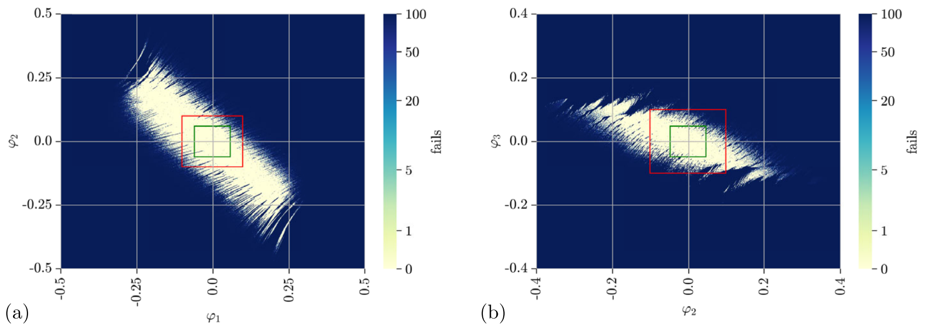


Fig. 14 The stable zones for the triple link model. The red squares are the intervals used for training, and the green for testing. The A2C agent of case 1 is tested using the initial states φ_1 and φ_2 in (a), and φ_2 and φ_3 in (b). An error in φ_3 cannot be compensated as well as in the other angles (Color figure online)

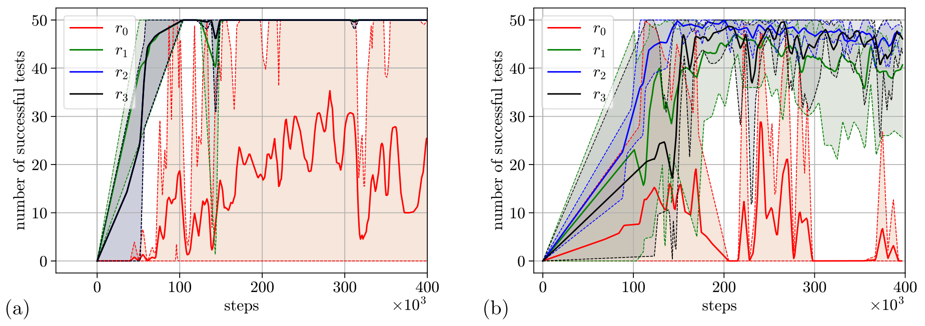


Fig. 15 The number of successful evaluations depends on the chosen reward function for the single link (a) and the double link (b). Note that reward functions r_2 and r_3 are identical for the single link as $\sum_{j=1}^{n} \varphi_i = \varphi_1$. The *baseline* reward r_0 can work for the single link model but worsens with increasing complexity. It can be observed that the cart position drifts slowly outside of the observation space χ_{cart}. The rewards r_2 and r_3 work comparably well (Color figure online)

in training as drift in the cart position is not reflected in training. The reward r_1 couples the position of the cart with the sum of absolute angles. Rewards r_2, which couples the tip's position p_x with the sum of absolute angles, and r_3, calculated from p_x and the last angle φ_N, work comparably well. For the single link model, the difference is notably smaller than the double link, where r_0 does not work.

5.5 Agents tested on environments with modified parameters

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 investigation is conducted using the inverted double pendulum on a cart model from Sect. 5.2. The A2C agent (case 1), shown in Fig. 10 (a), serves as a basis of the study. In each subplot of Fig. 16, the stability zone is calculated for the same agent within modified environments, and the contour of the stability zone using the original environment is added. Friction is implemented by assuming Coulomb friction based on a friction torque τ_0, which acts against rotation and is

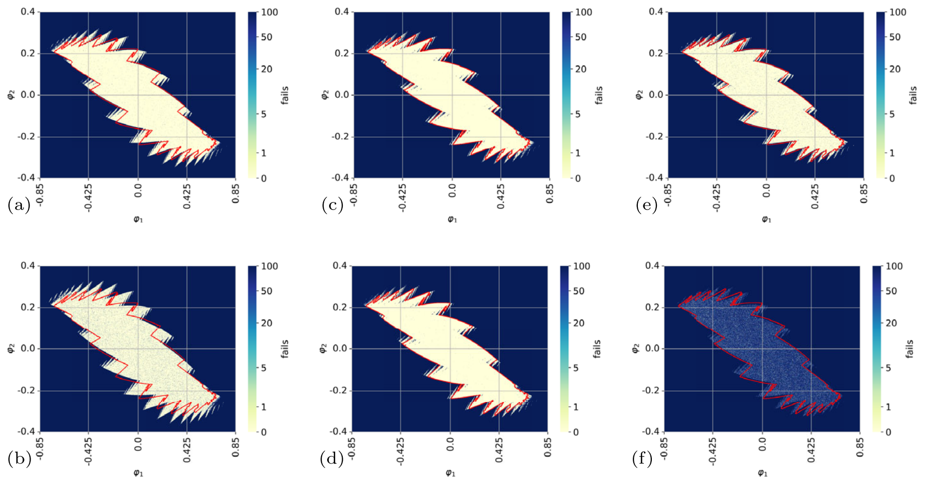


Fig. 16 The stability zone for the double link system using the A2C agent, case 1. The red boundary corresponds to that depicted in Fig. 10 (a). The environment parameters are modified as follows: (a) $1.1\,l$, (b) $1.2\,l$, (c) $1.1\,m$, (d) $1.2\,m$, (e) $f_{rel} = 0.01$, and (f) $f_{rel} = 0.02$. In the areas shown in the figure, the proportion of successes to the total number of tests is (a) 25.66%, (b) 26.97%, (c) 23.25%, (d) 23.13%, (e) 23.17%, and (f) 8.31%. In the original stability zone, 23.33% of the tests are successful (Color figure online)

independent of the applied forces. In the implementation we use

$$\tau_{f,i}(\omega_i) = \phi(\omega_i)\ \mathrm{sign}(\tau_0) = \phi(\omega_i)\ \mathrm{sign}(\omega_i)\ mg\frac{l}{2} f_{\mathrm{rel}}\,, \tag{16}$$

with the regularization

$$\phi(\omega) = \begin{cases} \frac{|\omega|}{\omega_{\mathrm{reg}}}, & |\omega| < \omega_{\mathrm{reg}} \\ 1 & \text{else}\,, \end{cases} \tag{17}$$

to the ith joint. The regularization provides improved numerical stability [50] by smoothing the discontinuity at $\omega = 0$ and is chosen as $\omega_{\mathrm{reg}} = 5 \cdot 10^{-3}$. We introduce the factor $f_{rel} \in]0, 1[$, denoted as relative friction, as the single pendulum would be self-locking up to the angle of $\varphi_1 = \arcsin(f_{\mathrm{rel}})$. For the presented tests, the cart's properties remain unchanged. In Fig. 16 (a) and (b) the link's length is increased by 10% and 20%, respectively, while in Fig. 16 (c) and (d) the link's mass is increased by 10% and 20%. In Fig. 16 (e) and (f) the relative friction f_{rel} is set to 1% and 2%, respectively. The observations show that increasing the length of the links increases the area inside of the contour, but also increases the number of blind spots observed. Increasing the link's mass has only a minimal effect on both, contour and blind spots, when changed in this scale. Note that longer links with the same mass also increase the inertia Θ quadratically as shown in Fig. 3.

A friction value of $f_{\mathrm{rel}} = 0.01$, being large compared to values of roller bearings, only slightly affects the shape of the stability zone of the agent but increases the number of blind spots. However, with a larger friction value, $f_{\mathrm{rel}} = 0.02$, the agent fails mostly, also inside the original contour. To improve the performance of the agent in the new environment, *e.g.*, with friction, either such effects could be randomly added during training of the agent or transfer learning [51] could be applied instead of training a new agent to reduce the computation time.

6 Conclusions

In this paper, we show the application of reinforcement learning (RL) methods in highly nonlinear multibody systems. We revised the classic benchmark example of the inverted single link pendulum on a cart, in the RL context often called *cartpole*, and extended it with further links to the inverted double and triple link models and showed the performance of the standard algorithms A2C, PPO, and DQN in solving these control environments. For the single and double link models, all algorithms succeeded, while DQN performed considerably worse and did not succeed in any tests for the triple link model, where it made no progress toward successful stabilization. We looked into the performance of the algorithms for the different environments, compared the training for different rewards, and explored in which regions the trained agents are successful. In addition to the accompanied testing of the agent while training, we performed extensive testing of the system after the training was finished. We demonstrated the *stability regions* of the agents and showed that the zones of success resp. failure of the agent are not closed or smooth surfaces in the parameter space but are highly fractured and nonsmooth, opposing to a classic controller. We observed the increasing importance of well-chosen rewards for more complex systems.

For the single and double link model, the stability zones can become significantly larger than the range of states during training. For the triple link model with a comparably slow control frequency of 50 Hz, the system can still be stabilized in a smaller region, although a higher control frequency would help to control the system. We found that the agent may still fail the control task with starting conditions close to the equilibrium, where it would be expected to work well. While this failure depends on the RL method and training parameters, in particular, small zones of failure are hard to detect by the randomized tests in the training and will be considered in future research. We also exemplarily showed that a trained RL agent can be applied to a new environment with considerably modified parameters, such as inertia or friction, the latter one limited to small values only. A future research goal will be finding a computationally more efficient way to compute the presented *stability zones*, as calculating millions of tests for every agent may not be practically feasible in engineering applications. Future research might include methods supporting continuous actions such as *Soft Actor critic* and *Twin Delayed DDPG*.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11044-023-09960-2.

Author contributions Peter Manzl: formal analysis, investigation, methodology, software, visualization and writing of the original draft. Oleg Rogov: methodology, visualization and writing of the original draft. Johannes Gerstmayr: conceptualization, project administration, software and investigation. Aki Mikkola: conceptualization and funding aquisition. Grzegorz Orzechowski: conceptualization, methodology and project administration All authors reviewed and approved the manuscript.

Funding Open access funding provided by University of Innsbruck and Medical University of Innsbruck. The computational results presented here have been achieved partly using the LEO HPC infrastructure of the University of Innsbruck.

The authors Rogov, Mikkola, and Orzechowski gratefully acknowledge funding provided by Business Finland's "Kumppanuusmalli – SANTTU – LUT" project under grant 8859/31/2021.

Declarations

Competing interests Non-financial interests: Johannes Gerstmayr is a member of the Editorial Advisory Board of Multibody System Dynamics. Aki Mikkola is co-editor-in-chief of the journal Multibody System Dynamics. Johannes Gerstmayr and Grzegorz Orzechowski are guest editors of the special issue on "Artificial intelligence within multibody system dynamics" in the journal Multibody System Dynamics. None of the listed editors or editorial board members receive compensation for the listed appointments.

References

1. Shabana, A.A.: Dynamics of Multibody Systems, 4th edn. Cambridge University Press, New York (2013)
2. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. A Bradford Book, Cambridge (2018)
3. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing Atari with deep reinforcement learning. CoRR (2013). arXiv:1312.5602
4. Hashemi, A., Orzechowski, G., Mikkola, A., McPhee, J.: Multibody dynamics and control using machine learning. Multibody Syst. Dyn. (2023). https://doi.org/10.1007/s11044-023-09884-x
5. Benatti, S., Young, A., Elmquist, A., Taves, J., Serban, R., Mangoni, D., Tasora, A., Negrut, D.: PyChrono and gym-chrono: a deep reinforcement learning framework leveraging multibody dynamics to control autonomous vehicles and robots. In: Advances in Nonlinear Dynamics, pp. 573–584 (2022). https://doi.org/10.1007/978-3-030-81166-2_50
6. Sun, C., Orbik, J., Devin, C., Yang, B., Gupta, A., Berseth, G., Levine, S.: Fully Autonomous Real-World Reinforcement Learning with Applications to Mobile Manipulation (2021)
7. Li, M., Wen, Y., Gao, X., Si, J., Huang, H.: Toward expedited impedance tuning of a robotic prosthesis for personalized gait assistance by reinforcement learning control. IEEE Trans. Robot. **38**(1), 407–420 (2022). https://doi.org/10.1109/TRO.2021.3078317
8. Song, S., Kidziński, Ł., Xue, B.P., Ong, C., Hicks, J., Levine, S., Atkeson, C.G., Delp, S.L.: Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. J. NeuroEng. Rehabil. **18**, 1–17 (2021). https://doi.org/10.1186/s12984-021-00919-y
9. Buşoniu, L., de Bruin, T., Tolić, D., Kober, J., Palunko, I.: Reinforcement learning for control: performance, stability, and deep approximators. Annu. Rev. Control **46**, 8–28 (2018). https://doi.org/10.1016/j.arcontrol.2018.09.005
10. Zhu, S., Liu, S., Feng, S., Huang, M., Sun, B.: An optimization method for the inverted pendulum problem based on deep reinforcement learning. J. Phys. Conf. Ser. **2296**(1), 012008 (2022). https://doi.org/10.1088/1742-6596/2296/1/012008
11. Manrique, C., Pappalardo, C., Guida, D.: A parametric study of a deep reinforcement learning control system applied to the swing-up problem of the cart-pole. Appl. Sci. **10**, 9013 (2020). https://doi.org/10.3390/app10249013
12. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning (2016). ArXiv preprint arXiv:1602.01783
13. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 3191–3199 (2017). JMLR.org
14. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. (2018). arXiv:1801.01290
15. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. arXiv (2018)
16. Abbass, M., Kang, H.-S.: Drone elevation control based on python-unity integrated framework for reinforcement learning applications. Drones **7**, 225 (2023). https://doi.org/10.3390/drones7040225
17. Saito, N., Oda, T., Hirata, A., Toyoshima, K., Hirota, M., Barolli, L.: Simulation results of a DQN based AAV testbed in corner environment: a comparison study for normal DQN and TLS-DQN. In: Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 156–167. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-79728-7_16
18. Mousa, A., Weiss, G.: Advanced energy management strategies for plug-in hybrid electric vehicles via deep reinforcement learning (2022)
19. Jalón, J., Bayo, E.: Kinematic and Dynamic Simulation of Multibody Systems. The Real-Time Challenge. Springer, New York (1994)

20. Cuadrado, J., Cardenal, J., Bayo, E.: Modeling and solution methods for efficient real-time simulation of multibody dynamics. Multibody Syst. Dyn. **1**(3), 259–280 (1997). https://doi.org/10.1023/A:1009754006096
21. Chaudhary, H., Saha, S.: Dynamics and Balancing of Multibody Systems, vol. 37, pp. 1–182. Springer, Berlin (2009). https://doi.org/10.1007/978-3-540-78179-0
22. Yu, X., Mikkola, A., Pan, Y., Escalona, J.L.: The explanation of two semi-recursive multibody methods for educational purpose. Mech. Mach. Theory **175**, 104935 (2022). https://doi.org/10.1016/j.mechmachtheory.2022.104935
23. Haug, E., Yen, J.: Generalized coordinate partitioning methods for numerical integration of differential-algebraic equations of dynamics. Comput. Syst. Sci. **69**(6), 97 (1990). https://doi.org/10.1007/978-3-642-76159-1_5
24. Liu, Y., Quan, F., Chen, H.: Adaptive nonlinear MPC for trajectory tracking of an overactuated tiltrotor hexacopter (2022). ArXiv preprint arXiv:2211.06762
25. Elagib, R., Karaarslan, A.: Sliding mode control-based modeling and simulation of a quadcopter. J. Eng. Res. Rep. **24**, 32–41 (2023). https://doi.org/10.9734/jerr/2023/v24i3806
26. Dorf, R., Bishop, R.: Modern Control Systems. Person Education Limited, Hoboken (2021)
27. Astrom, K.J., Murray, R.M.: Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press, Princeton (2008)
28. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: reliable reinforcement learning implementations. J. Mach. Learn. Res. **22**(268), 1–8 (2021)
29. Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V., Brevdo, E.: TF-Agents: a library for reinforcement learning in TensorFlow (2018). https://github.com/tensorflow/agents [Online; accessed 12-June-2023]
30. Ramakrishnan, R., Kamar, E., Dey, D., Shah, J., Horvitz, E.: Discovering blind spots in reinforcement learning. In: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), pp. 1017–1025. IFAAMAS, Stockholm (2018). arXiv:1805.08966
31. Bhagat, S., Banerjee, H.: Deep reinforcement learning for soft robotic applications: brief overview with impending challenges (2018). https://doi.org/10.20944/preprints201811.0510.v2
32. Nachum, O., Norouzi, M., Xu, K., Schuurmans, D.: Bridging the gap between value and policy based reinforcement learning (2017). arXiv:1702.08892
33. Watkins, C., Dayan, P.: Technical note: Q-learning. Mach. Learn. **8**, 279–292 (1992). https://doi.org/10.1007/BF00992698
34. Zheng, Y., Li, X., Xu, L.: Balance control for the first-order inverted pendulum based on the advantage actor-critic algorithm. Int. J. Control. Autom. Syst. **18** (2020). https://doi.org/10.1007/s12555-019-0278-z
35. Kurinov, I., Orzechowski, G., Hamalainen, P., Mikkola, A.: Automated excavator based on reinforcement learning and multibody system dynamics. IEEE Access **8**, 213998–214006 (2020). https://doi.org/10.1109/ACCESS.2020.3040246
36. Israilov, S., Fu, L., Sánchez Rodríguez, J., Fusco, F., Allibert, G., Raufaste, C., Argentina, M.: Reinforcement learning approach to control an inverted pendulum: a general framework for educational purposes. PLoS ONE **18**, 0280071 (2023). https://doi.org/10.1371/journal.pone.0280071
37. Gerstmayr, J.: Exudyn – a C++-based Python package for flexible multibody systems. Multibody Syst. Dyn. (2023). https://doi.org/10.1007/s11044-023-09937-1
38. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans. Syst. Man Cybern. Syst. **5**, 834–846 (1983). https://doi.org/10.1109/TSMC.1983.6313077
39. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI gym (2016). ArXiv preprint arXiv:1606.01540
40. Sonneborn, L., Van Vleck, F.: The bang-bang principle for linear control systems. J. Soc. Ind. Appl. Math., A, on Control **2**(2), 151–159 (1964). https://doi.org/10.1137/0302013
41. Featherstone, R.: The calculation of robot dynamics using articulated-body inertias. Int. J. Robot. Res. **2**(1), 13–30 (1983). https://doi.org/10.1177/027836498300200102
42. Featherstone, R., Orin, D.E.: Dynamics. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, pp. 37–66. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32552-1_3
43. Arnold, M., Brüls, O.: Convergence of the generalized-α scheme for constrained mechanical systems. Multibody Syst. Dyn. **18**, 185–202 (2007). https://doi.org/10.1007/s11044-007-9084-0
44. Spong, M.W.: Underactuated mechanical systems. In: Control Problems in Robotics and Automation, pp. 135–150. Springer, Berlin (2005). https://doi.org/10.1007/BFb0015081
45. Boubaker, O.: The inverted pendulum benchmark in nonlinear control theory: a survey. Int. J. Adv. Robot. Syst. **10**(5), 233 (2013). https://doi.org/10.5772/55058

46. Tsachouridis, V., Medrano-Cerda, G.: Discrete-time H∞ control of a triple inverted pendulum with single control input. IEE Proc., Control Theory Appl. **146**(6), 567–577 (1999). https://doi.org/10.1049/ip-cta:19990588
47. Todorov, E., Erez, T., Tassa, Y.: Mujoco: a physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033. IEEE, Los Alamitos (2012). https://doi.org/10.1109/IROS.2012.6386109
48. Mori, S., Nishihara, H., Furuta, K.: Hybrid controller for inverted pendulum. Trans. Soc. Instrum. Control Eng. **12**(4), 482–487 (1976). https://doi.org/10.9746/sicetr1965.12.482
49. Zhong, W., Rock, H.: Energy and passivity based control of the double inverted pendulum on a cart. In: Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01) (Cat. No. 01CH37204), pp. 896–901. IEEE, Los Alamitos (2001). https://doi.org/10.1109/CCA.2001.973983
50. Marques, F., Flores, P., Claro, J.P., Lankarani, H.M.: Modeling and analysis of friction including rolling effects in multibody dynamics: a review. Multibody Syst. Dyn. **45**, 223–244 (2019). https://doi.org/10.1007/s11044-018-09640-6
51. Zhu, Z., Lin, K., Jain, A.K., Zhou, J.: Transfer learning in deep reinforcement learning: a survey. IEEE Trans. Pattern Anal. Mach. Intell. (2023). https://doi.org/10.1109/TPAMI.2023.3292075

Authors and Affiliations

Peter Manzl[1] · Oleg Rogov[2] · Johannes Gerstmayr[1] · Aki Mikkola[2] · Grzegorz Orzechowski[2]

✉ P. Manzl
Peter.Manzl@uibk.ac.at

O. Rogov
Oleg.Rogov@student.lut.fi

J. Gerstmayr
Johannes.Gerstmayr@uibk.ac.at

A. Mikkola
Aki.Mikkola@lut.fi

G. Orzechowski
Grzegorz.Orzechowski@lut.fi

[1] Department of Mechatronics, Universität Innsbruck, Technikerstrasse 13, Innsbruck, 6020, Tyrol, Austria

[2] Department of Mechanical Engineering, LUT University, Yliopistonkatu 34, Lappeenranta, 53850, South Karelia, Finland