Scientific Research Publishing

Circuits and Systems, 2025, 16(1), 1-24
https://www.scirp.org/journal/cs
ISSN Online: 2153-1293
ISSN Print: 2153-1285

Reinforcement Learning in Mechatronic Systems: A Case Study on DC Motor Control

Alexander Nüßgen[1,2*], Alexander Lerch[3], René Degen[1,2], Marcus Irmer[1,2], Martin de Fries[1,2], Fabian Richter[1], Cecilia Boström[2], Margot Ruschitzka[1]

[1]CAD CAM Center Cologne, Institute of Automotive Engineering Cologne (IFK), Faculty of Automotive Systems and Production, Cologne University of Applied Science, Cologne, Germany
[2]Division of Electricity, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden
[3]Heringer Consulting GmbH, Cologne, Germany
Email: *alexander.nuessgen@th-koeln.de

How to cite this paper: Nüßgen, A., Lerch, A., Degen, R., Irmer, M., de Fries, M., Richter, F., Boström, C. and Ruschitzka, M. (2025) Reinforcement Learning in Mechatronic Systems: A Case Study on DC Motor Control. *Circuits and Systems*, **16**, 1-24.
https://doi.org/10.4236/cs.2025.161001

Received: November 22, 2024
Accepted: January 17, 2025
Published: January 20, 2025

Abstract

The integration of artificial intelligence into the development and production of mechatronic products offers a substantial opportunity to enhance efficiency, adaptability, and system performance. This paper examines the utilization of reinforcement learning as a control strategy, with a particular focus on its deployment in pivotal stages of the product development lifecycle, specifically between system architecture and system integration and verification. A controller based on reinforcement learning was developed and evaluated in comparison to traditional proportional-integral controllers in dynamic and fault-prone environments. The results illustrate the superior adaptability, stability, and optimization potential of the reinforcement learning approach, particularly in addressing dynamic disturbances and ensuring robust performance. The study illustrates how reinforcement learning can facilitate the transition from conceptual design to implementation by automating optimization processes, enabling interface automation, and enhancing system-level testing. Based on the aforementioned findings, this paper presents future directions for research, which include the integration of domain-specific knowledge into the reinforcement learning process and the validation of this process in real-world environments. The results underscore the potential of artificial intelligence-driven methodologies to revolutionize the design and deployment of intelligent mechatronic systems.

Keywords

Artificial Intelligence in Product Development, Mechatronic Systems, Reinforcement Learning for Control, System Integration and Verification, Adaptive Optimization Processes, Knowledge-Based Engineering

1. Introduction

In the context of state-of-the-art automation technology, the control of dynamic systems is of paramount importance, as it enables the reliable and precise operation of machines and devices. The control of complex and dynamically changing environments presents a significant challenge. Despite the proven reliability of traditional control methods, such as the proportional-integral (PI) controller, they are unable to cope with external conditions that are subject to strong fluctuations or sudden faults. In such scenarios, the adaptability of conventional controllers is constrained by their fixed parameterization, which lacks the flexibility required to respond effectively to changing conditions. Methods such as Model Predictive Control (MPC) for highly nonlinear systems also require many computations [1] [2].

The advent of reinforcement learning (RL) offers the potential for the development of control strategies based on experiential learning. In contrast to a predefined parameter-based approach, an RL-based controller is capable of continuously adapting its control strategy, thereby achieving a superior control quality in dynamic and fault-prone environments. In this study, the Advantage Actor-Critic (A2C) algorithm, a well-established RL method, is employed to train an RL agent to control a direct current (DC) electric motor. The DC motor provides an illustrative example of the types of applications that require precise and robust speed control, such as those found in robotics or production plants [3]-[5].

The objective of this study is to address the critical challenges of dynamic adaptability and system stability in control engineering, with a particular focus on industrial applications such as robotics, autonomous vehicles, and precision manufacturing. By employing reinforcement learning, specifically the Advantage Actor-Critic algorithm, this study aims to develop a framework for intelligent, adaptable control systems that can accommodate the evolving demands of these industries.

The novelty of this work lies in its integration of reinforcement learning into the structured design of dynamic systems, showcasing its potential to complement or even surpass traditional control strategies such as the PI controller. Unlike conventional methods that rely on static parameters and manual tuning, the proposed framework dynamically optimizes control actions in response to changing conditions. By applying this methodology to the control of a DC motor, the study provides a comparative analysis of reinforcement learning and traditional approaches, offering new insights into the capabilities of AI-driven control strategies for mechatronic applications.

1.1. Motivation

The motivation behind the utilization of reinforcement learning in the field of control engineering can be attributed to the adaptive and robust nature of the learning process. Reinforcement learning enables the agent to generate its own experiences, learn from them, and continuously refine its control strategy. In contrast to classic

PI controllers, which are based on fixed controller parameters, an RL agent is capable of reacting to changing environmental conditions and adapting dynamically to new requirements. The potential of an RL agent is particularly evident in complex systems in which the precise control of target variables, such as the speed of a motor, is required in the presence of disturbance variables [6].

The RL agent developed in this work is trained to maintain a stable speed for a DC motor, even in the event of changes to the target specifications or the presence of external influences. It is common for conventional PI controllers to respond to such conditions with overshoot or prolonged settling times, which is not a viable solution in certain applications. The use of the A2C algorithm enables the RL agent to learn to minimize overshoots and to compensate for disturbances in a more efficient manner. This capacity for adaptability confers a significant advantage upon the RL approach in comparison to classical controllers, thereby underscoring the importance of this work [4].

1.2. Objectives of the Approach

The overarching goal of this study is to explore the potential of reinforcement learning for enhancing dynamic control systems within the broader context of AI applications in mechatronic product development and production. A visual representation of this general goal is shown in **Figure 1**, which illustrates the potential deployment of AI across all phases of the product development lifecycle.

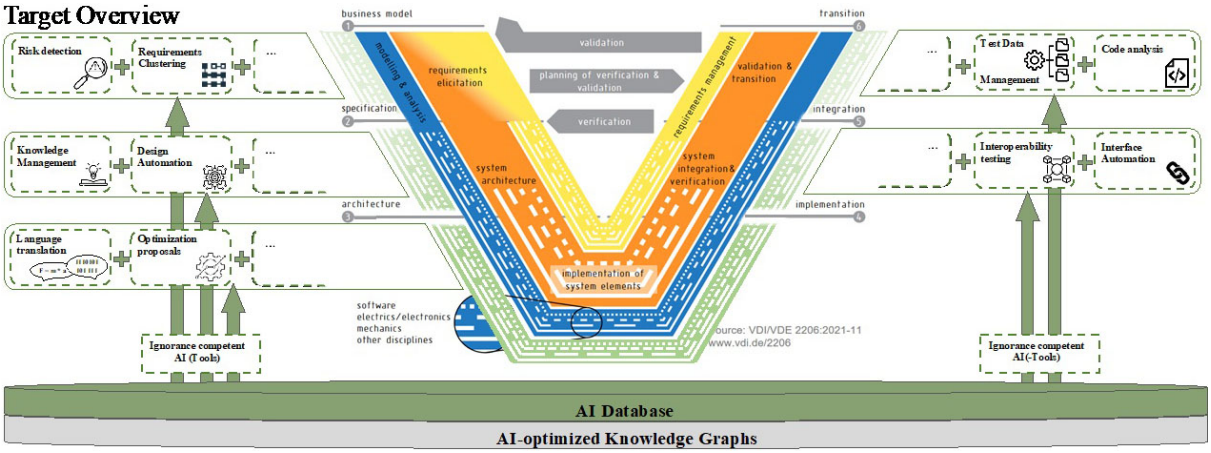

Figure 1. The V-cycle according to VDI 2206 and extended by the visualization of the possible AI support through the author [7].

The graphic illustrates the six principal phases of product development, which are aligned with the VDI/VDE 2206 standard [7]. The utilization of artificial intelligence tools, supported by a knowledge graph architecture, enables the implementation of functionalities such as risk detection, optimization proposals, design automation, and interoperability testing. Such tools facilitate the provision of context-aware and scalable solutions that span disparate disciplines, including software, mechanics, and electronics.

The methodology proposed in this study is primarily situated between Phase 3 ("System Architecture") and Phase 5 ("System Integration and Verification"). During this critical transition from architecture definition to implementation and validation, reinforcement learning serves as a dynamic tool to optimize system-level performance. Specifically, the developed RL-based controller complements traditional design workflows by:

- Automating optimization processes during system modelling.
- Facilitating interface automation through adaptable control strategies.
- Enhancing testing and verification workflows by enabling dynamic responses to varying environmental conditions.

The study focuses on this pivotal stage of the product development lifecycle with the objective of illustrating how reinforcement learning can address specific challenges in control engineering, including adaptability, stability, and fault tolerance. This approach has the potential to advance the state of the art in mechatronic system development.

Furthermore, the objective is to utilize Design of Experiments (DoE) to establish a structured test environment, thereby optimizing the RL agent in a range of test scenarios in future work. The application of DoE enables the RL agent to be trained in a manner that ensures consistent control performance under varying conditions, thereby enhancing its flexibility in more complex environments [8] [9].

2. Theoretical Background

In order to gain insight into the control of a DC motor with the help of reinforcement learning, it is essential to first consider the fundamental theoretical principles that underpin this field of study. This incorporates an examination of the functionality and structure of a DC motor, in addition to an investigation of the fundamental principles of control engineering and machine learning, with a particular focus on reinforcement learning and artificial neural networks (ANN). The following chapters provide an overview of the relevant concepts that are necessary for a comprehensive understanding of this work.

2.1. DC Motors: Functionality and Design

A direct current motor is a common electrical drive that converts electrical energy into mechanical energy. Due to its straightforward control mechanisms and high efficiency, it is employed in a multitude of applications, including those within the automotive industry, robotics, and automation technology.

2.1.1. Operating Principle

The DC motor operates on the fundamental principle of the Lorentz force, which acts upon a conductor carrying an electric current within a magnetic field. The generation of a magnetic field by an electric current flowing through the windings of the motor results in the exertion of torque on the rotor, whereby the magnetic field interacts with the static magnetic field of the motor. This torque causes the rotor to rotate, thereby converting electrical energy into mechanical work [10]-

[12].

To better visualize the basic functioning of a DC motor, Figure 2 shows a simplified representation of a conductor rod in a magnetic field, which illustrates the principles of the force effect on the current-carrying conductor.

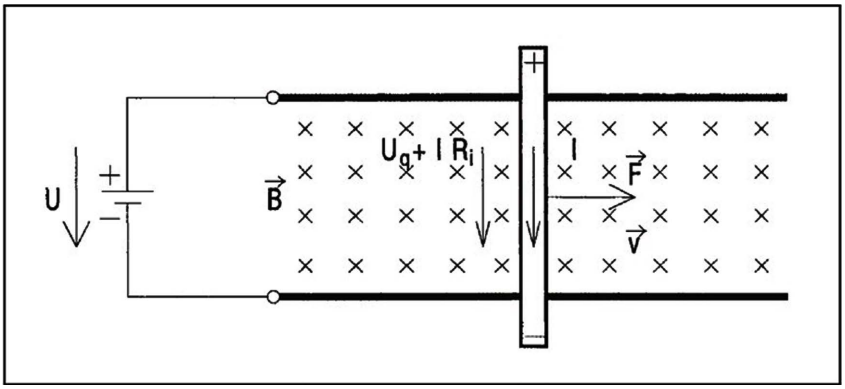

Figure 2. Conductor rod in a constant magnetic field [13].

In this context:

- U : Applied voltage to the conductor rod, enabling the current flow.
- $\vec{B}$: Magnetic flux density representing the constant magnetic field.
- I : Electric current flowing through the conductor rod, following the conventional current direction from top to bottom.
- R_i : Internal resistance of the conductor rod, affecting the current flow.
- $\vec{F}$: Lorentz force resulting from the interaction between the current and the magnetic field, accelerating the conductor rod to the right.
- $\vec{v}$: Velocity of the conductor rod, caused by the Lorentz force.
- U_q : Induced voltage in the conductor rod due to its motion in the magnetic field.

2.1.2. Structure

A typical DC motor consists of the following main components:

- **Stator**: The stationary part of the motor that generates a constant magnetic field.
- **Rotor (armature)**: The rotating part of the motor that carries current through the windings and generates torque.
- **Commutator**: A mechanical switch that ensures that the current flow through the windings of the rotor is periodically reversed to ensure continuous rotation.
- **Brushes**: These transmit the electrical current from an external power source to the rotor.

A significant attribute of a DC motor is its capacity to regulate speed with minimal effort, solely through the application of an appropriate voltage. An increase in voltage results in an acceleration of the rotational speed of the motor. This direct correlation between voltage and speed renders DC motors a favored option for precision control applications.

Figure 3 shows the rotational movement of a conductor loop in a magnetic field, which illustrates the principle of changing the direction of current in a DC motor. This mechanical switching movement forms the basis for the function of a commutator in the DC motor.

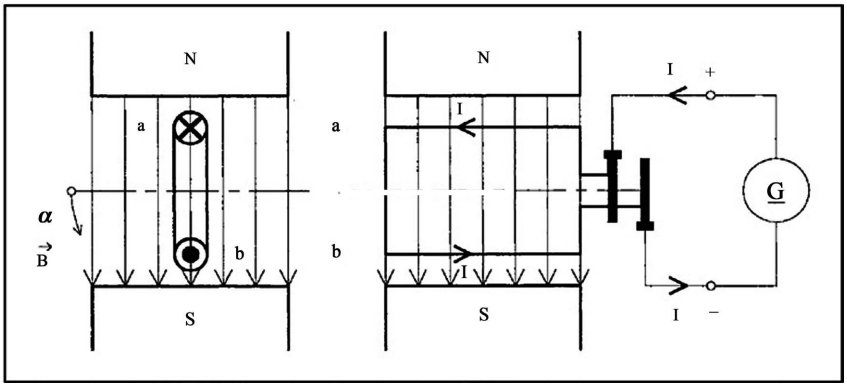

Figure 3. Rotatable conductor loop in the magnetic field [13].

In this context:

- $\vec{B}$: Magnetic flux density representing the uniform and constant magnetic field acting perpendicularly to the loop.
- a, b : Electric current flowing through the conductor rod, following the conventional current direction from top to bottom.
- I : Electric current flowing through the conductor loop, which is supplied by a constant voltage source via the slip rings.
- α : The angle of rotation of the conductor loop about its axis of symmetry. The diagram shows the loop in its initial position where $\alpha = 0$.

2.1.3. Motor Data

In this work, the maxon RE 65, 353297 **(24 ma)** is used as an example of a DC motor. The relevant motor data are shown in Table 1 and form the basis for modelling the motor as part of the simulations.

Table 1. List of the relevant parameters of the motor.

Parameter/Data	Value
Nominal voltage	48 V
Nominal speed	3420 min^{-1}
Nominal current (max. continuous current)	6.8 A
Terminal resistance	0.365 Ω
Terminal inductance	0.161 mH
Torque constant	123 $mN{\cdot}m{\cdot}A^{-1}$
Speed constant	77.8 $min^{-1}{\cdot}V^{-1}$
Rotor inertia	1.340 $g{\cdot}cm^2$

2.2. Control Theory

The field of control engineering is concerned with the control of dynamic systems, with the objective of achieving desired outcomes by selecting input variables in a manner that aligns the system behavior with the desired specifications. In the field of automation technology, controllers such as the proportional-integral-derivative (PID) controller and the PI controller are commonly employed for the control of systems including motors.

Proportional-Integral (PI) Controller

The PI controller is a simplified version of the PID controller in which the D component (derivative) is omitted. This simplifies the calculation but may result in a reduction in the degree of control that can be achieved in dynamic systems. The PI controller selects the control variable proportional to the control error and its integrated value. This implies that the controller responds to both the present error (proportional component) and the accumulated error (integral component). This guarantees a swift response to alterations in the control error, while the integrator component guarantees a reduction in the steady-state error.

In the field of DC motor control, the PI (proportional-integral) controller is utilized to stabilize the speed of a motor and set it to a desired value. The PI controller's principal advantage is its simplicity; however, adjustments to the control parameters are necessary to ensure optimal performance. Such adjustments necessitate a certain degree of expertise and must be adapted on a case-by-case basis, contingent on the specific environmental context [14].

3. Modeling of a DC Motor

Modeling a DC motor is an essential step to enable accurate simulation and control. This chapter explains the mathematical representation of the motor using the state-space model, more specifically a Linear Time Invariant (LTI) model, followed by the implementation of the simulation in the Python programming language.

3.1. State-Space Representation

A state-space representation is a mathematical model that describes the behavior of dynamic systems. In the case of the DC motor, the dynamic equations, which represent the relationship between the input variables (voltage) and the output variables (angular velocity, current), are converted into state variables. The state variables describe the internal state of the system at a specific point in time.

Figure 4 shows the rotational movement of a conductor loop in a magnetic field, which illustrates the principle of changing the direction of current in a DC motor. This mechanical switching movement forms the basis for the function of a commutator in the DC motor.

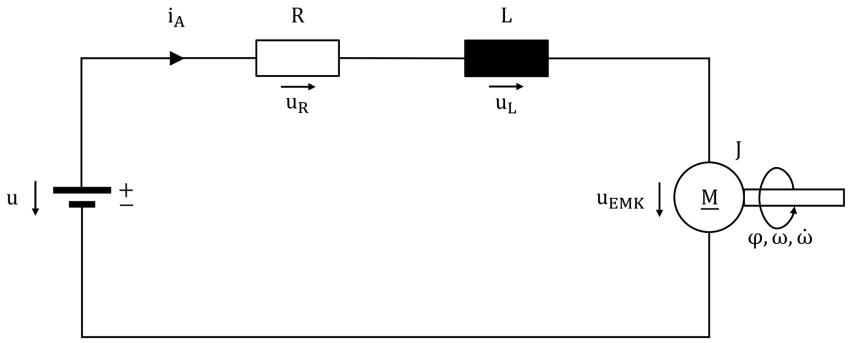

Figure 4. Physical equivalent circuit diagram of a direct current machine.

The DC motor can be described by the state-space representation:

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{A} \cdot \boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{B} \cdot \boldsymbol{u}(\boldsymbol{t})$$

$$\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{C} \cdot \boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{D} \cdot \boldsymbol{u}(\boldsymbol{t})$$

Where:

- $x(t)$ | *is the state vector* (*e.g. angular velocity*, *current*).
- $u(t)$ | *is the input vector* (*e.g. applied voltage*).
- *A*, *B*, *C*, *D* | *are state-space matrices that describe the dynamic behavior of the motor; with A*: *system matrix*, *B*: *input matrix*, *C*: *output matrix*, *D*: *feed-through matrix*.
- $y(t)$ | *is the output vector* (*e.g. measured angular velocity*).

This representation allows for the straightforward description of the motor's dynamic behavior in matrix form, thereby facilitating the design of simulations and controls based on this description [15]-[17].

3.2. Simulation in Python

The DC motor simulation was implemented in Python, a robust and widely utilized programming language for scientific calculations. The simulation is based on the numerical integration of the state-space presentation described in the preceding section. In order to ensure the most efficient calculation process, the NumPy and SciPy libraries were employed to facilitate matrix operations and to numerically solve the differential equations [18].

The initial stage of the simulation involved the implementation of the mathematical model representing the motor. The system matrices A and B were populated with parameters including resistance, inductance, and the motor's moment of inertia. Subsequently, a discrete-time model of the motor was constructed for the purpose of simulating its behavior under different input voltage conditions.

The selection of a discrete-time model is predicated on the observation that in practice, real sensors and actuators frequently operate with fixed sampling rates. The process of discretization enables a more realistic mapping of the environment. A sampling rate of 0.05 seconds was selected for this simulation, which corresponds to a frequency of 20 Hz. This provides an optimal balance between computational complexity and accuracy. A discrete-time model enables the realistic

simulation of the controller's behavior, thereby providing a robust foundation for subsequent reinforcement learning agent training [19].

The simulation results include both the step response of the motor and the reaction to a specified reference trajectory. The results presented here serve as the basis for subsequent analysis and comparison between the PI controller and the RL agent.

4. Methodology

The methodology of this study outlines the development and evaluation of two distinct control strategies for a DC motor: a conventional PI controller and a RL-based agent. The PI controller, widely recognized for its simplicity and reliability in industrial applications, serves as the baseline for comparison. Its performance provides a benchmark to assess the potential advantages of RL techniques, which promise enhanced adaptability and robustness in dynamic environments.

This chapter first details the design and implementation of the PI controller, focusing on parameter selection and performance evaluation through simulations. Subsequently, the reinforcement learning methodology is introduced, including the algorithmic framework, training environment, and evaluation metrics. Together, these approaches enable a comprehensive comparison of traditional and modern control strategies, highlighting the strengths and limitations of each.

4.1. Conventional PI Controller as a Reference

The PI controller is frequently employed in industrial contexts, offering a straightforward and resilient solution to a multitude of control issues. A comparison with the reinforcement learning agent is employed to ascertain whether contemporary machine learning techniques offer advantages over conventional control strategies.

4.1.1. Design and Implementation

The proportional-integral controller is one of the most frequently used control structures in automation technology. The PI controller is made up of two components:

Proportional component (P component): This component of the controller responds in a proportional manner to the discrepancy between the desired set-point and the actual system value, or control deviation. A high proportional component ensures rapid system response but can also result in instability.

Integral component (I component): The integral component calculates the accumulated control deviation over time, thereby ensuring that the system no longer exhibits a steady-state deviation over the long term. This enhances the precision of the system, although it may result in a slight reduction in the system's responsiveness.

The controller equation for the PI controller is:

$$\boldsymbol{u}(\boldsymbol{t}) = \boldsymbol{K}_{\boldsymbol{p}} \cdot \boldsymbol{e}(\boldsymbol{t}) + \boldsymbol{K}_{\boldsymbol{i}} \cdot \int \boldsymbol{e}(\boldsymbol{t}) \boldsymbol{dt}$$

Where:

- $u(t)$ *is the controlled variable, e.g. voltage.*
- $e(t)$ *is the control error difference between the reference variable and controlled variable.*
- K_p *is the proportional gain factor.*
- K_i *is the integral gain factor.*

To implement the PI controller, the parameters K_p and K_i were selected based on the dynamic characteristics of the DC motor. The selection of these parameters is decisive for the control quality and is optimized by several simulations. The objectives for the parameter selection were a low steady-state control error, a fast response time, and a high stability of the system. These criteria make it possible to achieve the desired control behavior of the DC motor and ensure precise adaptation to the specified target speed [20].

4.1.2. Simulation Results

The simulation results of the PI controller show a fast response of the motor to setpoint changes with minimal overshoot. The controller was able to maintain the speed of the motor in a steady state.

The main results of the simulation include:

- **Steady-state behavior**: The PI controller was able to completely eliminate the steady-state control deviation, allowing the motor to reach the exact desired speed.
- **Dynamic behavior**: In the event of sudden changes in the target specifications, the PI controller was able to restore the target speed within a short time, minimizing overshoot.
- **Disturbance suppression**: The controller demonstrated a high degree of stability by effectively compensating for disturbances and quickly restoring the system to the desired setpoint.

The results demonstrate the potential of reinforcement learning to minimize overshoot and stabilize control, as well as to address the unpredictability of dynamic environments. This adaptability positions reinforcement learning as a crucial enabling technology for the transition from rigid, parameter-driven approaches to flexible, context-sensitive control strategies.

Moreover, these findings align with broader industry demands for adaptive and scalable control systems in applications such as robotics, autonomous vehicles, and smart manufacturing. By demonstrating stable performance across a range of scenarios, the proposed approach provides a foundation for integrating reinforcement learning into real-world systems with diverse operational requirements.

4.2. Reinforcement Learning as a Controller

Reinforcement learning is a method by which agents can learn optimal control strategies through interaction with their environment and receipt of feedback in the form of rewards or penalties. This work employs the Advantage Actor-Critic algorithm, which integrates artificial neural networks to approximate value functions

and optimize control policies. The dual architecture of the algorithm, which separates the decision-making process (Actor) from the evaluation process (Critic), enables dynamic adaptation to environmental changes. Furthermore, the combination of RL and ANN enables the system to handle high-dimensional states and complex control tasks, which is difficult to achieve with traditional methods [21] [22].

The following section outlines the implementation details, including the training process and reward function [6].

The A2C algorithm splits the learning process into two main components: the Actor, which selects actions, and the Critic, which evaluates these actions. This division enables more precise control and more efficient adaptation of the control strategy.

- **Actor**: The Actor makes the control decisions by selecting the optimal voltage based on the current state of the system to influence the angular velocity ω. The Actor continuously adapts its strategy to minimize the deviation from the target speed ω_{target}.
- **Critic**: The Critic evaluates each action of the Actor by calculating the Advantage of a particular action compared to the expected performance. The advantage represents the difference between the actual reward and the expected estimated reward and provides information on how effective an action is. This process supports the actor in the selection of future actions that contribute to the optimization of the control quality.

Figure 5 shows the control structure of the training process for the RL agent. This figure shows how the RL agent perceives the environment, what feedback (reward R_t and state S_t) it receives, and how it influences the motor through the normalized manipulated variables. The control is performed by the agent, while the critic evaluates the feedback and determines the learning progress.

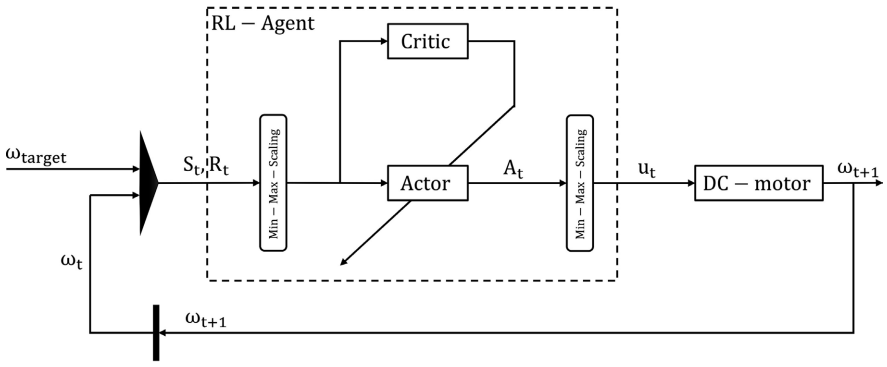

Figure 5. Control structure for training the RL agent.

The RL agent uses artificial neural networks to approximate both the actor's policy and the critic's value function. This architecture allows the RL agent to handle complex control tasks such as controlling a DC motor. The inputs to the neural network consist of the current states of the system (e.g. angular velocity, current),

while the outputs represent a Gaussian probability density function for possible control actions for different control actions.

The advantage of the A2C algorithm is that the actor and the critic are trained simultaneously. This leads to a more stable and efficient training process, especially for continuous control tasks such as the control of a motor [23] [24].

4.3. Training the RL Agent

The training process takes place in a simulated environment that reproduces the behavior of the DC motor. The RL agent runs through several training cycles in which it selects the voltage as the action and receives the current angular velocity ω as feedback. This feedback loop allows the agent to get to know the behavior of the motor and adapt the control strategy to minimize the control error of the angular velocity $\Delta\omega$ with $\Delta\omega = \omega_{\text{target}} - \omega$.

An essential part of the training process is the min-max scaling of the state values that serve as inputs to the neural network. The scaling normalizes the values of ω and i to reduce fluctuations and create a stable basis for training. This normalization makes the neural network more robust against extreme input values, which improves the stability and efficiency of the learning process.

The training process is illustrated in **Figure 6**, which shows the program flow chart according to DIN 66001.

The initial state (S_0) is defined first and the agent receives feedback in the form of a reward value (R_0) based on their action (A_0). This value indicates how well the agent has reached the desired angular velocity. In each step, the critic evaluates the agent's action and calculates the advantage, which is used to update the neural network.

The A2C algorithm combines the advantages of Temporal Difference Learning (TD Learning) and the Policy Gradient method. Here, the actor learns the strategy using the Policy Gradient method and executes new actions accordingly during training. In contrast, the critic evaluates the current state using the value function forced by the actor's actions. Similar to the actor, the critic learns the value function through TD-Learning. The goal, identical to TD-Learning, is to apply the actor-critic method to continuous problems. For a continuous domain, it is a common idea to choose a Gaussian probability density function for the actor's strategy π:

$$\pi(a \mid s,\theta) = \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} exp(-\frac{\left(a-\mu(s,\theta)\right)^2}{2\sigma(s,\theta)^2}$$

The advantage function A_t (not to be mixed up with the action A_t) is characterized by the adaptation to the baseline v (S_t, w_t) in order to limit the variance for the weighting of the gradients (w are the weights of the neural network of the critic):

$$A_t = R_{t+1} + \gamma v(S_{t+1}, w_t) - v(S_t, w_t)$$

This function is necessary for updating the weights θ of the actor's neural network (with a specific learning rate α):

$$\theta_{t+1} = \theta_t + \alpha A_t \nabla_\theta \left[\ln\left(\pi\left(A_t \mid S_t, \theta_t\right)\right)\right]$$

To update the weights of the critic, the following equation must be implemented (with a certain learning rate α):

$$w_{t+1} = w_t + \alpha A_t \nabla_w \left[v\left(S_t, w_t\right)\right]$$

The training process is comprised of numerous episodes, during which the agent develops an enhanced control strategy through the provision of continuous feedback. By means of repeated simulations, the RL agent is able to discern which actions are optimal for motor control, thereby reducing the discrepancy between the actual and desired speed [25] [26].

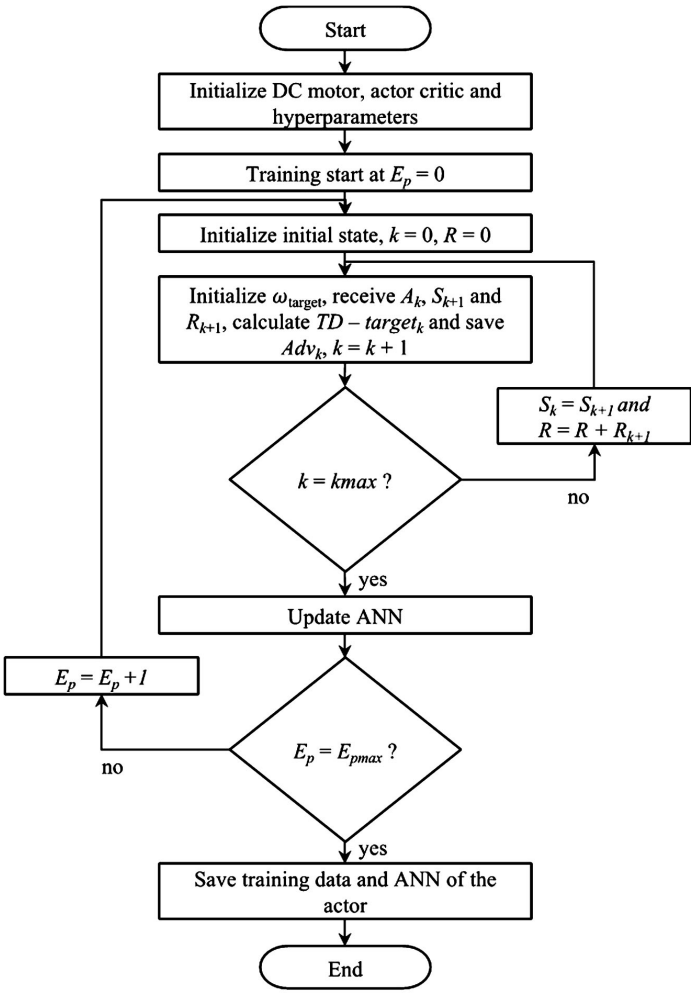

Figure 6. Program flow chart according to DIN 66001 of the RL agent.

4.4. Reward Function

The reward function plays a pivotal role in the training process, as it serves to motivate the RL agent to reinforce specific behaviors and suppress others. In this instance, the objective of the reward function is to minimize the discrepancy between the actual and target speeds, which serves as an indicator of the quality of

control. The mathematical formulation of the reward function, established for this particular application, is as follows:

$$\boldsymbol{R} = \boldsymbol{c} \cdot \left(\boldsymbol{\omega}_{\boldsymbol{target}} - \boldsymbol{\omega} \right)^2$$

Here, c is a penalty value that determines the extent to which deviations from the target velocity are penalized [27] [28].

4.5. Optimization and Performance Analysis of the RL Agent

The successful training of a reinforcement learning agent heavily depends on the careful selection of hyperparameters and training settings. These parameters govern the agent's learning process, influencing both its efficiency and effectiveness in achieving optimal control performance. Setting appropriate values for key parameters, such as the learning rate, discount factor, and episode count, is essential to ensure stability and convergence during training [29] [30].

This section outlines the chosen hyperparameters, explains their significance, and discusses how they were optimized to suit the dynamic control requirements of the DC motor system. Table 2 shows the most important hyperparameters and their values.

Table 2. Overview of hyperparameters and values.

Hyperparameter	Value	Meaning
Learning rate Actor	0.00006	Adaption speed of the Actor
Learning rate Critic	0.00025	Adaption speed of the Critic
Discount factor	0.5	Weight of future rewards
Number of episodes	2000	Number of training episodes
Penalty value	−0.95	Penalty for deviation from the rule

The learning rate of the actor and the critic determine the rate of adaptation of the control strategy employed by the agent. An excessive value may result in erratic behavior, whereas an insufficient value may impede the training process. The discount factor determines the relative weight given to future rewards, enabling the agent to consider both short-term and long-term objectives. The value of c determines the degree of penalty applied to deviations from the target speed. An elevated value for c engenders a heightened focus on precise tracking on the part of the agent, whereas a diminished value permits greater flexibility. By modifying the structure in this manner, the agent is able to learn to minimize the control error $\Delta\omega$ while maintaining stable control.

The simulation results show that after several training cycles, the RL agent can control the DC motor efficiently and achieve sufficient control quality. Important observations from the simulations include:

- **Adaptability**: The RL agent demonstrated enhanced capacity for adaptation to diverse operational scenarios throughout the training period. Although the selection of voltage at the outset of the training period was uncertain, the RL

agent was able to make increasingly stable decisions and achieve the angular velocity as the training progressed.

- **Reduction in control deviation**: As the training period increased, the RL agent reduced the deviation between target and actual speed. This shows that the agent optimized the control strategy and continuously improved the control quality, which is also reflected in the increasing reward values.
- **Stability**: At the end of the training, the RL agent showed a control deviation of less than 10%, even with nominal voltages. The agent shows a sufficient dynamic behavior regarding step responses and was able to minimize overshoots.

These findings underscore the potential of reinforcement learning for the control of dynamic systems, such as the DC motor. The RL agent was able to achieve the desired outcomes and respond effectively to external influences, suggesting that it may offer a viable alternative to traditional control methods.

4.6. Training Results, Challenges and Observations

The training process of the RL agent presented several challenges that influenced the learning progress. The main challenges include:

- **Stability issues**: Fluctuations in rule performance occurred at the beginning of training, which were addressed by adjusting the hyperparameters, particularly the learning rate and discount factor. These adjustments improved the stability of the learning process and led to more consistent results.
- **Fine-tuning the reward function**: The choice of the penalty value c proved to be critical for balancing precision and flexibility. Excessive punishment of large deviations initially led to overfitting, where the actor initially behaves exploratively before developing an optimized control strategy. By fine-tuning the value c, these effects could be corrected.
- **Slow learning curve**: In the initial training phases, the agent's learning progress was slower than expected. Increasing the number of episodes and fine-tuning the hyperparameters improved the control quality and accelerated the learning process.

Figure 7 illustrates that during the preliminary training stages, the agent's advancement was less rapid than anticipated. This was attributed to suboptimal initial hyperparameter configurations and the intrinsic exploration phase necessitated by reinforcement learning. During these initial stages, the agent demonstrated erratic behavior, at times selecting actions that deviated considerably from optimal control strategies. By increasing the number of training episodes and refining key hyperparameters, such as the learning rate, discount factor, and reward function penalties, the agent was able to gradually stabilize its learning process. These adjustments not only improved the control quality but also accelerated convergence, enabling the agent to adapt more effectively to dynamic environmental conditions. These results highlight the critical role of hyperparameter tuning in reinforcement learning, particularly in applications where precise and robust control is essential.

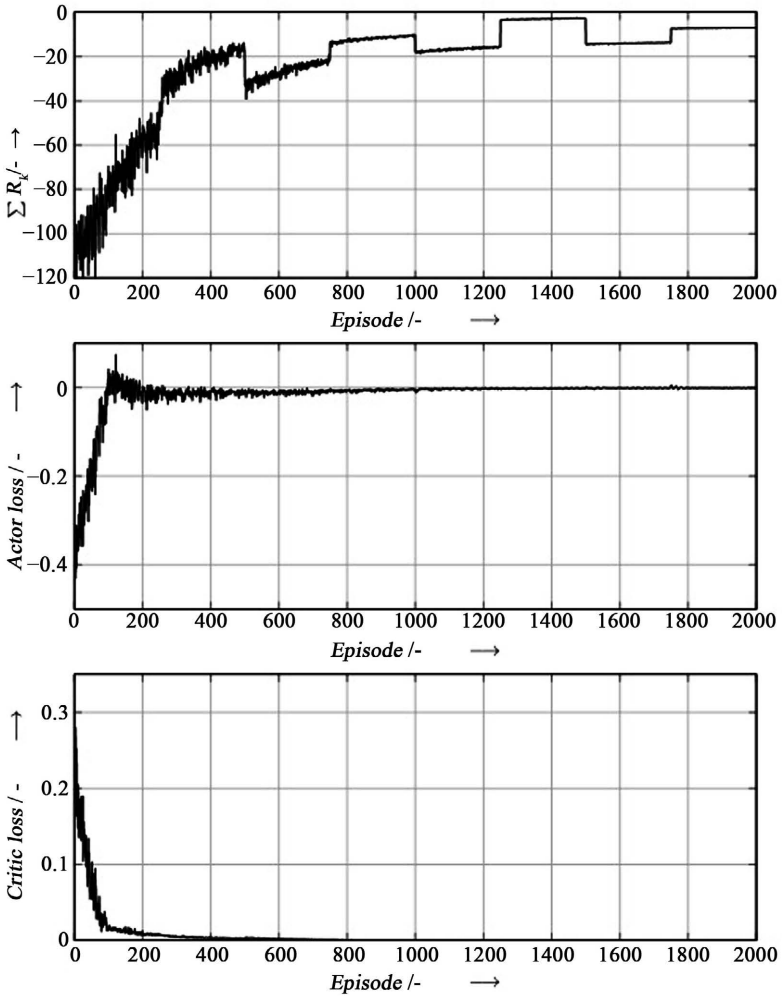

Figure 7. Training progress of the RL agent.

5. Research Results

This chapter presents a comparative analysis of the performance of the reinforcement learning agent (RL agent) and the conventional PI controller. The comparison is based on simulations in which both controllers are tested for different input signals and disturbances. The objective of this analysis is to examine the behavior of the two control strategies in different scenarios and to identify the advantages offered by reinforcement learning compared to classical control.

5.1. Step-Shaped Input Signal

In the initial simulation, an abrupt input signal was employed. This signal simulates a sudden change in the target speed of the motor from 0, which is a common occurrence in real-world applications where rapid motor responses are necessary. The simulation demonstrates the response of the RL agent and the PI controller to step responses.

- **RL agent**: The RL agent reacts with a time delay to the change in the input signal. There is no significant overshoot—stability is ensured in steady-state

operation.

- **PI controller**: The PI controller reacts very quickly to the sudden change in target speed. It achieves the specification precisely in the steady state by fully compensating for the control deviation. During the transition phase, however, a clear overshoot is noticeable, which only stabilizes after several oscillations.

Figure 8 shows the results of the step response for both controllers. The curve of the PI controller shows a faster initial response, while the RL agent is characterized by a more stable but slightly delayed adaptation.

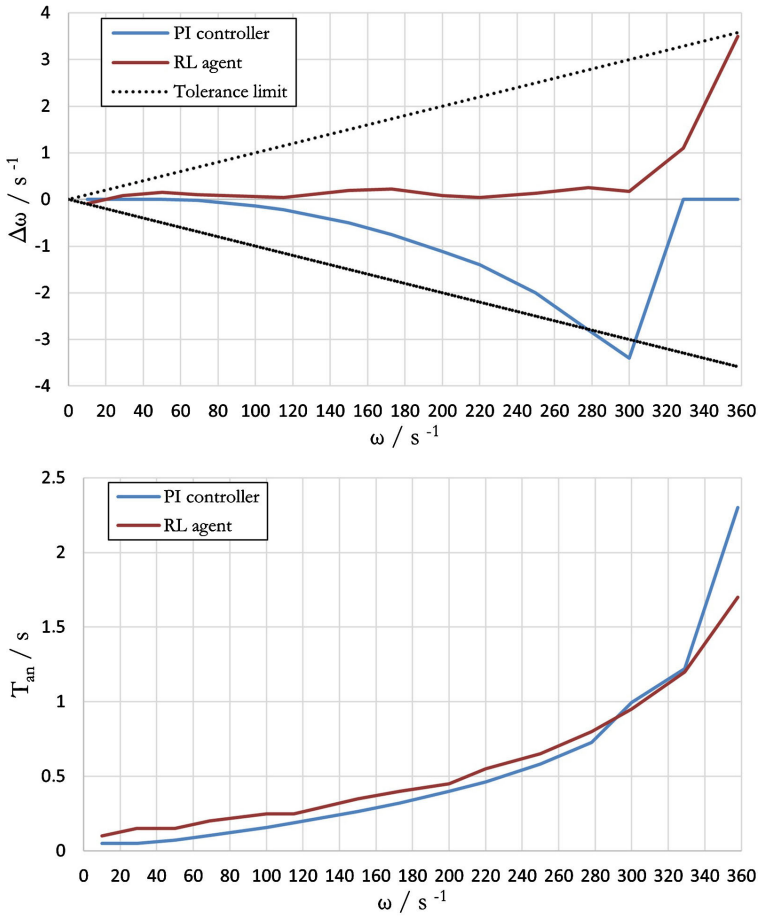

Figure 8. Results of the step responses.

5.2. Reference Trajectory

In the second simulation, a reference trajectory is used that consists of several gradual changes in the target velocity. This trajectory simulates a scenario in which the angular velocity of the motor is changed at fixed intervals to test the reaction and adaptability of both controllers to a dynamic input variable.

- **RL agent**: The RL agent shows adaptability to the gradual changes in the target speed. With each change, the RL agent adapts and can follow the trajectory with a time delay. It should be emphasized that there is no significant overshoot.

- **PI controller**: In this scenario, the PI controller also reacts quickly to any change in the target speed. However, an overshoot is noticeable with each stepwise adjustment, which impairs the stability of the system. Although the controller reaches the target speed after a certain time, the repeated fluctuations lead to reduced control quality. In applications with dynamic requirements, this behavior can be inefficient as it delays the stabilization of the system.

Figure 9 shows the simulation results for the reference trajectory. The difference in the control quality is particularly clear here: While the RL agent reaches the angular velocity with minimal deviations at each stage, the PI controller shows greater fluctuations with each change.

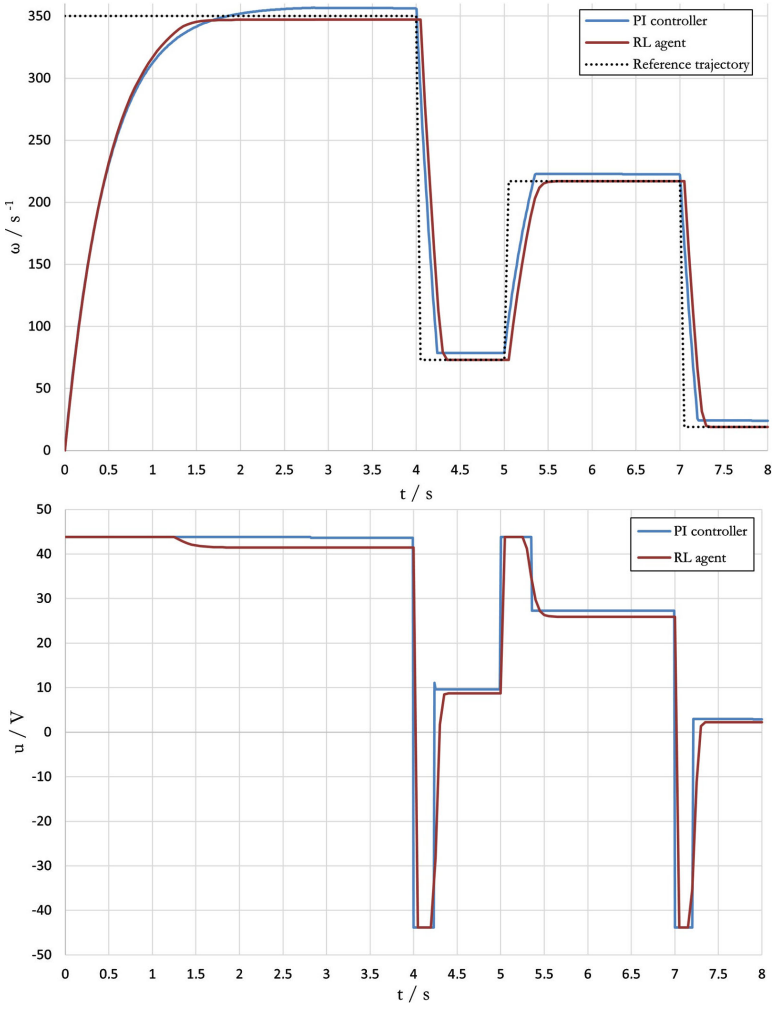

Figure 9. Simulation of the reference trajectory.

5.3. Comparison of the Control Strategies

The results of the simulations indicate that, in all scenarios considered, the RL agent exhibits sufficient control quality compared to the PI controller. The results demonstrate that the RL agent exhibits superior stability and precision, particularly in dynamic and complex environments. The discrepancies between the two

control strategies can be encapsulated as follows:

- **Response time**: The PI controller demonstrates a more rapid response to alterations in the angular velocity, achieving precise stabilization in the steady state. However, this rapid response is frequently accompanied by a considerable overshoot during the transition phases, which compromises the stability of the system. In comparison, the RL agent exhibits a somewhat slower reaction time but provides a markedly more stable adaptation, as it completely avoids overshoot. This characteristic renders it especially well-suited to applications where fluctuations are a significant issue.
- **Stability**: The RL agent demonstrates superior stability across all scenarios, exhibiting a capacity to act without overshooting and to adapt with precision to designated targets. This is particularly advantageous in scenarios involving gradual changes or dynamic requirements, as the RL agent develops a robust control strategy due to its continuous learning ability, which yields reliable results even in complex environments. In contrast, the PI controller demonstrates deficiencies in stability, particularly in scenarios characterized by frequent changes. However, this could be countered by changes to the draft regulation.
- **Adaptability**: A principal benefit of the RL agent is its capacity to adaptively respond to evolving circumstances and external disturbances. By continuously optimizing its control strategy, the RL agent is able to achieve a high level of precision even when faced with complex trajectories. In contrast, the PI controller is reliant on fixed parameters, which renders it susceptible to deficiencies in the event of unforeseen alterations. Its inability to adapt flexibly to novel conditions represents a notable limitation.

The advantages of reinforcement learning observed in this study are particularly relevant for applications where precision, adaptability, and fault tolerance are of paramount importance. For example, the capacity to adapt control strategies in real-time, without excessive overshoot, makes reinforcement learning an optimal choice for precision robotics, where minor discrepancies can result in substantial operational errors.

Similarly, industries such as aerospace and medical technology could benefit from the RL agent's capability to maintain stability under unexpected disturbances, as these fields often face stringent safety and reliability requirements. By situating the reinforcement learning methodology within this broader industrial context, the study demonstrates its potential for advancing control engineering.

In conclusion, it can be stated that reinforcement learning, as applied to the RL agent, represents a promising alternative to traditional control strategies such as the PI controller. In scenarios where stability, adaptability, and precision are required, the RL agent has advantages, although improvements to the controller would again improve these. The capacity of the RL agent to operate without overshooting and with consistently high control quality makes it particularly well-suited to dynamic and complex applications.

Potential for Optimization through Design of Experiments

One potential approach for further enhancing the performance of the RL agent is the incorporation of Design of Experiments. A DoE provides a systematic methodology for the specific testing of various scenarios and the investigation of the influence of variables, such as changes in target speed or the presence of external disturbances. By varying these parameters in a structured manner, the learning ability and adaptability of the RL agent may be enhanced [31].

The DoE approach may be employed to devise the parameters and training environments in a manner that enables the RL agent to be evaluated and optimized across a range of conditions. This would facilitate the enhancement of the agent's resilience against diverse operational scenarios, thereby augmenting its performance in intricate and dynamic environments. The systematic implementation of Design of Experiments could, therefore, enhance the long-term adaptability of the RL agent, thereby ensuring more stable and efficient control even in challenging environments [32].

6. Discussion and Conclusion

The incorporation of reinforcement learning into the mechatronic product development lifecycle presents a promising avenue for addressing challenges such as adaptability, robustness, and complexity. While this study focuses on the application of reinforcement learning within a holistic AI-driven framework, the proposed methodology has demonstrated particular value between Phase 3 ("System Architecture") and Phase 5 ("System Integration and Verification").

In these phases, reinforcement learning has proven to be a valuable tool for automating optimization processes, enabling adaptive interface control, and supporting verification tasks through dynamic learning capabilities. However, reinforcement learning is only one of several promising techniques that could be employed at various stages of the development lifecycle. Its successful application in this study suggests that similar techniques could enhance other stages of the product development process, such as early-stage risk assessment or late-stage system validation. This flexibility demonstrates how reinforcement learning, when integrated with other AI methodologies, can facilitate the transition from conceptual design to implementation—a traditionally iterative process reliant on manual adjustments and static strategies.

Key insights from this work include:

- **Adaptability in dynamic environments**: The RL-based controller demonstrated superior performance compared to traditional PI controllers in scenarios necessitating real-time adaptation and resilience to external disturbances.
- **Scalability for system-level integration**: By leveraging AI tools such as knowledge graphs and optimized training pipelines, the RL agent demonstrated potential for integration into more complex product architectures.
- **Automation potential**: The deployment of RL within Phase 3 to Phase 5 can significantly reduce manual tuning efforts, improving the efficiency of system

design and validation processes.

Despite these advances, certain limitations persist. The computational overhead associated with reinforcement learning training and the challenge of transferring simulation-based learning to real-world applications highlight areas for future improvement. Incorporating alternative reinforcement learning algorithms or hybrid control systems could address these challenges by combining the strengths of both approaches, thereby enhancing the robustness and scalability of the methodology.

While the simulation results provide valuable insights into the potential of reinforcement learning for adaptive control, the absence of practical implementation and testing on actual hardware represents a significant limitation of this study. Real-world environments often present additional challenges, such as hardware imperfections, sensor noise, and latency, which may impact the performance of the proposed reinforcement learning controller.

Future work will address these limitations through hardware-in-the-loop (HiL) experiments to validate the applicability of the RL-based control strategy in real-world scenarios. These experiments will enable a comprehensive evaluation of the controller's robustness and adaptability under physical constraints. Additionally, integrating reinforcement learning into practical systems offers the opportunity to explore hybrid control strategies that combine the adaptability of reinforcement learning with the computational efficiency of traditional methods.

While the proposed framework provides substantial benefits, reinforcement learning itself presents challenges that merit further discussion:

- **Computational Complexity**: RL algorithms, such as A2C, require significant computational resources during training. This can limit their applicability in time-sensitive scenarios. Future work could explore more efficient algorithms or transfer learning approaches to mitigate these limitations.
- **Sensitivity to Hyperparameters**: The performance of RL agents is highly dependent on hyperparameter tuning. Automated techniques, such as Bayesian optimization, could streamline this process and improve training outcomes.
- **Generalization to Real-World Conditions**: RL agents trained in simulations may struggle to adapt to real-world environments due to discrepancies such as unmodeled dynamics or sensor noise. Domain adaptation and uncertainty-aware training could enhance their robustness.
- **Exploration-Exploitation Trade-Off**: The exploration required by RL algorithms can pose risks in safety-critical applications. Safe reinforcement learning approaches and reward shaping may mitigate this issue.

Despite these challenges, the simulated environment in this study was carefully designed to approximate real-world conditions, including dynamic disturbances and varying setpoints. As such, the results establish a strong foundation for future investigations, demonstrating the feasibility of reinforcement learning as a promising control strategy for mechatronic systems.

The method focuses on the utilization of reinforcement learning for system architecture and integration. However, its broader implications extend to the creation of a unified AI-driven development process. By integrating this methodology with

other AI techniques, such as supervised learning for anomaly detection in Phase 1 or generative design tools in Phase 2, it is possible to develop a cohesive framework for intelligent product development.

Collaborations with industry partners to test the RL agent in hardware-in-the-loop setups will be pivotal in advancing its adoption in industrial settings. These next steps will ensure that the proposed methodology not only enhances theoretical insights but also delivers tangible improvements in engineering practice.

7. Outlook

This study offers a detailed investigation into the potential of reinforcement learning to enhance the mechatronic product development lifecycle, with a particular emphasis on the pivotal stages of system architecture definition and integration. By aligning the methodology with the structured V-model development process, the work underscores the applicability of AI-based solutions to genuine engineering challenges in the real world.

Future research could expand upon these findings by:

1) Enhancing scalability through hybrid approaches: Combining RL with classical control methods or alternative algorithms (e.g., PPO, DDPG) could improve computational efficiency and reduce reliance on large training datasets.

2) Integrating domain-specific knowledge into the RL process: Future research could focus on embedding domain-specific knowledge, such as safety requirements or regulatory constraints, directly into the RL process using AI-optimized knowledge graphs or tailored reward functions. This approach would accelerate learning, ensure compliance with critical guidelines, and improve the transferability of RL solutions to real-world industrial applications.

3) Validating in real-world environments: Implementing the RL methodology in HiL or real-system setups could bridge the gap between simulation and application, further solidifying its practicality in industrial settings.

4) Leveraging Design of Experiments: Systematic exploration of training scenarios and environmental variables using DoE could improve the adaptability and robustness of the RL agent across diverse operational conditions.

By concentrating on these elements, future research can build upon the groundwork laid here, propelling the integration of AI into mechatronic system development and production. Ultimately, reinforcement learning has the potential to fundamentally alter the manner in which intelligent systems are designed, optimized, and deployed, thereby paving the way for a new era of adaptive and resilient engineering solutions.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Åström, K.J. and Hägglund, T. (2006) Advanced PID Control. ISA—The Instrumen-

tation, Systems, and Automation Society.

[2] Lloyds Raja, G. and Ali, A. (2021) New PI-PD Controller Design Strategy for Industrial Unstable and Integrating Processes with Dead Time and Inverse Response. *Journal of Control, Automation and Electrical Systems*, **32**, 266-280. https://doi.org/10.1007/s40313-020-00679-5

[3] Jiménez, G.A., de la Escalera Hueso, A. and Gómez-Silva, M.J. (2023) Reinforcement Learning Algorithms for Autonomous Mission Accomplishment by Unmanned Aerial Vehicles: A Comparative View with DQN, SARSA and A2C. *Sensors*, **23**, Article 9013. https://doi.org/10.3390/s23219013

[4] De La Fuente, N. and Guerra, D.A.V. (2024) A Comparative Study of Deep Reinforcement Learning Models: DQN vs PPO vs A2C. arXiv: 2407.14151.

[5] Vouros, G.A. (2023) Explainable Deep Reinforcement Learning: State of the Art and Challenges. arXiv: 2301.09937.

[6] Sutton, R.S. and Barto, A. (2020) Reinforcement Learning: An Introduction. Second Edition, The MIT Press.

[7] Gräßler, I. (2021) VDI/VDE 2206: Entwicklung mechatronischer und cyber-physischer Systeme. Inhaltsverzeichnis.

[8] Gatti, C. (2015) Design of Experiments for Reinforcement Learning. Springer.

[9] Ladosz, P., Weng, L., Kim, M. and Oh, H. (2022) Exploration in Deep Reinforcement Learning: A Survey. *Information Fusion*, **85**, 1-22. https://doi.org/10.1016/j.inffus.2022.03.003

[10] Fazdi, M.F. and Hsueh, P. (2023) Parameters Identification of a Permanent Magnet DC Motor: A Review. *Electronics*, **12**, Article 2559. https://doi.org/10.3390/electronics12122559

[11] Shah, R. and Sands, T. (2021) Comparing Methods of DC Motor Control for UUVs. *Applied Sciences*, **11**, Article 4972. https://doi.org/10.3390/app11114972

[12] Aribowo, W., Supari, S. and Suprianto, B. (2022) Optimization of PID Parameters for Controlling DC Motor Based on the Aquila Optimizer Algorithm. *International Journal of Power Electronics and Drive Systems* (*IJPEDS*), **13**, 216-222. https://doi.org/10.11591/ijpeds.v13.i1.pp216-222

[13] Spring, E. (2009) Elektrische Maschinen: Eine Einführung. Springer.

[14] Borase, R.P., Maghade, D.K., Sondkar, S.Y. and Pawar, S.N. (2020) A Review of PID Control, Tuning Methods and Applications. *International Journal of Dynamics and Control*, **9**, 818-827. https://doi.org/10.1007/s40435-020-00665-4

[15] Chau, P.C. (2002) Process Control. Cambridge University Press. https://doi.org/10.1017/cbo9780511813665

[16] Zhang, A., McAllister, R., Calandra, R., Gal, Y. and Levine, S. (2020) Learning Invariant Representations for Reinforcement Learning without Reconstruction. arXiv: 2006.10742.

[17] Schwarzer, M., Anand, A., Goel, R., Hjelm, R.D., Courville, A. and Bachman, P. (2020) Data-Efficient Reinforcement Learning with Self-Predictive Representations. arXiv: 2007.05929.

[18] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., *et al.* (2020) Scipy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, **17**, 261-272. https://doi.org/10.1038/s41592-019-0686-2

[19] Pinto, V., Gonçalves, J. and Costa, P. (2020) Modeling and Control of a DC Motor Coupled to a Non-Rigid Joint. *Applied System Innovation*, **3**, Article 24.

https://doi.org/10.3390/asi3020024

[20] Wang, Y. and Shao, H. (2000) Optimal Tuning for PI Controller. *Automatica*, **36**, 147-152. https://doi.org/10.1016/s0005-1098(99)00130-2

[21] Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., *et al.* (2024) Deep Reinforcement Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, **35**, 5064-5078. https://doi.org/10.1109/tnnls.2022.3207346

[22] Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., *et al.* (2022) Deep Learning, Reinforcement Learning, and World Models. *Neural Networks*, **152**, 267-275. https://doi.org/10.1016/j.neunet.2022.03.037

[23] Arulkumaran, K., Deisenroth, M.P., Brundage, M. and Bharath, A.A. (2017) Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*, **34**, 26-38. https://doi.org/10.1109/msp.2017.2743240

[24] Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S. (2018) Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv: 1801.01290.

[25] Zhang, K., Wang, Z., Chen, G., Zhang, L., Yang, Y., Yao, C., *et al.* (2022) Training Effective Deep Reinforcement Learning Agents for Real-Time Life-Cycle Production Optimization. *Journal of Petroleum Science and Engineering*, **208**, Article ID: 109766. https://doi.org/10.1016/j.petrol.2021.109766

[26] Laskin, M., *et al.* (2021) URLB: Unsupervised Reinforcement Learning Benchmark. arXiv: 2110.15191.

[27] Eschmann, J. (2021) Reward Function Design in Reinforcement Learning. In: Belousov, B., Abdulsamad, H., Klink, P., Parisi, S. and Peters, J., Eds., *Reinforcement Learning Algorithms*: *Analysis and Applications*, Springer, 25-33. https://doi.org/10.1007/978-3-030-41188-6_3

[28] Kwon, M., Xie, S.M., Bullard, K. and Sadigh, D. (2023) Reward Design with Language Models. arXiv: 2303.00001.

[29] Kiran, M. and Ozyildirim, M. (2022) Hyperparameter Tuning for Deep Reinforcement Learning Applications. arXiv: 2201.11182.

[30] Felten, F., Gareev, D., Talbi, E.G. and Danoy, G. (2023) Hyperparameter Optimization for Multi-Objective Reinforcement Learning. arXiv: 2310.16487.

[31] Dean, A., Voss, D. and Draguljić, D. (2017) Design and Analysis of Experiments. Springer.

[32] Nüssgen, A., *et al.* (2023) Intelligent Component Manufacturability Testing in Virtual Product Development. *Artificial Intelligence und Machine Learning in der CAE-Basierten Simulation*, München, 23-24 Oktober 2023, 14-22.