Find Optimal Suspension Kinematics Targets for Vehicle Dynamics Using Reinforcement Learning

ARTICLE INFO
Article ID: 10-10-01-0002

doi:10.4271/10-10-01-0002

Find Optimal Suspension Kinematics Targets for Vehicle Dynamics Using Reinforcement Learning

Yansong Huang,[1] Max Boerboom,[2] Krister Wolff,[3] and Bengt Jacobson[3]

[1]Chalmers University of Technology, Vehicle Engineering and Autonomous Systems, Sweden
[2]Volvo Car AB, Sweden
[3]Chalmers University of Technology, Sweden

Abstract

Setting up suspension kinematics targets has been a challenging task for vehicle engineers. The challenges involve a high-dimensional search space, nonlinear relationships between the suspension kinematics and vehicle dynamics, exploration and exploitation trade-offs, and the need for domain-specific knowledge. Traditional multi-objective optimization methods are time-consuming, sensitive to initial conditions, and rarely converge to the global optimum in high-dimensional spaces. This article explores how reinforcement learning can be used to automate the design of suspension kinematics targets, addressing a longstanding challenge in vehicle dynamics design: the inverse problem of satisfying high-level handling objectives through low-level subsystem parameters. The method is based on the accumulation of knowledge through the interaction between an intelligent agent and a simulation environment. The agent optimizes suspension kinematics targets by receiving rewards tied to vehicle dynamics performance. The agent, employing a Gaussian policy and σ-based sensitivity analysis, enables the identification of critical and non-critical design parameters. The results show that the proposed method can find optimal suspension kinematics targets with the help of accumulated knowledge. The knowledge-guided learning process demonstrates a novel approach to solving high-dimensional optimization problems, offering good convergence time and valuable results. The proposed method contributes to the field by using reinforcement learning to set up suspension kinematics targets in the automotive industry.

History

Received: 02 Jun 2025
Revised: 12 Sep 2025
Accepted: 23 Sep 2025
e-Available: 11 Oct 2025

Keywords

Passenger car, Wheel suspension, Kinematics, Reinforcement learning

Citation

Huang, Y., Boerboom, M., Wolff, K., and Jacobson, B., "Find Optimal Suspension Kinematics Targets for Vehicle Dynamics Using Reinforcement Learning," *SAE Int. J. Veh. Dyn., Stab., and NVH* 10(1):25-41, 2026, doi:10.4271/10-10-01-0002.

ISSN: 2380-2162
e-ISSN: 2380-2170

1. Introduction

1.1. Background and Research Problem

The objective of this study is to develop a reinforcement learning framework capable of automatically determining suspension kinematics targets that fulfill predefined complete vehicle-level dynamic requirements. To ensure adequate stress-testing, the framework was developed using a case study based on industry-relevant requirements and representative vehicle models. Suspension kinematics targets are set by the attributes of the vehicle. The majority of these attributes are determined by the vehicle dynamics, which are influenced by the suspension kinematics, among other vehicle subsystems. The suspension kinematics targets are set by vehicle engineers based on their experience and knowledge, and then given to the suspension engineer. With the help of optimization techniques [1, 2], finding the optimal suspension kinematics targets can be programmed as a computational task. In the domain of engineering design optimization applications, the most common methods, such as gradient-based techniques [3–7] and gradient-free methods, for example, genetic algorithms (GA) [8–12] and biologically inspired algorithms [13–16], are widely used. However, challenges such as convergence, exploration–exploitation balancing, and computational efficiency remain, especially in high-dimensional design spaces [17–19]. Inspired by human engineers, knowledge accumulated from previous experience plays a crucial role when a new design is proposed. With the early foundation in neural networks [20, 21], the knowledge represented in the weights of the network can be updated through interaction with the environment. This is the foundation of reinforcement learning (RL), which has shown promise in solving complex problems [22]. This study addresses the lack of efficient optimization methods for high-dimensional suspension kinematics design by leveraging the RL method to find optimal suspension targets. While the authors hypothesize that RL models can address the research problem, the goal of this article is to evaluate the feasibility and practical implications of using RL-based methods for searching high-dimensional suspension kinematics targets. The suspension target generation task presented here involves an inverse mapping from vehicle-level objectives to subsystem-level parameters, where RL's policy-based formulation and reusability provide a natural advantage, enabling faster adaptation as vehicle-level requirements evolve. However, we acknowledge the importance of benchmarking and plan to investigate quantitative comparisons, including convergence behavior and solution quality, in future work.

1.2. Motivation and Contributions

Motivating the RL algorithm versus traditional optimization methods, a deep argument provided by [23, 24] shows that RL offers a potential solution to the limitations of traditional optimization methods. The results indicate RL has the potential to solve complex engineering problems. RL's policy-based formulation and reusability provide an advantage over optimization techniques like GA in high-dimensional, sequentially coupled variables. The effect of one decision depends on previous decisions; the problem unfolds in steps rather than all at once. RL handles this well because it treats the problem as a sequence of state–action–reward steps, learning to navigate dependencies over time. GA treats the problem as a static "find the best set of parameters" task, so it doesn't naturally model sequential dependencies. In contrast to optimization techniques such as GA and gradient-based methods, the RL framework used in this study offers several qualitative advantages for suspension target generation. The learned policy can be visualized, providing interpretable insight into how the agent navigates the trade-offs between, for example, longitudinal and lateral dynamics. A trained policy can be adapted to related design tasks, reducing computation in iterative development cycles. Unlike methods that require predefined datasets, RL learns by interacting with the simulation, enabling exploration of non-obvious design configurations. While GA remains well-suited for certain nonlinear, non-smooth problems [25], RL's sequential learning and experience reuse make it a compelling alternative for complex, high-dimensional design spaces. Following recent successful domain applications using RL, it has become a popular machine learning technique used to solve control problems [26–29] and gaming strategies [30–32]. It has demonstrated its potential search capabilities in complex environments. In this article, we propose a search tool using RL to find optimal suspension kinematics targets. The research gap is addressing the increase in search dimensionality by using the RL method instead of multi-objective optimization. We hypothesize that the proposed RL-based method will address these limitations and is suitable for high-dimensional, nonlinear design spaces compared to traditional multi-objective optimization techniques. The σ-based policy enables identification of critical and non-critical design parameters. This is valuable in handling high-dimensional optimization problems and guiding further dimension reduction.

A modified actor-critic framework inspired by [33] is deployed for this learning task. The goal for the RL learning agent is to obtain higher rewards from a simulation environment. The reward is formulated in a way that leads the suspension kinematics targets to converge to their optimal values. A probabilistic model gives a better chance to learn the uncertainty of the environment. The environment in this article uses VI-CarRealTime [34] since

VI-CarRealTime provides a sufficiently accurate vehicle model type with acceptably low computation time. The model contains a look-up table-based suspension property file, which is characterized by the suspension kinematics targets. The complete vehicle can be simulated "offline" as a set of pre-defined driving scenarios or as a driver-in-the-loop (DiL) simulation. The DiL simulation can involve the human driver in the simulation loop, thereby evaluating subjective criteria. Offline simulation is used in this article without DiL evaluation, but real-time simulation is also of interest for potential performance evaluation. This article contributes a methodology for an actor-critic RL framework and a simulation environment from which the RL agent can learn. The method helps the vehicle dynamics engineer to find or narrow down the design space in the early vehicle development stages. Once the training is complete, the knowledge represented by the agent's value function can be reused in later design stages, for example, for minor suspension target adjustments. Any trained model can be reused as long as the architecture of the neural network remains the same. Furthermore, compared with optimization-based methods, the new method utilizes the knowledge accumulated during the training process. Therefore, the agent continues to improve with more training data and has the potential to be used in future tasks as a pre-trained model.

The first part of the article introduces a method that generates suspension property files that amount to the vehicle model. In the second part, the RL agent that interacts with the simulation environment and proposes new suspension kinematics targets is introduced. While limited to a single vehicle model, the study provides insight into the challenges and potential of RL in target generation for suspension systems. The third part of the article shows the results of a case study using the proposed method. The results show that the proposed method can find optimal suspension kinematics targets with the help of accumulated knowledge.

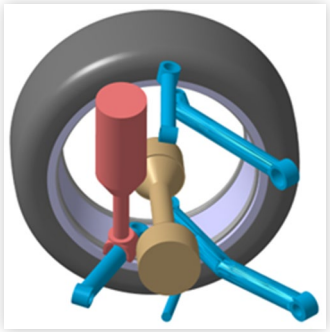
2. Simulation Environment

This section describes the suspension target selection, generating artificial suspension property files, and the RL environment.

2.1. Artificial Suspension Property File

2.1.1. Suspension Target Selection The selection of suspension kinematics targets in this article focuses on the targets that mainly influence longitudinal and lateral dynamic behaviors. An extended set of targets from [35] is used in this article. The targets are listed in Table 3. The targets are divided into three categories: front suspension jounce targets, front suspension steering targets, and rear suspension jounce targets. These targets are used to generate look-up table-based front and rear suspension property files. The kinematics targets used to generate artificial suspension property files do not guarantee a physically feasible suspension; they only guarantee a suspension that can be simulated in VI-CarRealTime. To ensure the feasibility of the suspension, a packaging check needs to be considered together with the kinematics targets setup. Figure 1 shows a physical suspension using the selected kinematics targets with a feasible packaging solution [36].

FIGURE 1 Physical suspension considers packaging feasibility.



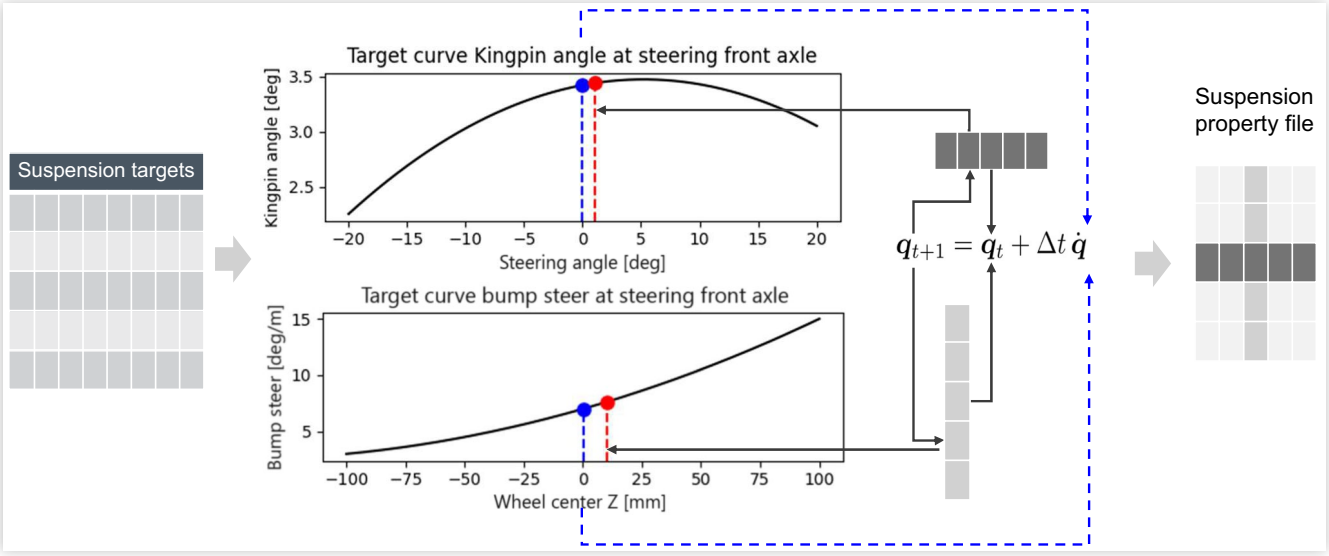
2.1.2. Generating Artificial Suspension Property Files The suspension property files utilize the targets from Section 2.1.1 to generate curve-based steering subsystems for the front axle and curve-based suspension subsystems for the rear axle. The targets for the front steering subsystem use 3D splines to capture the motion of the steering knuckle. For the rear axle, 2D splines are used to capture the motion of the non-steering knuckle. The motion includes steer at ground, camber angle, side view angle, X-coordinate variation, and Y-coordinate variation. The motion in the front axle depends on rack travel and jounce travel. The motion in the rear axle depends only on jounce travel.[1] A summary of the motion properties and corresponding input is shown in Table A.1.

To get a proper motion file, the first step is to calculate the general motion from the given suspension kinematics targets. The concept of general motions $\dot{q}$ is described in [37]. The relation between the general motion and the suspension kinematics targets is described in Section 3.1.1 [35] for a front axle and Section 3 [37] for a rear axle. At a given jounce position u, the targets are given by reading the target curve. For example, the target curve bump steer is shown in Equation 1.

$$\text{Bump Steer}(u) = \text{1st Bump Steer} + \text{2nd Bump Steer} \cdot u + \text{3rd Bump Steer} \cdot u^2 \quad (1)$$

[1] Jounce travel refers to the wheel center height change with respect to the design position—positive is termed jounce and negative is termed rebound.

FIGURE 2 Generating artificial suspension property files.



For jounce motion, the general motion $\dot{q}$ can be solved by a symbolic solver from the given target at a specific jounce position, according to the suspension jounce target in Table 3. The second step is to integrate the general motion with a small time step Δt to get the next position q_{t+1}. The targets for the new position q_{t+1} can be read again from the target curve. The process is repeated until the end of the jounce travel. The motion file is generated by the motion q at each jounce position. For the steering motion, the process is similar to the jounce motion. The steering motion file is generated by the motion q at each rack travel or steering wheel angle. To formulate the suspension property file represented by 3D splines for the front steering subsystem, a superposition of the steering motion and the jounce motion is used. Figure 2 shows the process of generating artificial suspension property files.

2.2. RL Environment

Once the suspension property files are generated, the RL environment shown in Figure 3 also includes a simulation environment [34], a reward function, and a pre-check function. The mechanism of the pre-check function ensures that only the suspension property files that meet the target range can be simulated. Invalid suspension property files will lead to a punishment in the reward function. The shortcut path stabilizes the training process and improves training efficiency. Many parts, such as wheel rate, tire models, suspension compliance, brake control, powertrain control, and driver model, have been modeled with fixed parameters, since they are needed for simulation, even though they are not subject to design in the present article.

2.2.1. Simulation Scenarios The suspension property files are attached to an existing vehicle in this study. The vehicle is simulated in a set of driving scenarios, including acceleration, braking, and ramp steer scenarios. The selected driving scenarios reflect the vehicle driving behaviors influenced by suspension kinematics targets. For example, the pitch angle during acceleration is influenced by the anti-lift in the front suspension and anti-squat in the rear suspension. The simulation scenarios aim to cover fundamental longitudinal and lateral vehicle dynamics behaviors. The measurement of the vehicle's performance is based on the postprocessing simulation results. The complete vehicle performance targets are measured according to Table 1. Figure 8 shows the results from the simulation.

FIGURE 3 RL environment overview.

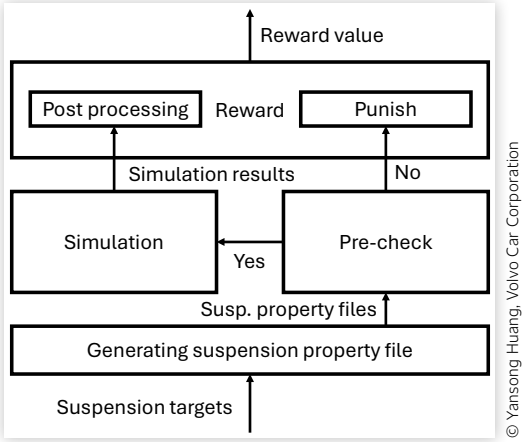


TABLE 1 Complete vehicle targets.

Acceleration scenario*	
Pitch angle at 0.1 g (deg)	Pitch angle at 0.35 g
Pitch angle at 0.5 g	
Braking scenario**	
Pitch angle at −0.1 g (deg)	Pitch angle at −0.35 g
Pitch angle at −0.6 g	
Ramp steer scenario for roll***	
Roll angle at −0.2 g (deg)	Roll angle at −0.5 g
Ramp steer scenario for pitch***	
Pitch angle at −0.2 g (deg)	Pitch angle at −0.6 g
Ramp steer scenario for Jacking***	
Vertical displacement at −0.4 g (mm)	Vertical displacement at −0.7 g
Handling diagram from ramp steer***	
Understeer gradient at 0.1 g (deg/g)	Understeer gradient at 0.5 g

* Acceleration scenario: The vehicle accelerates from 18 km/h in 3 s with open loop constant throttle.
** Braking scenario: The vehicle decelerates from 160 km/h in 1 s with open loop constant braking.
*** Ramp steer scenario: The vehicle steer 9 deg/s up to 6 s with constant velocity 100 km/h.

2.2.2. Reward Function The reward function rewards the agent by comparing the simulation results and the target values. For each complete vehicle target i in Table 1, it consists of complete vehicle target ranges $[\epsilon_{min,i}, \epsilon_{max,i}]$, maximum reward $R_{max,i}$, and punishment gradient $\dot{R}_i$. The relation between the individual target reward R_i and the simulation result ϵ_i is shown in Equation 2. If the suspension does not pass the pre-check in Figure 3, a punishment $R_{punish,i}$ is assigned to the reward R_i.

$$R_i = \begin{cases} R_{max,i}, & \text{if } \epsilon_{min,i} < \epsilon_i < \epsilon_{max,i} \\ R_{max,i} - \dot{R}_i \cdot \left|\epsilon_i - \epsilon_{min,i}\right|, & \text{if } \epsilon_i < \epsilon_{min,i} \\ R_{max,i} - \dot{R}_i \cdot \left|\epsilon_i - \epsilon_{max,i}\right|, & \text{if } \epsilon_i > \epsilon_{max,i} \end{cases} \quad (2)$$

Equation 2 shows that the reward R_i is a linear function of the simulation result ϵ_i. A linear function is simple to tune and is less sensitive to hyperparameters than a nonlinear function. Although a linear reward was used for simplicity and stability, the trained RL agent was able to discover configurations that reflect an implicit balance between different complete vehicle targets (see Figure 8). While longitudinal and lateral dynamic objectives are not shown separately, the resulting complete vehicle targets (see Table 2) are consistent with typical vehicle dynamics design expectations. The reward R_i is set to $R_{max,i}$ if the simulation result ϵ_i is within the target range $[\epsilon_{min,i}, \epsilon_{max,i}]$. The reward R_i decreases linearly if the simulation result ϵ_i is outside the target range. The total reward

TABLE 2 Complete vehicle targets and reward parameters setup*.

Parameter	R_{max}	$\dot{R}$	$\epsilon_{max,min}$**
Acceleration scenario (unit)	(1)	(1/deg)	(deg)
Pitch angle at 0.1 g (deg)***	0	1000	(−0.189, −0.189)
Pitch angle at 0.35 g	0	1000	(−0.730, −0.730)
Pitch angle at 0.5 g	0	100	(−1.087, −1.087)
Braking scenario	(1)	(1/deg)	(deg)
Pitch angle at −0.1 g (deg)	0	10,000	(0.038, 0.038)
Pitch angle at −0.35 g	0	1000	(0.337, 0.337)
Pitch angle at −0.6 g	0	1000	(0.664, 0.664)
Ramp steer scenario for roll	(1)	(1/deg)	(deg)
Roll angle at −0.2 g (deg)	0	1000	(−0.478, −0.478)
Roll angle at −0.5 g	0	100	(−1.211, −1.211)
Ramp steer scenario for pitch	(1)	(1/deg)	(deg)
Pitch angle at −0.2 g (deg)	0	1e5	(−0.0068, −0.0068)
Pitch angle at −0.6 g	0	10,000	(−0.073, −0.073)
Ramp steer scenario for Jacking	(1)	(1/mm)	(mm)
Vertical displacement at −0.4 g (mm)	0	1000	(1.157, 1.157)
Vertical displacement at −0.7 g	0	100	(4.443, 4.443)
Handling diagram from ramp steer	(1)	(1/[deg/g])	(deg/g)
Understeer gradient at 0.1 g (deg/g)	0	1000	(0.724, 0.724)
Understeer gradient at 0.5 g	0	100	(1.914, 1.914)

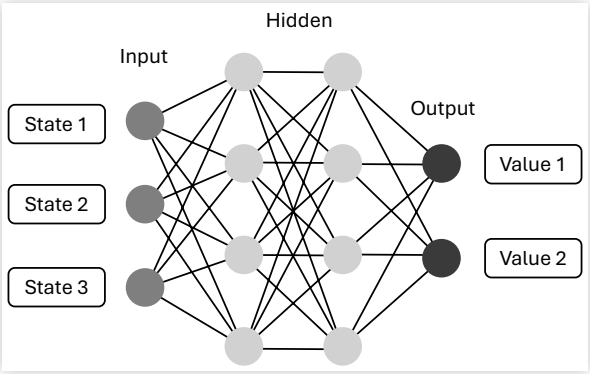
* These are example targets for a test vehicle used for the concept study at Volvo Cars.
** The complete vehicle targets are simplified as one value, therefore ϵ_{max} = = ϵ_{min} in the case study.
*** Target setup referenced to initial condition at 0 g.

R is the summation of each individual reward R_i from Equation 2, meaning $R = \sum_{i=1}^{n} R_i$. The punishment gradient $\dot{R}_i$ as weight factor is adjustable for different targets and learning tasks. The simulation environment is implemented in Python using the OpenAI Gym framework [38]. It cooperates with a learning agent to find the optimal suspension kinematics targets. The architecture of the learning agent is described in the next section.

3. Learning Agent

The learning agent is adapted based on the stochastic actor-critic framework [30]. The actor part consists of multiple Gaussian distributions that represent the suspension kinematics targets. The critic part is modeled by a neural network function and outputs a high-dimensional value function that estimates the expected reward.

FIGURE 4 Critic network architecture.



The actor gets updated by the critic part using temporal-difference learning, and the critic part gets updated by training examples from the memory buffer. This section goes through the key elements and describes the learning mechanism of the learning agent.

3.1. Agent Critic

The critic neural network is used to estimate values $\hat{V}$ for individual suspension kinematics targets according to certain states shown in Figure 4. The weight vector W includes scale weight *w* and bias term *b*. The states are observations from the simulation environment, for example, the complete vehicle targets from Table 1. The critic neural network is implemented using PyTorch [39] with customized layers and sizes. To stabilize the training process, batch normalization is applied to the critic network. The gradient of each layer, which is used for training, is trimmed automatically based on the connections between layers. The critic neural network aims to generate values that update the policy in the actor part. A well-trained critic network should provide a good estimation of the expected reward, which leads to the convergence of the policy in the actor part.

3.2. Agent Actor

The actor is supposed to generate suspension kinematics targets based on the policy. The policy $\boldsymbol{\pi}$ is a set of Gaussian distributions with mean $\boldsymbol{\theta}_{\mu}^{\top}$ and standard deviation $\boldsymbol{\theta}_{\sigma}^{\top}$ as shown in Equation 3.

$$\pi\left(a|\boldsymbol{\theta}\right)=\frac{1}{\sqrt{2\pi\sigma\left(\boldsymbol{\theta}\right)^{2}}}\exp\left(-\frac{\left(a-\mu\left(\boldsymbol{\theta}\right)\right)^{2}}{2\sigma\left(\boldsymbol{\theta}\right)^{2}}\right) \tag{3}$$

where

a is the action

$$\mu\left(\boldsymbol{\theta}\right)=\boldsymbol{\theta}_{\mu}^{\top}$$

$$\sigma\left(\boldsymbol{\theta}\right)=\exp\left(\boldsymbol{\theta}_{\sigma}^{\top}\right)$$

To update the policy, the gradient with action *a* is expressed in Equations 4 and 5 from [30],

$$\nabla\ \ln\pi\left(a|\boldsymbol{\theta}_{\mu}\right)=\frac{1}{\sigma\left(\boldsymbol{\theta}\right)^{2}}\left(a-\mu\left(\boldsymbol{\theta}\right)\right) \tag{4}$$

$$\nabla\ \ln\pi\left(a|\boldsymbol{\theta}_{\sigma}\right)=\frac{\left(a-\mu\left(\boldsymbol{\theta}\right)\right)^{2}}{\sigma\left(\boldsymbol{\theta}\right)^{2}}-1 \tag{5}$$

3.3. Training with Temporal-Difference Learning

The training process is based on the temporal-difference learning algorithm with an average reward [40]. The temporal-difference $\boldsymbol{\delta}$ is a multidimensional scale vector that is broadcast by one-dimensional reward and multi-dimensional predicted values, as shown in Equation 6.

$$\delta=R-\bar{R}+\hat{V}\left(S'|\mathbb{W}\right)-\hat{V}\left(S|\mathbb{W}\right) \tag{6}$$

where

R and $\bar{R}$ are one-dimensional scales reflecting total reward and average reward

$\hat{V}$ is a multidimensional vector from the critic network output

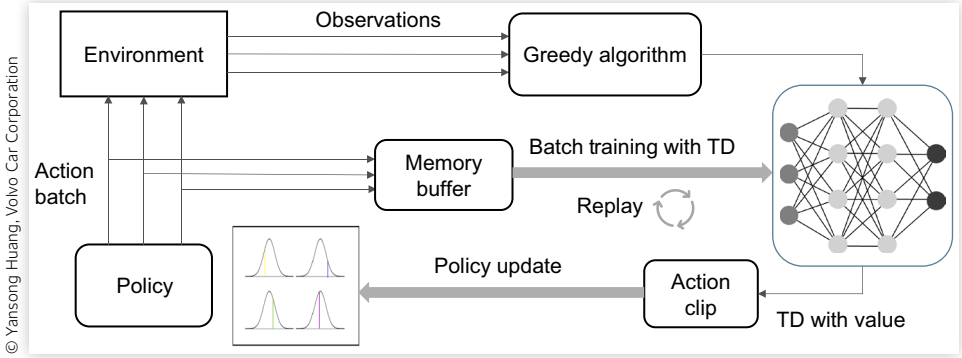
The weight update includes three steps for the average reward, critic, and actor parts. The average reward is updated by the temporal-difference $\boldsymbol{\delta}$ times the learning rate $\alpha^{\bar{R}}$. The gradient is essentially the key to weight updates, together with the temporal-difference and learning rate. The gradient of the critic network can be obtained thanks to the PyTorch Autograd feature, and the actor part is calculated from Equations 4 and 5. The process of weight update is shown in Equations 7, 8, and 9.

$$\bar{R}\leftarrow\bar{R}+\alpha^{\bar{R}}\delta \tag{7}$$

$$\mathbb{W}\leftarrow\mathbb{W}+\alpha^{\mathbb{W}}\delta\ \nabla\ \hat{V}\left(S,\mathbb{W}\right) \tag{8}$$

$$\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\alpha^{\theta}\delta\ \nabla\ ln\pi\left(A|\boldsymbol{\theta}\right) \tag{9}$$

FIGURE 5 Agent learning mechanism.



3.4. Agent Learning Mechanism

The learning mechanism is shown in Figure 5 and Algorithm 1. The agent starts with an initial state, policy parameters $\boldsymbol{\theta}$, and randomly initialized value network parameters W. The batch actions are sampled from the current policy distribution and fed into the simulation environment. The simulation experience is stored in the memory buffer. The action that has the highest reward will be used to update the actor weight factors. The experience in memory buffers is used to train the critic network with the help of the temporal-difference error shown in Equation 6. The memory buffer represents the historical experience. Notice that replay training with the memory buffer improves training efficiency. With the current state, a value can be estimated by the updated critic network and provide an updated TD error. Then, the policy parameters $\boldsymbol{\theta}$ can be updated by the gradient of the policy from Equations 4 and 5. A clip function is applied to the policy parameters to avoid gradient explosion. If all actions generated by the policy fail to pass the pre-check function, the agent resets the state, average reward, and policy parameters. The agent repeats the process until the end of the training episode.

4. Case Study

In this section, a demonstration of the proposed method is presented as a case study. The investigation begins by establishing 14 complete vehicle targets, which serve as input for the RL algorithm to determine the optimal 30 suspension targets as shown from vehicle level in Table 2 to subsystem level in Table 3. A set of suspension

ALGORITHM 1 Agent learning algorithm based on actor-critic framework.

Initialize policy parameters $\boldsymbol{\theta}$, value network parameters $\mathbb{W}$, initial state s_0, and initial average reward $\bar{R}$
Initialize learning rates α_θ and α_W for actor and critic
Initialize average reward learning rate $\alpha^{\bar{R}}$
for each of episode: **do**
  while All batch action A not pass Pre-check **do**
    Sample batch action $A \sim \pi(\cdot|\theta)$ {Select action according to current policy}
    Execute action $a_i \in A$, observe reward r_i and next state $s_i^{'}$
    Store transition $(s_i, a_i, r_i, s_i^{'})$ in memory buffer
    while end of replay: **do**
      Sample the experience with batch size
      Compute TD error: $\boldsymbol{\delta} = R - \bar{R} + \hat{\boldsymbol{V}}(S^{'}|\mathbb{W}) - \hat{\boldsymbol{V}}(S|\mathbb{W}) \quad \{\boldsymbol{\delta} \in \mathbb{R}^s\}$
      Update critic: $\mathbb{W} \leftarrow \mathbb{W} + \alpha_w \, \boldsymbol{\delta} \, \nabla \hat{\boldsymbol{V}}(S, \mathbb{W})$
    end while
    Select action a_{max} with the highest reward and update actor part with $(s_{max}, a_{max}, r_{max}, s_{max}^{'})$
    Compute TD error: $\boldsymbol{\delta} = R - \bar{R} + \hat{\boldsymbol{V}}(S^{'}|\mathbb{W}) - \hat{\boldsymbol{V}}(S|\mathbb{W}) \quad \{\boldsymbol{\delta} \in \mathbb{R}^s\}$
    Update actor: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_\theta \, I \, \boldsymbol{\delta} \, \nabla \ln \pi(a_{max}|\boldsymbol{\theta})$
    Clip $\boldsymbol{\theta}_\sigma$ with interval $[\boldsymbol{\theta}_\sigma^{min}, \boldsymbol{\theta}_\sigma^{max}]$
    $\bar{R} \leftarrow \bar{R} + \alpha^{\bar{R}} \, \delta_{max}$
    $I \leftarrow \gamma \, I$
    $s \leftarrow s_{max}^{'}$
  end while
  reset state s_0, average reward $\bar{R}$, and policy parameters $\boldsymbol{\theta}$, $I \leftarrow 1$
end for

TABLE 3 Targets selected to generate suspension property file.

First order	Second order	Third order
Front suspension jounce target*		
1st bump steer (deg/m)	2nd bump steer ([deg/m]/dm) **	3rd bump steer ([deg/m]/dm^2)
1st bump camber (deg/m)	2nd bump camber ([deg/m]/dm)	3rd bump camber ([deg/m]/dm^2)
1st anti-dive (%)	2nd anti-dive (%/dm)	3rd anti-dive (%/dm^2)
1st anti-lift (%)	2nd anti-lift (%/dm)	3rd anti-lift (%/dm^2)
1st RCH (mm)	2nd RCH (mm/dm)	3rd RCH (mm/dm^2)
Front suspension steering target		
1st Caster angle (deg)	2nd Caster angle (deg/25 deg)	3rd Caster angle (deg/25 deg^2)
1st Kingpin angle (deg)	2nd Kingpin angle (deg/25 deg)	3rd Kingpin angle (deg/25 deg^2)
1st Caster trail (mm)	2nd Caster trail (mm/25 deg)	3rd Caster trail (mm/25 deg^2)
1st Scrub radius (mm)	2nd Scrub radius (mm/25 deg)	3rd Scrub radius (mm/25 deg^2)
1st WLLA*** (mm)	2nd WLLA (mm/25 deg)	3rd WLLA (mm/25 deg^2)
Rear suspension jounce target*		
1st bump steer (deg/m)	2nd bump steer ([deg/m]/dm)	3rd bump steer ([deg/m]/dm^2)
1st bump camber (deg/m)	2nd bump camber ([deg/m]/dm)	3rd bump camber ([deg/m]/dm^2)
1st anti-squat (%)	2nd anti-squat (%/dm)	3rd anti-squat (%/dm^2)
1st anti-lift (%)	2nd anti-lift (%/dm)	3rd anti-lift (%/dm^2)
1st RCH (mm)	2nd RCH (mm/dm)	3rd RCH (mm/dm^2)

* Learning parameter used in case study.

** Unit explanation: 2nd bump steer ([deg/m]/dm) means the bump steer unit is (deg/m), and the target is measured at 100 mm jounce travel. For more explanation, refer to [37].

*** WLLA: Wheel load level arm.

kinematics targets is selected as the learning parameters. The task for the RL agent is to find the setup for the reference model without prior knowledge of the reference model's target setup. Figure 6 illustrates the case study workflow, which starts with the complete vehicle targets setup. The RL agent cascades the complete vehicle targets into suspension targets through the RL learning process. Once the training converges and the stopping criteria are met, a simulatable suspension property file is generated as the outcome from RL and artificial suspension property files. The complete vehicle targets are then simulated in VI-CarRealTime, enabling systematic verification that the design meets the specified performance criteria, as shown in Figure 11. The study is conducted with Volvo XC60 [41] body model and MF-Tyre 5.2 tire model [42].

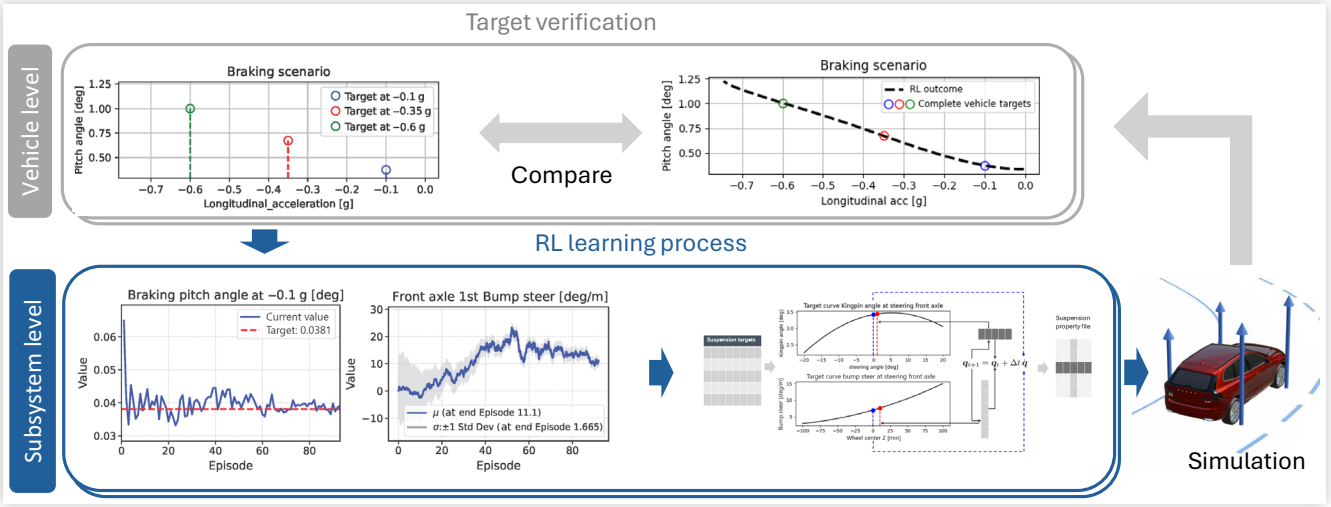
4.1. Complete Vehicle Target and Learning Parameter Setup

Table 2 summarizes the selected complete vehicle targets across various driving scenarios. For acceleration and braking scenarios, pitch angles are measured at three distinct levels of longitudinal acceleration to capture the suspension kinematics during both minor and significant wheel travel. This approach specifically influences higher-order targets such as 2nd and 3rd anti-lift coefficients. Roll motion is evaluated using a ramp steer scenario, with targets established for roll, pitch, and jacking motion. These parameters primarily influence kinematic targets like roll center height (RCH). Additionally, understeer gradient is incorporated as a complete vehicle target to regulate suspension kinematics parameters, particularly bump steer and bump camber characteristics. The reward parameters are set according to Equation 2. A reference model is used as an upper bound for the reward function, which means $R = \sum_{i=1}^{n} R_i$ is set to 0. Furthermore, the complete vehicle targets simplified as one value in this case study.

This case study focuses on suspension targets related to jounce motion, as detailed in Table 3. Using the methodology outlined in Section 3, the primary objective is to determine the optimal suspension jounce targets that fulfill the complete vehicle requirements. To facilitate a front suspension model that can be simulated, additional steering targets are listed as fixed parameters, as also presented in Table 3: Front suspension steering target category.

A collection of RL hyperparameters is presented in Table 4. These hyperparameters were tuned experimentally to achieve a balance with computational efficiency. This termination threshold for early stop was established based on experimental validation.

FIGURE 6 Case study with Volvo XC60 body model in VI-CarRealTime environment.



5. Results and Discussion

The case study results are shown in this section after completing the training process described in Section 4. Figure 7 illustrates the training progression, depicting the best reward achieved per episode. The learning curve exhibits both rapid advancement phases and periods of incremental improvement. The rapid advancement phases indicate an efficient learning step at the beginning of the training process, thanks to the pre-check function described in Section 2.2. After reaching a rapidly growing reward, the learning process stabilizes, and the reward converges to the maximum reward that triggered the stopping criteria. The training process concluded at episode 93, reaching an optimal reward value of −83. This termination threshold was established based on experimental validation. For this case study, the early stopping criteria also play a role in preventing overfitting. The continuous exploration and exploitation of the learning agent means the neural network in the critic part works together with the actor part to find the optimal suspension kinematics targets.

TABLE 4 RL hyperparameter setup.

Parameter	Critic	Actor	Environment
Environment			
Max episodes	—	—	100
Episodes replay size	—	—	10
Early stopping reward	—	—	−5000
Minibatch size	—	—	8
Replay buffer size	—	—	10,000
Learning agent			
Neural netwotk input dimension	14	—	—
Neural netwotk output dimension	30	—	—
Hidden layers	(20, 20, 30)	—	—
Training step size	1e−3	1e−3(μ), 1e−5(σ)	—
Average reward step size	1e−1	—	—
Actor discount factor γ	1	—	—
Prevent overfitting	Early stop	—	—

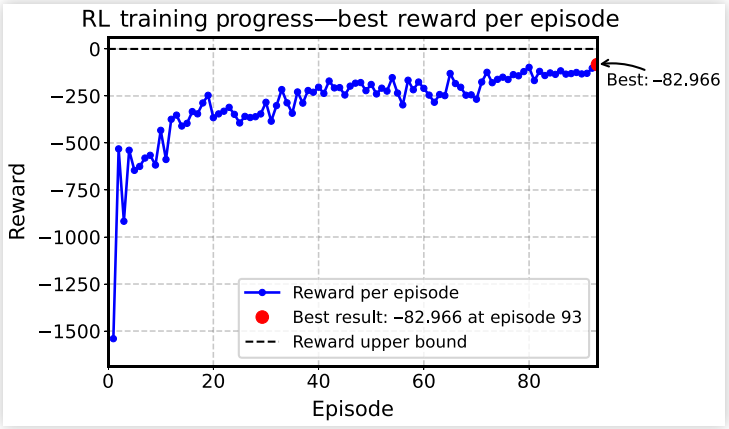
PyTorch 2.4.1.
CPU: Intel Xeon W5-2455x.
RAM: 64 GB.
GPU: NVIDIA Quadro RTX A4500 20 GB.
Training time with random initial conditions: 120 h.

Figure 8 illustrates the learning progression for each individual complete vehicle target. Throughout the training process, the RL agent effectively balances these multiple targets, achieving convergence by the final episode. The convergence characteristics of the suspension targets will be examined in a learning curve for each suspension target.

In alignment with the complete vehicle target learning curve illustrated in Figure 8, the suspension target learning process is characterized by two key parameters: mean μ and standard deviation σ per episode, as shown in Figures 9 and 10. These parameters correspond to the policy parameters defined in Equation 3. While some subfigures demonstrate convergence of both the mean value and its variation, others exhibit only partial convergence. As established in Equation 1, the nonlinear suspension target incorporates coefficients up to the third order. The value function assigned to each complete vehicle target helps the agent find the correct path toward the optimal setup. Furthermore, the convergence of suspension kinematics targets in Figure 9 demonstrates the agent's ability to find the optimal setup. The variation σ for each parameter also provides a sensitivity representation.[2] Typically, the

[2] Parameter sensitivity indicates how different values of an input variable impact a particular output variable.

FIGURE 7 Reward for each episode during the RL training process.



σ value converges to a small value, indicating the critical parameters for the setup. The other parameters have a higher σ value, which means they are less critical for the setup. For example, Figure 10 shows the convergence of the suspension target front axle 1st anti-dive. The σ value converges to a small value, and the μ shifts to a value that yields the highest reward. In contrast, Figure 10 shows the lack of convergence of the suspension target front axle 1st anti-lift. the σ remains high throughout the training process, which means the agent considers this particular suspension target less critical to achieving an overall high reward. Therefore, such a suspension target can be considered a less sensitive target.

The rapid convergence of front axle 1st anti-dive also indicates strong policy learning, whereas the persistent high σ in front axle 1st anti-lift reveals underlying trade-off tension. Interestingly, the RL agent deprioritized front axle 1st anti-lift, suggesting this target may not be critical for meeting the higher-level objectives. As the algorithm can automatically identify the critical suspension kinematics targets, the dimension of suspension targets can be increased without significantly increasing the computational cost. The interpretation of the σ and μ values reflects how the agent makes new design proposals according to policy and eventually finds the optimal suspension kinematics targets. The results show that the proposed method can be used to solve high-dimensional optimization problems in the vehicle dynamics field.

Figure 11 demonstrates that the target learning algorithm effectively achieved its intended purposes. The verification focuses on comparing the performance against the original targets established in Table 2. This comparison confirms the successful translation from vehicle-level targets to subsystem-level targets in the suspension field that was highlighted in Figure 6. This supports the hypothesis that reward shaping with linear penalties can still capture multi-objective tension under certain conditions.

This section has demonstrated the complete design workflow from vehicle targets to suspension hardpoints. The process began with 14 complete vehicle targets that were used by the RL algorithm to establish 30 suspension kinematic targets. Additionally, many fixed values were given, such as vehicle data but also some fixed suspension parameters, e.g., wheel rate, which was not allowed to vary in the design. Two findings are concluded in this section.

- The RL algorithm effectively converged on optimal suspension targets that fulfilled the specified complete vehicle targets.
- The implemented design demonstrated excellent correlation with the original vehicle targets across all evaluated performance metrics.

This case study provides validation of the methodology described in Section 2 and 3, demonstrating its capability in automotive suspension design applications.

6. Conclusion

This study demonstrates that RL can effectively address the challenge of high-dimensional suspension kinematics optimization in early vehicle development stages. It highlights both the promise and the practical limitations of RL in multi-objective suspension design field. Although not benchmarked against traditional methods in this study, the RL agent produced results consistent with physical expectations, suggesting the method's applicability for earlystage concept development. Through the implementation of an actor-critic framework, we learned that RL agents can successfully navigate complex parameter trade-offs without requiring detailed suspension hardpoint knowledge, making the approach particularly valuable during conceptual design phases.

FIGURE 8 Learning curve per episode: Individual completes vehicle target learning process.

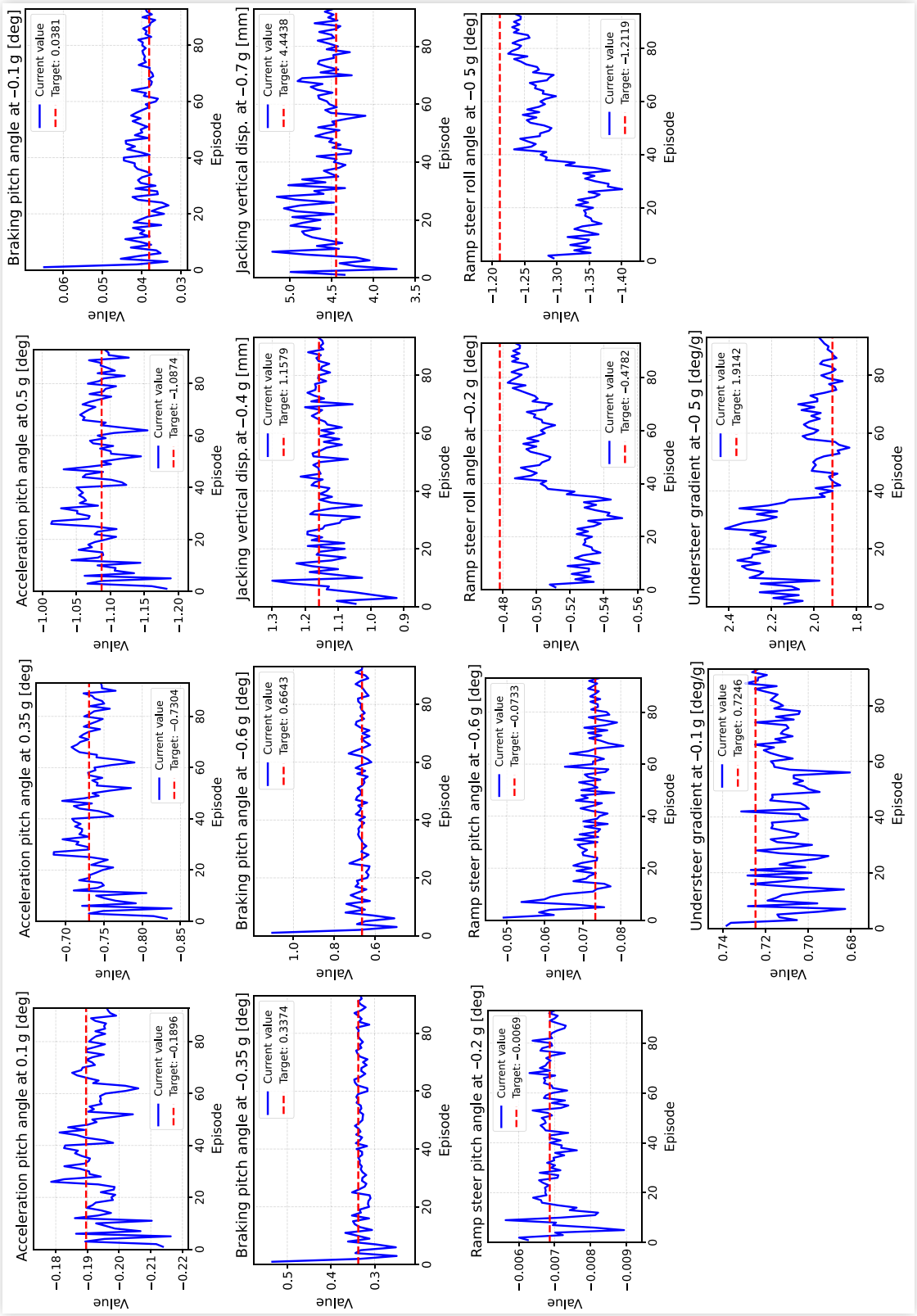


FIGURE 9 Learning curve per episode with μ and σ for front axle suspension targets.

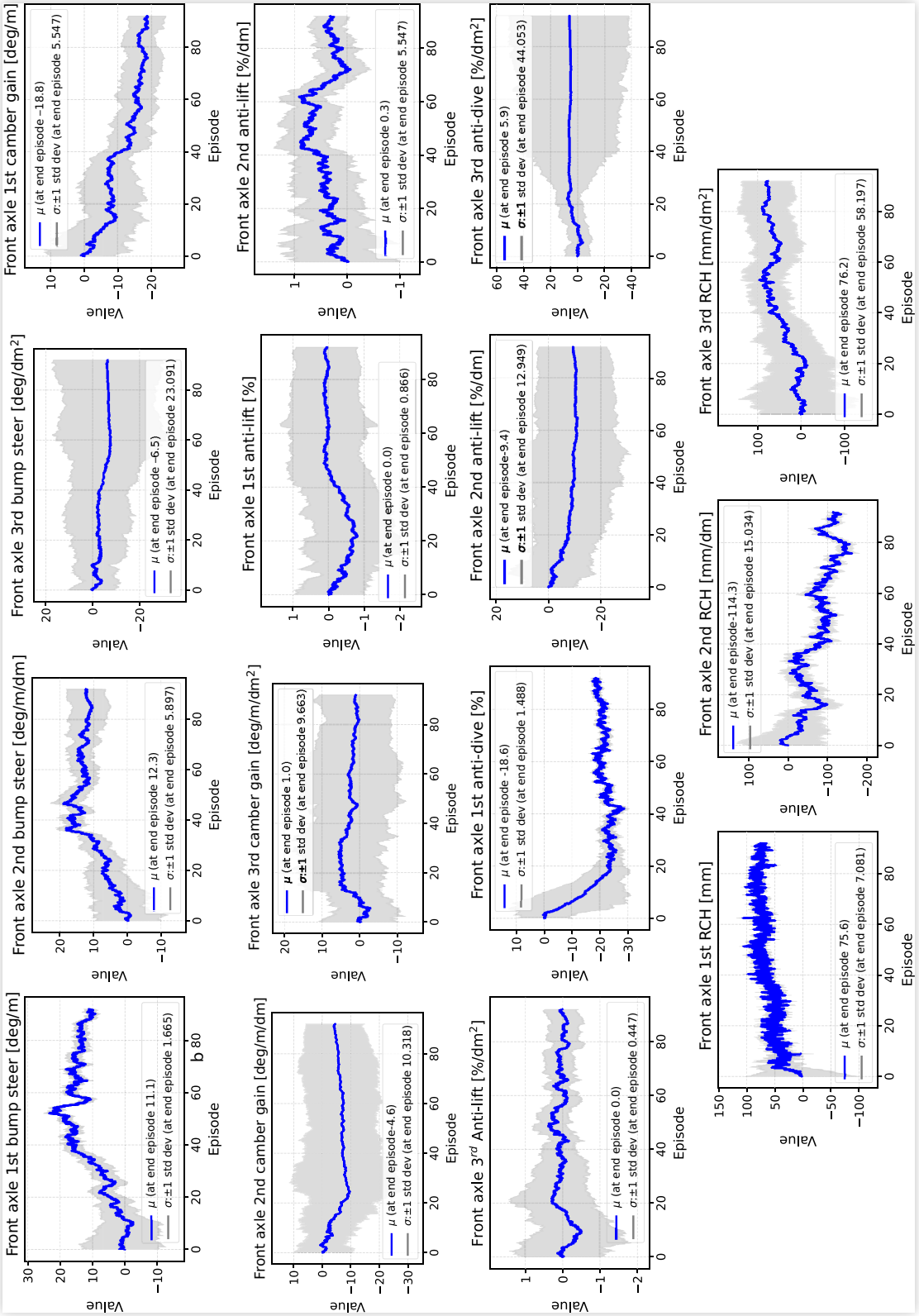


FIGURE 10 Learning curve per episode with μ and σ rear axle.

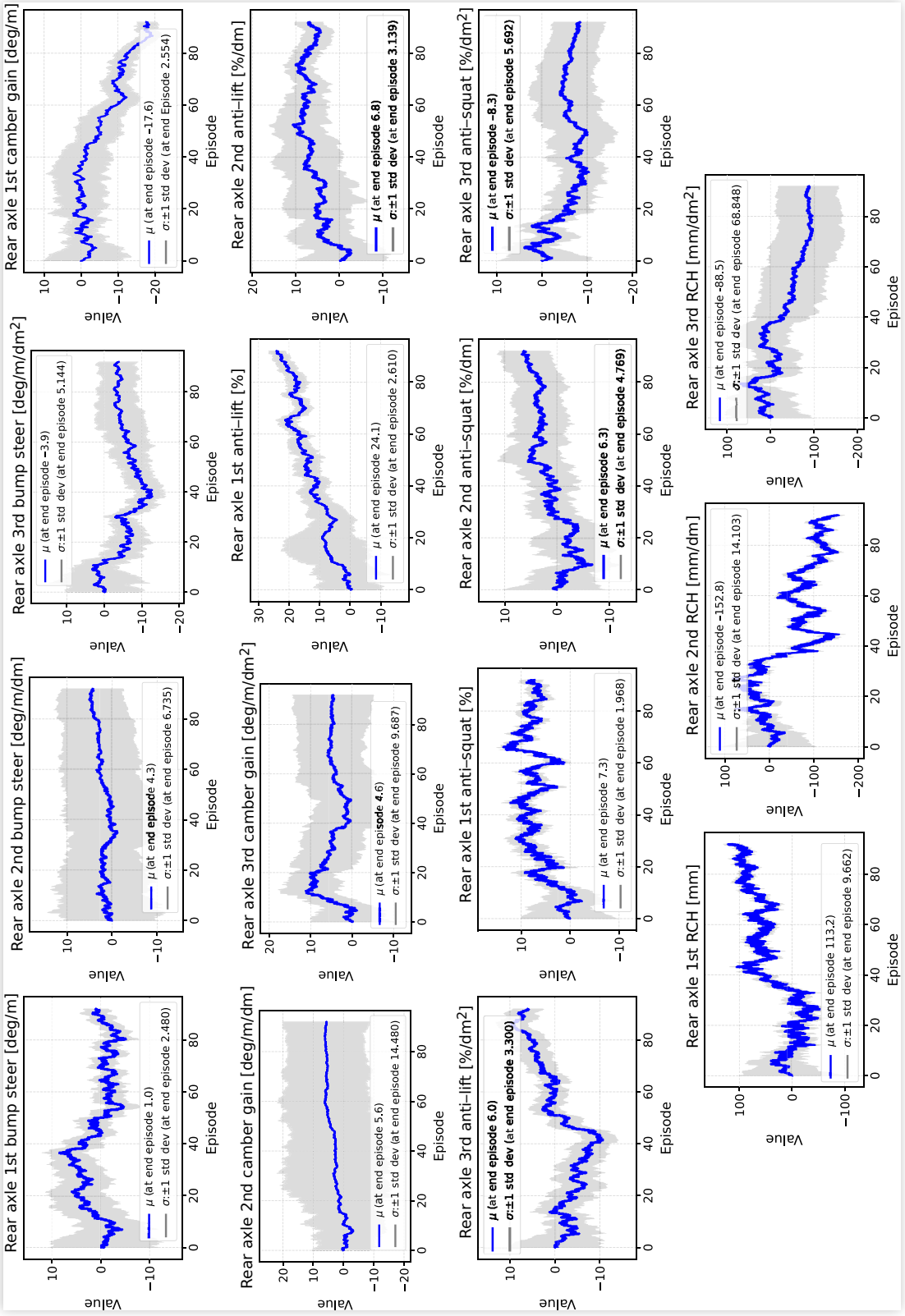
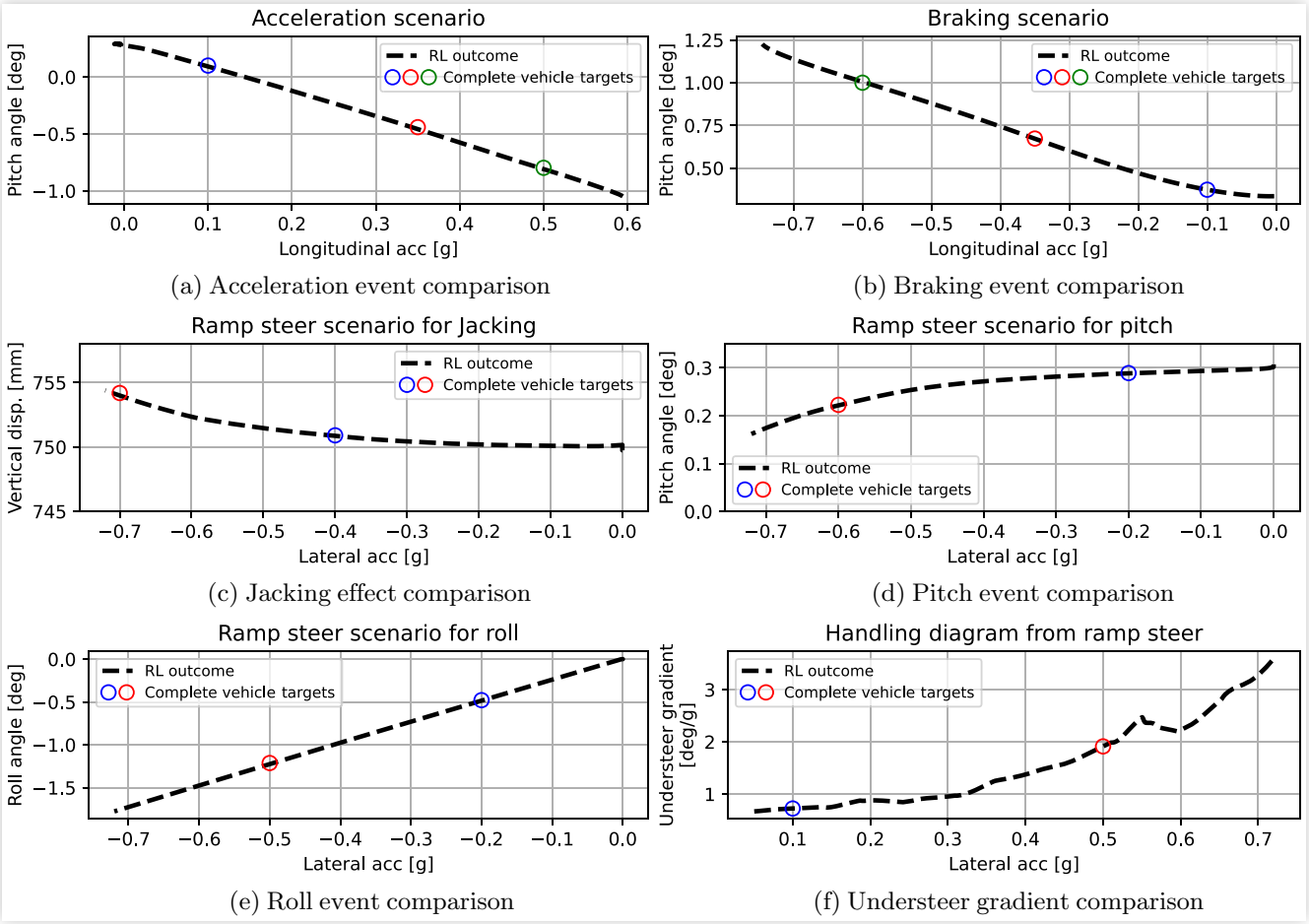


FIGURE 11 Comparison of original targets with RL simulation results.



(a) Acceleration event comparison

(b) Braking event comparison

(c) Jacking effect comparison

(d) Pitch event comparison

(e) Roll event comparison

(f) Understeer gradient comparison

The research revealed that RL-based optimization can achieve meaningful convergence in suspension targets, with the agent effectively learning value functions for complete vehicle performance and policy functions for parameter selection. The method's ability to identify critical design parameters through convergence patterns provides valuable sensitivity insights that complement traditional engineering approaches.

The actor-critic framework developed for suspension optimization demonstrates broader applicability to high-dimensional engineering design problems. The methodology of using RL for inverse mapping from vehicle performance requirements to design parameters can be extended beyond automotive applications to other complex engineering systems where traditional optimization struggles with dimensionality and parameter interdependencies. The approach of embedding domain knowledge through reward function design while allowing the agent to discover parameter relationships is transferable across engineering disciplines.

Several limitations constrain the current work's scope. The validation is limited to a single vehicle architecture within the VI-CarRealTime environment, raising questions about generalizability across different vehicle types and simulation platforms. The reward function design, while effective, remains relatively simple and may not capture all relevant performance trade-offs in real-world applications. The method's dependency on simulation fidelity means that model uncertainties in VI-CarRealTime could propagate to the optimization results. Additionally, the current approach focuses solely on kinematics targets without considering other subsystems that are crucial in vehicle development. The RL agent's performance is also sensitive to hyperparameter choices, which may require further tuning for different applications.

7. Future Work

Building on the identified limitations, several research directions emerge for advancing RL-based vehicle design optimization.

Methodological Extensions: Future investigations will explore nonlinear reward functions, particularly parabolic formulations, to better capture engineering trade-offs and potentially accelerate convergence. The integration of transfer learning capabilities will enable pre-trained models to reduce training time for new vehicle projects, addressing the practical concern of development timeline constraints.

Scope Expansion: The methodology will be extended to encompass additional subsystems including suspension compliance characteristics and tire design parameters. This multi-subsystem approach could provide valuable insights for cross-functional engineering teams by identifying critical design parameters across different domains and their interdependencies.

Enhanced Environment Complexity: Integration with DiL simulation environments represents a significant advancement opportunity, enabling the incorporation of subjective criteria alongside objective performance metrics. This capability would allow the RL agent to adapt to specific driver preferences and diverse testing scenarios, bridging the gap between objective engineering requirements and human-centered design considerations.

Validation and Robustness: Validation across multiple vehicle architectures and simulation platforms is essential to establish the method's generalizability. Future work will test generalizability across architectures. The demonstrated RL approach enables its application in multiple real-world vehicle development projects, which will provide valuable experience.

Contact Information

Yansong Huang
yansong.huang@chalmers.se
VEAS, Chalmers University of Technology,
Gothenburg, Sweden
yansong.huang@volvocars.com
Chassis department, Volvo Cars Corporation,
Gothenburg, Sweden

Max Boerboom
max.boerboom@volvocars.com
Vehicle Dynamics department, Volvo Cars Corporation,
Gothenburg, Sweden

Krister Wolff
krister.wolff@chalmers.se
VEAS, Chalmers University of Technology,
Gothenburg, Sweden

Jacobson Bengt
bengt.jacobson@chalmers.se
VEAS, Chalmers University of Technology,
Gothenburg, Sweden

References

1. Wahde, M., *Biologically Inspired Optimization Methods* (Southampton, UK: WIT Press, 2008), ISBN:978-1-84564-148-1.
2. Arora, R.K., *Optimization: Algorithms and Applications*. Vol. 53 (Boca Raton, FL: CRC Press, 2015).
3. Wei, Z., Hua, Z., and Jiang, Z., "Drag Reduction Study on Vehicle Shape Optimization Using Gradient-Based Adjoint Method," SAE Technical Paper 2024-01-2528 (2024), doi:https://doi.org/10.4271/2024-01-2528.
4. Albers, I., "Auslegungs-und Optimierungswerkzeuge für die effiziente Fahrwerkentwicklung: Analyse—Synthese—Optimierung," PhD thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, 2009.
5. Nguyen, S.H. and Kim, H.G., "Stress-Constrained Shape and Topology Optimization with the Level Set Method Using Trimmed Hexahedral Meshes," *Computer Methods in Applied Mechanics and Engineering* 366 (2020): 113061.
6. Shaik, I., Jain, P.C., Anjaneyulu, P.S.R. et al., "Shape Optimization of Shallow Domes Subjected to External Pressure," *Structural and Multidisciplinary Optimization* 57, no. 2 (2018): 903-908.
7. Mutter, F., "Eine effiziente achskonzeptentwicklung durch verwendung von optimierungsmethoden," PhD dissertation, Technische Universität Wien, 2018.
8. Mitchell, S., Smith, S., Damiano, A., Durgavich, J. et al., "Use of Genetic Algorithms with Multiple Metrics Aimed at the Optimization of Automotive Suspension Systems," SAE Technical Paper 2004-01-3520, doi:https://doi.org/10.4271/2004-01-3520.
9. Zhu, S., Ohsaki, M., Guo, X., and Zeng, Q., "Shape Optimization for Non-Linear Buckling Load of Aluminum Alloy Reticulated Shells with Gusset Joints," *Thin-Walled Structures* 154 (2020): 106830.
10. Garcia-Andrés, X., Gutiérrez-Gil, J., Martínez-Casas, J., and Denia, F.D., "Wheel Shape Optimization Approaches to Reduce Railway Rolling Noise," *Structural and Multidisciplinary Optimization* 62, no. 5 (2020): 2555-2570.
11. Sun, Q., Sun, Y., and Li, L., "Strength Analysis and Tooth Shape Optimization for Involute Gear with a Few Teeth," *Advances in Mechanical Engineering* 10, no. 1 (2018): 1687814017751957.
12. Brujic, D., Ristic, M., Mattone, M., Maggiore, P. et al., "CAD Based Shape Optimization for Gas Turbine Component Design," *Structural and Multidisciplinary Optimization* 41, no. 4 (2010): 647-659.
13. Liu, P., Jiang, W., Xu, Y., McCullough, T. et al., "A Particle Swarm Optimization-Based Method for Fast Parametrization of Transmission Plant Models," SAE Technical Paper 2019-01-0344 (2019), doi:https://doi.org/10.4271/2019-01-0344.
14. Dong, Y., Yao, X., and Xu, X., "Cross Section Shape Optimization Design of Fabric Rubber Seal," *Composite Structures* 256 (2021): 113047.

15. Sonmez, F.O., "Optimal Shape Design of Shoulder Fillets for Flat and Round Bars under Various Loadings," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 223, no. 8 (2009): 1741-1754.
16. Sonmez, F.O., "Shape Optimization of 2D Structures Using Simulated Annealing," *Computer Methods in Applied Mechanics and Engineering* 196, no. 35-36 (2007): 3279-3299.
17. Bottou, L., Curtis, F., and Nocedal, J., "Optimization Methods for Large-Scale Machine Learning," *SIAM Review* 60, no. 2 (2018): 223-311, doi:https://doi.org/10.1137/16M1080173.
18. Yang, T., "Advancing Non-Convex and Constrained Learning: Challenges and Opportunities," *AI Matters* 5, no. 3 (2019): 29-39, doi:https://doi.org/10.1145/3362077.3362085.
19. Kollmann, H.T., Abueidda, D.W., Koric, S., Guleryuz, E. et al., "Deep Learning for Topology Optimization of 2D Metamaterials," *Materials & Design* 196 (2020): 109098.
20. Rumelhart, D., Hinton, G., and Williams, R., "Learning Representations by Backpropagating Errors," *Nature* 323, no. 6088 (1986): 533-536, doi:https://doi.org/10.1038/323533a0.
21. Hinton, G., Osindero, S., and Teh, Y., "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation* 18, no. 7 (2006): 1527-1554, doi:https://doi.org/10.1162/neco.2006.18.7.1527.
22. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. et al., "Human-Level Control through Deep Reinforcement Learning," *Nature* 518, no. 7540 (2015): 529-533, doi:https://doi.org/10.1038/nature14236.
23. Nathan, K.B., Anthony, P.G., Georges, M.F., and Li, G., "Deep Reinforcement Learning for Engineering Design through Topology Optimization of Elementally Discretized Design Domains," *Materials & Design* 218 (2022): 110672, doi:https://doi.org/10.1016/j.matdes.2022.110672.
24. Dworschak, F., Dietze, S., Wittmann, M., Schleich, B. et al., "Reinforcement Learning for Engineering Design Automation," *Advanced Engineering Informatics* 52 (2022): 101612, doi:https://doi.org/10.1016/j.aei.2022.101612.
25. Vemireddy, K., Dittmar, T., Eckstein, L., Hesse, L. et al., "Development of a Driving Dynamics-Oriented Suspension Design during the Early Concept Phase," in *6th International Munich Chassis Symposium 2015* (Wiesbaden, Germany: Springer Fachmedien Wiesbaden, 2015), 233-255, ISBN:978-3-658-09711-0.
26. Buşoniu, L., De Bruin, T., Tolić, D. et al., "Reinforcement Learning for Control: Performance, Stability, and Deep Approximators," *Annual Reviews in Control* 46 (2018): 8-28, doi:https://doi.org/10.1016/j.arcontrol.2018.09.005.
27. Laflamme, C., Doppler, J., Palvolgyi, B., Dominka, S. et al., "Explainable Reinforcement Learning for Powertrain Control Engineering," *Engineering Applications of Artificial Intelligence* 146 (2025): 110135, doi:https://doi.org/10.1016/j.engappai.2025.110135.
28. Jeremy, B.K., Benjamin, D., and Kush, B., "Adaptive Control and Reinforcement Learning for Vehicle Suspension Control: A Review," *Annual Reviews in Control* 58 (2024): 100974, doi:https://doi.org/10.1016/j.arcontrol.2024.100974.
29. Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M. et al., "Solving Rubik's Cube with a Robot Hand," arXiv preprint arXiv:1910.07113, 2019.
30. Sutton, R.S. and Barto, A.G., *Reinforcement Learning: An Introduction* (Cambridge, MA: A Bradford Book, 2018), ISBN:0262039249.
31. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I. et al., "Mastering the Game of Go without Human Knowledge," *Nature* 550 (2017): 354-359, doi:https://doi.org/10.1038/nature24270.
32. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K. et al., "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model," *Nature* 588 (2020): 604-609, doi:https://doi.org/10.1038/s41586-020-03051-4.
33. Williams, R.J., "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning* 8, no. 3-4 (1992): 229-256, doi:https://doi.org/10.1007/BF00992696.
34. VI-Grade, "VI-CarRealTime," Version 2024.2, 2024, accessed June 2025, https://www.vi-grade.com/en/products/vi-carrealtime.
35. Huang, Y., "Target-Driven Road Vehicle Suspension Design," Licentiate thesis, Chalmers University of Technology, Gothenburg, Sweden, 2022, accessed November 2024, https://research.chalmers.se/en/publication/533119.
36. Johannes, K., Huang, Y., Jacobson, B., Brandin, T. et al., Automated Methods for the Suspension Pre-Development—Design of a Front Axle for a Long Range Electric Vehicle," in Pfeffer, P.E. (ed.), *14th International Munich Chassis Symposium 2023* (Berlin, Heidelberg: Springer Berlin Heidelberg, 2025), 223-243, ISBN:978-3-662-70348-9.
37. Huang, Y., Brandin, T., and Jacobson, B., "Linear and Nonlinear Kinematic Design of Multilink Suspension," *SAE Int. J. Passeng. Veh. Syst.* 16, no. 2 (2023): 105-117, doi:https://doi.org/10.4271/15-16-02-0007.
38. Brockman, G., Cheung, V., Pettersson, L., Schneider, J. et al., "OpenAI Gym," arXiv preprint arXiv:1606.01540, 2016.
39. Paszke, A., Gross, S., Massa, F., Lerer, A. et al., "Pytorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems* 32 (2019): 8024-8035.
40. Sutton, R.S., "Learning to Predict by the Methods of Temporal Differences," *Machine Learning* 3, no. 1 (1988): 9-44, doi:https://doi.org/10.1007/BF00115009.
41. Volvo Car Corporation, "Volvo XC60 Recharge," 2024, accessed December 18, 2024, accessed June 2025, https://www.volvocars.com/se/cars/xc60-hybrid/.
42. Delft-Tyre, "MF5.2 Equation Manual," 2013, accessed March 2025, https://pdfcoffee.com/mf52-equationmanual-pdf-free.html.

Appendix A: A Suspension Motion Property Files

TABLE A.1 Selected motion properties and correspond input.

Motion	Jounce travel	Steering travel
Front suspension jounce target		
Steer at ground	X	X
Camber angle	X	X
Side view angle	X	X
X-coordinate variation	X	X
Y-coordinate variation	X	X
Rear suspension jounce target		
Steer at ground	X	
Camber angle	X	
Side view angle	X	
X-coordinate variation	X	
Y-coordinate variation	X	