2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Applicability Study of Model-Free Reinforcement Learning towards an Automated Design Space Exploration Framework

1st Patrick Hoffmann
Corporate Research & Advance Eng.
Robert Bosch GmbH
Renningen, Germany
patrick.hoffmann3@bosch.com

2nd Kirill Gorelik
Corporate Research & Advance Eng.
Robert Bosch GmbH
Renningen, Germany
kirill.gorelik@de.bosch.com

3rd Valentin Ivanov
Department of Mechanical Engineering
Technische Universität Ilmenau
Ilmenau, Germany
valentin.ivanov@tu-ilmenau.de

***Abstract*—Design space exploration is a crucial aspect of engineering and optimization, focused on identifying optimal design configurations for complex systems with a high degree of freedom in the actor set. It involves systematic exploration while considering various constraints and requirements. One of the key challenges in design space exploration is the need for a control strategy tailored to the particular design. In this context, reinforcement learning has emerged as a promising solution approach for automatically inferring control strategies, thereby enabling efficient comparison of different designs. However, learning the optimal policy is computationally intensive, as the agent determines the optimal policy through trial and error. The focus of this study is on learning a single strategy for a given design and scenario, enabling the evaluation of numerous architectures within a limited time frame. The study also highlights the importance of plant modeling considering different modeling approaches to effectively capture the system complexity on the example of vehicle dynamics. In addition, a careful selection of an appropriate hyperparameter set for the reinforcement learning algorithm is emphasized to improve the overall performance and optimization process.**

***Index Terms*—Reinforcement Learning, Intelligent Control, Transportation and Vehicle Systems, Electric Vehicle**

I. Introduction

Cross-system integration is an emerging field in automotive development [1]. The incorporation of powertrain electrification, electromechanically controlled by-wire braking and steering systems, and a central E/E architecture introduces new possibilities for enhancing vehicle dynamics. For this reason, a design study is advisable to make the best possible use of functional and cost benefits. However, the design space is very large due to the available degrees of freedom. In this context, the design space encompasses a wide range of possibilities, including permutations of steering, braking, and propulsion systems acting at each of the four vehicle wheels. It also involves considering different centralization options, such as vehicle central, axle central, or wheel individual arrangements of actuators. Moreover, the design space accounts for the presence of redundant actuators, which introduces additional configurations and complexities that need to be investigated [2]. In order to find the best possible configuration fulfilling also the ISO 26262 standard [3], there is a need to analyze safety-critical failures of a (sub-) system in the design and to quantify safety metrics. Thus, for the Design Space Exploration (DSE), in addition to the design variants, the investigation of the seven fault classes [4] such as loss of function, more/less actuation than intended, intermittent actuation, wrong actuation direction, not requested actuation, and blocked function is required. In an exhaustive study, the incorporation of failure modes and component-level considerations, such as sizing characteristics, significantly increases the design space. Moreover, evaluating all designs across multiple driving scenarios is crucial to assess the fulfillment of objectives. The combination of these factors necessitates a substantial number of evaluations to enable a comprehensive DSE. To ensure optimal control and facilitate comparison between different over-actuated architectures, control strategies need to be developed for each potential configuration. To solve this problem, an automated derivation of control strategies for each topology is required to determine the most suitable configuration that satisfies the given requirements and achieves the desired performance across different operational scenarios and objectives.

The subsequent sections of this paper are structured as follows: Sec. II provides an overview of the state of the art and fundamental concepts underlying this study. The section is closed with the research gap and presents the contribution of this paper. Sec. III outlines the formulation of the problem definition. Sec. IV covers the implementation and evaluation of the plant model, while Sec. V focuses on closed-loop evaluations for the limited design space. Finally, Sec. VI concludes the paper.

II. State of the Art

To efficiently address the DSE problem, both control and system identification play crucial roles and must be considered holistically. The state of the art for control of dynamic systems is discussed in Sec. II-A. Sec. II-B explores Reinforcement Learning (RL) as a potential approach for solving the DSE.

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

Subsequently, Sec. II-C focuses on system identification, and in Sec. II-D, the research gap and contributions are presented.

A. Dynamic Control

Two categories are defined to classify the control approaches available in the literature: (1) generic control and (2) topology-specific control. Generic control focuses on optimal control from a functional point of view and the analysis of functional dependencies or synergies between subsystems. Goal is the identification of new system designs as well as the break down of system requirements to component requirements. Furthermore, within this context, it is essential to verify the functional safety of the actuator set to ensure compliance with safety requirements. Topology-specific control, on the other hand, focuses on the robustness of the controller, real-time capability, and functional safety of the algorithm. This class aims to achieve the best performance within a specified topology and application domain defined by requirements and fault characteristics.

An example of domain-specific generic control is [5]. The authors employ Dynamic Programming (DP) to perform a DSE and optimize the powertrain layout of an electric vehicle. However, in cases where problems involve complex state and action spaces, the computation of the cost function becomes computationally demanding or impractical, particularly for continuous state and action spaces [6]. Additionally, [5] is centered on a limited design space and does not encompass failure cases as required by ISO 26262 standard [3] for safety-critical functions. Topology-specific control can be effectively addressed by employing algorithms with a receding horizon approach. Some studies, such as [7], focus on stabilizing the vehicle at the limits of handling using control algorithms. Other research work optimizes energy consumption while taking trajectory tracking into account [8]. However, these studies focus on specific topologies and do not incorporate considerations for system failures. Studies such as [9], [10] address lateral vehicle dynamics, including compensation for steering failures through torque vectoring. Additionally, [11] focuses on optimizing automotive drive systems while considering machine derating. For this purpose, reference trajectories must be specified as well as due to redundancy of the over-actuated system, control allocation algorithms must usually be utilized to provide a unique solution, which hinders automation of the DSE. Nevertheless, the aforementioned studies provide compelling evidence that DSE can lead to improved system performance, such as reduced energy consumption, improved trajectory tracking performance, and increased system availability.

B. Reinforcement Learning for Optimal Control

With RL as a promising method for the automation of control strategies derivation [12]–[14], its suitability for exploring a large design space is further evaluated in this work. Typically, the learning period is in the range of several minutes to hours depending on the problem and the used algorithm [15]–[17]. Since for RL, unlike supervised learning, the data is generated during the learning phase, it is of particular importance that the data is generated rapidly and that the RL algorithm is very data efficient. To enable efficient learning of the control strategy, a holistic view on the interaction loop is necessary. The basic RL interaction loop is shown in Fig. 1. The controller applies controls $u_k \in U_k(x_k)$ based on the given state x_k and policy π on the environment. The environment transfers the inputs to the next state x_{k+1} using the state transition function $x_{k+1} = f_k(x_k, u_k)$ and returns a reward for each step according to $r_k = g_k(x_k, u_k, x_{k+1})$ and a terminal reward $r_N = g_N(x_N)$. The learning objective is to find the optimal policy π^* maximizing the accumulated reward for a finite horizon N problem with

$$\begin{aligned}\pi^* &= \arg\max_{\pi} J_\pi(x_0) \\ &= \max_{u} \left(g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, x_{k+1}) \right).\end{aligned} \tag{1}$$

Benchmarks for suitable RL agents without a focus on modeling system dynamics can be found in [18] for continuous control problems and in [19] for real-world problems. [20], on the other hand, evaluates model-based reinforcement learning algorithms. For the consideration of the algorithms we refer to the aforementioned publications, since benchmarking of different RL algorithms for the given problem is not considered in this study. In this study, the model-free on-policy algorithm Proximal Policy Optimization (PPO) [21] is employed. This algorithm strikes a favorable balance between sample complexity, simplicity, and wall-time and has demonstrated strong performance in [17], among others.

C. Plant model

Besides the agent, the environment is an integral part of RL. Ideally, the environment should exhibit the Markov property [22], allowing for the utilization of any initial state without considering past history. This eliminates the need for initializing starting states and significantly reduces the time required. Plant modeling can generally be divided into three approaches: (1) white-box, (2) black-box, and (3) gray-box modeling [23]. White-box modeling relies on first principles, i.e. physical laws, black-box modeling focuses on capturing the input-output relationship without prior knowledge, and gray-box modeling combines some knowledge of the system with empirical data. Numerous publications address different aspects and levels of granularity when it comes to modeling vehicle dynamics, providing a spectrum of approaches that fall within the gray-box approach. Some studies focus on first principles modeling [24], [25], while others employ black box models [26] and comparing the accuracy of different modeling approaches with ground truth data [27]. [28] models the tire using a Multi Layer Perceptron (MLP) and compares it to a Pacejka tire model, while [29] incorporates an approximation of the tire compensating limitations or shortcomings in the original Pacejka model for high dynamics. [30] models the longitudinal acceleration of a vehicle using an MLP network

architecture based on physical effects i.e. rolling and gradient resistance or actor dynamics. In addition, data from previous states and controls are fed back according to the Nonlinear Autoregressive Exogenous Model (NARX) approach. However, their study only covers longitudinal dynamics. [31] and [32] explore MLP modeling for low and high dynamics of longitudinal and lateral vehicle dynamics, respectively. [33] compares the timing requirements of MLP with a dynamic nonlinear single-track vehicle model, focusing on MLP design without considering the RL algorithm or timing requirements. In summary, the focus of existing publications is on the development of models for controller design rather than for a data efficient RL.

[34] considers acceleration of the plant model and an informal hyperparameter search of the RL agent to reduce the wall-time for a fixed set of training episodes. However, the different modeling approaches are not compared in terms of time requirements and no hyperparameter optimization is performed for the RL algorithm.

D. Research Gap and Contribution

The existing publications focus mainly on the accuracy of different modeling approaches while there is a gap in existing publications comparing these approaches in terms of runtime as well as the interaction of the plant models with the controller in an RL framework. Furthermore, there has been no analysis of how RL can be used to determine generic control within a DSE including failure cases. This paper aims to close these gaps and aims to expedite the learning process of a single policy for a specific design and scenario, with the objective of evaluating a variety of architectures within a limited time frame. Additionally, emphasis is placed on plant modeling and the careful selection of an appropriate hyperparameter set for the RL algorithm. This study is based on the premise that no prior knowledge is involved and that learning can occur purely from the interaction between the controller and the plant. Thus, methods that incorporate an existing base policy, such as transfer learning, are not addressed in this work. Furthermore, multitask learning for different topologies and scenarios is not the focus of this study, so a policy is learned for each combination of topology and scenario as well as failure cases.

III. Problem Definition

As this study investigates the potential of learning a single policy for a particular design and scenario, rather than conducting a comprehensive DSE, the methodology is demonstrated through several illustrative examples. These examples are introduced in the following.

Fig. 1 depicts the interaction loop between the controller and the plant for the specified Optimal Control Problem (OCP) exemplified on a vehicle architecture. The state vector is defined by $x_k = [d_{x,k}, d_{y,k}, \psi_k, v_{x,k}, v_{y,k}, \dot{\psi}_k]'$ with global vehicle longitudinal/lateral positions ($d_{x,k}/d_{y,k}$), vehicle longitudinal/lateral velocities in global coordinates ($v_{x,k}/v_{y,k}$) and yaw angle/rate ($\psi_k/\dot{\psi}_k$). The exemplary design variant considered in this study consists of single-wheel steering

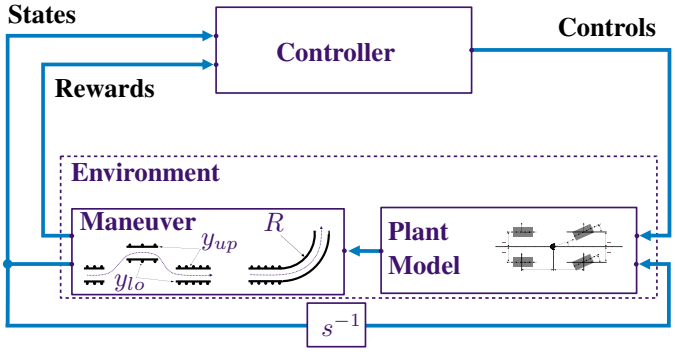


Fig. 1: Problem related controller and plant interaction scheme.

actuators on the front axle and single-wheel motors on the rear axle. Thus, the control vector $u_k = [\delta_{l,k}, \delta_{r,k}, T_{l,k}, T_{r,k}]$ consists of individually controlled left and right front steering angles $\delta_{l,k}$ and $\delta_{r,k}$ as well as individually controlled left and right wheel torques $T_{l,k}$ and $T_{r,k}$ at the rear wheels. The environment is divided into the elements of plant model and maneuver with the appropriate cost function in the subsequent subsections.

A. Plant Model

In this study, a comparison is conducted between four different model variants benchmarked against data generated by a base implementation of the state space model solved with a variable-step Dormand-Prince solver. The model architectures are shown in Fig. 2. The variant depicted in Fig. 2a utilizes a nonlinear state space model based on a dynamic two-track model, incorporating a Pacejka tire model to represent tire forces and torques. The only difference compared to the reference model is the solver. Instead of the variable step solver a Runge Kutta (RK) 4 fixed step solver is used to strike a balance between result accuracy and computational speed. Since the model operates with the same number of solver steps, parallelization can be easily implemented. Another variant (Fig. 2b) replaces the Pacejka tire model with an MLP for tire modeling. These state space models can be categorized as light gray-box models, as they integrate data-driven modeling of tires into momentum or angular momentum theorems. Furthermore, two MLPs are employed to model the entire vehicle dynamics for comparison. One MLP (Fig. 2c) is analogous to the nonlinear state space model and predicts the next state based on the current state and inputs. This variant falls into the category of dark gray-box models, as it incorporates domain knowledge for the data generation. The other machine learning approach corresponds to NARX (Fig. 2d), which incorporates domain knowledge in the form of time-delayed control signals and time-delayed system outputs as well as their derivatives. This approach allows the network to effectively learn the nonlinear system and its underlying differential equations.

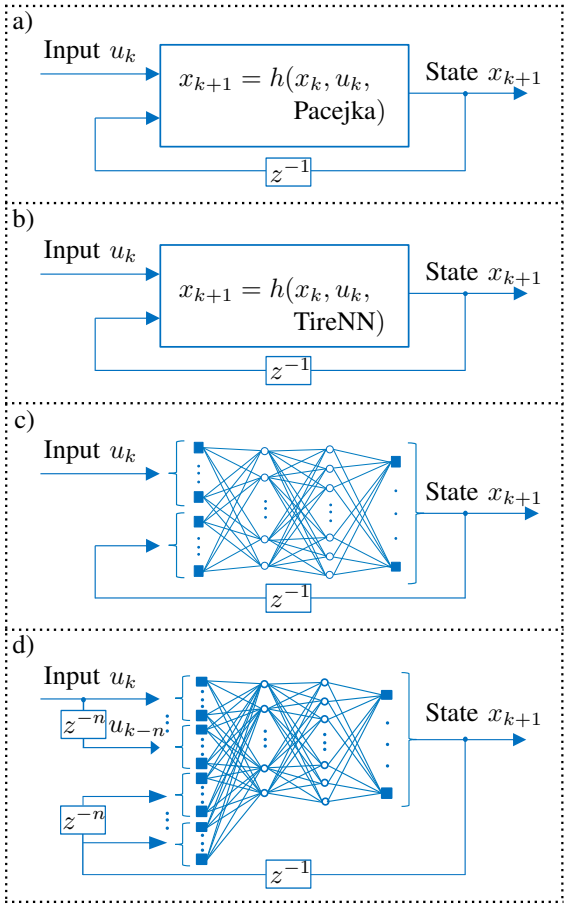


Fig. 2: Model architectures.

B. Scenarios

The study incorporates two maneuvers: the run-in to steady-state circular travel and the Double Lane Change (DLC) maneuver. The trajectories are shown in Fig.1 in the block maneuver for illustration.

The objective of the run-in to steady-state circular travel maneuver is to achieve steady-state in cornering on a track with a target radius R starting from straight driving at constant speed [35]. Depending on the cost function, achieving the desired outcome can be accomplished through various control signals due to the degrees of freedom of the system due to over-actuation. In addition, the maneuver can be used to investigate the path following ability of the vehicle involving system failures such as the sudden loss of a wheel-individual drive or brake torque or the malfunction of a power steering actuator.

The DLC maneuver is used to evaluate a controller's ability to track trajectory and stabilize vehicle motion [36]. The lower and upper boundary of a DLC is defined by $y_{lo,k} = f_{lo}(d_{x,k})$ and $y_{up,k} = f_{up}(d_{x,k})$ respectively, based on the longitudinal position. Furthermore, the maneuver can also serve as a mean to assess the vehicle's path following ability in dynamic scenarios.

C. Control Cost Function

The optimization objective is formulated in (2) and can be divided into two main parts. The primary part ϵ_1 is the path following accuracy, which aims to ensure that the vehicle remains within the lane limits of the DLC maneuver or follows the desired radius of the steady-state circular travel. Secondary components $\underline{\epsilon}_2$ include maintaining a consistent driving speed, a uniform driving torque, and steering angle distribution. In addition, the shortest path within the boundaries is also tracked to provide information on energy consumption. This objective can also be extended to incorporate the energy consumption of all involved actuators in future work. The two objectives are weighted by the corresponding weights w_1 and $\underline{w}_2$. Since there is no dedicated final state, the terminal penalty of (1) is set to zero.

$$J_1^*(x_0) = \min_{\delta_{i,k}, T_{i,k}} \left[\sum_{k=0}^{N-1} [w_1 \cdot \epsilon_1 + \underline{w}_2 \cdot \underline{\epsilon}_2] \right] \tag{2a}$$

s. t.

$$x_{k+1} = f(x_k, \delta_{i,k}, T_{i,k}), \text{with } i \in l, r \tag{2b}$$
$$-\delta_{max} \leq \delta_{i,k} \leq \delta_{max} \tag{2c}$$
$$-T_{max} \leq T_{i,k} \leq T_{max} \tag{2d}$$
$$x_0 = [0, 0, 0, v_0, 0, 0]' \tag{2e}$$

IV. Implementation and Analysis of Plant Model

This section focuses on the development of plant models and the analysis of their performance. This includes the implementation and evaluation of various models representing the dynamics of the system under study. All models considered in this study are derived from the base model. This is a nonlinear three degree of freedom (longitudinal and lateral motion as well as yaw rotation) state space model using a Pacejka tire model for the tire force and torque calculation solved by a variable step Dormand Prince solver. The nonlinear state space model with the Pacejka tire incorporates the same dynamics, but is suitable for parallelization and implemented in Python using Numba [37]. The data for all MLPs are generated by using the base model. Thus, the data for the tire MLP are generated based on the Pacejka implementation and are sampled with uniformly distributed inputs from the sub model. The MLPs for the whole vehicle dynamics, on the other hand, are based on trajectories. These trajectories are created during RL with the base model by recording the state transitions for a set of controls. The trajectories are decomposed into individual state transitions depending on the history for training the MLPs. Hyperparameter optimization is performed for all models to determine the optimal number of layers and the number of nodes per layer. The criterion used for hyperparameter optimization is the closed loop performance, which means that the models are validated with respect to all trajectories. The Mean Absolute Error (MAE) is used as the validation metric, since this metric gives less weight to individual outliers in the data and evaluates all standardized elements in the state vector equally. The results of the best MAE from hyperparameter

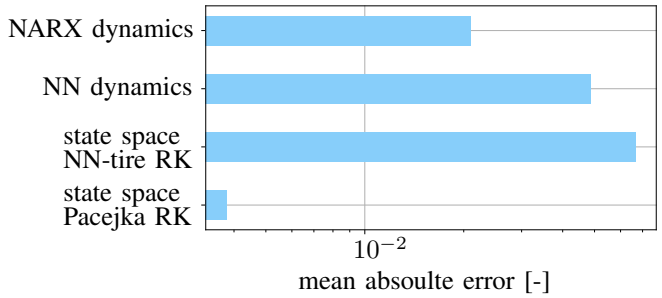


Fig. 3: Mean absolute errors of the state predictions compared to the base implementation.

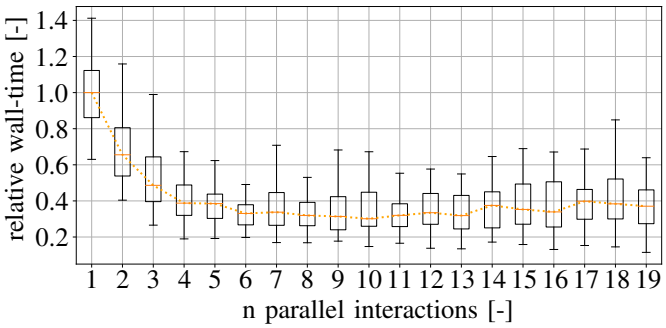


Fig. 4: Comparison of RL learning duration depending on the number of parallel environment interactions.

optimization are presented in Fig. 3 referenced to the base model (state space Dormand Prince).

For determining an appropriate level of parallelization, the MLP dynamics model with the DLC maneuver and the hyperparameters of Table I are wrapped in the RL loop. Thereby, the time until the convergence of the policy optimization was measured 50 times for each number of parallelized state transitions. The run-times of the experiments were measured on an Intel® Xeon® E3-1270 v6 with 8 logical processors. In addition, both the training process and data acquisition were performed on the Central Processing Unit (CPU), since the network for policy and value function are very small, resulting in little benefit from Graphics Processing Unit (GPU) computation, and data transfer between Random-Access Memory (RAM) and GPU takes considerable additional time. The results are depicted in Fig. 4. The shortest learning time is achieved using ten parallel interactions. Parallelization is limited by factors such as the number of CPU cores and memory usage, which restrict the number of parallel instances that can be effectively simulated. To overcome these limitations [17] proposes utilizing end-to-end training on GPU, which can offer improved performance and efficiency compared to pure CPU-based or CPU-GPU-mixed training methods. For comparison to existing publications, the evaluation times of the individual models are considered for a single as well as parallel state transitions. In parallel, ten trajectories are generated simultaneously, starting from the same initial state as defined in (2), but with different action sequences.

Fig. 5 illustrates the speed improvement ratio of the four models compared to the base implementation. The run times were calculated as the average of 100000 runs. All four modeling approaches are divided into *single* and *parallel* respectively. *Single* means that one state transition is performed after another, whereas *parallel* indicates that ten state transitions are performed in parallel and the proportion of one transition is represented in the bar chart. For each model, the first function call is not considered because the code is compiled only on the first call. Using the Numba-Jit decorator in compilation mode results in the best performance, as the function is compiled to run independently without the involvement of the Python interpreter.

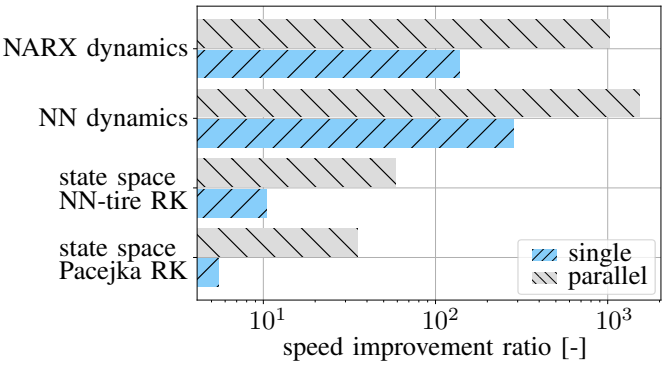


Fig. 5: Speed improvement ratio for the four models compared to the base implementation.

In summary, the parallelizable state space model with Pacejka tire shows the best accuracy, suffering only from the errors of simplified numerical integration. It gains a slight speed advantage of about 5.5 times over the base implementation for the single state transition by compilation and less solver steps, but becomes about 35 times faster by parallelization. The approximation of the Pacejka tire model does not provide a significant speed advantage compared to the base implementation. However, it is still valuable, particularly in areas where the Pacejka model exhibits poorer performance. The approximation helps to improve the accuracy and reliability of tire modeling, compensating for any limitations or shortcomings in the original Pacejka model, as shown in [29]. Significant speed advantages are provided by the two MLP models in approximating the entire vehicle dynamics. Since the training data are generated from the state space model and therefore each state has the Markov property, the history is not advantageous in the NARX approach. The NARX architecture or recurrent networks in general become relevant when the data comes from real field data or from a more complex vehicle dynamics model based on multi-body modeling such as CarMaker or ADAMS/Car when transient transitions occur. Especially in terms of parallelization, the two models are up to 1500 times faster than the base implementation.

TABLE I: Parameter of RL algorithm.

Parameter	Value
batch size	10
update frequency	5
learning rate actor	1e-4
learning rate critic	1e-2
subsampling fraction	0.3
multi step actor	40
multi step critic	80
likelihood ratio clipping	0.1
discount factor	0.99
parallel interactions	10
exploration	0.01
actor/ baseline NN	2 layers each 32 nodes

V. Evaluation of RL loop

This section considers the closed-loop system, incorporating model in the RL loop. The interaction between the plant models and the RL algorithm is examined, studying how the RL algorithm optimizes the control strategies within the closed-loop system in normal and failure mode. For solving the RL optimization problem a PPO algorithm [21] from the Python framework Tensorforce [38] is used. Based on the parallelized state space model with Pacejka tire model, a hyperparameter optimization is performed. The optimal parameters for both scenarios in terms of optimal cost and shortest learning time are shown in Table I.

Fig. 6 illustrates the learning process for the run-in to steady-state driving maneuver without failures. The gray envelope curve represents the reward collected during training, while the blue curve represents the reward collected during greedy testing. The orange dashed line indicates the best reward achieved throughout the training process. During training, the reward appears lower due to sub optimal actions resulting from Gaussian noise induced by the mixture density network of the policy. Nevertheless, irrespective of the initial parameter set, the reward consistently converges towards the optimal value of 1.0 throughout the learning process. It should be noted that the optimal reward of 1.0 cannot be fully achieved due to the influence of the run-in and the associated deviation of the yaw rate.

Fig. 7 depicts the curves of five repetitions for the DLC maneuver without failure (left) and with a loss of function failure in the left steering actuator (right). Based on the cost function, which requires equal torque on the left and right side, the failure of the left actuator must be compensated by the remaining steering actuator. This may include, for example, the energy consumption of the actuators to evaluate the efficiency of the system, or, for example, the maximum of control variables and control speeds to support the sizing of the actuators within the DSE. Additionally, it is evident that higher power output is required in the presence of a failure, as the wheel with the faulty actuator acts as a driving resistance. The lower part of the plot illustrates the aggregated states, where it can be observed that the longitudinal velocity of the vehicle remains constant, as any deviation from the desired velocity is penalized by the cost function. In addition, by looking at

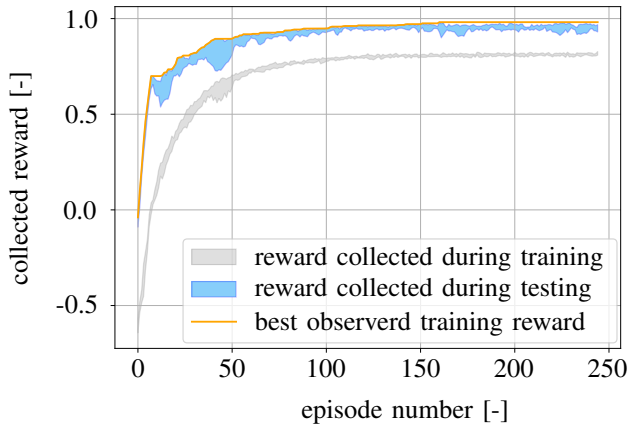


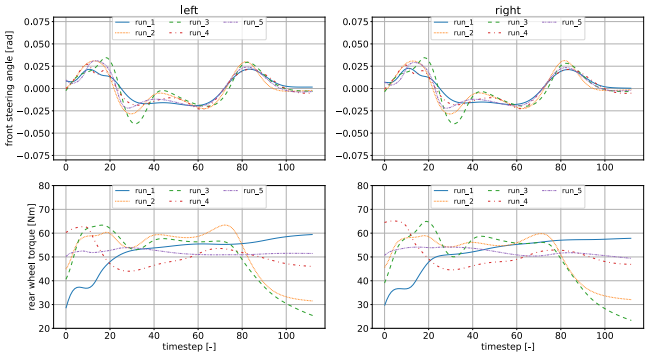
Fig. 6: Illustration of learning process of five random parameter sets.

the trajectories of the slip angles and the yaw angles, it can be observed that the handling of the vehicle is the same in both cases. Therefore, it is evident that the algorithm is capable of accommodating varying numbers of actuators, which is a crucial aspect within the scope of the application. Minor variations in the curve profiles can be attributed to differences in the initial parameterization of the agents.

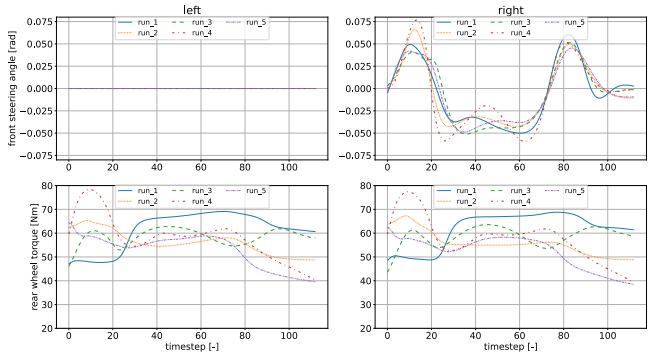
Table II serves as a summary of the study. It presents the average reward collected, the number of episodes, and the time required for the agent to learn for all scenarios and failure cases, as well as the four models across five evaluations. It is noteworthy that the duration of the run-in cornering maneuver is limited to a maximum of 5 minutes, whereas the DLC maneuver can take up to 19 minutes. This discrepancy arises from the fact that the run-in cornering maneuver simulates a maximum of 50 steps per episode, while the DLC maneuver requires approximately 120 steps. The rewards obtained for the DLC maneuver are slightly lower due to the inability to achieve the shortest path based on the initial velocity and the dynamics of the employed model [13]. The advantage in terms of speed offered by the models is limited since the duration of training and action generation depends on the RL algorithm, resulting in only a two- to threefold speed advantage provided by the models. Nevertheless, these results demonstrate that, under a specific set of hyperparameters for the RL algorithm, various scenarios and failure cases can be addressed with different models.

VI. Conclusion

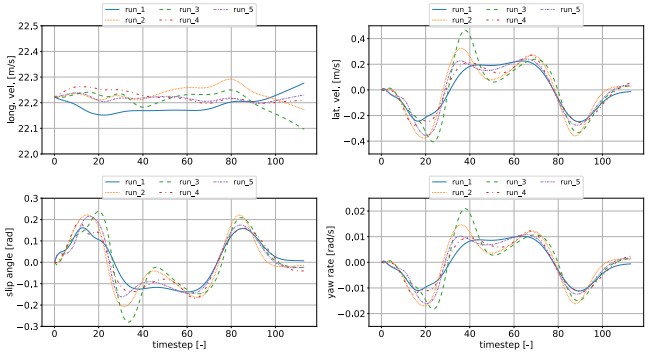
This study provides further insights and advancements in the field of DSE and environment modeling for RL methods based on the example of vehicle dynamics. By comparing different modeling approaches, the study contributes to the evaluation of vehicle performance and extends the state-of-the-art in temporal evaluation of vehicle dynamics models. This study demonstrated the applicability of RL for the automation of a design space exploration. The approach of training an agent for each topology, failure case, and maneuver was proven to be feasible, particularly in design spaces involving smaller



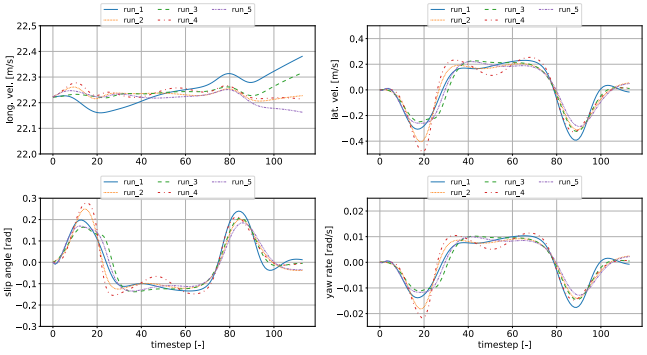
(a) Conrols for DLC without failure case.



(b) Conrols for DLC with failure of steering FL.



(c) States for DLC without failure case.



(d) States for DLC with failure of steering FL.

Fig. 7: Illustatration of trajectories for DLC with and without failure case determined with state space with Pacejka tire model.

TABLE II: RL optimal cost as well as learning epochs and times for each model variant.

		state space Pacejka RK	state space tire NN RK	NN dynamics	NARX dynamics
Run-in Cornering w/o failures	$\sum r$	0.9747	0.9757	0.9620	0.9747
	$\#ep$	259	278	426	285
	t	4m 19s	2m 45s	3m 17s	1m 58s
Run-in Cornering w/ FL steering failure	$\sum r$	0.9761	0.9753	0.9714	0.9784
	$\#ep$	327	320	393	363
	t	5m 12s	3m 24s	2m 55s	2m 42s
Run-in Cornering w/ RR torque failure	$\sum r$	0.9778	0.9764	0.9721	0.9765
	$\#ep$	322	274	254	290
	t	4m 57s	2m 39s	1m 37s	1m 57s
DLC w/o failures	$\sum r$	0.9226	0.9229	0.9150	0.9213
	$\#ep$	491	556	488	377
	t	18m 52s	16m 9s	8m 16s	6m 33s
DLC w/ FL steering failure	$\sum r$	0.9246	0.9237	0.9162	0.9262
	$\#ep$	503	498	504	640
	t	16m 38s	19m 33s	9m 27s	13m 17s
DLC w/ RR torque failure	$\sum r$	0.9244	0.9228	0.9255	0.9238
	$\#ep$	458	460	532	443
	t	12m 18s	13m 55s	9m 39s	7m 35s

design spaces. This approach enables an automated exploration of different configurations, allowing for a more thorough analysis and comparison of system behavior. Nevertheless, especially in high-dimensional design spaces, there is still further methodological potential for carrying out DSE in a time-efficient manner.

REFERENCES

[1] O. Burkacky, J. Deichmann, and J. P. Stein, "Automotive software and electronics 2030 - mapping the sectors future landscape," 2019.

[2] H. de Carvalho Pinheiro and M. Carello, "Design and validation of a high-level controller for automotive active systems," *SAE International Journal of Vehicle Dynamics, Stability, and NVH*, vol. 7, no. 10-07-01-0006, 2022.

[3] "Road vehicles - functional safety," International Organization for Standardization, Geneva, CH, Standard ISO 26262:2018(E), 2018.

[4] C. Becker, A. Nasser, F. Attioui, D. Arthur, A. Moy, and J. Brewer, "Functional safety assessment of a generic electric power steering system with active steering and four-wheel steering features," National Highway Traffic Safety Administration, Washington, DC, Technical Report DOT HS 812 575, 2018.

[5] M. Vaillant, M. Eckert, and F. Gauterin, "Energy management strategy to be used in design space exploration for electric powertrain optimization," in *Ninth International Conference on Ecological Vehicles and Renewable Energies (EVER)*. IEEE, 2014.

[6] D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[7] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2013.

[8] S. Koehler, A. Viehl, O. Bringmann, and W. Rosenstiel, "Energy-efficiency optimization of torque vectoring control for battery electric vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 59–74, 2017.

[9] K. Polmans, "Torque vectoring as redundant steering for automated driving or steer-by-wire." Springer Fachmedien Wiesbaden, 2014, pp. 163–177.

[10] A. Kirli, Y. Chen, C. E. Okwudire, and A. G. Ulsoy, "Torque-vectoring-based backup steering strategy for steer-by-wire autonomous vehicles with vehicle stability control," vol. 68, no. 8, pp. 7319–7328, 2019.

[11] O. Wallscheid and J. Bocker, "Derating of automotive drive systems using model predictive control," in *IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*. IEEE, 2017.

[12] K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever, Eds., *Handbook of Reinforcement Learning and Control*. Springer International Publishing, 2021.

[13] P. Hoffmann, K. Gorelik, and V. Ivanov, "Comparison of reinforcement learning and model predictive control for over-actuated systems," in *15th International Symposium on Advanced Vehicle Control (AVEC '22)*, no. Th01C-01. JSAE, 2022, pp. 562–567.

[14] A. Aksjonov and V. Kyrki, "A safety-critical decision-making and control framework combining machine-learning-based and rule-based algorithms," *SAE International Journal of Vehicle Dynamics, Stability, and NVH*, vol. 7, no. 10-07-03-0018, 2023.

[15] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Durr, "Super-human performance in gran turismo sport using deep reinforcement learning," vol. 6, no. 3, pp. 4257–4264, 2021.

[16] L. Puccetti, A. Yasser, C. Rathgeber, A. Becker, and S. Hohmann, "Speed tracking control using model-based reinforcement learning in a real vehicle," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021.

[17] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," 2021.

[18] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," 2016.

[19] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 2018, pp. 561–591.

[20] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," 2019.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[22] R. Sutton, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning series. Cambridge, Massachusetts London, England: The MIT Press, 2018.

[23] O. Nelles, *Nonlinear System Identification*. Springer International Publishing, 2020.

[24] B. Heißing, M. Ersoy, and S. Gies, Eds., *Fahrwerkhandbuch*. Springer Fachmedien Wiesbaden, 2013.

[25] D. Schramm, M. Hiller, and R. Bardini, *Modellbildung und Simulation der Dynamik von Kraftfahrzeugen*. Springer Berlin Heidelberg, 2018.

[26] S. J. Rutherford and D. J. Cole, "Modelling nonlinear vehicle dynamics with neural networks," *International Journal of Vehicle Design*, vol. 53, no. 4, p. 260, 2010.

[27] M. D. Lio, D. Bortoluzzi, and G. P. R. Papini, "Modelling longitudinal vehicle dynamics with neural networks," *Vehicle System Dynamics*, vol. 58, no. 11, pp. 1675–1693, 2019.

[28] L. C. Sousa and H. H. V. Ayala, "Nonlinear tire model approximation using artificial neural networks," in *Procedings do XV Simpósio Brasileiro de Automação Inteligente*. SBA Sociedade Brasileira de Automática, 2021.

[29] F. Djeumou, J. Y. M. Goh, U. Topcu, and A. Balachandran, "Autonomous drifting with 3 minutes of data via learned tire models," 2023.

[30] S. S. James, S. R. Anderson, and M. D. Lio, "Longitudinal vehicle dynamics: A comparison of physical and data-driven models under large-scale real-world driving conditions," *IEEE Access*, vol. 8, pp. 73 714–73 729, 2020.

[31] X. Nie, C. Min, Y. Pan, K. Li, and Z. Li, "Deep-neural-network-based modelling of longitudinal-lateral dynamics to predict the vehicle states for autonomous driving," *Sensors*, vol. 22, no. 5, p. 2013, 2022.

[32] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, 2019.

[33] F. Hegedüs, P. Gáspár, and T. Bécsi, "Fast motion model of road vehicles with artificial neural networks," *Electronics*, vol. 10, no. 8, p. 928, 2021.

[34] Y. Berthelot, "A journey towards faster reinforcement learning," Towards Data Science, 2021. [Online]. Available: https://towardsdatascience.com/a-journey-towards-faster-reinforcement-learning-1c97b2cc32e1

[35] "Road vehicles - vehicle dynamics test methods - part 1: General conditions for passenger cars," International Organization for Standardization, Geneva, CH, Standard ISO 15037-1:2019(E), 2019.

[36] "Passenger cars - test track for a severe lane-change manoeuvre - part 1: Double lane-change," International Organization for Standardization, Geneva, CH, Standard ISO 3888-1:2018(E), 2018.

[37] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.

[38] A. Kuhnle, M. Schaarschmidt, and K. Fricke, "Tensorforce: a tensorflow library for applied reinforcement learning," Web page, 2017. [Online]. Available: https://github.com/tensorforce/tensorforce