Preprints of the 23rd IFAC World Congress
Busan, Republic of Korea, August 23-28, 2026

Learning-based LPV control for autonomous vehicles

Dániel Fényes *, **Balázs Németh** *,**, **Péter Gáspár** *,**

* *HUN-REN Institute for Computer Science and Control (SZTAKI), Kende u. 13-17, H-1111 Budapest, Hungary, Kende u. 13-17, H-1111 Budapest, Hungary. E-mail: [daniel.fenyes;balazs.nemeth;peter.gaspar]@sztaki.hu*
** *Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics, Stoczek u. 2, H-1111 Budapest, Hungary E-mail: [balazs.nemeth;peter.gaspar]@kjk.bme.hu*

Abstract:
The paper presents a novel combination method for integrating an LPV-based controller with reinforcement learning. The main idea behind the combination is that the agent learns the scheduling parameters of the LPV observer to cope with the unmodeled or uncertain dynamics of the considered system. The RL aims to simultaneously minimize the error of the observer and the tracking performance of the controller. In this way, the exploration of the nonlinear system and the stability of the closed-loop system can be guaranteed. The proposed method is validated through two vehicle-oriented control problems, namely the trajectory tracking of ground vehicles and traction control of trains.

Keywords: Reinforcement learning, LPV system, autonomous vehicle

1. INTRODUCTION AND MOTIVATION

Linear Parameter-Varying (LPV) control is known as a powerful framework for dealing with nonlinear and time-varying systems by introducing scheduling parameters, which reflect on the nonlinear dynamics of the considered system. LPV synthesis techniques, including gain-scheduling and parameter-dependent Lyapunov approaches, provide stability and performance guarantees while maintaining the simplicity of the control design Mohammadpour and Scherer [2012]. However, the practical application of the LPV control has some difficulties. The main problem is the measurement of the scheduling parameters. In many cases, the scheduling parameter is not directly measurable thus an estimator or observer algorithm must be applied. These algorithms may have a significant uncertainty, which degrades the performance level of the LPV controller.

In parallel, reinforcement learning (RL) has achieved remarkable progress in both decision-making and control, offering data-driven policies capable of handling high-dimensional, nonlinear, and uncertain environments without requiring system models Sutton and Barto [2015], Recht [2019]. RL-based control has been successfully applied to robotics, autonomous systems, and complex process control, benefiting from policy gradient methods, actor-critic architectures, and deep representation learning Tang et al. [2024]. Nevertheless, RL-based controllers often lack proven guarantees on stability and safety that are essential in control engineering applications where constraint satisfaction and robustness cannot be compromised.

These advantages and drawbacks have motivated recent interest in hybrid solutions combining the model- and RL-based approaches. Methods such as safe RL, learning-based adaptive control, and RL-aided gain scheduling have been shown to improve performance while maintaining stability margins Timmerman et al. [2024], Yeh and Yang [2021]. Another approach is to use the RL-agent only for control selections, and the theoretical guarantees are given by the classical control solutions, see in Fenyes et al. [2025b], Fenyes et al. [2025a]. However, the integration of RL with LPV control remains relatively unexplored. In particular, the problem of learning the scheduling parameter itself, instead of relying on handcrafted or model-derived scheduling maps, has not been thoroughly addressed.

This paper proposes a combined LPV-RL control framework in which a reinforcement learning agent dynamically computes the LPV scheduling parameter based on the estimation error of the observer and performance objectives. In this way, the LPV-based control can use an accurate model, which can improve its performance level.

[1] The paper was funded by the National Research, Development and Innovation Office (NKFIH) under NKKP Grant Agreement No. STARTING 153675. The research was also supported by the National Research, Development and Innovation Office through the project 'Digital-Twin-Driven Resilience Modeling Framework for International Rail Freight Corridors' (NKFIH: 2025-1.2.4-TÉT-2025-00039. The work of Dániel Fényes was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

The contributions of this work are threefold. First, a general framework in which the RL agent serves as a scheduling-law generator for LPV controllers, enabling data-driven enhancement of classical gain-scheduled control, is formulated. Second, a training architecture is proposed, which incorporates the output of the LPV observer and the performance level and stability of the LPV controller. Third, the proposed method is validated through a vehicle-oriented control problems, namely: trajectory tracking of a vehicle, and traction control of trains showing improved performance and adaptability compared to conventional scheduling strategies.

The rest of the paper consists of the following sections. First, a short introduction is given to the applied methods, such as the LPV framework, and the used RL algorithm see in Section 2. Then, the scheduling approximation-oriented training is detailed in Section 3. Vehicle-oriented applications of the proposed method are presented in Section 4. Finally, the conclusion of the paper is summarized in Section 5.

2. APPLIED METHODS AND ALGORITHMS

In this section, the applied methods are briefly presented. Starting with the LPV modeling approach, followed by the LPV-based control and observer design. Then, the approximation of the scheduling parameter based on reinforcement learning is detailed.

2.1 Linear Parameter Varying system

The linear parameter varying approach can handle nonlinear dynamics by using a set of linear systems. The connection among the linear systems is described by the scheduling parameters Toth [2010]:

$$\Sigma : \begin{cases} \dot{x}(t) = A(\rho(t))\, x(t) + B(\rho(t))\, u(t), \\ y(t) = C(\rho(t))\, x(t) + D(\rho(t))\, u(t), \end{cases} \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control input, $y(t) \in \mathbb{R}^p$ is the measured output, and $\rho(t) \in \mathbb{R}^s$ is the scheduling parameter vector.

The scheduling parameters are:

$$\rho(t) \in \mathcal{P} \subset \mathbb{R}^s, \qquad \dot{\rho}(t) \in \dot{\mathcal{P}}. \tag{2}$$

The scheduling parameters are assumed to be rate-limited. The actual operating point is determined by the values of the scheduling parameters. The system can be transformed into an affine form:

$$A(\rho) = \sum_{i=1}^{N} \mu_i(\rho)\, A_i, \qquad B(\rho) = \sum_{i=1}^{N} \mu_i(\rho)\, B_i, \tag{3}$$

with weights $\mu_i(\rho) \geq 0$ and $\sum_{i=1}^{N} \mu_i(\rho) = 1$.
In general form, the LPV system with unique scheduling parameter for all nominal matrix values can be described as:

$$A(\rho) = A + \Delta = \begin{bmatrix} a_{11} + \Delta_{11} & a_{12} + \Delta_{12} & \cdots & a_{1n} + \Delta_{1n} \\ a_{21} + \Delta_{21} & a_{22} + \Delta_{22} & \cdots & a_{2n} + \Delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} + \Delta_{n1} & a_{n2} + \Delta_{n2} & \cdots & a_{nn} + \Delta_{nn} \end{bmatrix}.$$

$$B(\rho) = B + \Delta^B = \begin{bmatrix} b_1 + \Delta_1^B \\ b_2 + \Delta_2^B \\ \vdots \\ b_n + \Delta_n^B \end{bmatrix}. \tag{4}$$

In this form, all parameters have a nominal value (a_{ii}, b_i) and an uncertain value (Δ_{ii}), which represents the unmodeled/uncertain dynamics of the system. This general form is used when the parameters depend on a common scheduling structure. Note that the uncertain terms Δ_{ij} are implicit functions of the scheduling vector $\rho(t)$.

2.2 LPV control and observer design

In general, LPV control and observer design are carried out in parallel. However, they can also be designed separately. In this paper, the objective is to determine the values of the scheduling parameters using a reinforcement learning (RL) agent to identify the underlying nonlinear model. The resulting model can then be employed with other control techniques, such as NMPC, making it more convenient to design the observer and controller independently.

LPV control design

Let us consider the general LPV system presented in (1). The goal is to find a controller that satisfies the prescribed performances:

$$u(t) = K(\rho(t))\, x(t) \tag{5}$$

such that the closed-loop system

$$\dot{x}(t) = (A(\rho(t)) + B(\rho(t))K(\rho(t)))\, x(t) \tag{6}$$

is stable for all admissible $\rho(t)$.

Using a parameter-dependent Lyapunov function

$$V(x, \rho) = x^\top P(\rho) x \tag{7}$$

the closed-loop stability conditions can be expressed as LMIs. For each vertex i, one requires the existence of matrices $P_i > 0$ and Y_i such that

$$\begin{bmatrix} A_i P_i + B_i Y_i + (A_i P_i + B_i Y_i)^\top & P_i \\ P_i & -\gamma I \end{bmatrix} < 0. \tag{8}$$

The gain-scheduled controller is computed from

$$K_i = Y_i P_i^{-1}, \qquad K(\rho) = \sum_{i=1}^{N} \mu_i(\rho)\, K_i. \tag{9}$$

This guarantees stability and an $\mathcal{H}_\infty$ performance level γ for all admissible scheduling trajectories $\rho(t)$, see Toth [2010]

LPV Observer Design

The observer design is similar to the control design but the goal is to minimize the error between the estimated and the real states. The observer error is defined as

$$e(t) = x(t) - \hat{x}(t),$$

where $\hat{x}(t)$ denotes the estimated state vector provided by the LPV observer.

$$\dot{e}(t) = (A(\rho(t)) - L(\rho(t))C(\rho(t)))\, e(t). \tag{10}$$

It can be achieved by finding the optimal values for $L(\rho)$ stability of the error system can be guaranteed by finding matrices $P_i > 0$ and Z_i satisfying, for each vertex i,

$$\begin{bmatrix} P_i A_i^\top + Z_i C_i^\top + A_i P_i + C_i Z_i^\top & P_i \\ P_i & -\gamma I \end{bmatrix} < 0. \tag{11}$$

The observer gains are retrieved as

$$L_i = P_i^{-1} Z_i, \qquad L(\rho) = \sum_{i=1}^{N} \mu_i(\rho)\, L_i. \tag{12}$$

This yields a parameter-varying observer guaranteeing stability and an $\mathcal{H}_\infty$ performance level γ.

2.3 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) is a model-free, off-policy, actor-critic reinforcement learning algorithm designed for environments with continuous action spaces. It combines the deterministic policy gradient framework with deep neural network function approximation Lapan [2018].

DDPG employs two neural networks: an actor and a critic network. The actor represents a deterministic policy

$$a = \pi_\theta(s), \tag{13}$$

mapping each state s to a specific action a. The critic approximates the state-action value function

$$Q_\phi(s, a), \tag{14}$$

and is trained using the Bellman target

$$y = r + \gamma Q_{\phi^-}(s', \pi_{\theta^-}(s'))\,, \tag{15}$$

where ϕ^- and θ^- denote the parameters of the target critic and target actor networks, respectively.

DDPG uses a replay buffer containing experience tuples (s, a, r, s'). Mini-batches sampled from this buffer reduce correlations between successive transitions and improve training stability. To stabilize learning, DDPG maintains target networks for both actor and critic. These are updated through soft updates:

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-, \qquad \phi^- \leftarrow \tau\phi + (1-\tau)\phi^-, \tag{16}$$

with $0 < \tau \ll 1$.

The actor network is trained using the deterministic policy gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[\nabla_a Q_\phi(s,a)\, \nabla_\theta \pi_\theta(s)\right], \tag{17}$$

where $\mathbb{E}_{s\sim D}[\cdot]$ denotes the expectation over the sampled replay buffer distribution. DDPG is therefore well suited for continuous control tasks and offers good sample efficiency due to its off-policy nature.

3. DESIGN PROCEDURE

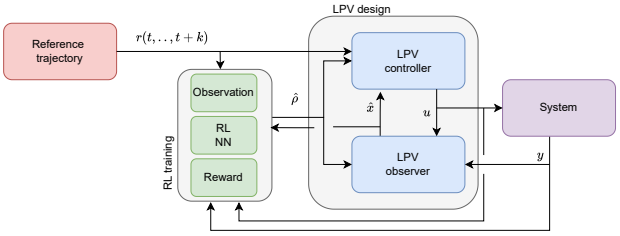


Fig. 1. Training of RL agent

The goal of the RL agent is to estimate the scheduling parameter(s) of the LPV system while the control system is operating. The RL agent works by comparing the states calculated by the observer with the actual, measured states from the system. The difference between these states is used by the agent to improve its estimation of the scheduling parameters. The estimated parameters are then forwarded to the LPV controller. In this way, the controller can use the current estimates to adjust its actions, while the system performance is also considered. This method allows the observer, controller, and RL agent to work together, which improves the overall behavior and robustness of the system while maintaining the stability of the closed-loop system. Note that during the training phase, the simulator provides the full vehicle states, which are used only for reward calculation and supervision of the RL agent. In the online implementation, only the measurable outputs are available; therefore, the LPV observer is required to reconstruct the unmeasured states needed by the controller.

The main steps of the training process are as follows:

- **Determine the nominal LPV model:** First, a nominal LPV model of the system is created. This model includes all the relevant scheduling parameters and their ranges. The nominal model is important because it provides a base for both the observer and the controller design.
- **Design the LPV controller:** Next, the controller is designed based on the desired performance of the system. The controller uses the nominal LPV model and the estimated parameters from the RL agent to ensure the system is stable and works according to the specifications.
- **Design the observer:** An observer is designed to estimate the system states in real time. The observer provides the RL agent with information about the system, which is necessary for accurate parameter estimation.

- **Select observations for the RL agent:** The inputs for the RL agent are carefully chosen. They include the states estimated by the observer, the actual measured states of the system, and the previous values of the scheduling parameters, $\rho(t-1)$. Including delayed values helps the agent understand the system dynamics.
- **Define the reward function:** The reward function guides the learning of the RL agent. A possible choice is $\mathcal{R} = w_1^T e + w_2^T z$, where e is the state error, z is a vector representing system performance, and w_1, w_2 are weights to balance the importance of state accuracy and performance. The reward function ensures that the agent learns to improve both estimation and control performance.
- **Train the RL agent:** Finally, the agent is trained using different operational scenarios. These scenarios cover a wide range of possible conditions and disturbances that the system may experience. Training in this way helps the RL agent to generalize and provide accurate scheduling parameter estimates in real operation.

The training scheme is illustrated in Figure 1. This figure shows the flow of information between the observer, controller, and RL agent during the training. It also highlights how the RL agent learns from the errors and system performance to improve the estimation of scheduling parameters while maintaining control quality.

4. VEHICLE-ORIENTED EXAMPLE

In this section, the proposed combined learning and control design technique is validated through two vehicle-oriented control problems: trajectory tracking and traction control.

4.1 Trajectory tracking

The simulation environment is based on the nonlinear bicycle model combined with the Pacejka tire model presented in Pacejka [2004]. The Pacejka parameters were selected to represent a standard passenger vehicle operating under nominal dry-road conditions. It consists of two main equations: lateral acceleration and yaw-motion.

$$\begin{cases} \dot{v}_y = \dfrac{1}{m}\left(F_{yf}\cos\delta + F_{yr}\right) - v_x r + \dfrac{1}{m}F_{xf}\sin\delta, \\ \dot{r} = \dfrac{1}{I_z}\left(l_f F_{yf}\cos\delta - l_r F_{yr} + l_f F_{xf}\sin\delta\right), \end{cases} \tag{18}$$

where m is the mass, I_z the yaw inertia, and l_f, l_r the distances from the vehicle center of mass to the front and rear axles, δ is the steering angle, r denotes the yaw-rate, v_y, v_x are the longitudinal and lateral velocities.

The tire slip angles can be computed as:

$$\begin{aligned} \alpha_f &= \arctan\left(\frac{v_y + l_f r}{v_x}\right) - \delta, \\ \alpha_r &= \arctan\left(\frac{v_y - l_r r}{v_x}\right). \end{aligned} \tag{19}$$

The lateral tire forces F_{yf}, F_{yr} are described by the Pacejka Magic Formula, see Pacejka [2004]:

$$F_y(\alpha) = D\sin\big(C\arctan\big(B\alpha - E(B\alpha - \arctan(B\alpha))\big)\big), \tag{20}$$

where B, C, D, and E are the stiffness, shape, peak, and curvature parameters of the tire model.

Thus,

$$F_{yf} = F_y(\alpha_f), \qquad F_{yr} = F_y(\alpha_r). \tag{21}$$

Polytopic state-space representation of the lateral vehicle model

The presented vehicle model can be transformed into a parameter-dependent state-space representation

$$x = \begin{bmatrix} v_y \\ r \end{bmatrix}, \qquad u = \delta, \tag{22}$$

The majority of the nonlinear dynamics of the vehicle model comes from the tire model. The Pacejka's tire model can be simplified by using a non-constant cornering stiffness

$$F_{yf} = -C_f(t)\alpha_f, \qquad F_{yr} = -C_r(t)\alpha_r. \tag{23}$$

Furthermore, using the small angles approximation:

$$\begin{aligned} \alpha_f &\approx \frac{v_y + l_f r}{v_x} - \delta, \\ \alpha_r &\approx \frac{v_y - l_r r}{v_x}. \end{aligned} \tag{24}$$

The polytopic state-space representation:

$$\dot{x} = A(\rho)x + B(\rho)u, \tag{25}$$

$$A(\rho)\begin{bmatrix} -\dfrac{C_f(t)+C_r(t)}{m v_x(t)} & -v_x(t) - \dfrac{l_f C_f(t) - l_r C_r(t)}{m v_x(t)} \\ -\dfrac{l_f C_f(t) - l_r C_r(t)}{I_z v_x(t)} & -\dfrac{l_f^2 C_f(t) + l_r^2 C_r(t)}{I_z v_x(t)} \end{bmatrix} \tag{26}$$

$$B(\rho) = \begin{bmatrix} \dfrac{C_f(t)}{m} \\ \dfrac{l_f C_f(t)}{I_z} \end{bmatrix} \tag{27}$$

with the scheduling vector:

$$\rho = [C_f, C_r, v_x] \tag{28}$$

From the three scheduling parameters, the longitudinal velocity is measurable; thus, the RL agent is only used to estimate the cornering stiffness. The measured state for the observer is the yaw-rate:

$$C = [0 \quad 1] \tag{29}$$

The model is gridded by using the following parameter ranges: $v_x \in [10, 30](m/s)$, $C_f \in [140000, 200000](N/rad)$, $C_r \in [140000, 200000](N/rad)$. The observer is designed based on the presented model using Equation (11).

LPV control design

During the control design, the main goal is to guarantee the trajectory tracking of the vehicle, which can be formalized as:

$$z_1 = |y_{ref} - y| \rightarrow min!$$
$$z_2 = |\delta| \rightarrow min!$$

These requirements can be fulfilled by choosing appropriate weighing functions. As shown in Figure 2, W_{ref} is used to weight the reference signal, $W_{z,1}$ and $W_{z,2}$ are to guarantee the predefined performances, while $W_{\omega,1}$,$W_{\omega,2}$ and $W_{\omega,1}$ are used to deal with the sensor noises. The

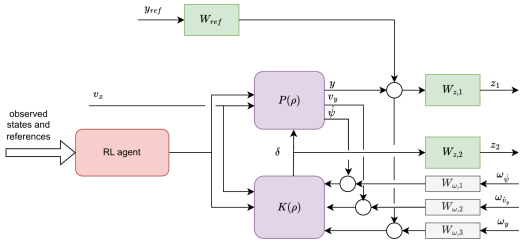


Fig. 2. LPV control design

controller gain $K(\rho)$ can be computed by solving the presented Equation (8).

RL training

The main goal of the RL agent is to estimate the cornering stiffness for both the observer and the controller. The observations of the network are in Table 1.

The observations for the RL agent are listed in Table 1. It includes the measured and estimated states with previous values, control signal, reference signal, and the previously estimated cornering stiffness as well.

Table 1. Components of the observation vector

Variable	**Symbol**
Estimated lateral velocity	$\hat{v}_y(t), \hat{v}_y(t-1)$
Measured yaw-rate	$r, r(t-1)$
Steering angle	$\delta(t), \delta(t-1)$
Lateral acceleration	$a_y(t)$
Est. C. stiffness (previous)	$\hat{C}_f(t-1), \hat{C}_r(t-1)$
Reference signal	$y_{ref}(t), y_{ref}(t+k), y_{ref}(t+2k)$

Finally, the reward function is defined as:

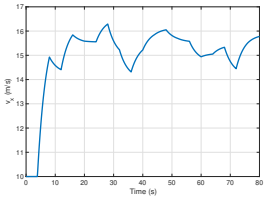
$$\mathcal{R} = (\dot{r} - \hat{\dot{r}})^2 w_1 + (v_y - \hat{v}_y)^2 w_2 + (y_{ref} - y)^2 w_3 \tag{30}$$

Note that the measured v_y is only used for training; it is not required afterwards.

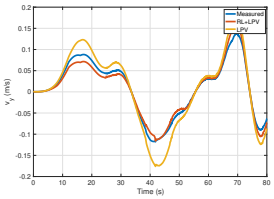
Simulation results

Finally, a simulation example is presented to show the operation and the effectiveness of the proposed combined control technique. In the test scenario, the nonlinear lateral model is used together with Pacejka's tire model. The vehicle is driven along a section of the F1 Hungaroring track, which contains several sharp bends. The longitudinal velocity profile of the vehicle is depicted in Figure 3(a). The test scenario has been conducted twice. First, using the baseline LPV controller with no information about the cornering stiffness. In the second run, the proposed LPV-RL algorithm is applied with the corresponding stiffness estimation.

The outcome of the observer can be seen in Figure 3(b) and 4(a). In both cases (yaw-rate and lateral velocity), the combined approach provides better results. However, it is more significant in the case of lateral velocity, where the peak values show more than 30% improvement. Finally, the real and the RL-estimated tire characteristics are shown in Figure 4(b). As can be seen, there is no significant difference between the curves; the maximal deviation is below 6%.

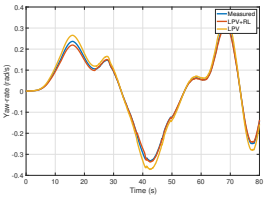


(a) Longitudinal velocity

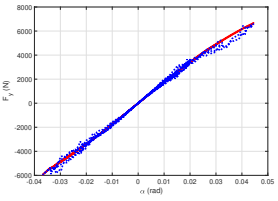


(b) Lateral velocity

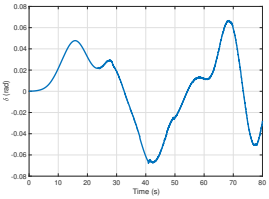
Fig. 3. Vehicle motion states



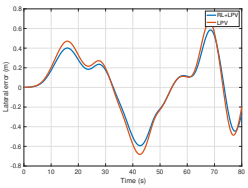
(a) Yaw-rate



(b) Estimated tire characteristics



(c) Control input



(d) Error

Fig. 4. Vehicle dynamics and control signals

4.2 Simplified railway-oriented example

As a second example, the proposed LPV-RL framework is demonstrated through a simplified railway longitudinal control problem. The objective is to maintain accurate

velocity tracking under varying wheel-rail adhesion conditions.

The longitudinal dynamics of the railway vehicle are described by

$$m\dot{v} = F_x - F_{res} \tag{31}$$
$$J\dot{\omega} = T - RF_x, \tag{32}$$

where v denotes the longitudinal velocity, ω is the wheel angular velocity, m is the mass of the railway vehicle, J is the rotational inertia of the wheelset, T is the applied traction/braking torque, R denotes the wheel radius, F_x represents the wheel-rail contact force, and F_{res} is the summed resistance force.

The longitudinal slip ratio is computed as

$$s = \frac{R\omega - v}{\max\left(|v|, |R\omega|, \epsilon\right)}. \tag{33}$$

The wheel-rail interaction is modeled using a simplified Pacejka-type nonlinear adhesion model, see Eq. (20)

A nonlinear observer is applied to estimate the slip ratio using the measured wheel angular velocity and longitudinal acceleration signals. The RL agent utilizes the estimated states to adapt the scheduling parameter online.

The observation vector of the RL agent is defined as $o = [\hat{v}(t), \hat{s}(t), \omega(t), a_x(t), v_{ref}(t), \hat{\mu}(t-1), \hat{v}(t-1), \hat{s}(t-1), \omega(t-1), a_x(t-1), v_{ref}(t-1), \hat{\mu}(t-2)]$

while the reward function is formulated as

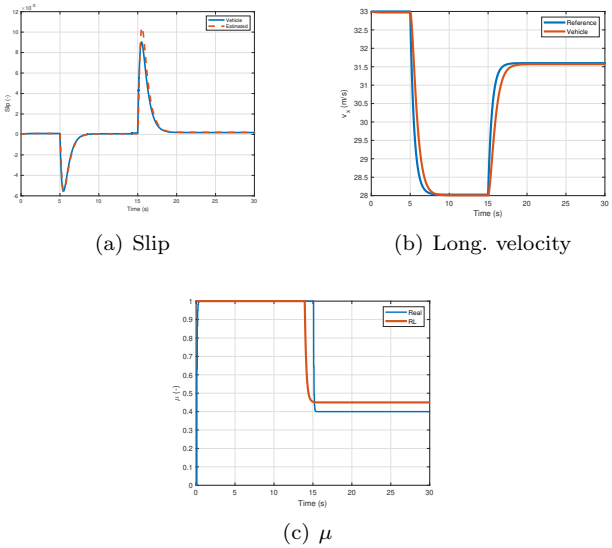
$$R = -\left(w_1(v - v_{ref})^2 + w_2(s - \hat{s})^2\right). \tag{34}$$

The RL agent is trained using the DDPG algorithm under varying adhesion conditions.

Figure 5(b) presents the longitudinal velocity tracking performance of the proposed method, while Figure 5(a) illustrates the estimated slip ratio. Furthermore, the evolution of the estimated adhesion parameter is shown in Figure 5(c). The results demonstrate that the RL-enhanced LPV controller can successfully adapt to changing wheel-rail contact conditions while maintaining stable operation.

5. CONCLUSION

In the paper, a combined control method has been proposed. The main idea was to create an RL agent that can learn the scheduling parameter of a polytopic system by minimizing both the observer error and the tracking error of the controller. The proposed algorithm has been applied to two vehicle-oriented control problems. The examples have shown that the RL agent can learn the scheduling parameters and, in this way, improve the overall performance of the closed-loop system.



(a) Slip (b) Long. velocity

(c) μ

Fig. 5. Train example

REFERENCES

Daniel Fenyes, Tamas Hegedus, and Peter Gaspar. Control informed neural network for controller selection. In *Proceedings of the 11th International Conference on Control, Decision and Information Technologies (CoDIT 2025)*, Split, Croatia, July 2025a.

Daniel Fenyes, Tamas Hegedus, Balazs Nemeth, and Peter Gaspar. Neural network based controller combination for automated vehicles. In *Proceedings of the 6th International Conference on Control and Fault Tolerant Systems (SysTol 2025)*, Ayia Napa, Cyprus, 2025b.

Maxim Lapan. *Deep Reinforcement Learning Hands On: Apply modern RL methods, with deep Q networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing, 2018.

Javad Mohammadpour and Carsten W. Scherer. An overview of lpv systems. In *Control of Linear Parameter Varying Systems with Applications*. Springer Boston, 2012.

H. B. Pacejka. *Tyre and vehicle dynamics*. Elsevier Butterworth-Heinemann, Oxford, 2004.

Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019. doi: 10.1146/annurev-control-053018-023825.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2015.

Chen Tang, Ben Abbatematteo1, Jiaheng Hu, Rohan Chandra, Roberto Martin-Martin, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. In *Annual Review of Robotics and Autonomous Systems*, 2024. Survey.

Mike Timmerman, Aryan Patel, and Tim Reinhart. Adaptive gain scheduling using reinforcement learning for quadcopter control. *arXiv preprint*, 2024. URL `https://arxiv.org/abs/2403.07216`.

Roland Toth. *Modeling and Identification of Linear Parameter-Varying Systems*, volume 403 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, Heidelberg, 2010.

Yi Liang Yeh and Po Kai Yang. Design and comparison of reinforcement learning based time varying pid controllers with gain scheduled actions. *Machines*, 9(12): 319, 2021.