Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-20 (2022) 97–102

Constrained Training for Guided Feedback Controller

Simon Gottschalk * **Matthias Gerdts** *

* *Universität der Bundeswehr München, 85579 Neubiberg (e-mail: simon.gottschalk@unibw.de, matthias.gerdts@unibw.de)*

Abstract: In this paper, we are focusing on a neural network feedback controller tailored to a lane following application. Besides a classical goal like avoiding a collision with obstacles, we aim at investigating an approach in order to guarantee not to leave the street at any point, since this is of major importance for the safety of the passengers. Unfortunately, guarantees like this are quite rare since the training of neural networks is often based on data, which only allows statistical conclusions. We derive a deterministic inequality equation guaranteeing this safety. Furthermore, we introduce a penalty strategy in order to integrate this inequality into a Reinforcement Learning algorithm.

Keywords: Neural Networks, Reinforcement Learning, Reinforce, Stability, Penalty method

1. INTRODUCTION

Nowadays, the range of applications, where neural networks (NNs) (Haykin, 1999) are used, expanses on and on. One of the reasons is their versatility since they can be used in the context of supervised, unsupervised and Reinforcement Learning (RL) (Bertsekas, 2019; Sutton and Barto, 2018). Statistical learning in order to train these NNs benefits from its ability to recognize structures, learn complex controls or mimic behaviors without needing a deeper insight into mathematical modeling. Unfortunately, this goes often along with a challenging quality assurance for trained results. In this contribution, we aim at focusing on such a quality assurance in the case of a trained feedback controller for a lane following task. We are interested in the assurance to stay always on the street. This should be the key property of the feedback controller, which we will found by applying a RL algorithm.

The assurance not to leave the street, can also be described by demanding trajectories which do not leave a tubular neighborhood around a reference trajectory. Thus, the described property belongs in a sense to the concept of stability. For plain differential equations (Grüne and Junge, 2016, pp.103 ff.) and for feedback controls especially in the context of linear systems (Kisačanin and Agarwal, 2001, pp. 178 ff), stability properties are well understood and known for years. Taking into account the NN, basic stability properties are, for example, considered in (Haber and Ruthotto, 2017). Furthermore, NNs embedded into a closed loop feedback control system are discussed in (Khlaeo-om et al., 2011). Therein, the authors derive conditions for stability at least for very basic neural networks (and control systems). In (Suykens et al., 1999), again a simple network architecture in a feedback control loop is treated. This time, the nonlinear system and the controller are parameterized by a multilayer perceptron with one hidden layer and based on this restriction stability is discussed. Even those publications restrict themselves to basic architectures, in the end the authors receive conditions, which are often based on the Lyapunov stability and can be taken into account in the training of the NN, e.g. via a modified backpropagation algorithm (Ekachaiworasin and Kuntanapreeda, 2000).

Beyond that, in the last years, publications regarding Safe Reinforcement Learning gain in interest. For instance, in (Berkenkamp et al., 2017) the authors train policies, which provide the largest level set of a suitable Lyapunov function, wherein the Lyapunov function is forward invariant. Since all these approaches aim at asymptotic stability, a safe state needs to be predefined. Such a safe state can hardly be assumed in our context, since a safe trajectory on the street can not be specified due to obstacles. In (Zanon and Gros, 2021) RL and Model Predictive Control (MPC) are combined in such a way that a safe controller is the result. In this framework, RL trains a parametrized Q function approximator of the underlying problem, which is only given as a Markov Process. Additional safety constraints are integrated in this approximator and the predictive solution provides the control for the original dynamical system. This approach falls back on the properties of MPC and thus highly depends on the capabilities of the approximator and the solvability of the resulting optimization problem with MPC.

In this contribution we train a NN feedback controller by a pure model-free RL methods in order to be independent on a potential model approximation. Thereby, we introduce a guidance for the controller, which is flexible enough to avoid collisions with obstacles, but ensures that the car does not leave the street. To this end, we furthermore aim at introducing simple deterministic constraints and integrating them into the RL approach.

2. GUIDED STABILITY CONTROLLER

In order to specify our goals, we start with an introduction into our underlying framework. We consider a dynamical system $\dot{x}(t) = f(x(t), u(t))$ with control $u \in \mathbb{R}^{n_u}, n_u \in \mathbb{N}$

10.1016/j.ifacol.2022.09.078

and state $x \in \mathbb{R}^{n_x}, n_x \in \mathbb{N}$. For this dynamical system, a reference control $u_{\text{ref}}(t)$ is given, which produces a reference trajectory $x_{\text{ref}}(t)$. We assume that they are sufficiently smooth. This trajectory and control exist for comparison reasons and describe the middle of the lane. We further assume the normalized orthogonal vector $n_{\text{ref}}(t)$ to be perpendicular to $x_{\text{ref}}(t)$. Although we know a reference trajectory, we are actually interested in a feedback control $u(x(t))$, which steers the dynamical system close to the middle lane, but is also able to avoid obstacles. In other words, our control is allowed to cause deviations of the corresponding trajectory from the middle of the lane with the restriction not to leave the street. We write $u_{\text{NN}}(x(t))$ for the control generated from the NN based on the current state $x(t)$. Overall, we have:

$$\dot{x}_{\text{ref}}(t) = f\Big(x_{\text{ref}}(t), u_{\text{ref}}(t)\Big), \quad x_{\text{ref}}(0) = x_0,$$
$$\dot{\tilde{x}}(t) = f\Big(\tilde{x}(t), u_{\text{NN}}(\tilde{x}(t))\Big), \quad \tilde{x}(0) = \tilde{x}_0.$$

For the sake of simplicity, we assume, that the controls are bounded between minus one and one. At first glance, this looks like a restriction but since scaling is always possible, we allow for arbitrary bounded controls. Figure 1 visualizes the idea of the considered scenario. The course of the lane can be seen as a trajectory, which is given by the reference trajectory (gray dotted line). Obviously, following the line would be easy, since we can just apply the reference control. But we are now interested in finding a controller, which is able to circumvent obstacles (red circle), which might appear on the street. Nevertheless, it is of crucial importance to stay on the street.

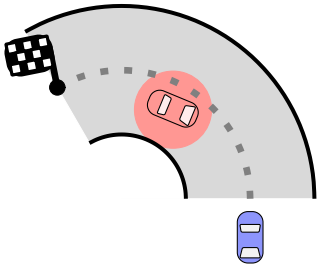
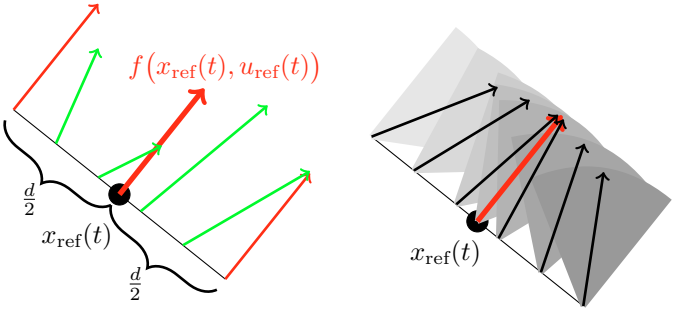


Fig. 1. Test track.

Thus, our goal is to find a controller, based on a NN, which is able to do an avoidance manoeuvre and for which we can guarantee that it is still close enough to the reference trajectory representing the lane center. Therefore, we will deduce constraints, which we include into the training of the NN.

2.1 Constraint for the Guided Controller

Typically, if one is interested in the discrepancy between two realizations of a dynamical system with different controls, one makes use of Gronwall's Lemma (see (Hale, 1977)). Unfortunately, this lemma leads to an estimation of an upper bound of this discrepancy, which grows exponentially with the time. Thus, this is not the right tool for guaranteeing that the trajectory stays close to the reference trajectory. Instead, we derive a guarantee by introducing a suitable constraint based on the idea of an



(a) Unconstrained directions. (b) Constraint visualization.

Fig. 2. Motivation for the guiding directions.

inward pointing condition (Frankowska, 2009). We stress at this point that the lane following example naturally arises in a two dimensional setting. This is not a restriction for the following discussion. Applications of higher dimensionality, i.e. a plane following an flight corridor in three dimensions, can be considered with small adjustments. We will motivate the derivation by considering Figure 2a and Figure 2b. In Figure 2a, we see a state $x_{\text{ref}}(t)$ on the reference trajectory. The red arrow, which is pointing out of the state is its derivative (the right hand side function f). States of other realizations of the dynamical system with the NN controller and a disturbance in the initial value may lay on the black line perpendicular to the derivative. Their directions of movement are exemplarily visualized by green arrows. By drawing red arrows parallel to the reference derivative at the borders of the track, it becomes clear that the green arrows are not allowed to point out of the corridor included by the red arrows. Thus, it is important to find a simple inequality or equality constraint ensuring this property. Figure 2b visualizes the idea for this constraint. For each point x of the black line with length d (roadway width), we define a direction, $\hat{f}$, which guides the actual derivative. Of course it is too restrictive to predefine the exact direction and this is, why we allow a deviation from this direction (gray scale circle segments) up to a certain angle $\frac{\beta}{2}$ $(0 < \beta \leq \pi)$. Thereby, we need to ensure, that no deviation of the guiding direction leaves the corridor of the reference solution. Thus, the guiding arrows close to the boundary have to point in the interior of the corridor. A smooth definition of these guiding arrows at position $x(s,t) := x_{\text{ref}}(t) + s * n_{\text{ref}}(t)$, $-\frac{d}{2} \leq s \leq \frac{d}{2}$ can be defined using a rotation matrix $A(s)$:

$$\hat{f}(s,t) := \underbrace{\begin{bmatrix} \cos\left(\beta\frac{s}{d}\right) & -\sin\left(\beta\frac{s}{d}\right) \\ \sin\left(\beta\frac{s}{d}\right) & \cos\left(\beta\frac{s}{d}\right) \end{bmatrix}}_{=:A(s)} f(x_{\text{ref}}(t), u_{\text{ref}}(t))$$

with the normalized orthogonal vector:

$$n_{\text{ref}}(t) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \frac{f(x_{\text{ref}}(t), u_{\text{ref}}(t))}{\|f(x_{\text{ref}}(t), u_{\text{ref}}(t))\|}.$$

Now, we can deduce a condition, which ensures that the derivative $f(x(s,t), u_{\text{NN}}(x(s,t)))$ is close to the guiding directions $\hat{f}(s,t)$, using $\varepsilon := \cos\frac{\beta}{2}$:

$$\varepsilon \leq \frac{f\big(x(s,t), u_{\text{NN}}(x(s,t))\big)^T \hat{f}(s,t)}{\|f\big(x(s,t), u_{\text{NN}}(x(s,t))\big)\| \|\hat{f}(s,t)\|} =: \eta(s,t), \qquad (1)$$

Note, that this condition has to be fulfilled for all s relating to the street width d and t going along the

reference trajectory. We stress at this point that for some dynamical system no controller can be defined such that this constraint is fulfilled. In order to guarantee that it is solvable, we need to restrict ourselves to systems, which are able to directly control the direction of change arbitrarily by the current control. Despite the very simple formulation of the inequality constraint (1), it is not clear, how we can best fulfill this condition. Since finding the global minimum of $\eta(s,t)$ analytically is futile, we make use of a numerical approach with a suitable error estimation. The idea is to compute the value of η at discretization points of a specified grid. A straightforward way to find such a grid would be to discretize the time interval $[0,T]$ with final time T and the interval $\left[-\frac{d}{2},\frac{d}{2}\right]$ and to evaluate η on these grid points. Thereon, we can directly check the inequality constraint. Nevertheless, we aim at fulfilling this condition on points in between these grid points as well. Thus, we consider $s \in [s_i, s_{i+1}]$ and $t \in [t_j, t_{j+1}]$. Then, the question is: How much can $\eta(s,t)$ deviate from the value at the grid point $\eta(s_i,t_j)$ at most? In order to answer this question, we use Taylor's theorem applied to $\eta(s,t)$ to obtain:

$$\eta(s,t) = \eta(s_i,t_j) + \eta_s(s_\xi,t_\xi)(s-s_i) + \eta_t(s_\xi,t_\xi)(t-t_j), \tag{2}$$

for $s_\xi \coloneqq (1-\xi)s_i + \xi s$, $t_\xi \coloneqq (1-\xi)t_j + \xi t$ and a suitable $\xi \in [0,1]$. Here, η_s and η_t denote the derivative with respect to s, respectively t. The estimation of the remainder is based on the Lagrange formula (see (Abramowitz and Stegun, 1964, p.14)). If we are able to show that the derivatives of η are bounded, which is our next step, we know that the deviation between $\eta(s,t)$ and $\eta(s_i,t_j)$ is bounded by equation (2). In order to estimate such a bound, we introduce the following definitions for a better overview:

$$F_1(s,t) \coloneqq f\big(x(s,t), u_{\mathrm{NN}}(x(s,t))\big),$$
$$F_2(t) \coloneqq f(x_{\mathrm{ref}}(t), u_{\mathrm{ref}}(t)).$$

Then, the derivatives can be stated as:

$$\frac{\partial \eta(s,t)}{\partial s} = \frac{\left(\frac{\partial}{\partial s}F_1(s,t)^T A + \frac{\beta}{d}F_1(s,t)^T A'\right)F_2(t)}{\|F_1(s,t)\|\|F_2(t)\|} - \frac{F_1(s,t)^T A F_2(t) F_1(s,t)^T \frac{\partial}{\partial s}F_1(s,t)}{\|F_1(s,t)\|^3\|F_2(t)\|},$$

$$\frac{\partial \eta(s,t)}{\partial t} = \frac{\left(\frac{\partial}{\partial t}F_1(s,t)^T A F_2(t) + F_1(s,t)^T A \frac{d}{dt}F_2(t)\right)}{\|F_1(s,t)\|\|F_2(t)\|} - \frac{F_1(s,t)^T A F_2(t) F_1(s,t)^T \frac{\partial}{\partial t}F_1(s,t)}{\|F_1(s,t)\|^3\|F_2(t)\|} - \frac{F_1(s,t)^T A F_2(t) F_2(t)^T \frac{d}{dt}F_2(t)}{\|F_1(s,t)\|\|F_2(t)\|^3}.$$

Here, the notation A' represents the derivative of the rotation matrix with respect to s. Finally, based on these computations, we can deduce an estimation of the norm of the derivatives η_s and η_t:

$$\left\|\frac{\partial \eta(s,t)}{\partial s}\right\| \le 2\frac{\|\frac{\partial}{\partial s}F_1(s,t)\|}{\|F_1(s,t)\|} + \frac{\beta}{d},$$

$$\left\|\frac{\partial \eta(s,t)}{\partial t}\right\| \le 2\frac{\|\frac{\partial}{\partial t}F_1(s,t)\|}{\|F_1(s,t)\|} + 2\frac{\|\frac{d}{dt}F_2(t)\|}{\|F_2(t)\|}.$$

This estimation results from the triangle inequality of norms, the inequality of Cauchy-Schwarz (Grinshpan, 2005, p.72) and the fact, that the euclidean norm of the rotation matrix is one. Now, it is clear that we can estimate the derivatives of η, if we can assume that the norms of F_1 and F_2 are bounded from below with values greater than zero and that the norms of the derivatives of them are bounded from above. In the numeric section, we will show how this can look like for a lane following example. First, we will discuss how the constraint can be integrated into a learning algorithm.

2.2 Integration into Reinforcement Learning

Up to now we have a rather abstract condition for guaranteeing to stay in the neighborhood of the reference trajectory. In the next step, we ask ourselves, how we can benefit from it. Therefore, we have a look at how a controller can be found for a dynamical system like the one describing a car driving on the street. One way to do this is RL, which we want to consider in this paper.

In general, RL is based on the idea, that the dynamical system is simulated forward in time with a given, incomplete controller and then the controller is updated based on these generated data afterwards. The underlying framework bases on a Markov Decision Process (Feinberg and Shwartz, 2002), which consists of the state space S, the action space A, a transition probability $p(o'|o,a)$ for $o, o' \in S$ and $a \in A$, an initial distribution p_0 as well as a reward/cost function $r : S \times A \to \mathbb{R}$. Note that the state space in our context is given as $S \coloneqq \mathbb{R}^{n_x}$. Since the transition probability represents the actual dynamical system, we only assume to have a probability for the next state under the condition that the previous state and the action are given. A further assumption is, that a reward/cost function rates the current situation of the system. This function is needed in order to compare the performance of different controllers. Finally, we assume that a distribution for the initial state is given. It remains to formalize the controller fitting into this setting. The controller is given by a policy $\pi_\theta(a|o)$, which shows the probability that the controller outputs the action a under the condition that the current state o is given. In the following, we consider parameterized controller π_θ with parameters θ such as a NN, where the weights and biases are trained. Overall, the controller consists of a NN, which maps the current state to parameters of a distribution (e.g. mean value of Gaussian distribution), from which the next action is sampled.

Based on these assumptions, the optimization problem is established. It is clear that we are interested in finding the best possible reward or, in our case, the minimal costs, which can be stated as:

$$\min_\theta \quad \mathbb{E}[R], \quad \text{with } R = \sum_{k=0}^{N-1} \gamma^k r(o_k, a_k). \tag{3}$$

Here, the costs of one trajectory are given by R, which sums up all costs of the trajectory and weights them by a discount factor γ. The objective function is written down by an expected value. With $\tilde{p}(o_0, a_0, \ldots, o_N) = p_0(o_0)\prod_{k=0}^{N-1} p(o_{k+1}|o_k,a_k)\,\pi_\theta(a_k|o_k)$, it is given by:

$$\mathbb{E}[R] \coloneqq \int_S \int_A \cdots \int_A \int_S \tilde{p}\left(o_0, a_0, o_1, \ldots, a_{N-1}, o_N\right) \cdot R \quad da_{N-1} \ldots da_0 do_0.$$

This optimization problem (3) can be solved in several ways. One example is Reinforce (Williams, 1992), which estimates a gradient direction of the objective function based on simulated data (the p-th trajectory consisting of o_0^p,a_0^p,o_1^p,a_1^p,..., a_{N-1}^p):

$$\nabla_\theta \mathbb{E}[R] \approx \sum_{p,l} \nabla_\theta \ln\left(\pi_\theta\left(a_l^p | o_l^p\right)\right) \left(\sum_{k=0}^{N-1} \gamma^k r(o_k^p, a_k^p)\right). \quad (4)$$

Overall, the general procedure is to initialize a random controller. Afterwards, this controller is used to generate several trajectories. These trajectories contain all data we need to improve the parameters of the controller in the direction of the estimated derivative.

Now, we aim at integrating the constraint (1) into this procedure. One can think of a projected gradient method (e.g. (Bertsekas, 1976)), where the new parameters are projected into the feasible set. But, since the amount of parameters of a controller like a NN is very high, this is very expensive. Thus, we do not further follow this idea. Instead, the gradient need to be adjusted such that our inequality condition is improved as well. However, it does not make sense to check the inequality only on the simulated trajectories, since we cannot deduce an overall guarantee from it. Rather, we additionally consider a fine grid on the whole street. It is not enough to modify the cost function. We circumvent this problem by introducing a penalty term in the cost function. This new part should vanish, if the inequality is fulfilled everywhere, but increase rapidly if it is not. For deeper insights into penalty approaches and their convergence, we refer to (Nocedal and Wright, 2006, p.487 ff.). This approach leads us to an artificial, second cost function $r_{\text{art}}(s,t) \coloneqq 100\,(\varepsilon - \eta(s,t))\mathbb{1}_{[\eta(s,t)<\varepsilon]}$. Note, that these costs are deterministic. Unfortunately, in this form the cost function does not fit into our RL framework. Thus, we need to adjust our notation a little bit. Remember, that s and t represent a specific position in the two dimensional space. In order to compute $\eta(s,t)$, we apply the NN to this position and follow the computation rule (1). In order to fit the new costs in the RL framework, we need data with distributed controls for every position. Only in this way, we can compare them by the cost function and update in the direction of improvement. Therefore, we introduce:

$$\tilde{\eta}(s,t,u) \coloneqq \frac{\tilde{F}_1(s,t,u)^T A F_2(t)}{\|\tilde{F}_1(s,t,u)\| \|F_2(t)\|},$$
$$\tilde{F}_1(s,t,u) \coloneqq f\big(x_{\text{ref}}(t) + s \cdot n_{\text{ref}}(t), u\big),$$
$$r_{\text{art}}(s,t,u) \coloneqq 100\,(\varepsilon - \tilde{\eta}(s,t,u))\mathbb{1}_{[\eta(s,t)<\varepsilon]}.$$

Then, we get the penalty part of the objective function based on the grid points $(s_i,t_j)_{i=0,\ldots,N,j=0,\ldots,M}$:

$$\sum_{j=0,i=0}^{M,N} \mathbb{E}_u\left[r_{\text{art}}(s_i,t_j,u)\right].$$

In this context, $\mathbb{E}_u\left[r_{\text{art}}(s_i,t_j,u)\right]$ denotes the expected value with respect to the stochastic variable u, which is normally distributed ($u \sim \mathcal{N}(u_{\text{NN}}(x_{\text{ref}}(t) + s n_{\text{ref}}(t)), \sigma)$) with the mean value generated by the NN and a fixed variance σ. In order to integrate it into our RL algorithm based on a gradient method, we consider the derivative:

$$\nabla_\theta \mathbb{E}_u\left[r_{\text{art}}(s_i,t_j,u)\right] = \mathbb{E}_u\left[\nabla_\theta \bar{u}(s_i,t_j,u)\, r_{\text{art}}(s_i,t_j,u)\right],$$
$$\text{with } \bar{u}(s_i,t_j,u) = \frac{\left(u - u_{\text{NN}}(x_{\text{ref}}(t_j) - s_i n_{\text{ref}}(t_j))\right)^2}{-2\sigma^2}. \quad (5)$$

In order to rewrite the derivative of the expected value again into an expected value (see Appendix A), we used the same steps as they are used in the derivation of the derivative in (4) for the Reinforce algorithm. Together, the derivatives (4) and the sum of the sampled estimation of (5) over all grid points form the new update direction, which we will test on an application case in the next section.

3. NUMERICAL RESULTS

As a suitable representative of a first order dynamical system, we consider the following right hand side for control $u = [v,\phi]^T$ and state $x = [x_1,x_2]^T$:

$$f(x,u) \coloneqq \begin{bmatrix}(v+1.5)\cos(\pi\phi+\pi)\\ (v+1.5)\sin(\pi\phi+\pi)\end{bmatrix}. \quad (6)$$

As reference trajectory, we decide to take a semicircle:

$$x_{\text{ref}}(t) \coloneqq \begin{bmatrix}\cos(t)\\ \sin(t)\end{bmatrix}, n_{\text{ref}}(t) = \begin{bmatrix}-\sin(t)\\ \cos(t)\end{bmatrix}, t \in \left[0, \frac{3\pi}{2}\right]. \quad (7)$$

In Figure 1, the idea of our task is sketched. We aim at steering the blue car to the finishing flag. Thereby, obstacles on the street like a parking car (red) should be avoided. Furthermore, the road width is set to $d = 1$. Note, that in our application case the norm of the derivative of the normalized orthogonal vector is $\|\dot{n}_{\text{ref}}\| = 1$. Furthermore, we can deduce, that the control in order to generate the reference trajectory is given as $u_{\text{ref}}(t) = \left[-0.5, \frac{t}{\pi} - 0.5\right]^T$ with $\dot{u}_{\text{ref}}(t) = \left[0, \frac{1}{\pi}\right]^T$ and $\|\dot{u}_{\text{ref}}(t)\| \leq b_{\text{ref}} \coloneqq \frac{1}{\pi}$, for all points in time t. As a controller, we choose a NN controller with two hidden layers and an output layer. Its parameters θ consists of weights W_q and biases b_q, $q = 1,2,3$. The activation function is the hyperbolic tangent for each layer:

$$u_{\text{NN}}(x) \coloneqq \tanh(y_3), \text{ with } \quad y_3 = \tanh(y_2) W_3 + b_3,$$
$$y_2 = \tanh(y_1) W_2 + b_2, \quad y_1 = x^T W_1 + b_1.$$

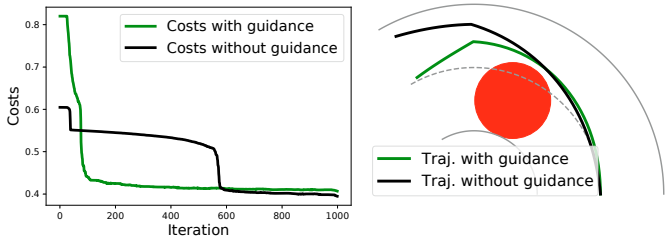
Then, the derivative with respect to the input is given as:

$$(u_{\text{NN}})_x(x) \coloneqq D_3 W_3^T D_2 W_2^T D_1 W_1^T \quad (8)$$
$$\text{with } D_q \coloneqq \text{diag}\left(\tanh'(y_q)\right).$$

Since $\tanh' = 1 - \tanh^2$ is bounded, we obtain a bounded derivative of the NN controller by assuming, that the weights are bounded as well. We denote this bound by $b_{\text{NN}} = \||W_3|^T \cdot |W_2|^T \cdot |W_1|^T\|$. Having all these information, we can estimate the norm of F_1 and F_2:

$$0.5 \leq \|F_1(s,t)\| = (v+1.5) \leq 2.5 \text{ and}$$
$$0.5 \leq \|F_2(t)\| = (v_{\text{ref}} + 1.5) \leq 2.5 \text{ with}$$
$$[v,\phi]^T = u_{\text{NN}}(x(s,t)) \text{ and } [v_{\text{ref}}, \phi_{\text{ref}}]^T = u_{\text{ref}}(t).$$

For an estimation of the norms of the derivatives of F_1 and F_2, we need to discuss a few estimations before. First, the norm of the normalized orthogonal vector n_{ref} is by definition one. Furthermore, the norm of the Jacobi matrix of f with respect to u (f_u) is bounded by 2.5π as discussed in the appendix (B.4). Based on the inequalities in the



(a) Comparison of the costs during training. (b) Trained trajectory for a left turn with obstacle (red).

Fig. 3. Training results.

appendix (B.1)-(B.3), inequality (B.4) and the estimations discussed so far, we end up with:

$$\left\| \frac{\partial \left(F_1(s,t)\right)}{\partial s} \right\| \leq 2.5\pi \cdot b_{\mathrm{NN}}, \left\| \frac{d\left(F_2(t)\right)}{dt} \right\| \leq 2.5\pi \cdot b_{\mathrm{ref}}, \quad (9)$$

$$\left\| \frac{\partial \left(F_1(s,t)\right)}{\partial t} \right\| \leq 2.5\pi \cdot b_{\mathrm{NN}} \left(2.5 + \frac{d}{2} \left\| \dot{n}_{\mathrm{ref}}(t) \right\| \right). \quad (10)$$

This leads to the final estimate of the derivatives of η:

$$\left\| \frac{\partial \eta(s,t)}{\partial s} \right\| \leq 2 \cdot \frac{2.5\pi \cdot b_{\mathrm{NN}}}{0.5} + \frac{\beta}{d},$$

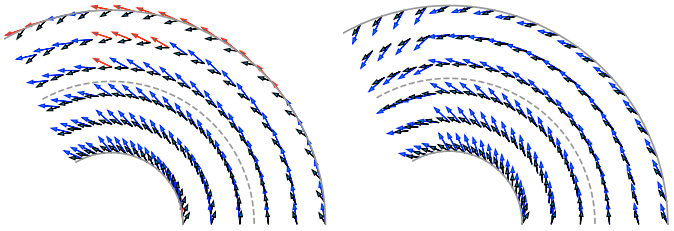
$$\left\| \frac{\partial \eta(s,t)}{\partial t} \right\| \leq 2 \left(\frac{2.5\pi \left(b_{\mathrm{NN}} \left(2.5 + \frac{d}{2} \left\| \dot{n}_{\mathrm{ref}}(t) \right\| \right) + b_{\mathrm{ref}} \right)}{0.5} \right).$$

It remains to estimate the bound b_{NN} based on (8). We set $b_{\mathrm{NN}} := 11$ and ensure its correctness during the training by monitoring the product of the absolute weight matrices. Overall, we have the estimations:

$$\left\| \frac{\partial \eta(s,t)}{\partial s} \right\| \leq 347, \quad \left\| \frac{\partial \eta(s,t)}{\partial t} \right\| \leq 1047.$$

Thus, we demand to fulfill the inequality constraint on the grid points with the angle $\hat{\beta} = \frac{\pi}{4}$ in order to guarantee the conditions for all points with angle $\beta = \frac{\pi}{3}$. Since, $\Delta\varepsilon = \cos\left(\frac{\pi}{4 \cdot 2}\right) - \cos\left(\frac{\pi}{3 \cdot 2}\right) \geq 0.05$, we need to ensure that the discretization size for the interval $[-0.5, 0.5]$ fulfills $h_s \leq \frac{0.05}{347}$ and $h_t \leq \frac{0.05}{1047}$ for the interval $[0, \frac{2\pi}{3}]$.

Then, we applied the RL approach, which we discussed in the previous sections. In order to receive an impression of the influence of the penalty term, we also apply the Reinforce algorithm without further adjustments. More details about the applied approach can be found in the Appendix B. In Figure 3a, the total costs during the training phase are depicted. It can be observed, that both strategies decrease the total costs during the training. The higher initial cost of the penalty approach obviously results from the additional penalty term in the objective function. Nevertheless, the learning curve is much steeper than the curve of the pure Reinforce and is reaching its stage much earlier. By contrast, the black curve has a plateau during its training, which is the part of the training, where the blue car has to figure out, how to avoid the red obstacle. Nevertheless, both algorithms lead to feasible trajectories as it can be observed in Figure 3b. There, the trajectory of each trained controller is depicted. They avoid a collision with the red obstacle and follow the street. The most interesting observation can be seen in Figure 4. Here,



(a) Constraints for training without guidance. (b) Constraints for training with guidance.

Fig. 4. Visualization of the corresponding stability constraint at grid points (blue=fulfilled, red=failed).

we draw the guiding black arrows and the final actual directions of the right hand side and a phase field of the ODE incorporating both controllers. In the case of a fulfilled inequality, the arrow appears in blue. Otherwise, it is red. We directly see, that in the pure Reinforce case red arrows appear. This means that the inequality (1) is not fulfilled and that there is the risk to leave the street. In the case of the penalty approach, everything is blue and also a test on a finer grid, which is due to a better overview not depicted here, was successful. Thus, we can ensure that the blue car does stay on the street, if the trained controller is used.

4. CONCLUSION AND FUTURE WORK

In this paper, we established a criterion for guaranteeing a stable trajectory staying on the street. It enables to determine, whether a controller can lead to an accident by leaving the street or not. It is shown how this criterion, which appears in form of an inequality constraint, can be successfully integrated into a RL approach.

The described procedure is limited in the range of application cases. For us it is of major importance to have a dynamical system, which allows a direct influence of the control into the right hand side of the system. Otherwise, the derived inequality is too restrictive. In future works, we aim at extending our theory to more complex models, which are able to describe the lane following situation even more realistically. We expect that the inequality constraint remains unchanged and only the set of points, which need to fulfill this inequality, varies. The new set may be determined by computing an estimation of a reachability set.

ACKNOWLEDGEMENTS

The authors are grateful for the funding by the Federal Ministry of Education and Research of Germany (BMBF), project number 05M20WNA (SOPRANN). Furthermore, this research has been conducted within the project frame of SeRANIS – Seamless Radio Access Networks in the Internet of Space. The project is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr.

REFERENCES

Abadi, M. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL

https://www.tensorflow.org/. Software available from tensorflow.org.

Abramowitz, M. and Stegun, I.A. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, ninth dover printing edition.

Berkenkamp, F., Turchetta, M., Schoellig, A.P., and Krause, A. (2017). Safe Model-based Reinforcement Learning with Stability Guarantees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

Bertsekas, D.P. (1976). On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 174–184.

Bertsekas, D.P. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific, 1 edition.

Ekachaiworasin, R. and Kuntanapreeda, S. (2000). A Training Rule Which Guarantees Finite-Region Stability of Neural Network Closed-Loop Control: An Extension to Non-Hermitian Systems. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks.*, volume 4, 325–330.

Feinberg, E.A. and Shwartz, A. (2002). *Handbook of Markov Decision Processes: Methods and Applications*. International Series in Operations Research & Management Science. Springer US.

Frankowska, H. (2009). Normality of the maximum principle for absolutely continuous solutions to Bolza problems under state constraints. *Control and Cybernetics*, 38, 1327–1340.

Grinshpan, A. (2005). General inequalities, consequences and applications. *Advances in Applied Mathematics*, 34, 71–100.

Grüne, L. and Junge, O. (2016). *Gewöhnliche Differentialgleichungen*. Springer Spektrum, second edition.

Haber, E. and Ruthotto, L. (2017). Stable Architectures for Deep Neural Networks. *Inverse Problems*, 34(1).

Hale, J.K. (1977). *Theory of Functional Differential Equations*. Springer, New York, NY, 2 edition.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

Khlaeo-om, P., Yenaeng, S., Pasuk, S., Aroonpun, S., and Aumpawan, S. (2011). Stability of Neural Network Control for Uncertain Sampled-Data Systems. In *ESANN, 19th European Symposium on Artificial Neural Networks*, 153–158.

Kisačanin, B. and Agarwal, G.C. (2001). *Linear Control Systems*. Springer US.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer, New York, USA, second edition.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition.

Suykens, J.A., Vandewalle, J., and Moor, B.D. (1999). Lur's Systems with Multilayer Perceptron and Recurrent Neural Networks: Absolute Stability and Dissipativity. In *IEEE Trans. Automatic Control*, volume 44, 770–774.

Williams, R.J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3), 229–256.

Zanon, M. and Gros, S. (2021). Safe Reinforcement Learning Using Robust MPC. *IEEE Transactions on Automatic Control*, 66, 3638–3652.

Appendix A. THE PENALTY DERIVATIVE

For the sake of completeness, we carry out the steps, which are needed in order to rewrite the gradient of the penalty term into an expected value. These steps are the same as used for the actual objective function in Reinforce:

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_u \left[r_{\mathrm{art}}(s_i, t_j, u) \right] \\
=& \int \frac{1}{\sqrt{2\pi\sigma^2}} \nabla_\theta \exp(\bar{u}(s_i, t_j, u))\, r_{\mathrm{art}}(s_i, t_j, u) du \\
=& \mathbb{E}_u \left[\frac{\nabla_\theta \exp(\bar{u}(s_i, t_j, u))}{\exp(\bar{u}(s_i, t_j, u))}\, r_{\mathrm{art}}(s_i, t_j, u) \right] \\
=& \mathbb{E}_u \left[\nabla_\theta \ln \exp(\bar{u}(s_i, t_j, u))\, r_{\mathrm{art}}(s_i, t_j, u) \right] \\
=& \mathbb{E}_u \left[\nabla_\theta \bar{u}(s_i, t_j, u)\, r_{\mathrm{art}}(s_i, t_j, u) \right].
\end{aligned}$$

Appendix B. DETAILS FOR THE NUMERICS

In order to find an upper bound for η_s and η_t in the case of our application example, we add a few intermediate results for the sake of completeness. After computing the derivatives of F_1 and F_2 by means of the chain rule, we end up with the following estimations:

$$\left\| \frac{\partial \left(F_1(s,t) \right)}{\partial t} \right\| \leq D(s,t) \left(\|F_2(t)\| + \|s \dot{n}_{\mathrm{ref}}(t)\| \right), \tag{B.1}$$

$$\left\| \frac{d \left(F_2(t) \right)}{dt} \right\| \leq \left\| f_u \left(x_{\mathrm{ref}}(t), u_{\mathrm{ref}}(t) \right) \right\| \left\| \frac{d u_{\mathrm{ref}}(t)}{dt} \right\|, \tag{B.2}$$

$$\left\| \frac{\partial \left(F_1(s,t) \right)}{\partial s} \right\| \leq D(s,t) \left\| n_{\mathrm{ref}}(t) \right\|, \text{ for} \tag{B.3}$$

$$D(s,t) \coloneqq \left\| f_u \big(x(s,t), u_{\mathrm{NN}}(x(s,t)) \big) \right\| \left\| (u_{\mathrm{NN}})_x \left(x(s,t) \right) \right\|.$$

The estimate $\|f_u(x,u)\| \leq 2.5\pi$ can be deduced by considering the eigenvalues of the matrix:

$$f_u(x,u)^T f_u(x,u) = \begin{bmatrix} 1 & 0 \\ 0 & (v+1.5)^2 \pi^2 \end{bmatrix}. \tag{B.4}$$

For the training, we used a batch of 2500 trajectories. The penalty term in the objective function is computed on a coarse grid, which turns out to be enough for the training part. For the final assurance the constraint has to be fulfilled on the finer grid, which we checked after the training. The ingredients of the cost function are based on the arc length from the starting point to the projection of the current state to the reference trajectory (quadratic influence), the distance to the obstacle (linear) and the distance to the left (linear) and right boundary (linear). We used a fixed learning rate of the NN $\alpha \coloneqq 10^{-7}$. The input layer of the NN has the size five and gets the normalized positions, distance to rotation center of the semicircle reference trajectory, distance to obstacle center and the arc length of its projected position. The hidden layers contain 20, respectively 16, neurons. The output layer has two neurons, which represent the manipulating quantities: velocity and steering angle. Each neuron uses the hyperbolic tangent. The main algorithm as well as the implementation of the environment, the policy and the class for checking the stability are implemented in Python 3.8.8. In case of the policy, we make use of the open source platform TensorFlow 2.4.1 (Abadi et al., 2015), which enables an efficient implementation of the underlying NN. For generating the trajectories we parallelize the computations on an AMD Ryzen Threadripper 3990X with 64-Cores and a boost clock of 4.3 GHz.