Proximal Policy Optimization for Tracking Control Exploiting Future Reference Information

Jana Mayer[a], *Johannes Westermann*[a], *Juan Pedro Gutiérrez H. Muriedas*[a], *Uwe Mettin*[b], *Alexander Lampe*[b]

[a]*Intelligent Sensor-Actuator-Systems Laboratory (ISAS)*
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
[b]*IAV GmbH, Berlin, Germany*

Abstract

In recent years, reinforcement learning (RL) has gained increasing attention in control engineering. Especially, policy gradient methods are widely used. In this work, we improve the tracking performance of proximal policy optimization (PPO) for arbitrary reference signals by incorporating information about future reference values. Two variants of extending the argument of the actor and the critic taking future reference values into account are presented. In the first variant, global future reference values are added to the argument. For the second variant, a novel kind of residual space with future reference values applicable to model-free reinforcement learning is introduced. Our approach is evaluated against a PI controller on a simple drive train model. We expect our method to generalize to arbitrary references better than previous approaches, pointing towards the applicability of RL to control real systems.

1. Introduction

In cars with automatic transmissions, gear shifts shall be performed in such a way that no discomfort is caused by the interplay of motor torque and clutch control action. Especially, the synchronization of input speed to the gears' target speed is a sensitive process mainly controlled by the clutch. Ideally, the motor speed should follow a predesigned reference, but not optimal operating feedforward control and disturbances in the mechanical system can cause deviations from the optimal behavior. The idea is to apply a reinforcement learning (RL) approach to control the clutch behavior regulating the deviations from the optimal reference. The advantage of a RL approach over classical approaches as the PI control is that no extensive experimental parametrization for every gear and every clutch is needed which is very complex for automatic transmissions. Instead, the RL algorithm is supposed to learn the optimal control behavior autonomously. Since the goal is to guide the motor speed along a given reference signal the problem at hand belongs to the family of tracking control problems.

In the following, a literature review on RL for tracking control is given. In [1], the deep deterministic policy gradient method, introduced in [2], is applied to learn the parameters of a PID controller. An adaptive PID controller is realized in [3] using an incremental Q-learning for real-time tuning. A combination of Q-learning and a PID controller is presented in [4], where the applied control input is a sum of the PID control input and a control input determined by Q-learning.

Another common concept applied to tracking control problems is model predictive control (MPC) which can also be combined with RL. A data-efficient model-based RL approach based on probabilistic

Email addresses: `jana.mayer@kit.edu` (Jana Mayer), `johannes.westermann@kit.edu` (Johannes Westermann), `juanpedroghm@gmail.com` (Juan Pedro Gutiérrez H. Muriedas), `uwe.mettin@iav.de` (Uwe Mettin), `alexander.lampe@iav.de` (Alexander Lampe)

model predictive control (MPC) is introduced in [5]. The key idea is to learn the probabilistic transition model using Gaussian processes. In [6], nonlinear model predictive control (NMPC) is used as a function approximator for the value function and the policy in a RL approach. For tracking control problems with linear dynamics and quadratic costs a RL approach is presented in [7]. Here, a Q-function is analytically derived that inherently incorporates a given reference trajectory on a moving horizon.

In contrast to the before presented approaches derived from classical controller concepts also pure RL approaches for tracking control were invented. A model-based variant is presented in [8] where a kernel-based transition dynamic model is introduced. The transition probabilities are learned directly from the observed data without learning the dynamic model. The model of the transition probabilities is then used in a RL approach.

A model-free RL approach is introduced in [9] where the deep deterministic policy gradient approach [2] is applied on tracking control of an autonomous underwater vehicle. In [10], images of a reference are fed to a convolutional neural network for a model-free state representation of the path. A deep deterministic policy gradient approach [2] is applied where previous local path images and control inputs are given as arguments to solve the tracking control problem. Proximal policy optimization (PPO) [11] with generalized advantage estimation (GAE) [12] is applied on tracking control of a manipulator and a mobile robot in [13]. Here, the actor and the critic are represented by a long short-term memory (LSTM) and a distributed version of PPO is used.

In this work, we apply PPO to a tracking control problem. The key idea is to extend the arguments of the actor and the critic to take into account information about future reference values and thus improve the tracking performance. Besides adding global reference values to the argument, we also define an argument based on residua between the states and the future reference values. For this purpose, a novel residual space with future reference values is introduced applicable to model-free RL approaches. Our approach is evaluated on a simple drive train model. The results are compared to a classical PI controller and a PPO approach, which does not consider future reference values.

2. Problem Formulation

In this work, we consider a time-discrete system with non-linear dynamics

$$\underline{x}_{k+1} = f\left(\underline{x}_k, \underline{u}_k\right) \quad , \tag{1}$$

where $\underline{x}_k \in \mathbb{R}^{n_x}$ is the state and $\underline{u}_k \in \mathbb{R}^{n_u}$ is the control input applied in time step k. The system equation (1) is assumed to be unknown for the RL algorithm. Furthermore, the states $\underline{x}_k$ are exactly known.

In the tracking control problem, the state or components of the state are supposed to follow a reference $\underline{x}_k^r \in \mathbb{R}^{n_h}$. Thus, the goal is to control the system in a way that the deviation between the state $\underline{x}_k$ and the reference $\underline{x}_k^r$ becomes zero in all time steps. The reference is assumed to be given and the algorithm should be able to track before unseen references.

To reach the goal, the algorithm can learn from interactions with the system. In policy gradient methods, a policy is determined which maps the RL states $\underline{s}_k$ to a control input $\underline{u}_k$. The states $\underline{s}_k$ can be the system states $\underline{x}_k$ but can also contain other related components. In actor-critic approaches, the policy is represented by the actor. Here, $\underline{s}_k$ is the argument given to the actor and in most instances also to the critic as input. To prevent confusion with the system state $\underline{x}_k$, we will refer to $\underline{s}_k$ as argument in the following.

3. Existing solutions and challenges

In existing policy gradient methods for tracking control, the argument $\underline{s}_k$ is either identical to the system state $\underline{x}_k$ [9] or is composed of the system state and the residuum between the system state and the reference in the current time step [13]. Those approaches show good results if the reference is fixed. Applied on arbitrary references the actor can only respond to the current reference value but is not able to act optimally for the subsequent reference values. In this work, we will show that this results in poor performance.

Another common concept, applied on tracking control problems, is model predictive control (MPC), e.g., [5]. Here, the control inputs are determined by predicting the future system states and minimizing their deviation from the future reference values. In general, the optimization over a moving horizon has to be executed in every time step as no explicit policy representation is determined. Another disadvantage of MPC is the need to know or learn the model of the system.

Our idea is to transfer the concept of utilizing the information given in form of known future reference values from MPC to policy gradient methods. A first step in this direction is presented in [7], where an adaptive optimal control method for reference tracking was developed. Here, a Q-function could be analytically derived by applying dynamic programming. The received Q-function depends inherently on the current state but also on current and future reference values. However, the analytical solution is limited to the case of linear system dynamics and quadratic costs (rewards). In this work, we transfer those results to a policy gradient algorithm by extending the arguments of the actor and the critic with future reference values. In contrast to the linear quadratic case, the Q-function cannot be derived analytically. Accordingly, the nonlinear dependencies are approximated by the actor and the critic. The developed approach can be applied to tracking control problems with nonlinear system dynamics and arbitrary references. In some applications, the local deviation of the state from the reference is more informative than the global state and reference values, e.g. operating the drive train in different speed ranges. In this case, it can be beneficial if the argument is defined as residuum between the state and the corresponding reference value, because this scales down the range of the state space has to be explored. Thus, we introduce a novel kind of residual space between states and future reference values which can be applied without knowing or learning the system dynamics. The key ideas are (1) extending the arguments of the actor and the critic of a policy gradient method by future reference values, and (2) introducing a novel kind of residual space for model-free RL.

4. Preliminaries: Proximal Policy Optimization Algorithm

As policy gradient method, proximal policy optimization (PPO) [11] is applied in this work. PPO is a simplification of the trust region policy optimization (TRPO) [14]. The key idea of PPO is a novel loss function design where the change of the stochastic policy $\pi_{\underline{\theta}}$ in each update step is limited introducing a clip function

$$J(\underline{\theta}) = \mathbb{E}_{(\underline{s}_k,\underline{u}_k)}\{\min\left(p_k(\underline{\theta})A_k, \text{clip}\left(p_k(\underline{\theta}), 1-c, 1+c\right)A_k\right)\} \ , \tag{2}$$

where

$$p_k(\underline{\theta}) = \frac{\pi_{\underline{\theta}}(\underline{u}_k|\underline{s}_k)}{\pi_{\underline{\theta}_{old}}(\underline{u}_k|\underline{s}_k)} \ . \tag{3}$$

The clipping motivates $p_k(\underline{\theta})$ not to leave the interval $[1-c, 1+c]$. The argument $\underline{s}_k$ given to the actor commonly contains the system state $\underline{x}_k$ of the system, but can be extended by additional information. The loss function is used in a policy gradient method to learn the actor network's parameters $\underline{\theta}_h$

$$\underline{\theta}_{h+1} = \underline{\theta}_h + \alpha_a \cdot \nabla_{\underline{\theta}} J(\underline{\theta})|_{\underline{\theta}=\underline{\theta}_h} \ ,$$

where α_a is referred as the actor's learning rate and $h \in \{0, 1, 2, \ldots\}$ is the policy update number. A proof of convergence for PPO is presented in [15]. The advantage function A_k in (2) is defined as the difference between the Q-function and the value function V

$$A_k(\underline{s}_k, \underline{u}_k) = Q(\underline{s}_k, \underline{u}_k) - V(\underline{s}_k) \ .$$

In [11], generalized advantage estimation (GAE) [12] is applied to approximate the value function

$$\hat{A}_k^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{k+l}^V \ , \tag{4}$$

where δ_k^V is the temporal difference error of the value function V [16]

$$\delta_k^V = r_k + \gamma V(\underline{s}_{k+1}) - V(\underline{s}_k) \ . \tag{5}$$

The discount $\gamma \in [0, 1]$ reduces the influence of future incidences. $\lambda \in [0, 1]$ is a design parameter of GAE. Note, the value function also has to be learned during the training process and serves as critic in the approach. The critic is also represented by a neural network and and receives the same argument as the actor. To ensure sufficient exploration the actor's loss function (2) is commonly extended by an entropy bonus $S[\pi_{\underline{\theta}}](\underline{s}_k)$ [17], [18]

$$J(\underline{\theta}) = \mathbb{E}_{(\underline{s}_k, \underline{u}_k)}\{\min\left(p_k(\underline{\theta})A_k, \text{clip}\left(p_k(\underline{\theta}), 1-c, 1+c\right) A_k\right) + \mu\, S[\pi_{\underline{\theta}}](\underline{s}_k)\} \ ,$$

where $\mu \geq 0$ is the entropy coefficient.

5. Proximal Policy Optimization for Tracking Control with future references

As mentioned before, the key idea of the presented approach is to add the information of future reference values to the argument $\underline{s}_k$ of the actor and the critic in order to improve the control quality. PPO was already applied to a tracking control problem in [13] where the argument $\underline{s}_k^0$, contains the current system state $\underline{x}_k$ as well as the residuum of $\underline{x}_k$ and the current reference $\underline{x}_k^r$

$$\underline{s}_k^0 = \left[\underline{x}_k, (\underline{x}_k^r - \underline{x}_k)\right]^T \ . \tag{6}$$

However, no information of future reference values is part of the argument. We take advantage of the fact that the future reference values are known and incorporate them to the argument of the actor and the critic. In the following, two variants will be discussed:

(1) Besides the system state $\underline{x}_k$ N future reference values are added to the argument

$$\underline{s}_k^{1N} = \left[\underline{x}_k, \underline{x}_k^r, \underline{x}_{k+1}^r, \ldots, \underline{x}_{k+N}^r\right]^T , \ \ N \in \mathbb{N} \ . \tag{7}$$

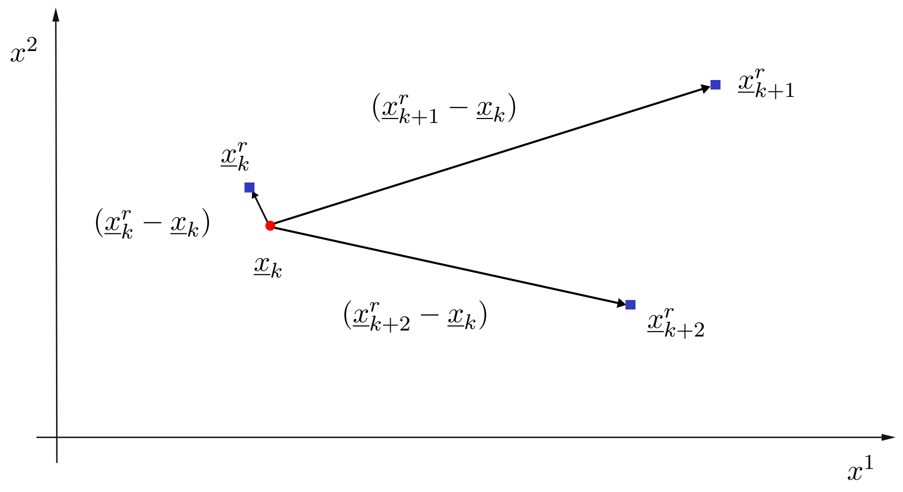
(2) We introduce a novel residual space where the future reference values are related to the current state and the argument is defined as

$$\underline{s}_k^{2N} = \left[\left(\underline{x}_k^r - \underline{x}_k\right), \left(\underline{x}_{k+1}^r - \underline{x}_k\right), \ldots, \left(\underline{x}_{k+N}^r - \underline{x}_k\right)\right]^T . \tag{8}$$

In Figure 1, the residual space is illustrated for two-dimensional states and references. Being in the current state $\underline{x}_k = [x_k^1, x_k^2]^T$ (red dot) the the residua between the current state and the future reference values (black arrows) indicate if the components of the state x_k^1 and x_k^2 have to be increased or decreased to reach the reference values $\underline{x}_{k+1}^r$ and $\underline{x}_{k+2}^r$ in the next time steps. Thus, the residual argument gives sufficient information to the PPO algorithm to control the system.

Please note, a residual space containing future states $\left(\underline{x}_{k+1}^r - \underline{x}_{k+1}\right), \ldots, \left(\underline{x}_{k+N}^r - \underline{x}_{k+N}\right)$ would suffer from two disadvantages. First, the model has to learned which would increase the complexity of the algorithm. Second, the future states $\underline{x}_{k+1}, \ldots, \underline{x}_{k+N}$ depend on the current policy thus the argument is a function of the policy. Applied as argument in (3) the policy becomes a function of the policy $\pi_{\underline{\theta}}(\underline{u}_k|\underline{s}_k(\pi_{\underline{\theta}}))$. This could be solved by calculating the policy in a recursive manner where several instances of the policy are trained in form of different neural networks. This solution would lead to a complex optimization problem which is expected to be computationally expensive and hard to stabilize in the training. Therefore, we consider this solution as impractical. On the other hand, the residual space defined in (8) contains all information about the future course of the reference, consequently a residual space including future states would not enhance the information content.

In the residual space, the arguments are centered around zero, independent of the actual values of the state or the reference. The advantage, compared to the argument with global reference values, is that only the deviation of the state from the reference is represented, which scales down the range of



x^2
$(\underline{x}^r_{k+1} - \underline{x}_k)$
$\underline{x}^r_{k+1}$
$\underline{x}^r_k$
$(\underline{x}^r_k - \underline{x}_k)$
$\underline{x}_k$
$(\underline{x}^r_{k+2} - \underline{x}_k)$
$\underline{x}^r_{k+2}$
x^1

Figure 1: Residual space defined by current state and future reference values.

the state space has to be learned. But the residual space argument is only applicable if the change in the state for a given control input is independent of the state itself.

As part of the tracking control problem a reward function has to be designed. The reward should represent the quality of following the reference. Applying the control input $\underline{u}_k$ in the state $\underline{x}_k$ leads to the state $\underline{x}_{k+1}$ and related to this a reward depending on the difference between $\underline{x}_{k+1}$ and the reference $\underline{x}^r_{k+1}$. Additionally, a punishment of huge control inputs can be appended. The resulting reward function for time step k is defined as

$$r_k = -\left(\underline{x}_{k+1} - \underline{x}^r_{k+1}\right)^2 - \beta \cdot \underline{u}_k^2 \ , \tag{9}$$

where $\beta \geq 0$ is a weighting parameter.

6. Simple Drive Train Model

An automatic transmission provides gear shifts characterized by two features: (1) the torque transfer, in which a clutch corresponding to the target gear takes over the drive torque, and (2) the speed synchronization, in which slip from input to output speed of the clutch is reduced such that it can be closed or controlled at low slip. In this work, we consider the reference tracking control problem for a friction clutch during synchronization phase. Its input is driven by a motor and the output is propagated through the gearbox to the wheels of the car. Speed control action can only be applied by the clutch when input and output force plates are in friction contact with slip. Generally, the aim in speed synchronization is to smoothly control the contact force of the clutch without jerking towards zero slip. This is where our RL approach for tracking control comes into account. For a smooth operation a reference to follow during the friction phase is predesigned by the developer of the drive train.

For easier understanding, a simple drive train model is used which is derived from the motion equations of an ideal clutch [19] extended by the influence of the gearbox

$$J_{in}\,\dot{\omega}_{in} = -T_{cl} + T_{in} \ , \tag{10}$$

$$J_{out}\,\dot{\omega}_{out} = \theta\, T_{cl} - T_{out} \ , \tag{11}$$

where ω_{in} is the input speed on the motor side and ω_{out} is the output speed at the side of the wheels. Accordingly, T_{in} is the input torque and T_{out} is the output torque. The transmission ratio θ of the gearbox defines the ratio between the input and output speed. The input and output moment of inertia J_{in}, J_{out} and the transmission ratio θ are fixed characteristics of the drive train. The clutch is controlled varying the torque transmitted from the clutch T_{cl}.

The input torque T_{in} is approximated as constant while the output torque is assumed to depend linear on the output speed $T_{out} = \eta \cdot \omega_{out}$ which changes (10) and (11) to

$$\begin{aligned} J_{in}\,\dot{\omega}_{in} &= -T_{cl} + T_{in} \ , \\ J_{out}\,\dot{\omega}_{out} &= \theta\, T_{cl} - \eta \cdot \omega_{out} \ . \end{aligned}$$

Solving the differential equations for a time interval ΔT, yields the discrete system equation

$$\begin{bmatrix} \omega_{in} \\ \omega_{out} \end{bmatrix}_{k+1} = \boldsymbol{A} \begin{bmatrix} \omega_{in} \\ \omega_{out} \end{bmatrix}_{k} + \boldsymbol{B}_1 \cdot T_{cl,k} + \boldsymbol{B}_2 \cdot T_{in} \ , \tag{12}$$

where

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(-\frac{\eta \cdot \Delta T}{J_{out}}\right) \end{bmatrix} \ , \\ \boldsymbol{B}_1 &= \begin{bmatrix} -\frac{\Delta T}{J_{in}} \\ \frac{\theta}{\eta}\left(1 - \exp\left(-\frac{\eta \cdot \Delta T}{J_{out}}\right)\right) \end{bmatrix} \ , \quad \boldsymbol{B}_2 = \begin{bmatrix} \frac{\Delta T}{J_{in}} \\ 0 \end{bmatrix} \ , \end{aligned}$$

with state $\underline{x}_k = [\omega_{in}\omega_{out}]_k^T$ and the control input $u_k = T_{cl,k}$. For a friction-plate clutch, the clutch torque $T_{cl,k}$ depends on the capacity torque $T_{cap,k}$

$$T_{cl,k} = T_{cap,k} \cdot \operatorname{sign}\left(\omega_{in,k} - \theta \cdot \omega_{out,k}\right) \, , \;\; T_{cap,k} \geq 0 \ , \tag{13}$$

which means $T_{cl,k}$ is changing its sign according to if the input speed or the output speed is higher. The capacity torque is proportional to the contact force which is applied on the plates $T_{cap,k} \sim F_{N,k}$.

In real drive trains, the change of the control input in a time step is limited due to underlying dynamics such as pressure dynamics. For simulating this behavior, a low pass filter is applied on the control inputs. We use a first-order lag element (PT1) with the cutoff frequency f_g

$$T'_{cl,k} = \begin{cases} \left(T_{cl,k} - T'_{cl,k-1}\right) \cdot (1-a) + T'_{cl,k-1}, & \text{if } T_{cl,k} > T'_{cl,k-1} \ , \\ \left(T'_{cl,k-1} - T_{cl,k}\right) \cdot a + T_{cl,k}, & \text{otherwise} \ , \end{cases}$$

where

$$a = \exp\left(-2\pi f_g \Delta T\right) \ .$$

In this case, the control input provided by the controller $T_{cl,k}$ is transformed to the delayed control input $T'_{cl,k}$ applied on the system.

The simple drive train model is used in three different experiments of input speed synchronization which are derived from use cases of real drive trains. In the first experiment, the references to be followed are smooth. This is associated with a usual gear shift. In real applications, delays, hidden dynamics, or change-of-mind situations can arise which make a reinitialization of the reference necessary. This behavior is modeled in the second experiment where the references contain jumps and discontinuities, respectively. Jumps in the reference can lead to fast changes in the control inputs. To still maintain a realistic behavior the lag element is applied on the control inputs. In the third experiment, we use again smooth references but in different input speed ranges. Varying drive torque demands typically cause the gear shift to start at different input speed levels. Our approach will be evaluated on all three experiments to determine the performance for different use cases. Note, that for demonstration purposes we consider speed synchronization by clutch control input only. In real applications, clutch control is often combined with input torque feedforward.

Table 1: Parameters of the simple drive train model.

Parameter	Value
Input moment of inertia J_{in}	$0.209\,\mathrm{kgm}^2$
Output moment of inertia J_{out}	$86.6033\,\mathrm{kgm}^2$
Transmission ratio θ	10.02
Input torque T_{in}	20 Nm
η	2 (Nms)/rad
Time step ΔT	10 ms

7. Evaluation

As mentioned in the last chapter, we will evaluate our approach using three different experiments representing three use cases of the drive train. In every experiment, the three different arguments of the actor and the critic introduced in Chapter 5 will be applied.

For the drive train system (12), the arguments of the actor and the critic have to be defined. The reference to be followed is only corresponding to the input speed. Since no reference for the output speed is given we add the output speed $\omega_{out,k}$ as global variable in all three arguments. Analogous to (6), (7) and (8), the arguments of the drive train system are

$$\begin{aligned}
\underline{s}_k^{0,cl} &= \left[\omega_{out,k}, \omega_{in,k}, \left(\omega_{in,k}^r - \omega_{in,k}\right)\right]^T \quad , \\
\underline{s}_k^{1N,cl} &= \left[\omega_{out,k}, \omega_{in,k}, \omega_{in,k}^r, \omega_{in,k+1}^r, \ldots, \omega_{in,k+N}^r\right]^T \quad , \\
\underline{s}_k^{2N,cl} &= \left[\omega_{out,k}, \left(\omega_{in,k}^r - \omega_{in,k}\right), \left(\omega_{in,k+1}^r - \omega_{in,k}\right), \ldots, \left(\omega_{in,k+N}^r - \omega_{in,k}\right)\right]^T .
\end{aligned}$$

Without future reference values as applied in [13] the argument is $\underline{s}_k^{0,cl}$, with global future reference values $\underline{s}_k^{1N,cl}$ and in residual space with future reference values $\underline{s}_k^{2N,cl}$.

The reward function for the simple drive train model derived from (9) is given as

$$r_k = -\left(\omega_{in,k+1} - \omega_{in,k+1}^r\right)^2 - \beta \cdot \left(T_{cl,k} - T_{in}\right)^2 \quad . \tag{14}$$

If $T_{cl,k} = T_{in}$ the input speed ω_{in} is not changing from time step k to time step $k+1$ according to (12). Thus, deviations of $T_{cl,k}$ from T_{in} are penalized to suppress control inputs which would cause larger changes in the state. All parameters used in the simple drive train model are given in Table 1.

7.1. Algorithm

The PPO algorithm, applied for the evaluation, is shown in Algorithm 1. We use two separate networks representing the actor and the critic. In order to improve data-efficiency we apply experience replay [20], [21]. While interacting with the system, the tuples $(\underline{s}_k, u_k, r_k, \underline{s}_{k+1})$ are stored in the replay buffer. In every epoch a training batch of size L is sampled from the replay buffer to update the actor's and the critic's parameters. As introduced in Chapter 4, the advantage function is determined via generalized advantage estimation (4) with the GAE parameter set to $\lambda = 0$ and the discount $\gamma = 0.7$. To improve the stability of the critic network during the training, we added a target critic network V' [21] which is only updated every m-th epoch (in our implementation $m = 2$). The critic loss is defined as the mean squared temporal difference error (5)

$$C(\underline{\phi}) = \frac{1}{L}\sum^{L}\left(r_k + \gamma\, V'(\underline{s}_{k+1}) - V(\underline{s}_k)\right)^2 \quad .$$

The output of the actor provides T_{cap} in (13) and the sign for T_{cl} is calculated through the input and output speed. To ensure $T_{cap} \geq 0$ the last activation function of the actor is chosen as ReLU. In Table 2, all parameters used in the PPO algorithm are shown. In our trainings, we perform 2000 episodes each with 100 time steps. Each run through the system is followed by 100 successive training epochs.

Algorithm 1 PPO for tracking control

```
Require: Replay buffer $\mathcal{D}$, critic parameters $\underline{\phi}$, actor parameters $\underline{\theta}$, actor learning rate $\alpha_a$, critic
    learning rate $\alpha_c$, target network delay factor $\tau$
 1: Init target critic parameters $\underline{\phi}' \leftarrow \underline{\phi}$ and $h = 0$
 2: for 1 .. Number of episodes do
 3:     Observe initial state $\underline{x}_0$ and new reference $\omega_{in}^r$
 4:     for 1 .. K do
 5:         Apply control input $u_k \leftarrow \pi_\theta(\underline{s}_k)$
 6:         Observe new state $\underline{x}_{k+1}$ and reward $r_k$
 7:         Add $(\underline{s}_k, u_k, r_k, \underline{s}_{k+1})$ to replay buffer $\mathcal{D}$
 8:     end for
 9:     for 1 .. Number of epochs do
10:         Sample training batch from $\mathcal{D}$
11:         Update critic $\underline{\phi}_{h+1} \leftarrow \underline{\phi}_h + \alpha_c \nabla_{\underline{\phi}} C(\underline{\phi}_h)$
12:         Calculate advantage $\underline{\hat{A}}$ using GAE
13:         Update actor $\underline{\theta}_{h+1} \leftarrow \underline{\theta}_h + \alpha_a \nabla_{\underline{\theta}} J(\underline{\theta}_h)$
14:         $h \leftarrow h + 1$
15:         Every m-th epoch
                Update target critic $\underline{\phi}' \leftarrow (1-\tau)\,\underline{\phi}' + \tau\,\underline{\phi}$
16:     end for
17: end for
```

7.2. Simulation procedure

In the following, we will evaluate our approach on three different experiments. For each experiment 15 independent simulations are performed. In each simulation the algorithm is trained using 2000 different training references (one for each episode). After every tenth episode of the training the actor is tested on an evaluation reference. Using the evaluation results the best actor of the training is identified. In the next step, the best actor is applied to 100 test references to evaluate the performance of the algorithm. In the following, the mean episodic reward over all 100 test references and all 15 simulations will serve as quality criterion.

The smooth references are cubic splines formed from eight random values around the mean 2000 rpm. In the second experiment, between one and 19 discontinuities are induced by adding periodic square-waves of different period durations to the reference. The references in the third experiment are shifted by adding an offset to the spline. Five different offsets are used, resulting in data between approximately 1040 rpm and 4880 rpm.

Table 2: Parameters used in the PPO algorithm.

Parameter	Value
Hidden layers size actor and critic networks	[400, 300]
Activation functions actor network	[tanh, tanh, ReLU]
Activation functions critic network	[tanh, tanh, -ReLU]
Actor's learning rate α_a	$5 \cdot 10^{-5}$
Critic's learning rate α_c	$1.5 \cdot 10^{-3}$
Batch size L	100
Reply buffer size	10000
Initial standard deviation stochastic actor	10
c	0.1
τ	0.001
Entropy coefficient μ	0.01

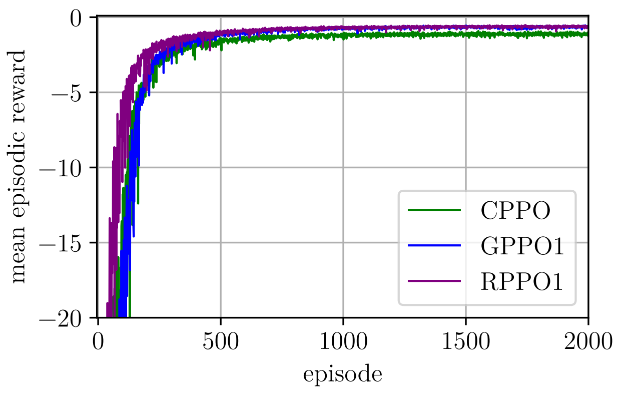


Figure 2: Training over 2000 episodes on smooth references.

As mentioned before, the results will be compared to the performance of a PI controller. We determine the parameters of the PI controller by minimizing the cumulative costs over an episode, respectively the negative of the episodic reward defined by the reward function (14). To avoid local minima, we first apply a grid search and then use a quasi-Newton method to determine the exact minimum. In the optimization, the cumulative costs over all training references (same as in the PPO training) are used as quality criterion.

7.3. Results

In the following, the results of the different experiments with (1) a class of smooth references, (2) a class of references with discontinuities and (3) a class of smooth references shifted by an offset will be presented.

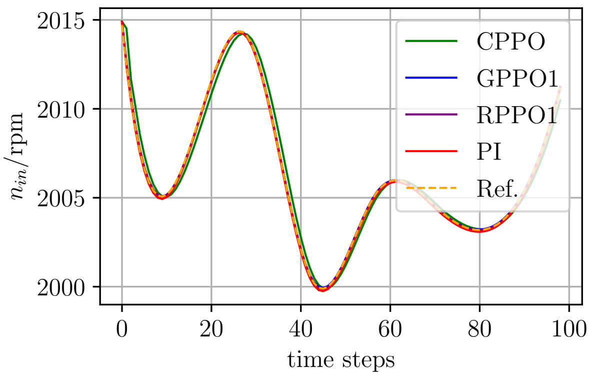
Class of smooth references

For smooth references the weighting parameter β in the reward function (14) is set to zero. The parameters of the PI controller were computed as described in Section 7.2, the parameter of the proportional term K_P was determined as 20.88 and the parameter of the integral term K_I is 11.08. The training curves of the PPO approaches are drawn in Figure 2. It can be clearly seen that the approaches using one future reference value in the argument of the actor and the critic (GPPO1, RPPO1) reach a higher mean episodic reward than the approach including only reference information of the current time step (CPPO). Furthermore, the approach with the argument defined in the residual space (RPPO1) achieves high rewards faster than the GPPO1 and the CPPO.

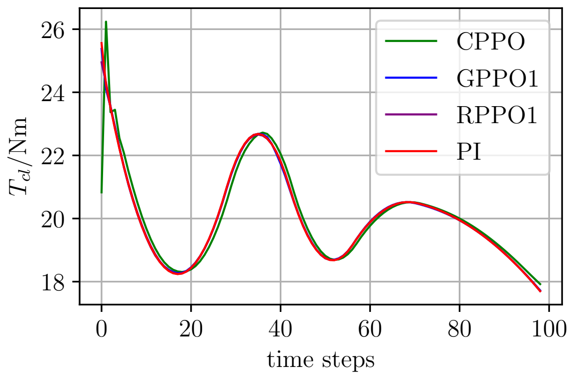
As mentioned before, the trained actors are evaluated on a set of test references. The obtained results are given in Table 3. The approach using global reference values in the argument (GPPO1) and the approach defining the argument in a residual space (RPPO1) provide the best results. The mean episodic reward of the CPPO is ten times lower than the ones of the GPPO1 and the RPPO1.

Table 3: Quality evaluated on test a set for smooth references.

Arguments of actor and critic	Acronym	Mean episodic reward	Standard deviation episodic reward
Current state and residuum ($\underline{s}_k^{0,cl}$)	CPPO	-0.303	0.027
Global space with one future reference ($\underline{s}_k^{11,cl}$)	GPPO1	-0.035	0.020
Residual space with one future reference ($\underline{s}_k^{21,cl}$)	RPPO1	-0.030	0.020
PI controller	PI	-0.069	1.534



(a) Input speeds and reference for one episode.



(b) Control inputs for one episode.

Figure 3: Performance of the best trained actors on a smooth test reference. Note that in real applications reference trajectories are designed in such a way that the input speed is synchronized quickly but smoothly towards zero slip at the clutch.

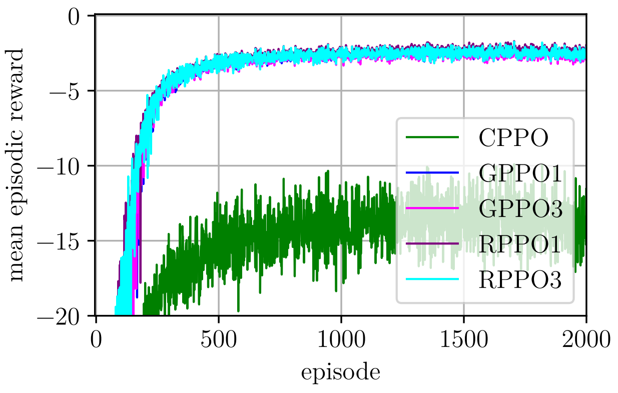


Figure 4: Training over 2000 episodes on references with discontinuities.

The performance of the classical PI controller lies in between them but the standard deviation is very high. This implies that optimizing the PI's parameters using the training data, no parameters can be determined which lead to equally good performance for all the different references.

In Figure 3, the best actors are applied to follow a test reference. The tracks of the PI controller, the GPPO1 and the RPPO1 show similar behavior. Only in the valleys the PI controller deviated slightly more from the reference. Since the CPPO has to decide for a control input only knowing the current reference value, its reaction is always behind. In Figure 3a, it can be seen that the CPPO applies a control input that closes the gap between the current input speed and the current reference value. But in the next time step the reference value has already changed and is consequently not reached. The same effect leads to a shift of the control inputs in Figure 3b.

Class of references with discontinuities

As mentioned before, to avoid unrealistic fast changes of the control inputs we add a first-order lag element with cutoff frequency $f_g = 100\,\mathrm{Hz}$ to the traindrive model in the experiment setting for references with discontinuities. In addition, β is set to 1/3000 in the reward function (14) to prevent the learning of large control inputs when discontinuities occur in the reference signal. The PI controller parameters were optimized on the same reward function resulting in $K_P = 18.53$ and $K_I = 5.67$. Besides including the information of one future reference value the performance of adding three future reference values (GPPO3 and RPPO3) is also investigated in this experiment. The learning curves of all five PPO approaches are illustrated in Figure 4. All approaches with future reference values show

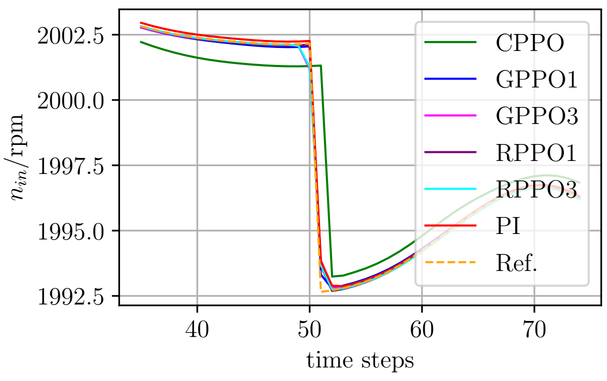
Table 4: Quality evaluated on test references with discontinuities.

Arguments of actor and critic	Acronym	Mean episodic reward	Standard deviation episodic reward
Current state and residuum ($\underline{s}_k^{0,cl}$)	CPPO	-10.936	0.618
Global space with one future reference ($\underline{s}_k^{11,cl}$)	GPPO1	-1.597	0.079
Global space with three future reference ($\underline{s}_k^{13,cl}$)	GPPO3	-1.499	0.082
Residual space with one future reference ($\underline{s}_k^{21,cl}$)	RPPO1	-1.614	0.088
Residual space with three future reference ($\underline{s}_k^{23,cl}$)	RPPO3	-1.510	0.086
PI controller	PI	-1.747	2.767

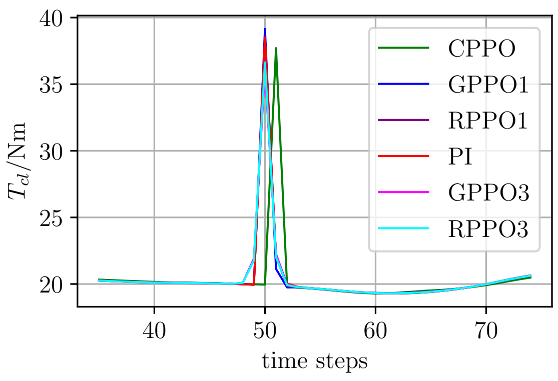
similar results in the training. The PPO approach containing only current reference values (CPPO) achieves a significant lower mean episodic reward.

A similar behavior can be observed in the evaluation on the test reference set for which the results are given in Table 4. Again, the PPO approaches with one global future reference (GPPO1) and one future reference in the residual space (RPPO1) show similar results and perform significantly better than the CPPO. Adding the information of more future reference values (GPPO3 and RPPO3) leads to an even higher mean episodic reward. The PI controller performs slightly worse than the PPO aproaches with future reference information. Due to the large standard deviation of the PI controller's performance, it cannot be guaranteed to perform well on a specific reference.

The performance of the best actors on a test reference with discontinuities is illustrated in Figure 5a. Note, for better visibility, the figure shows only a section of a full episode. The PPO approaches including future reference values and the PI controller are able to follow the reference even if a discontinuity occurs. The PPO approach without future reference values is performing even worse than on the smooth reference. It can be clearly seen that the jump cannot be detected in advance and the reaction of the actor is delayed. In Figure 5a the applied control inputs are shown, the approaches including three future reference values respond earlier to the upcoming discontinuities and cope with the discontinuity requiring only smaller control inputs (Figure 5b) which also leads to a higher reward. The PI controller shows a similar behavior than the PPO approaches with one future reference value (GPPO1 and RPPO1).



(a) Input speeds and reference.



(b) Control inputs.

Figure 5: Performance of the best trained actors on a test reference with discontinuities.

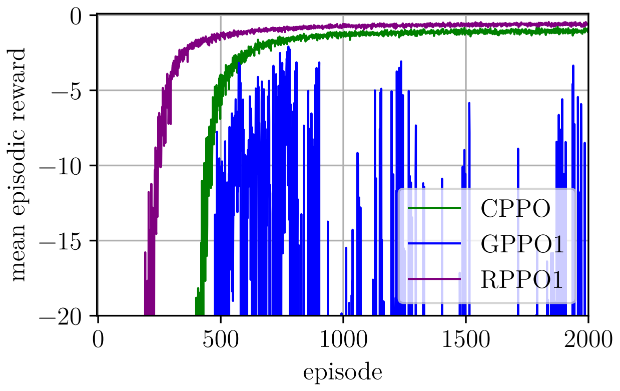


Figure 6: Training over 2000 episodes on smooth references with offsets.

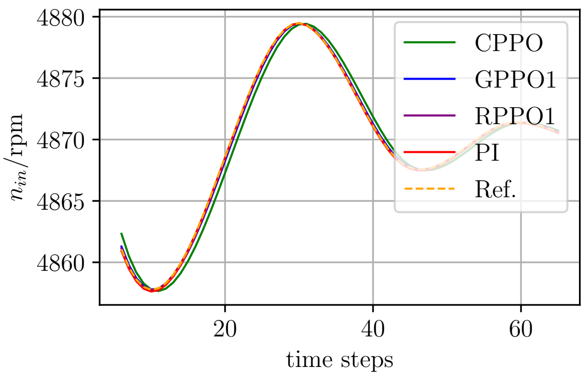
Class of smooth references with offsets

For the class of smooth references with offsets, the same settings, as used in the experiment for smooth references, are applied. The PI parameters are determined as 20.88 for K_P and 11.04 for K_I. The training curves of the PPO approaches are illustrated in Figure 6. Compared to the learning curves of the smooth references experiment the training on the references with offset needs more episodes to reach the maximum episodic reward. The training of the approach using global reference values (GPPO1) is extremely unstable. The reason might be that the training data contains only references in a finite number of small disjointed areas in a huge state space. As the argument of GPPO1 includes only global states and reference value no shared data between this areas exist and the policy is learned independently for each area. Consequently, the stability of the approaches with argument components in the residual space CPPO and RPPO is not effected by this issue. However, the RPPO1 is receiving higher rewards significantly faster than the CPPO which also contains the global input speed.

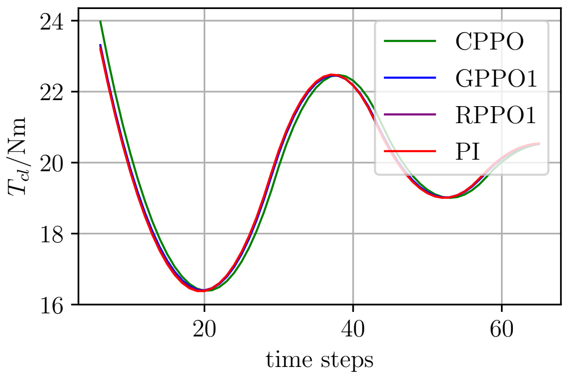
In Table 5, the performance on a set of test references is given. The mean episodic reward of the CPPO, the RPPO1 and the PI controller is in the same range as in the smooth references experiment. As in the other experiments, the PI controller shows a large standard deviation. Despite the instabilities in the training, the GPPO1 receives a higher mean episodic reward than the CPPO but its performance is significant worse than in the smooth references experiment. It can be clearly seen that residual space arguments lead to better performance for references with offsets. In Figure 7, the performance of the best actors on a test reference is illustrated. For better visibility, only a section of a full episode is drawn. The results are similar to the smooth reference experiment, only the GPPO1 shows slightly worse tracking performance.

Table 5: Quality evaluated on a test set for smooth references with offset.

Arguments of actor and critic	Acronym	Mean episodic reward	Standard deviation episodic reward
Current state and residuum ($\underline{s}_k^{0,cl}$)	CPPO	-0.377	0.091
Global space with one future reference ($\underline{s}_k^{11,cl}$)	GPPO1	-0.102	0.089
Residual space with one future reference ($\underline{s}_k^{21,cl}$)	RPPO1	-0.031	0.018
PI controller	PI	-0.069	1.541



(a) Input speeds and reference.



(b) Control inputs.

Figure 7: Performance of the best trained actors on a smooth test reference with offsets.

8. Conclusion

In this work, proximal policy optimization for tracking control exploiting future reference information was presented. We introduced two variants of extending the argument of both actor and critic. In the first variant, we added global future reference values to the argument. In the second variant, the argument was defined in a novel kind of residual space between the current state and the future reference values. By evaluating our approach on a simple drive train model we could clearly show that both variants improve the performance compared to an argument taking only the current reference value into account. If the approach is applied to references with discontinuities, adding several future reference values to the argument is beneficial. The residual space variant shows its advantages especially for references with different offsets. In addition, our approach outperforms PI controllers commonly used in drive train control. Besides higher tracking quality, the generalization to different references is significantly better than using a PI controller. This guarantees an adequate performance on arbitrary, before unseen, references.

In future work, our approach will be applied to a more sophisticated drive train model where noise behavior, model inaccuracies, non-modeled dynamics, and control input range limits, as being expected in a real drive train system, are systematically included in training.

References

[1] Y. Qin, W. Zhang, J. Shi, and J. Liu, "Improve PID Controller Through Reinforcement Learning," in *Proceedings of the 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, Xiamen, China, Aug. 2018.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[3] I. Carlucho, M. De Paula, S. A. Villar, and G. G. Acosta, "Incremental Q-learning Strategy for Adaptive PID Control of Mobile Robots," *Expert Systems with Applications*, vol. 80, pp. 183–199, 2017.

[4] S. Wang, X. Yin, P. Li, M. Zhang, and X. Wang, "Trajectory Tracking Control for Mobile Robots Using Reinforcement Learning and PID," *Iranian Journal of Science and Technology - Transactions of Electrical Engineerin*, vol. 44, pp. 1059–1068, 2020.

[5] S. Kamthe and M. Deisenroth, "Data-efficient Reinforcement Learning with Probabilistic Model Predictive Control," in *Proceedings of the 21th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Playa Blanca, Lanzarote, Canary Islands, April 2018.

[6] S. Gros and M. Zanon, "Data-driven Economic NMPC Using Reinforcement Learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2020.

[7] F. Köpf, J. Westermann, M. Flad, and S. Hohmann, "Adaptive Optimal Control for Reference Tracking Independent of Exo-system Dynamics," *Neurocomputing*, vol. 405, pp. 173–185, 2020.

[8] Y. Hu, W. Wang, H. Liu, and L. Liu, "Reinforcement Learning Tracking Control for Robotic Manipulator with Kernel-Based Dynamic Model," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3570–3578, 2020.

[9] R. Yu, Z. Shi, C. Huang, T. Li, and Q. Ma, “Deep Reinforcement Learning based Optimal Trajectory Tracking Control of Autonomous Underwater Vehicle,” in *Proceedings of the 36th Chinese Control Conference (CCC)*, Da lian, China, July 2017.

[10] D. Kamran, J. Zhu, and M. Lauer, “Learning Path Tracking for Real Car-like Mobile Robots From Simulation,” in *Proceedings of the 2019 European Conference on Mobile Robots (ECMR)*, Prague, Czech Republic, Sept. 2019.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” *arXiv:1707.06347*, 2017.

[12] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional Continuous Control Using Generalized Advantage Estimation,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[13] S. Zhang, Y. Pang, and G. Hu, “Trajectory-tracking Control of Robotic System via Proximal Policy Optimization,” in *Proceedings of the 2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Bangkok, Thailand, Nov. 2019.

[14] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust Region Policy Optimization,” *arXiv:1502.05477*, 2017.

[15] M. Holzleitner, L. Gruber, J. Arjona-Medina, J. Brandstetter, and S. Hochreiter, “Convergence Proof for Actor-Critic Methods Applied to PPO and RUDDER,” *arXiv:2012.01399*, 2020.

[16] R. S. Sutton and A. Barto, *Reinforcement Learning : An Introduction*, 2nd ed. The MIT Press, 2018.

[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous Methods for Deep Reinforcement Learning,” *arXiv:1602.01783*, 2016.

[18] R. J. Williams and J. Peng, “Function Optimization Using Connectionist Reinforcement Learning Algorithms in Simulation of Drive Systems,” *Connection Science*, vol. 3, no. 3, pp. 241–268, 1991.

[19] H. N. Quang, “Dynamic Modeling of Friction Clutches and Application of this Model in Simulation of Drive Systems,” *Periodica Polytechnica, Ser. Mech. Eng.*, vol. 42, no. 2, pp. 169–181, 1998.

[20] L.-J. Lin, “Self-improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching.” *Machine Learning*, vol. 8, pp. 293–321, 1992.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous Control with Deep Reinforcement Learning,” *arXiv:1509.02971*, 2015.