Deep Reinforcement Learning Current Control of Permanent Magnet Synchronous Machines

Tobias Schindler*†, Lara Broghammer†, Petros Karamanakos‡, Armin Dietz†, Ralph Kennel*
*Chair of Electrical Drive Systems and Power Electronics, Technical University of Munich, Germany
†Institute ELSYS, Technische Hochschule Nuremberg, Germany
‡Faculty of Information Technology and Communication Sciences, Tampere University, Finland
Email: tobias.schindler@th-nuernberg.de

***Abstract*—This paper presents a current control approach for permanent magnet synchronous machines (PMSMs) using the deep reinforcement learning algorithm deep deterministic policy gradient (DDPG). The proposed method is designed by examining different training setups regarding the reward function, the observation vector, and the actor neural network. In doing so, the impact of the different design factors on the steady-state and dynamic behavior of the system is assessed, thus facilitating the selection of the setup that results in the most favorable performance. Moreover, to provide the necessary insight into the controller design, the entire path from training the agent in simulation, through testing the control in a controller-in-the-loop (CIL) environment, to deployment on the test bench is described. Subsequently, experimental results are provided, which show the efficacy of the presented algorithm over a wide range of operating points. Finally, in an attempt to promote open science and expedite the use of deep reinforcement learning in power electronic systems, the trained agents, including the CIL model, are rendered openly available and accessible such that reproducibility of the presented approach is possible.**

***Index Terms*—Open science, current control, permanent magnet synchronous machine (PMSM), power electronics, deep reinforcement learning, deep deterministic policy gradient (DDPG)**

I. Introduction

Energy-efficient operation as well as fast dynamic behavior during transients are important aspects of variable speed drives, such as permanent magnet synchronous motor (PMSM) drives. The control method mostly used with PMSM drive systems that enables—at least to some extent—such a desired operation is field-oriented control (FOC). The popularity of FOC stems from the fact that it is designed in a rotating (dq) reference frame that facilitates the adoption of linear controllers, such as simple proportional-integral (PI) controllers, and it thus greatly simplifies the controller design and analysis.

Although FOC has been shown to be reasonably effective, alternatives have been considered in recent years that can further improve the drive system performance. Among those, model predictive control (MPC) has yielded promising results, which attracted the interest of industry [1]. Since MPC is formulated as a (constrained) optimization problem, it allows for design flexibility and can be tailored to the needs and characteristics of the system in question. To do so, the optimization problem underlying MPC is designed such that the control action belongs to either a continuous-control set (i.e., it is a real-valued variable), or a finite-control set (i.e., it is an integer-valued variable).

Another emerging control technique comes from the field of machine learning, with deep reinforcement learning (DRL) particularly receiving a lot of attention. As shown, e.g., in [2], [3], different DRL algorithms have been adopted and refined to address various control problems. While DRL algorithms feature many distinguishable characteristics, when control of power electronic systems is of interest, they can be classified into two categories depending on the nature of the control action. More specifically, similarly to MPC, the control action belongs to either a finite- or a continuous-control set. Finite-control set DRL algorithms, such as the deep Q-network (DQN) algorithm [4], have been employed, e.g., for the direct current control of PMSM drives [5], and fault detection in electrical drives [6].

As for the second group, algorithms with a continuous-control set, such as the deep deterministic policy gradient (DDPG) algorithm [7], have also been successfully applied to the current control of PMSM drives. For example, [8] provides a simulation-based proof of concept, wherein the feasibility of DRL algorithms within the context of motor control is shown. To this aim, an actor-critic algorithm is tested, while different neural network configurations are investigated along with the influence of the discount factor on the control performance. Moreover, a two-step training setup for DDPG-based current control is introduced in [9]. The agent is first pre-trained in simulation and then trained on the test bench by coupling the real-time controller to a workstation. Additionally, [9] shows measurements of the trained controller at a fixed speed, compares performance to state-of-the-art control algorithms, and discusses the used hyperparameters. Finally, [10] uses cascaded simulation-trained DDPG agents for current and speed control, the performance of which is tested in a software-in-the-loop environment.

In this paper, a current control algorithm based on DDPG—referred to as DDPG-CC in the sequel—is developed for PMSM drives, as depicted in Fig. 1. The concept of this method, which is based on approaches discussed in [8]–[10], is summarized first. Subsequently, the entire design and testing

Funded by Bundesministerium für Bildung und Forschung (BMBF, German Federal Ministry of Education and Research) - KI-Power (16ME0146K).

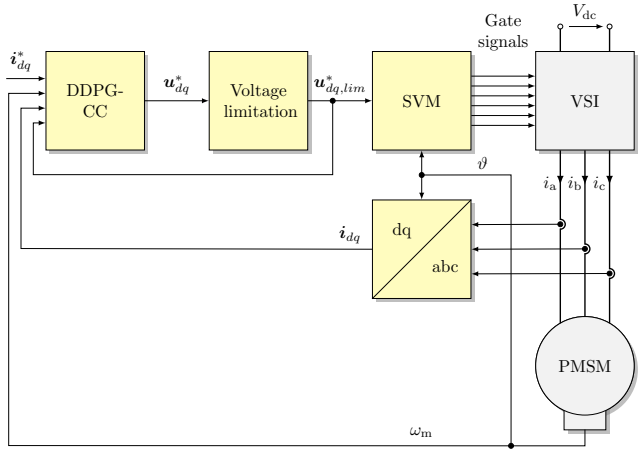


Fig. 1: Block diagram of the proposed DDPG-CC method.

chain are introduced. Specifically, the training of the agent in simulation is discussed, and the relevant trained DDPG-CC agents are presented. Following, the controller testing in a controller-in-the-loop (CIL) environment based on [11] is described. To this end, the DDPG-CC method is implemented on the real-time open-source control platform UltraZohm [12]. In the last step of the development chain, the control performance of the proposed control algorithm is assessed in a real-world setting, and the test bench measurements are compared with the corresponding simulation and CIL results. Finally, in addition to the aforementioned implementation, the use of an open-source platform in combination with CIL is explored, which facilitates the reproduction of the results by other researchers, thus supporting open science.

Regarding the last point, this paper aims to promote open science not only by means of the employed open-source framework but by essentially introducing a methodology that enables the reproduction of RL-based control methods for electric drives. To this aim, the presented DDPG-CC is investigated over a wide range of operating points and possible configurations are explored. A combination of random search and fixing hyperparameters based on the existing literature is applied to train suitable agents for each configuration. Hence, this systematic approach aspires to facilitate the comparison of different configurations and to provide insights into which design choices achieve favorable performance.

II. DDPG-CC Algorithm for PMSM Drive Systems

The proposed current control algorithm is designed in an orthogonal reference (dq) frame that rotates with the synchronous angular speed. It is developed for a PMSM drive system consisting of a two-level voltage source inverter (VSI) and a PMSM, as shown in Fig. 1. The discrete-time DDPG agent controls the plant based on measurements and derived quantities such that the absolute current reference tracking error on the d- and q-axis, defined as

$$e_d(k) = i_d(k)^* - i_d(k)\,, \tag{1a}$$
$$e_q(k) = i_q(k)^* - i_q(k)\,, \tag{1b}$$

is minimized. To this end, the controller manipulates the voltage reference, i.e., the control action of the DDPG-CC is chosen to be $\boldsymbol{a}(k) = \begin{bmatrix} v_d^*(k) & v_q^*(k) \end{bmatrix}^\top$. The specific configuration of DDPG-CC offers multiple degrees of freedom, such as the choice of observations, reward function, hyperparameters, and training procedure. A subset of the possible configurations is discussed in the following.

A. Model of the PMSM Drive System

In this paper, all DDPG-CC configurations are trained using the same simplified PMSM model described by

$$\frac{\mathrm{d}\psi_d}{\mathrm{d}t} = v_d(t) - R_1 i_d(t) + \omega_{\mathrm{el}}\psi_q(t) \tag{2a}$$
$$\frac{\mathrm{d}\psi_q}{\mathrm{d}t} = v_q(t) - R_1 i_q(t) - \omega_{\mathrm{el}}\psi_d(t)\,, \tag{2b}$$
$$\psi_d(t) = \psi_{\mathrm{PM}} + L_d i_d(t)\,, \tag{2c}$$
$$\psi_q(t) = L_q i_q(t) \tag{2d}$$

with the stator resistance R_1, the inductances L_d and L_q, the flux-linkage of the stator $\boldsymbol{\psi}_{dq}$ and of the permanent magnets ψ_{PM} as well as the stator voltage $\boldsymbol{v}_{dq}$ and stator current $\boldsymbol{i}_{dq}$ in the dq-plane. The electrical speed $\omega_{\mathrm{el}} = \omega_{\mathrm{m}} p$ is given by the rotational speed ω_{m} and the number of pole pairs p. The inverter of the drive system is fed by a constant dc-link voltage V_{dc}, which imposes limitations on the available voltage on the d- and q-axis. To ensure operation in the extended linear modulation region, i.e., modulation index $m \leq 2/\sqrt{3}$, the voltage reference given by the action $\boldsymbol{a}(k)$ is limited to $\boldsymbol{v}^*_{dq,\mathrm{lim}}(k)$ according to [13] and fed into a space vector modulation (SVM) stage. This implies that the amplitude of the applied voltage is bounded by $\|\boldsymbol{v}^*_{dq}\|_2 \leq V_{\max}$ with $V_{\max} = V_{\mathrm{dc}}/\sqrt{3}$. The environment for the training of the DDPG-CC is formed by the aforementioned plant model (2) in the dq-plane combined with the limitation of the voltage output.

B. Control Algorithm Configurations

DDPG-CC determines the action $\boldsymbol{a}(k)$ based on the observation vector $\boldsymbol{o}(k)$, which consists of measurements of the system state as well as derived quantities. The inputs of the observation are normalized to their respective rated values (e.g., I_r, $\omega_{\mathrm{m,r}}$), while the voltage $\boldsymbol{v}^*_{dq,\mathrm{lim}}$ is normalized based on $V_{\max}$.

Two variations of the observation are investigated. The first version of the observation vector

$$\boldsymbol{o}_1(k) = \begin{bmatrix} e_d(k) \\ e_q(k) \\ \int e_d(k) \cdot f_c \\ \int e_q(k) \cdot f_c \\ i_d(k) \\ i_q(k) \\ v^*_{d,\mathrm{lim}}(k-1) \\ v^*_{q,\mathrm{lim}}(k-1) \\ \omega_{\mathrm{m}}(k) \end{bmatrix} \tag{3}$$

features the integral of the current tracking error on the d- and q-axis, which is implemented in the discrete-time domain using the forward Euler method. The integration is stopped if the voltage vector limitation is active, i.e., $\boldsymbol{v}_{dq}^* \neq \boldsymbol{v}_{dq,\mathrm{lim}}^*$ (clamping). Furthermore, the result of the integration is scaled by the sampling frequency f_c of the DDPG-CC. In contrast to this, the alternative observation vector

$$\boldsymbol{o}_2(k) = \begin{bmatrix} e_d(k) \\ e_q(k) \\ i_d(k) \\ i_q(k) \\ v_{d,\mathrm{lim}}^*(k-1) \\ v_{q,\mathrm{lim}}^*(k-1) \\ \omega_\mathrm{m}(k) \end{bmatrix} \tag{4}$$

does not utilize the aforementioned integral of the tracking error; otherwise, it replicates (3).

The developed DDPG-CC method aims to track the stator current along its reference. During training, the agent aims to maximize a given reward function $r(k)$. Similarly to the observation, two variations of the reward function are examined, which are designed in line with the different objective function formulations in MPC [1]. Specifically, the reference tracking error in the reward function is based on either the ℓ_1-norm, resulting in the function

$$r_1(k) = \begin{cases} -\|\boldsymbol{e}_{dq}(k)\|_1, & \text{for} \quad i_1(k) \leq I_\mathrm{max} \\ -\|\boldsymbol{e}_{dq}(k)\|_1 - i_1(k), & \text{for} \quad i_1(k) > I_\mathrm{max} \end{cases}, \tag{5}$$

or the squared ℓ_2-norm, giving rise to the reward function

$$r_2(k) = \begin{cases} -\|\boldsymbol{e}_{dq}(k)\|_2^2, & \text{for} \quad i_1(k) \leq I_\mathrm{max} \\ -\|\boldsymbol{e}_{dq}(k)\|_2^2 - i_1(k), & \text{for} \quad i_1(k) > I_\mathrm{max} \end{cases}. \tag{6}$$

Regardless of the employed norm, however, the amplitude of the phase current $i_1(k) = \|\boldsymbol{i}_{dq}(k)\|_2$ is not allowed to exceed the maximum allowed current of the machine I_max. To ensure this, the agent receives a linearly increasing penalty in (5) and (6) to prevent damage to the system.

DDPG requires four neural networks: one for the actor, one for the critic, and a target network for each [7]. However, only the actor neural network is required for control after the training. Therefore, even though the number of hidden layers as well as of neurons of the critic and actor affects the computational burden during training, the neural network of the actor is the one that has a direct impact on the real-time requirements of the algorithm. To elucidate this point, the four different DDPG-CC configurations resulting by combining the different reward functions (i.e., (5) or (6)) with the possible observation vectors (i.e., (3) or (4)) are examined in Table I along with three different actor networks.

C. Training

The training is conducted using the Matlab 2022b Reinforcement Learning Toolbox [14]. The plant model is implemented in Simulink based on (2), and the duration of one training episode is set to $\tau_q = L_q/R_\mathrm{s}$. At the beginning of each episode, random values for the set-points i_d^* and i_q^* as well as for the rotational speed ω_m are set within the safe operating limits of the machine and held constant for the entire episode. An ideal plant model is used during training, i.e., no parameter variations of the machine are considered. The voltage vector limitation is implemented in the training environment, while the SVM and the VSI are neglected. The training is performed on three workstations with AMD Ryzen Threadripper Pro 3995WX (2x) and 3975WX (1x) CPUs. Training 512 agents for 500,000 samples per agent in parallel takes approximately 10 h.

TABLE I: Trained DDPG-CC configurations with different actor neural networks, reward functions, and observations.

Configuration	Layer	Neurons	Reward	Observation
1.1	1	64	r_1	$\boldsymbol{o}_1$
1.2	3	64	r_1	$\boldsymbol{o}_1$
1.3	1	128	r_1	$\boldsymbol{o}_1$
2.1	1	64	r_1	$\boldsymbol{o}_2$
2.2	3	64	r_1	$\boldsymbol{o}_2$
2.3	1	128	r_1	$\boldsymbol{o}_2$
3.1	1	64	r_2	$\boldsymbol{o}_1$
3.2	3	64	r_2	$\boldsymbol{o}_1$
3.3	1	128	r_2	$\boldsymbol{o}_1$
4.1	1	64	r_2	$\boldsymbol{o}_2$
4.2	3	64	r_2	$\boldsymbol{o}_2$
4.3	1	128	r_2	$\boldsymbol{o}_2$

D. Hyperparameters

Most hyperparameters of the DDPG agent are fixed to values derived from the DRL literature, see Table II. A random search over the hyperparameters critic learn rate, actor learn rate, exploration standard deviation, and exploration decay is performed. To this end, a random sample of 512 agents for each configuration listed in Table I is drawn from the defined ranges of the hyperparameters and trained according to the method outlined in this section. Exploration of DDPG is accomplished by adding noise according to the Ornstein-Uhlenbeck (OU) process to the output action of the agent during training [7]. The search space for the standard deviation of the OU noise is based on the bounds of $\boldsymbol{v}_{dq}^*$, while the half-life of the exploration noise decay is based on the total number of samples. Note that a limitation to comparing hyperparameters is their dependency on the implementation details. For example, the Matlab implementation of the OU exploration noise does not reset after each episode as opposed to the original DDPG description in [7].

III. Implementation

Before testing the performance of the proposed DDPG-CC method—initially in a CIL environment and subsequently in a real-world setting—implementation details are provided. As discussed before, the agent is solely trained in simulation. Thus, only the actor, consisting of the actor neural network, the observation vector including normalization, and the output scaling, is required for real-time operation.

TABLE II: Parameters of used hyperparameters and comparison to literature.

Parameter	**Value**	[7]	[15]	[16]	[9]	[17]
Minibatch size	64	64	64	64	128	16
Experience buffer length	500,000	1,000,000	5,000	-	5,000	5,000
Training samples	500,000	2,500,000	120,000	-	250,000	1,800,000
L2 regularization actor & critic	0.01	$1 \cdot 10^{-2}$	-	-	$1 \cdot 10^{-2}$	-
Target network update frequency actor & critic	1	1	1,000	-	1,000	-
Target smooth factor	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-2}$	-	-	-
Discount factor	0.9	0.99	0.9	0.9	0.9	0.9
Learn rate critic	$1 \cdot 10^{-6}$ - $1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Learn rate actor	$1 \cdot 10^{-6}$ - $1 \cdot 10^{-3}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$1 \cdot 10^{-4}$
Exploration standard deviation	$1\ \% - 10\ \% \cdot V_{\max}$	0.2	0.3	1.5	0.2	-
Exploration decay	$0.1\ \% - 20\ \%$ of samples	0.15	-	$1 \cdot 10^{-5}$	-	-
Actor hidden layer	1 or 3	2	2	1	1	3
Actor neurons	64 or 128	400/300	300/600	25	100	200/200/10
Critic hidden layer	3	2	2	2	3	2
Critic neurons	128/128/128	400/300	200/200	50/25	75	200/10
Critic hidden layer activation function	ReLu	ReLu	ReLu	ReLu	leaky ReLu	ReLu
Actor hidden layer activation function	ReLu	ReLu	ReLu	ReLu	leaky ReLu	ReLu

A. UltraZohm

The UltraZohm is an open-source rapid control prototyping (RCP) system, presented in detail in [12]. It is built around the Xilinx Zynq UltraScale+ MPSoC, which offers four ARM-A53 processors (APU), two ARM-R5 processors (RPU), and programmable logic (PL). The actor of the DDPG-CC is implemented in the RPU of the UltraZohm. The configurations 1.2, 2.2, 3.2, and 4.2 use a neural network with three hidden layers for the actor, which is implemented in the PL as described in [18] to keep the calculation time within the real-time limits.

B. CIL Environment

RCP systems reduce the implementation effort of complex control algorithms, and thus facilitate the pivotal—and often error-prone—step of testing on the target platform. This step is particularly important as the correctness and real-time capability of the implementation must be ensured. In this direction, an intermediate CIL step on the target improves testing before controlling the real plant [11]. Testing the controller in the CIL environment can provide sufficient test coverage, which is hard to achieve without the CIL step, as testing then only relies on comparisons of the control behavior to pre-computed data (e.g., based on offline simulations). Moreover, by testing the algorithm in a CIL environment over the full operating range, its evaluation in a real-world setting is greatly simplified, as the controller can be used without any implementation changes. This procedure is crucial for DRL-based control approaches as, unlike conventional controllers, there are no obvious simplifications for performing plausibility checks, such as using slow controller parameters during the first commissioning.

Given the above, a plant model is implemented in the PL of the UltraZohm, and the DDPG-CC is implemented in the PS and PL in line with [11]. The plant model is equivalent to the model based on (2) used for training. Thus, a correct real-time implementation should give matching results between simulation and CIL.

C. Reproducibility and Open Science

Reproducing and, in turn, extending the results of other researchers is a vital process in the scientific community. However, this process can be cumbersome for motor control applications. First, implementation details of algorithms may exist that are not included in publications due to space limitations, unconscious implementation decisions, or the fact that releasing source code is not a common practice in the field. Second, suitable measurement equipment, control hardware, power electronics, electrical machines, and several peripherals are required. In addition to the discussed practical benefits of using a CIL approach, it also serves as an enabler for open science. There are fewer equipment requirements for conducting CIL tests compared to full test bench setups, lowering the associated costs and enabling the reproduction of research results.

In accordance with the outlined ideas, the implementation of the trained DDPG-CC agents, the CIL test, and the FOC benchmark are released within the UltraZohm open-source project. Thus, any researcher with access to a Xilinx UltraScale+ MPSoC device, e.g., a development kit, can reproduce the results with manageable effort and without requiring the used PMSM and a compatible inverter. Hence, it can be claimed that this approach simplifies the reproduction of research results and reduces the required resources to do so. However, it is important to note that such an approach does not imply that independent implementation and reproduction of research results in the laboratory should be overlooked; on the contrary, ideally, this should be the ultimate goal when assessing a control algorithm.

IV. Performance Assessment

In this section, the performance of the developed control strategy is assessed. The parameters of the used PMSM are listed in Table III. The evaluation of the agents for each configuration is done in multiple steps. First, the best agent of each configuration is determined by simulation. Second, the best agent of each configuration is tested in CIL to ensure

TABLE III: Parameters of the used PMSM (Heidrive HMD06-005), inverter, and controller.

Parameter	Symbol	Value
d-axis inductance	L_d	1.13 mH
q-axis inductance	L_q	1.42 mH
Resistance	R_s	543 mΩ
PM flux linkage	ψ_{PM}	16.9 mVs
Pole pairs	p	3
Rated dc-link	$V_{\mathrm{dc},r}$	48 V
Rated current	I_r	4.2 A
Maximum current	$I_{\max}$	10.8 A
Rated speed	n_r	3.000 min^{-1}
Rated torque	T_r	0.48 N m
Rated power	P_r	150 W
Control frequency	f_c	10 kHz
Switching frequency	f_{PWM}	10 kHz

TABLE IV: Simulation results of the best agents for each configuration.

Config.	$\bar{Q}_{\mathrm{ITAE}}$	$\bar{t}_{\mathrm{r}}$	$\bar{t}_{\mathrm{set}}$	$\bar{\mathrm{PO}}$	$\bar{e}(\infty)$
FOC	16.8 ms	0.28 ms	13.9 ms	15 %	$<$ 1 mA
1.1	20.0 ms	0.15 ms	15.1 ms	24 %	$<$ 1 mA
1.2	19.6 ms	0.24 ms	14.7 ms	26 %	$<$ 1 mA
1.3	18.6 ms	0.25 ms	14.6 ms	21 %	$<$ 1 mA
2.1	202.4 ms	0.12 ms	13.5 ms	15 %	52 mA
2.2	191.4 ms	0.09 ms	13.5 ms	83 %	49 mA
2.3	196.6 ms	0.09 ms	13.5 ms	29 %	50 mA
3.1	21.4 ms	0.14 ms	15.1 ms	58 %	$<$ 1 mA
3.2	25.4 ms	0.40 ms	15.5 ms	52 %	$<$ 1 mA
3.3	21.1 ms	0.11 ms	15.0 ms	73 %	$<$ 1 mA
4.1	237.9 ms	0.08 ms	14.1 ms	292 %	62 mA
4.2	241.4 ms	0.12 ms	13.6 ms	107 %	63 mA
4.3	225.5 ms	0.13 ms	13.6 ms	37 %	59 mA

safe operation before, lastly, the suitable agents are deployed to the test bench and evaluated.

A. Simulation

Each trained agent is simulated with the same set of set-point changes for $\boldsymbol{i}_{dq}^*$ at fixed rotational speeds $n \in \{0, 500, 1000, 2000, 3000\}\,\mathrm{min}^{-1}$. The unit jump and ten arbitrary set-point changes are simulated for each rotational speed with the simulation time $t_{\mathrm{sim}} = 15\tau_q$. To extract the agent with the best performance over a wide operating range, the ITAE criteria of the d- and q-axis current is calculated by

$$Q_{\mathrm{ITAE}} = \frac{1}{2}\left(\int_0^{t_{\mathrm{sim}}} |e_d(t)| \cdot t \,\mathrm{d}t + \int_0^{t_{\mathrm{sim}}} |e_q(t)| \cdot t \,\mathrm{d}t \right). \quad (7)$$

The arithmetic mean of (7) over all simulated speeds yields the mean performance metric $\bar{Q}_{\mathrm{ITAE}}$. In the same manner, the rise time $\bar{t}_{\mathrm{r}}$, settling time $\bar{t}_{\mathrm{set}}$, percentage overshoot ($\bar{\mathrm{PO}}$), and steady-state error $\bar{e}(\infty)$ are calculated.

Table IV lists the best-performing agents for each configuration according to $\bar{Q}_{\mathrm{ITAE}}$. The simulation results show that agents using the observation (3) perform better in terms of $\bar{Q}_{\mathrm{ITAE}}$ compared with those based on (4). Furthermore, all agents that use (4) exhibit a noticeable steady-state error. As for the reward functions, there are no significant differences between the agents trained with the two functions (see (5) and (6)), albeit the agents yield different trade-offs with respect to the chosen performance metrics. Finally, for benchmark purposes, the performance of FOC is also shown in Table IV.[1]

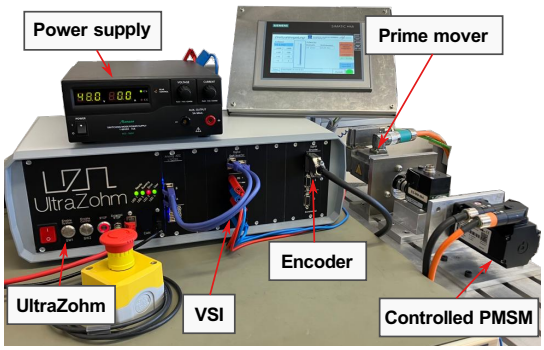


Fig. 2: Test bench setup.

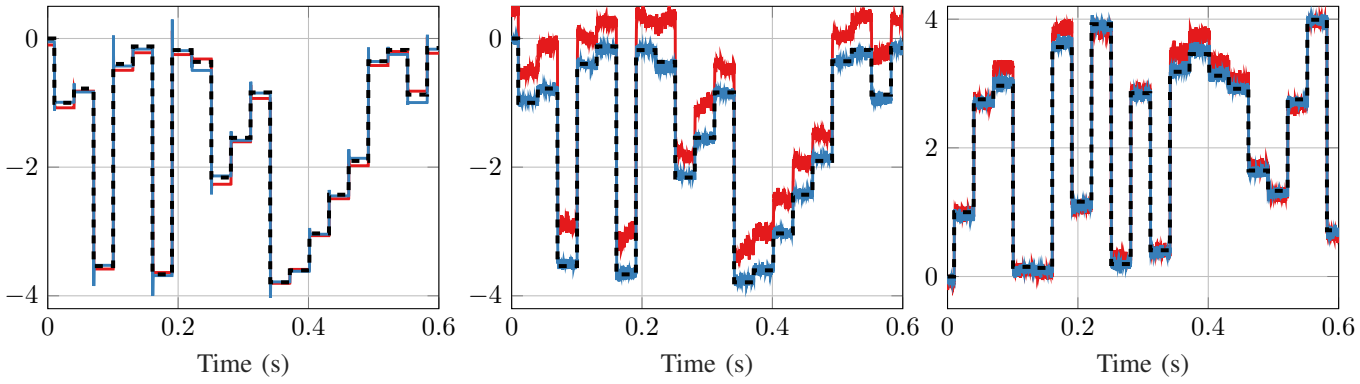
B. CIL and Test Bench Measurements

Experimental studies are carried out to investigate the performance of the developed DDPG-CC method while the PMSM is coupled to a speed-controlled prime mover. Fig. 2 shows the laboratory setup used for the experimental tests. The validation profile of Section IV-A with arbitrary set-point changes for $\boldsymbol{i}_{dq}^*$ is extended by nine additional set-points to assess the performance of the trained agents in the CIL environment and on the test bench.

Fig. 3 shows the CIL and measurement results of the best agent in configuration 2.1. The agent shows a steady-state error, as indicated in Table IV, which increases with increasing rotational speed. This error appears in both the CIL and experimental results. Nevertheless, the steady-state error in the d-axis is distinctly bigger in the experimental results compared to the CIL ones. This indicates that due to the lack of an integrating element in the examined configuration, any minor parameter mismatch, effects that are not modeled (e.g., iron loss resistance), and slight misalignment of the dq-frame between the training model and real machine cannot be compensated. For example, when the operating point depicted in Fig. 3b ($n = 3000\,\mathrm{min}^{-1}$) is concerned, FOC applies $u_d \approx -0.4\,\mathrm{V}$ to counteract the aforementioned effects. However, agents using observation (4) cannot compensate for these effects in the conducted tests.

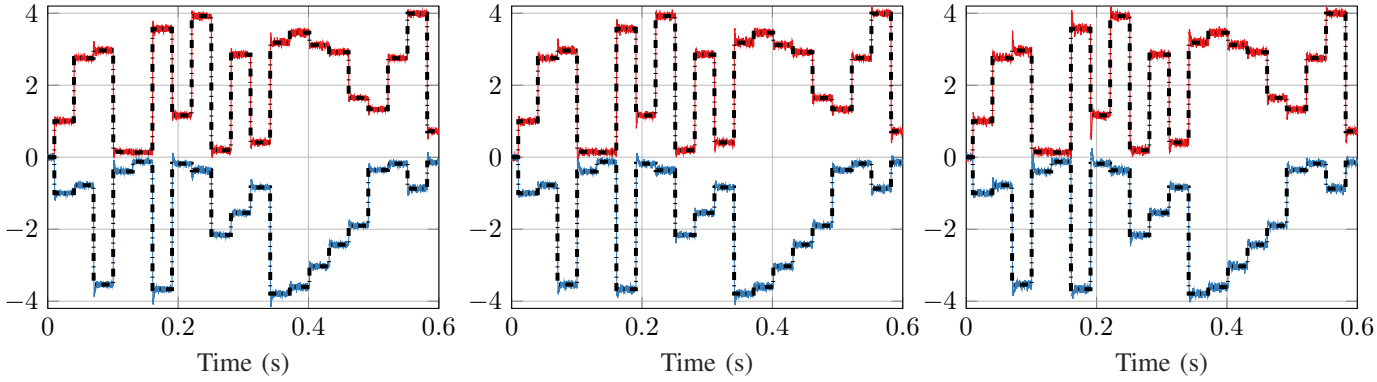
On the other hand, no steady-state error is expected for agents that use (3), as this observation introduces an integrating element. This is experimentally confirmed at different speeds, see, e.g., the best agent in configuration 1.1 that is depicted in Fig. 4. The best agents of configurations 1.1, 1.2, 1.3, 3.1, 3.2, and 3.3 track the validation profile with varying performance. Table V lists Q_{ITAE} for all measured speeds as well as the arithmetic mean over all operating points $\bar{Q}_{\mathrm{ITAE}}$. As can be seen, configurations 1.1 and 3.1 have comparable performance with respect to $\bar{Q}_{\mathrm{ITAE}}$. However, this is not the case for all operating points.

[1]The developed FOC is tuned according to the modulus optimum method, including a decoupling network, while all small time constants of the system are lumped into the time constant $\tau_\sigma = 1.5 \cdot f_{\mathrm{C}}^{-1}$.



(a) Stator current i_d in A (CIL). (b) Stator current i_d in A (experiment). (c) Stator current i_q in A (experiment).

Fig. 3: Validation profile with current reference of best agent in configuration 2.1 at $n = 3000\,\mathrm{min}^{-1}$ (red) and $n = 500\,\mathrm{min}^{-1}$ (blue).



(a) $n = 1000\,\mathrm{min}^{-1}$. (b) $n = 2000\,\mathrm{min}^{-1}$. (c) $n = 3000\,\mathrm{min}^{-1}$.

Fig. 4: Validation profile of best agent in configuration 1.1 with stator current i_q (red) and i_q (blue) in A.

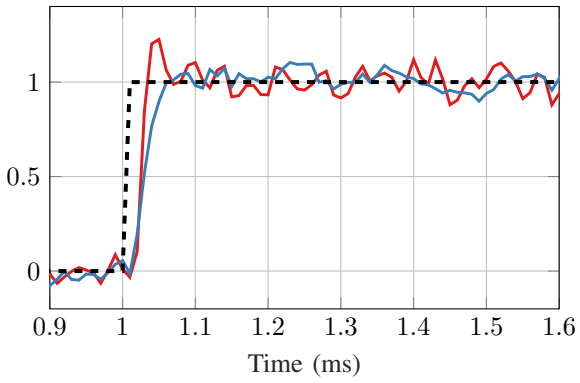


Fig. 5: Comparison of a step-up set-point change of current i_q in A between the best agent in configuration 1.3 (red) and FOC (blue) at $n = 2000\,\mathrm{min}^{-1}$.

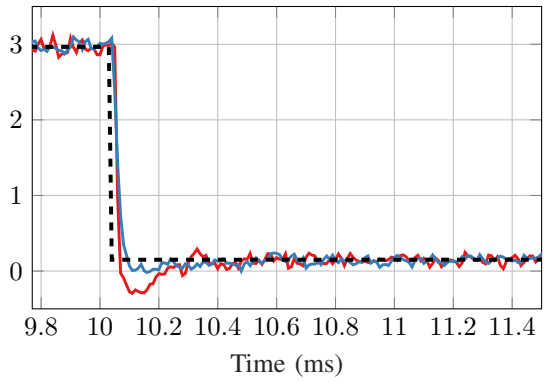


Fig. 6: Comparison of a step-down set-point change of current i_q in A between the best agent in configuration 1.3 (red) and FOC (blue) at $n = 2000\,\mathrm{min}^{-1}$.

Finally, the transient performance of the best agent in configuration 1.3 is tested and compared with that of FOC, see Figs. 5 and 6. With regards to the step-up change in the current set-point in the q-axis (Fig. 5), the developed DDPG-CC method achieves better performance compared with FOC, even though the agent does not outperform the reference FOC with respect to $\bar{Q}_{\mathrm{ITAE}}$. On the other hand, for the step-down i_q set-point change in Fig. 6, FOC is able to regulate i_q along

TABLE V: Q_{ITAE} of the measured validation profile at different rotational speeds.

	FOC	1.1	1.2	1.3	3.1	3.2	3.3
$500\,\mathrm{min}^{-1}$	83 s	86 s	93 s	99 s	84 s	132 s	118 s
$1000\,\mathrm{min}^{-}$	89 s	89 s	98 s	101 s	85 s	130 s	117 s
$2000\,\mathrm{min}^{-1}$	101 s	99 s	115 s	114 s	98 s	138 s	120 s
$3000\,\mathrm{min}^{-}$	115 s	110 s	132 s	141 s	116 s	140 s	131 s
Q_{ITAE}	97 s	96 s	110 s	114 s	96 s	135 s	122 s

its reference with shorter rise time and less overshoot.

V. Conclusion

This paper investigated a DRL-based current control scheme for PMSM drives. To this aim, different setups regarding the reward function, observation vectors, and the actor neural network were first examined. As shown, the use of the integrated error in the observation vector achieves a tracking error-free behavior. At the same time, it equips the trained agent with the tools required to compensate for unmodeled dynamics and model mismatches, thus rendering these agents suitable for a wide range of operating points. In contrast, designing the reward function based on the squared ℓ_2-norm or the ℓ_1-norm does not seem to have a significant impact on the actor. Likewise, utilizing a larger neural network for the actor does not lead to clearly improved behavior.

Overall, the presented method for training, implementation, CIL testing, and real-world experiments highlighted the possibility of applying DRL to control problems in the area of variable speed drives. This was verified with the presented results, which demonstrated the efficacy of the presented algorithm over a wide range of operating points as well as the possibility of using agents trained exclusively in simulation for control in a real-world setting. Finally, this work opted to promote open science by employing an open-source prototyping system in combination with a CIL-based verification approach. In doing so, obstacles in reproducing research results can be potentially bypassed with less effort and hardware requirements.

For future research, it can be investigated how to improve the control performance over the whole operating range. Reward shaping is one possibility to define the requirements of the controller more precisely, and it can potentially allow for the inclusion of secondary control goals. Moreover, dropping the limitation of a fixed neural network architecture of the critic for the experiments—as it was done in this work—might lead to improved agent performance. To this end, recurrent neural networks could be used.

Acknowledgment

The authors would like to thank Dennis Hufnagel, the main author of the open-source FOC implementation which is used as a reference in this paper, for his support.

References

[1] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel, "Model predictive control of power electronic systems: methods, results, and challenges," *IEEE Open Journal of Industry Applications*, vol. 1, pp. 95–114, 2020.

[2] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," 2018. [Online]. Available: https://arxiv.org/abs/1808.00177

[3] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. Riedmiller, "Magnetic control of tokamak plasmas through deep reinforcement learning," pp. 414–419, 2022. [Online]. Available: https://doi.org/10.1038/s41586-021-04301-9

[4] V. Mnih, K. Kavukcuoglu, and D. e. a. Silver, "Human-level control through deep reinforcement learning," *Nature*, 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[5] M. Schenke and O. Wallscheid, "A deep Q-learning direct torque controller for permanent magnet synchronous motors," *IEEE Open Journal of the Industrial Electronics Society*, vol. 2, pp. 388–400, 2021.

[6] S. Attestog, J. S. L. Senanayaka, H. V. Khang, and K. G. Robbersmyr, "Robust active learning multiple fault diagnosis of PMSM drives with sensorless control under dynamic operations and imbalanced datasets," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2022.

[7] 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." [Online]. Available: https://arxiv.org/pdf/1509.02971.pdf

[8] M. Schenke, W. Kirchgässner, and O. Wallscheid, "Controller design for electrical drives by deep reinforcement learning: a proof of concept," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4650–4658, 2019.

[9] G. Book, A. Traue, P. Balakrishna, A. Brosch, M. Schenke, S. Hanke, W. Kirchgässner, and O. Wallscheid, "Transferring online reinforcement learning for electric motor control from simulation to real-world experiments," *IEEE Open Journal of Power Electronics*, vol. 2, pp. 187–201, 2021.

[10] S. Bhattacharjee, S. Halder, Y. Yan, A. Balamurali, L. V. Iyer, and N. C. Kar, "Real-time SIL validation of a novel PMSM control based on deep deterministic policy gradient scheme for electrified vehicles," *IEEE Transactions on Power Electronics*, vol. 37, no. 8, pp. 9000–9011, 2022.

[11] E. Liegmann, T. Schindler, P. Karamanakos, A. Dietz, and R. Kennel, "Ultrazohm—an open-source rapid control prototyping platform for power electronic systems," in *2021 International Aegean Conference on Electrical Machines and Power Electronics (ACEMP) & 2021 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, 2021, pp. 445–450.

[12] S. Wendel, A. Geiger, E. Liegmann, D. Arancibia, E. Durán, T. Kreppel, F. Rojas, F. Popp-Nowak, M. Diaz, A. Dietz, R. Kennel, and B. Wagner, "UltraZohm— a powerful real-time computation platform for MPC and multi-level inverters," Quanzhou, China, 2019, pp. 1–6.

[13] P. Q. Nguyen and D. Jörg-Andreas, *Vector control of three-phase AC Machines*. Springer, 2015.

[14] "Matlab documentation: deep deterministic policy gradient (DDPG) agents," https://de.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html, accessed: 20220-12-08.

[15] S. Guo, X. Zhang, Y. Zheng, and Y. Du, "An autonomous path planning model for unmanned ships based on deep reinforcement learning," *Sensors*, 2020.

[16] R. Siraskar, "Reinforcement learning for control of valves," *Machine Learning with Applications*, vol. 4, p. 100030, 2021.

[17] Y. Liu, Z. Jiang, S. Zhang, and S. Xu, "Deep reinforcement learning-based beam tracking for low-latency services in vehicular networks," 2020.

[18] T. Schindler and A. Dietz, "Real-time inference of neural networks on FPGAs for motor control applications," in *2020 10th International Electric Drives Production Conference (EDPC)*, 2020, pp. 1–6.