Meta-Reinforcement-Learning-Based Current Control of Permanent Magnet Synchronous Motor Drives for a Wide Range of Power Classes

Darius Jakobeit, Maximilian Schenke, and Oliver Wallscheid, *Member, IEEE*

***Abstract*—Data-driven reinforcement-learning-based controller schemes have much potential to aid the design of model-free control algorithms that can be trained without the necessity of plant-specific parameter knowledge. Unfortunately, the corresponding training phase is a time-consuming and possibly money-consuming process, which needs to be repeated whenever application to a new plant system is requested. To reduce the total training time for a large set of heterogeneous plant systems, this article proposes a meta-reinforcement-learning-based approach that is to be utilized for the control of hundreds of different permanent magnet synchronous motor drives ranging from a few watts to hundreds of kilowatts. The so-called context variables carry the meta-information about the set of considered drive systems. Their estimation using a corresponding artificial neural network as context approximator is a core aspect of this article. The context information allows the reinforcement-learning-based control algorithm to automatically adapt itself to individual motor drives without requiring individual plant training. Since the found context variables can also be interpreted as an implicit system identification result, they allow us to determine irregular plant behavior (e.g., faulty drives) as an added bonus of the proposed meta-reinforcement learning scheme. Empirical results during this proof of concept successfully validate the potential of the proposed approach to drastically reduce the total training time and encourage further research.**

***Index Terms*—Artifical neural networks, control, meta-reinforcement learning (MRL), permanent magnet synchronous motors (PMSMs), system identification.**

I. Introduction

FROM the prominent role of electric drives in industrial applications and their rising importance for automotive applications originates a comprehensive but also intricate control theory. The set of model-driven control approaches is well studied, spanning from linear field-oriented controllers [1] over direct torque controllers [2] to model-predictive control patterns [3], to just name a few cornerstones.

Manuscript received 10 October 2022; revised 16 December 2022 and 30 January 2023; accepted 9 March 2023. Date of publication 13 March 2023; date of current version 19 May 2023. This work was supported in part by the German Research Foundation (DFG) under Grant 459524199 and in part by computing time provided by the Paderborn Center for Parallel Computing (PC2). Recommended for publication by Associate Editor K.-B. Lee. *(Corresponding author: Maximilian Schenke.)*

The authors are with the Department of Power Electronics and Electrical Drives, Paderborn University, 33098 Paderborn, Germany (e-mail: jakobeit@mail.upb.de; schenke@lea.uni-paderborn.de; wallscheid@lea.upb.de).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPEL.2023.3256424.

Digital Object Identifier 10.1109/TPEL.2023.3256424

While these already established approaches are highly efficient in scenarios where accurately parameterized drive models are available, they tend to lack performance whenever precise system knowledge is not available. Moreover, those model-based controller designs require a lot of human expert knowledge as well as manual tuning effort [4], both of which are not available in abundance (especially in the industry). In the recent past, electric drives [5], [6], [7] and further power electronics (in particular DC–DC converters) [8], [9], [10], [11] control setups on the basis of reinforcement learning (RL) have been proposed to deal with such situations by providing a data-driven fully automatable control scheme with appealing potentials.

1) Knowledge about the plant model is not required as optimal control actions are learned through the direct interaction of the RL algorithm and the plant system [12].
2) Challenging higher order and parasitic effects, e.g., iron losses, magnetic (cross-)saturation, or the inverter nonlinearity, do not need to be characterized in beforehand as their effects are directly learned within the control policy from real-world measurements [6].
3) Different goals can be incorporated into the learning problem, allowing multiobjective control with only one learning phase [7].

These advantages of data-driven control are opposed by a time-consuming learning phase during which the plant system is unavailable for its intended application, and the computationally demanding learning algorithms must be run on costly computer hardware. Naturally, it is, therefore, of great interest to reduce the time spent with training by finding a more universal approach to RL-based drive control that, instead of being optimized to run on only one specific drive setup, is able to adapt itself to a large set of different plant systems as a part of its objective, which is the core problem addressed in the following.

A. Contribution

This article, therefore, proposes a meta-RL (MRL) current control algorithm for permanent magnet synchronous motors (PMSMs), which is equipped to adapt to a given motor system [13], [14]. After the learning phase, this capability enables the MRL algorithm to be operated on previously unseen drives of different parameterization without necessitating new learning effort and, hence, introducing several striking advantages.

1) The learning phase is conducted only once and expanded to a wide variety of drive systems. This reduces the total training time for a large set of different drives drastically.
2) Control performance is very robust concerning drive variations.
3) The underlying context variable identification scheme, although only implicit (physical motor parameters are not required and also not identified), gives information about pathological system behavior, which could indicate irregular drive conditions such as faults.

In order to advance toward these benefits, this article presents a proof of concept on the preparation, conduction, and validation of the MRL for PMSM control.[1]

II. Drive System Model

In the following, a brief overview of the PMSM drive system model is provided for the sake of completeness. It is used as part of the RL-oriented training and testing open-source software gym-electric-motor (GEM)[2] [15]. However, it should be noted that the considered RL-based control algorithms do not have access to the drive model.

The PMSM is a three-phase electric motor with a large variety of application scenarios, ranging from industrial automation to electric traction [16]. In the domain of drive control, three-phase drives are concisely modeled via field-oriented coordinates. Utilizing this two-phase representation, the dynamic motor behavior can be described by a system of first-order ordinary differential equations (ODEs):

$$\begin{aligned}\frac{\mathrm{d}i_\mathrm{d}(t)}{\mathrm{d}t} &= \frac{u_\mathrm{d}(t) + p\omega_\mathrm{me}(t)L_\mathrm{q}i_\mathrm{q}(t) - R_\mathrm{s}i_\mathrm{d}(t)}{L_\mathrm{d}}\\ \frac{\mathrm{d}i_\mathrm{q}(t)}{\mathrm{d}t} &= \frac{u_\mathrm{q}(t) - p\omega_\mathrm{me}(t)(L_\mathrm{d}i_\mathrm{d}(t) + \Psi_\mathrm{p}) - R_\mathrm{s}i_\mathrm{q}(t)}{L_\mathrm{q}}.\end{aligned} \tag{1}$$

Herein, the direct and quadrature current components are denoted by i_d and i_q, respectively. The corresponding voltages are labeled u_d and u_q, and the angular velocity is denoted by ω_me. The continuous time t will be omitted in the rest of this article wherever possible to shorten notation. A definition of the physical parameters within these equations is given in Table I, wherein the first five parameters describe the motor's physical properties, while the last three define the operation space. Note that the dynamic system model defined by (1) is a simplification of the real-world application that is known as the PMSM's fundamental wave model. The rotor of a PMSM contains the namesake permanent magnets, which can be located either on its surface (SPMSM) or within its interior (IPMSM). In terms of electrical properties, $L_\mathrm{d} = L_\mathrm{q}$ is always satisfied for SPMSMs.The task under investigation is to control the dq current components, by actuating a two-level power inverter (B6-bridge) using pulsewidth modulation. The control algorithm is able to command u^*_d and u^*_q directly within the range of $[-\frac{U_\mathrm{DC}}{2}, \frac{U_\mathrm{DC}}{2}]$. The resulting voltage vector $\boldsymbol{u}^*_\mathrm{dq}$ is then subjected to the limitations of the physical three-phase system:

$$\begin{aligned}\begin{bmatrix}u_\mathrm{d}\\ u_\mathrm{q}\end{bmatrix} &= \boldsymbol{T}_\mathrm{abc,dq}(\varepsilon_\mathrm{el})\ \mathrm{clip}\left(\boldsymbol{T}_\mathrm{dq,abc}(\varepsilon_\mathrm{el})\begin{bmatrix}u^*_\mathrm{d}\\ u^*_\mathrm{q}\end{bmatrix}, -\frac{U_\mathrm{DC}}{2}, \frac{U_\mathrm{DC}}{2}\right)\\ \boldsymbol{T}_\mathrm{dq,abc}(\varepsilon_\mathrm{el}) &= \begin{bmatrix}1 & 0\\ -\frac{1}{2} & \frac{\sqrt{3}}{2}\\ -\frac{1}{2} & -\frac{\sqrt{3}}{2}\end{bmatrix}\begin{bmatrix}\cos(\varepsilon_\mathrm{el}) & -\sin(\varepsilon_\mathrm{el})\\ \sin(\varepsilon_\mathrm{el}) & \cos(\varepsilon_\mathrm{el})\end{bmatrix}\\ \boldsymbol{T}_\mathrm{abc,dq}(\varepsilon_\mathrm{el}) &= \begin{bmatrix}\cos(\varepsilon_\mathrm{el}) & \sin(\varepsilon_\mathrm{el})\\ -\sin(\varepsilon_\mathrm{el}) & \cos(\varepsilon_\mathrm{el})\end{bmatrix}\begin{bmatrix}\frac{2}{3} & -\frac{1}{3} & -\frac{1}{3}\\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}}\end{bmatrix}\end{aligned} \tag{2}$$

with $\boldsymbol{u}^*_\mathrm{dq}$ being the commanded and $\boldsymbol{u}_\mathrm{dq}$ the applied voltages. The operator $T_\mathrm{abc,dq}$ corresponds to the subsequent application of the Clarke transformation [17] and the Park transformation [18], whereas $T_\mathrm{dq,abc}$ denotes the associated inverse.

TABLE I
Parameters That Define the Behavior of a PMSM

Parameter	Unit	Description
L_d	H	d-axis inductance
L_q	H	q-axis inductance
R_s	Ω	Stator resistance
p	1	Pole pair number
Ψ_p	V·s	Permanent flux
U_DC	V	DC-link voltage
I_n	A	Nominal current
$\omega_\mathrm{me,n}/\omega_\mathrm{me,max}$	s^{-1}	Nominal/maximal mechanical angular velocity

III. RL Basics

Classical RL settings consist of an RL control algorithm that interacts repeatedly with the control plant (referred to as environment) at each discrete time step k. First, it is assumed that all states are measurable and are, hence, directly visible in the system output. Then, the actor of a trained RL controller determines the action signal $\boldsymbol{a}_k$ based on the momentary environment state $\boldsymbol{s}_k$. The applied action leads to the next state $\boldsymbol{s}_{k+1}$, which conforms the idea of modeling the environment as a Markov decision process (MDP), defined via the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ denote the possible state and action spaces, respectively. Moreover

$$\mathcal{P}(\boldsymbol{s}_{k+1}, \boldsymbol{s}_k, \boldsymbol{a}_k) = \Pr(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k) \tag{3}$$

is the transition probability function for transfer from $\boldsymbol{s}_k$ to the state $\boldsymbol{s}_{k+1}$ via action $\boldsymbol{a}_k$, which represents the mapping $\mathcal{P}: \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$. Furthermore, the action is rewarded with r_{k+1}, defined by a reward function $\mathcal{R}: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. From a control engineering point of view, the reward represents intended control objectives. Hence, sensible design can improve the learning speed considerably. The discount factor $\gamma \in [0, 1[$ defines how far-sighted the algorithm chooses its control actions.

[1]Moreover, the entire training and testing MRL routines of this article are disclosed as open-source code at https://github.com/upb-lea/meta_RL_PMSM.

[2]GEM can be found under https://github.com/upb-lea/gym-electric-motor.

In an MDP, the transition from state $\boldsymbol{s}_k$ to state $\boldsymbol{s}_{k+1}$ only depends on the momentary state

$$\Pr(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k) = \Pr(\boldsymbol{s}_{k+1}|\boldsymbol{s}_0, \ldots, \boldsymbol{s}_k, \boldsymbol{a}_k). \quad (4)$$

When utilizing RL within a control scenario, the general task is to determine the optimal policy that relates each state to an action that can maximize the return g_k

$$\begin{aligned} g_k &= \mathrm{E}\{G_k|\boldsymbol{S}_k = \boldsymbol{s}_k, \boldsymbol{A}_k = \boldsymbol{a}_k\} \\ &= \mathrm{E}\left\{\sum_{i=0}^{\infty} \gamma^i R_{k+i+1}|\boldsymbol{s}_k, \boldsymbol{a}_k\right\}. \end{aligned} \quad (5)$$

Herein, $\mathrm{E}\{G_k\}$ denotes the expected value of the return with capital letters denoting random variables, while lowercase letters denote respective realizations. The optimal return can be defined by means of the Bellman optimality equation

$$q^*(\boldsymbol{s}_k, \boldsymbol{a}_k) = \mathrm{E}\{R_{k+1} + \gamma \max_{\boldsymbol{a}_{k+1}} q^*(\boldsymbol{s}_{k+1}, \boldsymbol{a}_{k+1})|\boldsymbol{s}_k, \boldsymbol{a}_k\} \quad (6)$$

where $q(\boldsymbol{s}_k, \boldsymbol{a}_k)$ denotes the action value function $q: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ that links the action $\boldsymbol{a}_k$ in state $\boldsymbol{s}_k$ to the expected return. To find the best applicable action, a policy function $\boldsymbol{a}_k = \boldsymbol{\pi}(\boldsymbol{s}_k)$ with $\boldsymbol{\pi}: \mathcal{S} \rightarrow \mathcal{A}$ needs to be learned that satisfies

$$\max_{a_k} q(\boldsymbol{s}_k, \boldsymbol{a}_k) = q(\boldsymbol{s}_k, \boldsymbol{\pi}^*(\boldsymbol{s}_k)). \quad (7)$$

For environments with continuous sets of states and/or actions, approximate solutions such as artificial neural networks (ANNs) for the policy function and value estimation are required. Prominent utilization of ANNs in RL can be found in actor–critic methods, which make use of two separated ANNs to estimate the action values and the policy separately. This is achieved by employing an actor network $\boldsymbol{\pi}_{\boldsymbol{\phi}}$ and a critic network $\hat{q}_{\boldsymbol{\theta}}$ with network parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. Contemporary algorithms from this class are able to learn in an off-policy fashion, which means that they are also capable of learning from actions that do not fit the momentary policy. This allows us to utilize a replay buffer $\mathcal{B}$ where past samples are stored, enabling the consideration of larger batches of experiences to learn in a data-efficient way.

The definition of the Bellman equation (6) suggests the corresponding cost function that is used for optimizing the critic network parameters $\boldsymbol{\theta}$

$$J(\boldsymbol{\theta}) = \left(r_{k+1} + \gamma \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{s}_{k+1}, \boldsymbol{\pi}_{\boldsymbol{\phi}}(\boldsymbol{s}_{k+1})) - \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{s}_k, \boldsymbol{a}_k)\right)^2. \quad (8)$$

Assuming that the critic delivers a reliable estimate of the action value q, the gradient

$$\nabla_{\boldsymbol{\phi}} J(\boldsymbol{\phi}) = \nabla_{\boldsymbol{\phi}} \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{s}_k, \boldsymbol{\pi}_{\boldsymbol{\phi}}(\boldsymbol{s}_k)) \quad (9)$$

points in the direction of policy improvement and can, hence, be utilized to optimize the actor parameters $\boldsymbol{\phi}$. In order to keep this introduction to the basics of RL concise, further details are omitted at this point. More in-depth summaries of the RL fundamentals can be found, e.g., in [12] and [19].

In electric drive control, off-policy RL-based algorithms have already been implemented successfully to control currents and torque, respectively, but only for individual drive applications [5], [6], [7]. The transfer to a wide set of different drives was not investigated so far and is, therefore, the central challenge addressed in this contribution.

IV. Context-Based MRL

In spite of the beneficial traits of RL control that allow independence from plant-specific model knowledge, a main drawback of such methods is the lack of generalizability concerning a learned policy. If presented to another task from the same problem class (e.g., a plant system that is described by the same ODE but is characterized by different parameters), the learned optimal policy $\boldsymbol{\pi}^*$ would not fit the new task and must, therefore, be retrained. In terms of PMSM control, this means that for each motor with different physical properties, the algorithm's training needs to be conducted again, which may still be tolerable for special applications, but is rather aggravating for large-scale production.

The goal of meta-learning is to systematically speed up the training of machine learning (ML) algorithms when presented to new but similar tasks [13]. Concerning RL, this means that the algorithm would be able to adapt to different environments. One way to model this scenario is based on picturing each different motor as a partially observable MDP (POMDP). In a POMDP, the state $\boldsymbol{s}_k$ is not completely measurable and is, hence, not entirely available to the RL algorithm. It is described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega, \gamma)$, wherein Ω is the set of observations available to the algorithm and observation function $\mathcal{O}$, which is a probability distribution over possible observations given an action with a resulting state.

Different algorithms in MRL have emerged to handle such scenarios. However, most of them necessitate on-policy training and may, therefore, lack sample efficiency [20], [21]. A class of algorithms that have shown promising and sample efficient results on simulated and real tasks are context-based off-policy algorithms [14]. Here, a further ANN is introduced, which generates additional information about the momentary environment. These so-called context variables $\boldsymbol{z}$ are computed by means of the context network $\hat{c}_{\boldsymbol{\xi}}$ with parameters $\boldsymbol{\xi}$

$$\boldsymbol{z}_j = \hat{c}_{\boldsymbol{\xi}}(\mathcal{B}_{\mathrm{c}}^j) \quad (10)$$

wherein $\mathcal{B}_{\mathrm{c}}^j$ is a commissioning buffer that contains several state transitions of the form $(\boldsymbol{s}_k, \boldsymbol{a}_k, \boldsymbol{s}_{k+1})$ that have been recorded on the jth environment.

Furthermore, feature engineering is usually employed to present an enriched observation vector $\boldsymbol{o} = \boldsymbol{f}(\boldsymbol{s})$ to the control algorithm in order to facilitate the training process. Here, $\boldsymbol{f}(\cdot)$ is the feature function that extracts important information from the state $\boldsymbol{s}$, which is highly problem dependent.

In the corresponding framework, the action value network $\hat{q}_{\boldsymbol{\theta}}$ is augmented to accept the context $\boldsymbol{z}$ as an additional input. The Bellman equality equation (6), therefore, results to

$$\begin{aligned} q^*(\boldsymbol{o}_k, \boldsymbol{a}_k, \boldsymbol{z}_j) = \mathrm{E}\{&R_{k+1} \\ &+ \gamma \max q^*(\boldsymbol{o}_{k+1}, \boldsymbol{a}_{k+1}, \boldsymbol{z}_j)|\boldsymbol{s}_k, \boldsymbol{a}_k, \boldsymbol{z}_j\}. \end{aligned} \quad (11)$$

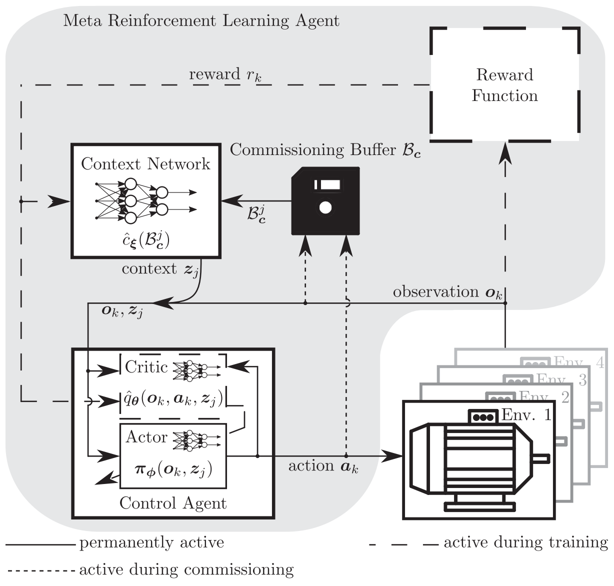
Meta Reinforcement Learning Agent
reward r_k
Reward Function
Context Network
$\hat{c}_{\boldsymbol{\xi}}(\mathcal{B}_c^j)$
Commissioning Buffer $\mathcal{B}_c$
$\mathcal{B}_c^j$
context $\boldsymbol{z}_j$
observation $\boldsymbol{o}_k$
$\boldsymbol{o}_k, \boldsymbol{z}_j$
Critic
$\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, \boldsymbol{a}_k, \boldsymbol{z}_j)$
Actor
$\boldsymbol{\pi}_{\boldsymbol{\phi}}(\boldsymbol{o}_k, \boldsymbol{z}_j)$
Control Agent
action $\boldsymbol{a}_k$
Env. 1
Env. 2
Env. 3
Env. 4
permanently active
active during training
active during commissioning

Fig. 1. Overall scheme of the MRL setup.

The cost function (8) is altered accordingly and can then be used to also optimize the context network $\hat{c}_{\boldsymbol{\xi}}$:

$$J(\boldsymbol{\theta}, \boldsymbol{\xi}) = (r_{k+1} + \gamma \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_{k+1}, \boldsymbol{\pi}_{\boldsymbol{\phi}}(\boldsymbol{o}_{k+1}), \boldsymbol{z}_j) - \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, \boldsymbol{a}_k, \boldsymbol{z}_j))^2. \tag{12}$$

Important contributions to this class of MRL algorithms are probabilistic embeddings for actor–critic RL (PEARL) [22] and meta-Q-learning (MQL) [23], which differ concerning their design of context variables. PEARL uses random nonconsecutive transitions from the recent additions to the replay buffer to generate a probabilistic context. MQL uses a recurrent ANN to generate a context from the last buffered state transitions. This contribution has taken inspiration from both of these approaches; a schematic of the targeted MRL structure is depicted in Fig. 1.

This contribution utilizes twin delayed deep deterministic policy gradient (TD3) [21] as its base RL algorithm, which is a state-of-the-art method for RL problems with continuous state and action space. Less-recent RL algorithms, such as deep deterministic policy gradient [24] or advantage actor critic [25], are also suited for the current control task [5], [6], but are not discussed within the scope of this article to focus the extension to meta learning.

In addition to the usual actor and critic networks that are standard for the TD3 structure, a further context ANN is introduced. This context network processes the plant specific observation transitions, which will be investigated in the following section. The RL controller operates on a given plant with random initialization and acts according to the momentary policy $\boldsymbol{\pi}_{\boldsymbol{\phi}}$, superimposed by a Gaussian exploration noise β_e. The resulting transitions and rewards are then saved to the replay buffer $\mathcal{B}$. This rollout procedure is described in Algorithm 1.

After finishing the rollout, the MRL algorithm's networks are updated via standard gradient descent. For this, the update routine of TD3 is applied under the consideration of the added

Algorithm 1: Rollout.

```
Require: Policy $\boldsymbol{\pi}_{\boldsymbol{\phi}}$, motor $m_j$, buffer $\mathcal{B}$, number of steps
  $N_\mathrm{steps}$, number of steps per episode $N_\mathrm{episode}$,
optional: context $\boldsymbol{z}_j$
1: $n \leftarrow 0$
2: while $n < N_\mathrm{steps}$ do
3:    $k \leftarrow 0$
4:    Initialize motor $m_j$ with random
      $\widetilde{i}_{\mathrm{d},0}, \widetilde{i}_{\mathrm{q},0} \sim \mathcal{U}(-\frac{2}{3}, \frac{2}{3})$,
      $\widetilde{\varepsilon}_{\mathrm{el},0}, \widetilde{\omega}_{\mathrm{me},0}, \widetilde{i}^*_{\mathrm{q},0} \sim \mathcal{U}(-1, 1), \widetilde{i}^*_{\mathrm{d},0} \sim \mathcal{U}(-1, 0)$ under
      constraint (17)
5:    Obtain initial state $\boldsymbol{o}_0$
6:    while $k < N_\mathrm{episode}$ and $\widetilde{i}^2_{\mathrm{d},k} + \widetilde{i}^2_{\mathrm{q},k} < 1$ do
7:       $\beta_\mathrm{e} \sim \mathrm{clip}(\mathcal{N}_\mathrm{e}(0, \sigma_\mathrm{e}), -c_\mathrm{e}, c_\mathrm{e})$
8:       Determine action $\boldsymbol{a}_k = \boldsymbol{\pi}_{\boldsymbol{\phi}}(\boldsymbol{o}_k, \boldsymbol{z}_j) + \beta_\mathrm{e}$
9:       Execute $\boldsymbol{a}_k$ on $m_j$
10:      Observe $r_{k+1}$ and $\boldsymbol{o}_{k+1}$
11:      Store transition $(\boldsymbol{o}_k, \boldsymbol{a}_k, r_{k+1}, \boldsymbol{o}_{k+1}, j)$ in $\mathcal{B}$
12:      $k \leftarrow k + 1$
13:   end while
14:   $n \leftarrow n + k$
15: end while
16: return Buffer $\mathcal{B}$
```

context network. The context network is updated with the same loss as the critic to ensure that the context variables allow a sensible distinction of plant systems with regard to their action values. The update routine is described in Algorithm 2 and makes use of the well-established target networks [26] in order to stabilize the training process.

V. Considered Motor Drive Set

This section presents the preliminary considerations and software tools to prepare a representative and robust dataset of heterogeneous PMSM drives for the MRL training. This is necessary because both the selection of the motors and the design of the context depend on a balanced set of drive systems, whereas a badly prepared training set could decrease the MRL algorithm's ability to generalize.

A. Overview

To train and validate an MRL algorithm for the control of PMSMs, different PMSM parameterizations need to be available. Since no comprehensive database was publicly available to the best knowledge of the authors, parameter sets have been collected first. For this, only sources with complete parameter sets were considered. Necessary parameters were not only electrical characteristics of the PMSM but also operating limitations, as described in Table I. Different sources were used, such as publicly available catalogs for industrial motors as well as scientific papers. Through this, a total of 566 different motor parameter sets were collected from power classes ranging from a few watts to hundreds of kilowatts. As a supplementary part of this contribution, the corresponding motor database has been made

Algorithm 2: Update.

```
Require: Actor $\boldsymbol{\pi}_{\boldsymbol{\phi}}$, critics $\hat{q}_{\boldsymbol{\theta}_1}, \hat{q}_{\boldsymbol{\theta}_2}$, context network $\hat{c}_{\boldsymbol{\xi}}$, target networks for each (denoted by $'$), buffer $\mathcal{B}$, commissioning buffers $\mathcal{B}_\mathrm{c}$, number of update steps $N_\mathrm{update}$
1: $n \leftarrow 0$
2: while $n < N_\mathrm{update}$ do
3:   Sample minibatch $(\boldsymbol{o}_k, \boldsymbol{a}_k, r_{k+1}, \boldsymbol{o}_{k+1}, j)^{N_\mathrm{mb}}$ from $\mathcal{B}$
4:   for each $j$ in minibatch do
5:     $\boldsymbol{z}_j = \hat{c}_{\boldsymbol{\xi}}(\mathcal{B}_\mathrm{c}^j)$, $\boldsymbol{z}'_j = \hat{c}_{\boldsymbol{\xi}'}(\mathcal{B}_\mathrm{c}^j)$
6:   end for
7:   Calculate policy noise
     $\beta_{\boldsymbol{\pi}} \sim \mathrm{clip}(\mathcal{N}_{\boldsymbol{\pi}}(0, \sigma_{\boldsymbol{\pi}}), -c_{\boldsymbol{\pi}}, c_{\boldsymbol{\pi}})$
8:   Update critic and context network:
9:     $l \leftarrow \underset{l=1,2}{\arg\min} \underbrace{\hat{q}_{\boldsymbol{\theta}'_l}(\boldsymbol{o}_{k+1}, \boldsymbol{\pi}_{\boldsymbol{\phi}'}(\boldsymbol{o}_{k+1}, \boldsymbol{z}'_j) + \beta_{\boldsymbol{\pi}}, \boldsymbol{z}'_j)}_{g_{l,k+1}}$
10:    $J(\boldsymbol{\theta}_l, \boldsymbol{\xi}) =$
11:      $N_\mathrm{mb}^{-1} \sum (r_{k+1} + \gamma\, g_{l,k+1} - \hat{q}_{\boldsymbol{\theta}_l}(\boldsymbol{o}_k, \boldsymbol{a}_k, \boldsymbol{z}_j))^2$
12:    $\boldsymbol{\theta}_l \leftarrow \boldsymbol{\theta}_l - \alpha_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}_l} J(\boldsymbol{\theta}_l, \boldsymbol{\xi})$
13:    $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \alpha_{\boldsymbol{\xi}} \nabla_{\boldsymbol{\xi}} J(\boldsymbol{\theta}_l, \boldsymbol{\xi})$
14:  if $n \bmod d == 0$ then
15:    Update actor:
16:      $J(\boldsymbol{\phi}) = N_\mathrm{mb}^{-1} \sum \hat{q}_{\boldsymbol{\theta}_1}(\boldsymbol{o}_k, \boldsymbol{\pi}_{\boldsymbol{\phi}}(\boldsymbol{o}_k), \boldsymbol{z}_j)$
17:      $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} J(\boldsymbol{\phi})$
18:  Update target networks:
19:      $\boldsymbol{\theta}'_l \leftarrow \tau_\mathrm{t} \boldsymbol{\theta}_l + (1-\tau)\boldsymbol{\theta}'_l$ for $l = 1, 2$
20:      $\boldsymbol{\phi}' \leftarrow \tau_\mathrm{t} \boldsymbol{\phi} + (1-\tau)\boldsymbol{\phi}'$
21:      $\boldsymbol{\xi}' \leftarrow \tau_\mathrm{t} \boldsymbol{\xi} + (1-\tau)\boldsymbol{\xi}'$
22:  end if
23:  $n \leftarrow n + 1$
24: end while
```

publicly available (cf. [27]). The collected database contains 552 sets of SPMSM parameters and 14 sets of IPMSM parameters.

B. Normalization and Preprocessing

For the selection of the drives for training and test sets, the first intuition is to choose a balanced distribution of physical parameters. However, this might not be the best distribution over actual dynamic behavior of the motors. Instead, a different representation of a motor can be derived, which describes its dynamic behavior in a more target-oriented way. Transforming the model presented in (1) to a representation with normalized variables $\widetilde{\omega}_\mathrm{me}, \widetilde{u}_\mathrm{d}, \widetilde{u}_\mathrm{q}, \widetilde{i}_\mathrm{d}, \widetilde{i}_\mathrm{q} \in [-1, 1]$, the ODEs can be reformulated to yield a different representation of the parameter space. With $s_\omega = \omega_\mathrm{me,max}$, $s_\mathrm{i} = 1.5\, I_n$, $s_\mathrm{u} = \frac{U_\mathrm{DC}}{2}$ as scaling factors of ω, i, and u, respectively, the first equation from (1) evaluates to

$$s_\mathrm{i} \frac{\mathrm{d}\widetilde{i}_\mathrm{d}}{\mathrm{d}t} = \frac{s_\mathrm{u}\widetilde{u}_\mathrm{d} + p s_\omega \widetilde{\omega}_\mathrm{me} L_\mathrm{q} s_\mathrm{i} \widetilde{i}_\mathrm{q} - R_\mathrm{s} s_\mathrm{i} \widetilde{i}_\mathrm{d}}{L_\mathrm{d}}$$
$$\Leftrightarrow \quad \frac{\mathrm{d}\widetilde{i}_\mathrm{d}}{\mathrm{d}t} = \underbrace{\frac{s_\mathrm{u}}{L_\mathrm{d} s_\mathrm{i}}}_{p_1} \widetilde{u}_\mathrm{d} + \underbrace{\frac{p s_\omega L_\mathrm{q}}{L_\mathrm{d}}}_{p_2} \widetilde{\omega}_\mathrm{me} \widetilde{i}_\mathrm{q} - \underbrace{\frac{R_\mathrm{s}}{L_\mathrm{d}}}_{p_3} \widetilde{i}_\mathrm{d}. \tag{13}$$

TABLE II
PARAMETERS DESCRIBING THE DYNAMIC BEHAVIOR OF A MOTOR IN A NORMALIZED SPACE

p_1	p_2	p_3	p_4	p_5	p_6	p_7
$\frac{s_\mathrm{u}}{L_\mathrm{d} s_\mathrm{i}}$	$\frac{p s_\omega L_\mathrm{q}}{L_\mathrm{d}}$	$-\frac{R_\mathrm{s}}{L_\mathrm{d}}$	$\frac{s_\mathrm{u}}{L_\mathrm{q} s_\mathrm{i}}$	$-\frac{p s_\omega L_\mathrm{d}}{L_\mathrm{q}}$	$-\frac{p s_\omega \Psi_\mathrm{p}}{L_\mathrm{q} s_\mathrm{i}}$	$-\frac{R_\mathrm{s}}{L_\mathrm{q}}$

Analogously, the second ODE changes to

$$s_\mathrm{i} \frac{\mathrm{d}\widetilde{i}_\mathrm{q}}{\mathrm{d}t} = \frac{s_\mathrm{u}\widetilde{u}_\mathrm{q} - p s_\omega \widetilde{\omega}_\mathrm{me}(L_\mathrm{d} s_\mathrm{i} \widetilde{i}_\mathrm{d} + \Psi_\mathrm{p}) - R_\mathrm{s} s_\mathrm{i} \widetilde{i}_\mathrm{q}}{L_\mathrm{q}}$$
$$\Leftrightarrow \quad \frac{\mathrm{d}\widetilde{i}_\mathrm{q}}{\mathrm{d}t} = \underbrace{\frac{s_\mathrm{u}}{L_\mathrm{q} s_\mathrm{i}}}_{p_4} \widetilde{u}_\mathrm{q} - \underbrace{\frac{p s_\omega L_\mathrm{d}}{L_\mathrm{q}}}_{p_5} \widetilde{\omega}_\mathrm{me} \widetilde{i}_\mathrm{d} - \underbrace{\frac{p s_\omega \Psi_p}{L_\mathrm{q} s_\mathrm{i}}}_{p_6} \widetilde{\omega}_\mathrm{me} - \underbrace{\frac{R_\mathrm{s}}{L_\mathrm{q}}}_{p_7} \widetilde{i}_\mathrm{d}. \tag{14}$$

Note that for SPMSMs, $p_1 = p_4$, $p_2 = -p_5$, and $p_3 = p_7$ hold. To have a balanced distribution of the dynamic motor behavior within training and test, the ODE coefficients from Table II were used for the preparation of the corresponding datasets. To reduce the number of SPMSMs, a downsampling strategy was applied: the first batch of 35 SPMSMs was drawn with the routine described in Algorithm 3, which targets an extensive coverage of the parameter space by maximizing the sum of distances between the points. This selection routine tends to select parameter sets on the edges of the parameter space. Therefore, an addition of 40 SPMSMs were randomly added in order to also get an even coverage of the center, adding up to a utilized set of 75 parameters.

To include edge cases of dynamic motor behavior into the training set, the convex hull over these 75 SPMSM's nonequal parameters is determined. The SPMSM parameter sets that are identified to be vertices of this hull are added to the training set. Further parameter sets were selected randomly using a uniform distribution until the training set contained 50 different parameter vectors. The remaining 25 parameter sets are attributed to the test set.

Owing to the low number of publicly accessible IPMSM parameter sets, additional feasible sets were to be generated synthetically from the acquired ones. These data were created using the synthetic minority oversampling technique (SMOTE), which is a method originally designed to deal with classification tasks [28]. There, unbalanced datasets are problematic because a classifier can achieve a high score on training data even without the ability to correctly classify the minority class if that class is strongly underrepresented. SMOTE does sample synthetically on the basis of available data by generating new samples in between close data points of a given class, which is specified in Algorithm 4. Owing to the algebraic dependence between several ODE coefficients, it is not necessary to generate new values for the complete set of coefficients. Instead, it can be exploited that

$$\frac{p_3}{p_7} = \frac{p_1}{p_4} = \sqrt{-\frac{p_2}{p_5}} = \frac{L_\mathrm{q}}{L_\mathrm{d}}. \tag{15}$$

Algorithm 3: Downsampling for Space Coverage.

```
Require: Data points $\mathcal{X}_{\text{original}}$ from majority class, goal
  number of data points $N_{\text{goal}}$
1: Initialize empty data buffer $\mathcal{X}_{\text{new}}$
2: Sample random data point $\boldsymbol{x}$ from $\mathcal{X}_{\text{original}}$
3: Remove $\boldsymbol{x}$ from $\mathcal{X}_{\text{original}}$ and put it into $\mathcal{X}_{\text{new}}$
4: $\Sigma_{\text{old}} \leftarrow 0$
5: while $|\mathcal{X}_{\text{new}}| < N_{\text{goal}}$ do
6:    $\Sigma_{\text{new}} \leftarrow 0$
7:    for each $\boldsymbol{x}_1 \in \mathcal{X}_{\text{original}}$ do
8:       for each $\boldsymbol{x}_2 \in \mathcal{X}_{\text{new}}$ do
9:          $\Sigma_{\text{new}} \leftarrow \Sigma_{\text{new}} + ||(\boldsymbol{x}_1 - \boldsymbol{x}_2)||_2^2$
10:      end for
11:      if $\Sigma_{\text{new}} > \Sigma_{\text{old}}$ then
12:         $\boldsymbol{x}_{\text{next}} \leftarrow \boldsymbol{x}_1$
13:         $\Sigma_{\text{old}} \leftarrow \Sigma_{\text{new}}$
14:      end if
15:   end for
16:   Remove $\boldsymbol{x}_{\text{next}}$ from $\mathcal{X}_{\text{original}}$ and put it into $\mathcal{X}_{\text{new}}$
17: end while
```

Algorithm 4: SMOTE.

```
Require: Set of parameter vectors $\mathcal{X}_{\text{original}}$ from
  underrepresented class, targeted number of data points
  $N_{\text{goal}}$, number of nearest neighbors $K$
1: Find $K$ nearest neighbors for each element $\boldsymbol{p} \in \mathcal{X}_{\text{original}}$
2: Initialize data buffer $\mathcal{X}_{\text{goal}} \leftarrow \mathcal{X}_{\text{original}}$
3: while $|\mathcal{X}_{\text{goal}}| < N_{\text{goal}}$ do
4:    Sample random $\boldsymbol{p}_1 \in \mathcal{X}_{\text{original}}$
5:    Sample random $\boldsymbol{p}_2$ from $\boldsymbol{p}_1$'s $K$ nearest neighbors
6:    Generate $\boldsymbol{p}_{\text{new}} = \alpha\boldsymbol{p}_1 + (1-\alpha)\boldsymbol{p}_2$ with $\alpha \sim \mathcal{U}(0,1)$
7:    $\mathcal{X}_{\text{goal}} \leftarrow \{\mathcal{X}_{\text{goal}}, \boldsymbol{p}_{\text{new}}\}$
8: end while
```

TABLE III
NUMBER OF CONSIDERED MOTOR PARAMETER SETS FOR THE TRAINING AND TESTING OF CONTEXT-BASED MRL ALGORITHMS

real/synthetic parameter sets	SPMSM	IPMSM
training set	50/0	14/36
test set	25/0	0/25

Therefore, SMOTE needs to be applied only to the coefficients p_1, p_2, p_3, p_6, and p_7. For each synthetically derived set of coefficients, it was verified that it lies within the convex hull of the 14 original parameter sets.[3]

For the IPMSMs, the original 14 parameter vectors are directly utilized as training data, while the rest was selected randomly using a uniform distribution. The breakdown of training and testing sets is concisely listed in Table III. Fig. 2 features the distribution of p_1 against p_4 for training and test sets in an exemplary fashion. Further visualizations as well as all 150 normalized PMSM parameter sets are available at [27] to supplement this article.

[3]To derive the corresponding physical parameters needed for simulation, $R_s = 1\,\Omega, p = 4$, and $I_n = 5$ A were assumed as fixed values.

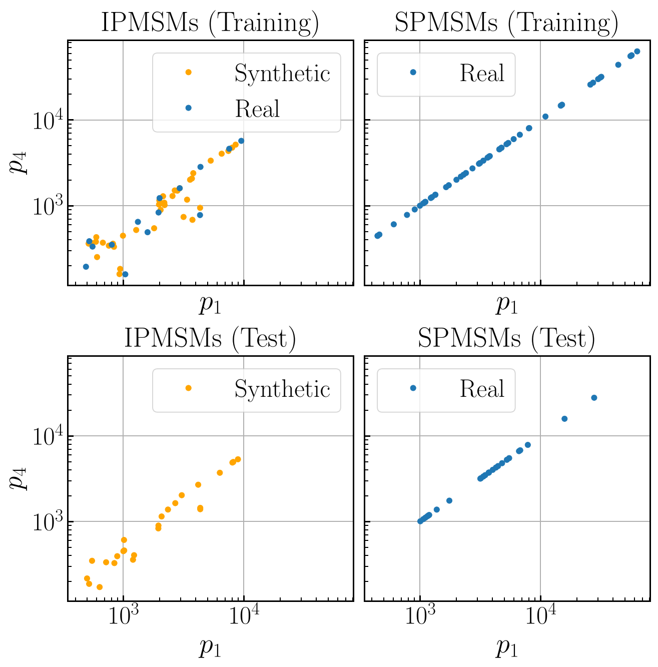

Fig. 2. p_1 plotted against p_4 for training and test sets.

C. Context Design

The context design of this work differs from the ones suggested by PEARL and MQL: because the ODE coefficients $p_{1\ldots7}$ can be assumed to be constant for each PMSM, learning static context variables is sufficient to characterize individual environments. To yield a static context, the input of the context network $\hat{c}_{\boldsymbol{\xi}}$ must be kept static as well for each given environment. Therefore, the transitions used are not sampled from the regular replay buffer $\mathcal{B}$ but from a separated prefilled commissioning buffer $\mathcal{B}_{\text{c}}^j$ for each motor m_j. This buffer contains observed transitions $\boldsymbol{e}_j = (\boldsymbol{o}_{\text{c},k}, \boldsymbol{a}_{\text{c},k}, \boldsymbol{o}_{\text{c},k+1})_j$, with

$$\boldsymbol{o}_{\text{c},k} = \begin{bmatrix} \tilde{i}_{\text{d},k} & \tilde{i}_{\text{q},k} & \tilde{\omega}_{\text{me},k} & \cos(\varepsilon_{\text{el},k}) & \sin(\varepsilon_{\text{el},k}) \end{bmatrix}^\top$$
$$\boldsymbol{a}_{\text{c},k} = \begin{bmatrix} \tilde{u}^*_{\text{d},k} & \tilde{u}^*_{\text{q},k} \end{bmatrix}^\top. \quad (16)$$

To ensure a balanced specification of each motor's dynamic behavior within these buffers $\mathcal{B}_{\text{c}}^j$, coverage of the entire observation-action space is necessary. Since only a finite set of sampled transitions are computationally feasible, it is targeted to cover the observation-action space as well as possible with a limited number of observation transitions. For this, the density-estimation-based state-space coverage acceleration (DESSCA) is used [29]. DESSCA utilizes kernel density estimation to evaluate the coverage of the observation space and suggests a new sample to minimize the difference to a reference coverage density. Here, the reference coverage is a uniform distribution across the possible values of the observations and actions. In

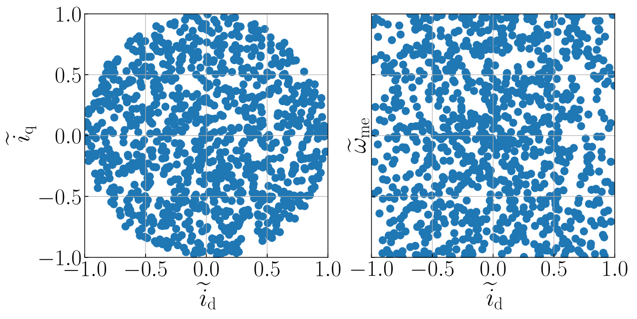

Fig. 3. Joint distribution of $\widetilde{i}_\mathrm{q}$ over $\widetilde{i}_\mathrm{d}$ and $\widetilde{\omega}_\mathrm{me}$ over $\widetilde{i}_\mathrm{d}$ as sampled by DESSCA.

Algorithm 5: DESSCA Sampling for Motor Buffer.

```
Require: Targeted number of DESSCA samples $N_\mathrm{DESSCA}$,
  $\mathcal{M}$ set of motors
1: for each $m_j \in \mathcal{M}$ do
2:   Initialize buffer $\mathcal{B}_\mathrm{c}^j$, $\mathcal{B}_\mathrm{DESSCA}$
3:   Sample $N_\mathrm{DESSCA}$ DESSCA samples to $\mathcal{B}_\mathrm{DESSCA}$
4:   for each $(\boldsymbol{o}_{\mathrm{c},k}, \boldsymbol{a}_{\mathrm{c},k}) \in \mathcal{B}_\mathrm{DESSCA}$ do
5: Initialize motor $m_j$ with observation $\boldsymbol{o}_{\mathrm{c},k}$
6: Execute $\boldsymbol{a}_{\mathrm{c},k}$ on $m_j$ and observe $\boldsymbol{o}_{\mathrm{c},k+1}$
7: Store tuple $(\boldsymbol{o}_{\mathrm{c},k}, \boldsymbol{a}_{\mathrm{c},k}, \boldsymbol{o}_{\mathrm{c},k+1})_j$ in $\mathcal{B}_\mathrm{c}^j$
8:   end for
9: end for
```

addition, the length of the current vector$[i_\mathrm{d} \quad i_\mathrm{q}]^\top$ is limited

$$\sqrt{i_\mathrm{d}^2 + i_\mathrm{q}^2} \leq I_\mathrm{max}. \tag{17}$$

Algorithm 5 outlines how the $\mathcal{B}_c$ buffer is filled for each motor parameter set j. The exemplary distribution featured in Fig. 3 showcases that DESSCA is able to determine a balanced coverage of the observation space. The MRL setup as presented in Fig. 1 shows how these commissioning buffers $\mathcal{B}_\mathrm{c}^j$ are used to generate the context $\boldsymbol{z}_j$ for a given motor m_j. This enables the RL algorithm to adapt its control behavior according to the motor.

VI. Empirical Investigation

In the following, the training and testing of the MRL algorithm is described in detail, and the obtained results are discussed.

A. Training Routine

During training, the MRL algorithm performs a rollout on a specific motor and stores its observations in the replay buffer $\mathcal{B}$. The observation $\boldsymbol{o}$ of the PMSM is defined as follows:

$$\boldsymbol{o} = \begin{bmatrix} \widetilde{i}_\mathrm{d} & \widetilde{i}_\mathrm{q} & \widetilde{\omega}_\mathrm{me} & \cos(\varepsilon_\mathrm{el}) & \sin(\varepsilon_\mathrm{el}) & \widetilde{i}_\mathrm{d}^* & \widetilde{i}_\mathrm{q}^* & \widetilde{i}_\mathrm{d}^2 + \widetilde{i}_\mathrm{q}^2 \end{bmatrix}^\top. \tag{18}$$

The feature $(\widetilde{i}_\mathrm{d}^2 + \widetilde{i}_\mathrm{q}^2)$ aids the RL-based controllers to recognize annealing to the current limit as of (17). The references $\widetilde{i}_\mathrm{d}^*$ and $\widetilde{i}_\mathrm{q}^*$ are the targeted currents for the next sample step. They are generated using a stochastic Wiener process

$$\widetilde{\boldsymbol{i}}_k^* = \begin{bmatrix} \widetilde{i}_{\mathrm{d},k-1}^* + \sim \mathcal{N}(0, \sigma_{\mathrm{d},k}) \\ \widetilde{i}_{\mathrm{q},k-1}^* + \sim \mathcal{N}(0, \sigma_{\mathrm{q},k}) \end{bmatrix} \tag{19}$$

wherein $\sim \mathcal{N}(0, \sigma)$ denotes sampling from a normal distribution with zero mean and σ variance with $\sigma_{\mathrm{d},k}$ and $\sigma_{\mathrm{q},k}$ changing each episode. The generated references are also subjected to the current constraint. In addition, the condition $i_\mathrm{d}^* < 0$ A is respected. By doing so, the current reference values are randomly sampled from the entire feasible dq current half-plane. The reward r_{k+1} is depending on the reference of the last observation and the present observation's current

$$r_{k+1} = \begin{cases} \left(1 - \left(\frac{||(\widetilde{\boldsymbol{i}}_k^* - \widetilde{\boldsymbol{i}}_{k+1})||_1}{4}\right)\right)(1-\gamma), & \text{if } ||\widetilde{\boldsymbol{i}}_{k+1}||_2^2 \leq 1 \\ \frac{(1 - ||\widetilde{\boldsymbol{i}}_{k+1}||_2^2)}{16}(1-\gamma), & \text{else} \end{cases}. \tag{20}$$

The reward function is defined on the interval between $(1-\gamma)$ and $-(1-\gamma)$, ensuring the action values to always lie between -1 and 1 for numerical reasons [7]. During training, a motor parameter set from the training set is drawn. This parameter set is used for the simulation of the motor within the GEM software toolbox [15].

As function-approximation-based RL comes with the risk of training divergence, learning checkpoints are created to allow access to each RL controller's parameterization at its historical performance peak in hindsight [30], [31].

The hyperparameters of the TD3 algorithm, of the MRL setup, and of the training routine in this contribution are depicted in Table IV. A sampling time of $T_\mathrm{s} = 100$ µs was chosen. The number of context variables was set to 8, leaving enough degrees of freedom to allow, e.g., identification of the physical parameters or alternatively of the ODE coefficients without bloating the context vector unnecessarily.

B. Test Routine

After completing the training phase, the RL algorithms' performance has to be evaluated. For this, a testing routine has been developed to validate the control performance on a representative set of situations. The test routine is executed for each RL algorithm variant on each available motor. The given motor is initialized with respect to the currents and the speed. The RL controller then needs to handle its control task by following the given reference. To ensure a balanced coverage of the state space, DESSCA is once again employed to generate the corresponding initial states $(\widetilde{i}_{\mathrm{d},0}, \widetilde{i}_{\mathrm{q},0}, \widetilde{\omega}_{\mathrm{me},0}, \widetilde{i}_{\mathrm{d},0}^*, \widetilde{i}_{\mathrm{q},0}^*)$, wherein the currents are again subjected to the current limit and $\widetilde{i}_\mathrm{d}^* < 0$ holds. The pseudocode of the testing routine is presented in Algorithm 6. Here, a special reference generator was used

$$\widetilde{\boldsymbol{i}}_k^* = \begin{bmatrix} \widetilde{i}_{\mathrm{d},k-1}^* + \sim \mathcal{N}(0, \sigma_{\mathrm{d},k}) - \zeta(\widetilde{i}_{\mathrm{d},k-1}^* - \widetilde{i}_{\mathrm{d},0}^*) \\ \widetilde{i}_{\mathrm{q},k-1}^* + \sim \mathcal{N}(0, \sigma_{\mathrm{q},k}) - \zeta(\widetilde{i}_{\mathrm{q},k-1}^* - \widetilde{i}_{\mathrm{q},0}^*) \end{bmatrix} \tag{21}$$

where a drift back to the initial $\widetilde{\boldsymbol{i}}_0^*$ is added to the Wiener process. The stiffness ζ tunes the strength of this drift. This

TABLE IV
HYPERPARAMETERS AND THEIR VALUES IN THIS ARTICLE

Hyperparameter	Description	Value
TD3 Hyperparameters		
Critic network	Critic architecture	$[100, 100]$
Actor network	Actor architecture	$[100, 100]$
$\alpha_{\boldsymbol{\theta}}$	Critic learning rate	2.5×10^{-3}
$\alpha_{\boldsymbol{\phi}}$	Actor learning rate	3×10^{-4}
γ	Discount factor	0.9
τ_{t}	Target network update rate	5×10^{-3}
σ_{e}	Standard deviation of exploration noise	0.03
c_{e}	Clip value of exploration noise	0.05
$\sigma_{\boldsymbol{\pi}}$	Standard deviation of policy noise	0.01
$c_{\boldsymbol{\pi}}$	Clip value of policy noise	0.02
d	Update frequency	2
N_{mb}	Minibatch size	32
MRL Hyperparameters		
Context network	Context network architecture	$[25, 15]$
$\alpha_{\boldsymbol{\xi}}$	Context network learning rate	2.5×10^{-4}
N_{DESSCA}	Transitions for motor buffers	1000
$\lvert\boldsymbol{z}\rvert$	Number of context variables	8
Training Hyperparameters		
N_{steps}	Rollout steps per motor	1000
N_{up}	Update steps after rollout	200
N_{episode}	Maximum episode length	500
N_{total}	Total training steps	5×10^{6}
$N_{\mathrm{checkpoint}}$	Checkpoint frequency	10^{5}

TABLE V
TEST ROUTINE HYPERPARAMETERS AND THEIR VALUES

Hyperparameter	Description	Value
Test Hyperparameters		
N_{DESSCA}	Number of DESSCA samples / test episodes	20000
N_{episode}	Maximum episode length	50
ζ	Stiffness	0.3

approach is meant to provoke oscillation around a reference point. Table V lists the hyperparameters chosen for the test routine. Episodes have been configured to be rather short, such that the RL algorithm's transient control behavior is weighted roughly equivalent to its steady-state behavior.

C. Evaluation of the Control Performance

The training checkpoints allow us to evaluate the peak control quality of the MRL approach in hindsight. In this article, the decision on the peak performance is based on the training rewards: the rewards were averaged using a moving average filter with a window length of $w = 10^5$ and a shift of $s = 10^5$. The checkpointed control configuration at the reward maximum then got selected for evaluation. As a comparison to the MRL

Algorithm 6: Test Routine.

```
Require: Policy $\boldsymbol{\pi}$, motor $m_j$, initial states sampled with
         DESSCA $\mathcal{B}_{\mathrm{DESSCA}}$, number of steps per episode
         $N_{\mathrm{episode}}$, optional: context $\boldsymbol{z}$
 1: $r_{\mathrm{sum}} \leftarrow 0$
 2: for each $\boldsymbol{b} \in \mathcal{B}_{\mathrm{DESSCA}}$ do
 3:   $k \leftarrow 0$
 4:   Initialize $m_j$ with $\widetilde{i}_{\mathrm{d},0}, \widetilde{i}_{\mathrm{q},0}, \widetilde{\omega}_{\mathrm{me},0}, \widetilde{i}^*_{\mathrm{q},0}, \widetilde{i}^*_{\mathrm{d},0}$ from $\boldsymbol{b}$
 5:   Receive initial observation $\boldsymbol{o}_0$
 6:   while $k < N_{\mathrm{episode}}$ and (17) holds do
 7:     Receive action from policy: $\boldsymbol{a}_k = \boldsymbol{\pi}(\boldsymbol{o}_k, \boldsymbol{z}_j)$
 8:     Execute $\boldsymbol{a}_k$ on $m_j$
 9:     Observe $r_{k+1}$ and $\boldsymbol{o}_{k+1}$
10:     $r_{\mathrm{sum}} \leftarrow r_{\mathrm{sum}} + r_{k+1}$
11:     $k \leftarrow k + 1$
12:   end while
13: end for
14: return $r_{\mathrm{sum}} \cdot (N_{\mathrm{episode}} \cdot \lvert\mathcal{B}_{\mathrm{DESSCA}}\rvert)^{-1}$
```

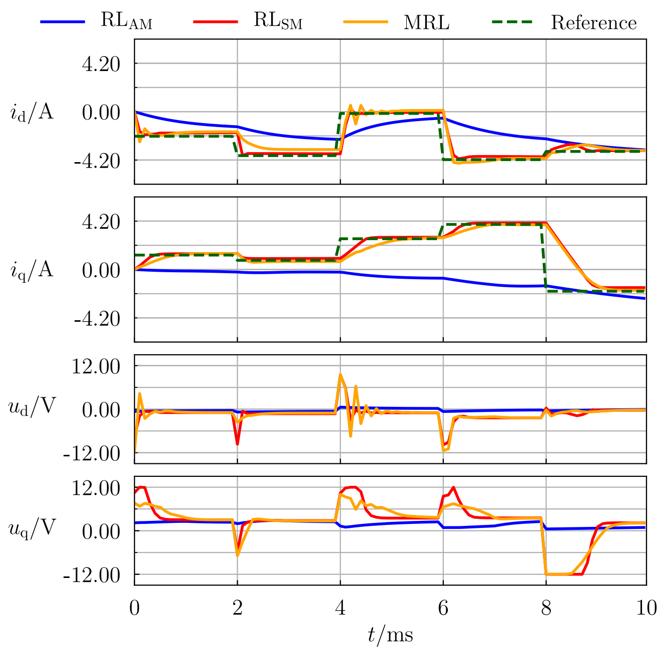

Fig. 4. Test episode on motor m_{67} at $\omega_{\mathrm{me}} = 0.2\,\omega_{\mathrm{me,max}} = 35.6\ \mathrm{s}^{-1}$.

approach, a TD3 algorithm without context variables has been trained on all training motors. This experiment is labeled RL$_{\mathrm{AM}}$ (AM: "all motors"). Moreover, individual TD3 controllers were trained for each motor separately labeled as RL$_{\mathrm{SM}}$ (SM: "single motor"). The corresponding hyperparameters were also configured, as stated in Table IV, and have been trained using the rollout and update loop. Each RL$_{\mathrm{SM}}$ controllers' training was executed only once. For MRL and RL$_{\mathrm{AM}}$, ten trainings were conducted. From these, the RL controller configuration with the highest peak in its learning curve was chosen.

Furthermore, Figs. 4 and 5 show exemplary test episodes that were observed on motors of different power classes. The

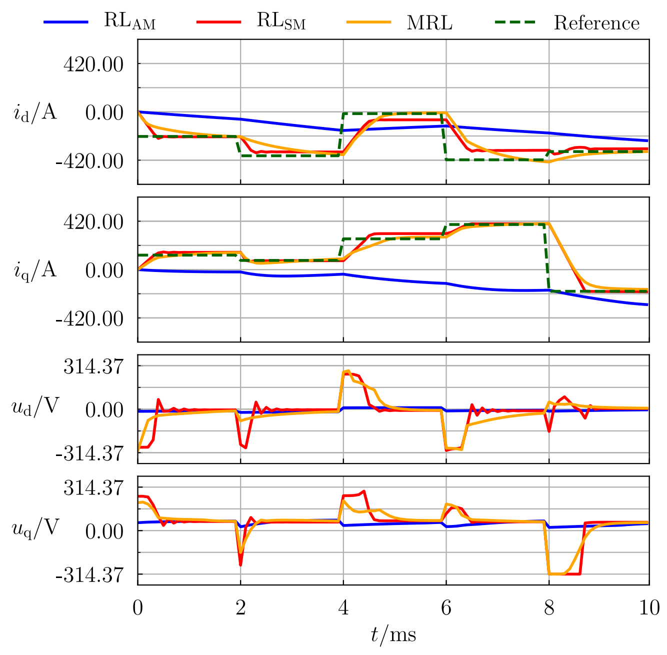

Fig. 5. Test episode on motor m_{116} at $\omega_{\text{me}} = 0.2\,\omega_{\text{me,max}} = 20.94\ \text{s}^{-1}$.

conducted episodes feature several steps of the reference currents, which relate to corresponding changes of the drive torque. Hence, dynamic behavior is highly requested as a characteristic of satisfying control performance, and it can be seen that the RL_{SM} and the MRL controllers react similarly fast, whereas the RL_{AM} reacts slowly and, concerning i_{q}, even away from the reference. Fig. 6 features the behavior of a drive during a speed ramp from negative to positive velocities, which verifies the capability of the RL_{SM} and MRL controller to deal with changing speeds. Both are able to follow the current reference as long as the available voltage allows it, which is (for the presented motor) not the case for speeds close to the maximum speed $\pm\omega_{\text{me,max}}$.

Fig. 7 shows another except from a test episode that demonstrates the adaptive capabilities of the MRL approach. This motor has a rather strong input sensitivity (comparably high p_1, p_4) [cf. (13) and (14)], and a low-performing controller may easily violate the current limitation. It has to be noted that motors with the given characteristics were rather underrepresented in the training dataset for the MRL and RL_{AM} agents. Hence, the RL_{AM} agent's lack in performance and unstable behavior was to be expected. Yet, the MRL agent is able to stabilize the plant system and, although a steady-state error remains, the currents visibly move toward their references.

These experiments highlight the potential of the MRL approach as it is able to stabilize the plant system even without being comprehensively optimized. Fig. 7 also confirms the findings listed in Table VI with the MRL's performance being much better than the RL_{AM} but slightly worse than the RL_{SM}. However, the reference tracking behavior of the MRL controller looks highly promising for a first proof of concept, and an improvement can be expected if optimal hyperparameters for the MRL can be found. Nonetheless, Fig. 7 also highlights the limitations of the MRL's setup. The training routine considers

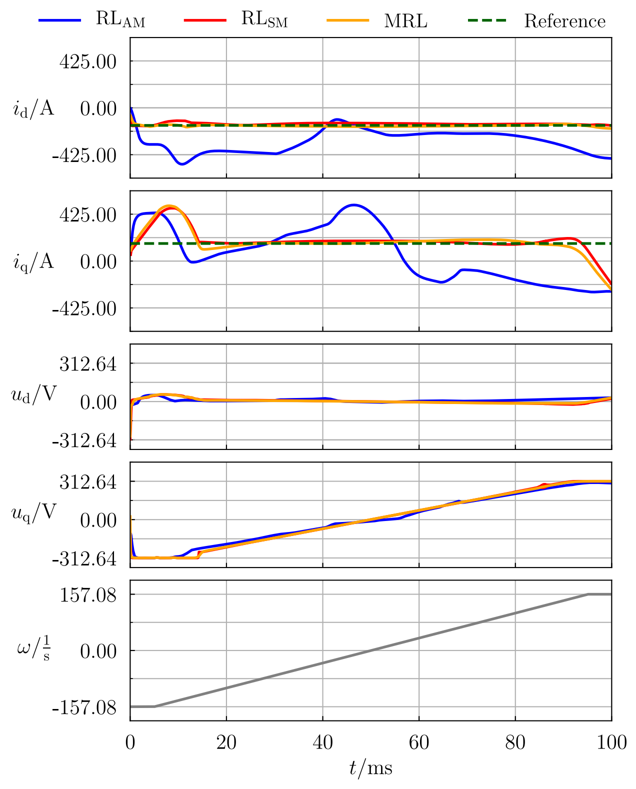

Fig. 6. Test episode with speed ramp from $-\omega_{\text{me,max}}$ to $\omega_{\text{me,max}}$ on motor m_{45} with $\omega_{\text{me,max}} = 157.08\ \text{s}^{-1}$.

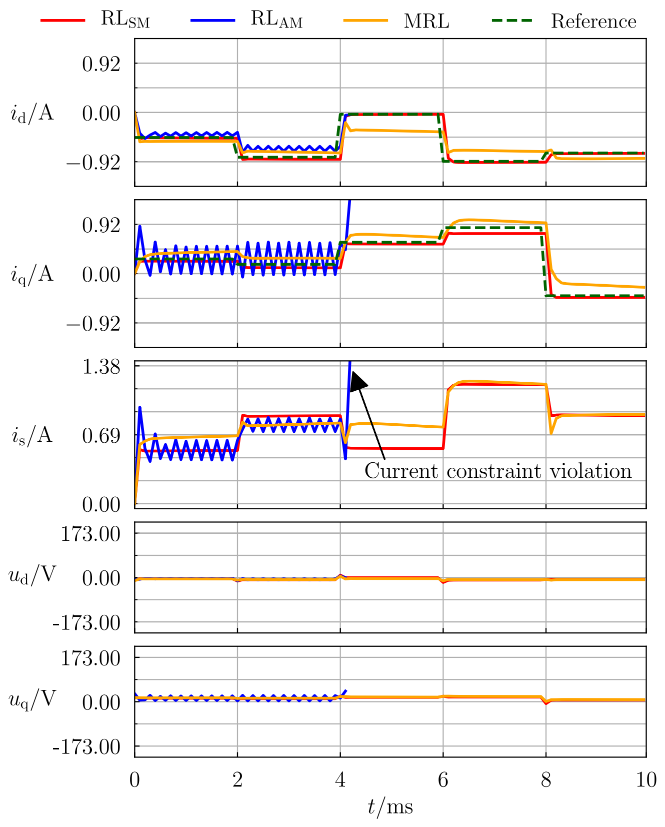

Fig. 7. Test episode on motor m_5 at $\omega_{\text{me}} = 0.2\,\omega_{\text{me,max}} = 125.7\ \text{s}^{-1}$.

TABLE VI
TEST ROUTINE RESULT STATISTICS

Parameter	$\mathrm{RL_{SM}}$	MRL	$\mathrm{RL_{AM}}$
Mean reward training	0.0904	0.0812	0.0632
Mean reward test	0.0922	0.0817	0.0709
Median reward training	0.0926	0.0838	0.0678
Median reward test	0.0939	0.0833	0.0746
Standard deviation training rewards	0.0077	0.0091	0.0168
Standard deviation test rewards	0.0062	0.0075	0.0130
Number of neural network parameters	3375450	24921	22503

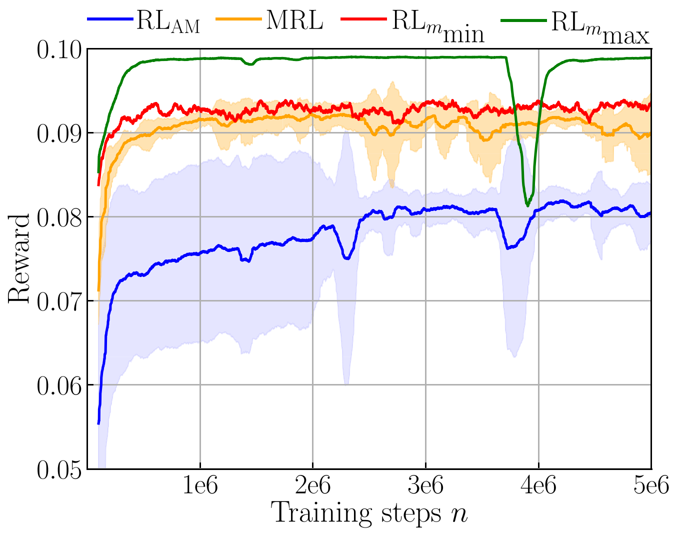

Fig. 8. Mean and standard deviation of MRL and $\mathrm{RL_{AM}}$ training rewards. Training rewards of $\mathrm{RL_{SM}}$ on best $m_{\max}$ and worst $m_{\min}$ performing motor.

all available motors with identical probability. Therefore, the MRL's control quality is more adapted to motor classes that are overrepresented in the training set. Further extensions are needed to avoid corresponding overfitting, which could be done by, e.g., including more sophisticated exploration schemes for selecting a motor environment during training, or via an improved minibatch sampling routine. In addition, expanding the training dataset with more motors that have uncommon characteristics would level out the distribution of featured motor dynamics, which should improve the final performance of the MRL agent.[4]

D. Evaluation of the Training

Fig. 8 shows the average training rewards of the last 10^6 steps and their standard deviation for the MRL and the $\mathrm{RL_{AM}}$ algorithms. The training rewards seem to be consistently better for the MRL approach compared to the $\mathrm{RL_{AM}}$, which was expected due to the information advantage implemented through the context. Also, the variance around the mean is smaller, indicating

[4]$\mathrm{RL_{AM}}$ and $\mathrm{RL_{SM}}$ use the same actor and critic architecture and, hence, the same number of ANN parameters. However, creating an individual RL agent per motor for $\mathrm{RL_{SM}}$ results in $22\,503 \times 150 = 3\,375\,450$ total network weights. The MRL's actor and critic structure is also identical to the $\mathrm{RL_{AM}}$ and requires only a few additional parameters for the context ANN.

TABLE VII
REQUIRED TRAINING TIME COMPARISON

MRL total training time	200.26 h
$\mathrm{RL_{SM}}$ per motor training time	16.34 h
$\mathrm{RL_{SM}}$ total training time	2451.00 h

TABLE VIII
EXEMPLARY SPMSM PARAMETERS FOR THE FAULT DETECTION INVESTIGATION

parameter	value
p	5
R_s	$1.396\,\Omega$
L_d	5.485 mH
L_q	5.485 mH
Ψ_p	120.390 mV·s
U_DC	658 V
I_n	6.03 A
ω_n	$314.159\,\mathrm{s}^{-1}$

higher performance reliability. In addition, the training rewards of the best performing and worst performing $\mathrm{RL_{SM}}$ result are shown. Both are consistently above the MRL's average reward. Table VI shows the statistical results of the tests where, again, the MRL controller shows a higher reward than the $\mathrm{RL_{AM}}$ approach, while both $\mathrm{RL_{SM}}$ algorithms are performing better. This is also an expected result, because the 150 $\mathrm{RL_{SM}}$ individually trained RL controllers together contain also 136 times more parameters than the MRL algorithm and have, therefore, much more learning capacity. Also note that the MRL controller has only slightly more network weights than the $\mathrm{RL_{AM}}$ approach, which are located within the context network. The complete data of all test results from the given 150 motors as well as all utilized code are available at [27] to supplement this contribution.

An insight of the measured training time requirements is listed in Table VII, whereas the training was conducted on a high-performance computing cluster [32]. As can be seen, the MRL's training time exceeds the $\mathrm{RL_{SM}}$'s expected training time by a factor of about 12.25. Therefore, the initially larger training effort for the MRL amortizes already after utilizing it within 13 different drive motors. Since there are nearly uncountable drive variants used in industrial applications, which would require an individual training with standard RL techniques (i.e., the training time scales linearly with the number of drive configurations considered), this training effort offset of the MRL algorithm pays off very quickly.

Also note that this training time analysis is based on a first proof of concept. Further optimization of the MRL, e.g., in terms of its hyperparameters or changes of the training routine, has the potential to decrease the necessary MRL training time, which

TABLE IX
IMPACT OF MOTOR FAULTS ON THE CREATED CONTEXT VARIABLES FOR AN EXEMPLARY SPMSM AS DEFINED IN TABLE VIII

case	deviation	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
Fault-free baseline		0.878	0.705	–0.871	–0.217	0.181	–0.238	0.230	0.129
Destroyed copper winding	$\tilde{R}_\text{s} = 1000 R_\text{s}$	–0.998	–1.000	–1.000	0.891	0.939	0.978	0.927	0.892
Destroyed winding isolation	$\{\tilde{R}_\text{s}, \tilde{L}_\text{d}, \tilde{L}_\text{q}\} = \frac{1}{10}\{R_\text{s}, L_\text{d}, L_\text{q}\}$	–0.118	–0.504	–0.147	0.143	0.156	–0.145	0.062	–0.097
Voltage drop in the dc link	$\tilde{U}_\text{DC} = \frac{1}{10} U_\text{DC}$	0.954	0.684	–0.917	0.534	0.163	–0.119	0.076	0.378
Permanent demagnetization	$\tilde{\Psi}_\text{p} = \frac{1}{10} \Psi_\text{p}$	0.898	0.714	–0.889	–0.113	0.151	–0.202	0.250	0.161

would make the time-saving potential even more worthwhile. In particular, the update rate of the context network and the size of the context buffer are what dramatically increased the MRL's training time. Both have not yet been investigated in terms of more time-efficient choices and are, therefore, an obvious degree of freedom that may enable a (drastic) reduction of training time.

E. Evaluation and Utilization of the Context Variables

Given the improvements of the MRL algorithm over the RL_AM, the information encoded within the context is of major interest. To fully characterize the dynamic behavior of the motor system, it is sufficient to learn either the physical parameters or the normalized ODE coefficients. For this, the evaluated MRL controller's context network was used to generate context variables z_1–z_8 for each of the 150 motors. These contexts were then analyzed concerning their correlation to the physical parameters and the normalized ODE coefficients. Fig. 9 shows the correlation matrix of the context, which indicates that the context variables correlate much stronger with the ODE coefficients than with the physical parameters.

Finally, the output of the context network has been investigated concerning different error cases that change the physical parameters of a PMSM. Since the parameter change will affect the resulting context vector, it is reasonable to utilize this chain of effects for detection of irregular drive conditions including faults. The analyzed error cases and the corresponding context vector have been investigated for one exemplary motor (specified in Table VIII) and are listed in Table IX. As can be seen, the context variables deviate from their original values for each of the error cases. Especially, for the first three experiments, this deviation is severe. Only for the last experiment, no obvious linear deviation trend from the original context is observable. However, it cannot be excluded that even this assumed fault is accessible from the context if nonlinear classifiers are considered for the detection.

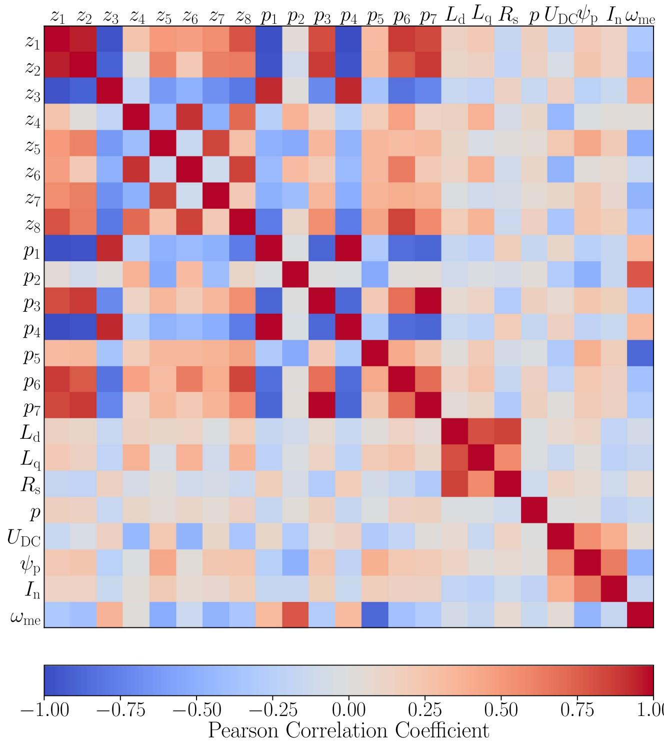

Fig. 9. Correlation matrix of context variables z_1–z_8 with physical and normalized space parameters.

VII. HARDWARE-IN-THE-LOOP (HIL) INVESTIGATION

Real-time capability is often a major concern when ML applications are to be utilized within or in conjunction with time-critical systems. In order to validate the feasibility of the proposed MRL architecture, an exemplary testcase is conducted on rapid control prototyping hardware (RCPH). This HIL experiment allows the monitoring of the utilized turnaround time T_TA, which is the time that is needed to compute the next control action. Hence, $T_\text{TA} < T_\text{s}$ must be fulfilled to guarantee controllability at all times. The corresponding HIL test setup is presented in Fig. 10 with a dSPACE MicroLabBox [33] posing as RCPH.

As visualized in Fig. 11, the found exemplary control performance during the HIL experiment is comparable with the offline simulations. It can also be seen that T_TA is strictly smaller than T_s. The corresponding HIL experiment only made use of the RCPH's CPU; the utilization of the built-in field-programmable gate array was not necessary. This verifies the viability of the MRL control scheme, as the resulting control agent turns out to be relatively lightweight concerning its computational burden.

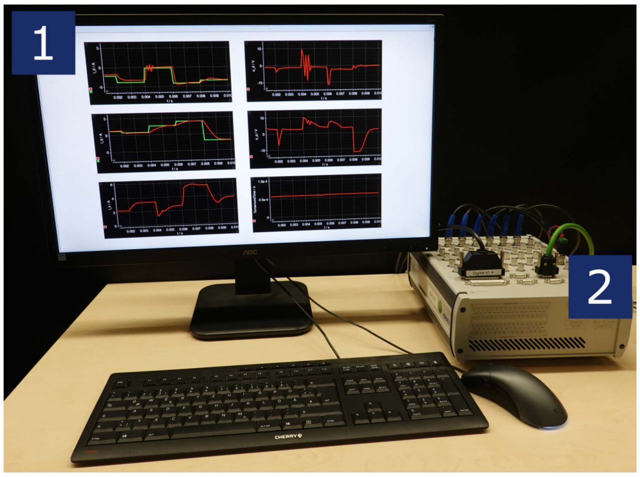

Fig. 10. HIL test setup to validate the real-time capability of the proposed MRL. 1) dSPACE Control Desk and 2) dSPACE MicroLabBox as RCPH.

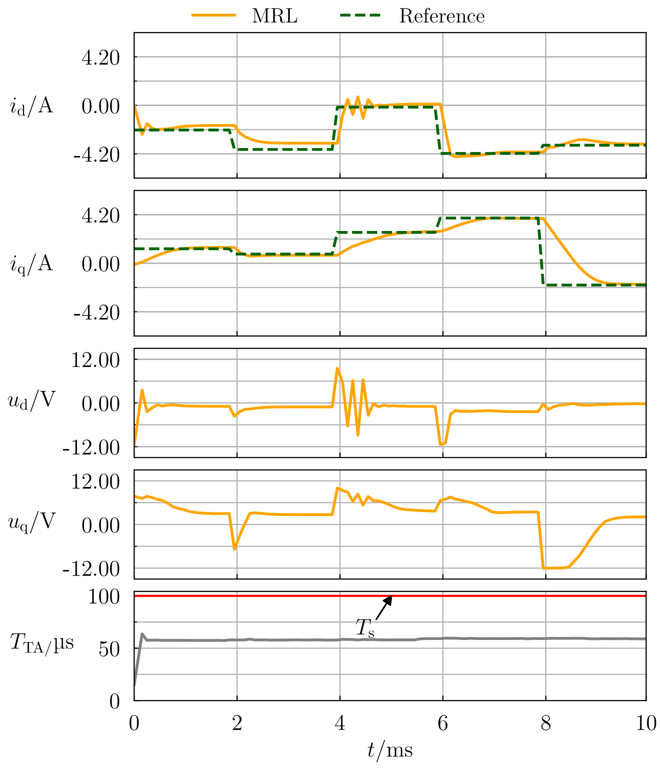

Fig. 11. Real-time HIL test of an exemplary control episode with the measurement of the turnaround time T_{TA} (same scenario as depicted in Fig. 5).

VIII. Conclusion

This article presented a drive control framework utilizing a context-based off-policy MRL approach. It was applied within the current control scenario on PMSMs from a wide range of different power classes and has shown to perform significantly better than a general RL algorithm without context variables. The overall procedure is promising for the time-efficient design of RL-based controllers, which is a major drawback of approaches with individual training. Furthermore, an HIL experiment also validated sufficiently quick inference of the presented MRL, despite the comprehensive training routine. The investigation of the learned context showed that the identified context variables correlate to the dynamic behavior of the motors as described by ODE coefficients. Therefore, the secondary goal of implicit motor behavior characterization was achieved, whose potential application to motor fault detection was also outlined with promising perspectives.

For upcoming investigations, closing the gap between MRL and individually trained RL should be prioritized. For that, a comprehensive optimization of the utilized hyperparameter configuration should be considered. In addition, the training routine should be extended to include methods of nonuniform motor and minibatch selection. Moreover, the level of sophistication in the preparation and selection of training data could be elaborated even further, especially in terms of synthetic data generation and in the representation of uncommon motor characteristics. Finally, this contribution assumed PMSMs that are described by static parameters, which is a simplification of the real-world behavior with parasitic effects such as (cross-)saturation or temperature and aging effects. These assumptions rendered a static context sufficient to fully characterize the environment. However, usual real-world applications may show dominant time-varying behavior, and therefore, it would be of interest to recreate the context at runtime to adapt to changes of the drive in an online fashion. Naturally, the utilization of the given setup should also be considered for different power electronic applications.

References

[1] R. Gabriel, W. Leonhard, and C. J. Nordby, “Field-oriented control of a standard AC motor using microprocessors,” *IEEE Trans. Ind. Appl.*, vol. IA-16, no. 2, pp. 186–192, Mar. 1980.
[2] I. Takahashi and T. Noguchi, “A new quick-response and high-efficiency control strategy of an induction motor,” *IEEE Trans. Ind. Appl.*, vol. IA-22, no. 5, pp. 820–827, Sep. 1986.
[3] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel, “Model predictive control of power electronic systems: Methods, results, and challenges,” *IEEE Open J. Ind. Appl.*, vol. 1, pp. 95–114, 2020.
[4] P. Karamanakos and T. Geyer, “Guidelines for the design of finite control set model predictive controllers,” *IEEE Trans. Power Electron.*, vol. 35, no. 7, pp. 7434–7450, Jul. 2020.
[5] M. Schenke, W. Kirchgässner, and O. Wallscheid, “Controller design for electrical drives by deep reinforcement learning: A proof of concept,” *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4650–4658, Jul. 2020.
[6] G. Book et al., “Transferring online reinforcement learning for electric motor control from simulation to real-world experiments,” *IEEE Open J. Power Electron.*, vol. 2, pp. 187–201, 2021.
[7] M. Schenke and O. Wallscheid, “A deep q-learning direct torque controller for permanent magnet synchronous motors,” *IEEE Open J. Ind. Electron. Soc.*, vol. 2, pp. 388–400, 2021.
[8] C. Cui, T. Yang, Y. Dai, C. Zhang, and Q. Xu, “Implementation of transferring reinforcement learning for DC-DC buck converter control via duty ratio mapping,” *IEEE Trans. Ind. Electron.*, vol. 70, no. 6, pp. 6141–6150, Jun. 2023.
[9] S. Kwon, C. Yoon, and Y.-I. Lee, “Practical implementation method of reinforcement learning for power converter,” *IFAC-PapersOnLine*, vol. 55, no. 9, pp. 437–441, 2022.
[10] B. Huangfu, C. Cui, C. Zhang, and L. Xu, “Learning-based optimal large-signal stabilization for DC/DC boost converters feeding CPL via deep reinforcement learning,” *IEEE J. Emerg. Sel. Topics Power Electron.*, to be published, doi: 10.1109/JESTPE.2022.3189078.

[11] Y. Chen, J. Bai, and Y. Kang, "A non-isolated single-inductor multiport DC-DC topology deduction method based on reinforcement learning," *IEEE Trans. Emerg. Sel. Topics Power Electron.*, vol. 10, no. 6, pp. 6572–6585, Dec. 2021.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[13] J. Vanschoren, "Meta-learning," in *Automated Machine Learning* (The Springer Series on Challenges in Machine Learning). F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., Cham, Switzerland: Springer, 2018, doi: 10.1007/978-3-030-05318-5_2.

[14] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, "Meta-reinforcement learning for robotic industrial insertion tasks," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2020, pp. 9728–9735.

[15] P. Balakrishna, G. Book, W. Kirchgässner, M. Schenke, A. Traue, and O. Wallscheid, "Gym-electric-motor (GEM): A Python toolbox for the simulation of electric drive systems," *J. Open Source Softw.*, vol. 6, no. 58, 2021, Art. no. 2498.

[16] R. De Doncker, D. Pulle, and A. Veltman, *Advanced Electrical Drives: Analysis, Modeling, Control*. Dordrecht, The Netherlands: Springer, 2010.

[17] E. Clarke, *Circuit Analysis of A-C Power System*, vol. I. Hoboken, NJ, USA: Wiley, 1943.

[18] R. H. Park, "Two-reaction theory of synchronous machines generalized method of analysis—Part I," *Trans. Amer. Inst. Elect. Eng.*, vol. 48, no. 3, pp. 716–727, 1929.

[19] D. Bertsekas, *Reinforcement Learning and Optimal Control*. Nashua, NH, USA: Athena Sci., 2019.

[20] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," 2016, *arXiv:1611.02779*.

[21] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, vol. 80, pp. 1587–1596.

[22] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 5331–5340.

[23] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola, "Meta-Q-learning," 2019, *arXiv:1910.00125*.

[24] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2019, *arXiv:1509.02971*.

[25] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, M. F. Balcan and K. Q. Weinberger, Eds. New York, NY, USA, Jun. 20–22, 2016, vol. 48, pp. 1928–1937. [Online]. Available: https://proceedings.mlr.press/v48/mniha16.html

[26] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[27] D. Jakobeit, "Meta-RL PMSM drive database set," 2022. [Online]. Available: https://github.com/upb-lea/meta_RL_PMSM/tree/main/MotorDB

[28] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[29] M. Schenke and O. Wallscheid, "Improved exploring starts by kernel density estimation-based state-space coverage acceleration in reinforcement learning," 2021, *arXiv:2105.08990*.

[30] L. Prechelt, "Early stopping-but when?," in *Neural Networks: Tricks of the Trade* (Series Lecture Notes in Computer Science), vol. 1524, G. B. Orr and K.-R. Müller Eds. Berlin, Germany: Springer1996, pp. 55–69.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[32] Paderborn Center for Parallel Computing, Accessed: Mar. 21, 2023. [Online]. Available: https://pc2.uni-paderborn.de/

[33] dSPACE, Accessed: Mar. 21, 2023. [Online]. Available: https://www.dspace.com/en/pub/home/products/hw/microlabbox.cfm

Darius Jakobeit received the bachelor's and master's degrees in computer engineering from Paderborn University, Paderborn, Germany, in 2019 and 2021, respectively.

Since 2022, he has been a Research Assistant with the Department of Power Electronics and Electrical Drives, Paderborn University. His research interests include the application of (meta-)reinforcement learning in the domain of drive control and temperature estimation in electric power systems using supervised machine learning.

Maximilian Schenke received the bachelor's and master's (Hons.) degrees in electrical engineering in 2017 and 2019, respectively, from Paderborn University, Paderborn, Germany, where he is currently working toward the Doctoral degree in electrical engineering with the Department of Power Electronics and Electrical Drives.

His research interests include the application of reinforcement learning methods in the domain of drive control.

Oliver Wallscheid (Member, IEEE) received the bachelor's and master's (Hons.) degrees in industrial engineering in 2010 and 2012, respectively, and the Doctorate (Hons.) degree in electrical engineering in 2017 all from Paderborn University, Paderborn, Germany.

Since 2017, he has been a Senior Research Fellow with the Department of Power Electronics and Electrical Drives, Paderborn University. He is also an Acting Professor with the Department of Automatic Control, Paderborn University. His research interests include data-driven identification and intelligent control of electrical power systems in decentralized grids, power electronics, and drives.