Optimal Weighting Factors Design for Model Predictive Current Controller for Enhanced Dynamic Performance of PMSM Employing Deep Reinforcement Learning

Muhammad Usama, Amine Salaje, Thomas Chevet, Nicolas Langlois

applied sciences

MDPI

Article

Optimal Weighting Factors Design for Model Predictive Current Controller for Enhanced Dynamic Performance of PMSM Employing Deep Reinforcement Learning

Muhammad Usama *, Amine Salaje, Thomas Chevet and Nicolas Langlois

ESIGELEC, IRSEEM, Université de Rouen Normandie, 76000 Rouen, France;
amine.salaje@groupe-esigelec.org (A.S.); thomas.chevet@esigelec.fr (T.C.); nicolas.langlois@esigelec.fr (N.L.)
* Correspondence: muhammad.usama@esigelec.fr

Abstract: This paper presents a novel control strategy employing a deep reinforcement learning (DRL) scheme for online selection of optimal weighting factors in cost functions of the finite control set model predictive current controller of a permanent magnet synchronous motor (PMSM). Indeed, when designing predictive controllers for PMSMs' phase currents, competing objectives appear, such as managing current convergence and switching transitions. These objectives result in an asymmetric cost function where they have to be balanced through weighting factors in order to enhance the inverter and motor performance. Leveraging the twin delayed deep deterministic policy gradient algorithm, the optimal weighting factor selection policy is obtained for online balancing of the choice between current deviation in the *dq* frame and inverter commutations. For comparison, a metaheuristic-based artificial neural network is trained on static data obtained through a multi-objective genetic algorithm to predict the weights. The key performance markers, such as torque ripple, total harmonic distortion, switching frequency, steady-state, and dynamic performance, are provided through numerical simulations to verify the effectiveness of the proposed tuning scheme. The results of these simulations confirm that the proposed dynamic control scheme effectively resolves the challenges of weighting factor choice, meeting inverter performance requirements, and delivering better dynamic and steady-state performance.

Keywords: permanent magnet synchronous motor; inverter performance; model predictive current control; asymmetric cost function; deep reinforcement learning; multi-objective optimization

Academic Editors: Stelios Ioannou, Mohamed Darwish and Nicholas G. Christofides

Received: 9 April 2025
Revised: 16 May 2025
Accepted: 21 May 2025
Published: 23 May 2025

Citation: Usama, M.; Salaje, A.; Chevet, T.; Langlois, N. Optimal Weighting Factors Design for Model Predictive Current Controller for Enhanced Dynamic Performance of PMSM Employing Deep Reinforcement Learning. *Appl. Sci.* **2025**, *15*, 5874. https://doi.org/10.3390/app15115874

1. Introduction

With growing concerns over the greenhouse effect, the energy efficiency of industrial plants has become a critical priority. The automotive and aerospace industries are thus shifting toward greater efficiency, increasingly replacing hydraulic actuators with permanent magnet synchronous motors (PMSMs), which offer better energy performance [1,2]. Indeed, PMSMs are widely preferred in industrial applications for their compact design, high power density, exceptional efficiency, low inertia, and elimination of field windings.

The predominant control methods for surface PMSM (SPMSM) drive systems are vector control and direct torque control (DTC) [3]. These methods often include a linear proportional–integral (PI) controller in the control loop. Such a controller has limitations, particularly in applications that require minimal overshoot and high dynamic performance. Traditional DTC also suffers from significant torque fluctuations at low speeds [4,5]. As

a result, researchers are actively seeking advanced control strategies to enhance the performance of the PMSM drive system. Among other advanced control strategies, model predictive control (MPC) has gained much attention in the control of power electronics and motor drives due to its ability to handle multivariable problems with constraints and nonlinearities [6]. MPC has been extensively used in various industrial applications, including wind power generation systems, computer numerical control machine drive systems, and high-speed rail traction drive systems [7–9].

As a result, the last decade has seen the study of various types of MPC for drive applications based on fixed or randomly distributed switching frequencies across converters. These predictive strategies can be classified into two categories, namely continuous control set (CCS) MPC and finite control set (FCS) MPC. In CCS-MPC, a constant switching frequency can be achieved through the modulation stage. However, the prediction process involved is complex [10]. On the other hand, FCS-MPC has become more popular among other predictive control strategies due to its intuitive nature and straightforward implementation. Indeed, it optimizes the switching states directly without needing an external modulation stage, resulting in a randomly distributed switching frequency [11]. Various approaches have been proposed to achieve a constant switching frequency with FCS-MPC in the literature [12–14]. Furthermore, a data-driven predictive controller has been employed to achieve gating without requiring the explicit model of the plant, helping to improve robustness and transient response [15].

Although FCS-MPC offers impressive advantages, such as high-performance multi-objective control with inherent constraints, it faces several challenges. In addition, this control strategy can lead to variable switching frequency in the inverter output, which is often undesirable, as it can cause the risk of triggering an unexpected resonance and affect the power efficiency of the inverter [16]. Then, the central element of the FCS-MPC strategy is its cost function, essential to balance various control objectives, including tracking error, control effort, total harmonic distortion (THD), switching frequency, and torque ripple, all under state constraints. All or part of these conflicting objectives can appear simultaneously in an asymmetric cost function, each weighted by a different factor, balancing their relative importance. One of the most critical issues is therefore the choice of optimal weights within the asymmetric cost function, which can significantly impact the overall effectiveness of the controller [17]. It is important to note that the weighting factors within the cost function directly impact the control performance of PMSM drives, particularly regarding total harmonic distortion (THD) and switching frequency, as they manage the trade-offs between the different objectives. In addition, current convergence ensures accurate tracking of the reference current, which directly affects torque precision and dynamic response. However, achieving fast convergence often requires fast inverter switching, which leads to increased switching frequency, power losses, and thermal stress on the semiconductor devices (IGBT, MOSFET). On the other hand, minimizing switching transitions improves inverter efficiency, reduces electromagnetic interference, and enhances the reliability and lifespan of power electronics, but may compromise current tracking performance and introduce torque ripples.

Incorrectly set weighting factors can lead to misalignment in prioritizing control targets, ultimately reducing control effectiveness [18]. It is consequently important to choose appropriate weighting factors in the FCS-MPC's cost function, reflecting the relative importance of each control objective, as the accuracy of these factors significantly influences the overall performance of the control strategy. The online calculation of the weights in MPC's cost function is highly challenging due to the complexity and coupling between multiple control objectives [19]. Furthermore, online solutions often use observer-based approaches to dynamically optimize the weighting coefficients, but these methods significantly increase

the computational burden on MPC-based systems [20,21]. To address this, researchers have proposed alternative methods, such as using a symmetric cost function, where each term has equal priority and focuses solely on tracking errors [17]. Another approach is the use of sequential model predictive control (SMPC), which decomposes multiple control objectives into separate sub-cost functions, evaluated sequentially [22]. This method simplifies the control structure, reduces computational load, and offers flexibility. However, SMPC performance is dependent on the execution order of the sub-cost functions, making it difficult to guarantee optimal performance and preventing simultaneous optimization of all targets [23]. Furthermore, offline tuning schemes have been proposed in the literature [24]. While these schemes are straightforward, they are time-consuming and computationally expensive. Additionally, using a fixed weighting factor does not ensure optimal performance across various operating conditions.

Recently, data-driven strategies have become more popular in the field of power conversion applications. It has been applied successfully in power electronic systems for model-free control approaches, particularly with the use of an artificial neural network (ANN)-based controller and estimator for the identification and control of power electronics and motor drives [25,26]. Furthermore, data-driven schemes are used for the design of weighting factors [27]. Intelligent approaches for optimizing cost functions include heuristic methods and fuzzy logic, which often increase computational complexity. Heuristic approaches such as genetic algorithms (GAs) have been applied to optimize weighting factors in MPC, but they significantly increase computational complexity due to their iterative nature [28]. Similarly, fuzzy decision-making models, which replace the cost function to bypass weighting factor selection, are mainly suitable for online optimization and add complexity, particularly in multilevel inverters. Although both methods can improve accuracy, they impose a high computational burden on MPC systems [29]. However, data-driven designs based on ANN give a significant advantage due to their strong capabilities in nonlinear regression, prediction, and classification [30,31]. Data-driven design reduces computational complexity, as the training and testing of the model is done offline, and when integrated online, it dynamically updates the weighting factors of the cost function to enhance the performance of the MPC, which is better than fixed weighting factors in the cost function.

This paper then presents a novel method for online optimization of weighting factors in the asymmetric cost function of an FCS model predictive current controller (MPCC), targeting reduced computational complexity and enhanced dynamic performance. The approach leverages a twin delayed deep deterministic policy gradient (TD3) [32] reinforcement learning (RL) agent to formulate an optimal policy for weighting factor design, with plant dynamics and inverter nonlinearities modeled within the control environment. Through agent–environment interaction, the TD3 agent adaptively optimizes the weighting factors by maximizing a reward function, effectively addressing multi-objective control requirements. Furthermore, to the authors' knowledge, this study represents the first effort to establish a dynamic, data-driven tuning framework using deep reinforcement learning (DRL) for an asymmetric cost function. The proposed tuning scheme aims to optimize control objectives by balancing multiple competing criteria and enhancing control system performance.

For comparative analysis, we propose a multiobjective GA (MOGA)-based neural network scheme that is employed to optimize the weighting factors. MOGA constructs a Pareto front of optimal weights [33] from which a dataset under varied operating conditions is established. These datasets are used as training data for a feedforward neural network (NN). This network provides a baseline data-driven approach for online tuning of the weighting factors across variable conditions. Test cases using a DRL-based dynamic tuning

scheme and an NN-based static tuning approach are presented to validate the proposed method, demonstrating an improved weighting factor design in MPC for voltage source inverter (VSI)-fed PMSMs.

The remainder of the paper is structured as follows. Section 2 presents the mathematical model of the considered electric drives. In addition, the general formulation of FCS-MPCC with an asymmetric cost function is presented. Then, Section 3 delves into a detailed explanation of the static and dynamic data-based training methods for the weighting factors' optimization strategies. Section 4 is dedicated to the evaluation of the effectiveness of the reinforcement learning approach, where we compare its performance under various test conditions with the baseline MOGA-NN-based method for the online selection of weighting factors in the asymmetric cost function. Finally, the significant findings of the proposed DRL online tuning scheme are summarized in Section 5.

2. Mathematical Modeling

The mathematical model of the considered plant and its control strategy are established in the following.

2.1. Modeling of a Three-Phase Two-Level Voltage Source Inverter

A three-phase VSI consisting of six switches and a DC rail, as presented in Figure 1, is commonly used in industrial applications.

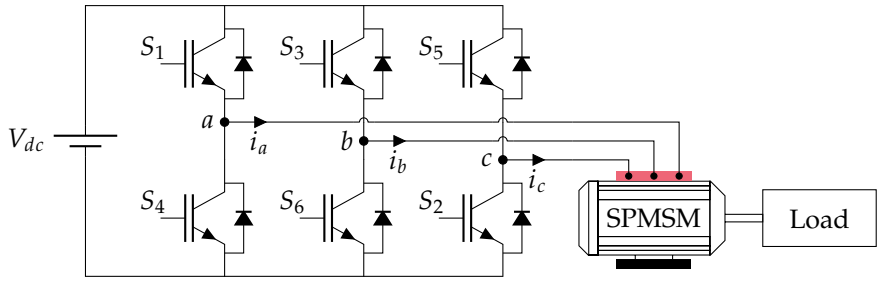


Figure 1. The schematic representation of VSI-fed SPMSM.

A modulation scheme modulates the DC input voltage V_{dc} from the DC link to generate a three-phase AC output for driving electric machines by providing appropriate sequences of gate signals to the transistors. To prevent shoot-through, the gate signals for the upper and lower transistors in each inverter leg are complementary, and their transitions are managed with dead time intervals to avoid shoot-through phenomena; see Chapter 10 in [1]. The instantaneous stator voltage of the three-phase inverter is expressed as

$$\begin{bmatrix} v_a(t) \\ v_b(t) \\ v_c(t) \end{bmatrix} = \frac{1}{3} V_{dc} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} S_1(t) \\ S_3(t) \\ S_5(t) \end{bmatrix}. \tag{1}$$

Here, $S_1(t)$, $S_3(t)$, and $S_5(t)$ represent the switching states of the upper transistors of the three inverter legs at time t, such that $S_i(t) = 1$, with $i \in \{1, 3, 5\}$, if the transistor i is closed and 0 otherwise. The output voltage vectors of a two-level VSI in the abc coordinate frame are given in Table 1. For ease of representation and analysis, the time dependency in the equations is omitted in further discussions.

To simplify the control process, three-phase voltages are transformed into two-phase components in both stationary ($\alpha\beta$) and synchronous rotating (dq) reference frames using Clarke and Park transformations, respectively. These transformations are respectively expressed as

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix}\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix}, \tag{2}$$

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & \sin(\theta_r) \\ -\sin(\theta_r) & \cos(\theta_r) \end{bmatrix}\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} \tag{3}$$

where θ_r is the angle between the $\alpha\beta$ and dq frames, i.e., in the case of a motor, between the stator and the rotor. The possible switching configurations for a two-level VSI in the dq frame are shown in Figure 2. In this figure, the V_i, with $i \in \{0, \ldots, 7\}$, correspond to the projection of the voltage vectors of Table 1 on the dq axes.

Table 1. Output voltage vectors of a two-level three-phase inverter.

$S_1S_3S_5$	v_a	v_b	v_c
000	0	0	0
100	$2V_{dc}/3$	$-V_{dc}/3$	$-V_{dc}/3$
110	$V_{dc}/3$	$V_{dc}/3$	$-2V_{dc}/3$
010	$-V_{dc}/3$	$2V_{dc}/3$	$-V_{dc}/3$
011	$-2V_{dc}/3$	$V_{dc}/3$	$V_{dc}/3$
001	$-V_{dc}/3$	$-V_{dc}/3$	$2V_{dc}/3$
101	$V_{dc}/3$	$-2V_{dc}/3$	$V_{dc}/3$
111	0	0	0

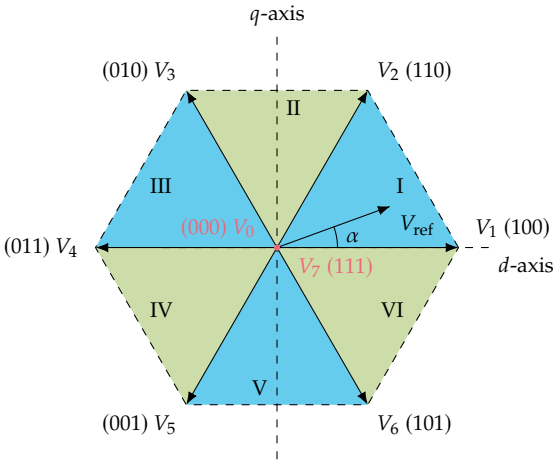


Figure 2. Inverter possible switching configuration.

2.2. Modeling of Surface Permanent Magnet Synchronous Motor

The mathematical modeling of the three-phase SPMSM is considered in the rotating reference frame. Due to the non-salient characteristic of the motor drive, we consider that the d- and q-axis inductances L_d and L_q of the motor satisfy $L_d = L_q = L_s$. To ensure model simplicity and real-time implementation feasibility, high-frequency effects such as magnetic saturation, core losses, and temperature variations were neglected. The direct-axis and quadrature-axis currents, i_d and i_q, respectively, are obtained from the following equations, as given in Chapter 6 in [1],

$$\frac{di_d}{dt} = -\frac{r_s}{L_s}i_d + \omega_r i_q + \frac{1}{L_s}v_d, \tag{4a}$$

$$\frac{di_q}{dt} = -\frac{r_s}{L_s}i_q - \omega_r i_d + \frac{1}{L_s}v_q - \frac{\psi_m}{L_s}\omega_r \tag{4b}$$

where v_d and v_q are the voltage components in the dq frame, r_s is the stator resistance, ω_r is the electrical angular velocity, and ψ_m is the magnetic flux due to permanent magnets.

The voltages v_d and v_q fed to the SPMSM are obtained through the VSI described in the previous section, as shown in Figure 1.

For a given sampling period T_s, we use the explicit forward Euler discretization approach [34] to derive a discrete-time prediction model of the dq frame currents as

$$i_d(k+1) = i_d(k) + T_s\left(-\frac{r_s}{L_s}i_d(k) + \omega_r(k)i_q(k) + \frac{1}{L_s}v_d(k)\right), \tag{5a}$$

$$i_q(k+1) = i_q(k) + T_s\left(-\frac{r_s}{L_s}i_q(k) - \omega_r(k)i_d(k) + \frac{1}{L_s}v_q(k) - \frac{\psi_m}{L_s}\omega_r(k)\right). \tag{5b}$$

This equation describes the updated current values $i_d(k+1)$ and $i_q(k+1)$ at the next time step $k+1$ in terms of the different signal values and system parameters at the current time step k.

The motor speed is considered to remain constant over several time steps, as the electromechanical time constant of the motor drive is significantly smaller than the mechanical time constant [35]. This leads to the approximation $\omega_r(k+1) \approx \omega_r(k)$. The electromagnetic torque T_e produced by the SPMSM is

$$T_e(k) = \frac{3}{2}p\psi_m i_q(k) \tag{6}$$

where p is the number of pole pairs of the motor. This model provides a fundamental understanding of the motor's operation employed in this paper, enabling effective design and analysis of control systems for SPMSMs.

2.3. Decision Function

The FCS-MPCC control strategy relies on the principle that a static power converter can only produce a finite set of switching states, each impacting the system variables differently. This is the case for the VSI described above, where we have eight possible switching states and the associated values of the voltages v_d and v_q. Using the control plant model (5), we can predict the behavior of the currents i_d and i_q in each of these switching states.

The selection of the optimal switching state is then based on a selection criterion, specifically a cost function. In classical FCS-MPCC, the cost function is evaluated at each time step for all possible switching states, and the optimal state is selected as the one yielding the minimal cost function. The same is done in this paper, with the cost function being composed of three elements.

The first element J_r of the cost function aims to minimize the deviation of the currents i_d and i_q from their respective references i_d^* and i_q^*. This deviation is evaluated over a given prediction horizon N by considering

$$J_r = (i_d^* - i_d(k+N|k))^2 + (i_q^* - i_q(k+N|k))^2 \tag{7}$$

where $i_d(k+N|k)$ and $i_q(k+N|k)$ are the predictions of i_d and i_q obtained by applying N times the dynamical model (5) from the knowledge of $i_d(k)$ and $i_q(k)$ for a given choice of voltage vectors $v_d(k+i)$ and $v_q(k+i)$, with $i \in \{0, \ldots, N-1\}$, assuming ω_r remains constant and equal to $\omega_r(k)$. In the following, we consider $N = 2$.

Remark 1. *In this paper, we consider, for current prediction, that v_d and v_q remain constant over the prediction horizon in order to minimize the number of cost function evaluations made at each time instant. Indeed, as seen in Table 1, there are eight possible choices for the pair (v_d, v_q) at each time step, leading to $8^N = 64$ (since $N = 2$ in this paper) possible combinations to test over the prediction horizon. It follows that when evaluating J_r, we only choose voltages $v_d(k)$ and $v_q(k)$ while $v_d(k+1) = v_d(k)$ and $v_q(k+1) = v_q(k)$.*

Due to the lack of a modulator in the control design, the switching frequency is variable, leading to increased commutation between the consecutive sampling periods. As a result, this causes switching losses and reduces the converter's efficiency. To limit this issue, a second element

$$J_s = \sum_{i=0}^{N-1} \underbrace{\sum_{j=1,3,5} \left| S_j(k+i+1) - S_j(k+i) \right|}_{\Xi(k+i+1)} = \sum_{i=0}^{N-1} \Xi(k+i+1), \tag{8}$$

aiming to reduce the switching frequency is added to the decision function to reduce the commutation burden on the transistors at each sampling time. We introduce in (8) the quantity $\Xi(k)$, corresponding to the number of legs of the VSI where a commutation occurred between two time instants $k-1$ and k.

Remark 2. *Following Remark 1, we have, in this paper,* $\Xi(k+i) = 0$ *for any* $i \in \{1, \ldots, N-1\}$.

Finally, a third term

$$J_m = \hat{f}\left(i_d(k+N|k), i_q(k+N|k)\right), \tag{9}$$

playing the role of a constraint on the maximum current that can pass through the SPMSM is added to the decision function. Indeed, the function $\hat{f}$ is defined as

$$\hat{f}(x,y) = \begin{cases} +\infty & \text{if } |x| > i_{\max} \text{ or } |y| > i_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

adding an infinite penalty to the cost in case the predicted currents get higher than the rated phase current.

Finally, the total cost function has three objectives: reducing the deviation of the current from the reference, reducing the total amount of switches between transistor states, and limiting the maximum current through the SPMSM, and is expressed as

$$J = \lambda_i J_r + \lambda_f J_s + J_m \tag{11}$$

where λ_i and λ_f are weighting factors of current tracking and switching frequency regulation, respectively.

We see that the cost function (11) is asymmetric. Indeed, reducing the switching frequency affects the current tracking error J_r, as the chosen switching states influence the voltage sent to the motor. On the other hand, getting a better current tracking has an impact J_s, as it imposes switching states and, as a consequence, a switching frequency on the VSI. To overcome this complexity, this paper presents an optimization algorithm using a multi-objective approach as well as a data-driven approach employing static and dynamic data to find the balance between current tracking error and variable frequency reduction, which gives overall enhanced drive performance.

3. Proposed Intelligent Weighting Factors Design Approach

Based on what precedes, Figure 3 illustrates the proposed control system structure. The FCS-MPCC block encapsulates the minimization strategy described in Section 2.3, providing the appropriate gating pulses to the VSI. Indeed, it receives the values $i_d(k)$, $i_q(k)$, $\omega_r(k)$, and the gate pulses at time $k-1$ in order to predict the different $i_d(k+2 \mid k)$ and $i_q(k+2 \mid k)$ for the eight possible values of $v_d(k) = v_d(k+1)$ and $v_q(k) = v_q(k+1)$

of Figure 2 and the associated gate pulses. The goal of this section is then to describe a DRL approach to determine the optimal weighting factors λ_i and λ_f of the asymmetric decision function (11). Reinforcement learning effectively manages a dynamic set of data online to update the weights of a neural network, aiding both transient and steady-state responses of the control system. To assess the efficiency of the proposed DRL approach, we propose running a multiobjective genetic algorithm to generate static data [33] to tune a neural network providing the weighting factors as a baseline for comparison with the proposed method.

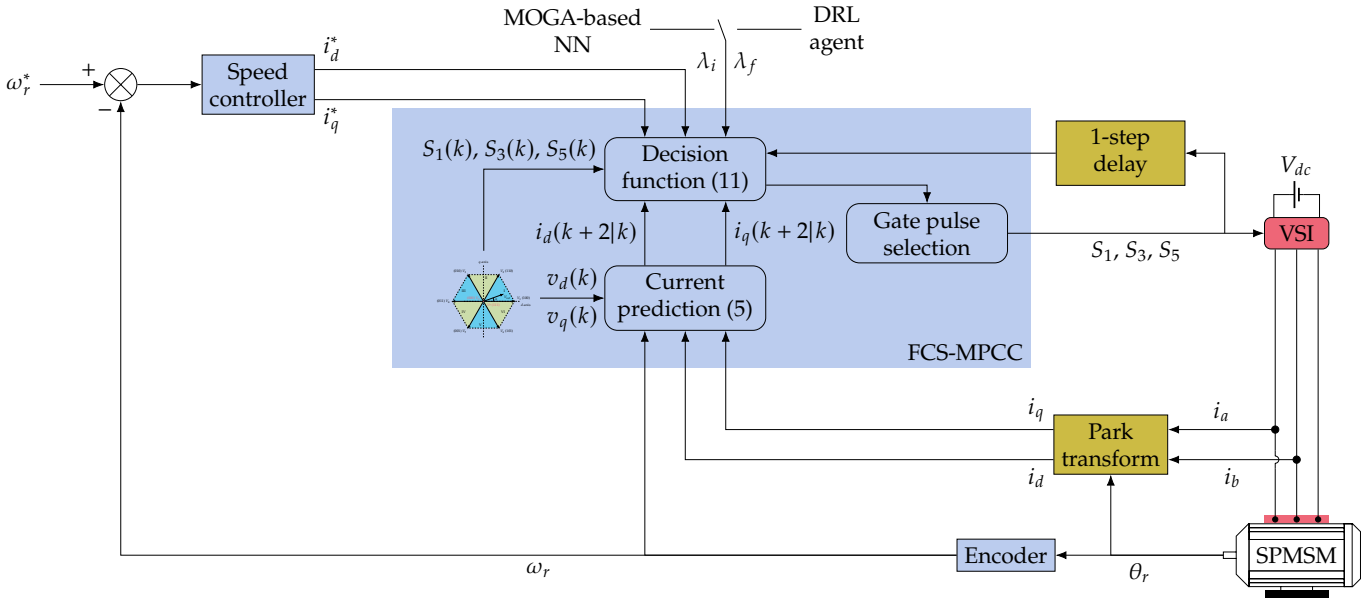


Figure 3. FCS-MPCC-based control design of SPMSM.

3.1. MOGA-Based NN Weighting Factor Tuning Approach

As explained, the DRL-based strategy has to be compared to a baseline method to assess its efficiency. The baseline considered in this paper is the neural network presented in Figure 4. This network has two hidden layers, the first one having 15 neurons using the poslin activation function and the second one having 10 neurons using the tansig activation function. The inputs of the network are the q-axis current error $\Delta i_q(k) = i_q^*(k) - i_q(k)$ at time instant k, the integral of this error, and the number of commutations $\Xi(k)$ between time instants $k-1$ and k. The outputs are the weighting factors λ_i and λ_f.

Remark 3. *To select the best set of signals as the neural network inputs, several combinations were tried. For each set of input signals, the performance of the control strategy of Figure 3 was evaluated using the neural network resulting from the supervised learning procedure described below. For test cases, the integral square error (ISE), integral time-weighted square error (ITSE), integral absolute error (IAE), and integral time-weighted absolute error (ITAE) of the errors Δi_q and $\Delta i_d(k) = i_d^*(k) - i_d(k)$ were evaluated. As can be seen on Table 2 on one example of input signals, the set of inputs we selected always performed better, hence its selection.*

To train this neural network, an optimal value of the weighting factors is first obtained using a MOGA implemented through the Matlab 2024a function `gamultiobj`. This algorithm uses a population of N_{MOGA} individuals. To each individual is assigned a pair of weighting factors (λ_i, λ_f) drawn randomly such that $\lambda_i \in [\underline{\lambda}_i, \overline{\lambda}_i]$ and $\lambda_f \in [\underline{\lambda}_f, \overline{\lambda}_f]$. For each individual, a simulation using the structure Figure 3 is run by fixing the weighting factors to those of the individual and using the parameters Table 3 for the different elements

in the structure. The speed controller is a PI controller whose parameters are also given in Table 3. The best individuals are selected based on two objectives:

- The total number of transitions over the simulation, i.e., the sum of the $\Xi(k)$;
- The ITAE of the q-axis current $\sum_{k=0}^{N} k|\Delta i_q(k)|$ where N is the total number of time steps in the simulation.

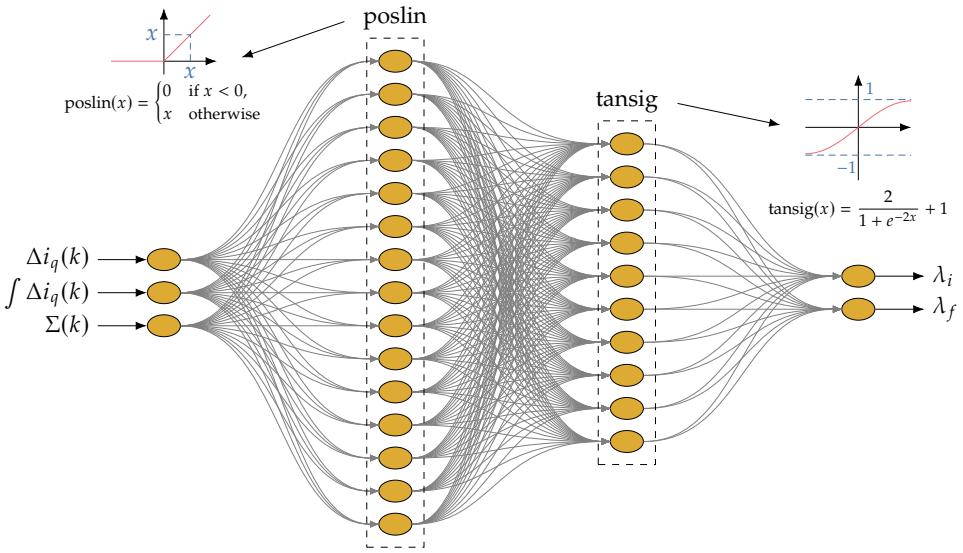


Figure 4. Neural network architecture.

Table 2. Performance metrics for neural network input features.

Input Features	Δi_q Error Metrics				Δi_d Error Metrics			
	ISE	**ITSE**	**IAE**	**ITAE**	**ISE**	**ITSE**	**IAE**	**ITAE**
$[\Delta i_q(k), \int \Delta i_q(k), \Xi(k), S_1(k-1), S_3(k-1), S_5(k-1)]$	0.08091	0.00500	0.02630	0.02046	0.05033	0.00207	0.00775	0.01292
$[\Delta i_q(k), \int \Delta i_q(k), \Xi(k)]$	0.068 27	0.00289	0.02044	0.01581	0.03613	0.00083	0.00395	0.00836

Based on the obtained results, a new generation of individuals is generated using the default methods of the `gamultiobj` function. This process is then repeated over a maximum of N_{gen} generations.

Table 3. Simulation model parameters.

Parameter	Description	Value
T_s	Sampling period	1 µs
f_{sw}^*	Switching frequency reference	10 kHz
r_s	Stator resistance	85 mΩ
L_s	Stator inductance	1.2 mH
ψ_m	Flux linkage	0.175 Wb
K_t	Torque constant	76.71 mNm · A^{-1}
p	Pole pairs	4
J	Rotor inertia	21.5 kg · cm^2
K_p	Speed proportional gain	45.4168
K_i	Speed integral gain	0.967

Through the use of MOGA, a Pareto front of weighting factors balancing the two objectives is obtained. The structure of Figure 3 is then simulated under different conditions using different pairs from the Pareto front as weighting factors and the parameters of Table 3. The different conditions include varied steady-state and transient conditions, load

variations, external disturbances, noise, parameter variations, and sampling time changes. The results of these simulations are gathered to constitute a comprehensive dataset used to train the neural network of Figure 4. The variety of simulation conditions helps enhance the network's overall robustness and generalization capabilities by simulating varied realistic operating environments.

The neural network is then trained using a supervised learning approach with the dataset. The performance of the neural network is evaluated over its training phase by using the mean squared error of the predicted values with respect to the ones from the dataset.

The flowchart presented in Figure 5 summarizes the procedure described in this section for the training of a neural network based on the use of MOGA.

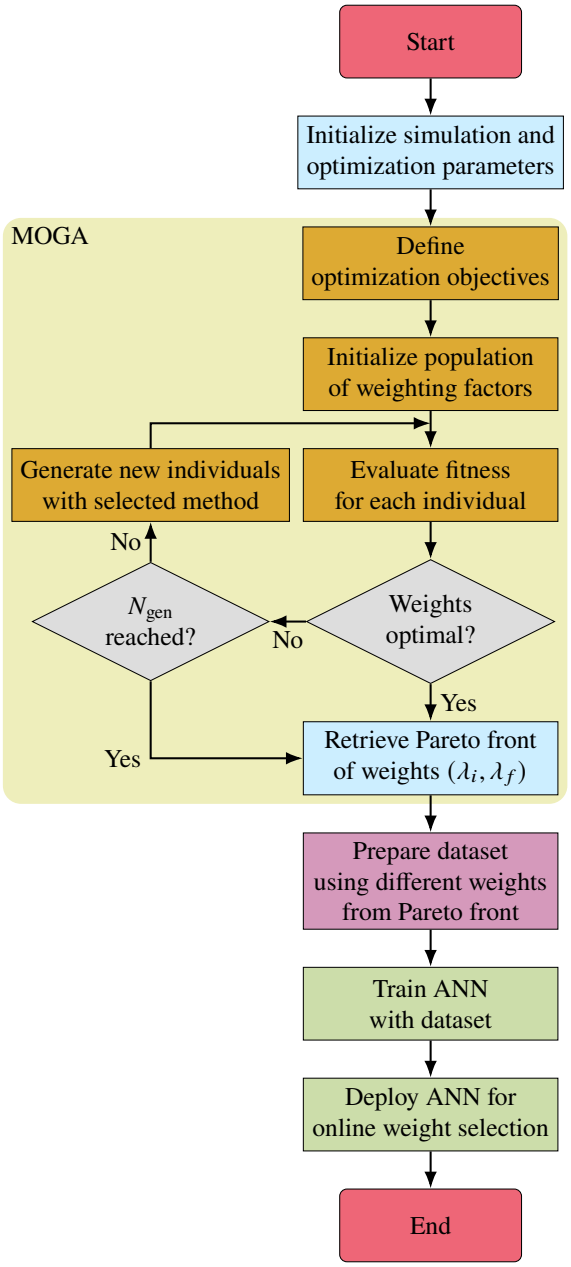


Figure 5. Process flowchart of MOGA-based NN training for weighting factor selection of FCS-MPCC for SPMSM.

3.2. Proposed Weighting Factor Tuning Scheme Based on DRL

The baseline data-driven approach we previously described has one important limitation. Indeed, in order to be able to train the neural network, we first obtained a Pareto

front of weighting factors on the nominal operating conditions. Then, we generated a dataset through test cases, and even though we included many different cases in order for the network to be able to generalize its choice of weighting factors, some unforeseen possibilities or operating conditions might have been overlooked. These omissions might lead to poor weight selection in some situations. The data-driven method we propose is then based on deep reinforcement learning, since in this case, the neural network used for weighting factor selection is trained online, directly on the simulation of the structure of Figure 3. The remainder of this section describes the proposed method.

3.2.1. Generalities

Reinforcement learning is a distinct paradigm within the field of machine learning [36]. It focuses on teaching an agent how to make sequences of decisions by interacting with an environment in order to solve a so-called sequential decision problem. The sequential decision problem can be represented using a Markov decision process (MDP). An MDP is defined by a list of components, $(\mathcal{S}, \mathcal{A}, \mathbb{P}, \rho, \gamma)$ which are described as follows:

- The state space $\mathcal{S}$ corresponding to the set of all possible values of the states describing the environment with which the agent interacts;
- The action space $\mathcal{A}$ corresponding to the set of all possible choices the agent can make to act on the environment;
- The transition dynamics $\mathbb{P}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ characterizing the dynamics of the environment by giving the probability $\mathbb{P}(s(k+1) = S^+ \mid a(k) = A, s(k) = S)$ of obtaining a state $S^+ \in \mathcal{S}$ at time $k+1$ knowing a state-action pair $(S, A) \in \mathcal{S} \times \mathcal{A}$ at time k;
- The reward function $\rho\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ quantifying the desirability for the agent to perform action $A \in \mathcal{A}$ when the environment is in some state $S \in \mathcal{S}$;
- The discount factor $\gamma \in (0,1)$ determining the relative importance of future rewards compared to immediate ones, therefore influencing the agent's consideration of long-term consequences when selecting an action to perform.

Remark 4. *When the transition dynamics are stochastic as described above, the state $s(k+1) \in \mathcal{S}$ at time $k+1$ is a random variable following the probability distribution $\mathbb{P}(\cdot \mid a(k), s(k))$ depending on the values at the time k of the state $s(k) \in \mathcal{S}$ and the action $a(k) \in \mathcal{A}$. It is often denoted $s(k+1) \sim \mathbb{P}(\cdot \mid a(k), s(k))$. When these transition dynamics are deterministic, there is only one possible value for the random variable $s(k+1)$ with probability $\mathbb{P}(s(k+1) \mid a(k), s(k)) = 1$. To simplify the notations, such a case is often denoted $s(k+1) = \mathbb{P}(a(k), s(k))$.*

The RL aims to solve the sequential decision problem described above by constructing a policy function $\pi\colon \mathcal{A} \times \mathcal{S} \to [0,1]$ that can be either stochastic or deterministic. This policy function corresponds to the probability for the agent to perform a given action $a(k) \in \mathcal{A}$ at time k while being in a given state $s(k) \in \mathcal{S}$ at the same time instant. In other words, it is defined such that for any $A \in \mathcal{A}$ and $S \in \mathcal{S}$, we have, at time k, that $\pi(A|S) = \mathbb{P}(a(k) = A | s(k) = S)$. This policy defines the agent's behavior so that, if the environment is in state $s(k)$ at time k, the action $a(k) \in \mathcal{A}$ is a random variable following the probability distribution $\pi(\cdot \mid s(k))$, denoted by $a(k) \sim \pi(\cdot \mid s(k))$. The goal of RL is then to find the optimal policy π^* achieving the maximum expected reward from all states,

$$\pi^* = \arg\max_{\pi} \sum_{k=0}^{+\infty} \gamma^k \mathbb{E}_{(s(k),a(k)) \sim \tau_\pi} [\rho(s(k), a(k))] \tag{12}$$

where $\mathbb{E}_{(s(k),a(k))\sim\tau_\pi}[\gamma^k\rho(s(k),a(k))]$ denotes the expected value of $\gamma^k\rho(s(k),a(k))$ when the pair $(s(k),a(k))$ is sampled from the distribution τ_π of trajectories induced by a given policy π.

Remark 5. *As is the case for the transition dynamics as presented in Remark 4, the policy can also be deterministic. In this case, there is only one possible value for the random variable $a(k)$ with probability $\pi(a(k) \mid s(k)) = 1$. To simplify the notations, such a case is often denoted $a(k) = \pi(s(k))$.*

It is worth highlighting that RL can be model-free or model-based depending on the use of a model of the environment (an approximation or the exact transition dynamics $\mathbb{P}$) in the training process. While the algorithm discussed in the upcoming sections outlines an RL agent interacting with a model-based system, the agent itself is designed without direct access to the system's model. Consequently, the agent's implementation aligns with the model-free RL paradigm.

3.2.2. Twin Delayed Deep Deterministic Policy Gradient

As presented above, the overarching goal of RL is to find the policy π^* maximizing the objective (12). Such a policy often has an intractable mathematical expression. For this reason, we often replace it by a neural network parameterized by a vector of real parameters ϕ that we denote by π_ϕ. One of the purposes of the RL research field is then to define algorithms allowing us to find the optimal vector of parameters ϕ^* such that $\pi_{\phi^*} \approx \pi^*$.

One such algorithm is the TD3 [32]. It is an actor–critic algorithm for continuous action spaces aiming to find a deterministic policy represented by the neural network π_ϕ, also called the *actor*. Since the resulting policy is deterministic, we can adopt the notation of Remark 5 such that the action $a(k) \in \mathcal{A}$ at time k is $a(k) = \pi(s(k))$, with $s(k) \in \mathcal{S}$ the state of the environment at time k. The *critic* is another neural network parameterized by a vector of real parameters χ denoted by Q_χ, approximating the action-value function of the agent, that is the expected return when taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. Like the policy, the true action-value function $Q^\pi\colon \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is often intractable for large MDPs. To maximize the objective (12) with the TD3 algorithm, a policy iteration, which is a method of alternating between policy evaluation and policy improvement within the actor–critic framework, is used.

In the policy evaluation step, the critic parameter vector χ is obtained by minimizing the Bellman error

$$J_Q(\chi) = \mathbb{E}_{(s,a,s^+)\sim\mathcal{D}}\left[\left(Q_\chi(s,a) - \mathrm{Y}(s,a,s^+)\right)^2\right] \tag{13}$$

where $\mathcal{D}$ is a replay buffer of past experiences—triplets containing a state $s \in \mathcal{S}$ of the environment that the agent observed during its training—associated with the action $a \in \mathcal{A}$ it performed following its policy, and the subsequent environment's state $s^+ \in \mathcal{S}$ – and Y is the Bellman regression target defined, for any $s, s^+ \in \mathcal{S}$ and $a \in \mathcal{A}$, as

$$\mathrm{Y}(s,a,s^+) = \rho(s,a) + \gamma Q_\chi(s^+, \pi_\phi(s^+)). \tag{14}$$

In TD3, two critic networks Q_{χ_1} and Q_{χ_2}, with χ_1 and χ_2 their respective vectors of parameters, are trained simultaneously using (13) by replacing χ by either χ_1 or χ_2. To reduce overestimation bias, TD3 uses the minimum of the two Q-values to compute the Bellman regression target as

$$Y(s, a, s^+) = \rho(s, a) + \gamma \min_{i=1,2} Q_{\chi_i}(s^+, \pi_\phi(s^+)) \tag{15}$$

for both critics Q_{χ_1} and Q_{χ_2}.

In the policy improvement step, the actor parameter vector ϕ is updated to maximize the expected return according to the current critic estimates, that is, to maximize

$$J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}[Q_{\chi_1}(s, \pi_\phi(s))]. \tag{16}$$

3.2.3. Comparison of TD3 with Other Reinforcement Learning Algorithms

To show the efficacy of TD3 in optimizing the weighting factors in the MPC asymmetric decision function, we summarize a comprehensive qualitative analysis from the literature comparing TD3 with other state-of-the-art RL methods, such as deep deterministic policy gradient (DDPG) [37] and soft actor–critic (SAC) [38], in Table 4. TD3 provides a good trade-off between control performance, computational complexity, and stability, which is essential for real-time applications.

Table 4. Qualitative comparison of RL algorithms for weighting factor tuning in MPC [37,38].

Algorithm	Stability	Sample Efficiency	Computational Cost	Control Performance
DDPG	Low	Moderate	Low	Moderate
SAC	High	High	High	High
TD3	High	Moderate	Moderate	High

While SAC shows superior robustness and sample efficiency, its computational complexity and memory requirements during training and inference make it less effective for deployment in embedded systems. TD3, on the other hand, addresses the overestimation bias in DDPG and uses delayed policy updates and target smoothing to improve training stability without introducing significant complexity.

Furthermore, TD3 achieved faster convergence to optimal weights and demonstrated improved transient and steady-state performance compared to DDPG, while being computationally lighter than SAC. These qualities make TD3 particularly suitable for industrial motor control applications.

3.2.4. Problem Formulation

Based on what precedes, we propose a TD3-based method to train an RL agent to learn the best policy for tuning the weighting factors of the decision function (11). After training, the resulting actor network is then used in real-time to provide the weighting factors λ_i and λ_f in the control strategy of Figure 3.

To be able to apply the TD3 algorithm, some elements presented in Section 3.2.1 have to be reformulated in terms of the control objective of the paper. We mentioned that the goal of the RL agent's policy is to provide the weighting factors to the predictive controller of Figure 3. It follows naturally that the action space is then $\mathcal{A} = [\underline{\lambda}_i, \overline{\lambda}_i] \times [\underline{\lambda}_f, \overline{\lambda}_f]$ where $\underline{\lambda}_i$, $\overline{\lambda}_i$, $\underline{\lambda}_f$, and $\overline{\lambda}_f$ are as in Section 3.1, with the actions being λ_i and λ_f. The environment on which the RL agent acts is the structure Figure 3, except for the weighting factors, as they are defined to be the action of the RL agent. Finally, to describe the environment, a set of states has to be chosen. Numerous possibilities arise from the structure of Figure 3. However, as for the inputs of the MOGA-based neural network of Figure 4, the environment's states were chosen by trial and error, resulting in

$$s(k) = \begin{bmatrix} \tau_a(k) & \Delta i_a(k) & \int \Delta i_a(k) & f_{\text{sw}}(k) & f^*_{\text{sw}}(k) & \Delta f_{\text{sw}(k)} \end{bmatrix}^\top$$

where $\tau_a(k)$ is the instantaneous total harmonic distortion of the SPMSM's phase a current i_a, $\Delta i_a(k) = i_a^*(k) - i_a(k)$, with $i_a^*(k)$ the phase a current reference obtained from $i_d^*(k)$ and $i_q^*(k)$ using the transforms (2) and (3), and $\Delta f_{\text{sw}}(k) = f_{\text{sw}}^*(k) - f_{\text{sw}}(k)$, with $f_{\text{sw}}^*(k)$ the switching frequency reference for the VSI at time k and $f_{\text{sw}}(k)$ the switching frequency of the VSI at time k.

We see in the state vector the appearance of the switching frequency f_{sw}, its reference f_{sw}^*, and its error Δf_{sw}. This information might seem redundant as the error is directly obtained from the values of the switching frequency and its reference. However, the fact that the agent performs better by using the three input features at the same time might indicate that it is unable to infer their relationship by itself.

In addition, we see that the set of states, that is, the set of inputs to the neural network approximating the agent's policy, is not the same as the one considered in Section 3.1. This is due to the trial and error performed to select the set of inputs for both methods and the learning strategies (supervised learning in Section 3.1 and DRL here), which are relatively sensitive to the information they receive in their input layer. However, we can see that they are related:

- The number of commutations of the VSI's legs is directly linked to the switching frequency;
- The information contained in the q-axis current and its error is contained, at least partly, in the SPMSM's phase a current, its error, and its harmonic distortion.

It therefore makes sense to compare the method of Section 3.1 and the proposed one, as the chosen set of inputs is the one giving the best performance for each strategy.

With the state space $\mathcal{S}$ and the action space $\mathcal{A}$ described (albeit implicitly in the case of $\mathcal{S}$), one element remains to learn the weighting factor selection policy: the reward function. It plays a central role in guiding the agent's learning process and influencing its behavior. It has to be designed in order for the resulting weighting factors to balance multiple objectives in the asymmetric decision function, including accurate current tracking, minimizing switching frequency, maintaining stability in control actions, and ensuring smooth control transitions. These different objectives can be split into three reward terms:

1. **The current tracking reward** R_{current} which incentivizes the SPMSM's phase a current to follow its reference within a given tolerance margin i_{tol} and is expressed as

$$R_{\text{current}}(s(k), a(k)) = \exp\left(-\left(\frac{\Delta i_a(k)}{i_{\text{tol}}}\right)^2\right); \tag{17}$$

2. **The switching frequency reward** R_{switch} ensuring that the switching frequency f_{sw} of the VSI follows its reference f_{sw}^* and is expressed as

$$R_{\text{switch}}(s(k), a(k)) = \exp\left(-\left(\frac{\Delta f_{\text{sw}}(k)}{f_{\text{sw}}^*(k)}\right)^2\right); \tag{18}$$

3. **The specific goal reward** R_{goal} increasing the reward if some specific performance thresholds are achieved, that is expressed as

$$R_{\text{goal}}(s(k), a(k)) = \begin{cases} 0.5 & \text{if } |\Delta i_a(k)| < 0.05 i_{\text{tol}} \text{ and } |\Delta f_{\text{sw}}(k)| < 0.1 f_{\text{sw}}^*(k), \\ 0 & \text{otherwise;} \end{cases} \tag{19}$$

and two penalty terms:

1. **The weighting factor variation penalty** P_{weight} discourages rapid changes in the weighting factors' values to increase stability and is expressed as

$$P_{\text{weight}}(s(k), a(k)) = 0.01\Big((\lambda_i(k) - \lambda_i(k-1))^2 + (\lambda_f(k) - \lambda_f(k-1))^2\Big); \quad (20)$$

2. **The control unbalance penalty** P_{balance} penalizing the gap between the values of the two weighting factors, having for expression

$$P_{\text{balance}}(s(k), a(k)) = 0.1\Big|\lambda_i(k) - \lambda_f(k)\Big|. \quad (21)$$

Combining these components results in the global reward function

$$\rho(s(k), a(k)) = R_{\text{current}}(s(k), a(k)) + 0.5R_{\text{switch}}(s(k), a(k)) + R_{\text{goal}}(s(k), a(k)) - P_{\text{weight}}(s(k), a(k)) - P_{\text{balance}}(s(k), a(k)) \quad (22)$$

to achieve the objectives mentioned above.

The content of Figure 6 is a rewriting of the structure of Figure 3, aiming to highlight the position of the RL agent and its interaction with the environment according to what we defined in this section.

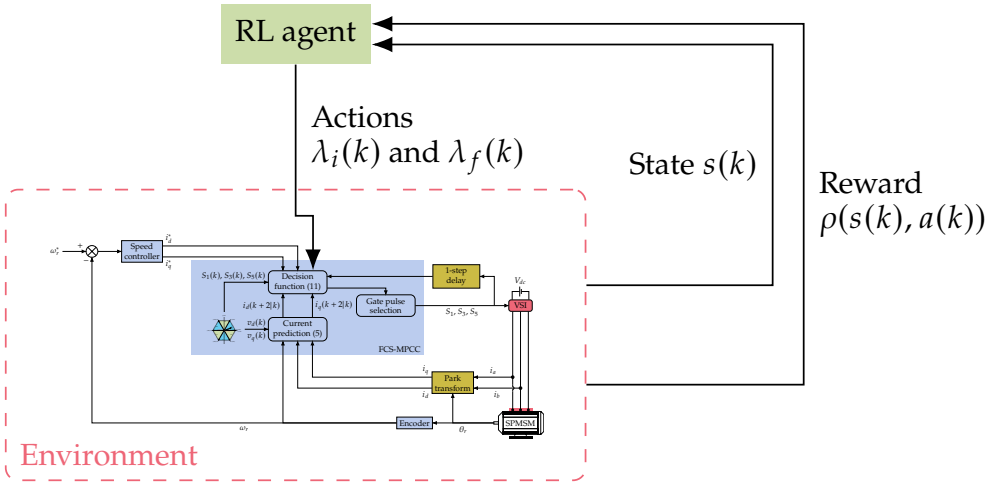


Figure 6. Schematic representation of the interactions between the RL agent and its environment.

3.2.5. Learning Scheme

The proposed approach to train the RL agent with the TD3 algorithm is summarized in Algorithm 1. The algorithm's inputs are three vectors of real parameters, χ_1, χ_2, and ϕ, used to parameterize the two critics Q_{χ_1} and Q_{χ_2} and the actor π_ϕ, the discount factor γ, and the structure of Figure 3 using the parameters from Table 3 for the environment's simulation. We use two critic networks as presented in Section 3.2.2 to take advantage of the clipped double-Q learning trick [32], allowing us to reduce the overestimation bias by using the minimum of the two Q-values when updating the policy. From these initial parameters, we initialize two vectors $\bar{\chi}_1$ and $\bar{\chi}_2$ and use them to parameterize two target critic networks $Q_{\bar{\chi}_1}$ and $Q_{\bar{\chi}_2}$. To stabilize the learning process, these target networks are updated slowly as a weighted average of the critic and target critic parameters χ_i and $\bar{\chi}_i$, with $i \in \{1, 2\}$, through a hyperparameter $\zeta \in (0, 0.5)$ as shown on line 20 of Algorithm 1. We finally initialize a set $\mathcal{D}$, the replay buffer, that is used to store past experiences. After training, the algorithm outputs are χ_1^*, χ_2^*, and ϕ^*, the optimized values of the parameter vectors χ_1, χ_2, and ϕ.

Algorithm 1: TD3 training algorithm.

Input : χ_1, χ_2, ϕ, γ, the FCS-MPCC controlled SPMSM of Figure 3 (the environment)

Output: χ_1^*, χ_2^*, ϕ^*

Initialization:
  $\bar{\chi}_1 \leftarrow \chi_1, \bar{\chi}_2 \leftarrow \chi_2$ `// Target networks' parameters' initialization`
  $\mathcal{D} \leftarrow \emptyset$ `// Replay buffer initialization`
end
for $n \leftarrow 1$ **to** $N_{episode}$ **do**
  Initialize the environment with parameters from Table 3 and initial conditions
  for $k \leftarrow 0$ **to** N_{steps} **do**
    Select the action to apply to the environment $a(k) = \pi_\phi(s(k))$
    Apply $a(k)$ to the FCS-MPCC
    Find the gating pulses $S_1(k)$, $S_3(k)$, and $S_5(k)$ to apply to the VSI with the FCS-MPCC procedure described in Section 2.3
    Apply the gating pulses to the VSI to make the simulated system evolve one step and retrieve $s(k+1)$
    Compute $\rho(s(k), a(k))$ with (22)
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s(k), a(k), \rho(s(k), a(k)), s(k+1))\}$
    if $\mathrm{Card}(\mathcal{D}) \geq \mathrm{Card}(\mathcal{B})$ **then**
      Sample mini-batch $\mathcal{B}$ of 4-tuples $(\varsigma, \eta, \varrho, \varsigma^+)$ from replay buffer $\mathcal{D}$
      $\mathrm{Y}(k) = \mathbb{E}_{(\varsigma,\eta,\varsigma^+)\sim\mathcal{B}}\left[\rho(\varsigma,\eta) + \gamma \min_{i\in\{1,2\}} Q_{\bar{\chi}_i}(\varsigma^+, \pi_\phi(\varsigma^+))\right]$
      for $i \leftarrow 1$ **to** 2 **do**
        $J_Q(\chi_i) = \mathbb{E}_{\varsigma\sim\mathcal{B}}\left[(Q_{\chi_i}(\varsigma) - \mathrm{Y}(k))^2\right]$ `// Compute critic loss`
        $\chi_i \leftarrow \chi_i - \alpha_{\mathrm{lr}} \nabla_{\chi_i} J_Q(\chi_i)$ `// Update critic weights`
        $\bar{\chi}_i \leftarrow \zeta\chi_i + (1-\zeta)\bar{\chi}_i$ `// Soft-update target networks`
      end
      $J_\pi(\phi) = \mathbb{E}_{\varsigma\sim\mathcal{B}}\left[\min_{i\in\{1,2\}} Q_{\bar{\chi}_i}(\varsigma, \pi_\phi(\varsigma))\right]$ `// Compute actor loss`
      $\phi \leftarrow \phi - \alpha_{\mathrm{lr}} \nabla_\phi J_\pi(\phi)$ `// Update actor weights`
    end
  end
end

The learning scheme runs in an episodic fashion for a given number N_{episode} of episodes of N_{steps} time steps each. For each episode, the system, i.e., the SPMSM controlled in a rotation speed closed loop of Figure 3, starts with the same given initial conditions and parameters from Table 3. At each time step $k \leq N_{\mathrm{steps}}$ of the episode, the agent receives the current value of the state of the system $s(k)$ and applies the action $a(k) = \pi_\phi(k)$ to the environment using the current value of the parameter vector ϕ. This action is used as the weighting factors of the FCS-MPCC decision function (11) so that the correct gating pulses can be selected to be applied to the VSI. Using them, the system evolves to the next time step. The resulting reward $\rho(s(k), a(k))$ and next state $s(k+1)$ are stored in the replay buffer as the 4-tuple $(s(k), a(k), \rho(s(k), a(k)), s(k+1))$.

This process is run as long as the number of 4-tuples in the replay buffer $\mathcal{D}$ is below a given threshold $\mathrm{Card}(\mathcal{B})$. Whenever this threshold is reached, an update of the parameter vectors ϕ, χ_1, χ_2, $\bar{\chi}_1$, and $\bar{\chi}_2$ is performed according to the procedure described in Section 3.2.2. The only difference that can be seen in Algorithm 1 is that the Bellman

regression target Y and the actor loss $J_\pi(\phi)$ are computed using the minimum of both target critics $Q_{\bar{\chi}_1}$ and $Q_{\bar{\chi}_2}$. As explained before, this stabilizes the learning process. At the same time, the losses $J_\pi(\phi)$, $J_Q(\chi_i)$, with $i \in \{1,2\}$, and Y are not directly calculated from the current values of the state and action as was presented in Section 3.2.2, but as an average of their values on past experiences to improve the learning process. This is accomplished by sampling tuples from the replay buffer $\mathcal{D}$ into a so-called mini-batch $\mathcal{B}$. The gradients used to update ϕ, χ_1, and χ_2 are calculated using deep deterministic policy gradient [37].

To summarize key elements of Section 3, Table 5 provides a qualitative comparison of the DRL method we propose with the MOGA-NN one on some crucial aspects for real-life implementation of the methods.

Table 5. Qualitative comparison of DRL and MOGA-based ANN for weighting factor optimization in MPCC.

Aspect	DRL	MOGA-Based ANN
Learning approach	Trial-and-error interaction	Offline optimization followed by supervised learning
Adaptability	Dynamically adjusting to changing conditions	Pre-trained for specific scenarios
Computational load	High during training and deployment	High for training but low for online use
Multi-objective handling	Implicit in reward design	Explicit via Pareto-optimal solutions
Implementation complexity	High (requires advanced DRL setup)	Moderate (MOGA and ANN integration)
Robustness to variations	Strong but need proper selection of state and reward	Dependent on training dataset

4. Test Results and Discussion

To verify the efficiency of the proposed data-driven weighting factor tuning approach, numerical simulations of varied motor operating conditions were run. The main objective of this work is to achieve high-performance control of PMSMs used in spindles or servo drive applications that are commonly used in industry. A comprehensive comparative study between a baseline MOGA-based neural network and a TD3-based DRL agent, detailed in Section 3, was performed using a laptop running Windows 11 with an Intel i7, 32 GB RAM, and an NVIDIA RTX 3000 GPU. The parameters used for the MOGA and training of the RL agent are presented in Table 6.

Table 6. MOGA and TD3 learning parameters.

Parameter	Description	Value
$\underline{\lambda}_i$, $\overline{\lambda}_i$	Lower and upper bound for the weighting factor λ_i	0.01 and 20
$\underline{\lambda}_f$, $\overline{\lambda}_f$	Lower and upper bound for the weighting factor λ_f	0.01 and 2.5
N_{MOGA}	MOGA population size	120
N_{gen}	Number of MOGA generations	50
N_{episode}	Number of RL training episodes	800
N_{steps}	Episode steps	4×10^6
i_{tol}	Tolerance margin for the current error	0.3 A
$\text{card}(\mathcal{B})$	Size of the mini-batch in Algorithm 1	256
$\text{card}(\mathcal{D})$	Size of the replay buffer	1×10^4
α_{lr}	Learning rate	1×10^{-4}
ζ	Weight for average update of the target critics	0.005
γ	Agent discount factor	0.9

Using the learning strategy described in Section 3.2, the evolution of the cumulative reward obtained per episode and the average reward are presented in Figure 7. Although the reward is initially high, it shows notable fluctuations as a result of the agent's exploratory

behavior and early policy selection. These variations show the agent's effort to learn from the dynamic system environment while balancing between switching transition reduction and current convergence. After episode 300, the agent has converged on an ideal tuning strategy, as the reward stabilizes with fewer oscillations. Both the per-episode cumulative reward and average reward become consistent, reflecting the successful learning of weighting factor adjustments for robust MPCC performance.

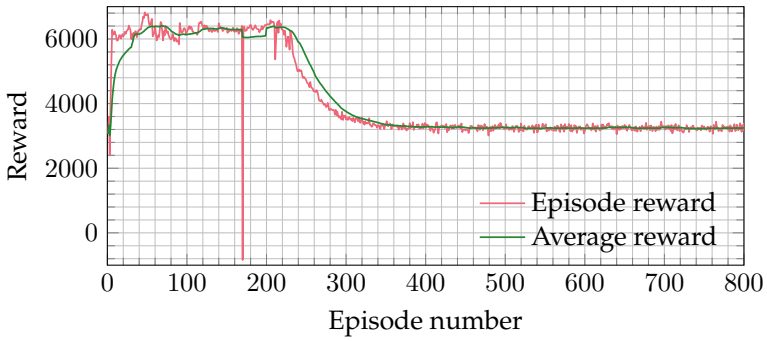
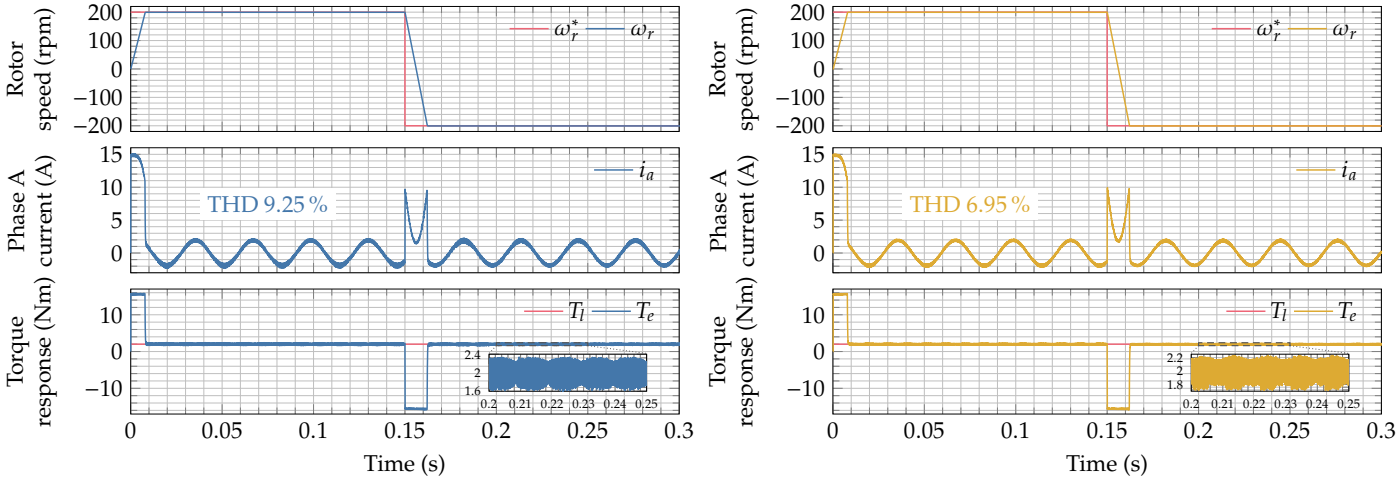


Figure 7. Training process of TD3 agent.

We start by evaluating rotor speed, phase current, and torque responses of the drive motor during acceleration and deceleration at a constant load torque of 2 Nm, as presented in Figure 8. The results obtained with the DRL-based strategy show a significant improvement on the current response with respect to the MOGA-NN-based approach, with reduced harmonic content. Indeed, it can be seen on the phase current response using DRL of Figure 8b that the total harmonic distortion (THD) is of 6.95 % while it is of 9.25 % for the MOGA-NN in Figure 8a. Furthermore, torque ripple is observed to have a total amplitude 0.4 Nm higher with the MOGA-NN approach than with the DRL approach, which achieves smoother torque transitions.



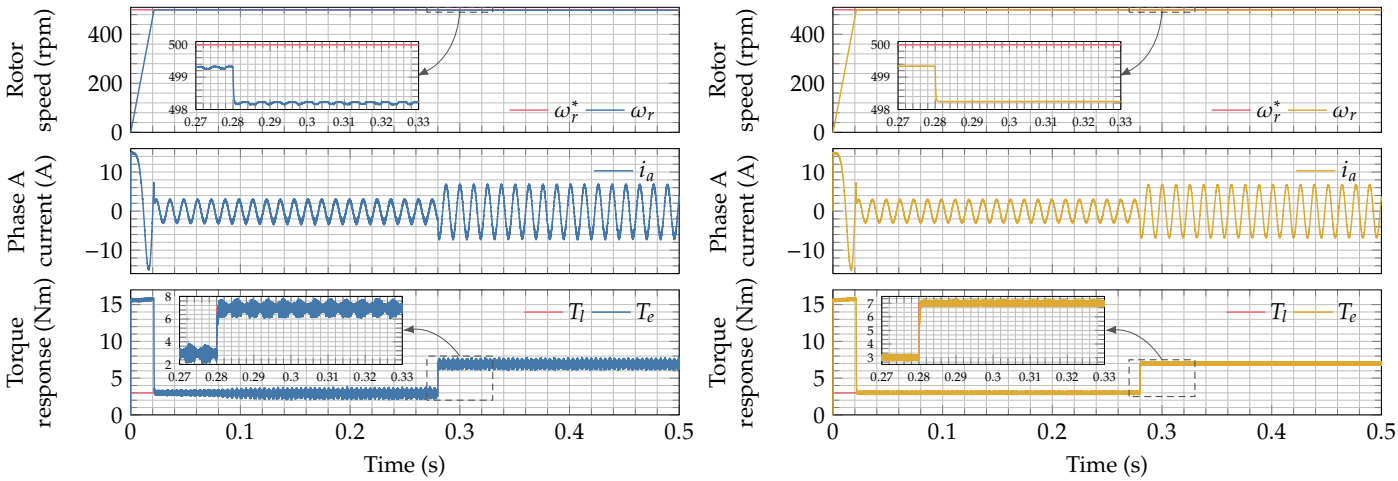
(**a**) With weighting factors obtained through MOGA-NN

(**b**) With weighting factors obtained through DRL

Figure 8. Performance evaluation under a 2 Nm load torque during motor acceleration and deceleration.

We now consider a case where the rotor shaft speed is fixed at 500 rpm and a load variation from 3 Nm to 7 Nm is introduced at 0.28 s. We then study the same outputs as before, as presented in Figure 9. First, without even evaluating the THD, we can see in the phase current responses of Figure 9a,b that the phase currents obtained with the DRL approach present a reduced harmonic content with respect to the currents obtained through the MOGA-NN approach. In the same way, we see in Figure 9a,b that the rotation

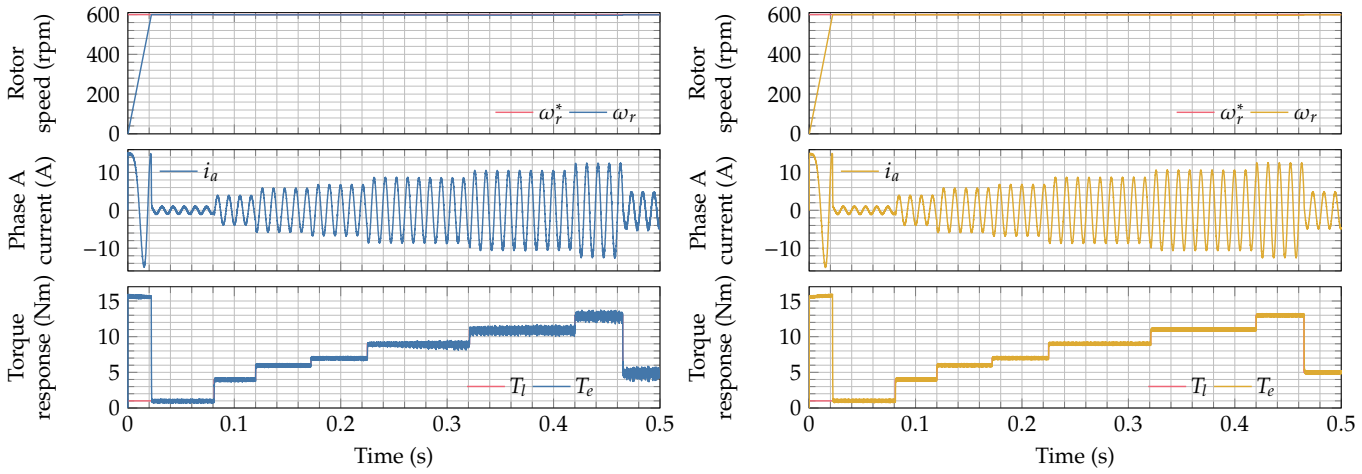
speed and electromagnetic torque responses obtained with the DRL approach present fewer ripples, even after load variation, than what we obtained with the other approach. It shows that the DRL approach achieves better current convergence and demonstrates enhanced robustness to system variations, ensuring smoother transitions and improved overall stability under dynamic load changes.



(**a**) With weighting factors obtained through MOGA-NN (**b**) With weighting factors obtained through DRL

Figure 9. Performance evaluation under load variation at a shaft speed of 500 rpm.

This conclusion is further emphasized by the results presented in Figure 10. In this case, the motor has a fixed shaft speed 600 rpm with different loads of 1 Nm, 4 Nm, 6 Nm, 7 Nm, 9 Nm, 11 Nm, 13 Nm, and 5 Nm applied sequentially at the respective times 0 s, 0.081 s, 0.12 s, 0.172 s, 0.225 s, 0.321 s, 0.42 s, and 0.465 s. As expected, the torque and current ripples obtained with the DRL approach have a constant and reduced amplitude compared to those obtained with the MOGA-NN approach. In addition, Figure 11 gives more information regarding the operation of the proposed control strategies. First, we see that the DRL-based approach reduces the number of switch transitions compared to the MOGA-NN method. Fewer switch transitions directly implies an overall lower switching frequency, which not only minimizes switching losses but also improves the overall efficiency and thermal performance of the inverter. This reduction in switching activity is critical for extending the lifespan of the power electronic components, making the DRL method more practical for long-term applications. Furthermore, the reduced current errors Δi_d and Δi_q in the DRL approach ensure better electromagnetic torque stability, reduced harmonic content, and overall improved dynamic and steady-state performance of the PMSM drive. We also see that the DRL-based approach demonstrates better consistency and adaptability in the values of the weighting factors λ_i and λ_f, maintaining a stable and optimal profile compared to the fluctuating and unstable adjustments of MOGA-NN. This stability enhances MPCC robustness, enabling reliable performance under dynamic conditions. Additionally, the DRL method adapts efficiently to system variations in real time, offering faster response times and better control performance compared to the static MOGA-NN approach.



(**a**) With weighting factors obtained through MOGA-NN (**b**) With weighting factors obtained through DRL

Figure 10. Performance evaluation under different load variations at a shaft speed of 600 rpm.

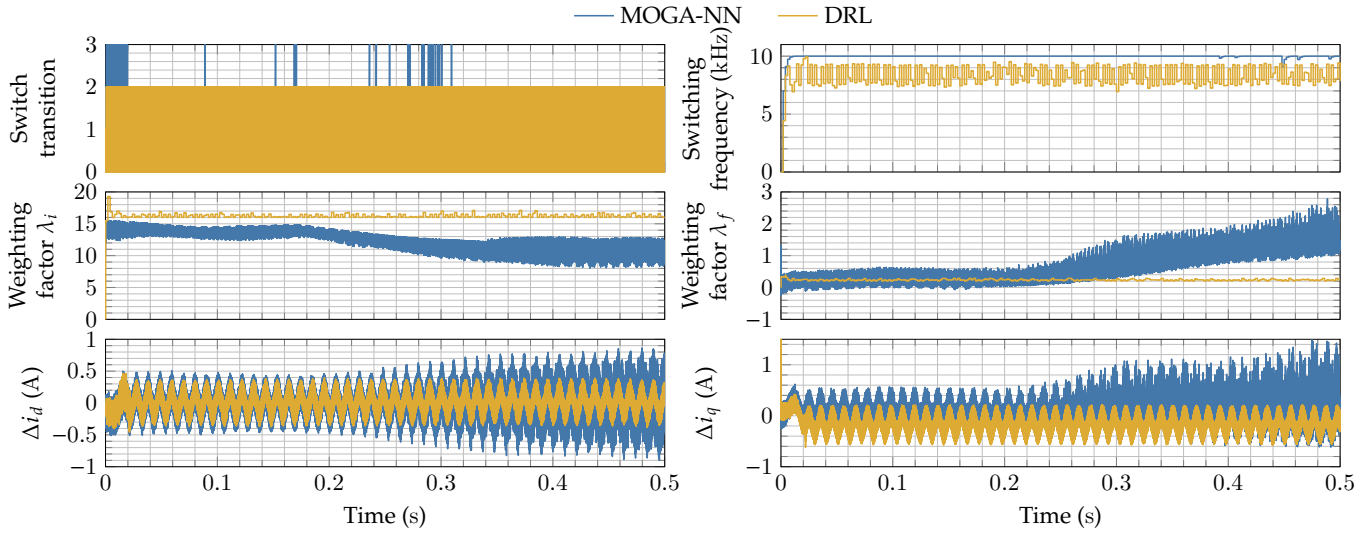
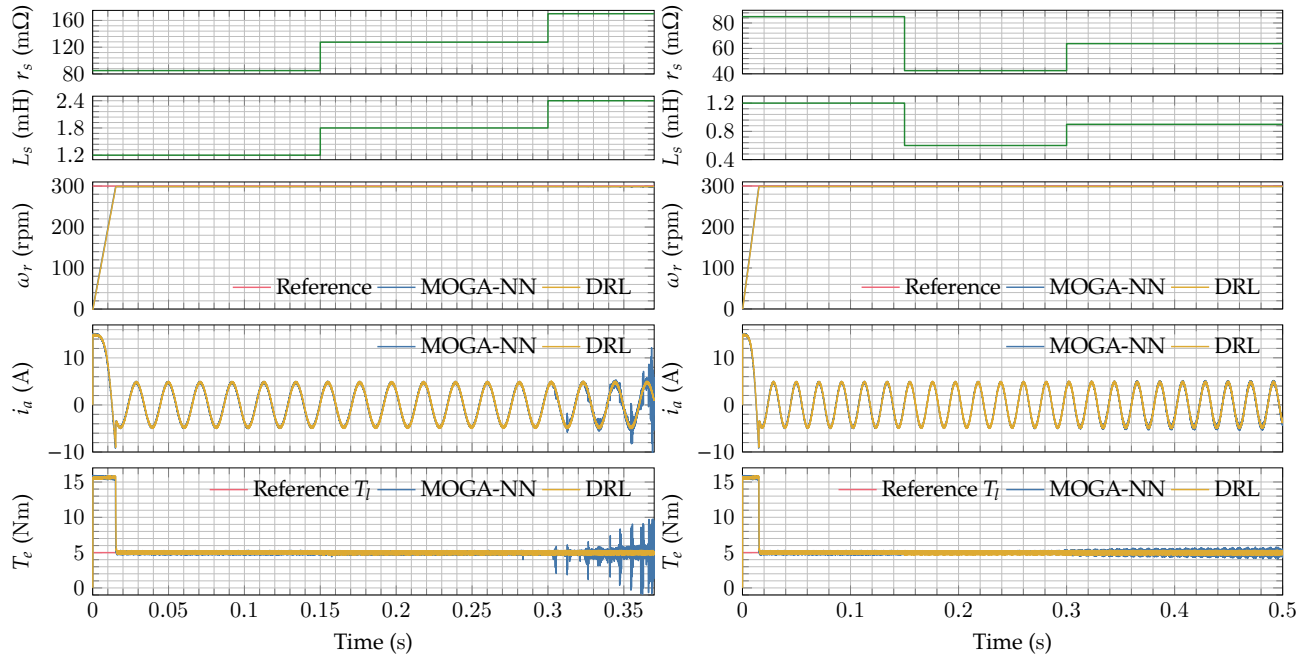


Figure 11. Model predictive current controller performance under different load variations at a shaft speed of 600 rpm.

For the operation of electrical machines, it is also interesting to study what happens when the electrical parameters, that is, the stator resistance and inductance, do not match the values for which the controller is tuned. To this end, we first considered operations under nominal parameter values, and we then increased them by 50 % and 100 % of their nominal values with respect to their nominal values, as illustrated in Figure 12a. In addition, we considered operations under nominal parameter values, and we then decreased them by 50 % and 25 % of their nominal values with respect to their nominal values, as illustrated in Figure 12b. Once again, the results demonstrate the superiority of the proposed DRL-based weighting factor tuning approach. Indeed, the current and torque responses are more stable and present fewer ripples with the DRL approach than with the MOGA-NN approach. In addition, the MOGA-NN approach tends to be unstable for important mismatches in electrical parameters, as can be seen in Figure 12a.

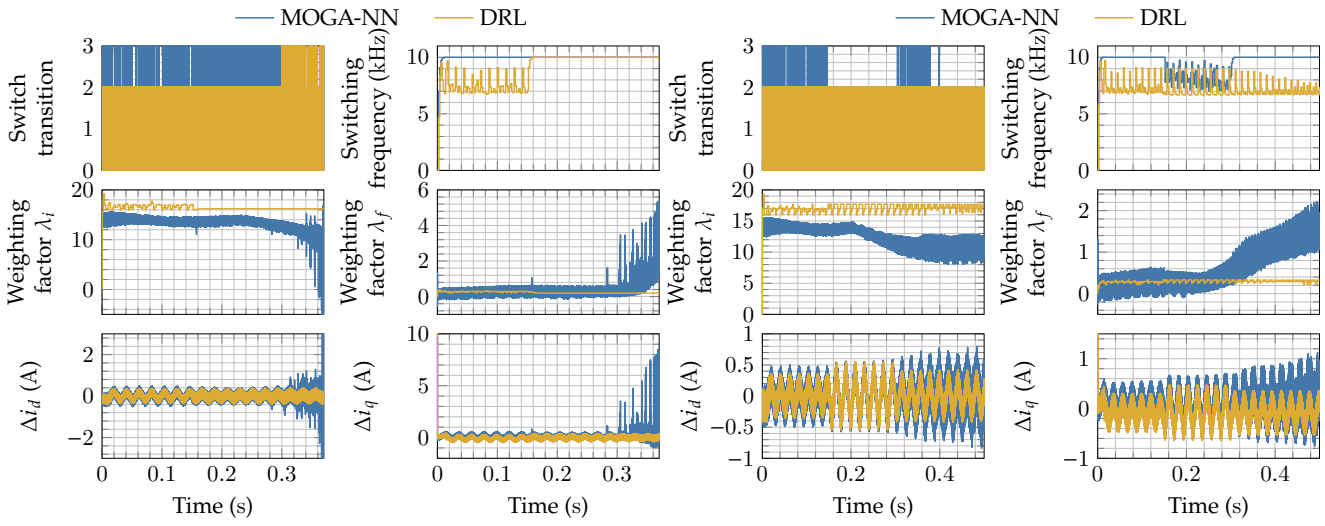
As in the previous case, Figure 13 gives more information regarding the operation of the proposed control strategies. The same overall behavior of the DRL and MOGA-NN approaches as the ones of Figure 11 can be observed. Indeed, the DRL approach gives fewer switch transitions and, as a consequence, a reduced switching frequency, except in the case

of increased electrical parameter values. However, even in this last case, we observe a stable and consistent value for the weighting factors and reduced current errors Δi_d and Δi_q with the DRL approach, while the MOGA-NN approach exhibits unstable behaviors. These results therefore validate the proposed DRL approach as a superior solution for MPCC optimization, ensuring stable and efficient motor operation even in the presence of significant parameter variations. The proposed approach effectively handles the external disturbance and parametric variation due to its capability to explore and exploit under a wide range of operating conditions during training. By dynamically adapting the cost function weights, the controller maintains a balanced response between competing objectives such as current tracking accuracy and switching transition minimization, improving the overall performance of the control system.



(**a**) Increment in electrical parameters (**b**) Decrement in electrical parameters

Figure 12. Performance analysis for mismatched electrical parameters.



(**a**) Increment in electrical parameters (**b**) Decrement in electrical parameters

Figure 13. Model predictive current controller performance under mismatched electrical parameters.

Finally, while maintaining a constant load and motor speed of 5 Nm and 300 rpm, respectively, we measured the instantaneous THD of the motor phase currents. Figure 14 presents the THD values at 0.05 s, 0.2 s, and 0.35 s. We can observe that the DRL-based controller achieves consistently lower THD compared to the MOGA-NN-based one as we obtain at the different time instants, respectively, 9.38 %, 11.43 %, and 15.77 %, while the MOGA-NN approach gives 9.64 %, 11.76 %, and 16.78 %. This reduction in THD showcases the DRL-based controller's enhanced capacity to mitigate harmonics dynamically, contributing to improved drive performance and reduced harmonic distortion in varying operational conditions. The results once again affirm the effectiveness of the DRL approach in maintaining a cleaner current waveform, which is critical for robust and efficient motor operation.

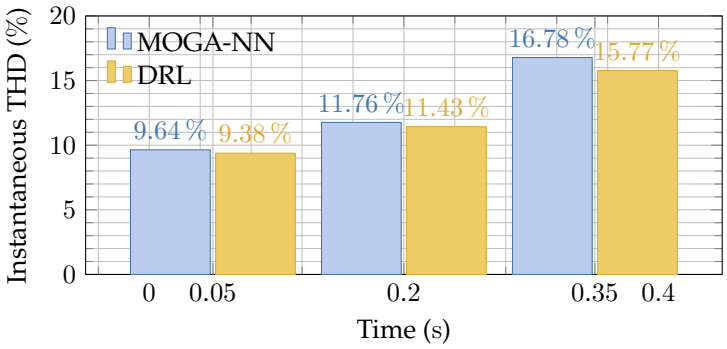


Figure 14. Total harmonic distortion under a load of 5 Nm and shaft speed of 300 rpm.

5. Conclusions

This paper presents an online approach for tuning weighting factors in the finite control set model predictive current control of permanent magnet synchronous motor drive systems. The proposed method effectively tackles the challenge of designing weighting factors for asymmetric cost functions to achieve a balance among multiple control objectives inherent in traditional model predictive current control. By using a TD3-based deep reinforcement learning agent to choose the weighting factors online, we enable real-time tuning across various operating conditions and multiple objectives, ensuring optimal control performance. The method is specifically designed to optimize weighting factors for multi-objective system performance, allowing for efficient real-time adjustments to meet dynamic operational requirements. Comparative validation test results confirm the effectiveness of the proposed approach in determining optimal weighting factors for the asymmetric cost function.

Furthermore, the proposed TD3-based algorithm, by allowing online optimization of the weighting factors in an asymmetric cost function, effectively adapts to varying operating conditions without requiring manual tuning. However, due to the lack of a general theoretical framework to ensure the stability of RL algorithms, extensive validation remains essential prior to industrial deployment. The proposed DRL-based control strategy represents a significant step forward in intelligently addressing multiple conflicting objectives within the model predictive control framework. By learning to dynamically tune the weighting factors, the algorithm improves both steady-state and transient performance, demonstrated through reduced torque ripple, lower current harmonics, and smoother switching transitions. This intelligent weighting approach allows the control system to balance performance trade-offs more effectively than static or manually tuned approaches. The proposed design is capable of incorporating multiple additional objectives, such as voltage constraints and efficiency optimization, which will be key directions in future work.

This enables the development of a more adaptive control strategy capable of maintaining optimal performance across a wide range of operating conditions.

Author Contributions: Conceptualization, M.U.; methodology, M.U.; validation, M.U.; writing—original draft preparation, M.U. and A.S.; writing—review and editing, M.U., A.S., T.C. and N.L.; supervision, T.C. and N.L.; funding acquisition, N.L. All authors have read and agreed to the submitted version of the manuscript.

Funding: This work was supported by ANR and Région Normandie through the HAISCoDe (ANR-20-THIA-0021) and PAMAP projects.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no competing interests.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial neural network
CCS	Continuous control set
DDPG	Deep deterministic policy gradient
DRL	Deep reinforcement learning
FCS	Finite control set
IAE	Integral absolute error
ISE	Integral squared error
ITAE	Integral time-weighted absolute error
ITSE	Integral time-weighted squared error
MOGA	Multi-objective genetic algorithm
MDP	Markov decision process
MPC	Model predictive control
MPCC	Model predictive current control
NN	Neural network
PI	Proportional–integral
PMSM	Permanent magnet synchronous motor
RL	Reinforcement learning
SAC	Soft Actor Critic
SMPC	Sequential model predictive control
SPMSM	Surface permanent magnet synchronous motor
TD3	Twin Delayed Deep Deterministic Gradient
THD	Total harmonic distortion
VSI	Voltage source inverter

References

1. Nam, K.H. *AC Motor Control and Electrical Vehicle Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2010.
2. Xu, W.; Ismail, M.M.; Islam, M.R. *Permanent Magnet Synchronous Machines and Drives: Flux Weakening Advanced Control Techniques*; CRC Press: Boca Raton, FL, USA, 2024.
3. Bida, V.M.; Samokhvalov, D.V.; Al-Mahturi, F.S. PMSM vector control techniques–A survey. In Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, Moscow, Russia, 29 January–1 February 2018; pp. 577–581.
4. Yao, H.; Yan, Y.; Shi, T.; Zhang, G.; Wang, Z.; Xia, C. A novel SVPWM scheme for field-oriented vector-controlled PMSM drive system fed by cascaded H-bridge inverter. *IEEE Trans. Power Electron.* **2021**, *36*, 8988–9000. [CrossRef]

5. Sun, X.; Diao, K.; Lei, G.; Guo, Y.; Zhu, J. Direct torque control based on a fast modeling method for a segmented-rotor switched reluctance motor in HEV application. *IEEE J. Emerg. Sel. Top. Power Electron.* **2021**, *9*, 232–241. [CrossRef]
6. Yu, Z.; Long, J. Review on advanced model predictive control technologies for high-power converters and industrial drives. *Electronics* **2024**, *13*, 4969. [CrossRef]
7. Mahmoud, H.; Mohamed, M.A.; Hassan, A.A.; Mossa, M.A. Optimal predictive voltage control of a wind driven five phase PMSG system feeding an isolated load. *e-Prime Adv. Electr. Eng. Electron. Energy* **2024**, *9*, 100697. [CrossRef]
8. Zhang, Z.; Li, Z.; Kazmierkowski, M.P.; Rodríguez, J.; Kennel, R. Robust predictive control of three-level NPC back-to-back power converter PMSG wind turbine systems with revised predictions. *IEEE Trans. Power Electron.* **2018**, *33*, 9588–9598. [CrossRef]
9. Zhang, Y.; Liu, X.; Li, H.; Zhang, Z. A model independent predictive control of PMSG wind turbine systems with a new mechanism to update variables. *Energies* **2023**, *16*, 3764. [CrossRef]
10. Karimi, M.H.; Rostami, M.A.; Zamani, H. Continuous control set model predictive control for the optimal current control of permanent magnet synchronous motors. *Control Eng. Pract.* **2023**, *138*, 105590. [CrossRef]
11. Karamanakos, P.; Geyer, T. Guidelines for the design of finite control set model predictive controllers. *IEEE Trans. Power Electron.* **2020**, *35*, 7434–7450. [CrossRef]
12. Tomlinson, M.; du Toit Mouton, H.; Kennel, R.; Stolze, P. A fixed switching frequency scheme for finite-control-set model predictive control – Concept and algorithm. *IEEE Trans. Ind. Electron.* **2016**, *63*, 7662–7670. [CrossRef]
13. Xu, S.; Sun, Z.; Yao, C.; Zhang, H.; Hua, W.; Ma, G. Model predictive control with constant switching frequency for three-level T-type inverter-fed PMSM drives. *IEEE Trans. Ind. Electron.* **2022**, *69*, 8839–8850. [CrossRef]
14. Mossa, M.A.; Mohamed, R.A.; Al-Sumaiti, A.S. Performance enhancement of a grid connected wind turbine-based PMSG using effective predictive control algorithm. *IEEE Access* **2025**, *13*, 64160–64185. [CrossRef]
15. Usama, M.; Kim, J. Model-free current control solution employing intelligent control for enhanced motor drive performance. *Sci. Rep.* **2025**, *15*, 60. [CrossRef] [PubMed]
16. Dragičević, T.; Novak, M. Weighting factor design in model predictive control of power electronic converters: An artificial neural network approach. *IEEE Trans. Ind. Electron.* **2019**, *66*, 8870–8880. [CrossRef]
17. Zhang, Y.; Zhang, Z.; Babayomi, O.; Li, Z. Weighting factor design techniques for predictive control of power electronics and motor drives. *Symmetry* **2023**, *15*, 1219. [CrossRef]
18. Cortes, P.; Kouro, S.; La Rocca, B.; Vargas, R.; Rodriguez, J.; Leon, J.I.; Vazquez, S.; Franquelo, L.G. Guidelines for weighting factors design in Model Predictive Control of power converters and drives. In Proceedings of the 2009 IEEE International Conference on Industrial Technology, Churchill, Australia, 10–13 February 2009.
19. Rodriguez, J.; Garcia, C.; Mora, A.; Davari, S.A.; Rodas, J.; Valencia, D.F.; Elmorshedy, M.; Wang, F.; Zuo, K.; Tarisciotti, L.; et al. Latest advances of model predictive control in electrical drives—Part II: Applications and benchmarking with classical control methods. *IEEE Trans. Power Electron.* **2022**, *37*, 5047–5061. [CrossRef]
20. Caseiro, L.M.A.; Mendes, A.M.S.; Cruz, S.M.A. Dynamically weighted optimal switching vector model predictive control of power converters. *IEEE Trans. Ind. Electron.* **2019**, *66*, 1235–1245. [CrossRef]
21. Li, X.; Zhang, H.; Shadmand, M.B.; Balog, R.S. Model predictive control of a voltage-source inverter with seamless transition between islanded and grid-connected operations. *IEEE Trans. Ind. Electron.* **2017**, *64*, 7906–7918. [CrossRef]
22. Norambuena, M.; Rodriguez, J.; Zhang, Z.; Wang, F.; Garcia, C.; Kennel, R. A very simple strategy for high-quality performance of AC machines using model predictive control. *IEEE Trans. Power Electron.* **2019**, *34*, 794–800. [CrossRef]
23. Zhang, J.; Li, L.; Norambuena, M.; Rodriguez, J.; Dorrell, D.G. Sequential model predictive control of direct matrix converter without weighting factors. In Proceedings of the IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018; pp. 1477–1482.
24. Arshad, M.H.; Abido, M.A.; Salem, A.; Elsayed, A.H. Weighting factors optimization of model predictive torque control of induction motor using NSGA-II with TOPSIS decision making. *IEEE Access* **2019**, *7*, 177595–177606. [CrossRef]
25. Li, S.; Won, H.; Fu, X.; Fairbank, M.; Wunsch, D.C.; Alonso, E. Neural-network vector controller for permanent-magnet synchronous motor drives: Simulated and hardware-validated results. *IEEE Trans. Cybern.* **2020**, *50*, 3218–3230. [CrossRef]
26. Fu, X.; Li, S. A novel neural network vector control technique for induction motor drive. *IEEE Trans. Energy Convers.* **2015**, *30*, 1428–1437. [CrossRef]
27. Guazzelli, P.R.U.; de Andrade Pereira, W.C.; de Oliveira, C.M.R.; de Castro, A.G.; de Aguiar, M.L. Weighting factors optimization of predictive torque control of induction motor by multiobjective genetic algorithm. *IEEE Trans. Power Electron.* **2019**, *34*, 6628–6638. [CrossRef]
28. Zanchetta, P. Heuristic multi-objective optimization for cost function weights selection in finite states model predictive control. In Proceedings of the 2011 Workshop on Predictive Control of Electrical Drives and Power Electronics, Munich, Germany, 14–15 October 2011; pp. 70–75.

29. Villarroel, F.; Espinoza, J.R.; Rojas, C.A.; Rodriguez, J.; Rivera, M.; Sbarbaro, D. Multiobjective switching state selector for finite-states model predictive control based on fuzzy decision making in a matrix converter. *IEEE Trans. Ind. Electron.* **2013**, *60*, 589–599. [CrossRef]
30. Novak, M.; Xie, H.; Dragicevic, T.; Wang, F.; Rodriguez, J.; Blaabjerg, F. Optimal cost function parameter design in predictive torque control (PTC) using artificial neural networks (ANN). *IEEE Trans. Ind. Electron.* **2021**, *68*, 7309–7319. [CrossRef]
31. Vazquez, S.; Marino, D.; Zafra, E.; Peña, M.D.V.; Rodríguez-Andina, J.J.; Franquelo, L.G.; Manic, M. An artificial intelligence approach for real-time tuning of weighting factors in FCS-MPC for power converters. *IEEE Trans. Ind. Electron.* **2022**, *69*, 11987–11998. [CrossRef]
32. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; PMLR: New York, NY, USA, 2018; pp. 1587–1596.
33. Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; John Wiley & Sons: Chichester, UK, 2001.
34. Atkinson, K.E. *An Introduction to Numerical Analysis*; John Wiley & Sons: Hoboken, NJ, USA, 1989.
35. Zhang, Y.; Xu, D.; Liu, J.; Gao, S.; Xu, W. Performance improvement of model-predictive current control of permanent magnet synchronous motor drives. *IEEE Trans. Ind. Appl.* **2017**, *53*, 3683–3695. [CrossRef]
36. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
37. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2019**, arXiv:1509.02971.
38. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; PMLR: New York, NY, USA, 2018.