


Article

Artificial Neural Network Based Optimal Feedforward Torque Control of Interior Permanent Magnet Synchronous Machines: A Feasibility Study and Comparison with the State-of-the-Art

Max A. Buettner [†], Niklas Monzen [†]  and Christoph M. Hackl ^{*}

Department of Electrical Engineering and Information Technology, Hochschule München (HM) University of Applied Sciences, Lothstr. 64, 80335 München, Germany; max.buettner@hm.edu (M.A.B.); niklas.monzen@hm.edu (N.M.)

* Correspondence: christoph.hackl@hm.edu

[†] These authors contributed equally to this work.

Abstract: A novel Artificial Neural Network (ANN) Based Optimal Feedforward Torque Control (OFTC) strategy is proposed which, after proper ANN design, training and validation, allows to analytically compute the optimal reference currents (minimizing copper and iron losses) for Interior Permanent Magnet Synchronous Machines (IPMSMs) with highly operating point dependent nonlinear electric and magnetic characteristics. In contrast to conventional OFTC, which either utilizes large look-up tables (LUTs; with more than three input parameters) or computes the optimal reference currents numerically or analytically but iteratively (due to the necessary online linearization), the proposed ANN-based OFTC strategy does not require iterations nor a decision tree to find the optimal operation strategy such as e.g., Maximum Torque per Losses (MTPL), Maximum Current (MC) or Field Weakening (FW). Therefore, it is (much) faster and easier to implement while (i) still machine nonlinearities and nonidealities such as e.g., magnetic cross-coupling and saturation and speed-dependent iron losses can be considered and (ii) very accurate optimal reference currents are obtained. Comprehensive simulation results for a real and highly nonlinear IPMSM clearly show these benefits of the proposed ANN-based OFTC approach compared to conventional OFTC strategies using LUT-based, numerical or analytical computation of the reference currents.

Keywords: electrical drive control system; operation management; optimal feedforward torque control; optimal reference current computation; transformer-like nonlinear machine model; artificial neural network; synchronous motor; interior permanent magnet synchronous machine; machine learning



Citation: Buettner, M.A.; Monzen, N.; Hackl, C.M. Artificial Neural Network Based Optimal Feedforward Torque Control of Interior Permanent Magnet Synchronous Machines: A Feasibility Study and Comparison with the State-of-the-Art. *Energies* **2022**, *15*, 1838. <https://doi.org/10.3390/en15051838>

Academic Editors: Nicola Bianchi, Ludovico Ortoombina and K.T. Chau

Received: 10 January 2022

Accepted: 25 February 2022

Published: 2 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Notation

\mathbb{N}, \mathbb{R} : natural, real numbers; $\mathbf{x} := (x_1, \dots, x_n)^\top \in \mathbb{R}^n$: column vector, $n \in \mathbb{N}$ where “ \top ” and “ $:=$ ” mean “transposed” and “is defined as”, resp.; $\mathbf{a}^\top \mathbf{b} := a_1 b_1 + \dots + a_n b_n$: scalar product of vectors \mathbf{a} & \mathbf{b} ; $\|\mathbf{x}\| := \sqrt{\mathbf{x}^\top \mathbf{x}} = \sqrt{x_1^2 + \dots + x_n^2}$: Euclidean norm of \mathbf{x} ; $\mathbf{X} \in \mathbb{R}^{n \times n}$: matrix (n rows & columns); $\mathbf{X}^{-1}, \mathbf{X}^{-\top}$: inverse, inverse transpose of \mathbf{X} (if exist), resp.; $\mathbf{I}_n := \text{diag}(1, \dots, 1) \in \mathbb{R}^{n \times n}$: identity matrix; $\mathbf{0}_n := (0, \dots, 0)^\top \in \mathbb{R}^n$: zero vector; $\mathbf{J} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$: rotation matrix (by $\frac{\pi}{2}$).

Remark: All physical quantities are introduced and explained in the text to ease reading.

1. Introduction

1.1. Motivation

Electric machines are widely used in plenty of applications for manufacturing, grinding, pumping or in robots, electric vehicles, wind turbines or conventional power plants.

For all applications, reliability and efficiency play key roles. Though electric motors still consume over 50% of the world's generated electric energy [1] from which 84.3% is generated from fossil fuels [2]. In particular for the mobility sector, for the consumer, besides the lacking charging opportunities, the range of electric vehicles is considered the major obstacle for quick penetration of the future mobility market [3]. Nevertheless, hybrid electric vehicles (HEV) and battery electric vehicles (BEV) will become standard vehicles in the mobility sector. Therefore, engineers worldwide are striving to optimize the electrical drive system in order to reduce energy consumption and increase efficiency.

In this regard, the optimal feedforward torque control (OFTC) problem must be solved to compute optimal reference currents which minimize iron and copper losses by invoking optimal operation strategies for all operating conditions [4,5]. The different operation strategies are usually *Maximum Torque per Ampere/Current* (MTPA/MTPC), *Maximum Torque per Losses* (MTPL; considering also iron losses), *Field Weakening* (FW), *Maximum Current* (MC) and *Maximum Torque per Voltage* (MTPV). As alternative, *Maximum Torque per Flux* (MTPF) has been proposed as it is easier to compute, but it should be avoided as it represents an oversimplification with *suboptimal* operation and MTPV should be used instead [6]. All of those optimal operation strategies exploit the non-uniqueness of the current pairs which produce the same torque [4]. In order to minimize conduction and switching losses additionally, optimal pulse patterns (OPPs) must be implemented [7] which have to be computed separately for nonlinear machines with significant anisotropy [8,9].

In this paper, a novel ANN-based OFTC is proposed. The OFTC problem has been discussed in numerous publications (see e.g., [10–19] for MTPC/MTPA, [12,14,18,20–24] for FW, [4,5,12,25] for MC, [4,5,13,26–29] for MTPL, and [6,14,20–23] for MTPV). The goal of OFTC is to compute or look up optimal reference currents for each of the aforementioned operation strategies. This is mostly done *numerically* or in few cases *analytically* imposing simplifying assumptions such as the *neglect* of the nonidealities: (a) varying stator resistance, (b) nonlinear magnetic cross-coupling and saturation, (c) physical current and voltage constraints, (d) iron losses and (e) temperature dependencies.

Analytical approaches are in particular of interest due to their rather simple, equation-based implementation, high accuracy and fast computation. However, if all nonidealities (see (a)–(e) above) shall be considered, the analytical solution must be computed iteratively due to the online linearization of the highly nonlinear OFTC problem. Numerical approaches usually rely on look-up tables (LUTs). To account for all nonidealities (see (a)–(e) above), the LUTs become rather huge and often depend on more than three inputs. In conclusion, both approaches come with drawbacks either due to limited computational power or limited memory storage.

In this paper, these drawbacks shall be overcome by using a rather simple ANN to solve the OFTC problem still analytically but not iteratively. To not hit computational and memory constraints, the ANN must be tailored to the capability of the utilized real-time platforms (e.g., digital signal processor (DSP) or field-programmable gate array (FPGA)).

As artificial neural networks are a promising technology, machine learning-based approaches in electrical drives have already been reported in numerous publications (see the recent overview preprint [30] with 259 references). Exemplary applications of ANNs in the field of electrical drive systems are: ANN-based speed, current or speed and current controllers [31–38], ANN-based parameter/system identification [39–41], ANN-based temperature or resistance estimation [42,43], ANN-based direct/predictive torque or model predictive control [44–46], ANN-based torque observers [47], ANN-based current waveform prediction [48], ANN-based encoderless control [49–52], ANN-based torque ripple reduction [53,54], ANN-based condition monitoring or fault detection [55–58], ANN-based optimal pulse patterns [59], and ANN-based multi-objective optimization for machine design [60].

However, to the best knowledge of the authors, ANNs have not been used this far to solve the OFTC problem as proposed in this paper. Therefore, this paper is the very first considering ANN-based OFTC.

The remainder of the paper is structured as follows: In the following three subsections, a detailed literature review is presented, the problem is formally stated and the proposed solution is briefly introduced (see Sections 1.2–1.4, respectively.). In Section 2, the novel idea of ANN-based OFTC is discussed in detail; starting with the ANN design in Section 2.1, followed by the ANN training (including a thorough explanation of the required data set creation) in Section 2.2, completed by the ANN validation in Section 2.3. In Section 3, the ANN-based OFTC strategy is implemented in the closed-loop control system of an electrical drive with highly nonlinear magnetic characteristics and iron losses and its performance is compared to conventional state-of-the-art OFTC with analytical optimal reference current computation (ORCC). Finally, Section 4 concludes the paper by (a) summarizing the proposed idea and obtained results and (b) by giving a short outlook on future research directions.

1.2. Detailed Literature Review

In this subsection, the references listed above concerning OFTC and ANN in electrical drives are discussed in more detail.

1.2.1. Optimal Feedforward Torque Control (OFTC)

The most common OFTC strategy is MTPC (often also called MTPA) which minimizes copper (Joule) losses only. A comprehensive literature review on MTPC was recently published in [19]. The early contribution [10] computes optimal stator current angles considering current and voltage limits in PMSM electrical drives to minimize copper losses. Iron losses, magnetic cross-coupling and saturation effects are neglected. The MTPC algorithm in [11] is based on [10] but in addition, considers cross-coupling effects. The optimal current angles are obtained by online perturbation of the current magnitude. In [14], an analytical computation of the optimal MTPC currents is presented based on Lagrangian multipliers and Ferrari's method. However, iron losses and magnetic cross-coupling and saturation effects are not considered.

Iron losses complicate efficiency enhancement but have also been considered in several publications. Loss Minimization Control (LMC) was proposed in [26] by formulating and solving a convex optimization problem numerically and iteratively to minimize copper and iron losses. In [27], a novel calculation method for the iron loss resistance is introduced which is based on a slope evaluation of the linear approximation of the relation between semi-input power and squared speed-dependent electromotive force. The loss minimization algorithm (LMA) proposed in [28] conducts an iterative search to minimize copper and iron losses online. Magnetic cross-coupling and saturation effects are not considered. In [13], it proposes an optimal current feed-forward angle to minimize copper and iron losses in nonlinear traction drives. In [29], Maximum Efficiency per Ampere (MEPA; i.e., MTPL) control is proposed and performs an online search to find optimal stator current angles minimizing copper and iron losses for PMSMs with significant magnetic cross-coupling and saturation effects.

In [20], an MTPV strategy for reluctance synchronous machines is proposed which allows to analytically compute the optimal MTPV reference currents. Stator resistance and magnetic saturation are considered while iron losses and magnetic cross-coupling effects are not considered. In [21], the effect of neglecting the stator resistance on the MTPV operation is discussed and an analytical computation of the optimal MTPV currents is proposed. Iron losses, magnetic saturation and cross-coupling effects are neglected. [6] computes the MTPV hyperbola analytically while considering the stator resistance. Magnetic saturation and iron losses are neglected. Magnetic cross-coupling effects were not explicitly addressed but the presented approach allows to incorporate those as well. [23] combines FW and MTPV with sensorless control. Iron losses are considered and the position estimation error due to iron losses is compensated for by a modified disturbance observer.

Only a few publications deal with the overall operation management for OFTC including MTPC, MC, FW and MTPV. LUT-based OFTC strategies for MC, MTPC and MTPV are proposed in [12] while magnetic cross-coupling effects and iron losses are neglected. OFTC for PMSMs including MTPC, MC, FW and MTPV is proposed in [15]. The optimal stator currents are obtained by an analytical computation but iron losses, magnetic cross-coupling and saturation effects are not considered. In [16], a complete operation management for nonlinear PMSMs including MTPC, MC, FW and MTPV is presented. An iterative algorithm finds the optimal stator currents by evaluating machine maps (e.g., flux linkages) online. Iron losses are neglected. The algorithm presented in [17] achieves MTPC, MC and FW by “incremental current setpoint shaping” (online linearization of torque, voltage and current limit). However, current-dependent inductances, magnetic cross-coupling effects or iron losses are not considered. In [18], an MTPC and FW based speed controller is derived using Lyapunov’s direct method. The controller is combined with online parameter estimation of inductance, torque and viscous friction but iron losses, magnetic cross-coupling and saturation effects are not covered. In [24], optimal saliency ratio and power factor are considered during machine design to minimize copper and iron losses for high efficiency operation of nonlinear IPMSMs. Magnetic cross-coupling is not considered. In [22], MTPC, MC, FW and MTPV are discussed in combination with model predictive direct torque control. Iron losses, magnetic cross-coupling and saturation effects are neglected. In [25], a unified theory for OFTC is presented which allows to analytically compute the optimal reference currents for MTPC, MC, FW, MTPV and Maximum Torque per Flux (MPTF) while magnetic cross-coupling and saturation effects are considered. Iron losses are neglected. The approach has been extended in [4] by MTPL to account for iron losses as well. In [5], MTPL, FW, MC and MTPV are introduced for current and speed-dependent iron losses and the optimal reference currents are obtained analytically but iteratively due to the necessary online linearization of the nonlinear optimization problem.

1.2.2. Artificial Neural Networks in Electrical Drives

As already mentioned in the motivation, there exists a huge variety of ANN-based approaches in electrical drive systems (recall the overview preprint [30] with more than 250 references). Although [30] is very comprehensive, in the following, several additional and exemplary applications of ANNs in electrical drives are (re-)discussed in more detail to highlight that ANN-based OFTC has neither been considered nor published this far.

ANN-based speed and/or current controllers are the most common applications of ANNs in electrical drives. [31] proposes an ANN-based speed control method for PMSMs. The ANN has two inputs (actual and previous machine speed) and one output (the predicted machine speed) and is trained to identify the nonlinear machine dynamics to improve control performance. [32] presents an ANN-based model reference adaptive controller (MRAC) whose controller parameters are adjusted with the help of an ANN-based state observer performing online model identification. The closed-loop system performance shows negligible overshooting and fast rise times under various loading conditions and speed reversals. In [33], an ANN-based speed controller for a PMSM has been implemented, where the ANN is used to predict the actual machine torque which is then fed to a classical field-oriented current control system with flux observer. In [34], the PI speed controller for a PMSM is also replaced by an ANN-based speed controller, whereas an ANN-based PID speed controller is proposed in [35]. In [36], both, PI speed controller and PI current controllers, are replaced by ANN-based controllers. In general, overshoot as well as settling time were improved by the proposed ANN-based controllers. [37] proposes a real-time capable ANN-based controller for a linear tubular permanent magnet direct current motor (LTPMDCM) prototype. The used multilayer perceptron (MLP) recurrent neural network (RNN) is trained with conventional backpropagation. [38] presents an ANN-based speed controller for induction motor drives using an ANN with adaptive linear neurons (ADALINE) trained with the Widrow-Hoff learning rule. The closed-loop

system shows a significantly improved tracking performance compared to the standard PI speed controller.

In [44], it proposes an ANN-based direct torque control method for synchronous motors. The proposed ANN controller outputs optimal voltages to increase the motor's efficiency. However, losses or OFTC with ORCC are neither considered nor discussed. In [45], an ANN is utilized to enable long-horizon finite control set model predictive control (FCS-MPC) for IPMSM drive systems. The ANN can identify the optimal switching states with high accuracy (85–90%) which allows for a real-time capable implementation of the long-horizon FCS-MPC algorithm. In [46], the weighting factors of the cost functions of a predictive torque controller (PTC) were calculated by ANNs which not only improved the control function but also the calculation time and the computational effort.

Besides ANN-based controllers, ANN-based parameter and/or system identification is very popular. In [39], a feedforward ANN with sigmoid activation functions is used to identify the electro-magnetic dynamics for stator current and rotor speed control of induction machines. The ANN is trained by arbitrarily injected noise signals during closed-loop operation. In [40], an ANN with two hidden layers and sigmoid activation functions is used to identify the nonlinear DC motor drive dynamics to implement a nonlinear speed controller for the electrical drive system. The ANN is trained online by backpropagation with adjustable learning rate. [41] proposes an ANN with radial basis functions (RBF) for the identification of the rotor resistance and the estimation of the rotor speed of induction machines. The ANN is trained online during closed-loop operation.

In [42], recurrent and convolutional neural networks (RNNs & CNNs) are utilized and compared for temperature estimation in the stator (resistance) and rotor (permanent magnet) of PMSMs. The ANN's optimal architecture (number of layers and neurons) is found by Bayesian optimization. Nevertheless, only local minima are reached. The ANNs are implemented on a Raspberry Pi allowing for real-time execution. In [43], an ANN is trained to predict the wire insulation resistance with respect to its aging time and temperature. Compared to conventional thermal qualification methods used in electrical machines, the ANN-based approach saves about 57% of the time needed for a conventional accelerated aging test campaign.

An ANN-based torque observer has been implemented in [47] achieving very high accuracies (up to 98%) in the observed torque using a relatively small ANN with only one hidden layer. The observed torque is fed to a PI controller which outputs the q -current reference for the underlying current PI controllers. The d -current reference is computed based on the q -current reference with a conventional MTPC method (ANN is not involved).

In [48], an artificial recurrent neural network (RNN) with long short-term memory (LSTM) is proposed to predict three-phase current waveforms to account for parameter uncertainties in PMSM modeling. The prediction error reduces with the number of learning epochs and is minimized to deviations below 3.0% after 7000 iterations.

Another important application of ANNs is in the field of encoderless (or sensorless) control of electrical drives. In [49], it proposes a feedforward neural network (FNN) with one hidden layer capable of achieving sensorless speed control of an induction machine. The rotor angle estimation errors during speed transients are lower than 3%. In [50], structured ANNs are used for saliency-tracking-based sensorless control of AC drives. The ANN designs are based on physical system knowledge leading to a fixed number of layers and neurons with physically motivated interconnections. Due to their structured nature, the trained ANNs provide physical insights as activation functions and weights have physical meaning. In [51], the rotor position of a switched reluctance machine is obtained with the help of an ANN used to estimate the unsaturated stator inductances. The proposed ANN has a very simple structure and reduces the computational effort significantly compared to conventional encoderless methods. In [52], it describes advantages and limitations of ANN-based encoderless control. In particular, at zero speed, the ANN-based encoderless control approach must be extended by conventional high-frequency signal injection methods. The

additional consideration of the DC-link voltage improves control performance during field weakening. The used ANN has one hidden layer and is trained by supervised learning.

In many electrical drive systems, torque ripples induced by cogging torques or non-sinusoidal back-electromotive forces (back-EMFs) must be suppressed to their minimum to improve system performance. In [53], it proposes the utilization of ANNs for torque ripple reduction in PMSM based electrical drive systems with significant cogging torque and non-sinusoidal back-EMF. The structured ANN takes geometry (presence of harmonics in back-EMF and cogging torque) into account and are implemented as ADALINE-ANN-based speed or torque controllers. Alternatively, in [54], an unstructured ANN with one hidden layer in combination with a voltage matching circuit (VMC) is used to compute optimal state-feedback controller parameters online in order to minimize the torque ripple factor of a PMSM. To realize the VMC, an additional buck converter with state-feedback dc-link controller must be installed in the electrical drive system. The approach is simulatively tested for a two-level and three-level voltage source inverter.

To increase live time and reduce maintenance costs, condition monitoring and fault detection in electrical drive systems became very popular in the research community recently. In [55], an MLP-based ANN is used for fault detection. The approach is capable of proper fault/non-fault identification by different fault index evaluations. Detectable faults are e.g., significant changes in the stator windings or open-phase faults. For ANN training, data from simulations, machine design and closed-loop experiments was used. In [56], it shows that probabilistic neural networks (PNNs) or radial basis function networks (RBFNs), both with more than 100 hidden layers, are capable of detecting rotor failures such as broken rotor bars, bearing damage or air gap eccentricity. In contrast to the commonly used motor current signature analysis (MCSA), axial flux monitoring and vibration monitoring, the two proposed ANN-based methods allow to detect broken rotor bar faults online with very high accuracy even under different voltage supplies. In [57], a convolutional neural network (CNN) based condition monitoring system for permanent magnet synchronous motors with interturn and demagnetization faults is proposed. The CNN-based condition monitoring system is able to detect those motor faults in the deteriorated current response with extremely high accuracy (errors less than 0.15%). In [58], it presents a deep neural network (DNN) based condition monitoring and data evaluation system for highly automated laboratory test benches. The proposed approach aims at a fully automated implementation of predictive maintenance, test process automation, fault detection and downtime reduction of motor test stands. In the paper, a first study shows that fault detection and failure type classification can be achieved by the proposed DNN-based condition monitoring system.

Another important aspect in drive control are loss minimization in the power electronic devices. In [59], an ANN is trained by supervised learning and used to reduce voltage harmonics induced by low switching frequency modulation. The obtained ANN-based optimal pulse patterns (OPPs) are then implemented and compared to conventional patterns obtained by pulse width modulation. The ANN-based OPPs reduces harmonic losses in the induction machine-based electrical drive system in V/f operation and its efficiency can be improved by roughly 1%.

Finally, ANN-based multi-objective optimization during machine design has been proposed in [60]. The ANN is used to reduce the need for excessive finite element (FE) simulations in order to minimize computational effort and, hence, to find pareto optimal machine designs in a simpler and faster manner. To do so, the ANN is trained by extensive FE simulation data.

1.2.3. Summary of Detailed Literature Review

In conclusion, the detailed literature review above underpins the absolute novelty of the proposed ANN-based OFTC approach. This far, ANNs have not yet been proposed, implemented or tested for OFTC or ORCC in particular. Hence, to the best knowledge of the authors, this publication is the very first in this regard and of its kind. Therefore,

the goal of this publication is an initial but comprehensive description of the idea and a feasibility study and performance comparison of ANN-based OFTC with state-of-the-art OFTC approaches.

1.3. Problem Statement

The main goal of OFTC is to analytically (or numerically) compute (or to look up) the *optimal* and *feasible* reference currents

$$\mathbf{i}_{s,ref}^{dq}(m_{m,ref}, \hat{u}_{s,max}, \hat{i}_{s,max}, \omega_p, \dots) = \begin{pmatrix} i_{s,ref}^d(m_{m,ref}, \hat{u}_{s,max}, \hat{i}_{s,max}, \omega_p, \dots) \\ i_{s,ref}^q(m_{m,ref}, \hat{u}_{s,max}, \hat{i}_{s,max}, \omega_p, \dots) \end{pmatrix} \quad (1)$$

which are functions of the reference torque $m_{m,ref}$, the voltage limit $\hat{u}_{s,max}$, the current limit $\hat{i}_{s,max}$, the electrical angular velocity ω_p and possibly also of the electrical angle ϕ_p , the stator temperature ϑ_s and the rotor temperature ϑ_r to account for e.g., slotting effects and temperature dependency of stator (winding) resistance(s) and permanent magnet, respectively.

OFTC consists of two steps: Optimal reference current computation (ORCC) and optimal operation strategy selection (OOSS) which, depending on the actual operating conditions, allows to select the optimal operation strategy such as MTPC, MTPL, FW, MC or MTPV as illustrated in Figure 1.

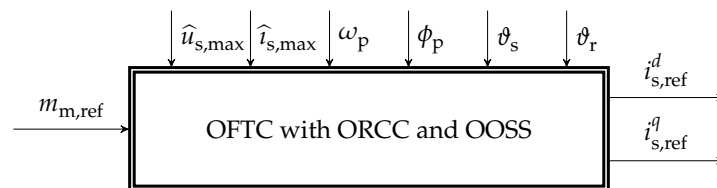


Figure 1. Optimal feedforward torque control (OFTC) with optimal reference current computation (ORCC) and optimal operation strategy selection (OOSS): The reference currents depend on desired (reference) torque, machine constraints, actual operating conditions and selected operation strategy (such as MTPL, FW, MC or MTPV).

The overall block diagram of a typical control system with OFTC and current controllers of an electrical drive consisting of IPMSM and inverter is shown in Figure 2.

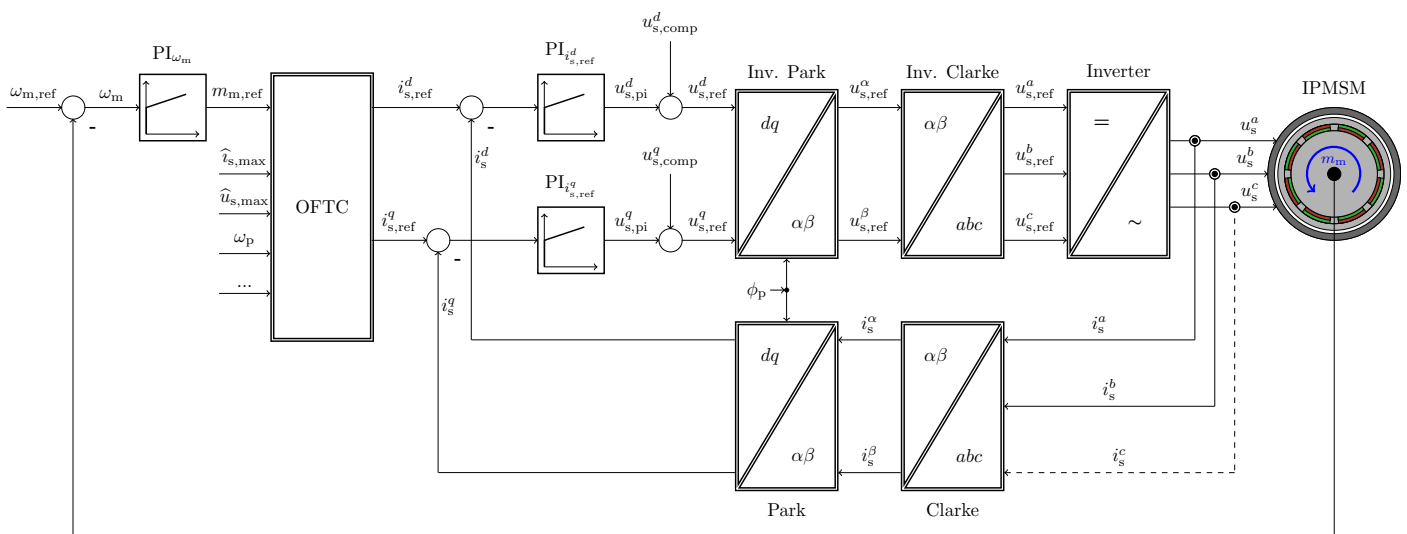


Figure 2. Block diagram of the control system with OFTC and underlying current controllers of a typical electrical drive system consisting of IPMSM and inverter.

The reference currents $(i_{s,ref}^d, i_{s,ref}^q)$, obtained from OFTC, are then fed to the underlying current control system with feedforward disturbance compensation $(u_{s,comp}^d, u_{s,comp}^q)$ eliminating cross-coupling terms in the current dynamics (see e.g., [61,62]). The reference voltages $(u_{s,ref}^d, u_{s,ref}^q)$ are the sum of the outputs $(u_{s,pi}^d, u_{s,pi}^q)$ of the PI controllers and the compensation voltages $(u_{s,comp}^d, u_{s,comp}^q)$. The references are then transformed back to the three-phase reference voltages $(u_{s,ref}^a, u_{s,ref}^b, u_{s,ref}^c)$ which are fed to the modulator of the inverter to obtain duty cycles or switching (gate) signals for the power semiconductors. Speed and angle are assumed to be measured and are available for feedback.

To solve the OFTC problem and to compute the optimal reference currents as in (1), a precise machine model must be available. A generic, dynamic transformer-like model for synchronous machines (SMs; considering magnetic cross-coupling and saturation and iron losses) can be used [4,5]. The electrical equivalent circuit of the stator windings and the iron core is shown in Figure 3: Similar to a transformer, the stator windings are magnetically coupled with the stator iron core by the stator flux linkages $\psi_s^{dq} = (\psi_s^d, \psi_s^q)^\top$.

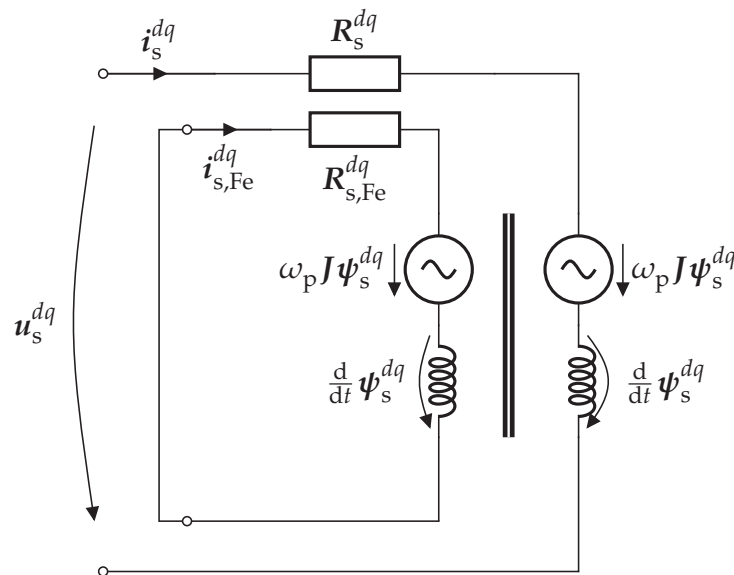


Figure 3. Electrical equivalent circuit of the stator windings and the magnetically coupled iron core in the (d, q) -reference frame.

The governing stator and stator iron voltage equations are, respectively, given by

$$\left. \begin{aligned} \mathbf{u}_s^{dq} &= \mathbf{R}_s^{dq} \mathbf{i}_s^{dq} + \omega_p \mathbf{J} \psi_s^{dq} + \frac{d}{dt} \psi_s^{dq} \\ \mathbf{0}_2 &= \mathbf{R}_{s,Fe}^{dq} \mathbf{i}_{s,Fe}^{dq} + \omega_p \mathbf{J} \psi_s^{dq} + \frac{d}{dt} \psi_s^{dq}, \end{aligned} \right\} \quad (2)$$

with stator voltages $\mathbf{u}_s^{dq} := (u_s^d, u_s^q)^\top$, stator currents $\mathbf{i}_s^{dq} := (i_s^d, i_s^q)^\top$, iron currents $\mathbf{i}_{s,Fe}^{dq} := (i_{s,Fe}^d, i_{s,Fe}^q)^\top$, and electrical speed $\omega_p = n_p \omega_m$ (i.e., pole pair number n_p times mechanical angular velocity ω_m ; see (3)). Furthermore, \mathbf{R}_s^{dq} and $\mathbf{R}_{s,Fe}^{dq}$ (both $\in \mathbb{R}^{2 \times 2}$; all entries may be non-zero) denote the resistance matrices of stator windings and iron core, respectively. Often, if the windings are symmetric (i.e., the resistances are identical in all phases), the matrices simplify to diagonal matrices $\mathbf{R}_s^{dq} = R_s \mathbf{I}_2$ and $\mathbf{R}_{s,Fe}^{dq} = R_{s,Fe} \mathbf{I}_2$ with scalar stator R_s and stator iron $R_{s,Fe}$ resistances. The dynamics of the mechanical subsystem can be described as follows

$$\left. \begin{aligned} \frac{d}{dt} \omega_m &= \frac{1}{\Theta_m} (m_m - m_1) \\ \frac{d}{dt} \phi_m &= \omega_m, \end{aligned} \right\} \quad (3)$$

with mechanical angular velocity $\omega_m = \frac{\omega_p}{n_p}$, mechanical angle ϕ_m , inertia constant Θ_m , load torque m_l and (nonlinear) machine torque (derivation details omitted, see [4])

$$m_m = \frac{2n_p}{3\kappa^2} (\mathbf{i}_s^{dq} + \mathbf{i}_{s,Fe}^{dq})^\top \mathbf{J} \boldsymbol{\psi}_s^{dq} = \frac{2n_p}{3\kappa^2} \left(\mathbf{i}_s^{dq} - (\mathbf{R}_{s,Fe}^{dq})^{-1} (\omega_p \mathbf{J} \boldsymbol{\psi}_s^{dq} + \frac{d}{dt} \boldsymbol{\psi}_s^{dq}) \right)^\top \mathbf{J} \boldsymbol{\psi}_s^{dq}. \quad (4)$$

The constant $\kappa \in \{\frac{2}{3}, \sqrt{\frac{2}{3}}\}$ is used to choose between amplitude or power invariant Clarke transformation, respectively. The right-hand side of (4) is obtained by solving (2) for $\mathbf{i}_{s,Fe}^{dq}$ and inserting the result into the middle part of (4). The sum $\mathbf{i}_s^{dq} + \mathbf{i}_{s,Fe}^{dq}$ of the stator and iron currents can be interpreted as *magnetizing current*. The machine torque of IPMSMs is highly nonlinear; an exemplary torque map is illustrated in Figure 4a. It shows the nonlinear torque of a 4 kW IPMSM (used later for implementation) over the current locus overlaid by an exemplary reference torque (gray shaded plane) and the intersection torque hyperbola (black line) with the admissible currents pairs which can produce this desired torque. Moreover, several torque hyperbolas for several reference torques are projected to the current locus (colored lines). Figure 4b depicts the speed-dependent iron losses of the 4 kW IPMSM. The current-dependent flux linkages $\psi_s^d(i_s^d, i_s^q)$ and $\psi_s^q(i_s^d, i_s^q)$ of this machine are shown in Figure 4c,d, respectively.

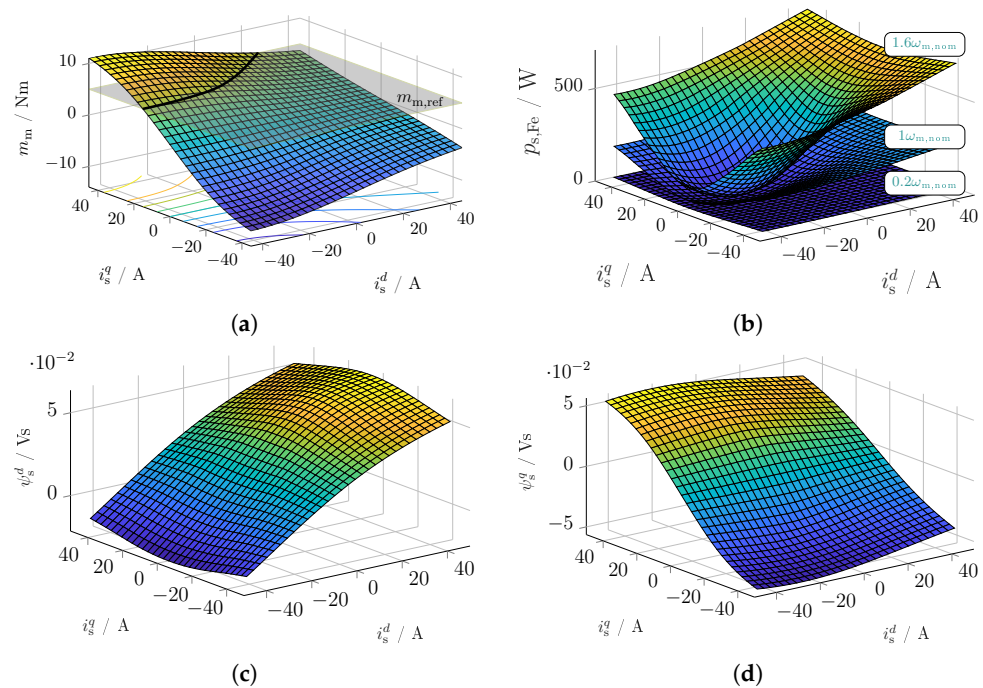


Figure 4. Illustration of machine torque, iron loss and flux linkage nonlinearities of a 4 kW IPMSM (losses are shown for different speeds). (a) Nonlinear machine torque. (b) Nonlinear iron losses. (c) Nonlinear *d*-flux linkage. (d) Nonlinear *q*-flux linkage.

Remark 1 (Machine nonlinearities). *In the most general case, stator flux linkages are nonlinear functions of currents, speed, angle and temperatures, i.e., $\psi_s^{dq} = \psi_s^{dq}(i_s^{dq}, \omega_p, \phi_p, \vartheta_s, \vartheta_r)$. Similarly, the stator and stator iron resistance matrices may depend on currents (proximity and skin effect), speed, angle (asymmetric case) and stator temperature, i.e., $\mathbf{R}_s^{dq} = \mathbf{R}_s^{dq}(i_s^{dq}, \omega_p, \phi_p, \vartheta_s)$ and $\mathbf{R}_{s,Fe}^{dq} = \mathbf{R}_{s,Fe}^{dq}(i_s^{dq}, \omega_p, \phi_p, \vartheta_s)$, respectively. To simplify notation, those dependencies (arguments) are dropped in the remainder of this paper but should be kept in mind as they highlight the complexity and nonlinearity of the OFTC problem.*

With the electrical and mechanical model, i.e., (2) and (3) above, the ORCC iteratively solves the nonlinear optimization problem (NLP) with two inequality constraints and one equality constraint, given by

$$\mathbf{i}_{s,\text{ref}}^{dq} = \arg \min_{\mathbf{i}_s^{dq}} p_{s,L} \tag{5a}$$

$$\text{s.t.} \quad \|\mathbf{i}_s^{dq}\| \leq \hat{i}_{s,\text{max}}, \tag{5b}$$

$$\|\mathbf{u}_s^{dq}\| \leq \hat{u}_{s,\text{max}}, \tag{5c}$$

$$m_m = \bar{m}_{m,\text{ref}} = \text{sat}_{m_{m,\text{max}}}(m_{m,\text{ref}}) \tag{5d}$$

where stator copper losses *and* stator iron losses (see Figure 4b)

$$p_{s,L} := \underbrace{(\mathbf{i}_s^{dq})^\top \mathbf{R}_s^{dq} \mathbf{i}_s^{dq}}_{=:p_{s,\text{Cu}} \text{ (copper losses)}} + \underbrace{(\mathbf{i}_{s,\text{Fe}}^{dq})^\top \mathbf{R}_{s,\text{Fe}}^{dq} \mathbf{i}_{s,\text{Fe}}^{dq}}_{=:p_{s,\text{Fe}} \text{ (iron losses)}} \tag{6}$$

are minimized while the machine constraints, i.e., the current limit $\hat{i}_{s,\text{max}}$ in (5b) and the voltage limit $\hat{u}_{s,\text{max}}$ in (5c) and, must not be violated and the saturated (feasible) reference torque $\bar{m}_{m,\text{ref}}$ must be produced. The reference torque $\bar{m}_{m,\text{ref}}$ must be saturated to the sign-correct maximally available (feasible) machine torque

$$m_{m,\text{max}} := \max_{\mathbf{i}_s^{dq}} \text{sign}(m_{m,\text{ref}}) \cdot m_m \tag{7}$$

as *not* all reference torques are feasible and, hence, can not be produced due to the current and/or voltage constraints in (5b) and (5c). The per-unit machine constraints and the reference torque are illustrated in Figure 5 in the current locus as (a) *current circles* for multiples of the rated current magnitude $i_{s,R}$, (b) *voltage ellipses* for multiples of rated electrical angular velocity $\omega_{p,R}$ and (c) (*reference*) *torque hyperbolas* for multiples of rated machine torque $m_{m,R}$.

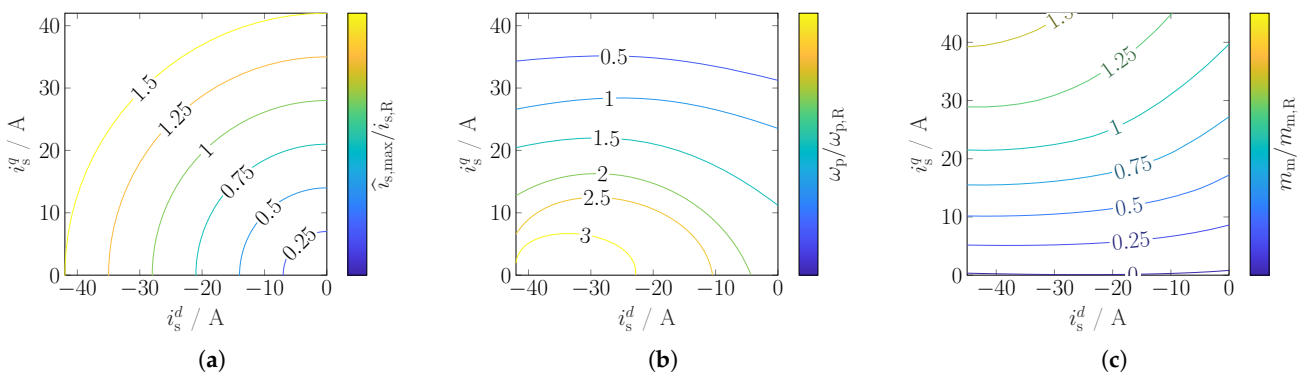


Figure 5. Illustration of machine constraints and reference torque in current locus for multiples of rated current $i_{s,R}$, angular velocity $\omega_{p,R}$ and machine torque $m_{m,R}$. (a) Current circles. (b) Voltage ellipses. (c) Torque hyperbolas.

Remark 2 (Nonideal voltage ellipses and reference torque hyperbolas). *Voltage ellipses and reference torque hyperbolas in the current locus can (locally) be represented as quadrics of the form $(\mathbf{i}_s^{dq})^\top \mathbf{A} \mathbf{i}_s^{dq} + 2 \mathbf{a}^\top \mathbf{i}_s^{dq} + \alpha$ with constant matrix $\mathbf{A} \in \mathbb{R}^{2 \times 2}$, vector $\mathbf{a} \in \mathbb{R}^2$ and scalar $\alpha \in \mathbb{R}$ [4]. However, for nonlinear machines with highly current dependent magnet cross-coupling and saturation, these global “ellipses” and “hyperbolas” are actually distorted (see Figure 5) and $(\mathbf{i}_s^{dq})^\top \mathbf{A}(\mathbf{i}_s^{dq}) \mathbf{i}_s^{dq} + 2 \mathbf{a}(\mathbf{i}_s^{dq})^\top \mathbf{i}_s^{dq} + \alpha(\mathbf{i}_s^{dq})$ holds where matrix, vector and scalar depend on the currents as well. That is why an online linearization must be performed which eventually only allows solving the NLP iteratively.*

1.4. Proposed Solution

In this paper, in contrast to the conventional (LUT-based or numerically or analytically computed) OFTC approaches, an ANN-based OFTC for IPMSMs is proposed. Basically, the OFTC block in Figure 2 is replaced by a properly *designed, trained and validated* ANN as illustrated in Figure 6. Therefore, the modularity of the control system is preserved and the implementation effort is minimized. In addition, for the implementation of the ANN, (*data preprocessing*) is required in order to accelerate and improve training and validation and to assure the real-time capability of the ANN. The ANN will compute analytically but recursively the desired optimal reference currents which then are fed to the inner current control loop. The ANN-based OFTC is performed based on (a) its inputs (such as voltage and current constraints, reference torque, electrical angular velocity, etc.) and (b) its input, hidden and output ANN layers with simple but trained activation functions. The aimed at advantages of the ANN-based OFTC strategy are (i) a simple implementation, (ii) a faster computation of the reference currents than with classical ORCC (as no iterations are required) and (iii) no OOSS is necessary as the trained ANN directly outputs the optimal reference currents without following a decision tree (as required for conventional OFTC; see e.g., [4,25]).

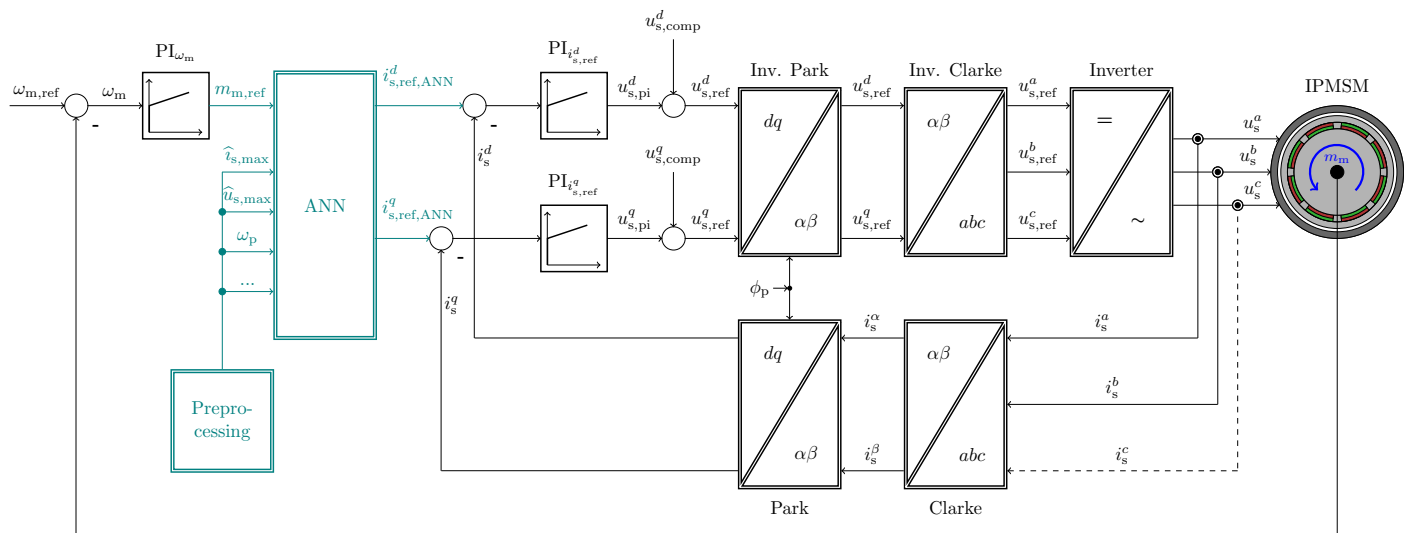


Figure 6. Block diagram of the *newly proposed* control system with ANN-based OFTC and underlying current controllers of a typical electrical drive system consisting of IPMSM and inverter.

2. Artificial Neural Network based Optimal Feedforward Torque Control

To replace ORCC by an Artificial Neural Network (ANN; ANNs are often also called Multi-Layer Perceptron (MLP) networks, see Section 3.10 in [63]), the ANN must be properly designed such that input, hidden and output layers have adequate size and the used activation functions are simple but powerful enough to be capable of reduplicating the nonlinear machine behavior. After ANN design, it must be trained and validated by properly selected training and validation data sets. The data sets must be created and preprocessed with care such that ANN training and ANN validation are feasible.

For the present application, the following specifications are imposed on ANN design, training and validation:

- The approximation accuracy of the ANN for OFTC must be (very) high, i.e., approximation errors should be smaller than 1%;
- Assuming the required data is available, there are no stringent restrictions on training and validation as it is performed offline, i.e., there are (almost) no computational or memory storage constraints during training and validation; and

- The trained ANN should be simple to implement and run in real-time, i.e., a rather small feedforward network architecture (i.e., without dynamics; no recurrent network) with a low number of neurons and one or (at most) two hidden layers should be used. In the following, ANN design, training and validation are explained in more detail.

2.1. Artificial Neural Network Design

The goal of the ANN used for OFTC is to approximate the nonlinear behavior

$$\mathbf{y} := \begin{pmatrix} i_{s,\text{ref}}^d \\ i_{s,\text{ref}}^q \\ i_{s,\text{ref}}^0 \end{pmatrix} = \mathbf{f}(m_{m,\text{ref}}, \hat{u}_{s,\text{max}}, \hat{i}_{s,\text{max}}, \omega_p, \dots) = \mathbf{f}(\mathbf{x}) \quad (8)$$

between reference torque, physical constraints and operating conditions (the inputs collected in the vector $\mathbf{x} := (x_1, \dots, x_m)^\top$; recall Figure 1) and the optimal reference currents (the outputs collected in the vector $\mathbf{y} := (y_1, \dots, y_n)^\top$). As the nonlinear function $\mathbf{f}(\cdot)$ is not known, the ANN must be trained properly to obtain approximated output $\hat{\mathbf{y}}$ and approximating function $\hat{\mathbf{f}}(\cdot)$ (approximated quantities are indicated by hat) such that the following holds

$$\hat{\mathbf{y}} = \begin{pmatrix} i_{s,\text{ref,ANN}}^d \\ i_{s,\text{ref,ANN}}^q \\ i_{s,\text{ref,ANN}}^0 \end{pmatrix} = \hat{\mathbf{f}}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}). \quad (9)$$

The ANN output $\hat{\mathbf{y}}$ is (i) for binary classification tasks typically one or zero and (ii) for non-binary classification or regression tasks a numerical value. Every input x_i with $i \in \{1, \dots, n\}$ of the input vector \mathbf{x} has a different impact, also called weight, on the outcome of the neural network.

2.1.1. Preliminaries: Neurons, Layers and Recursive Output Computation

In general, an ANN consists of several layers and each layer of several neurons. Each neuron of the j -th layer is fed by the m_j -dimensional input vector $\mathbf{x}_j := (x_{j,1}, \dots, x_{j,m_j})^\top \in \mathbb{R}^{m_j}$. The i -th neuron of the j -th layer is equipped with its respective activation function $\Phi_{j,i}(\cdot)$ [see Section 2.1.2], which processes the input vector \mathbf{x}_j —weighted by the weight vector $\mathbf{w}_j := (w_{j,1}, \dots, w_{j,m_j})^\top \in \mathbb{R}^{m_j}$ and biased by the bias $b_{j,i} \in \mathbb{R}$ in order to compute the neuron's output

$$\hat{y}_{j,i} = \Phi_{j,i}(\mathbf{w}_{j,i}^\top \mathbf{x}_j + b_{j,i}) \in \mathbb{R}. \quad (10)$$

Therefore, the j -th layer with n_j neurons generates the output vector

$$\hat{\mathbf{y}}_j = \begin{pmatrix} \hat{y}_{j,1} \\ \vdots \\ \hat{y}_{j,n_j} \end{pmatrix} = \begin{pmatrix} \Phi_{j,1}(\mathbf{w}_{j,1}^\top \mathbf{x}_j + b_{j,1}) \\ \vdots \\ \Phi_{j,n_j}(\mathbf{w}_{j,n_j}^\top \mathbf{x}_j + b_{j,n_j}) \end{pmatrix} = \Phi_j(\mathbf{W}_j \mathbf{x}_j + \mathbf{b}_j) \in \mathbb{R}^{n_j} \quad (11)$$

where

$$\mathbf{W}_j := \begin{bmatrix} \mathbf{w}_{j,1}^\top \\ \vdots \\ \mathbf{w}_{j,n_j}^\top \end{bmatrix} \in \mathbb{R}^{n_j \times m_j} \quad \text{and} \quad \mathbf{b}_j := \begin{pmatrix} b_{j,1} \\ \vdots \\ b_{j,n_j} \end{pmatrix} \in \mathbb{R}^{n_j} \quad (12)$$

collect the weights and biases in compact form in the overall weighting matrix and bias vector of the j -th layer, respectively. The overall output vector $\hat{\mathbf{y}}$ of the ANN then is recursively defined by the outputs of the activation functions in the different layers; i.e., for an ANN with L layers and input vector \mathbf{x} , it is given by

$$\hat{y} = \Phi_L \left(\underbrace{W_L \Phi_{L-1} \left(\underbrace{W_{L-1} \Phi_{L-2} \left(\cdots \Phi_2 \left(\underbrace{W_2 \Phi_1 \left(W_1 x + b_1 \right) + b_2}_{=\hat{y}_1} \right) + b_{L-1}}_{=\hat{y}_2} \right) + b_{L-1}}_{=\hat{y}_{L-1}} \right) + b_L \right), \quad (13)$$

where the first layer is the input layer, the second to the $L - 1$ -th layers are hidden layers and the L -th layer is the output layer. From (13), the recursive nature of the output computation becomes clear. The output vector \hat{y}_{j-1} of the preceding $j - 1$ -th layer becomes the input to the j -th layer. The output vector \hat{y}_L of the L -th layer represents the overall output \hat{y} of the neural network. For further details, the interested reader is referred to e.g., [64] or [63].

2.1.2. Activation Functions

The core of each ANN are its neurons with their respective activation function $\Phi(\cdot)$; recall (10), (11) and (13). Depending on the location of the neuron in the input, hidden or output layer, different activation functions have proven to be useful. Most common activation functions [64], such as sigmoid, tangens hyperbolicus, signum (sign), identity, rectified linear unit (ReLU) and saturation are illustrated in Figure 7. Their mathematical definitions with respective derivatives are collected in Table 1.

For input and output layer, simple activation functions such as the identity are most common whereas for the hidden layers typically sigmoid or tangens hyperbolicus functions are used [64].

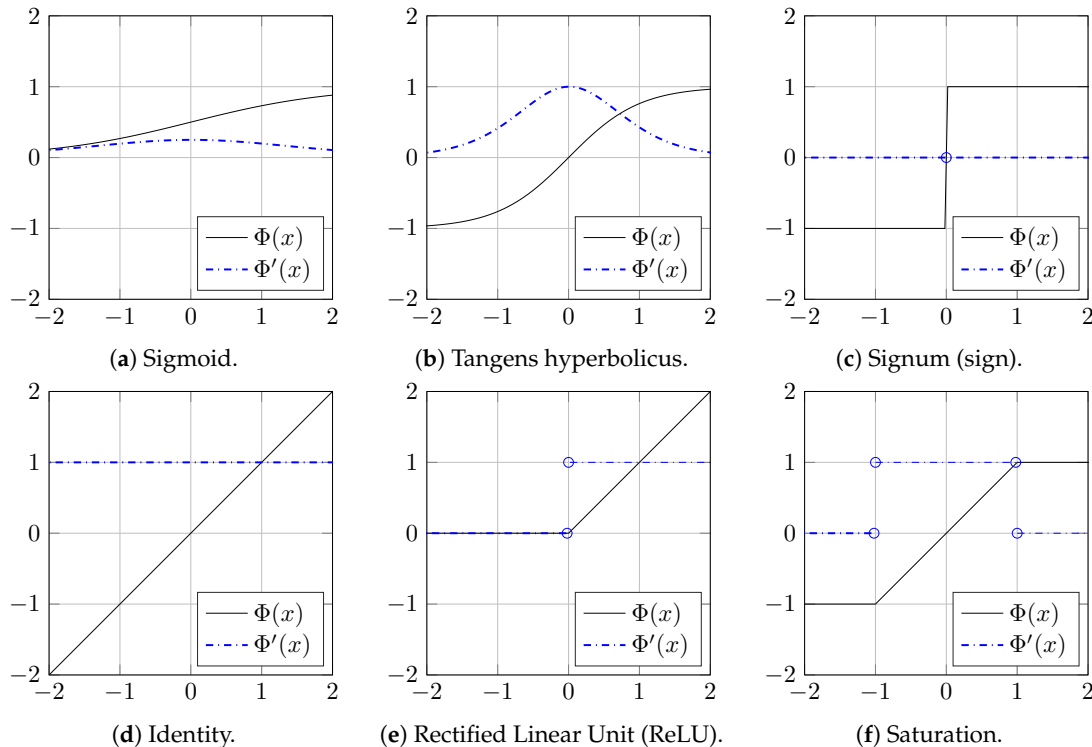


Figure 7. Illustration of different activation functions and their derivatives, i.e., (a) $\Phi(x) = \Phi_{\text{sig}}(x)$, (b) $\Phi(x) = \Phi_{\text{tanh}}(x)$, (c) $\Phi(x) = \Phi_{\text{sgn}}(x)$, (d) $\Phi(x) = \Phi_{\text{id}}(x)$, (e) $\Phi(x) = \Phi_{\text{ReLU}}(x)$ and (f) $\Phi(x) = \Phi_{\text{sat}}(x)$ (for mathematical definitions, see Table 1).

Table 1. Mathematical definitions of different activation functions and their derivatives.

Function Name	Activation Function $\Phi(x)$	Derivative $\Phi'(x)$
Sigmoid	$\Phi_{\text{sig}}(x) = \frac{1}{1+e^{-x}}$	$\Phi'_{\text{sig}}(x) = \Phi_{\text{sig}}(x)(1 - \Phi_{\text{sig}}(x))$
Tangens hyperbolicus	$\Phi_{\text{tanh}}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\Phi'_{\text{tanh}}(x) = 1 - \Phi_{\text{tanh}}(x)^2$
Signum (sign)	$\Phi_{\text{sgn}}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\Phi'_{\text{sgn}}(x) = 0$ (not at $x = 0$)
Identity	$\Phi_{\text{id}}(x) = x$	$\Phi'_{\text{id}}(x) = 1$
Rectified Linear Unit	$\Phi_{\text{ReLU}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$\Phi'_{\text{ReLU}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Saturation	$\Phi_{\text{sat}}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$	$\Phi'_{\text{sat}}(x) = \begin{cases} 0 & \text{if } x < -1 \\ 1 & \text{if } -1 \leq x \leq 1 \\ 0 & \text{if } x > 1 \end{cases}$

2.1.3. Implemented Artificial Neural Network Architectures

For the considered application in electrical drive systems with small cycle times (<10 ms) sophisticated networks with hundreds of neurons, more than two hidden layers and complicated activation functions are (still) too complex for real-time implementation. In particular, implementation and evaluation of the sigmoid and tangens hyperbolicus activation functions and their derivatives (recall Table 1) are computationally demanding.

Therefore, in this paper, due to its low computational requirements, simple ReLU activation functions with their super simple derivative are utilized for the hidden layers. Input and output layer are equipped with the identity activation function as the approximation problem can be considered as a regression problem.

Regarding the layers, it is stated in [65] that a single hidden layer is sufficient to train the ANN to match “any non-linearity”. Following this statement, a network with one hidden layer is chosen as the minimum requirement and, for comparison, a network with two layers is implemented as well (the maximum depth of the hidden layers considered here).

For OFTC of IPMSMs, the ANN must output two reference currents and process at least four input variables. The number of input and output layer neurons should match the input and output dimensions. For this paper, four input variables, the reference torque, the (possibly varying) current and voltages limits and the electrical angular velocity, i.e.,

$$\mathbf{x} = (m_{m,\text{ref}}, \hat{u}_{s,\text{max}}, \hat{i}_{s,\text{max}}, \omega_p)^\top \in \mathbb{R}^4, \quad (14)$$

are considered. Clearly, more inputs such as angular position or rotor and stator temperature are feasible and might be of interest but are considered as future work. The output vector comprises the two optimal reference currents, i.e.,

$$\hat{\mathbf{y}} = (i_{s,\text{ref,ANN}}^d, i_{s,\text{ref,ANN}}^q)^\top \in \mathbb{R}^2, \quad (15)$$

and, therefore, the output layer consists of two neurons.

Lastly, the number of neurons must be defined. A sufficient amount of neurons in the hidden layer(s) to approximate one single (scalar) output varies between three (see [34,36]) and up to 100 (see [47,66]). To specify the number of neurons per hidden layer, the guidelines presented in [47] were adapted to obtain an initial number as starting point or initial guess. The formula

$$n_j = n_{\text{output}} \cdot \left(\sqrt{n_{i,j} + n_{o,j}} + h \right) \quad (16)$$

is proposed to calculate the approximate number n_j of neurons of the j -th hidden layer (for ANN-based OFTC, one (i.e., $j = 2$) or two hidden layers (i.e., $j \in \{2, 3\}$) were chosen). $n_{i,j}$ and $n_{o,j}$ represents the numbers of the input and output neurons of the j -th hidden layer, whereas n_{output} specifies the number of ANN outputs and $h \in \{1, \dots, 10\}$ is an integer. In [47], the formula was presented for a single output neuron (i.e., $n_{\text{output}} = 1$). Evaluating (16) for the considered case and all $h \in \{1, \dots, 10\}$ yields

- $n_2 \in \{6, \dots, 24\}$ for one hidden layer (i.e., $j = 2$) and
- $n_2, n_3 \in \{8, \dots, 28\}$ for two hidden layers (i.e., $j \in \{2, 3\}$).

Based on these initial guesses, several ANN designs with one and two hidden layers and different number of neurons per hidden layer have been implemented, trained and validated. The best compromise between accuracy and complexity were obtained for *twenty* neurons per hidden layer as illustrated in Figure 8 which shows exemplarily estimation accuracy (norm of estimation error) and floating point operations (FLOPs; i.e., the number of required computations incorporating summations and multiplications) over the number of neurons in the hidden layers of Architecture (A_2). Obviously, the estimation accuracy increases quickly (norm of estimation error reduces) until, for 20 hidden neurons, one of the lowest values of the error norm (i.e., $\|e\| \approx 0.2428$) is reached. For 23 neurons, a similar value is obtained with $\|e\| \approx 0.2425$ and, finally, for 28 neurons, the best accuracy is achieved with $\|e\| \approx 0.2343$. In contrast to that, the FLOPs increase quadratically with the increasing number of hidden neurons.

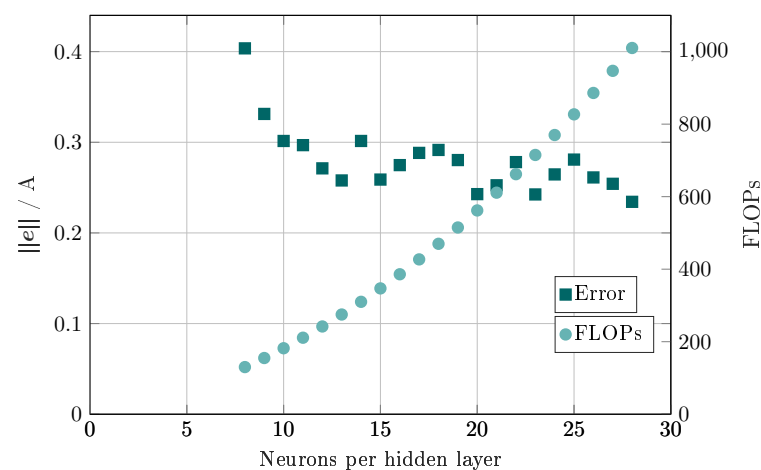
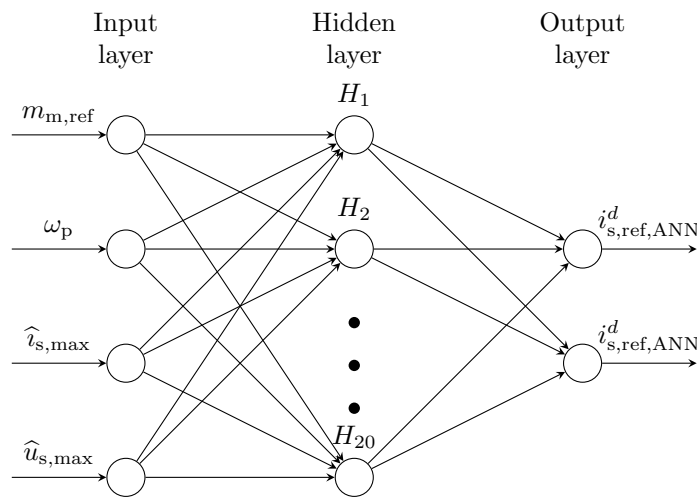
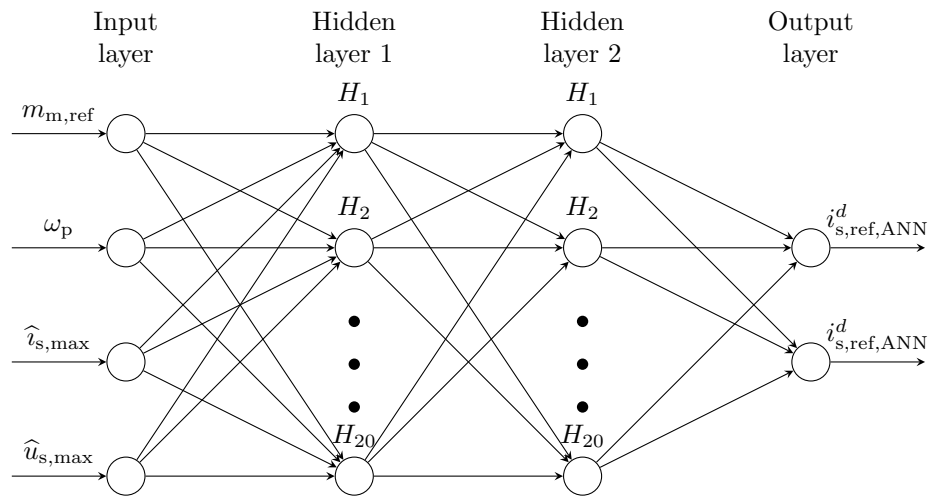


Figure 8. Illustration of estimation accuracy (norm of estimation error) and floating point operations of ANN Architecture (A_2) for different numbers of hidden layer neurons.

Based on the discussion above, two ANN architectures were implemented: Architecture (A_1) with three layers including one hidden layer and Architecture (A_2) with four layers including two hidden layers. Both are illustrated in Figure 9. The key data of both ANN designs is collected in Table 2.



(a) Architecture (A₁).



(b) Architecture (A₂).

Figure 9. Illustration of the two implemented ANN architectures: Each architecture has four input layer neurons and two output layer neurons; Architecture (A₁) has one hidden layer, whereas Architecture (A₂) has two hidden layers with 20 hidden layer neurons each.

Table 2. Key data of implemented ANN architectures (centered entries hold for both architectures).

Description	Architecture (A ₁)	Architecture (A ₂)
ANN type	Feedforward Neural Network	
Model prediction	Regression	
Training algorithm	Levenberg-Marquardt	
Error function	Mean Squared Error (MSE)	
Overall layers	3	4
Hidden layers	1	2
Input vector	$x := (m_{m,ref}, \hat{u}_{s,max}, \hat{i}_{s,max}, \omega_p)^\top$	
Input neurons	$n_1 = 4$	
Input layer activation function	$\Phi_{id}(x)$ [see Table 1] (without weighting and bias)	
Hidden layer activation function	$\Phi_{ReLU}(x)$ [see Table 1]	
Hidden layer neurons	$n_2 = 20$	$n_2 = n_3 = 20$
Output layer activation function	$\Phi_{id}(x)$ [see Table 1]	
Output layer neurons	$n_3 = 2$	$n_4 = 2$
Output vector	$\hat{y} := (i_{s,ref,ANN}^d, i_{s,ref,ANN}^d)^\top$	

2.2. Artificial Neural Network Training

Neural networks can be trained by using one of three training or learning methods which are supervised, unsupervised and reinforced learning (for details see e.g., [64] or [63]). Supervised learning requires (comprehensive) data, including both input and corresponding output data. Unsupervised learning mainly is used to cluster input data, therefore no output data is required. Reinforced learning needs the interaction of a machine or person that rewards the neural network if it is working as expected. In this paper, supervised learning is utilized, in order to train both ANN Architectures (A_1) and (A_2), as input and corresponding output data are available.

2.2.1. Preliminaries: Data Sets, Performance Measures and Training Algorithms

The available data must be collected and preprocessed. To do so, all individual (e.g., for each sampling instant n) input-output data sets

$$(\mathbf{y}[n]; \mathbf{x}[n])$$

are collected, for all $n \in \{1, \dots, N\}$, in the overall input-output data set

$$(\mathbf{Y}; \mathbf{X}) := ((\mathbf{y}[1]; \mathbf{x}[1]), \dots, (\mathbf{y}[N]; \mathbf{x}[N])),$$

which is then preprocessed, split into *training data set*

$$(\mathbf{Y}_T; \mathbf{X}_T) := ((\mathbf{y}_T[1]; \mathbf{x}_T[1]), \dots, (\mathbf{y}_T[N_T]; \mathbf{x}_T[N_T])) \subset (\mathbf{Y}; \mathbf{X})$$

of size $N_T < N$ and *validation data set*

$$(\mathbf{Y}_V; \mathbf{X}_V) := ((\mathbf{y}_V[1]; \mathbf{x}_V[1]), \dots, (\mathbf{y}_V[N_V]; \mathbf{x}_V[N_V])) \subset (\mathbf{Y}; \mathbf{X})$$

of size $N_V < N$ and finally used for ANN training and validation.

With these two data sets $(\mathbf{Y}_T; \mathbf{X}_T)$ and $(\mathbf{Y}_V; \mathbf{X}_V)$ at hand, training and validation of the ANN can be conducted. To analyze the learning and validation performance of the ANN, performances measures are required, which quantify the *approximation error*

$$\mathbf{e}[k] := \mathbf{y}[k] - \hat{\mathbf{y}}[k] \stackrel{(8),(11)}{=} \mathbf{f}(\mathbf{x}[k]) - \hat{\mathbf{f}}(\mathbf{x}[k]) \quad (17)$$

of the ANN for each data subset k (or each sampling instant k) chosen from the training set $(\mathbf{Y}_T; \mathbf{X}_T)$ or validation set $(\mathbf{Y}_V; \mathbf{X}_V)$ (i.e., $k \in \{1, \dots, K := N_T\}$ or $k \in \{1, \dots, K := N_V\}$). The performance measures for a regression neural network, such as the used ANNs, differ considerably from the accuracy measures for a classification network. Usually, the following performance measures (see e.g., [64,66] or [63]):

- *Mean Error (ME)*

$$\bar{e} := \frac{1}{K} \sum_{k=1}^K \mathbf{e}[k] = \frac{1}{K} \sum_{k=1}^K \mathbf{y}[k] - \hat{\mathbf{y}}[k] \quad (18)$$

- *Standard Error Deviation (SED)*

$$\sigma_e := \sqrt{\mathbf{e}[k]^2} = \sqrt{\frac{1}{K} \sum_{k=1}^K (\mathbf{y}[k] - \hat{\mathbf{y}}[k])^2} \quad (\text{computed component-wise}) \quad (19)$$

- *Quadratic Euclidean Distance (or Mean Squared Error (MSE))*

$$\mathbf{d}(\mathbf{e}[k]) := \|\mathbf{e}[k]\|^2 = \mathbf{e}[k]^\top \mathbf{e}[k] = (\mathbf{y}[k] - \hat{\mathbf{y}}[k])^\top (\mathbf{y}[k] - \hat{\mathbf{y}}[k]) \quad (20)$$

are used (i) to quantify the approximation error and its probability and (ii) to finally train and validate the regression network. Note that mean error and standard error deviation are computed (“averaged”) over *all* entries $k \in \{1, \dots, K\}$ of the chosen data set, whereas the quadratic Euclidean distance is evaluated for each *individual* entry k (as it is used for data training and adaption of the weights of the ANN).

Based on mean error and standard error deviation, usually three probability distribution regions are of interest:

- $\bar{e} \pm 1 \sigma_e = 68.7\%$ (Region I);
- $\bar{e} \pm 2 \sigma_e = 95.4\%$ (Region II); and
- $\bar{e} \pm 3 \sigma_e = 99.7\%$ (Region III);

as those cover pre-defined regions of the normal distribution [67] and are used later for performance analyses.

Finally, to train (or optimize) the ANN in order to update the weights and to minimize the approximation error (17) (more, precisely, its error norm, the Euclidean distance (20)), a training algorithm must be chosen [68]. Very often, the *Gradient descent* method is applied to update the weights by using the negative and by some $0 < \eta < 1$ scaled gradient of the error norm [63], Section 10.4.4, i.e.,

$$w[k + 1] := w[k] - \eta \underbrace{\frac{1}{2} \frac{d}{dw} \|e[k]\|^2}_{\text{gradient}} \stackrel{(20)}{=} w[k] - \eta \frac{1}{2} \frac{d}{dw} d(e[k]) \tag{21}$$

where all weights of all L layers (recall (12)) are collected in the overall weighting vector

$$w := \left(\underbrace{(w_{1,1}^\top, \dots, w_{1,n_1}^\top)}_{\text{1-st Layer}}, \dots, \underbrace{(w_{L,1}^\top, \dots, w_{L,n_L}^\top)}_{\text{L-th layer}} \right)^\top.$$

However, the main problem of the gradient descent method is its slow to moderate convergence speed. As alternative, the *Levenberg-Marquardt* algorithm has become popular to update the weights as follows [63], Section 4.1

$$w[k + 1] := w[k] - \underbrace{\left(J_C(w[k])^\top J_C(w[k]) + \mu I \right)^{-1}}_{\text{scaled \& approximated Hessian}} J_C(w[k])^\top e[k] \tag{22}$$

with Jacobian matrix

$$J_C(w[k]) := \frac{d}{dw} e[k],$$

identity matrix I (of appropriate dimensions) and scaled & approximated Hessian matrix $J_C(w[k])^\top J_C(w[k]) + \mu I$ scaled by some $\mu > 0$. The Levenberg-Marquardt algorithm represents a mixture between gradient descent and the Gauss-Newton method in order to combine both of their strengths. It uses the Gradient descent method to converge when the current error is far from its minimum and the Gauss-Newton method to converge quickly when near a local minimum. The downsides of this algorithm are the relatively high computational demand during training since the inverse of the (approximated) Hessian matrix must be computed for each sub-data set or sampling time k to update the weights. The Levenberg-Marquardt algorithm works very effectively for small(er) networks but loses against the gradient descent methods in terms of convergence speed for large (deep) networks.

In this paper, as rather small networks are considered and there are no restrictions on computational power, the Levenberg-Marquardt algorithm is used for ANN training. The ANN training (and validation) procedure is illustrated in Figure 10 for the two chosen ANN architectures.

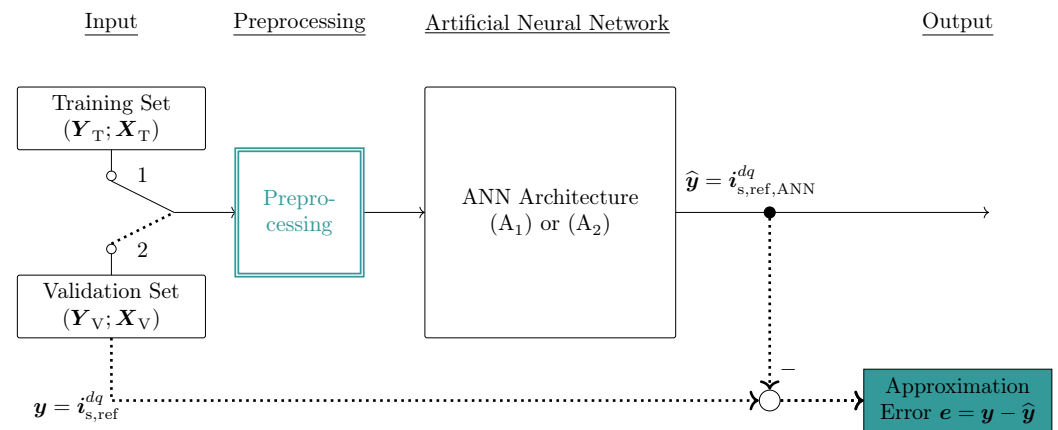


Figure 10. Illustration of ANN training and validation procedure, training set and validation set, respectively: Approximation error $\mathbf{e} := \mathbf{y} - \hat{\mathbf{y}} = (\mathbf{i}_{s,\text{ref}}^d, \mathbf{i}_{s,\text{ref}}^q)^\top - (\mathbf{i}_{s,\text{ref,ANN}}^d, \mathbf{i}_{s,\text{ref,ANN}}^q)^\top$ is the difference between the numerically computed optimal reference currents and the ANN outputs.

2.2.2. Data Set(s) Creation for Training (and Validation)

Before the ANNs can be trained and validated, in this subsection, it will be shown how, based on available measurement data or available data from Finite Element Analysis (FEA) of the IPMSM, the required data sets ($\mathbf{Y}_T; \mathbf{X}_T$) and ($\mathbf{Y}_V; \mathbf{X}_V$) for training and validation can be created, respectively. Starting points are

- the available rated machine parameters (see Table 3) such as e.g., pole pair number n_p , resistance R_s at rated temperature, moment of inertia Θ_m , rated torque $m_{m,R}$ and rated current magnitude $i_{s,R}$;
- the available LUTs for iron losses, flux linkages and machine torque parametrized by the angular velocity ω_p (or ω_m ; see Figure 4) and the (d, q) -currents within the current circle

$$\left\{ \mathbf{i}_s^{dq} = (i_s^d, i_s^q) \in \mathbb{R}^2 \mid \sqrt{(i_s^d)^2 + (i_s^q)^2} \leq \hat{i}_{s,\text{max}} \right\}; \quad (23)$$

- the admissible ranges of the current and voltage limits, given by

$$\hat{i}_{s,\text{max}} \in [\hat{\underline{i}}_{s,\text{max}}, \hat{\bar{i}}_{s,\text{max}}] \quad \text{and} \quad \hat{u}_{s,\text{max}} \in [\hat{\underline{u}}_{s,\text{max}}, \hat{\bar{u}}_{s,\text{max}}], \quad (24)$$

with lower $(\hat{\underline{i}}_{s,\text{max}}, \hat{\underline{u}}_{s,\text{max}})$ and upper $(\hat{\bar{i}}_{s,\text{max}}, \hat{\bar{u}}_{s,\text{max}})$ bounds on these limits.

Based on this data, either measurements or simulations can be performed for a pre-specified range of different electrical angular velocities (or speeds), i.e.,

$$\omega_p \in [\underline{\omega}_p, \bar{\omega}_p] \quad \text{with} \quad \underline{\omega}_p < \bar{\omega}_p, \quad (25)$$

and a pre-specified range of different reference torques, i.e.,

$$m_{m,\text{ref}} \in [\underline{m}_{m,\text{ref}}, \bar{m}_{m,\text{ref}}] \quad \text{with} \quad \underline{m}_{m,\text{ref}} < \bar{m}_{m,\text{ref}}. \quad (26)$$

The implemented automated software framework for data set creation is illustrated as workflow diagram in Figure 11.

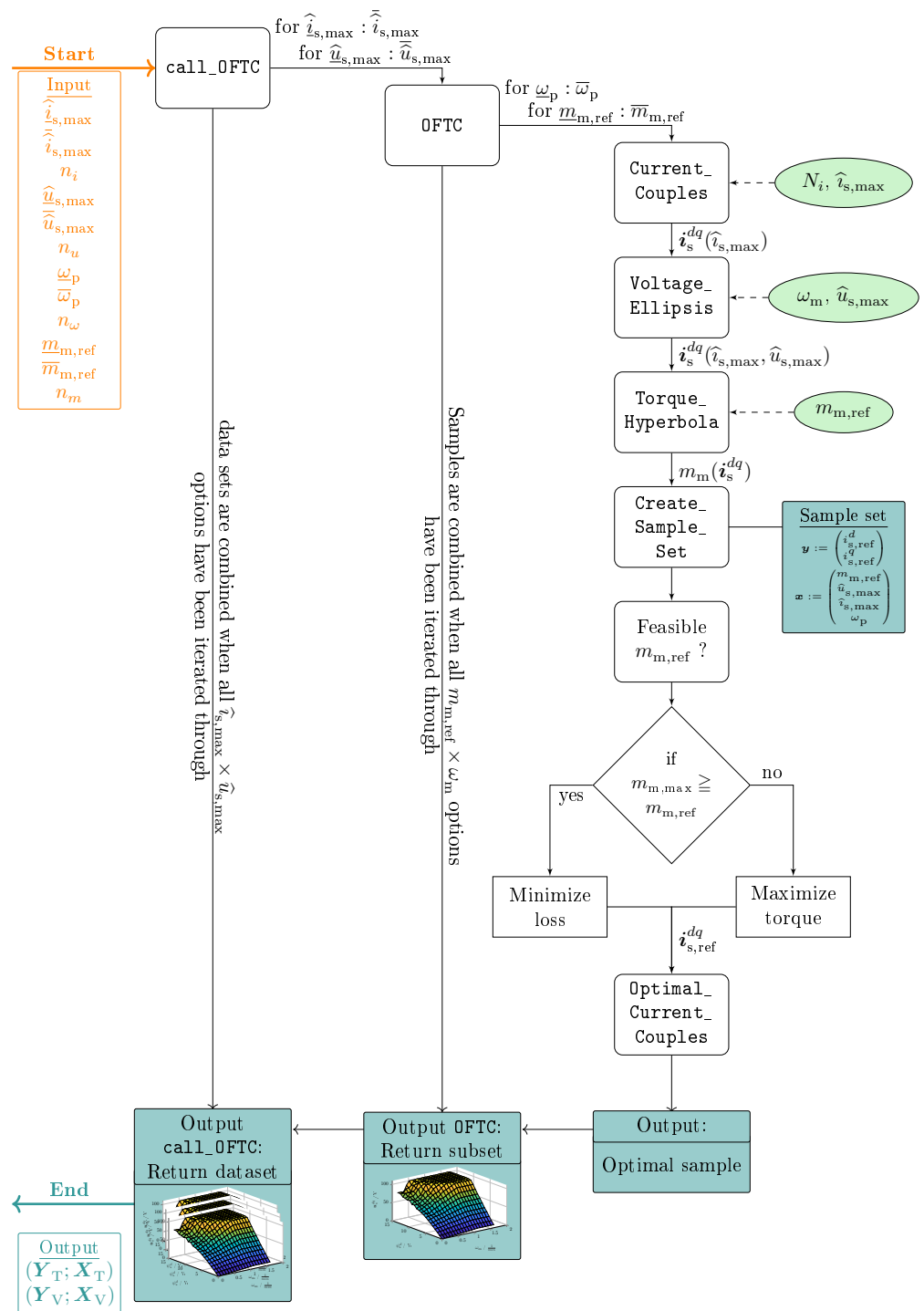


Figure 11. Software framework and workflow for data set creation: Outputs are the Training Set(s) ($Y_T; X_T$) and Validation Set(s) ($Y_V; X_V$).

The user has to provide the upper and lower bounds on current and voltage constraints, angular electrical velocity and reference torque (i.e., $\hat{l}_{s,max}$, $\hat{i}_{s,max}$, $\hat{u}_{s,max}$, $\hat{u}_{s,max}$, ω_p , $\bar{\omega}_p$, $m_{m,ref}$, $\bar{m}_{m,ref}$) as inputs to the software framework. In addition, the user must specify the desired resolution (number of data points) of the intervals (24)–(26) by n_i , n_u , n_ω and n_m , respectively. With this input data, the software framework starts the routine call_OFTC which will iteratively compute the optimal reference currents numerically based on the available measurement and/or FEA data and finally outputs training sets and validation sets for ANN training and validation, respectively.

The iteration over the current and voltage limit intervals (24) starts by calling the subroutine OFTC which then iterates over the speed (25) and torque (26) intervals by subsequently calling the subroutines Current_Couples, Voltage_Ellipsis, Torque_Hyperbola and Create_Sample_Set. If the actual reference torque is feasible (i.e., no current or voltage constraint is violated), the losses are minimized numerically or, if it is not feasible, the feasible torque is maximized by the subroutine Optimal_Current_Couples to finally obtain the optimal reference currents $\mathbf{y} = (i_{s,ref}^d, i_{s,ref}^q)^\top$ as comparative output for later training and validation. For each iteration, the software framework returns and stores a data subset. These individual data subsets are then combined to the overall data set $(Y; X)$ which can be split in training set(s) $(Y_T; X_T)$ and validation set(s) $(Y_V; X_V)$.

Table 3. Rated machine parameters of the considered 4 kW IPMSM.

Description	Symbol & Value	Unit
Rated power (mechanical)	$p_{m,R} = 4$	kW
Rated torque (mechanical)	$m_{m,R} = 6.6$	Nm
Rated speed (mechanical)	$n_{m,R} = 5500$ ($\omega_{m,R} = 575.96$)	rpm ($\frac{\text{rad}}{\text{s}}$)
Rated current (magnitude)	$i_{s,R} = 35$	A
Maximal stator current (magnitude)	$\hat{i}_{s,max} = 35$	A
Maximal stator voltage (magnitude)	$\hat{u}_{s,max} = 110$	V
Stator resistance (at nominal temperature)	$R_s = 0.20$	Ω
Inertia	$\Theta_m = 825 \times 10^{-4}$	kgm ²
Pole pair number	$n_p = 4$	-

In the following, the subroutines introduced above are explained in more detail. At first, Current_Couples returns a prespecified and large number N_i of current couples $i_s^{dq} = (i_s^d, i_s^q)^\top$ within the admissible current circle, satisfying the current constraint (5b), as data basis for the later computation. The current couples are randomly chosen by Monte Carlo sampling [69] over a square and equidistant current grid and then truncated in view of the limitations imposed by the circular current constraint $\hat{i}_{s,max}$. The remaining admissible current couples within the current circle are shown in teal color in Figure 12a.

In the next step, the subroutine Voltage_Ellipsis extracts from the current couples those current couples which satisfy the voltage constraint (5c) and, hence, lie within the voltage ellipse (see teal elliptic area in Figure 12b).

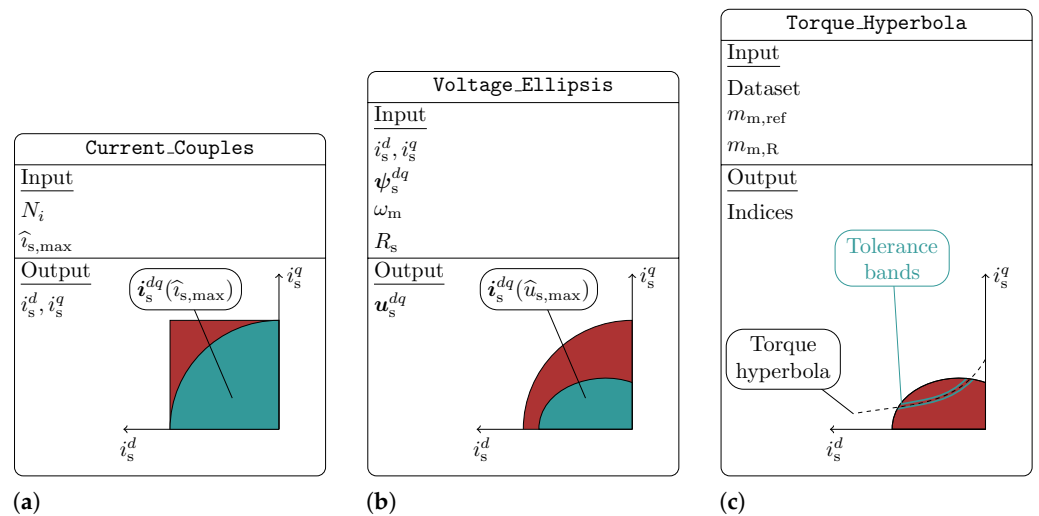


Figure 12. Illustration of subroutines Current_Couples, Voltage_Ellipsis and Torque_Hyperbola. (a) Current constraint. (b) Voltage constraint. (c) Reference torque.

The subroutine Torque_Hyperbola determines those remaining current couples within the voltage ellipse which satisfy the equality constraint for the given reference torque $m_{m,ref}$. Therefore, Torque_Hyperbola returns those current couples on the torque hyperbola which satisfy current and voltage constraint (see teal line in Figure 12c). The thickness of the teal line can be set by a tolerance band around the torque hyperbola depending on the actual reference torque $m_{m,ref}$ and the rated torque $m_{m,R}$. As the current couples lying on this teal line are not necessarily equidistantly located on a square grid, the returned current couples are interpolated. If the reference torque is not feasible as current and/or voltage constraint would be violated, the subroutine returns an empty set of current couples.

The subroutine Create_Sample_Set merges the obtained data in a sample set of the desired form $(y; x)$.

Finally, after a feasibility check for the reference torque is performed (see Figure 11), the subroutine Optimal_Current_Couples selects the optimal reference current couple $i_{s,ref}^{dq} = (i_{s,ref}^d, i_{s,ref}^q)^T$ which either is feasible and produces the desired torque while minimizing copper and iron losses or which maximizes the torque if voltage and/or current constraints are violated. This selection is illustrated in Figure 13.

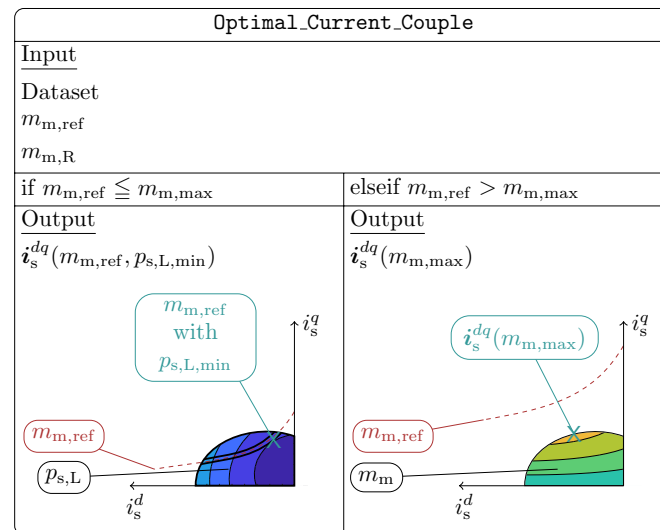


Figure 13. Illustration of subroutines Optimal_Current_Couples with input $m_{m,ref}$ (dashed line) and output $i_{s,ref}^{dq} = i_s^{dq}$: If $m_{m,ref}$ is feasible, the current couple is returned which minimizes losses (see left X); if $m_{m,ref}$ is not feasible the current couple is returned which maximizes the torque (see right X).

In conclusion, based on the inputs to the software framework, it automatically creates the *three* small, medium and large training data sets $(Y_{TS}; X_{TS})$, $(Y_{TM}; X_{TM})$ and $(Y_{TL}; X_{TL})$ and the *two* small and large validation data sets $(Y_{VS}; X_{VS})$ and $(Y_{VL}; X_{VL})$. The key data of these created data sets is collected in Table 4. Amount and size of training and validation sets can be adjusted if required.

Table 4. Inputs for data set creation by the software framework illustrated in Figure 11: The created data set(s) are then split into three training and two validation data sets used for ANN training and validation, respectively.

Inputs for data set creation:												
$\hat{i}_{s,max}$	$\hat{i}_{s,max}$	n_i	$\hat{u}_{s,max}$	$\hat{u}_{s,max}$	n_u	ω_p	$\bar{\omega}_p$	n_ω	$\underline{m}_{m,ref}$	$\bar{m}_{m,ref}$	n_m	N_D
Output: Small training set ($Y_{TS}; X_{TS}$) with 10k samples												
14	42	5	110	185	5	0	1.6	20	0	2	20	10k
Output: Medium training set ($Y_{TM}; X_{TM}$) with 50k samples												
14	42	4	110	185	5	0	1.6	50	0	2	50	10k
Output: Large training set ($Y_{TL}; X_{TL}$) with 200k samples												
14	42	8	110	185	10	0	1.6	50	0	2	50	10k
Output: Small validation set ($Y_{VS}; X_{VS}$) with 10k samples												
14	42	5	110	185	5	0	1.6	20	0	2	20	10k
Output: Large validation set ($Y_{VL}; X_{VL}$) with 200k samples												
14	42	8	110	185	10	0	1.6	50	0	2	50	10k

2.2.3. Data Set Preprocessing for Training (and Validation)

In general, before the training of the ANN, it is beneficial to preprocess the training set(s). This data set preprocessing workflow is illustrated in Figure 14. It consists of data correction, flattening and normalization.

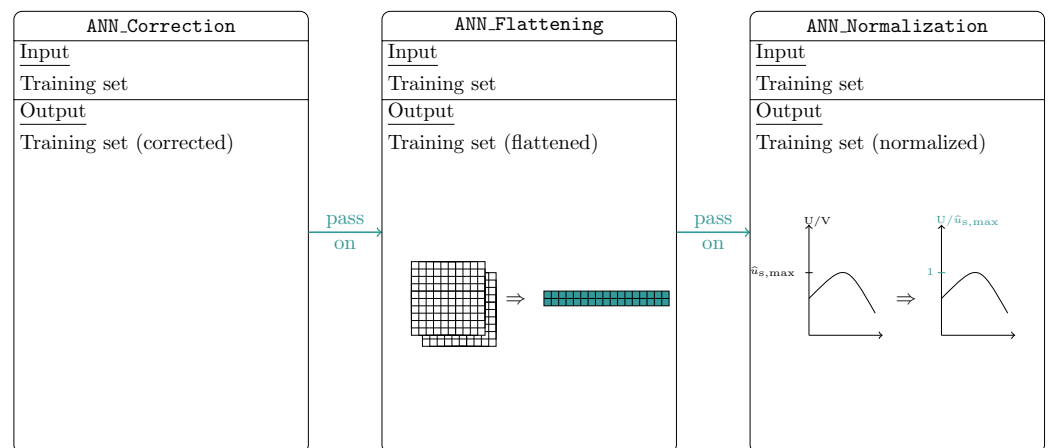


Figure 14. Illustration of the workflow of the data preprocessing.

The subroutine ANN_Correction removes and corrects erroneous data entries. After that the subroutine ANN_Flattening adjusts the data dimensions to the input dimensions of the neural network (recall $x \in \mathbb{R}^4$ in (14) and $\hat{y} \in \mathbb{R}^2$ in (15) for the considered case). This is required as the data set creation provides data with the dimensions

$$(2 + 4) \times n_i \times n_u \times n_\omega \times n_m \tag{27}$$

which must be reduced to

$$(2 + 4) \times \underbrace{(n_i \cdot n_u \cdot n_\omega \cdot n_m)}_{=:N_D} \tag{28}$$

resulting in N_D data sets of the form $(\mathbf{y}[n]; \mathbf{x}[n])$ with $n \in \{1, \dots, N_D\}$ which can be represented as single column vectors $(\mathbf{y}[n]^\top, \mathbf{x}[n]^\top)^\top$ of dimension \mathbb{R}^6 . In the last subroutine ANN_Normalization, the corrected and flattened data vectors $(\mathbf{y}[n]^\top, \mathbf{x}[n]^\top)^\top$ are normalized for all $n \in \{1, \dots, N_D\}$ to accelerate the training time as shown in Table 5 for three different training scenarios. An average decrease in the training time of at least 26% can be achieved, as by reducing the absolute values of the inputs, the weight values will decrease as well resulting in less computational effort.

Table 5. Comparison of training duration and achievable time reduction of the training process by data set normalization for ANN Architecture (A_2) trained with small, medium and large training data set $(Y_{TS}; X_{TS})$, $(Y_{TM}; X_{TM})$ or $(Y_{TL}; X_{TL})$.

Scenario	$(A_2) + (Y_{TS}; X_{TS})$	$(A_2) + (Y_{TM}; X_{TM})$	$(A_2) + (Y_{TL}; X_{TL})$
Training duration <i>without</i> normalization (in s)	136	298	1450
Training duration <i>with</i> normalization (in s)	81	288	937
Relative time reduction (in %)	−40.4%	−3.4%	−35.4%
Average time reduction (in %)		−26.4%	

The normalization is realized by dividing all input and output data entries by their respective absolute maximal value in order to return a normalized data set with values within the intervals $[-1, 1]$. This division also motivates for the initial data correction step to assure that the interval $[-1, 1]$ is fully exploited.

2.2.4. Network Training

During the training, the Levenberg-Marquardt algorithm minimizes the squared Euclidean distance (20) by adapting the ANN weights according to (22). Such an ANN training can easily be implemented in MATLAB R2019b using the feedforwardnet-function of MATLAB’s Deep Learning Toolbox. The training should happen in several epochs (during which the whole training data set $(Y_T; X_T)$ is used once), to improve the generalization capability of the ANN. Before the training, multiple training terminators must be specified to provide stopping criteria for the training process. The most important termination parameters are collected in Table 6.

In Figure 15, the training results are shown for ANN Architectures (A_1) and (A_2) and three different training sets $(Y_{TS}; X_{TS})$ [small set], $(Y_{TM}; X_{TM})$ [medium set] and $(Y_{TL}; X_{TL})$ [large set]. It can be seen that, independently of the training data sets in use, Architecture (A_2) with two hidden layers, reaches smaller error norms $\|e\| = \sqrt{d(e)}$ after a smaller number of epochs. Moreover, note that only for Architecture (A_1) with the large training set $(Y_{TL}; X_{TL})$ all 400 epochs are required for training. All other trainings terminate earlier due to the fact that no further training improvements are achieved.

2.2.5. Avoidance of Overfitting

Overfitting must be avoided to circumvent deteriorated performance of the trained ANN outside (and within) the used training data set. To do so, two main aspects must be considered:

- proper training (and validation) sets must be chosen in order to cover all relevant regions (operating points) for ANN-based OFTC; in particular, a proper size of the training set (number of operating points) must be found. Therefore, three different sizes (small, medium and large) of training sets for training and two validation sets for validation were used and their respective results were compared. The sets have been generated by Monte Carlo simulation while an optimal coverage of the desired operating points was assured. Those sets leading to the best approximation performance of the ANN were finally used; and

- proper termination criteria for the training process must be formulated (see Table 6 and, for more details, see Chapter 11.5.2 in [66]), in particular, during training a quick parallel validation is performed and if here the validation approximation error increases for a certain number of epochs in a row (see solid green lines in Figure 15), the training is stopped and the trained weights *before* the increase of the approximation error are used as final weights for the trained ANN.

Table 6. Training termination parameters to end training after requirements have been met.

Termination Parameter	Value	Description
Maximum number of epochs	400	Training is terminated if the specified number of epochs is reached (to avoid overfitting and to guarantee termination).
Maximum validation failure runs	10	Training stops if performance (accuracy) worsens for a certain number of epochs in a row (to avoid overfitting).
Maximum training time	∞	Training is terminated when the time limit is exceeded (as no computational limit was imposed/required during training and validation).

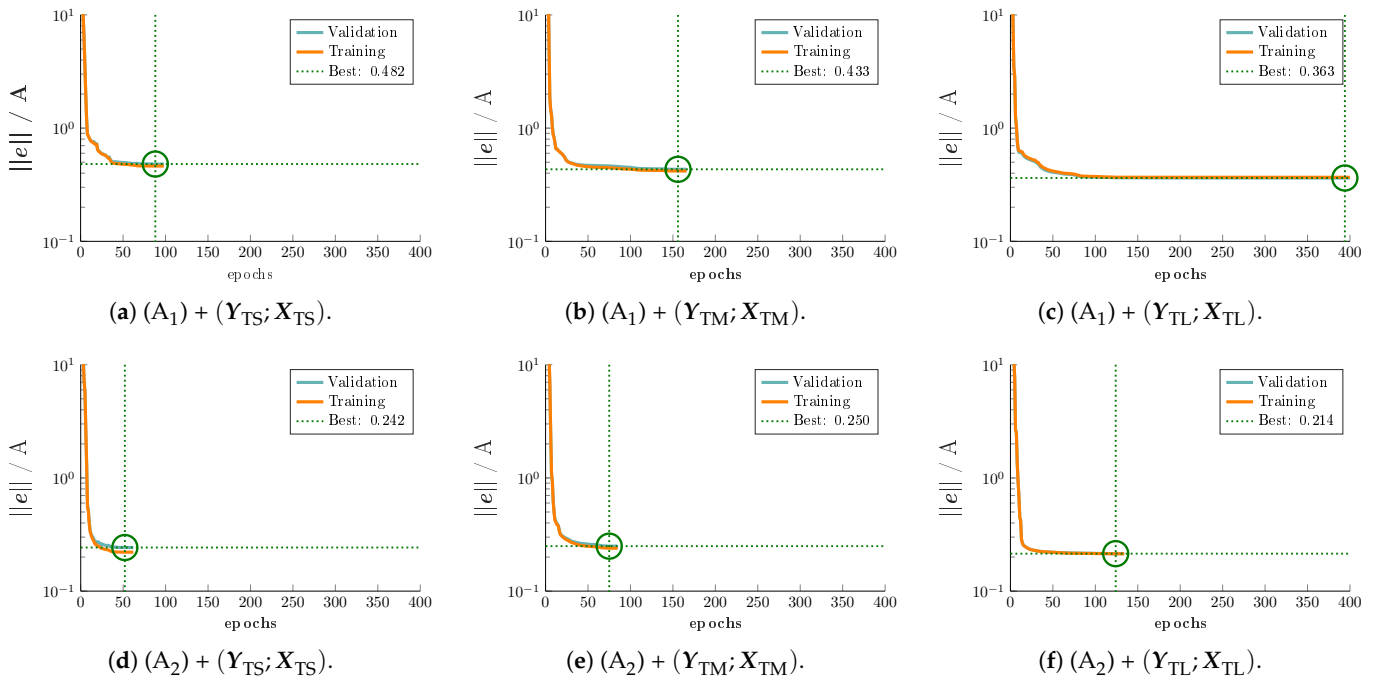


Figure 15. Evolution of approximation error norm (learning error) over epochs during the training of the ANN Architectures (A_1) & (A_2) with the small, medium and large training data sets ($Y_{TS}; X_{TS}$) ($Y_{TM}; X_{TM}$) and ($Y_{TL}; X_{TL}$), respectively.

2.3. Artificial Neural Network Validation

After the training of both ANN Architectures (A_1) and (A_2) with the created small, medium and large training sets ($Y_{TS}; X_{TS}$), ($Y_{TM}; X_{TM}$) and ($Y_{TL}; X_{TL}$), both are validated with the help of the created small and large validation sets ($Y_{VS}; X_{VS}$) and ($Y_{VL}; X_{VL}$).

2.3.1. Validation Procedure

The validation procedure is similar to the training procedure as illustrated in Figure 10. However, instead of using as input the training sets, the validation sets are fed to the preprocessing block while weight adaption is turned off. Finally, the trained ANNs will output the optimal reference currents according to the validation input data.

The validation scenarios as well as the closed-loop simulations are performed on a conventional personal computer (see specification in Table 7).

Table 7. Computer specification of the utilized MacBook Pro (16-inch, 2019).

Parameter	Value
Operating System	macOS Monterey Version 12.0.1
Processor	2.6 Hz 6-Core Intel Core i7
Memory	16 2 667 MHz DDR4
Graphics	AMD Radeon Pro 5300M 4

Two validation scenarios are considered. The first scenario aims at evaluating the execution time of the ANN to compute the optimal reference currents $(i_{s,\text{ref,ANN}}^d, i_{s,\text{ref,ANN}}^q)^\top$ based on the provided input validation data $x_{\text{VS}}[n]$ and $x_{\text{VL}}[n]$ (with $n \in \{1, \dots, N_D\}$) of the validation sets $(Y_{\text{VS}}; X_{\text{VS}})$ and $(Y_{\text{VL}}; X_{\text{VL}})$, respectively. The second scenario aims at validating the approximation accuracy of the trained ANNs by computing and evaluating the approximation error

$$e[n] = \mathbf{y}[n] - \hat{\mathbf{y}}[n] = \begin{pmatrix} i_{s,\text{ref}}^d[n] \\ i_{s,\text{ref}}^q[n] \end{pmatrix} - \begin{pmatrix} i_{s,\text{ref,ANN}}^d[n] \\ i_{s,\text{ref,ANN}}^q[n] \end{pmatrix}$$

between numerically obtained optimal reference currents and those computed by the ANN. The results of both validation scenarios are discussed in the following two subsections.

2.3.2. Validation Scenario I: Execution Time

The execution times of one execution call (i.e., computing an individual sample $(y_V[n]; x_V[n])$) of the ANN Architectures (A_1) and (A_2) trained each with the training sets $(Y_{\text{TS}}; X_{\text{TS}})$, $(Y_{\text{TM}}; X_{\text{TM}})$ and $(Y_{\text{TL}}; X_{\text{TL}})$ are validated using the small validation set $(Y_{\text{VS}}; X_{\text{VS}})$. The results are shown in Figure 16 over the 10,000 samples of the small validation set. All six validation test results show that almost the same average execution time of $3.45 \mu\text{s}$ per ANN execution are obtained. The size of the training set (small, medium or large) does not significantly have impact on the execution time. Only the size of the ANN architecture affects the execution time. The execution time of Architecture (A_2) increases by 16–35% to $3.85 \mu\text{s}$ on average compared to that of Architecture (A_1) .

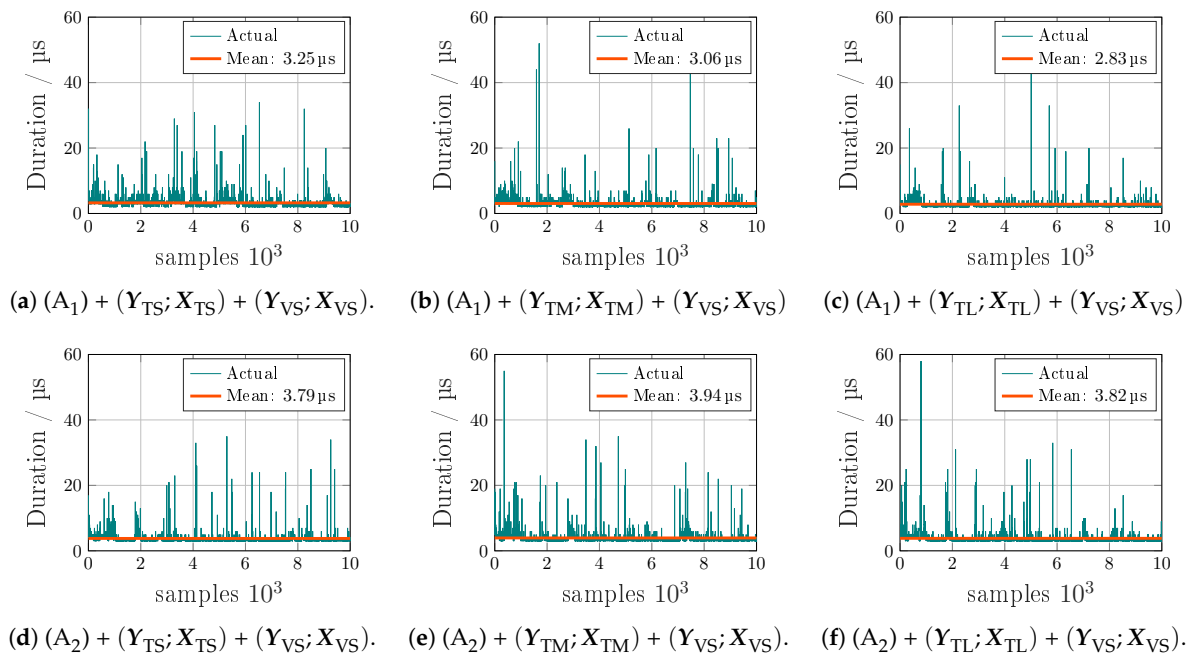


Figure 16. Illustration of execution times over 10,000 samples during the validation of the ANN Architectures (A_1) [top row] & (A_2) [bottom row] with the small validation data set $(Y_{\text{VL}}; X_{\text{VL}})$ (trained each with the small, medium and large $(Y_{\text{TS}}; X_{\text{TS}})$, $(Y_{\text{TM}}; X_{\text{TM}})$ and $(Y_{\text{TL}}; X_{\text{TL}})$).

2.3.3. Validation Scenario II: Approximation Accuracy

In contrast to the execution time, both, the size of the training sets as well as the size of the ANN architecture, have a strong impact on the achievable approximation accuracy (i.e., the norm of the approximation error). The results of the second validation scenario are shown in Figure 17. For each architecture, the individual approximation errors

$$\varepsilon[n] = i_{s,\text{ref}}^d[n] - i_{s,\text{ref,ANN}}^d[n] \quad \text{and} \quad \varepsilon[n] = i_{s,\text{ref}}^q[n] - i_{s,\text{ref,ANN}}^q[n],$$

of d - and q -reference currents are computed for all samples $n \in \{1, \dots, N_D\}$ of the considered validation set and, then, associated to and collected in error bars with a width of 0.1 A. Finally, mean error $\bar{\varepsilon}$ as in (18), standard error deviation σ_ε as in (19) and error distribution probability

$$P(\varepsilon) = \frac{n_\varepsilon}{N_D}$$

of each error bar are computed, where $n_\varepsilon < N_D$ represents the number of samples bundled in one error bar.

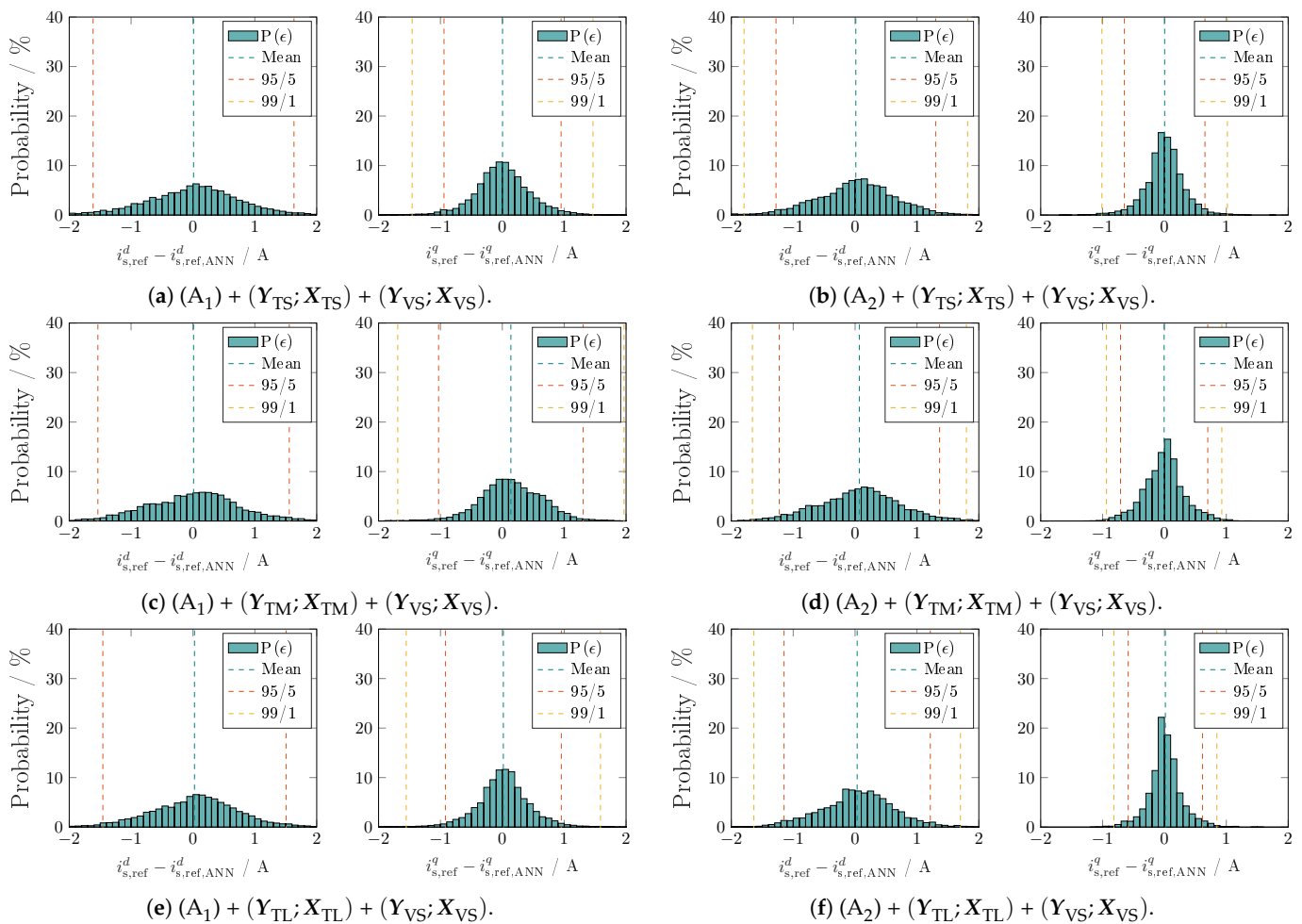


Figure 17. Illustration of individual error distributions $\varepsilon = i_{s,\text{ref}}^d - i_{s,\text{ref,ANN}}^d$ [left subplots] and $\varepsilon = i_{s,\text{ref}}^q - i_{s,\text{ref,ANN}}^q$ [right subplots] obtained during the validation of the ANN Architectures (A₁) [left column, i.e., (a,c,e)] & (A₂) [right column, i.e., (b,d,f)] with the small, medium and large validation data sets (Y_{VS}; X_{VS}) (Y_{VM}; X_{VM}) and (Y_{VL}; X_{VL}), respectively.

Figure 17 shows six subplots [see Figure 17a–f] with two error distributions for d - and q -current approximation error each. The left column shows the results for ANN Architecture (A₁) [see Figure 17a,c,e]; whereas the right column contains the results for ANN Architecture (A₂) [see Figure 17b,d,f]. The vertical red and yellow lines in each

subplot indicate the error regions which cover 95% (i.e., $\approx \bar{e} \pm 2\sigma_e$ similar to Region II above) and 99% (i.e., $\approx \bar{e} \pm 3\sigma_e$ similar to Region III above) of all error bars, i.e., 95% and 99% of all errors lie between the red and yellow lines, respectively. It becomes clear that the larger Architecture (A_2) has a better approximation accuracy than Architecture (A_1). Moreover, it becomes even better for larger training sets (i.e., red and yellow line move closer to zero). Note that no approximation errors larger than ± 2 A occur. For example, in the right subplot of Figure 17f, more than 90% of the q -current error bars yield approximation errors smaller than $\frac{0.35A}{35A} = 1\%$, which is conform to the specified approximation accuracy. For the d -current error bars [see left subplot of Figure 17f] about 65% satisfy the specified approximation accuracy, which is still acceptable. This significant difference is due to the used distribution of the current couples shown in Figure 18, where clearly more q -currents were in the exemplary training set due to the voltage and current constraints. Nevertheless, as the following closed-loop implementation results will show, the approximation accuracy of the ANN Architecture (A_2) trained with the large training set ($Y_{TL}; X_{TL}$) definitely achieves a very acceptable overall performance for ANN-based OFTC.

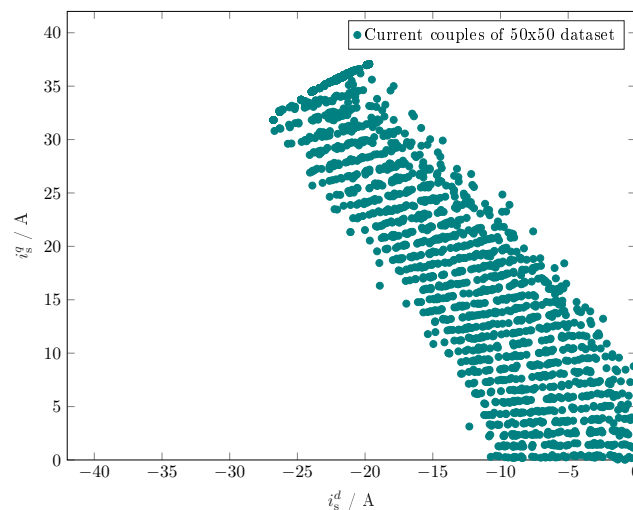


Figure 18. Distribution of the current couples of an exemplary dataset with constant current and voltage constraints $\hat{i}_{s,\max}$ and $\hat{u}_{s,\max}$ but variable ω_m and $m_{m,\text{ref}}$.

3. Closed-Loop Implementation and Comparison

Finally, five OFTC approaches are implemented in closed-loop operation of the electrical drive system and compared concerning approximation accuracy and execution time:

- OFTC_{NLP}: OFTC with ORCC using the MATLAB-function `fmincon` (solving the nonlinear problem (NLP) directly);
- OFTC_{LUT}: OFTC with ORCC using pre-generated LUTs;
- OFTC_{NUM}: OFTC with numerical ORCC;
- OFTC_{ANA}: OFTC with analytical ORCC; and
- OFTC_{ANN}: OFTC with ORCC utilizing the proposed ANN.

The trained and validated ANN with two hidden layers and 20 neurons per layer (as described in Section 2) is implemented to achieve ANN-based OFTC (OFTC_{ANN}) as illustrated in Figure 6. OFTC_{NLP} is considered as the benchmark for all other approaches concerning the accuracy of the obtained reference currents as the nonlinear optimization problem with all constraints is solved directly with the powerful `fmincon` MATLAB-function. However, concerning the execution time, OFTC_{NLP} is by far the slowest with $111\,500\ \mu\text{s} = 111.5\ \text{ms} = 0.112\ \text{s}$ and, hence, it is clearly not applicable in real-time. For OFTC_{NUM} and OFTC_{ANA}, the respective ORCC and decision tree are implemented as discussed in e.g., [4,25] with numerical and analytical solution of the quadric intersection

problem (i.e., finding roots of fourth-order polynomials). Both, OFTC_{NUM} and OFTC_{ANA}, require an iterative computation due to the necessary online linearization of the nonlinear optimization problem. OFTC_{LUT} is based on pre-generated LUTs from which the optimal reference currents are extracted online with inverse interpolation (as mostly used in industry). More details about implementation, results and comparison are provided in the following subsections.

3.1. Description of Closed-Loop Implementation

The presented machine model of Section 1.3 is implemented in MATLAB & Simulink R2019b. The simulated machine is a real, nonlinear 4 kW IPMSM with rated parameters as introduced in Table 3 and nonlinearities like flux linkages and iron losses as shown in Figure 4. An inverter with pulse-width modulation (PWM) and a switching frequency of 18 kHz applies the stator voltages to the machine. The idealized inverter model is taken from [70], Chapter 14 and neglects switching and conduction losses but allows for switching and varying dc-link voltages (not considered here). For all five OFTC approaches, a field-oriented control (FOC) system as shown in Figure 6 is used.

For the ANN-based OFTC approach, the utilized ANN Architecture (A_2) was trained with the large Training Set ($Y_{TL}; X_{TL}$), validated by the large Validation Set ($Y_{VL}; X_{VL}$) and implemented in Simulink using the `feedforwardnet`-function which allows to specify the to be used activation functions, number of layers and training terminators (recall Tables 2 and 6). Architecture (A_2) was chosen as it is more accurate than Architecture (A_1) [recall Figure 17] and its execution time is not significantly longer [recall Figure 16].

All five OFTC approaches (i.e., OFTC_{NLP}, OFTC_{LUT}, OFTC_{NUM}, OFTC_{ANA} and OFTC_{ANN}) are implemented in Simulink R2019b in combination with FOC, modulator and voltage source inverter and nonlinear machine model as a closed-loop electrical drive system. An identical simulation scenario is run for all five OFTC approaches to allow for a fair and direct comparison of their individual performances (accuracies) and execution times.

The chosen simulation scenario represents a typical start-up operation of an electrical drive system: The IPMSM is accelerated from stand-still to 150 % of its rated speed $n_{m,R}$ while a constant load torque of about 64 % of the rated torque $m_{m,R}$ is applied to illustrate the optimal operation management over a wide range of different operating conditions and for several optimal operation strategies (such as MTPL, MC and FW). For all operation strategies, optimal and feasible reference currents are computed and tracked to stay within the imposed current and voltage constraints.

3.2. Implementation Results and Discussion

In Figures 19–21, the implementation results are shown. To ease readability of the presented results, not all OFTC approaches are depicted. Figure 19 presents the time series plots of OFTC_{ANA} (as best conventional OFTC approach) and OFTC_{ANN} (as newly proposed OFTC approach) and Figure 21 shows the comparison of these two time series plots against those of OFTC_{NLP} (as benchmark concerning accuracy); whereas in Figure 20, the current loci of OFTC_{ANA} and OFTC_{ANN} are depicted. The results of all five approaches are actually very similar (as will be discussed later; see also Table 8) and, hence, no more insight would be obtained by showing all implementation results.

The time series of actual [—] and reference or maximum values [---] of stator currents i_s^q and i_s^d , current magnitude $\|i_s^{dq}\|$, voltage magnitude $\|u_s^{dq}\|$, machine speed n_m (in 1/min = rpm) and torque m_m are shown in Figure 19, where the left column [see Figure 19a] presents the simulation results for OFTC with analytical ORCC (OFTC_{ANA}) and the right column [see Figure 19b] for ANN-based OFTC (OFTC_{ANN}).

The background colors in Figure 19a,b indicate the active optimal operation strategy such as MTPL [], MC_{ext} [], MC [] and FW [] (for details see [4,5,25]). Despite the fact, that the ANN-based OFTC does not distinguish between different operation strategies, the background colors from Figure 19a are used in Figure 19b as well to allow for an easier comparison.

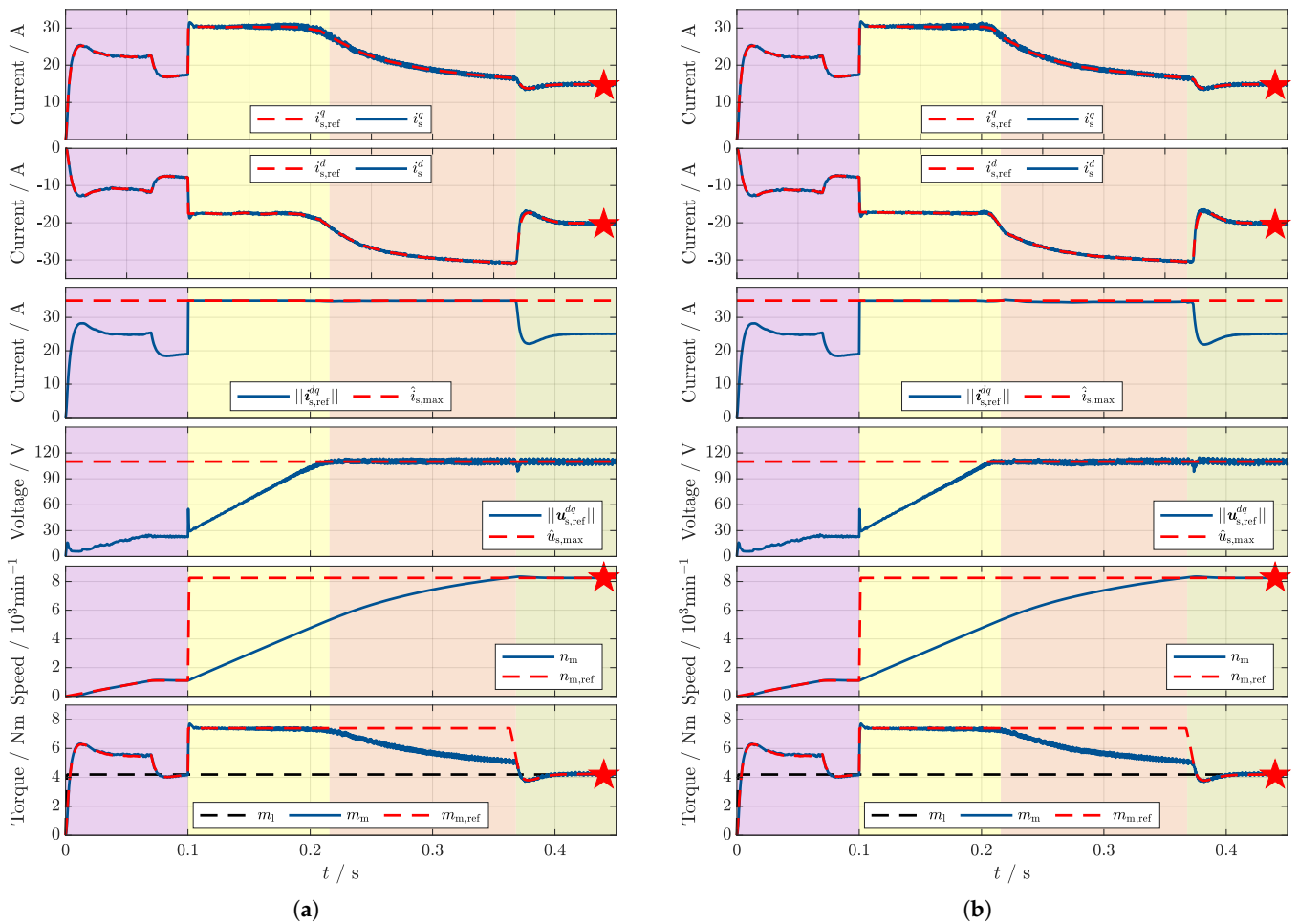
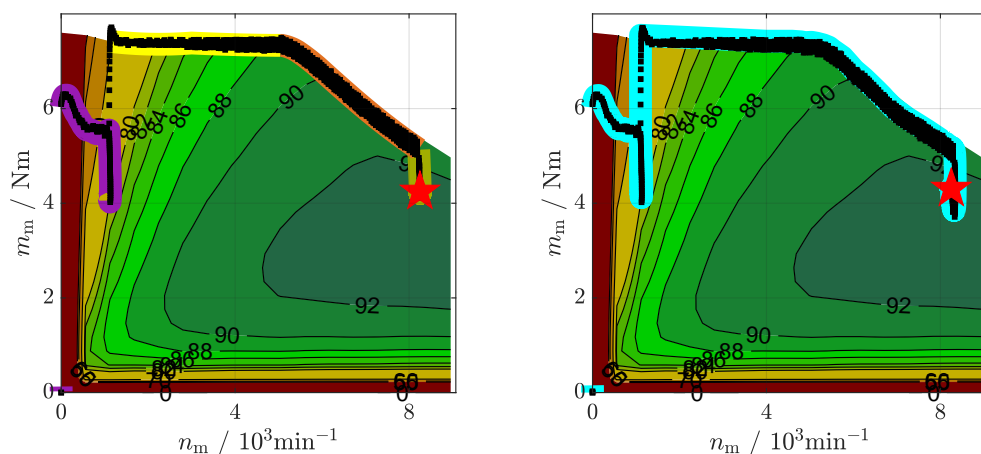


Figure 19. Implementation results: (a) OFTC with analytical ORCC (OFTC_{ANA}) with smooth transition between all operation strategies (i.e., MTPL [], MC_{ext} [], MC [] and FW []) and (b) ANN-based OFTC (OFTC_{ANN}).

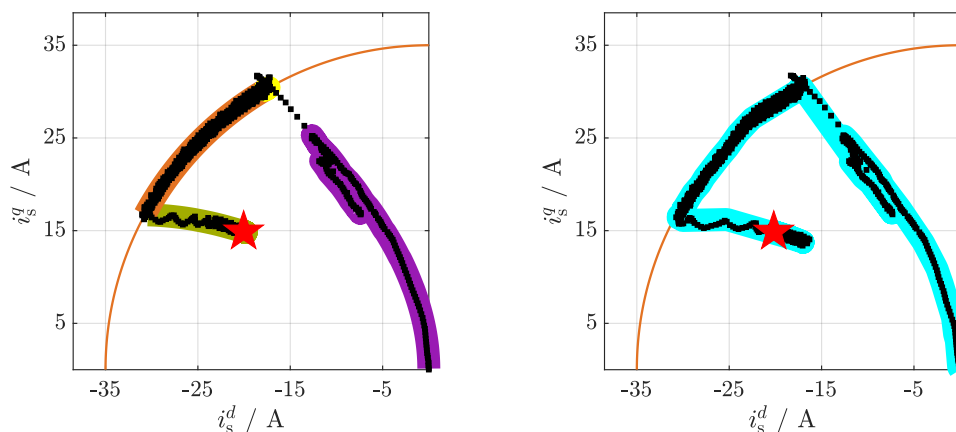
Besides the time series plots, the speed-torque maps with efficiency contour lines [top row] and the current evolution in the current locus (plane) [bottom row] are depicted in Figure 20 for the same closed-loop implementation scenario: Again, the left column shows the results for OFTC with analytical ORCC [i.e., OFTC_{ANA}, see Figure 20a,c], whereas the right column contains the results for ANN-based OFTC [i.e., OFTC_{ANN}, see Figure 20b,d]. The actual speed-torque pairs (n_m, m_m) or actual current pairs (i_s^d, i_s^q) are marked by [■]. The final (optimal) operating point is marked by [★].

Initially ($t \leq 0.1$ s), the machine is slowly ramped up to 20% of $n_{m,R}$ [cf. Figure 19a,c]. Neither current nor voltage constraints are violated. Therefore, the OFTC goal is to maximize efficiency by minimizing copper and iron losses during MTPL operation. The requested reference torques are feasible and nicely tracked.

At time $t = 0.1$ s, the reference speed jumps to 150% of $n_{m,R}$ and subsequently, the reference torque reaches its maximum at $m_{m,max}$. Consequently, the feasible reference currents are limited to the current circle and the optimal operation strategy MC_{ext} [] is active until $t = 0.22$ s.



(a) OFTC with analytical ORCC: Speed-torque map. (b) ANN-based OFTC: Speed-torque map.



(c) OFTC with analytical ORCC: Current locus. (d) ANN-based OFTC: Current locus.

Figure 20. Implementation results: Illustration of optimal operation management considering copper and iron losses minimization and voltage and current constraints—Left column: OFTC with analytical ORCC (OFTC_{ANA}) with (a) speed-torque map and (c) current locus; Right column: ANN-based OFTC (OFTC_{ANN}) with (b) speed-torque map and (d) current locus.

Due to the increasing speed, the voltage constraint is reached at $t = 0.22$ s and, therefore, the operation strategy MC [] becomes active. The valid current couples move on the current circle to more negative reference currents [i.e., $i_{s,ref}^d < 0$, see also current loci in Figure 20b,d, respectively]. Consequently, the produced machine torque is smaller than the reference torque, since it is not feasible anymore due to the active current and voltage constraints.

The reference speed is reached at $t = 0.368$ s for OFTC with ORCC and at $t = 0.372$ s for ANN-based OFTC, respectively. As no accelerating torque is required, the reference torque drops to the load torque. The current constraint does not limit operation anymore, but the voltage constraint is still active as the IPMSM operates at high speeds. Hence, the optimal operation strategy now is FW []. After some transients due to the current control system, the final operating point [★] is reached at $t = 0.44$ s and the machine speed remains constant until the simulation ends at $t = 0.45$ s. The efficiency for both strategies is at its maximum for all operating conditions [cf. Figure 20a,b for OFTC with analytical ORCC and for ANN-based OFTC, respectively].

Finally, in Figure 21, the simulation results of OFTC with analytical ORCC [—], ANN-based OFTC [- - -] are compared with the optimal reference currents $i_{s,ref}^d$ and $i_{s,ref}^q$ obtained by directly solving the Nonlinear Optimization Problem (NLP) problem (5) with the MATLAB function fmincon [- - -]. The NLP values are considered the benchmark values for the other two OFTC approaches. The plotted signals in Figure 21 are the actual reference

currents, utilized reference current magnitude $\|i_{s,\text{ref}}^{dq}\|$, utilized voltage magnitude $\|u_s^{dq}\|$, actual machine speed n_m (in 1/min = rpm) and actual machine torque m_m . The results show that, in particular, all reference current pairs of OFTC with ORCC and ANN-based OFTC are very similar and close to those obtained by the OFTC solved directly by `fmincon` (NLP). Only very small deviations can be observed: From $t = 0.23$ s to $t = 0.37$ s, the current magnitude of ANN-based OFTC is slightly below the current limit. Therefore, the torque potential is not completely utilized to its full extent. That is why the target speed is reached about $\Delta t = 0.004$ s later by the ANN-based OFTC than by the other OFTC approaches.

Table 8 summarizes the results of the comparison by computing the Integral Absolute Errors (IAE)

$$\text{IAE} := \int_{t_{\text{start}}}^{t_{\text{end}}} |e(t)| dt$$

of reference current differences, reference torque differences (during MTPL, MC_{ext} and FW operation), voltage utilization difference (during MC & FW operation), current utilization difference (during MC_{ext} & MC operation) and mechanical speed difference. Moreover, the mean execution times of the OFTC with analytical ORCC and ANN-based OFTC are listed. The mean execution time \bar{t}_{exec} is averaged over 10 simulation runs which results in $10 \cdot 0.45 \text{ s} \times 18 \text{ kHz} = 81,000$ OFTC calls for each of the five OFTC approaches.

Table 8. Comparison of the time series of the implementation results of OFTC_{LUT} (LUT-based OFTC), OFTC_{NUM} (OFTC with numerical ORCC), OFTC_{ANA} (OFTC with analytical ORCC) and proposed OFTC_{ANN} (ANN-based OFTC) against OFTC_{NLP} (OFTC with `fmincon`-based ORCC) evaluating the Integral Absolute Error (IAE) performance measure and the execution time.

IAE of	Time Interval	$\text{IAE}_X := \int e dt$	$X = \text{OFTC}_{\text{LUT}}$	$X = \text{OFTC}_{\text{NUM}}$	$X = \text{OFTC}_{\text{ANA}}$	$X = \text{OFTC}_{\text{ANN}}$
d -current	whole	$\int i_{s,\text{ref},\text{OFTC}_{\text{NLP}}}^d - i_{s,\text{ref},X}^d dt$	0.176 As	0.186 As	0.384 As	0.227 As
q -current	whole	$\int i_{s,\text{ref},\text{OFTC}_{\text{NLP}}}^q - i_{s,\text{ref},X}^q dt$	0.195 As	0.207 As	0.256 As	0.213 As
torque	MTPL, MC_{ext} , FW	$\int m_{\text{ref},X} - m_m dt$	0.018 Nms	0.017 Nms	0.026 Nms	0.020 Nms
voltage limit	MC, FW	$\int \hat{u}_{s,\text{max}} - \ u_{s,\text{ref},X}^{dq}\ dt$	0.445 Vs	0.446 Vs	0.454 Vs	0.459 Vs
current limit	MC, MC_{ext}	$\int \hat{i}_{s,\text{max}} - \ i_{s,\text{ref},X}^{dq}\ dt$	0.010 As	0.000 As	0.000 As	0.059 As
speed	whole	$\int n_{m,\text{ref}} - n_{m,X} dt$	$752.4 \frac{\text{s}}{\text{min}}$	$750.9 \frac{\text{s}}{\text{min}}$	$752.6 \frac{\text{s}}{\text{min}}$	$757.8 \frac{\text{s}}{\text{min}}$
mean execution time	one OFTC call	$\bar{t}_{\text{exec},X}$	2734.782 μs	448.745 μs	439.671 μs	5.855 μs

The first observation is that all differences between the IEA values of all five OFTC approaches with ORCC and ANN-based OFTC are negligibly small. Only the IAE value of the current utilization of the ANN-based OFTC is slightly larger than the IAE of the worst of all other OFTC algorithms (here: OFTC_{LUT}). As already mentioned that is because the utilized current magnitude is slightly below the current limit during the time interval [0.23 s, 0.37 s].

Most important, the ANN-based approach requires only 5.855 μs for one OFTC execution, i.e., almost less than a hundredth of the execution time of the implemented OFTC with analytical or numerical ORCC and nearly less than a five hundredth of the execution time of the LUT-based approach. This is a rather impressive result as the execution time without OFTC approach takes about 2.52 μs . In this case, “without OFTC” means that the q -reference current is computed by $i_{s,\text{ref}}^q = \frac{3\kappa^2}{2\psi_{\text{pm}} n_p} m_{m,\text{ref}}$ whereas the d -reference current is simply set to zero, i.e., $i_{s,\text{ref}}^d = 0$.

In conclusion, the execution time of ANN-based OFTC is significantly shorter than those of the state-of-the-art OFTC approaches and, hence, it seems definitely more promising and suitable for real-time implementation.

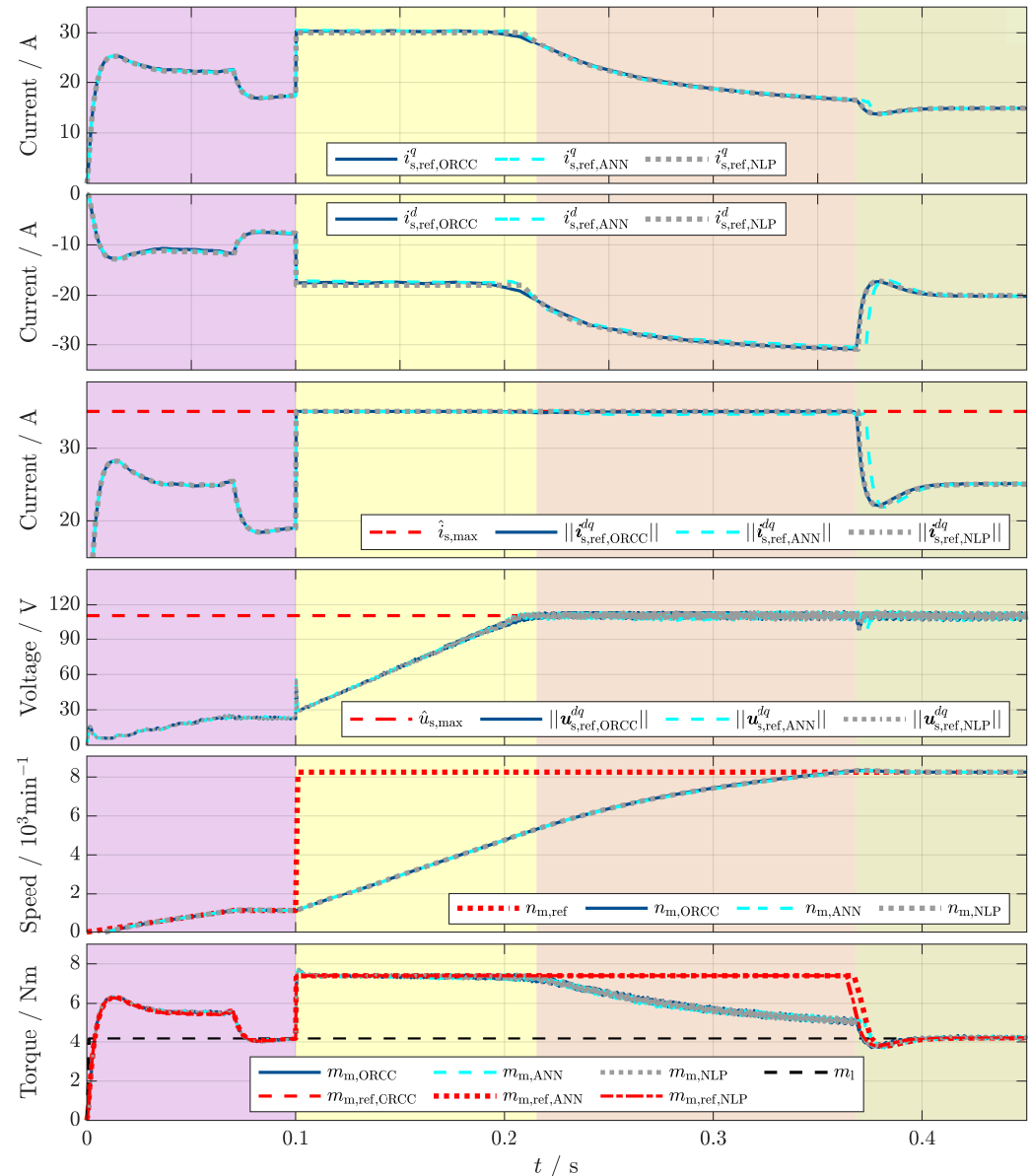


Figure 21. Implementation results: Direct comparison between OFTC with analytical ORCC=ORCC_{ana} [—], ANN-based OFTC [-.-.] and OFTC solved directly by fmincon (NLP) [.....].

4. Summary and Outlook

A novel artificial neural network (ANN) based optimal feedforward torque control (OFTC) strategy has been proposed where the ANN replaces the computationally expensive numerical, analytical or LUT-based optimal reference current computation (ORCC) to minimize copper and iron losses in an electrical drive system with highly nonlinear interior permanent magnet synchronous machine (IPMSM). ANN design, training and validation have been explained in detail. Special focus has been put on the data set(s) creation to obtain training and validation sets for the implemented ANNs.

Finally, the ANN-based OFTC strategy has been implemented in Matlab/Simulink in combination with field-oriented control of a real, nonlinear IPMSM electrical drive system and it has been compared to state-of-the-art OFTC approaches with e.g., analytical,

numerical and LUT-based ORCC. The validation tests have shown that the ANN is capable of (a) approximating the nonlinear OFTC behavior accurately with approximation errors of less than 1% and (b) computing the optimal reference currents within less than 5.9 μs compared to at least 440 μs of the fastest state-of-the-art OFTC approach. Drive performance and achieved efficiency for all OFTC strategies are almost identical, which proves not only that the novel ANN-based OFTC is *capable* of controlling an electrical drive system *requiring less computational time* but also doing so *with very high accuracy*.

Future research will aim at (i) optimizing ANN design (architecture) for typically used microcontrollers in electrical drive systems (e.g., exploiting DSP resources and/or FPGA parallelization capabilities as much as possible), (ii) selecting other reasonable and suitable input parameters (features) for ANN training to achieve even higher approximation accuracies, (iii) considering additional operating conditions (such as angular position and stator and/or rotor temperature) during ANN training, validation and closed-loop implementation to improve OFTC performance and efficiency at their maxima, (iv) implementing recursive or structured neural networks to be able to possibly even cover dynamical effects (such as e.g., current and/or dc-link dynamics) or optimal pulse patterns (OPP) and (v) implementing optimized ANN-based OFTC on different real-time platforms to validate its performance and real-time capability via measurements in the laboratory.

Author Contributions: Conceptualization, C.M.H. and N.M.; methodology, M.A.B. and N.M.; software, M.A.B.; validation, M.A.B. and N.M.; formal analysis, M.A.B., N.M. and C.M.H.; investigation, M.A.B. and N.M.; resources, M.A.B., N.M. and C.M.H.; data curation, M.A.B. and N.M.; writing—original draft preparation, M.A.B., N.M. and C.M.H.; writing—review and editing, C.M.H.; visualization, M.A.B., N.M. and C.M.H.; supervision, N.M. and C.M.H.; project administration, C.M.H.; funding acquisition, C.M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADALINE	Adaptive Linear Neuron
ANN	Artificial Neural Network
back-EMF	back-Electromotive Force
BEV	Battery Electric Vehicle
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DSP	Digital Signal Processor
FE	Finite Element
FEA	Finite Element Analysis
FNN	Feedforward Neural Network
FLOPS	Floating Point Operations
FOC	Field-Oriented Control
FPGA	Field-Programmable Gate Array
FW	Field Weakening
HEV	Hybrid Electric Vehicle
IPMSM	Interior Permanent Magnet Synchronous Machine
IAE	Integral Absolute Error
LMA	Loss Minimization Algorithm
LMC	Loss Minimization Control
LSTM	Long Short-Term Memory
LUT	Look-Up Table
MC	Maximum Current
MCSA	Motor Current Signature Analysis
MLP	Multilayer Perceptron

MDPI	Multidisciplinary Digital Publishing Institute
ME	Mean Error
MEPA	Maximum Efficiency per Ampere
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Controller
MSE	Mean Squared Error
MTPA	Maximum Torque per Ampere
MTPC	Maximum Torque per Current
MTPF	Maximum Torque per Flux
MTPL	Maximum Torque per Losses
MTPV	Maximum Torque per Voltage
NAN	Not A Number
NLP	Nonlinear Optimization Problem
OFTC	Optimal Feedforward Torque Control
OOSS	Optimal Operation Strategy Selection
OPP	Optimal Pulse Pattern
ORCC	Optimal Reference Current Computation
PHEV	Hybrid Electric Vehicles
PMSM	Permanent Magnet Synchronous Machine
PNN	Probabilistic Neural Network
PTC	Predictive Torque Controller
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RNG	Random Number Generator
RNN	Recurrent Neural Network
SED	Standard Error Deviation
SM	Synchronous Machine
VMC	Voltage Matching Circuit

References

- De Almeida, A.; Ferreira, F.; Fong, J. Standards for Efficiency of Electric Motors. *IEEE Ind. Appl. Mag.* **2011**, *17*, 12–19. [[CrossRef](#)]
- Ritchie, H.; Roser, M. Energy Mix. 2021. Available online: <https://ourworldindata.org/energy-s^bmix> (accessed on 28 June 2021).
- Eroglu, I.; Horlbeck, L.; Lienkamp, M.; Hackl, C. Increasing the Overall Efficiency of Induction Motors for BEV by using the Overload Potential through Downsizing. In Proceedings of the IEEE International Electric Machines & Drives Conference (IEMDC 2017), Miami, FL, USA, 21–24 May 2017.
- Hackl, C.; Kullick, J.; Monzen, N. Optimale Betriebsführung für nichtlineare Synchronmaschinen. In *Elektrische Antriebe—Regelung von Antriebssystemen*; Böcker, J., Griepentrog, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2020.
- Hackl, C.M.; Kullick, J.; Monzen, N. Generic loss minimization for nonlinear synchronous machines by analytical computation of optimal reference currents considering copper and iron losses. In Proceedings of the 2021 IEEE International Conference on Industrial Technology (ICIT), Valencia, Spain, 10–12 March 2021. [[CrossRef](#)]
- Horlbeck, L.; Hackl, C.M. Analytical solution for the MTPV hyperbola including the stator resistance. In Proceedings of the IEEE International Conference on Industrial Technology (ICIT 2016), Taipei, Taiwan, 14–17 March 2016; pp. 1060–1067. [[CrossRef](#)]
- Holtz, J.; Qi, X. Optimal Control of Medium-Voltage Drives—An Overview. *IEEE Trans. Ind. Electron.* **2012**, *60*, 5472–5481. [[CrossRef](#)]
- Birda, A.; Reuss, J.; Hackl, C.M. Synchronous Optimal Pulsewidth Modulation for Synchronous Machines With Highly Operating Point Dependent Magnetic Anisotropy. *IEEE Trans. Ind. Electron.* **2021**, *68*, 3760–3769. [[CrossRef](#)]
- Birda, A.; Grabher, C.; Reuss, J.; Hackl, C.M. Dc-link capacitor and inverter current ripples in anisotropic synchronous motor drives produced by synchronous optimal PWM. *IEEE Trans. Ind. Electron.* **2021**, *69*, 4484–4494. [[CrossRef](#)]
- Morimoto, S.; Takeda, Y.; Hirasawa, T.; Taniguchi, K. Expansion of operating limits for permanent magnet motor by current vector control considering inverter capacity. *IEEE Trans. Ind. Appl.* **1990**, *26*, 866–871. [[CrossRef](#)]
- Niazi, P.; Toliyat, H.A.; Goodarzi, A. Robust Maximum Torque per Ampere (MTPA) Control of PM-Assisted SynRM for Traction Applications. *IEEE Trans. Veh. Technol.* **2007**, *56*, 1538–1545. [[CrossRef](#)]
- Cheng, B.; Tesch, T. Torque Feedforward Control Technique for Permanent-Magnet Synchronous Motors. *IEEE Trans. Ind. Electron.* **2010**, *57*, 969–974. [[CrossRef](#)]
- Finken, T. Fahrzyklusgerechte Auslegung von Permanentmagnet Synchronmaschinen für Hybrid- und Elektrofahrzeuge. Ph.D. Thesis, Institut für elektrische Maschinen, RWTH Aachen, Aachen, Germany, 2012.
- Jung, S.Y.; Hong, J.; Nam, K. Current Minimizing Torque Control of the IPMSM using Ferrari’s Method. *IEEE Trans. Power Electron.* **2013**, *28*, 5603–5617. [[CrossRef](#)]

15. Preindl, M.; Bolognani, S. Optimal State Reference Computation with Constrained MTPA Criterion for PM Motor Drives. *IEEE Trans. Power Electron.* **2015**, *30*, 4524–4535. [[CrossRef](#)]
16. Gemaßmer, T. Effiziente und Dynamische Drehmomenteinprägung in hoch Ausgenutzten Synchronmaschinen mit Eingebetteten Magneten. Ph.D. Thesis, Fakultät für Elektrotechnik und Informationstechnik, Karlsruher Institut für Technologie (KIT), Karlsruhe, Germany, 2015.
17. Lemmens, J.; Vanassche, P.; Driesen, J. PMSM Drive Current and Voltage Limiting as a Constraint Optimal Control Problem. *IEEE J. Emerg. Sel. Top. Power Electron.* **2015**, *3*, 326–338. [[CrossRef](#)]
18. Schoonhoven, G.; Uddin, M.N. MTPA- and FW-Based Robust Nonlinear Speed Control of IPMSM Drive Using Lyapunov Stability Criterion. *IEEE Trans. Ind. Appl.* **2016**, *52*, 4365–4374. [[CrossRef](#)]
19. Dianov, A.; Tinazzi, F.; Calligaro, S.; Bolognani, S. Review and classification of MTPA control algorithms for synchronous motors. *IEEE Trans. Power Electron.* **2021**, *37*, 3990–4007. [[CrossRef](#)]
20. Ahn, J.; Lim, S.B.; Kim, K.C.; Lee, J.; Choi, J.H.; Kim, S.; Hong, J.P. Field weakening control of synchronous reluctance motor for electric power steering. *IET Electr. Power Appl.* **2007**, *1*, 565–570. [[CrossRef](#)]
21. Tursini, M.; Chiricozzi, E.; Petrella, R. Feedforward Flux-Weakening Control of Surface-Mounted Permanent-Magnet Synchronous Motors Accounting for Resistive Voltage Drop. *IEEE Trans. Ind. Electron.* **2010**, *57*, 440–448. [[CrossRef](#)]
22. Preindl, M.; Bolognani, S. Model Predictive Direct Torque Control with Finite Control Set for PMSM Drive Systems, Part 2: Field Weakening Operation. *IEEE Trans. Ind. Inform.* **2013**, *9*, 648–657. [[CrossRef](#)]
23. Kim, J.; Jeong, I.; Nam, K.; Yang, J.; Hwang, T. Sensorless Control of PMSM in a High-Speed Region Considering Iron Loss. *IEEE Trans. Ind. Electron.* **2015**, *62*, 6151–6159. [[CrossRef](#)]
24. Zhang, P.; Ionel, D.M.; Demerdash, N.A.O. Saliency Ratio and Power Factor of IPM Motors With Distributed Windings Optimally Designed for High Efficiency and Low-Cost Applications. *IEEE Trans. Ind. Appl.* **2016**, *52*, 4730–4739. [[CrossRef](#)]
25. Eldeeb, H.; Hackl, C.M.; Horlbeck, L.; Kullick, J. A unified theory for optimal feedforward torque control of anisotropic synchronous machines. *Int. J. Control.* **2018**, *91*, 2273–2302 [[CrossRef](#)]
26. Panaitescu, R.C.; Topa, I. Optimal Control Method For PMSM By Minimization Of Electrical Losses. In Proceedings of the IEEE International Conference on Optimization of Electrical and Electronic Equipments (OPTIM), Brasov, Romania, 14–15 May 1998; Volume 2, pp. 451–456.
27. Urasaki, N.; Senjyu, T.; Uezato, K. A novel calculation method for iron loss resistance suitable in modeling permanent-magnet synchronous motors. *IEEE Trans. Energy Convers.* **2003**, *18*, 41–47. [[CrossRef](#)]
28. Cavallaro, C.; Tommaso, A.O.D.; Miceli, R.; Raciti, A.; Galluzzo, G.R.; Trapanese, M. Efficiency Enhancement of Permanent-Magnet Synchronous Motor Drives by Online Loss Minimization Approaches. *IEEE Trans. Ind. Electron.* **2005**, *52*, 1153–1160. [[CrossRef](#)]
29. Ni, R.; Xu, D.; Wang, G.; Ding, L.; Zhang, G.; Qu, L. Maximum Efficiency Per Ampere Control of Permanent-Magnet Synchronous Machines. *IEEE Trans. Ind. Electron.* **2015**, *62*, 2135–2143. [[CrossRef](#)]
30. Zhang, S. Artificial Intelligence in Electric Machine Drives: Advances and Trends. *arXiv* **2021**, arXiv:2110.05403.
31. Kumar, R.; Gupta, R.A.; Bansal, A.K. Identification and Control of PMSM Using Artificial Neural Network. In Proceedings of the 2007 IEEE International Symposium on Industrial Electronics, Vigo, Spain, 4–7 June 2007; pp. 30–35. [[CrossRef](#)]
32. Gaur, P.; Singh, B.; Mittal, A. Artificial Neural Network based Controller and Speed Estimation of Permanent Magnet Synchronous Motor. In Proceedings of the 2008 Joint International Conference on Power System Technology and IEEE Power India Conference, New Delhi, India, 12–15 October 2008. [[CrossRef](#)]
33. Zare, J. Vector control of permanent magnet synchronous motor with surface magnet using artificial neural networks. In Proceedings of the 2008 43rd International Universities Power Engineering Conference, Padua, Italy, 1–4 September 2008; pp. 1–4. [[CrossRef](#)]
34. Nagarajan, V.; Balaji, M.; Kamaraj, V.; Seetha, B. Comparative analysis of neural and P-I controller for PMSM Drive. In Proceedings of the 2014 IEEE 2nd International Conference on Electrical Energy Systems (ICEES), Chennai, India, 7–9 January 2014; pp. 126–131. [[CrossRef](#)]
35. Leena, N.; Shanmugasundaram, R. Artificial neural network controller for improved performance of brushless DC motor. In Proceedings of the 2014 International Conference on Power Signals Control and Computations (EPSCICON), Kerala, India, 6–11 January 2014; pp. 1–6. [[CrossRef](#)]
36. Suman, S.K.; Gautam, M.K.; Srivastava, R.; Giri, V.K. Novel approach of speed control of PMSM drive using neural network controller. In Proceedings of the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 3–5 March 2016; pp. 2780–2783. [[CrossRef](#)]
37. Guney, E.; Dursun, M.; Demir, M. Artificial neural network based real time speed control of a linear tubular permanent magnet direct current motor. In Proceedings of the 2017 International Conference on Control, Automation and Diagnosis (ICCAD), Hammamet, Tunisia, 19–21 January 2017. [[CrossRef](#)]
38. Ghlib, I.; Messlem, Y.; Chedjara, Z. ADALINE-Based Speed Control For Induction Motor Drive. In Proceedings of the 2019 International Conference on Advanced Electrical Engineering (ICAEE), Algiers, Algeria, 19–21 November 2019. [[CrossRef](#)]
39. Wishart, M.; Harley, R. Identification and control of induction machines using artificial neural networks. *IEEE Trans. Ind. Appl.* **1995**, *31*, 612–619. [[CrossRef](#)]

40. Rubaai, A.; Kotaru, R. Online identification and control of a DC motor using learning adaptation of neural networks. *IEEE Trans. Ind. Appl.* **2000**, *36*, 935–942. [[CrossRef](#)]
41. Kenne, G.; Ahmed-Ali, T.; Lamnabhi-Lagarrigue, F.; Nkwawo, H. Identification of electrical parameters and rotor speed of induction motor using radial basis neural network. In Proceedings of the 2004 IEEE International Symposium on Industrial Electronics, Ajaccio, France, 4–7 May 2004. [[CrossRef](#)]
42. Kirchgassner, W.; Wallscheid, O.; Böcker, J. Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors. In Proceedings of the 2019 IEEE International Electric Machines & Drives Conference (IEMDC), San Diego, CA, USA, 12–15 May 2019. [[CrossRef](#)]
43. Turabee, G.; Khowja, M.R.; Giangrande, P.; Madonna, V.; Cosma, G.; Vakil, G.; Gerada, C.; Galea, M. The Role of Neural Networks in Predicting the Thermal Life of Electrical Machines. *IEEE Access* **2020**, *8*, 40283–40297. [[CrossRef](#)]
44. Mosaad, M.I.; Banakher, F.A. Direct Torque Control of Synchronous Motors Using Artificial Neural Network. In Proceedings of the 2019 IEEE International Conference on Electro Information Technology (EIT), Brookings, SD, USA, 20–22 May 2019. [[CrossRef](#)]
45. Hammoud, I.; Hentzelt, S.; Oehlschlaegel, T.; Kennel, R. Long-Horizon Direct Model Predictive Control Based on Neural Networks for Electrical Drives. In Proceedings of the IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, Singapore, 19–21 October 2020. [[CrossRef](#)]
46. 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](#)]
47. Yan, Y.B.; Liang, J.N.; Sun, T.F.; Geng, J.P.; Xie, G.; Pan, D.J. Torque Estimation and Control of PMSM Based on Deep Learning. In Proceedings of the 2019 22nd International Conference on Electrical Machines and Systems (ICEMS), Harbin, China, 11–14 August 2019; pp. 1–6. [[CrossRef](#)]
48. Matsuura, K.; Akatsu, K. A motor control method by using Machine learning. In Proceedings of the 2020 23rd International Conference on Electrical Machines and Systems (ICEMS), Hamamatsu, Japan, 24–27 November 2020; pp. 652–655. [[CrossRef](#)]
49. Wlas, M.; Krzeminski, Z.; Guzinski, J.; Abu-Rub, H.; Toliyat, H. Artificial-Neural-Network-Based Sensorless Nonlinear Control of Induction Motors. *IEEE Trans. Energy Convers.* **2005**, *20*, 520–528. [[CrossRef](#)]
50. Garcia, P.; Briz, F.; Raca, D.; Lorenz, R.D. Saliency-Tracking-Based Sensorless Control of AC Machines Using Structured Neural Networks. *IEEE Trans. Ind. Appl.* **2007**, *43*, 77–86. [[CrossRef](#)]
51. Echenique, E.; Dixon, J.; Cardenas, R.; Pena, R. Sensorless Control for a Switched Reluctance Wind Generator, Based on Current Slopes and Neural Networks. *IEEE Trans. Ind. Electron.* **2009**, *56*, 817–825. [[CrossRef](#)]
52. Zine, W.; Makni, Z.; Monmasson, E.; Idkhajine, L.; Condamine, B. Interests and Limits of Machine Learning-Based Neural Networks for Rotor Position Estimation in EV Traction Drives. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1942–1951. [[CrossRef](#)]
53. Flieller, D.; Nguyen, N.K.; Wira, P.; Sturtzer, G.; Abdeslam, D.O.; Merckle, J. A Self-Learning Solution for Torque Ripple Reduction for Nonsinusoidal Permanent-Magnet Motor Drives Based on Artificial Neural Networks. *IEEE Trans. Ind. Electron.* **2014**, *61*, 655–666. [[CrossRef](#)]
54. Tarczewski, T.; Niewiara, L.; Grzesiak, L.M. Torque ripple minimization for PMSM using voltage matching circuit and neural network based adaptive state feedback control. In Proceedings of the 2014 16th European Conference on Power Electronics and Applications, Lappeenranta, Finland, 26–28 August 2014. [[CrossRef](#)]
55. Azzini, A.; Cristaldi, L.; Lazzaroni, M.; Monti, A.; Ponci, F.; Tettamanzi, A. Incipient Fault Diagnosis in Electrical Drives by Tuned Neural Networks. In Proceedings of the 2006 IEEE Instrumentation and Measurement Technology Conference Proceedings, Sorrento, Italy, 24–27 April 2006. [[CrossRef](#)]
56. Marmouch, S.; Aroui, T.; Koubaa, Y. Application of statistical neuronal networks for diagnostics of induction machine rotor faults. In Proceedings of the 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, 21–23 December 2017. [[CrossRef](#)]
57. Pasqualotto, D.; Zigliotto, M. A comprehensive approach to convolutional neural networks-based condition monitoring of permanent magnet synchronous motor drives. *IET Electr. Power Appl.* **2021**, *15*, 947–962. [[CrossRef](#)]
58. Botache, D.; Bethke, F.; Hardieck, M.; Bieshaar, M.; Brabetz, L.; Ayeb, M.; Zipf, P.; Sick, B. Towards Highly Automated Machine-Learning-Empowered Monitoring of Motor Test Stands. In Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), Washington, DC, USA, 27 September–1 October 2021. [[CrossRef](#)]
59. Puron, L.D.R.; Neto, J.E.; Fernandez, I.A. Neural networks based estimator for efficiency in VSI to PWM of induction motors drives. In Proceedings of the 2016 IEEE International Conference on Automatica (ICA-ACCA), Curico, Chile, 19–21 October 2016. [[CrossRef](#)]
60. Zăvoianu, A.C.; Bramerdorfer, G.; Lughofer, E.; Silber, S.; Amrhein, W.; Klement, E.P. Hybridization of multi-objective evolutionary algorithms and artificial neural networks for optimizing the performance of electrical drives. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1781–1794. [[CrossRef](#)]
61. Hackl, C.M.; Kamper, M.J.; Kullick, J.; Mitchell, J. Current control of reluctance synchronous machines with online adjustment of the controller parameters. In Proceedings of the 2016 IEEE International Symposium on Industrial Electronics (ISIE 2016), Santa Clara, CA, USA, 8–10 June 2016; pp. 153–160. [[CrossRef](#)]
62. Hackl, C.; Kullick, J.; Landsmann, P. Nichtlineare Stromregelverfahren für Reluktanz-Synchronmaschinen. In *Elektrische Antriebe—Regelung von Antriebssystemen*; Böcker, J., Griepentrog, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2020.

63. Schröder, D.; Buss, M. *Intelligente Verfahren: Identifikation und Regelung Nichtlinearer Systeme*; Springer: Berlin/Heidelberg, Germany, 2017. [[CrossRef](#)]
64. Aggarwal, C.C. *Neural Networks and Deep Learning: A Textbook*; Springer International Publishing: Cham, Switzerland, 2018. [[CrossRef](#)]
65. Stursa, D.; Dolezel, P. Comparison of ReLU and linear saturated activation functions in neural network for universal approximation. In Proceedings of the 2019 22nd International Conference on Process Control (PC19), Strbske Pleso, Slovakia, 11–14 June 2019; pp. 146–151. [[CrossRef](#)]
66. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning*; Springer Series in Statistics; Springer: New York, NY, USA, 2009.
67. Lee, D.K.; In, J.; Lee, S. Standard deviation and standard error of the mean. *Korean J. Anesthesiol.* **2015**, *68*, 220. [[CrossRef](#)]
68. Jaiswal, P.; Gupta, N.K.; Ambikapathy, A. Comparative study of various training algorithms of artificial neural network. In Proceedings of the 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, Uttar Pradesh, India, 12–13 October 2018; pp. 1097–1101. [[CrossRef](#)]
69. Ruszczyński, A.P.; Shapiro, A. (Eds.) *Stochastic Programming*; Number v. 10 in Handbooks in Operations Research and Management Science; Elsevier: Amsterdam, The Netherlands; Boston, MA, USA, 2003.
70. Hackl, C.M. *Non-Identifier Based Adaptive Control in Mechatronics: Theory and Application*; Number 466 in Lecture Notes in Control and Information Sciences; Springer International Publishing: Berlin/Heidelberg, Germany, 2017. [[CrossRef](#)]