Electrical Engineering (2020) 102:1103–1116
https://doi.org/10.1007/s00202-020-00935-6

ORIGINAL PAPER

Model-based control algorithm development of induction machines by using a well-defined model architecture and rapid control prototyping

Krisztián Horváth[1] · Márton Kuslits[2] · Szilárd Lovas[1]

Received: 3 January 2019 / Accepted: 25 January 2020 / Published online: 10 February 2020

Abstract

This paper presents a new control algorithm development approach for induction machines by using model-based design and a systematically built model architecture implemented in MATLAB/Simulink. The model architecture follows a three-layer structure, and it is developed according to the principle of functional decomposition and the needs of reusability and expandability. The first model layer consists of elementary model and algorithm components, the second contains a machine simulation model and a field-oriented control (FOC) algorithm, built upon the first layer's components, and the third realises the executable models by connecting the models and algorithms defined in the second layer. Furthermore, rapid control prototyping (RCP) is discussed as an experimental validation method, and an experimental setup with RCP is also introduced. The application of the presented methods is demonstrated by simulations as well as by experiments, and by using a control algorithm based on FOC as an example.

Keywords Induction machine · Control algorithm development · Field-oriented control (FOC) · Model-based design (MBD) · Modular architecture · Rapid control prototyping (RCP) · MATLAB/Simulink

Abbreviations

DRFOC	Direct rotor field-oriented control
DSC	Direct self-control
DTC	Direct torque control
ECU	Electronic control unit
FOC	Field-oriented control
HIL	Hardware-in-the-loop
MBD	Model-based design
NIC	Network interface control
PMSM	Permanent magnet synchronous machine
PWM	Pulse width modulation
RCP	Rapid control prototyping
SRTT	Simulink Real-Time target
TI	Texas instruments
UI	User interface

List of symbols

i_{ds}	d-Axis stator current (A)
i_{qs}	q-Axis stator current (A)
i_{dr}	d-Axis rotor current (A)
i_{qr}	q-Axis rotor current (A)
i_a, i_b, i_c	Phase currents (A)
v_{ds}	d-Axis stator voltage (V)
v_{qs}	q-Axis stator voltage (V)
v_{dr}	d-Axis rotor voltage (V)
v_{qr}	q-Axis rotor voltage (V)
v_a, v_b, v_c	Phase voltages (V)
ψ_{ds}	d-Axis stator flux (Wb)
ψ_{qs}	q-Axis stator flux (Wb)
ψ_{dr}	d-Axis rotor flux (Wb)
ψ_{qr}	q-Axis rotor flux (Wb)
ω	Speed of reference frame (rad/s)
ω_r	Rotor electrical speed (rad/s)
ω_m	Rotor mechanical speed (rad/s)
φ	Position of reference frame (rad)
φ_m	Rotor mechanical position (rad)
R_s	Stator resistance (Ω)
R_r	Rotor resistance (Ω)
L_s	Stator inductance (H)
L_r	Rotor inductance (H)
L_m	Mutual inductance (H)
T_e	Electromagnetic torque (Nm)

✉ Krisztián Horváth
krisztian.horvath@sze.hu

[1] Széchenyi István University, Egyetem tér 1., Gyor 9026, Hungary

[2] Independent researcher, Gyor, Hungary

T_l	Load torque (Nm)
p	Number of pole pairs
J	Rotor inertia (kg m^2)
D	Viscous friction coefficient (Nm s/rad)
T_0	Static friction (Nm)

1 Introduction

Induction machines are widely used for their well-known advantages such as reliability, robustness, and low cost. Besides their advantages, induction machines may require complex control solutions when used in demanding, sometimes safety-critical applications like, e.g. electric vehicle drives. Development of control algorithms and embedded software for motor controllers, however, may be challenging, and software quality, reusability, and functional safety measures may be required. In this context, model-based design[1] (MBD) methods may help to fulfil the emerging requirements. MBD relies on the joint simulation model of the machine and the associated control algorithm under development, i.e. the control algorithm is developed in a simulation environment which provides early validation capabilities by simulation experiments. This approach speeds up the development and improves the software quality as it helps to reveal errors and design flaws early [1].

Although the systematic application of MBD is not widespread, yet in motor control development, several papers deal with related problems. Since models have a distinguished role in MBD, one may review first the literature which deals with simulation models of machines. These models are often implemented in MATLAB/Simulink environment as that is used widely for simulation and control design of electric drives. The fundamental literature in this topic is [2]. In [3], an early implementation of electric drive models in MATLAB/Simulink is described. An important contribution of that study is the introduction of a higher abstraction level approach based on physical signal connections. The latter is the state-of-the-art simulation technology today as used in many simulation environments like MATLAB/Simscape and Modelica. Paper [4] describes an induction motor simulation model, including control algorithms and their implementations in MATLAB/Simulink. Studies [5–7] focus on the application of MBD methods with emphasis on motor control software development. Paper [8] describes the MBD-procedure of the embedded control software of an on-board charger device. Besides the general presentation of the MBD-procedure, it focuses on safety and software quality issues of embedded systems and points out that the application of MBD is a possible way to overcome the related problems. Study [9] discusses concepts w.r.t. the utilisation of the block diagram interface and block hierarchy of MATLAB/Simulink. The same study describes modelling considerations like, e.g. functional decomposition as well. Paper [4] also describes some aspects of functional decomposition similarly to [9] and presents a reference implementation of models and algorithms by using the user-defined library features of MATLAB/Simulink. However, the usage of these features is spontaneous rather than systematic. In paper [10], component variability and reusability concepts in MATLAB/Simulink environment are discussed.

Regarding the control algorithms of induction machines, fundamental methods are field-oriented control (FOC) by Blaschke [11] and direct torque control (DTC) by Takahashi [12]. Alternatively, Depenbrock [13] proposed an approach similar to DTC, the direct self-control (DSC). Study [14] compares FOC and DTC methods and concludes that FOC usually shows lower current and torque ripple and may provide better steady-state performance besides very low speeds. In contrary, better torque response may be achieved by using DTC (or DSC), and DTC does not require any mechanical transducers on the machine shaft in contrast to FOC (see in [14] as well). By using estimator algorithms, however, rotational sensors might be eliminated from the FOC scheme as well, see [15–17] for example. Furthermore, there are several variants of FOC schemes. For squirrel cage induction machines, the direct rotor field-oriented control (DRFOC) technique is usually applied, as discussed in studies [18,19]. Since FOC-based methods have become industrial standard algorithms for induction machine control (see [20]) and this paper aims at a new development methodology of complex motor control algorithms, only DRFOC is considered as a control algorithm example in the rest of the paper.

If one regards the deployment of control algorithms into an actual physical environment, rapid control prototyping (RCP) becomes a key technology in the MBD workflow. RCP consists of a dedicated high-performance real-time capable target computer and the corresponding software tools with which the control algorithms may be validated efficiently in the actual physical environment, replacing all or bypassing some of the functions of the electronic control unit (ECU) of the system. Depending on the latter, one may consider *ECU fullpass* and *ECU bypass* RCP configurations, see [21–23]. The same papers discuss the background and principles of RCP applications, describe the general requirements of RCP systems (surplus performance, broad range of I/O, transparency through virtual instrumentation, etc.), and distinguish between ECU fullpass and ECU bypass configurations. Textbook [24] describes RCP as a method applied typically for ECU bypassing and presents an application

[1] For avoiding confusion, one should be aware that the term *model-based* refers to the development method instead of models potentially applied in control algorithms as in the case of internal model control methods, for example.

example related to engine control development. Study [25] discusses various RCP applications, especially ECU bypassing over CAN. RCP is applied in control development of electric drive systems as well. Giving a big picture, paper [26] discusses the complete development procedure of electric drive control systems, including MBD, RCP, and hardware-in-the-loop (HIL) testing methods. Other works focus on more specific problems. For example, Rubaai et al. [27] apply RCP in a cascade ECU fullpass configuration for control development of servo drives, study [28] for a cascade position control of a permanent magnet synchronous machine (PMSM)-based servo system, and [29] in ECU fullpass configuration for development of speed control algorithm for brushless DC machine. In paper [30], RCP and HIL solutions are utilised for failure analysis of induction machine drives. An RCP system is also applied in [31], where a rotor bar break detection method is presented and investigated experimentally. Studies [32,33] use RCP approach to verify a dynamic emulation strategy of mechanical loads and for a novel model predictive speed control method of a DC motor, respectively.

There are mixed methods (e.g. on-target prototyping) involving some RCP-like solutions as well. For example, work [34] applies on-target RCP for FOC and servo algorithm development of induction machines. Similarly, Kang and Park [35] use on-target RCP for induction machine FOC with space vector modulation. Paper [36] presents an RCP-like approach for on-target simulation in order to develop FOC for PMSMs.

Code generation for ECUs is also an aspect of MBD methodology, and application of code generation tools represents the state-of-the-art method for production target code implementation of embedded systems. Paper [37] discusses how the code generation capabilities of MATLAB/Simulink can be applied in the development process of critical embedded systems. Code generation applied in controller development of PMSMs is discussed in [38].

The present paper describes a new approach to control algorithm development of induction machines. The presented approach follows the MBD methodology, strongly relies on MATLAB/Simulink features, and—contrary to the prior works—strictly follows a systematic modelling concept. For algorithm design and simulation, a reusable model architecture is introduced, which is a novelty in motor control development. The presented architecture follows a three-layer structure. The first model layer contains the elementary model and algorithm components, defined by the principle of functional decomposition. The second layer contains a machine simulation model and a DRFOC algorithm, built upon the first layer's components, while the third realises the executable models by connecting the models and algorithms defined in the second layer. Moreover, the application of the proposed method including the experimental validation by RCP is demonstrated by using the DRFOC-based control algorithm as an example. The RCP configuration is built in such a way that the software component realising the DRFOC algorithm is deployed onto a specialised high-performance target computer ensuring the excessive computing power and transparency required for research or early-stage development activities. The authors disclose that the paper extends the results of the former conference paper [39]. As novelties over [39], the model architecture is extended, an RCP-application is included, the demonstration models and algorithms are improved, and an experimental setup is developed as well.

The content of the paper is organised as follows. Section 2 describes the theoretical induction machine model and the applied DRFOC algorithm used in the rest of the paper. Section 3 describes the core contribution of the paper, namely the new model-based development approach, including the model architecture, principles of RCP, and the experimental setup. Finally, Sect. 4 presents an implementation example with experimental results based on the principles and methods described in Sect. 3.

2 Theoretical models and algorithms applied in the paper

This section describes the theoretical models and algorithms applied in the paper. The simplified system model of an induction motor drive consists of a plant model and a control algorithm where the plant model is a lumped parameter model of a squirrel cage induction machine, and the control algorithm is the DRFOC. All of the models and algorithms presented in this section are derived according to [20,40,41].

2.1 Lumped parameter induction machine model

The dynamic model of an induction machine is a lumped parameter type by its nature and consists of an electrical and a mechanical submodel. The standard notations of the machine model are listed in the list of symbols.

2.1.1 Electrical model

The electrical model characterises the electromagnetic behaviour of the induction machine. The model may be formalised by voltage equations written in a two-phase reference frame. The well-known stator voltage equations in rotating reference frame follow as

$$v_{\mathrm{ds}} = R_{\mathrm{s}} i_{\mathrm{ds}} + \frac{\mathrm{d}}{\mathrm{d}t}\psi_{\mathrm{ds}} - \omega\psi_{\mathrm{qs}} \tag{1}$$

and

$$v_{qs} = R_s i_{qs} + \frac{d}{dt}\psi_{qs} + \omega\psi_{ds}. \tag{2}$$

Similarly, rotor voltage equations may be written in the same rotating reference frame as

$$v_{dr} = R_r i_{dr} + \frac{d}{dt}\psi_{dr} - (\omega - \omega_r)\psi_{qr} \tag{3}$$

and

$$v_{qr} = R_r i_{qr} + \frac{d}{dt}\psi_{qr} + (\omega - \omega_r)\psi_{dr} \tag{4}$$

where $v_{dr} = 0$ and $v_{qr} = 0$ because in squirrel cage induction machines the rotor electrical circuit is short-circuited.

$\psi_{ds}, \psi_{qs}, \psi_{dr}$, and ψ_{qr} may be calculated by flux equations. Flux equations may be written for both the stator and the rotor. Stator flux equations result as

$$\psi_{ds} = L_s i_{ds} + L_m i_{dr} \tag{5}$$

and

$$\psi_{qs} = L_s i_{qs} + L_m i_{qr}. \tag{6}$$

Similarly, rotor flux equations are

$$\psi_{dr} = L_r i_{dr} + L_m i_{ds} \tag{7}$$

and

$$\psi_{qr} = L_r i_{qr} + L_m i_{qs}. \tag{8}$$

By using the stator current and rotor flux components, the electromagnetic torque of the machine follows as

$$T_e = \frac{3}{2} p \frac{L_m}{L_r}(i_{qs}\psi_{dr} - i_{ds}\psi_{qr}). \tag{9}$$

Equations (1)–(9) represent the electromagnetic behaviour of the induction machine in any rotating reference frame. Equations (1)–(4) become simpler by substituting $\omega = 0$; therefore, this choice of ω is applied in the following. This choice of ω, however, is applied only for the electrical model of the machine. For the control algorithm, there is a different choice.

2.1.2 Mechanical model

The mechanical part of the machine model is defined by the equation of motion as

$$\frac{d}{dt}\omega_m = \frac{T_e - T_l - D\omega_m - \mathrm{sgn}(\omega_m)T_0}{J} \tag{10}$$

where $\omega_m = \frac{1}{p}\omega_r$.

2.2 DRFOC algorithm

This section introduces the applied control algorithm, which is a variant of the DRFOC. The block diagram of a commonly used DRFOC algorithm is depicted in Fig. 1. As shown in that figure, the magnetic field and the electromagnetic torque are controlled independently from each other, while the rotating reference frame is fixed to the rotor flux vector.

In the chosen reference frame, the rotor flux is equal to ψ_{dr} and $\psi_{qr} = 0$. Based on Eqs. (3) and (7), the ψ_{dr} may be calculated from the i_{ds} stator current component by using the transfer function

$$G(s) = \frac{\Psi_{dr}(s)}{I_{ds}(s)} = \frac{L_m}{\frac{L_r}{R_r}s + 1}. \tag{11}$$

The rotor flux position may be calculated from Eqs. (4) and (8) as

$$\varphi = p\varphi_m + \frac{R_r L_m}{L_r}\int \frac{i_{qs}}{\psi_{dr}}dt \tag{12}$$

where $\varphi_m = \int \omega_m dt$.

According to the DRFOC block diagram in Fig. 1, the flux and torque of the machine are controlled in the rotor flux-oriented reference frame through the d- and q-components of the stator current, respectively. In the applied scheme, phase currents i_a and i_b and the mechanical position of the rotor φ_m are measured. The phase current i_c is calculated according to Kirchhoff's law for Y-circuit:

$$i_a + i_b + i_c = 0. \tag{13}$$

In order to obtain the stator current components i_{ds} and i_{qs} from the phase currents, the Clarke- and Park- transformations are applied, which may be written by the following matrices:

$$C = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \text{ and } P = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

The current references for the control loops are calculated from the flux and torque references as follows. The d-axis current reference results as

$$i_{ds}^{ref} = \frac{\psi^{ref}}{L_m} \tag{15}$$

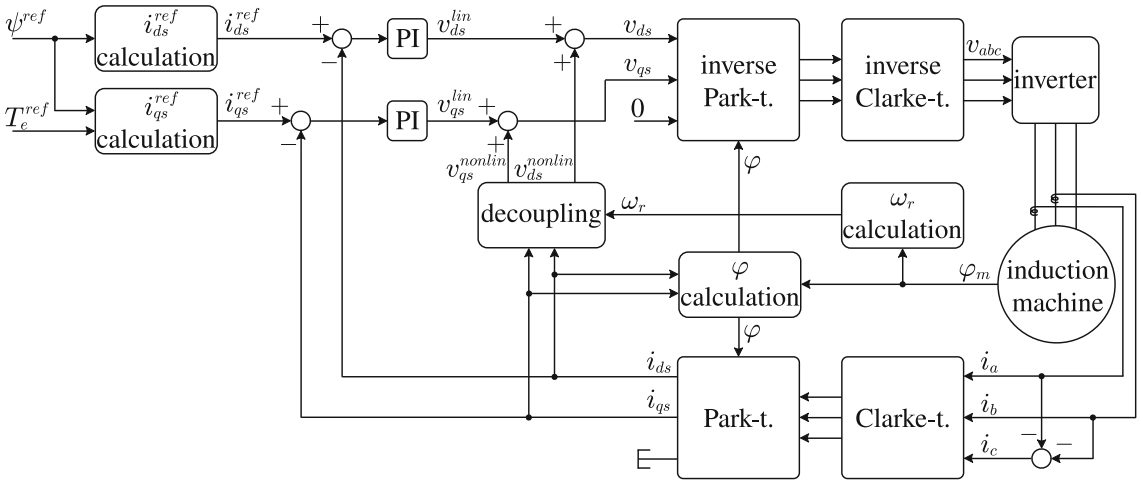

Fig. 1 Block diagram of the DRFOC algorithm

where ψ^{ref} is the flux reference. The q-axis current reference may be calculated as

$$i_{\mathrm{qs}}^{\mathrm{ref}} = \frac{2}{3}\frac{1}{p}\frac{L_{\mathrm{r}}}{L_{\mathrm{m}}}\frac{T_{\mathrm{e}}^{\mathrm{ref}}}{\psi^{\mathrm{ref}}} \tag{16}$$

where $T_{\mathrm{e}}^{\mathrm{ref}}$ is the electromagnetic torque reference.

In the current control loops, the PI controllers may be designed to the linear parts of the plant model, which may be written as

$$v_{\mathrm{ds}}^{\mathrm{lin}} = R_{\mathrm{s}} i_{\mathrm{ds}} + L_{\mathrm{s}}\sigma\frac{\mathrm{d}}{\mathrm{d}t} i_{\mathrm{ds}} \tag{17}$$

and

$$v_{\mathrm{qs}}^{\mathrm{lin}} = R_{\mathrm{s}} i_{\mathrm{qs}} + L_{\mathrm{s}}\sigma\frac{\mathrm{d}}{\mathrm{d}t} i_{\mathrm{qs}} \tag{18}$$

where σ is the leakage coefficient, i.e. $\sigma = 1 - \frac{L_{\mathrm{m}}^2}{L_{\mathrm{s}} L_{\mathrm{r}}}$.

In order to improve the control performance, the outputs of the PI controllers should be corrected by the nonlinear parts of the voltage equations. These nonlinear parts are calculated by the decoupling algorithm according to the following equations:

$$v_{\mathrm{ds}}^{\mathrm{nonlin}} = \frac{L_{\mathrm{m}}^2 R_{\mathrm{r}}}{L_{\mathrm{r}}^2}\left(i_{\mathrm{ds}} - \frac{\psi_{\mathrm{dr}}}{L_{\mathrm{m}}}\right) - L_{\mathrm{s}}\sigma\left(\omega_{\mathrm{r}} + \frac{R_{\mathrm{r}} L_{\mathrm{m}}}{L_{\mathrm{r}}}\frac{i_{\mathrm{qs}}}{\psi_{\mathrm{dr}}}\right) i_{\mathrm{qs}} \tag{19}$$

and

$$v_{\mathrm{qs}}^{\mathrm{nonlin}} = \left(\omega_{\mathrm{r}} + \frac{R_{\mathrm{r}} L_{\mathrm{m}}}{L_{\mathrm{r}}}\frac{i_{\mathrm{qs}}}{\psi_{\mathrm{dr}}}\right)\left(L_{\mathrm{s}}\sigma i_{\mathrm{ds}} + \frac{L_{\mathrm{m}}}{L_{\mathrm{r}}}\psi_{\mathrm{dr}}\right). \tag{20}$$

Finally, the phase voltages may be calculated from v_{ds} and v_{qs} by using inverse coordinate transformations. The inverse Park- and Clarke-transformations may be obtained as multiplications by P^{-1} and C^{-1}. As shown in Fig. 1, the outputs of the inverse Clarke-transformation block act as control signals of the inverter.

3 Key elements of the proposed MBD-approach: modular model architecture and RCP

This section is the core contribution of the paper as it presents the new MBD-based development approach through its key elements. Firstly, the new, well-defined three-layer modular model architecture is introduced with emphasis on software development and maintenance aspects. Functional decomposition and componentisation aspects of the machine model and the DRFOC algorithm are discussed as well. Subsequently, an RCP concept including an experimental setup is described for validation of the control algorithm in the actual physical environment. Finally, certain aspects of embedded code generation are discussed.

3.1 Derivation of the three-layer model architecture

Before the implementation of the simulation model of the induction machine, one may take some modelling considerations into account. For example, Eqs. (1)–(9) and (10) are two loosely coupled descriptions of the electrical and mechanical behaviour of the induction machine. Because of the loose connection, it is straightforward to treat these equations separately when they are implemented in a simulation model. Similar considerations also may apply to the DRFOC algorithm. As shown in Fig. 1, the control algorithm consists of many well-defined components like the coordinate transformations, PI controllers, etc., as described by expres-

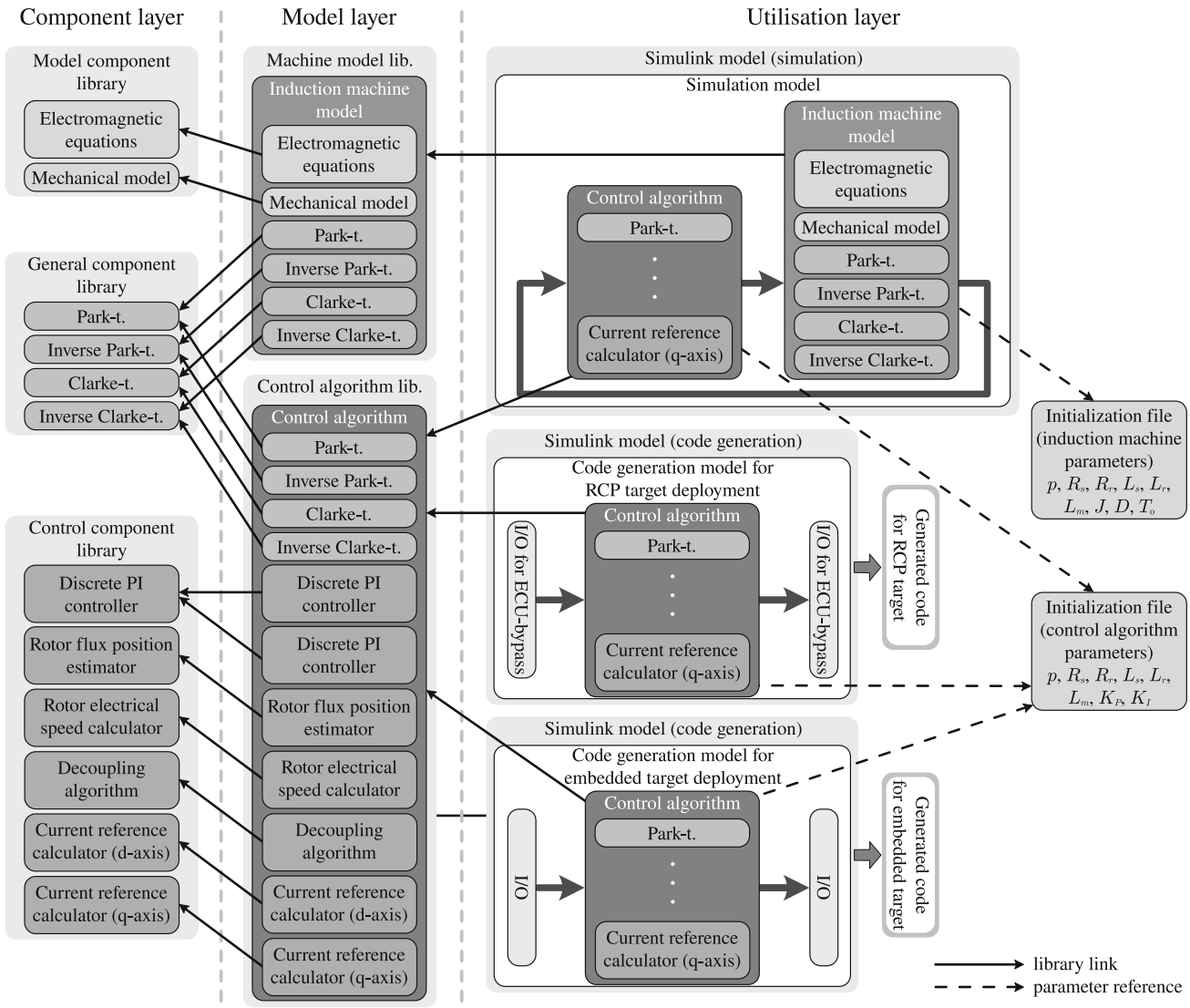

Fig. 2 Three-layer architecture of libraries and models

sions (12)–(20) as well. These components might be treated separately during the implementation procedure, leading to functional decomposition of the complete system.

Although concepts like functional decomposition and componentisation are general, this section presents techniques which are partially dependent on the application of MATLAB/Simulink. Features of MATLAB/Simulink like *block libraries*, *library links*, *masks*, and *model referencing* provide algorithm development tools which are similar in function to object-oriented programming methods [42]. Libraries contain the prototypes of blocks and library links, and masks realise instantiation of the linked components. These properties render MATLAB/Simulink to be a useful environment for the development of complex algorithms like control algorithms of induction machines.

By using the mentioned features of MATLAB/Simulink, almost infinite variations of functional decompositions and model architectures may be implemented. In order to find a suitable choice, one should consider the complexity of the system. In this case, a three-layer architecture of libraries and models has been created, see in Fig. 2. Layers are the *component layer*, the *model layer*, and the *utilisation layer*. The component layer contains the atomic components of the system, which result from the functional decomposition. Components are grouped into three separate libraries containing simulation model components, control algorithm components, and general components. The applied structure of grouping follows from the nature of the motor control problem; however, a different structure might be applicable as well. The second layer is the model layer, which contains the main components of the complete system, namely

the simulation model of the induction motor, and the control algorithm[2] (DRFOC). Components applied in these subsystems are linked to the prototypes in the component layer libraries. The third layer is the utilisation layer, where the models are actually applied, either for simulation or for code generation. This layer realises the instantiation of components; thus, parameters are applied in this place.

The presented three-layer structure ensures the variability of components and models as well, similarly to [10]. Variability means that libraries in the first and second layers might be changed or replaced by alternate versions. By using this capability, it is simple to develop modified versions of the control algorithm. Similarly, it is also easy to adapt the system for different machines by associating the existing algorithms with altered parameter files.

The presented architecture also facilitates software maintenance. Regarding the general component library, for example, it is apparent that any change in that library will propagate automatically through the links to all instances of the affected components. Thus, if a component is instantiated multiple times, it is sufficient to maintain only its prototype implementation.

3.2 Rapid control prototyping

RCP may be regarded within the MBD-procedure as a counterpart of simulations, as it similarly allows early validation, but in the actual physical environment. By using RCP, control algorithms may be efficiently validated in the actual physical environment, even if the final target hardware (ECU) is not available yet. The core of RCP is a dedicated real-time capable target computer and the associated software stack which provide

- surplus computing performance and a high number of I/O channels,
- quick and easy access by automated target configuration, code generation, and deployment driven by the modelling environment,
- a transparent execution environment in which all parameters may be tuned, and all signals may be monitored in real-time and logged with high sampling rate, and
- additional supportive tools like user interface for virtual instrumentation, experiment control, and automation.

By using RCP, we may embed and execute control algorithms in the actual physical environment, besides the advantageous conditions listed above. These properties also open the opportunity to execute computationally ineffective experimental algorithms and to iterate with them rapidly, as may occur in early development phases and research projects often; therefore, RCP holds a key role in MBD. The same properties apply for HIL applications as well, although the aim is different in that case. Usually, similar or the same tools may be used for both RCP and HIL applications.

RCP systems may be set up in different configurations. In the simplest case, the RCP target computer completely replaces and substitutes the ECU of the actual system. In this case, the RCP target computer has the same interface as the replaced ECU has and obviously, all the functions performed by the ECU are done by it. In the literature, this configuration is often called as *ECU fullpass* [22,23].

An alternate and more widely used configuration is that the original ECU still remains in the loop and does the interface handling and maybe executes some parts of its original functions. In this case, only a specific part of the control algorithm is executed on the RCP target computer, which is connected to the ECU through a dedicated communication interface. This configuration is usually called as *ECU bypass*, and the dedicated communication interface as the *ECU bypass interface* [21–25].

Some literature mentions the possibility of the so-called *on-target prototyping* as a possible variant of RCP, see [23] for example. In that case, however, the target platform is still the original ECU; thus, some of the above listed properties are not available.

In this work, an ECU bypass RCP configuration is used as follows. A PC-based target computer (HP BX383AV with Intel Core i5-2500 @ 3.3 GHz, see [43]) is applied with software tools provided by the Simulink Real-Time toolbox; thus, the target computer is called Simulink Real-Time target (SRTT) hereinafter. The bypassed ECU is a motor controller which is based on a Texas Instruments (TI) Hercules TMS570LC437 microcontroller [44] and the associated TI DRV8301-HC-EVM Rev-D power stage, which includes a three-phase full bridge circuit driven by pulse width modulation (PWM) [45]. In addition, an Analog Devices EVAL-AD2S1210EDZ resolver interface is applied for rotor position measurement [46]. The ECU bypass interface is based on raw Ethernet communication, which is performed by a TI TMDX570LC43HDK 10/100 Mbit/sec network interface controller (NIC) on the ECU side, and by an Intel Gigabit Ethernet NIC on the SRTT side [47]. Ethernet is applied here due to its relative high package frequency and bandwidth capacity which are required for this ECU bypass configuration since the DRFOC algorithm is executed at high sampling rate. The complete experimental setup is supervised and controlled by the host computer, which provides reference and enable signals from its virtual instrumentation user interface (UI). The layout of the physical connections of these components is depicted in Fig. 3, while the actual experimental setup is shown in Fig. 4.

[2] To be precise: the term *control algorithm* refers to *model implementation of the control algorithm* here.

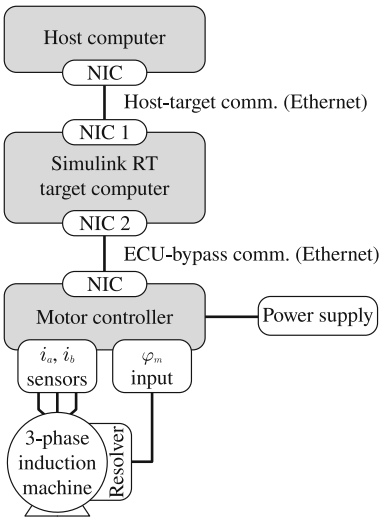

Fig. 3 Components of the applied RCP configuration

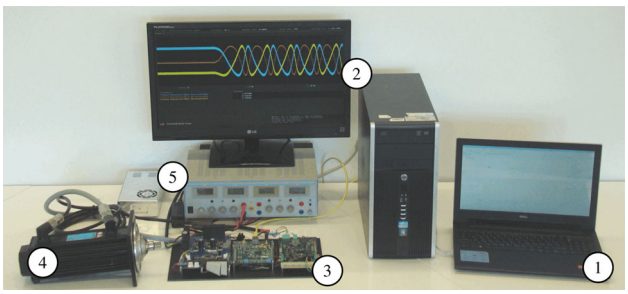

Fig. 4 Experimental setup: host computer 1, SRTT 2, motor controller 3, induction machine 4, power supplies 5

The motor controller performs the measurement of currents and the rotor mechanical position, controls the PWM output, and deals with low-level functions like, e.g. overcurrent protection. The DRFOC algorithm itself, however, is bypassed and executed by the SRTT. In order to execute the DRFOC algorithm on the SRTT, all the necessary signals like measured phase currents and resolver position are sent through the bypass interface. After receiving these, the SRRT executes the DRFOC algorithm, and sends back the results (phase voltages) to the motor controller through the bypass interface. Additional signals like references and enabling are originated from the host computer's UI. The flow of these signals is observed in Fig. 5.

The procedure is driven by the SRTT which acts as a master in the ECU bypass communication. After the motor controller receives a message consisting the updated phase voltages, an interrupt is executed, which triggers the response message, which includes the last measurements of i_a, i_b, and φ_m. In the meantime, the update of the PWM outputs and the synchronised measurement of i_a and i_b are performed on the motor controller side. The whole cycle is executed with the sampling time $T_\mathrm{s} = 10^{-4}$ s, which is the sampling time of

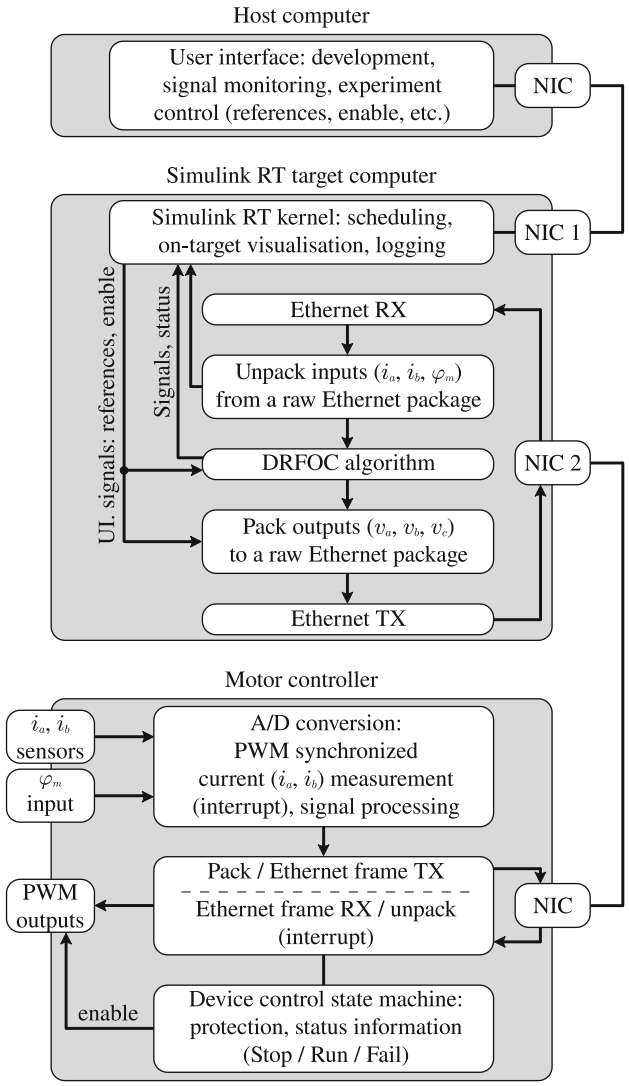

Fig. 5 Signal flow within the RCP configuration

the control algorithm as well. The sequence of execution is seen in Fig. 6.

3.3 Automated code generation and integration into an embedded target device

When the developed control algorithm reaches the required maturity, it may be deployed on its final embedded target (ECU). Within the frames of MBD, state-of-the-art tools provide capabilities for automated target code generation from algorithm models defined on higher abstraction levels. In embedded system development, MATLAB/Simulink and Embedded Coder are widely used tools for this purpose. Target code can be directly generated from Simulink subsystems. However, it is practical to perform the code generation procedure from a dedicated Simulink model. The main reason for this is that this approach allows target-

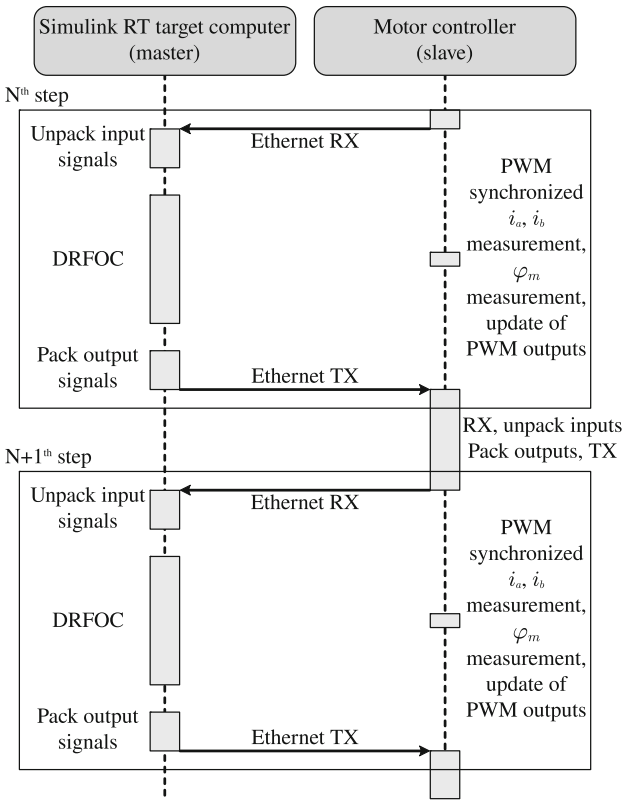

Fig. 6 Execution sequence diagram

specific components (e.g. interface blocks) to be added to the model, which are not relevant in a simulation otherwise. Integration of the generated code into the complete software stack might be easier by using target-specific modifications. The model and library architecture presented in Fig. 2 support this approach (see the part of the code generation model for embedded target deployment) because the control algorithm in the code generation model is linked to the corresponding library and therefore consistency between the algorithm design and the generated code is ensured.

Typically, the top level of the control algorithm is generated into a single function in the target language. In order to ensure the proper operation of the algorithm, the generated function has to be integrated into a real-time execution environment on the target device. The execution environment must provide the same sampling time (T_s), which is considered for the control design and applied in the simulation model. This may be fulfilled either by running the algorithm on a real-time operating system or by calling it directly via timed interrupts. Besides the real-time execution, the necessary inputs (phase currents i_a, i_b and rotor position φ_m) must be provided by the target environment as well.

Table 1 Parameters of the applied induction machine

Description	Symbol	Value	Dimension
Rated power		0.8	kW
Rated speed		3950	1/min
Rated frequency		140	Hz
Stator resistance	R_s	4.7	Ω
Rotor resistance	R_r	5.2	Ω
Stator inductance	L_s	0.1788	H
Rotor inductance	L_r	0.1790	H
Mutual inductance	L_m	0.1690	H
Number of pole pairs	p	2	
Rotor inertia	J	0.0008658	kg m^2
Viscous friction coefficient	D	0.005028	Nm s/rad
Static friction	T_0	0.02276	Nm

4 Implementation example

By recalling the introductory section, one may remember that MBD enables the early validation of control algorithms by simulations and RCP. This capability is especially useful for such closed-loop control algorithms like the DRFOC (note the closed current control loops in Fig. 1), as the behaviour of closed-loop controls is hardly predictable otherwise. This section presents an implementation example for such a system by using MATLAB/Simulink and utilising the methods and principles described in the previous section. The first subsection presents how the executable simulation model, including the DRFOC algorithm, can be implemented. Afterwards, a code generation model for RCP target deployment is described. Finally, the simulation and experimental results are discussed and compared. In the experimental setup, a Lenze MCA 10I40-RS0B2-Z0C0-STBS00N-R0SU induction machine is applied with nominal parameters described in Table 1 (for more details see [48]). These parameter values of the machine are used in the forthcoming simulations as well.

4.1 Simulation model

By starting from the top level, the executable simulation model implemented in MATLAB/Simulink environment is seen in Fig. 7. The figure may be associated with the simulation model depicted in the top right of Fig. 2. Featured subsystems are the DRFOC control algorithm and the induction motor simulation model subsystems. Both of them are featured in the model as links pointing to the corresponding model layer library blocks in accordance with the previously described principles. It is also important to note that the DRFOC subsystem is designated with a thick boundary line. This designation indicates the *atomic subsystem* feature

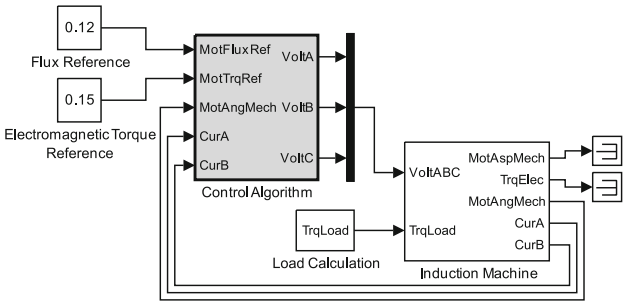

Fig. 7 Top level of simulation model implemented in Simulink

of the MATLAB/Simulink environment. This feature allows control of the execution and code generation properties of blocks, like execution order and sampling rate, for example. Since the DRFOC block is distinguished in this way, it is also possible to control the simulation execution properties of subsystems separately, e.g. the sampling rate of the DRFOC subsystem may be different compared to the rest of the simulation model.

Implementation of the DRFOC algorithm is shown in Fig. 8. Subsystems of the MATLAB/Simulink implementation of the control algorithm are in correspondence with the functional components depicted in Figs. 1 and 2. Subsystems are featured as links pointing to the corresponding component layer library blocks. Note that the subsystems are designated as atomic subsystems. By using this feature, it is possible to control the automated code generation process and ensure the generation of each subsystem into separate functions in the target language, thus enforcing the functional decomposition principle in the generated code as well.

4.2 Code generation model for RCP target deployment

By using the proposed RCP approach, the DRFOC algorithm may be investigated experimentally as well.

As shown in Fig. 2, a specific code generation model is necessary for RCP. Similarly to the simulation model, the code generation model for RCP is also implemented in MATLAB/Simulink environment, as shown in Fig. 9. This model consists of subsystems for the Ethernet communication of the bypass interface and the DRFOC control algorithm, in accordance with the structure depicted in Fig. 5. Just as in the case

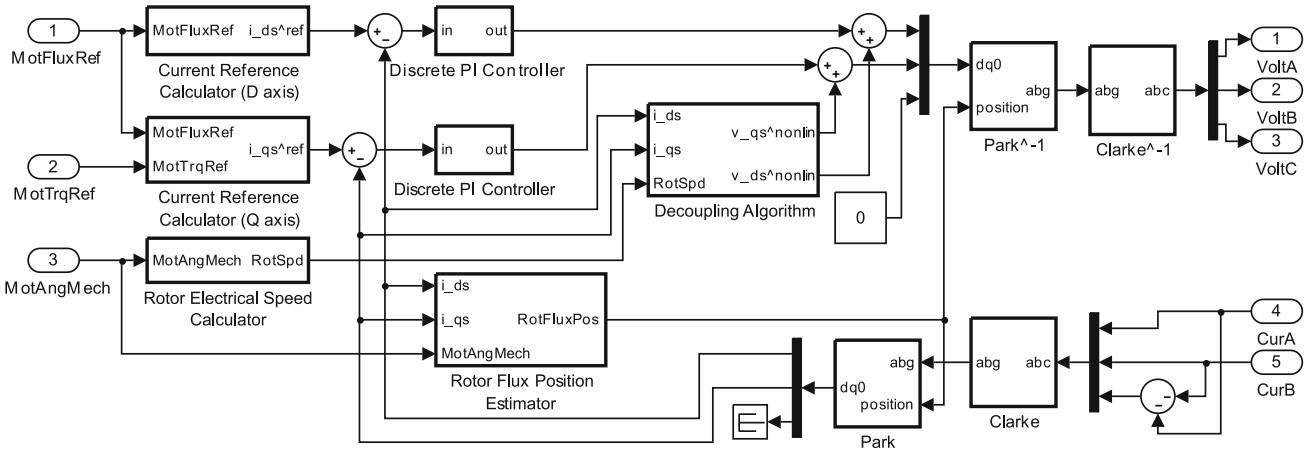

Fig. 8 DRFOC algorithm implementation in Simulink

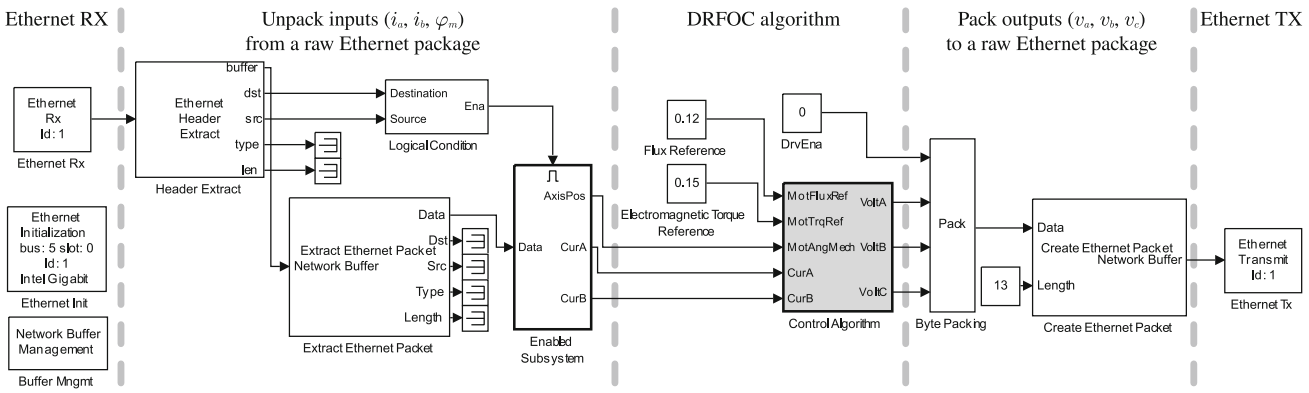

Fig. 9 Top level of code generation model for RCP implemented in Simulink

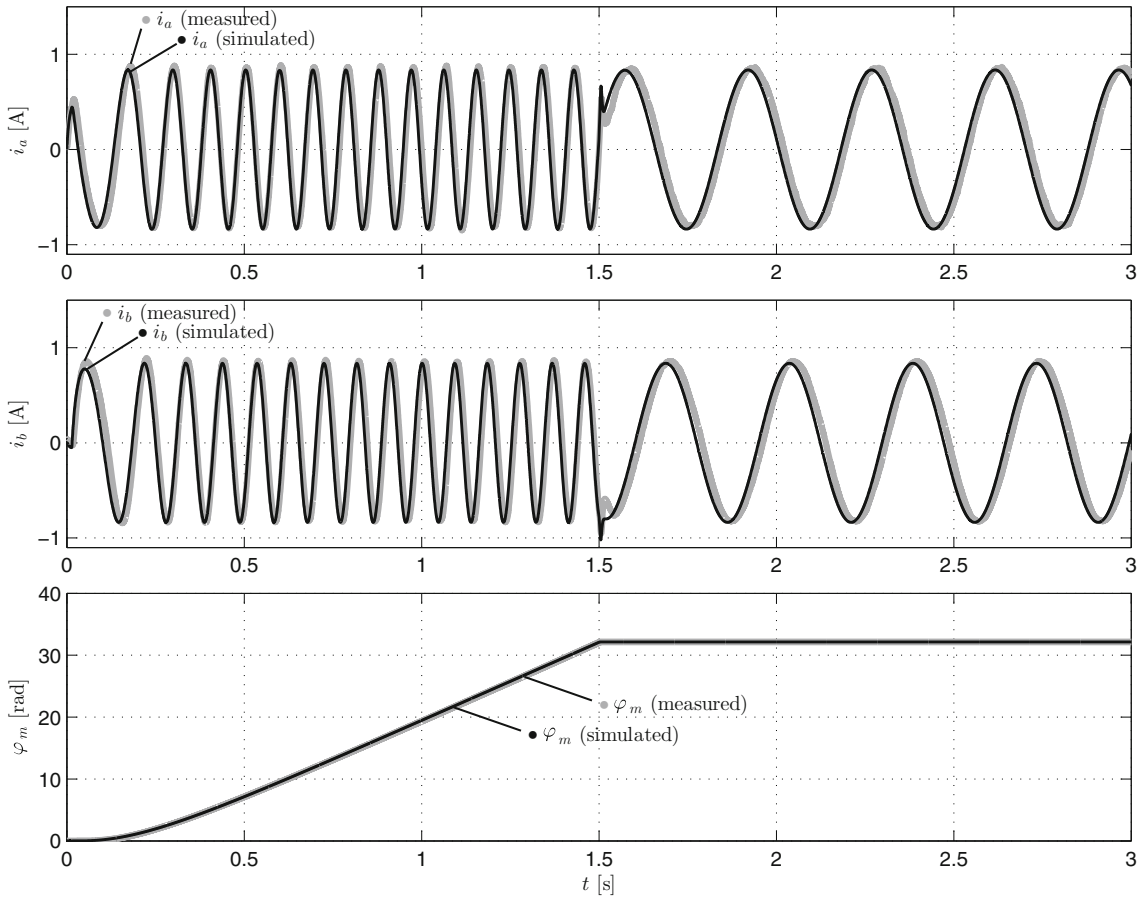

Fig. 10 Comparison of simulated and measured variables

of the simulation model, the DRFOC block is utilised as a link pointing to the control algorithm block in the model layer as depicted in Fig. 2; therefore, it contains the same algorithm that is shown in Fig. 8. During the deployment procedure, a self-contained code may be generated from the model shown in Fig. 9, which may then be compiled and deployed to the SRTT.

4.3 Simulation and experimental validation

In order to demonstrate the proposed development approach through a working example, the DRFOC algorithm is investigated by simulation, and the results are validated experimentally as well. The validation is based on the comparison of the simulated signals to their measured counterparts. In addition, some derived quantities are also available in the RCP environment. The derived signals are not measured or not measurable directly, but might be calculated online and also compared to their simulated doubles.

According to the principles of the MBD methodology, the simulation model and the RCP enable the investigation of the overall system performance, both within the simulation and experimental environments. Although various types of investigations could be performed in this framework, the usual method is to check the step responses of the simulated and the actual systems. In this case, the step responses represent the starting transient behaviour of the induction machine as well. After the machine reaches the steady-state rotation, a sudden load is also applied by engaging an electromechanical brake. For comparison, the same scenarios are investigated both experimentally and in the simulation environment. According to this consideration, the torque and flux references are set to $T_{\mathrm{e}}^{\mathrm{ref}} = 0.15\,\mathrm{Nm}$ and $\psi^{\mathrm{ref}} = 0.12\,\mathrm{Wb}$ at $t = 0\,\mathrm{s}$, respectively, while controller parameters are proportional gains $K_{\mathrm{P}} = 2.35$ and integral gains $K_{\mathrm{I}} = 287.01$. For the simulation, the rated parameters of the induction machine are set according to the values of Table 1.

The results of the simulation and the experiment are seen in Figs. 10 and 11. Figure 10 shows the directly measured signals compared to their simulated counterparts. As can be seen in this figure, the measured stator currents i_a and i_b, and measured rotor position φ_{m} are close to the simulated signals. The minor differences between the measured and simulated phase currents result from the measurement inaccuracies. In Fig. 11, the most important derived signals are compared to their simulated pairs, showing similar conformity.

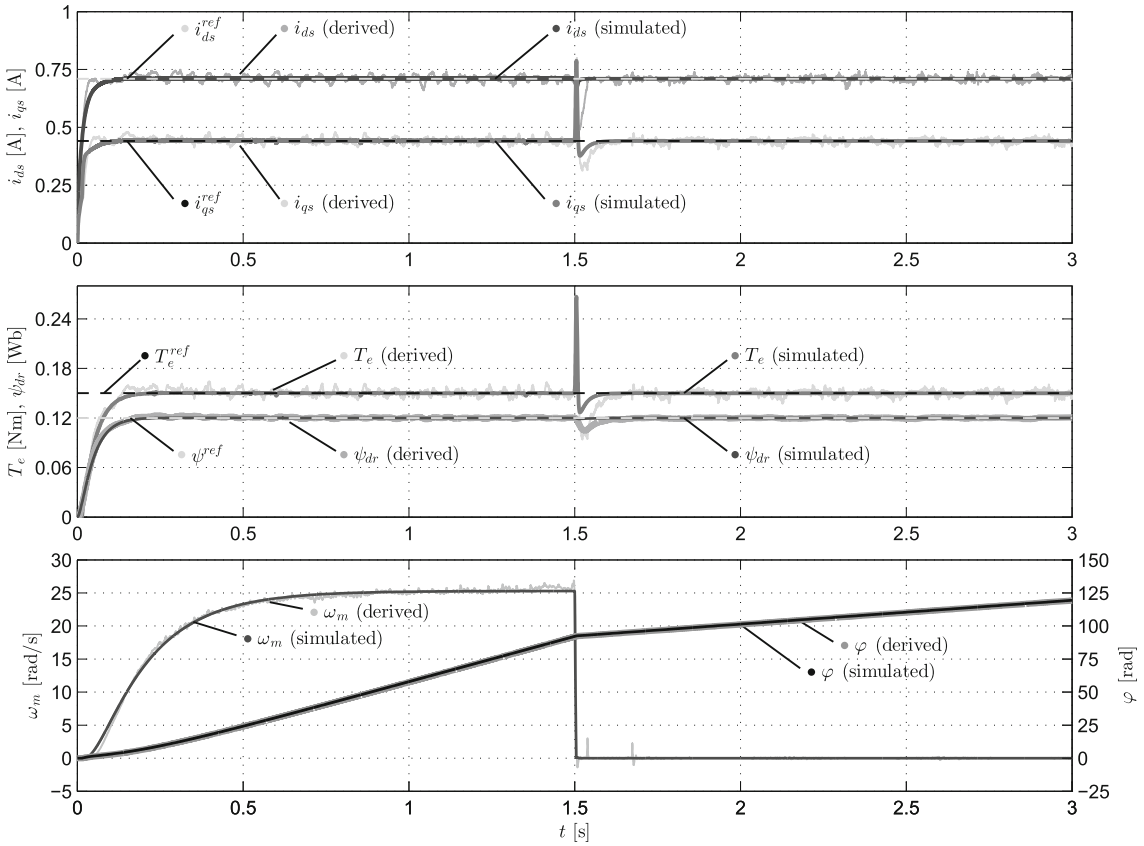

Fig. 11 Comparison of simulated and derived variables

The flexibility of the simulation model and the transparency of the RCP system allow the monitoring of any internal variable as well. For example, the simulation model may be extended by a virtual power meters by which the electric power flow or certain losses of the machine may be monitored in the simulations, or in real-time through the virtual instrumentation, or by the acquired and post-processed measurement data. For example, we may monitor the stator Ohmic loss P_R being the most significant loss component, see Fig. 12. However, there are limitations, as well. Obviously, the model captures only a limited set of features; therefore, properties beyond the capabilities of the lumped parameter model cannot be investigated. Typically, phenomena associated closely with the material properties of the machine, e.g. stator core losses, are out of scope.

Basically, the DRFOC algorithm provides satisfactory performance as it follows the references, and both current and speed ramp-up transients are quick enough. Also, we may observe that the controller remains operable and keeps generating the rotating field even when the external load suddenly stops the rotation. Although these results do not hold significance here on their own, they show that the presented framework is suitable for setting up a properly working machine model and control algorithm. Furthermore,

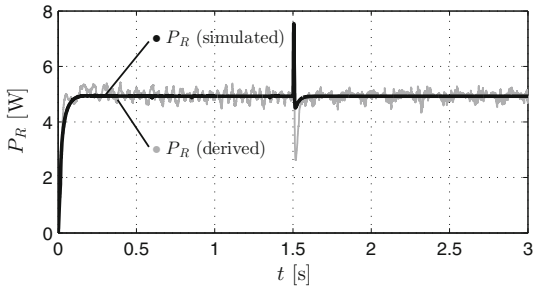

Fig. 12 Stator Ohmic loss

the results show that the run of the measured and derived variables are in correspondence with the simulation results; therefore it may be concluded that the presented RCP configuration and the corresponding experimental setup enable the validation of the models and control algorithms.

5 Conclusion and future work

Application of MBD methods may help to cope with increasing challenges in the field of embedded system development. This consideration applies to control systems of electric

machines as well. In this paper, an application of the MBD methodology is presented for control algorithm development of induction machines. Within the confines of the work, a modular, reusable three-layer model architecture and an RCP approach for experimental validation are introduced, and it is also shown, how the model architecture can support the software development procedure. By using these findings, an implementation example is also presented, including simulation and experimental validation, where the latter shows how the discussed methods can lead to an actually working system.

Besides the discussed advantages, there are limitations as well, especially in the applied machine model, which is a simple lumped parameter type one, and therefore, does not capture the properties related closely to the material properties or structure of the machine. Thus, a possible future development is the extension of the machine model by such properties, e.g. by current-dependent inductances or temperature-dependent resistances.

The authors also plan to use the presented development approach in various future research projects. For example, an application may be found already in [49], where Kalman filters have been added to the control component library as new components and then utilised in the model layer for speed sensorless state estimation of the induction machine.

Acknowledgements Open access funding provided by Széchenyi István University (SZE).

References

1. Lennon T, Mass N (2008) Model-based design for mechatronic systems. Electron World 114(1865):23–26
2. Ong C (1998) Dynamic simulation of electric machinery using MATLAB/Simulink. Prentice Hall PTR, Upper Saddle River
3. Le-Huy H (2001) Modeling and simulation of electrical drives using MATLAB/Simulink and power system blockset. In: Proceedings of IECON'01. 27th annual conference of the IEEE Industrial Electronics Society, Denver, pp 1603–1611. https://doi.org/10.1109/iecon.2001.975530
4. Costa A, Vilaragut M, Travieso-Torres JC, Duarte-Mermoud M, Muñoz J, Yznaga I (2012) MATLAB based simulation toolbox for the study and design of induction motor FOC speed drives. Comput Appl Eng Educ 20(2):295–312. https://doi.org/10.1002/cae.20396
5. Frederiksen A (2013) Model-based design of advanced motor control systems. Technical report Analog Devices, Inc
6. O'Sullivan D, Sorensen J, Frederiksen A (2014) Model based design tools in closed loop motor control. In: Proceedings of PCIM Europe 2014; international exhibition and conference for power electronics, intelligent motion, renewable energy and energy management, Nuremberg, pp 1643–1651
7. O'Sullivan D, Sorensen J, Murray A (2015) Model-based design streamlines embedded motor control system development. Technical report. Analog Devices, Inc
8. Liu XF, Cui SM, Gao WF, Li JT, Xu SM (2014) Software development of on-board power electronics equipment using model-based design methodology. Appl Mech Mater 494–495:1524–1528. https://doi.org/10.4028/www.scientific.net/AMM.494-495.1524
9. Rau A (2001) On model-based development: decomposition and data abstraction in Simulink. Softwaretechnik-Trends 21(3):22–27
10. Leitner A, Ebner W, Kreiner C (2013) Mechanisms to handle structural variability in MATLAB/Simulink models. In: Proceedings of 13th international conference on software reuse (ICSR), Pisa, pp 17–31. https://doi.org/10.1007/978-3-642-38977-1_2
11. Blaschke F (1972) The principle of field orientation as applied to the new transvector closed-loop control system for rotating field machines. Siemens Rev 34(3):217–220
12. Takahashi I, Noguchi T (1986) A new quick-response and high-efficiency control strategy of an induction motor. IEEE Trans Ind Appl 22(5):820–827. https://doi.org/10.1109/tia.1986.4504799
13. Depenbrock M (1985) Direkte selbstregelung (DSR) fuer hoch dynamische drehfeldantriebe mit stromrivhterschaltung. Elektrotech Z Arch 7(7):211–218
14. Casadei D, Profumo F, Serra G, Tani A (2002) FOC and DTC: two viable schemes for induction motors torque control. IEEE Trans Power Electron 17(5):779–787. https://doi.org/10.1109/tpel.2002.802183
15. Orłowska-Kowalska T, Dybkowski M (2010) Stator-current-based MRAS estimator for a wide range speed-sensorless induction-motor drive. IEEE Trans Ind Electron 57(4):1296–1308. https://doi.org/10.1109/tie.2009.2031134
16. Zbede YB, Gadoue SM, Atkinson DJ (2016) Model predictive MRAS estimator for sensorless induction motor drives. IEEE Trans Ind Electron 63(6):3511–3521. https://doi.org/10.1109/tie.2016.2521721
17. Yang S, Li X, Xie Z, Zhang X (2018) A combined speed estimation scheme for indirect vector-controlled induction motors. Electr Eng 100(4):2243–2252. https://doi.org/10.1007/s00202-018-0699-3
18. Orłowska-Kowalska T, Dybkowski M (2011) Performance analysis of the sensorless induction motor drive system under faulted conditions. In: Proceedings of 2011 IEEE EUROCON—international conference on computer as a tool, Lisbon, pp 801–804. https://doi.org/10.1109/eurocon.2011.5929414
19. Bojoi R, Guglielmi P, Pellegrino GM (2008) Sensorless direct field-oriented control of three-phase induction motor drives for low-cost applications. IEEE Trans Ind Appl 44(2):475–481. https://doi.org/10.1109/tia.2008.916735
20. Wu B, Narimani M (2017) High-power converters and AC drives. Wiley-IEEE Press, Piscataway, pp 321–352. https://doi.org/10.1002/9781119156079.ch14
21. Hanselmann H (1996) Automotive control: from concept to experiment to product. In: Proceedings of joint conference on control applications intelligent control and computer aided control system design, Dearborn, pp 129–134. https://doi.org/10.1109/cacsd.1996.555244
22. Freund U, Kraft D (2004) Model-based design & rapid prototyping. IFAC Proc 37(22):35–40. https://doi.org/10.1016/s1474-6670(17)30318-x

23. Schäuffele J, Zurawka T (2010) Automotive software engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen. Vieweg+Teubner, Wiesbaden, pp 231–242
24. Isermann R (2007) Mechatronic systems: fundamentals. Springer, London, pp 569–573
25. Yacoub Y, Chevalier A (2001) Rapid prototyping with the controller area network (CAN). In: SAE Technical Paper, https://doi.org/10.4271/2001-01-1224, Article ID 2001-01-1224
26. Abourida S, Dufour C, Bélanger J (2005) Real-time and hardware-in-the-loop simulation of electric drives and power electronics: process, problems and solutions. In: Proceedings of the international power electronics conference (IPEC-Niigata 2005), Niigata, pp 1908–1913
27. Rubaai A, Castro-Sitiriche MJ, Ofoli AR (2008) Design and implementation of parallel fuzzy PID controller for high-performance brushless motor drives: an integrated environment for rapid control prototyping. IEEE Trans Ind Appl 44(4):1090–1098. https://doi.org/10.1109/tia.2008.926059
28. Carpiuc S, Villegas C (2018) Real-time position control in permanent magnet synchronous machine drives. In: Proceedings of 2018 20th European conference on power electronics and applications (EPE'18 ECCE Europe), Riga, pp 1–8, Article number 8515606
29. Potnuru D, Alice Mary K, Saibabu C (2018) Design and implementation methodology for rapid control prototyping of closed loop speed control for BLDC motor. J Electr Syst Inf Technol 5(1):99–111. https://doi.org/10.1016/j.jesit.2016.12.005
30. Aiello G, Cacciato M, Scarcella G, Scelba G (2017) Failure analysis of AC motor drives via FPGA-based hardware-in-the-loop simulations. Electr Eng 99(4):1337–1347. https://doi.org/10.1007/s00202-017-0630-3
31. Bednarz SA, Dybkowski M (2018) On-line detection of the rotor faults in the induction motor drive using parameter estimator. In: Proceedings of 2018 international symposium on electrical machines (SME), Andrychów, pp 1–5. https://doi.org/10.1109/isem.2018.8443020
32. Kyslan K, Kušnír E, Fedák V, Lacko M, Ďurovský F (2014) Dynamic emulation of mechanical loads with backlash based on rapid control prototyping. In: Proceedings of 2014 16th international power electronics and motion control conference and exposition, Antalya, pp 1209–1215. https://doi.org/10.1109/epepemc.2014.6980676
33. Šlapák V, Kyslan K, Lacko M, Fedák V, Ďurovský F (2016) Finite control set model predictive speed control of a DC motor. Math Probl Eng 2016: Article ID 9571972. https://doi.org/10.1155/2016/9571972
34. French CD, Finch JW, Acarnley PP (1998) Rapid prototyping of a real time DSP based motor drive controller using Simulink. In: Proceedings of international conference on simulation '98, York, pp 284–291. https://doi.org/10.1049/cp:19980652
35. Kang MH, Park YC (2006) A real-time control platform for rapid prototyping of induction motor vector control. Electr Eng 88(6):473–483. https://doi.org/10.1007/s00202-005-0307-1
36. Tursini M, Leonardo LD, Olivieri C, Loggia ED (2013) Rapid control prototyping of IPM drives by real time simulation. In: Proceedings of 2013 8th EUROSIM congress on modelling and simulation, Cardiff, pp 364–371. https://doi.org/10.1109/eurosim.2013.116
37. Krizan J, Ertl L, Bradac M, Jasansky M, Andreev A (2014) Automatic code generation from MATLAB/Simulink for critical applications. In: Proceedings of 2014 IEEE 27th Canadian conference on electrical and computer engineering (CCECE), Toronto, pp 836–841. https://doi.org/10.1109/ccece.2014.6901058
38. Morkoç C, Önal Y, Kesler M (2014) DSP based embedded code generation for PMSM using sliding mode controller. In: Proceedings of 2014 16th international power electronics and motion control conference and exposition (PEMC), Antalya, pp 472–476. https://doi.org/10.1109/epepemc.2014.6980537
39. Horváth K, Kuslits M (2016) Model-based development of induction motor control algorithms with modular architecture. In: Proceedings of 2016 IEEE international power electronics and motion control conference (PEMC), Varna, pp 133–138. https://doi.org/10.1109/epepemc.2016.7751987
40. Filizadeh S (2013) Electric Machines and drives: principles, control, modeling, and simulation. Taylor & Francis, Boca Raton
41. de Pelegrin J, Torrico C, Carati E (2016) A model-based suboptimal control to improve induction motor efficiency. J Control Autom Electr Syst 27(1):69–81. https://doi.org/10.1007/s40313-015-0216-0
42. MATLAB R2014a Documentation: Large-Scale Modeling. The MathWorks, Inc
43. HP Compaq 8200 Elite Microtower PC—Specifications. HP Development Company, L.P. https://support.hp.com/us-en/product/hp-compaq-8200-elite-microtower-pc/5037940/model/5037947/document/c02779504. Downloaded at 25 Oct 2018
44. User's Guide, TMS570LC43x Hercules Development Kit (HDK). Texas Instruments, Inc., SPNU597
45. DRV830x Rev D. Hardware Quick Start Guide. Texas Instruments, Inc., Version 1.0.5
46. Evaluation Board for 10-Bit to 16-Bit R/D Converter with Reference Oscillator. Analog Devices, Inc., EVAL-AD2S1210, Rev. 0
47. MATLAB R2014a Documentation: Model-Based Ethernet Communications. The MathWorks, Inc
48. L-force catalogue. Lenze SE, V08-en_GB-06/2016. http://www.lenze.com/fileadmin/lenze/documents/en/catalogue/CAT_MT_MC_13513046_en_GB.pdf. Downloaded at 25 Oct 2018
49. Horváth K, Kuslits M (2018) Dynamic performance of estimator-based speed sensorless control of induction machines using extended and unscented Kalman filters. Power Electron Drives 3(1):129–144. https://doi.org/10.2478/pead-2018-0003