CHALMERS
UNIVERSITY OF TECHNOLOGY

Field-Oriented Control of PMSMs within CPU-Constrained Applications

Firmware implementation of an FOC-based control strategy, focusing on robust CPU interrupt handling

Master thesis in Systems, Control and Mechatronics

LINUS JOHANSSON
ANDERS LOGREN

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master thesis 2024

Field-Oriented Control of PMSMs within CPU-Constrained Applications

Firmware implementation of an FOC-based control strategy, focusing on robust CPU interrupt handling

LINUS JOHANSSON
ANDERS LOGREN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Field-Oriented Control of PMSMs within CPU-Constrained Applications
Firmware implementation of an FOC-based control strategy, focusing on robust CPU interrupt handling
LINUS JOHANSSON
ANDERS LOGREN

Supervisor: Jacob Andersson, Mikael Bengtsson
Examiner: Prof. Jonas Fredriksson, Department of Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Abstract

Efficient control of Permanent Magnet Synchronous Motors (PMSMs) is central for a wide range of applications, where Field-Oriented Control (FOC) is a commonly used control method. In embedded systems where computational resources are constrained, implementing reliable and efficient FOC can be challenging. Particularly when higher-priority tasks occupy the CPU, thereby interfering with the motor control. Ensuring smooth rotation and constant torque production is crucial regardless if the CPU is available for computing new motor control signals or not.

This thesis proposes and explores an FOC-based strategy to ensure efficient control of PMSMs within CPU-constrained applications, focusing on robust interrupt handling to maintain motor performance during various operating conditions. The strategy involves pre-computing future control signals and storing them in memory so that, if closed-loop control is interrupted, the CPU peripherals can switch to open-loop control and output the pre-computed signals. The implemented strategy proved successful at maintaining the angular velocity during interrupts using open-loop control, regardless of when they occur and their longevity. During interrupts, the produced torque fluctuates and the power consumption increases, but stability is maintained as long as the operating conditions remain unchanged.

Keywords: Permanent Magnet Synchronous Motor (PMSM), Field-Oriented Control (FOC), Interrupt handling

Acknowledgements

We would like to express our gratitude to our supervisors, Jacob Andersson and Mikael Bengtsson, for their guidance, support, and encouragement throughout our research. Their help and expertise regarding the utilized hardware were very valuable for our work.

We are also sincerely thankful to our examiner, Prof. Jonas Fredriksson, for his constructive comments and for keeping us on track. His feedback has significantly improved the quality and clarity of this master thesis.

Linus Johansson & Anders Logren, Gothenburg, June 2024

Contents

List of Figures

List of Tables

1
Introduction

Efficient control of Permanent Magnet Synchronous Motors (PMSMs) is central for a wide range of applications, where Field-Oriented Control (FOC) is a commonly used control method. In embedded systems where computational resources are constrained, implementing reliable and efficient FOC can be challenging. Particularly when higher-priority tasks occupy the CPU, thereby interfering with the motor control. Ensuring smooth rotation and constant torque production is crucial regardless if the CPU is available for computing new motor control signals or not. This thesis proposes and explores an FOC-based strategy to ensure efficient control of PMSMs within CPU-constrained applications, focusing on robust interrupt handling to maintain motor performance during various operating conditions.

1.1 Background

Brushed DC motors rotate when a voltage is applied to its two poles, utilizing a brush to transfer power to the motor's coils. This power transfer results in the creation of magnetic fields that interact with the motor's armature, inducing rotational motion. Control of brushed DC motors is rather simple, but the brushes wear out over time and the efficiency is lower compared to a PMSM. In PMSMs, the mechanical commutation function is replaced by electronic control of the stator coils to make the motor rotate, i.e. precise control of the stator coils' magnetic fields. As long as the coils are excited with precise timing and correct permutation, the motor will move smoothly and with high motor efficiency. FOC is a common and efficient PMSM control technique and has been proven to work efficiently in systems where the control is executed on a dedicated CPU. In various embedded systems, however, it is of interest to utilize a single processor for numerous regularly occurring tasks. This means that the execution of PMSM control could potentially be interrupted at any time and for varying periods, resulting in undesirable and possibly dangerous behavior.

1.2 Problem definition

The primary focus of this research is to develop and implement a robust interrupt-handling mechanism to complement the FOC algorithm for PMSMs. The goal is to ensure that the motor continues to rotate at the expected velocity with the same torque when the CPU is intermittently occupied by higher-priority tasks. This involves smart configuration of the CPU's peripherals and preparing specific buffers

in RAM with appropriate control signals for when the CPU cannot actively update them.

Addressing this problem is important for several reasons. First, maintaining consistent motor performance despite CPU interruptions is essential for applications requiring reliability on low-cost systems, where maximizing the utilization of the processor is desired. Second, the ability to handle interrupts effectively can significantly enhance the robustness and flexibility of embedded systems, allowing them to perform multiple tasks concurrently without compromising the performance of critical functions like motor control too much.

This research fills a gap in current knowledge by providing a solution to the challenge of maintaining motor control in the presence of CPU interrupts. The performance of the implemented solution will be evaluated in terms of how well the motor is able to track the velocity reference and the variations in produced torque in varying operating conditions. The operating conditions include acceleration, deceleration, and constant velocity, with interrupts of varying lengths and frequencies.

1.3 Scope and delimitations

The scope of this thesis can be split into two main parts, the first being an FOC implementation on the hardware described in chapter 3. The second part is an evaluation of the effects of CPU interrupts, and the development and implementation of a strategy to mitigate the effects of said interrupts. The implementation will be a modification of the control algorithm from the first part, such that the output to the motor will be that of the FOC algorithm when the CPU is available and that of the custom-made strategy when it is not.

The research is limited to the following delimitations:

- **Control Method**: This thesis focuses exclusively on the implementation of FOC. No other control methods will be examined.
- **Sensor Usage**: Only sensored FOC will be considered; sensorless FOC is outside the scope of this research.
- **Cascaded Control**: A velocity controller will be implemented as an outer control loop for FOC. It is possible to also add position control, however this is not included in the thesis.
- **Controllers**: The implementation will utilize only Proportional-Integral (PI) controllers. Other types of controllers will not be explored.
- **Interrupt Handling Strategy**: Only one strategy for handling CPU interrupts will be developed and evaluated.
- **Constant Velocity**: The focus will be on maintaining a constant velocity. The strategy will not address handling transients differently.
- **Hardware**: No new electronic circuits will be designed or produced. The project will rely solely on commercially available hardware.

2

Theory

This chapter explains how PMSMs are constructed and how they can be controlled using six-step commutation and FOC. The mathematical model of a three-phase PMSM is presented along with the transformed mathematical model in the rotating *dq*- reference frame in section 2.1, and the control techniques are presented in section 2.3-2.4. The theory behind the reference frames and transforms used for the transformation of the mathematical model, which is utilized in FOC, is presented in section 2.2. Information regarding the phase currents, rotor angle, and angular velocity is crucial for FOC, and methods of acquiring this information are explained in detail in section 2.5-2.6. In order to apply the appropriate phase voltages and transform DC power to AC power, pulse-width modulation (PWM) and a three-phase inverter are utilized as explained in section 2.7-2.8.

2.1 Permanent Magnet Synchronous Motor

PMSMs are a type of brushless DC motor consisting of a stator and a rotor as illustrated in figure 2.1. The rotor consists of an axle with permanent magnets, and the stator consists of the outer frame and a core with copper windings. PMSMs typically have a three-phase winding, which is referred to in this thesis as phases A, B, and C. By supplying current to the phase windings, a stator magnetic field is created. The principle of operation of PMSMs is based on the interaction between the stator magnetic field and the rotor magnetic field. According to Ampère's circuital law, this interaction creates torque, forcing the rotor to rotate in the desired manner with proper control of the stator magnetic field.

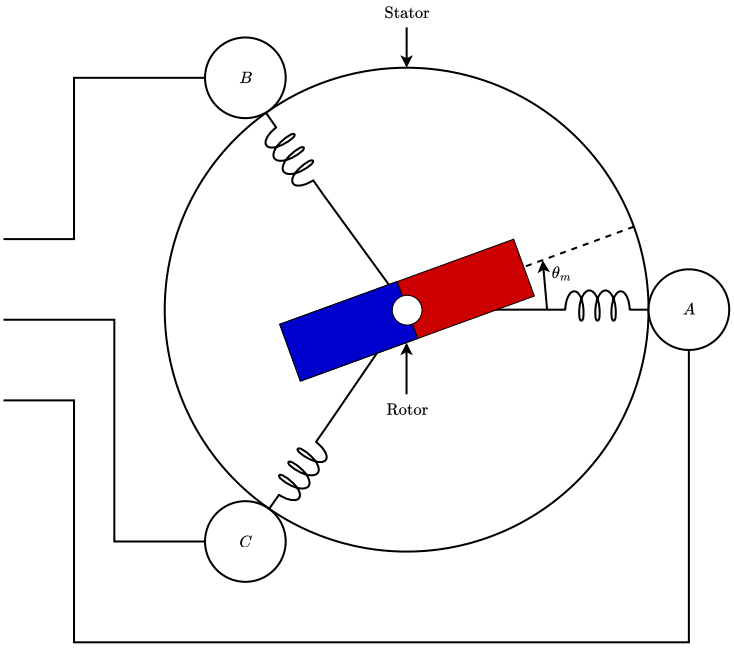


Figure 2.1: Basic structure of a PMSM

Figure 2.2 shows the physics of a rotating motor. The relation between the electromagnetic torque, load torque, inertia, angular velocity, and friction of the motor is given by

$$T_e = T_L + J\frac{d}{dt}\omega_m + k_f\omega_m \tag{2.1}$$

where T_e is the electromagnetic torque, T_L is the load torque, J is the inertia, ω_m is the mechanical angular velocity, and k_f is the friction constant.

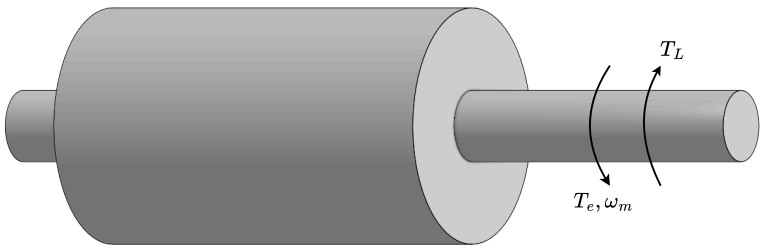


Figure 2.2: Physics of a rotating motor

The mechanical angle of the rotor, θ_m, is defined as positive in the counterclockwise direction, as seen in figure 2.1. The motor in the figure has a single pole pair, but it is common for PMSMs to have more pole pairs depending on the application since the number of poles affects the speed and torque characteristics. The electrical angle, θ_e, and electrical angular velocity, ω_e, of the rotor are given by

$$\theta_e = n_{pp}\theta_m \tag{2.2}$$
$$\omega_e = n_{pp}\omega_m \tag{2.3}$$

where n_{pp} is the number of pole pairs and ω_m is the mechanical angular velocity.

2.1.1 Mathematical model

A mathematical model of a PMSM is essential for understanding their dynamic behavior and for designing optimal control strategies. A PMSM can be seen as a balanced three-phase system [2], and the electrical dynamics in a stationary three-phase reference frame is described by

$$\boldsymbol{V}_{sabc} = R_s \boldsymbol{i}_{sabc} + \frac{d}{dt}\boldsymbol{\psi}_{sabc} \tag{2.4}$$

where $\boldsymbol{V}_{sabc}$ are the stator phase voltages, R_s is the stator winding resistance, $\boldsymbol{i}_{sabc}$ are the stator phase currents and $\boldsymbol{\psi}_{sabc}$ are the stator fluxes. The stator fluxes are, in turn, given by

$$\boldsymbol{\psi}_{sabc} = \boldsymbol{L}_{ss}\boldsymbol{i}_{sabc} + \psi_r \begin{bmatrix} cos(\theta_e) \\ cos(\theta_e - 2\pi/3) \\ cos(\theta_e + 2\pi/3) \end{bmatrix} \tag{2.5}$$

where $\boldsymbol{L}_{ss}$ is the stator inductance matrix and ψ_r is the flux generated by the rotor's permanent magnets.

Equation (2.4) can now be rewritten as

$$\boldsymbol{V}_{sabc} = R_s \boldsymbol{i}_{sabc} + \frac{d}{dt}\left(\boldsymbol{L}_{ss}\boldsymbol{i}_{sabc} + \psi_r \begin{bmatrix} cos(\theta_e) \\ cos(\theta_e - 2\pi/3) \\ cos(\theta_e + 2\pi/3) \end{bmatrix}\right) \tag{2.6}$$

2.1.2 Transformed mathematical model

The equations in 2.1.1 describe the motor dynamics in a stationary three-phase reference frame, the so-called *abc*-frame. It is, however, common practice to transform PMSM models into other reference frames, namely the $\alpha\beta$- and dq-frames, to simplify the analysis and control of PMSMs. The $\alpha\beta$-frame is a stationary two-phase reference frame and the dq-frame is a rotating two-phase reference frame. The two transforms combined separate the dynamics related to torque production and flux generation from each other, which makes the analysis and control of PMSMs more intuitive.

A transformed mathematical model of the PMSM is derived using the equations in section 2.1.1 as well as the Clarke and Park transforms presented in section 2.2 [2]. The motor's stator voltages in the dq-frame (V_d and V_q) can be formulated as

$$V_d = R_s i_d + L_d \frac{di_d}{dt} - \omega_e L_q i_q \tag{2.7}$$

$$V_q = R_s i_q + L_q \frac{di_q}{dt} + \omega_e (L_d i_d + \psi_R) \tag{2.8}$$

where L_d and L_q are the stator d- and q-axis inductances respectively and i_d and i_q are the d- and q-axis currents respectively [3]. For surface mount PMSMs, it is assumed that $L_d = L_q$.

The electrical torque produced by the motor is given by

$$T_e = \frac{3 n_{pp}}{2} (\psi_r i_q + (L_d - L_q) i_d i_q) \tag{2.9}$$

2.2 Reference Frames and Transforms

This section presents the Clarke and Park transforms, which are used to transform the three phase currents and voltages from the stationary *abc*-reference frame to the rotating *dq*-reference frame.

2.2.1 Clarke Transform

The Clarke transform converts stationary three-phase quantities into stationary two-phase orthogonal quantities, going from the *abc*-frame to the $\alpha\beta$-frame according to

$$\boldsymbol{f}_{\alpha\beta} = \frac{2}{3} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \boldsymbol{f}_{abc} \tag{2.10}$$

where $\boldsymbol{f}$ is a quantity such as current or voltage. There exist two versions of the Clarke transform, power invariant and amplitude invariant, and (2.10) presents the amplitude-invariant version.

Figure 2.3 illustrates the $\alpha\beta$-frame with respect to the *abc*-frame.

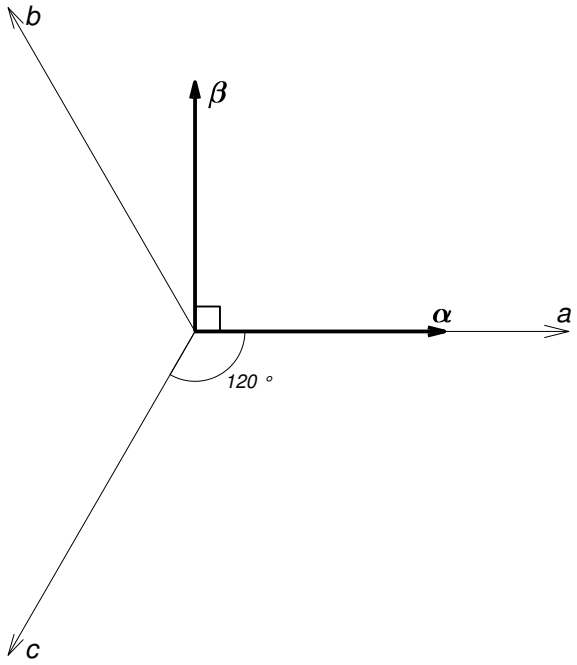


Figure 2.3: $\alpha\beta$-frame with respect to the *abc*-frame

The transform results in two components that represent the projections of the three-phase quantities onto an orthogonal two-axis frame, as illustrated in figure 2.3.

As stated in section 2.1.1, an assumption that can be made when analyzing three-phase motors is that they are balanced three-phase systems. The zero sequence component is always zero because the currents flowing through each phase are balanced and symmetrical. The currents in AC motors can therefore be represented as

$$i_a + i_b + i_c = 0 \tag{2.11}$$

Using the fact that only two currents are required in order to calculate the third, the Clarke transform can be further simplified as

$$\boldsymbol{f}_{\alpha\beta} = \begin{bmatrix} 1 & 0 & 0 \\ 1/\sqrt{3} & 2/\sqrt{3} & 0 \end{bmatrix} \boldsymbol{f}_{abc} \tag{2.12}$$

The inverse Clarke transform converts the $\alpha\beta$-quantities back into the original three-phase quantities as

$$\boldsymbol{f}_{abc} = \frac{3}{2} \begin{bmatrix} 2/3 & 0 \\ -1/3 & \sqrt{3}/3 \\ -1/3 & -\sqrt{3}/3 \end{bmatrix} \boldsymbol{f}_{\alpha\beta} \tag{2.13}$$

2.2.2 Park Transform

The transform from the $\alpha\beta$-frame to the rotating direct-quadrature reference frame (*dq*-frame) is known as the Park transform. The Park transform converts stationary and orthogonal two-phase quantities into rotating and orthogonal two-phase quantities, transforming the $\alpha\beta$-frame to the *dq*-frame, given by

$$\boldsymbol{f}_{dq} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \boldsymbol{f}_{\alpha\beta} \tag{2.14}$$

where $\boldsymbol{x}$ is a quantity such as current or voltage and θ is the angle between the α-axis and the d-axis. The inverse Park transform, used for the dq to $\alpha\beta$ transformation, is given by

$$\boldsymbol{f}_{\alpha\beta} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \boldsymbol{f}_{dq} \tag{2.15}$$

It should be noted that (2.14) and (2.15) holds for the case when the d-axis is aligned with the α-axis, and that the equations differ if the q-axis is aligned with the α-axis instead. A visualization of the Park transform is shown in figure 2.4.

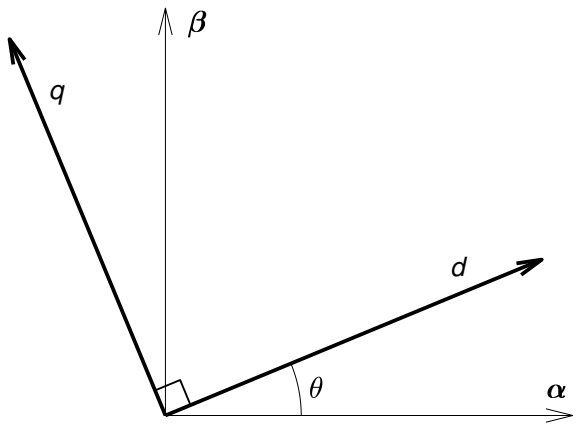


Figure 2.4: dq-frame with respect to the $\alpha\beta$-frame

Maximum torque is produced when the rotor and stator magnetic fields are orthogonal, i.e. when the stator magnetic field leads the rotor magnetic field by 90°. By aligning the d-axis with a rotor magnetic north pole, rotating the reference frame based on the rotor angle θ_m, and letting the q-axis lead the d-axis by 90°, maximum produced torque can be achieved by applying phase voltages that drive the d-current towards 0 while maximizing the magnitude of the q-current.

The results of applying both the Clarke and Park transforms are shown in figure 2.5, where it is shown how the phase currents are transformed to the stationary $\alpha\beta$-frame using the Clarke transform, and subsequently to the rotating dq-frame using the Park transform.

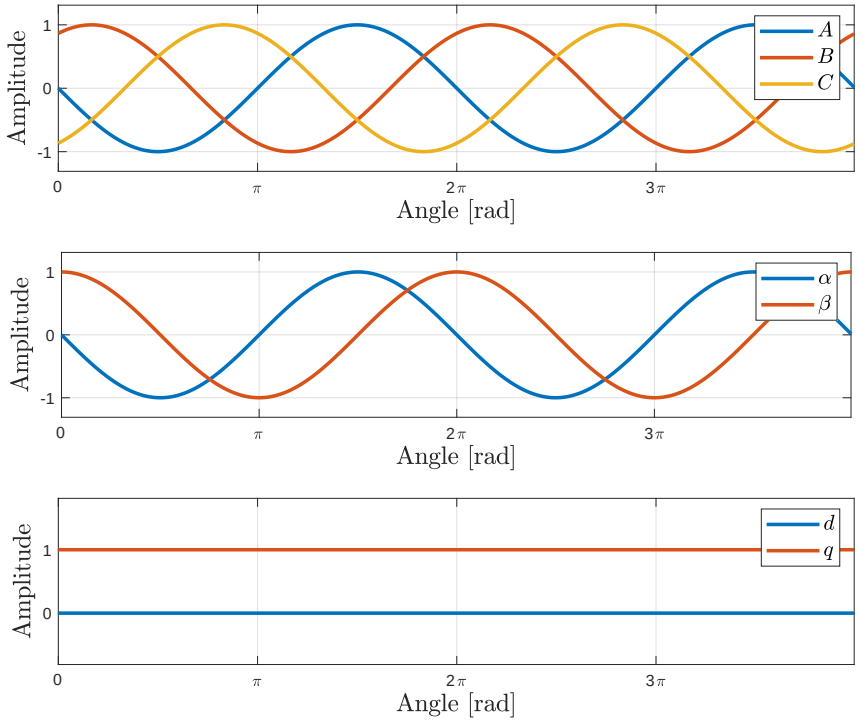


Figure 2.5: ABC-frame to $\alpha\beta$-frame to dq-frame

2.3 Six-step Commutation

A simple and inexpensive method of controlling the angular velocity of the motor is to use what is known as six-step commutation, or trapezoidal control [4]. The main idea behind the six-step commutation is to excite the coils in a predetermined order, based on the position of the rotor, to make the rotor rotate in the desired direction. A common approach to acquiring the position of the rotor is to use three Hall-effect sensors, placed with 120° offset from each other. Each Hall-effect sensor will produce a low or a high signal, detecting if it is near a south or north pole, i.e. the rotor poles. Assuming that the Hall-effect sensors are perfectly calibrated, there are six combinations of outputs from the sensors, since the output is binary. Using a PMSM with one pole pair, the output from the Hall-effect sensors for a full rotor rotation is shown in figure 2.6, assuming that the sensors are placed in alignment with the phases and that the north pole of the rotor is aligned with Hall-effect sensor A at angle 0°.

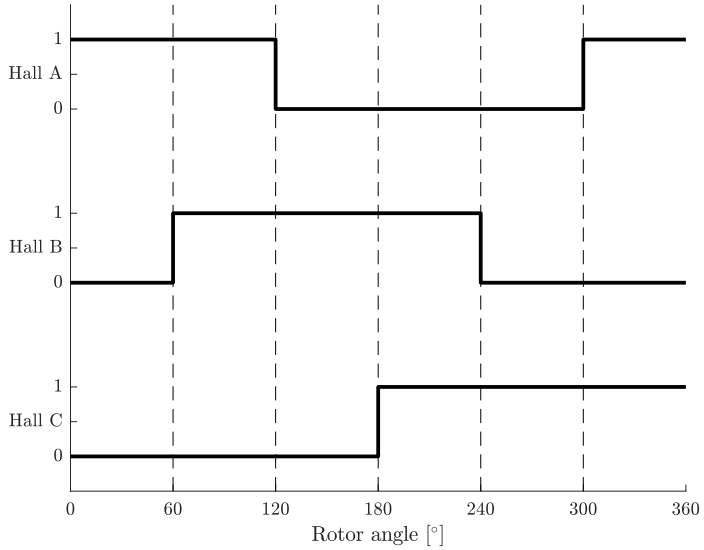


Figure 2.6: Hall-effect sensor output for a full rotor rotation for a PMSM with one pole pair

Using the output from the Hall-effect sensors, the stator windings are excited in a way that forces the rotor in the desired direction, where an example is shown in figure 2.7. In this example, the coils are excited based on the rotor position to generate counter-clockwise torque. By commutating the electric field based on the output from the Hall-effect sensors continuously and by controlling the current flowing through the coils, it is possible to control rotor speed and torque.

The main drawback of using Hall-effect sensors is that the angular estimation is not precise since the output from the Hall-effect sensors is identical in 60° intervals, as shown in figure 2.6. If the stator magnetic field is updated in 60° increments, the stator magnetic field will lead the rotor magnetic field by a range between 60-120°. As stated in section 2.2.2, maximum torque is generated when the stator magnetic field leads the rotor magnetic field by 90°. Since the angle between the two magnetic fields varies using six-step commutation, the produced torque will also vary as shown in figure 2.8. This is known as torque ripples and it also causes undesired variations in angular velocity and increased audible noise [5]. Thus, other control strategies are needed in applications requiring low noise levels and more constant torque production.

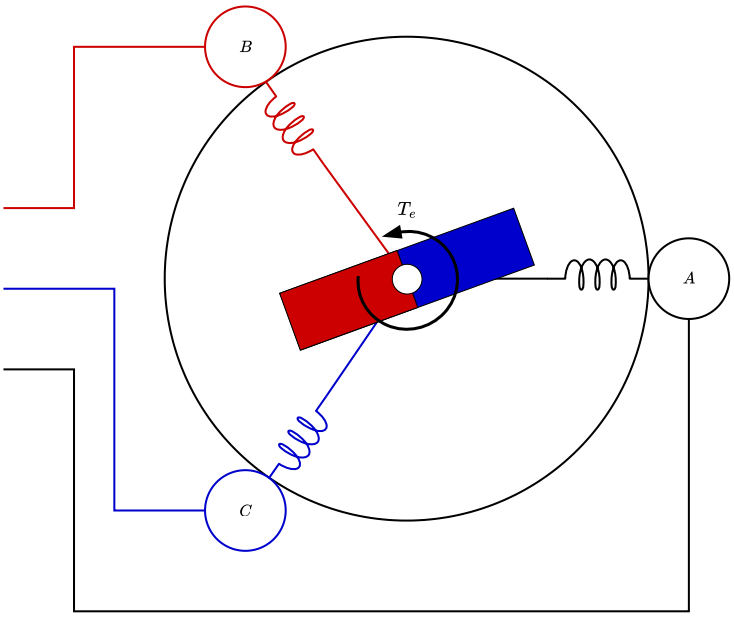


Figure 2.7: Example of how the coils can be excited to generate counter-clockwise torque

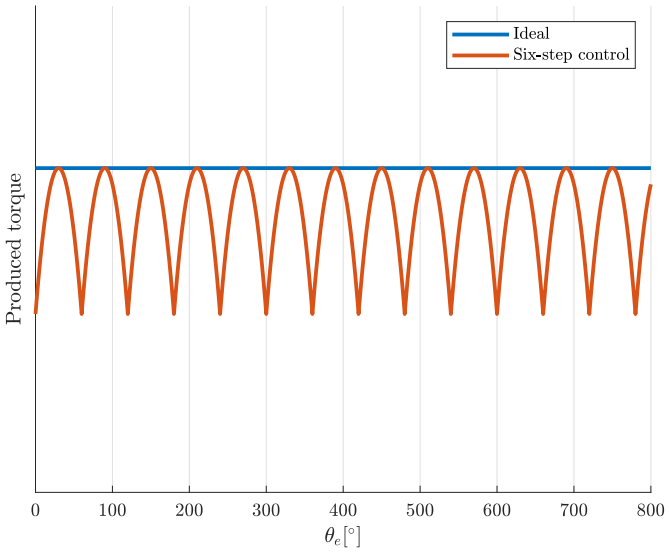


Figure 2.8: Produced torque using ideal control compared to six-step commutation

2.4 Field-Oriented Control

Field-oriented control is a method of rotating a PMSM by controlling the stator magnetic field such that it constantly leads the rotor magnetic field by 90° in the direction of desired rotation. FOC is thus similar to six-step commutation in terms of the working principle, with the key difference that the stator magnetic field is controlled with higher resolution using FOC compared to six-step commutation. The main purpose of FOC is to reduce the torque ripples shown in figure 2.8, thus increasing efficiency, decreasing audible noise, and rotating the rotor more smoothly.

The commutation waveforms, or in other words, the phase current input to the three phases for the two control strategies are shown in figure 2.9. One key difference, that can be seen in the figure, is that the coils of all three phases are energized simultaneously and with varying amplitudes when using sinusoidal commutation. This is what enables full control of the stator magnetic field, and using FOC it is possible to ensure that the stator magnetic field leads the rotor magnetic field by 90° at all times. Using FOC, the produced torque corresponds to what is shown in figure 2.8 in the ideal case.

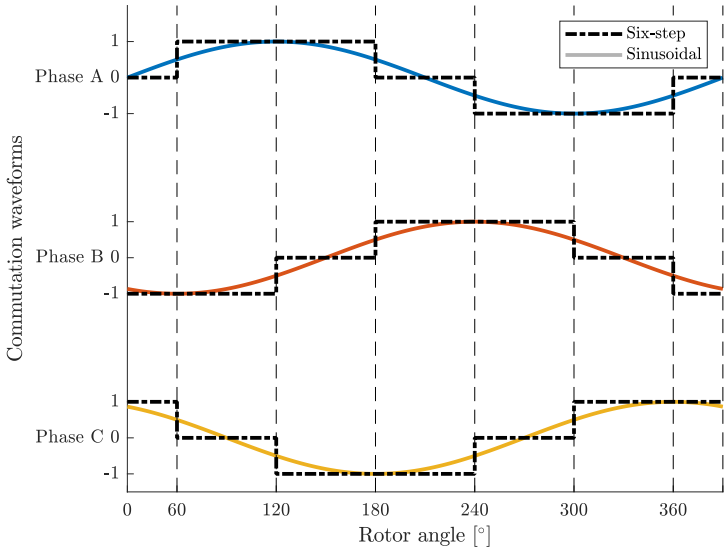


Figure 2.9: Commutation waveforms for six-step commutation and FOC

2.4.1 System overview

An overview of the control loop used in FOC is shown in figure 2.10. As can be seen, two references need to be set as input to the control loop, namely the d- and q-axis currents. They are compared to the measured values to compute the errors which is the input to the two separate current controllers. The output from the controllers is the d- and q-axis voltages that should be applied to minimize the errors. These are converted to the phase voltages V_A, V_B, and V_C, using the inverse Park and Clarke transforms, described in section 2.2.1 and 2.2.2 respectively. The computed

phase voltages are input to the three-phase inverter, which transforms DC power into three-phase AC power, further explained in section 2.8. Information regarding the phase currents and the position of the rotor is continuously acquired, which is utilized to compute both the d- and q-axis currents and the inverse Park transform. Figure 2.10 shows the working principle of current control in all FOC algorithms, even though there are a lot of variations to it in terms of how to estimate position, the waveforms of the applied phase voltages, how to estimate or measure the phase currents, etc.

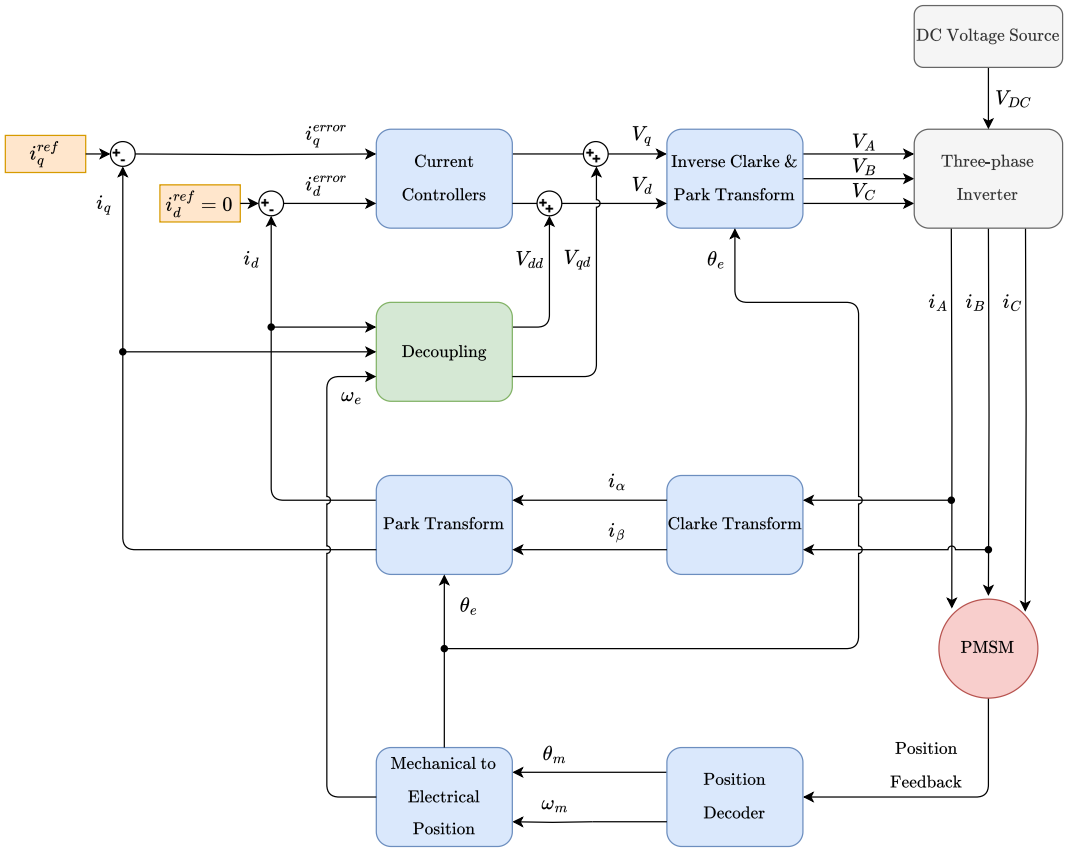


Figure 2.10: Overview of the current control loop used in FOC

Instead of setting the reference values for the d- and q-axis currents manually, a common approach is to implement cascaded control to control properties such as electromagnetic torque, rotor speed and position [6]. For the use case proposed in this thesis, it is mostly of interest to control the rotor velocity. Thus, an additional controller is needed to achieve this, and the system overview for this type of architecture is shown in figure 2.11. As can be seen in the figure, the input to the system is a velocity reference, ω_m^{ref}, i.e. the desired angular velocity of the rotor. The output from the velocity controller is the q-axis current reference value, which is the input to the current control loop shown in figure 2.10.

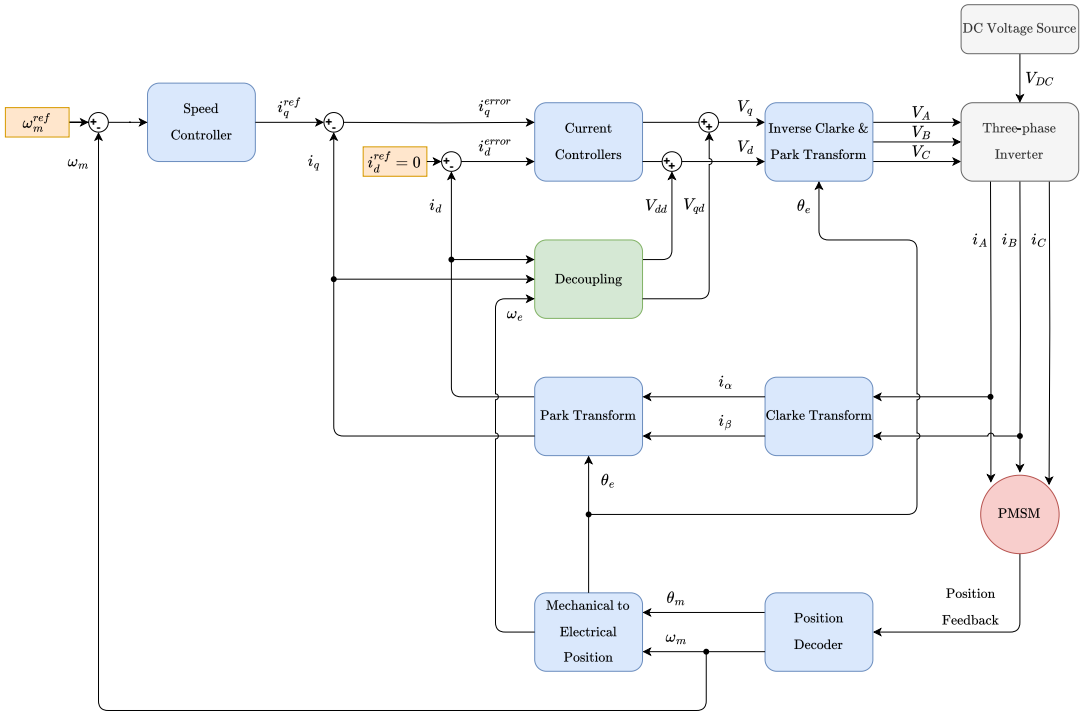


Figure 2.11: Overview of a cascaded FOC architecture

The main drawback of FOC compared to six-step commutation is that it requires a more complex control algorithm and often more expensive sensors than three Hall-effect sensors. This is because continuously updating the stator magnetic field in the previously described manner requires more involved computations, more frequent updates of the magnetic field, and more accurate information regarding the position of the rotor.

2.4.2 Stator voltage decoupling

As shown in (2.7) and (2.8), the stator voltage V_d depends on i_q and the stator voltage V_q depends on i_d, meaning that the stator voltages are cross-coupled. To achieve precise and independent control of the torque and flux, enabling efficient and smooth motor operation, the cross-coupling effects should be minimized [7]. This can be conducted by separating the components of V_d and V_q into the output from the current controllers and the decoupled terms [8] as:

$$V_d = V_d^{out} + V_{dd} \tag{2.16}$$
$$V_q = V_q^{out} + V_{qd} \tag{2.17}$$

where

$$V_{dd} = R_s i_d - \omega_e L_q i_q \tag{2.18}$$
$$V_{qd} = R_s i_q + \omega_e L_d i_d + \psi_r \omega_e \tag{2.19}$$

This is what is computed in the decoupling block shown in figure 2.10, and the results of using the decoupling methodology compared to not using decoupling is presented in appendix A.

2.5 Current measurement

One of the essential parts of FOC is measuring the phase currents in order to control them. Current sensing is typically conducted by connecting a shunt resistor in series with the current-conducting path and measuring the voltage drop across the resistor since it is proportional to the current flow according to Ohm's law. This approach is cheap and simple but also introduces a power loss, which increases with the square of the current. This may restrict the use of shunt resistors in high current applications [9].

The three standard types of current sensing are in-line current sensing, low-side current sensing, and high-side current sensing [10]. In-line current sensing is easy to use and precise, as the shunt resistors are placed in-line with the motor phases. This placement means that the current can be sampled at any time, regardless of the state of the three-phase inverter, and still gives accurate phase current readings [11]. The downside of this technique is hardware-related, as it requires high-precision bidirectional amplifiers with good PWM rejection to reduce the effects of noise. Thus, a more commonly used technique is low-side current sensing, where the shunt resistors are placed between the low-side transistors and ground, ensuring low voltages on the amplifiers at all times. High-side current sensing is similar to low-side current sensing, but with the shunt resistors placed between the DC voltage source and the high-side transistors instead, which means that this technique requires high-voltage supporting amplifiers. Furthermore, when using low-side current sensing, the current passing through a shunt resistor is phase current only if the corresponding low-side transistor is on. Similarly, when using high-side current sensing, the current passing through a shunt resistor is phase current only if the corresponding high-side transistor is on. The benefits and drawbacks of the three techniques are summarized in table 2.1, [12] and [13].

Table 2.1: Main benefits and drawbacks of current sensing techniques

Technique	Benefits	Drawbacks
Low-side	Inexpensive Simple circuitry	Requires synchronization between PWM and current measurement Can cause ground loop issues
High-side	No ground loop issues	Requires synchronization between PWM and current measurement Requires high-voltage supporting amplifiers
In-line	Current can be sampled at any time	Requires high-precision bidirectional amplifiers with adequate PWM rejection Expensive

All three phase currents must be known for the FOC algorithm to work, but that does not necessarily mean that all three currents have to be measured. Assuming that the system is balanced and that the currents are measured simultaneously, Kirchoff's 1st law can be used to calculate one of the phase currents by measuring the two others, given by

$$i_A + i_B + i_C = 0 \Rightarrow i_C = -i_A - i_B \tag{2.20}$$

It is worth noting that there are ways to estimate all three phase currents based on measuring a single current [14], but such approaches will not be examined further in this thesis.

2.6 Angular measurement

For FOC to work adequately, accurate information regarding the rotor's position is needed as described in section 2.4. This information is typically acquired using one of two main sensor techniques; optical encoders or magnetic encoders. An optical encoder consists of a code disk with two or three tracks, with a light source on one side of the disk, and one receiver per channel on the other side. An illustration of such a disk is shown in figure 2.12, where the gray depicts the disk and the white parts represent the slots in the disk that allow the light to pass through from the source to the receivers. There are an equal number of slots in channels A and B, evenly spaced around the disk with a 90° phase shift with respect to each other, ensuring that one of the channels always leads the other. Depending on which channel leads the other, it is possible to detect not just the incremental changes, but also the direction of rotation. The third channel, I, is optional and has a single slot and can be used to count the amount of full rotations. When the disk is moving at constant velocity, the waveforms of the output are square waves meaning that the A and B outputs are quadrature-encoded, and such encoders are therefore often referred to as quadrature encoders. The disk is mounted on the axle of the motor,

and a section of the output from the quadrature encoder when rotating the axle counterclockwise is shown in figure 2.13. The number of slots on the primary and secondary channels, i.e. A and B, is called pulses per revolution (PPR), which determines the accuracy of the angular measurement. Registering both rising and falling edges on both channels as shown in figure 2.14 and keeping track of the total pulse count, $P_{tot,count}$, the angle is given by

$$\theta_m = \frac{P_{tot,count} \cdot 360}{4 \cdot PPR} \tag{2.21}$$

with a measurement accuracy of

$$\theta_{m,acc} = \pm \frac{360}{4 \cdot PPR} \tag{2.22}$$

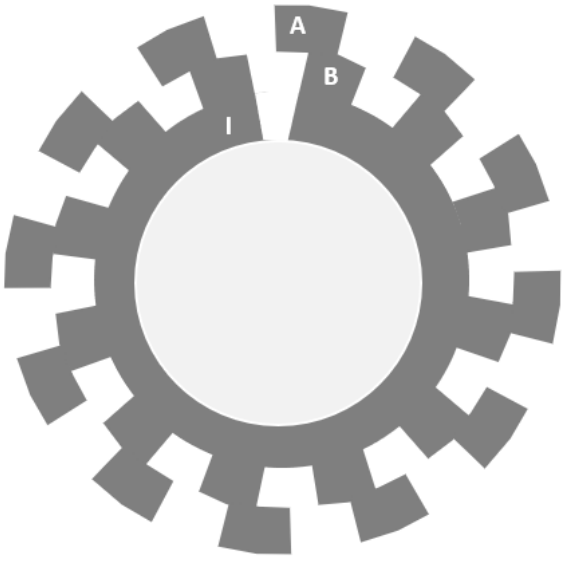


Figure 2.12: Code disk of a quadrature encoder with three tracks. Adapted from [1].

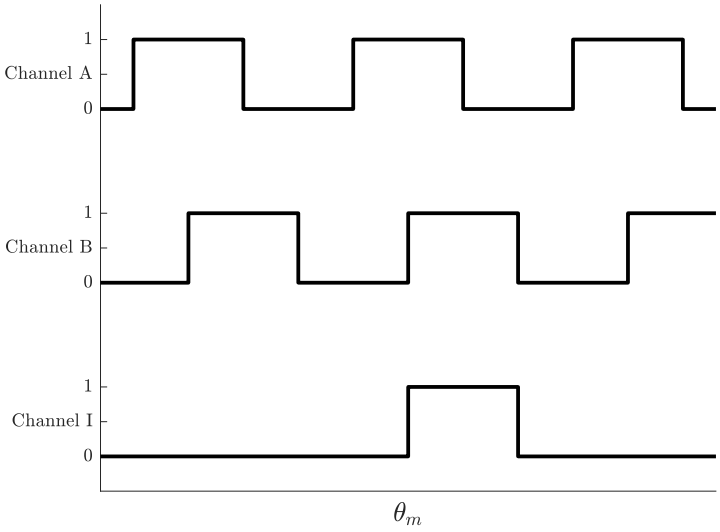


Figure 2.13: Quadrature encoder output, counterclockwise rotation.

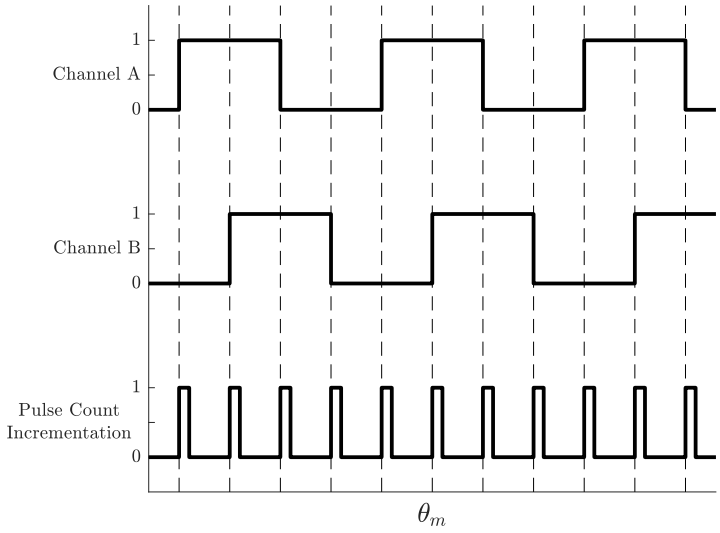


Figure 2.14: Quadrature encoder and pulse count incrementation output, counterclockwise rotation.

The main drawbacks of optical encoders include that the code disks are susceptible to damage and external particles, such as dust, dirt, or moisture since they can interfere with the light beam from the source to the receiver [15]. A more robust alternative to optical encoders is magnetic encoders, such as the Hall effect sensors described in detail in section 2.3, since they aren't as susceptible to damage and external particles compared to optical encoders. Magnetic incremental encoders are implemented by mounting a magnetic disk on the rotor axle instead of the optical code disk used in quadrature encoders and detecting changes in the magnetic field

using a Hall latch. By using two Hall latches offset by 90 electrical degrees, it is possible to measure both rotational speed and direction. By increasing the number of poles on the magnetic disk, the resolution of the position sensing also increases, but the generated magnetic flux density decreases. If the magnetic flux is too small, it won't affect the output of the Hall latches, meaning that the position can no longer be measured. Thus, magnetic encoders are unable to provide as high resolutions as optical encoders.

A third option, which combines high resolution with robustness is capacitive incremental encoders. Capacitive incremental encoders consist of three main components, namely a stationary transmitter, a rotor with a sinusoidal pattern, and a stationary receiver [16]. As the rotor rotates, a high-frequency reference signal is modulated predictably, and detected by the receiver. The encoder detects changes in capacitance-reactance and translates them into increments of rotary motion, mimicking the output shown in figure 2.14. In summary, capacitive incremental encoders offer high accuracy and identical output as optical incremental encoders, but with lower current consumption and high resistance to external particles.

2.7 PWM

The DC voltage source can supply either 0% or 100% of its maximum capacity at any given time instance. To control the motor efficiently, it is necessary to be able to output the whole range of voltages between 0-100%. This is achieved by switching between 0 and 100% at a high frequency, thus controlling the average voltage over time, as shown in figure 2.15. The figure shows an example with a DC voltage source and a constant duty cycle of 50%. The duty cycle is given by the period T, and the time the PWM-signal is on during that period t_{on}, according to

$$\text{Duty cycle} = \frac{t_{on}}{T} \cdot 100\% \tag{2.23}$$

Given that the PWM frequency, $\frac{1}{T}$, is sufficiently high, the applied voltage will be $\approx$ Duty cycle $\cdot V_{max}$, where V_{max} is the maximum capacity of the DC voltage source. By updating the duty cycles and switching the direction of the current based on the angle of the rotor, it is possible to accurately apply the desired voltage at each of the three phases of the motor.

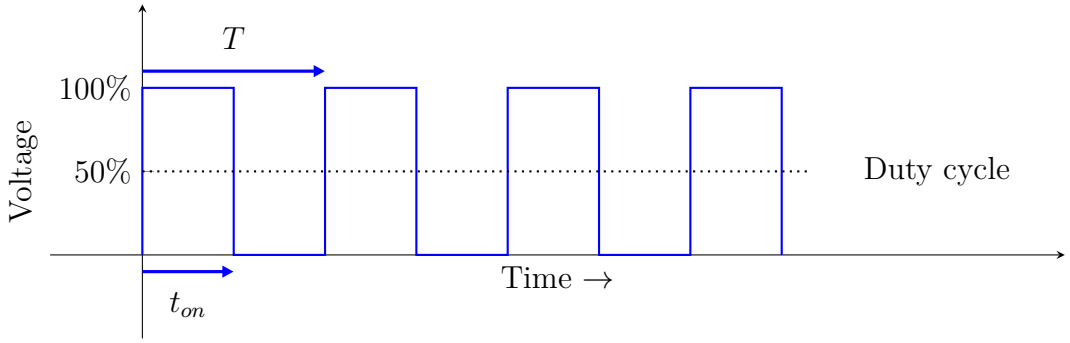


Figure 2.15: Pulse-width modulation principle, 50% duty cycle

2.8 Three-phase inverter

The stator magnetic field can be controlled using a three-phase inverter which consists of 6 transistors, as shown in figure 2.16. The three-phase inverter converts DC power to AC power, providing the required AC voltage and frequency to drive the motor. Each phase has a low-side transistor and a high-side transistor, respectively. If a transistor is closed, i.e. if current flows through it, it will be referred to as "on". If a transistor is open, i.e. no current flows through it, it will be referred to as "off". This is further illustrated in figure 2.17, where the transistor configuration is presented in table 2.2. If both transistors of a phase are off, there is no current passing through the phase from the voltage source. If both transistors of a phase are on, however, the circuit is shorted. Thus, it must be assured that at least one of the transistors of a phase is off at all times, typically conducted by introducing a small delay between switching states. By using PWM to control the rapid switching of states, it is possible to create the desired phase currents. For example, the sinusoidal phase currents shown in figure 4.5a can be generated using this technique.

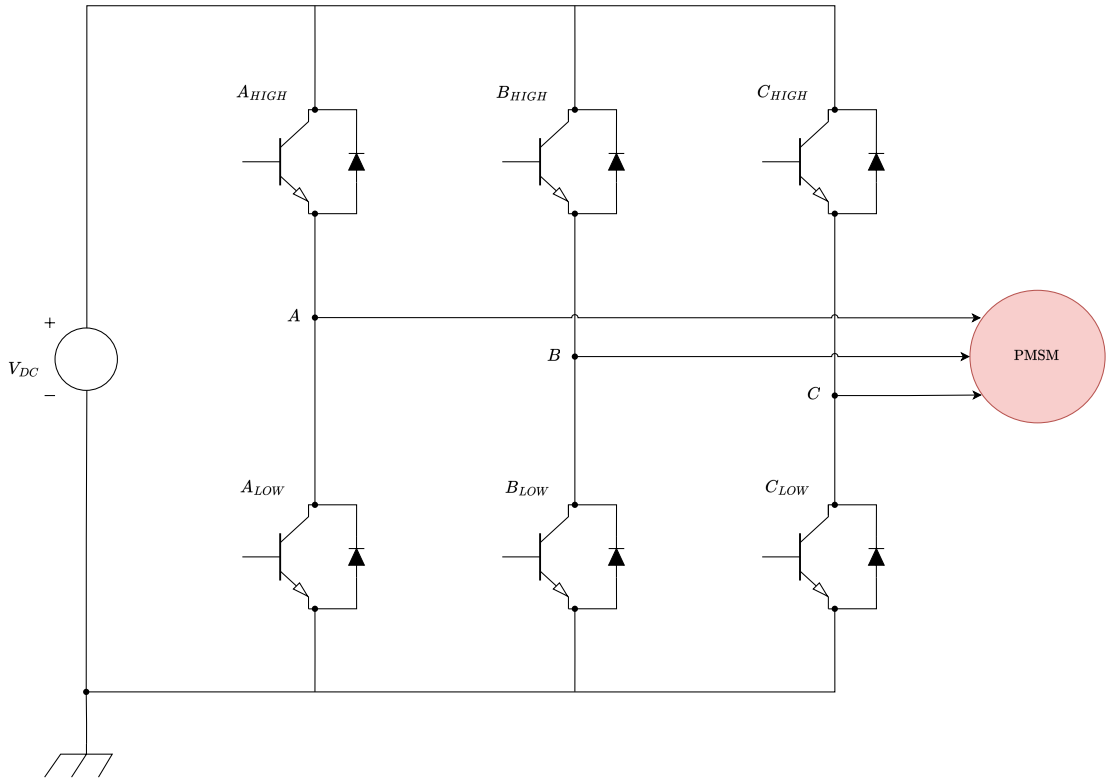
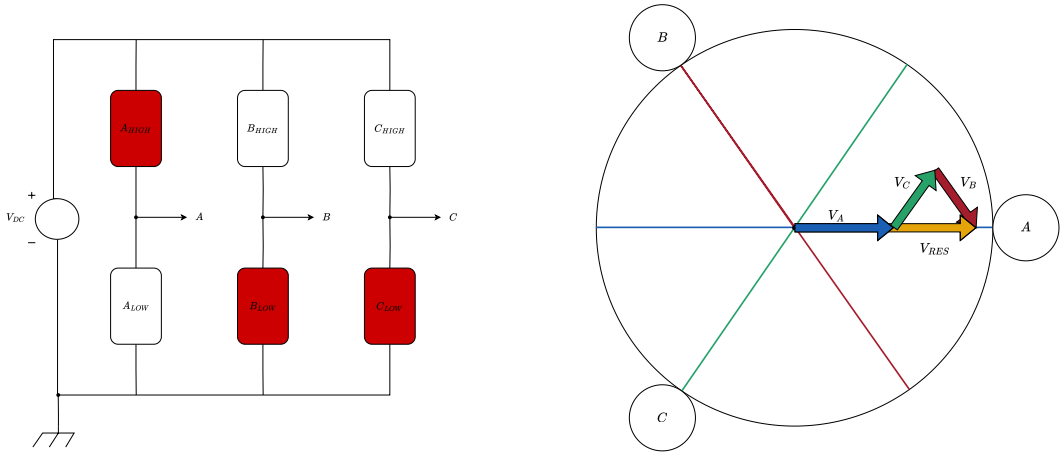


Figure 2.16: Schematic of three-phase inverter, connected to a DC voltage source and a PMSM

Table 2.2: Transistor configuration in figure 2.17a

Phase A	Phase B	Phase C
high-side on low-side off	high-side off low-side on	high-side off low-side on



(a) Three-phase inverter state

(b) Corresponding phase voltage

Figure 2.17: Arbitrary state of the three-phase inverter and the corresponding phase voltages

3

Hardware

This chapter presents the hardware utilized for developing and testing PMSM control strategies. It is divided into several sections that discuss the individual components of the entire system and the fully assembled system.

3.1 Microcontroller

The nRF52 development kit (DK) for the nRF52832 SoC by Nordic Semiconductor [17] was chosen as the system's development board due to its high versatility and large array of advanced peripherals.

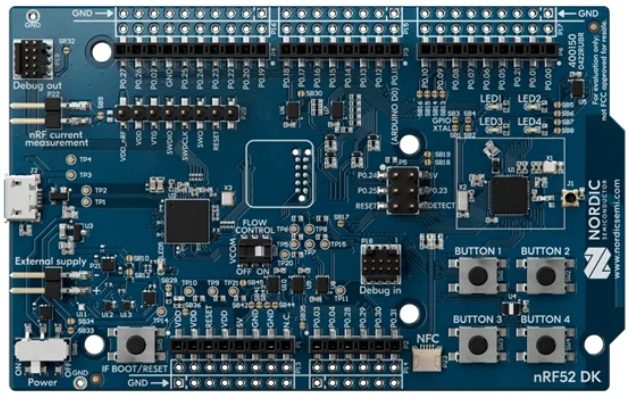


Figure 3.1: nRF52 development kit by Nordic Semiconductor

3.2 Motor driver

The X-NUCLEO-IHM07M1 [18] by STMicroelectronics was employed as an intermediary interface between the nRF52 development kit and the PMSM, facilitating shunt resistors for current sensing on each motor phase, connections for an incremental encoder as well as a DMOS driver for seamless power delivery. These are crucial features for PMSM motor control.

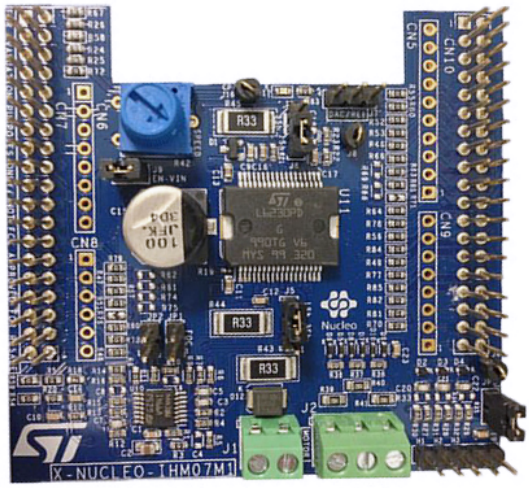


Figure 3.2: X-NUCLEO-IHM07M1 motor driver board by STMicroelectronics

3.3 Brushless DC motor

An RS Pro Brushless DC Motor [19], a surface mount PMSM, was utilized throughout the development and testing phases of the firmware.

Figure 3.3: RS Pro Brushless DC Motor (surface mount PMSM)

The parameters presented in table 3.1 are the most relevant parameters from the motor's datasheet.

Table 3.1: Motor parameters

Pole Pairs (n_{pp})		2
Line to Line Resistance (R_s)	Ω	0.65
Line to Line Inductance (L_s)	mH	2.1
Torque Constant (K_T)	Nm/A	0.063
Rotor Inertia (J)	gcm^2	119

From the parameters presented in table 3.1 it is possible to compute the back-EMF

constant, K_e, and the rotor magnetic flux linkage ,ψ_r, according to

$$K_e = \frac{\sqrt{3}K_T}{2} \tag{3.1}$$

$$\psi_r = \frac{\sqrt{2/3}K_e}{n_{pp}} \tag{3.2}$$

3.4 Encoder

The motor shown in section 3.3 has built-in hall effect sensors that could be used for motor control. These were not utilized, however, since more precise rotor angle measurements were desired. Instead, the encoder AMT102-V [20] by CUI Devices was affixed onto the rear section of the motor for precise position feedback. It is a capacitive incremental encoder and it was set up to output 48 PPR.

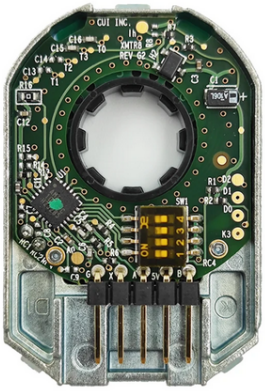


Figure 3.4: AMT102-V encoder by CUI Devices

3.5 Assembly

Figure 3.5 shows the entire hardware assembly.

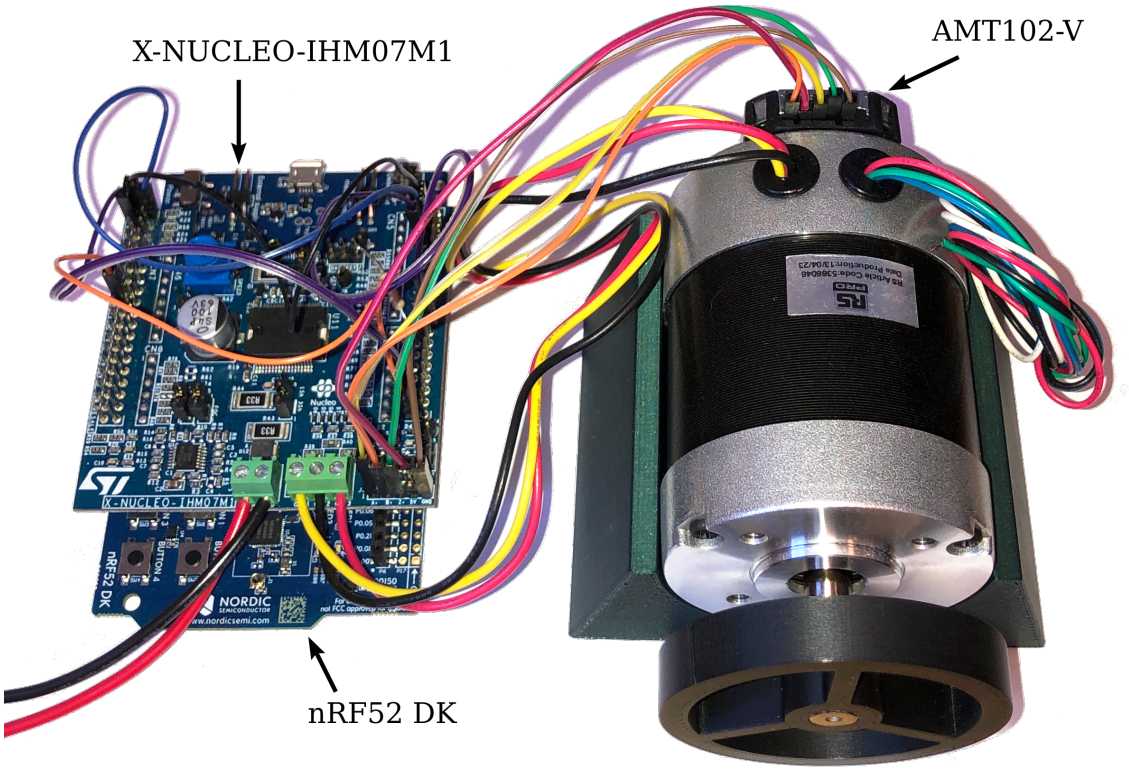


Figure 3.5: Assembled hardware

4
Embedded firmware

This chapter details the implementation of the system depicted in Figure 2.11 on the nRF52 DK for the nRF52832 SoC [17]. Section 4.1 covers the implementation of FOC on the system. Section 4.2 discusses the CPU interrupts that interfere with the motor control, highlighting the potential issues arising from these interrupts. Section 4.3 describes the modifications made to the firmware to manage these interrupts. Finally, the chapter concludes with an analysis of code execution time.

4.1 Implementation without interrupts

The initial phase of firmware development involved implementing functional FOC and velocity control on a system with uninterrupted CPU operation. This included measuring the phase currents, tracking the rotor angle as well as computing and outputting the correct phase voltages to achieve the desired motor rotation. Figure 4.1 illustrates how the firmware was divided into different modules and the information that is exchanged between them. Detailed explanations of each module are provided in the subsections below.

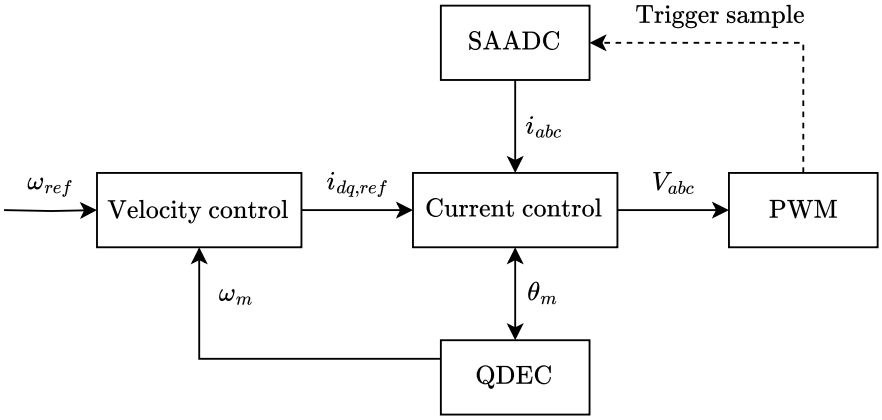


Figure 4.1: Map of the firmware modules

4.1.1 PWM

The PWM module on the nRF52832 facilitates the generation of pulse width modulated signals. It operates through an up or up-and-down counter system, offering four PWM channels per instance that can drive assigned GPIO pins. The PWM module contains a "wave counter", which is a hardware function responsible for generating pulses at configured duty cycles based on the compare values and at a frequency that depends on the counter top value. The PWM module has a fixed base clock frequency of 16 MHz and thus increments the counter at the same frequency. The counter is automatically reset to zero when the counter top value is reached in the up counter mode, which results in edge-aligned duty cycles. In the up-and-down counter mode, however, the counter starts decrementing to zero when the counter top value is reached, resulting in center-aligned duty cycles. The PWM frequency is determined by the PWM base clock frequency in combination with the clock pre-scaler, counter top value and counter mode.

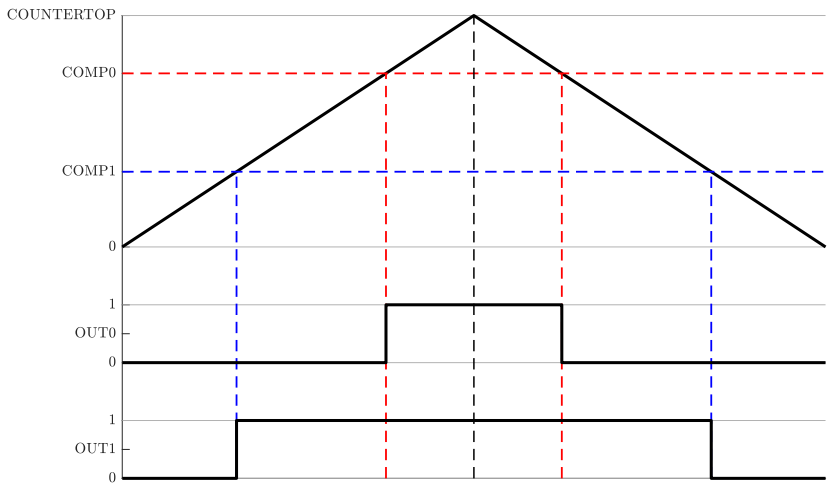


Figure 4.2: Counter and two channel outputs for a PWM instance configured for center-aligned duty cycles

Figure 4.2 illustrates one period of the PWM counter in the up-and-down mode and two of the four channel outputs. The figure illustrates only two channel outputs for simplicity, but all four channels were used in the firmware (one PWM channel for each motor phase and the last channel for triggering current measurements). A benefit of using center-aligned PWM is that it reduces acoustic noise for a given PWM switching frequency by doubling the effective current ripple frequency, providing the optimum tradeoff between switching losses and acoustic noise [21].

The required counter top value, C_{top}, for a desired PWM frequency with center-aligned duty cycles, is computed as

$$C_{top} = \frac{f_{PWM,baseclock}}{2 \cdot f_{PWM,desired}} \tag{4.1}$$

The PWM frequency was set to 20kHz, corresponding to a counter top value of 400. A PWM frequency of 20 kHz or higher is desirable since it moves the acoustic frequencies outside of the human hearing range [22].

The needed compare values (COMP) for the desired duty cycles of each phase are computed as

$$COMP_i = C_{top} - \left(\frac{C_{top}}{2} + MAX\_DUTY * \frac{V_i}{V_{max} C_{top}/2} \right) \qquad i \in [A, B, C] \quad (4.2)$$

where V_{max} is the maximum voltage that can be outputted.

4.1.2 QDEC

The nRF52832 features a quadrature decoder (QDEC), a peripheral that samples a quadrature encoder and decodes transitions automatically and independently from the CPU. The QDEC has a minimum sample time of $128\mu s$, corresponding to a maximum sample frequency of 7812.5 Hz. This introduces some limitations on the system. The decoder compares the previous sample to the current sample to determine if the rotor has rotated or not, the direction it has rotated, or if an invalid transition has occurred.

The decoding of the samples is described in table 4.1 [17]. As can be seen, if both encoder pulses change from the previous sample to the next, it results in an invalid transition. The limitation this introduces is that the rotor is not allowed to rotate at velocities large enough to introduce such invalid transitions between samples. The minimum angle between samples for a double transition to occur, $\theta_{dbl,min}$, is given by (2.22), and the maximum angular velocity while still ensuring valid transitions thus depends on the PPR. For example, if the quadrature encoder has a resolution of 48 PPR, then $\theta_{dbl,min} \approx 1.84°$. The maximum angular velocity of the rotor, RPM_{max}, without risking double transitions is given by

$$\text{RPM}_{max} = \frac{\theta_{dbl,min}}{6 \cdot T_s} \quad (4.3)$$

where T_s is the sample time in seconds of the decoder. With a sample time of $128\mu s$, and a resolution of 48 PPR, the maximum angular velocity of the rotor without theoretically risking double transitions is approximately 2440 RPM, according to (4.3). There is, thus, a trade-off between the resolution of the encoder and the maximum angular velocity of the rotor, dependent on the sample time of the decoder.

Table 4.1: Decoding of quadrature encoder sample pairs

Previous Sample pair		Current Sample pair		Sample value	Description
A	**B**	**A**	**B**		
0	0	0	0	0	No movement
0	0	0	1	1	Movement in positive direction
0	0	1	0	-1	Movement in negative direction
0	0	1	1	2	Invalid transition
0	1	0	0	-1	Movement in negative direction
0	1	0	1	0	No movement
0	1	1	0	2	Invalid transition
0	1	1	1	1	Movement in positive direction
1	0	0	0	1	Movement in positive direction
1	0	0	1	2	Invalid transition
1	0	1	0	0	No movement
1	0	1	1	-1	Movement in negative direction
1	1	0	0	2	Invalid transition
1	1	0	1	-1	Movement in negative direction
1	1	1	0	1	Movement in positive direction
1	1	1	1	0	No movement

The QDEC accumulates both valid and invalid transitions in two registers named ACC and ACCDBL. These registers are read by the firmware and a total pulse count ($P_{tot,count}$) is maintained which is linearly proportional to the rotor angle of the PMSM as

$$\theta_m = \frac{2\pi \cdot P_{tot,count}}{4 \cdot PPR} \tag{4.4}$$

The firmware updates the total pulse count (angle) at every sample, i.e., at 7812.5 Hz. To get an angular velocity measurement the firmware utilizes a timer to measure the time since the last angular velocity update, Δt [17]. The angular velocity is updated based on the accumulated transitions since the last update, ΔP, and translated to rad/s, according to

$$\omega_m = \frac{\Delta P}{\Delta t} \cdot \frac{2\pi}{4 \cdot PPR} \tag{4.5}$$

The value of ΔP is reported every 200 samples of the QDEC, meaning that the angular velocity is updated at a frequency of approximately 39Hz.

4.1.3 SAADC

The nRF52832 features a successive approximation analog-to-digital converter (SAADC), which enables precise and efficient conversion of analog signals into digital measurements for accurate sensor readings, in this case, current measurements. Successive approximation is a technique used in analog-to-digital conversion (ADC) where the analog input voltage is compared to a reference voltage. The conversion process

iteratively approximates the analog input voltage by successively narrowing down the range of possible values until the digital representation is obtained [17].

To sample the voltage over the shunt resistor on a single channel, the input is connected to a capacitor during the sampling. The capacitor is connected during a short period, called the acquisition time t_{acq}, whose longevity depends on the shunt resistance. Lower resistances enable shorter acquisition times, with a minimum of $3\mu s$. There is also a conversion time, t_{conv} of $<2\mu s$ at which a sample is converted, occurring immediately after a sample has been acquired, and before starting an acquisition of another sample.

As mentioned in section 2.5, (2.20), it is possible to measure two phase currents and calculate the third. An assumption in the FOC algorithm is that the phase currents are measured simultaneously, which is not always possible in physical systems. On the nRF52832, for example, the SAADC can only sample one channel at a time. If more than one channel is sampled, the SAADC samples the channels successively. The total sample time, t_{tot}, is given by

$$n_{ch} \cdot t_{acq} < t_{tot} < n_{ch} \cdot (t_{acq} + t_{conv}) \tag{4.6}$$

where n_{ch} denotes the number of channels that are sampled. According to (4.6), the total sample time for two channels is $6\mu s < t_{tot} < 10\mu s$, with an acquisition time of $3\mu s$. The time between the end of the acquisition of two samples is $t_{conv} + t_{acq} < 5\mu s$, since the first sample has to be acquired and converted before the sampling of the second channel is started.

This has two main implications for the system, the first being related to the maximum phase voltage output. Assuming low-side current measurement, as explained in section 2.5, phase-current has to be measured when the corresponding low-side transistor is on. This subsequently means that the PWM signal on that specific phase has to be set to low for at least t_{acq} to enable phase current measurement to take place. It should be noted that the state of the 3-phase inverter does not affect the conversion of the sample, i.e., the PWM signal is allowed to go from low to high during this time. When measuring two channels, both PWM signals should be set to low while the measurement takes place, which corresponds to a time of $t_{acq} + t_{conv} + t_{acq} < 8\mu s$. Assuming a PWM period of $50\mu s$, corresponding to a frequency of 20 kHz, and a deadtime between going from high-side transistor on to low-side transistor on of $1\mu s$, the maximum duty cycle should be at most $(50 - 8 - 1)/50 = 82\%$ to ensure that the phase currents are sampled correctly. Preferably, even more time should be allocated for the ADC sampling to avoid the acquisition of noise-induced on one phase current feedback by switch commutations on other phases [23]. It is, therefore, not possible to fully utilize the input voltage using the low-side current measurement technique, resulting in a drop in the maximum amount of torque that can be generated. To conclude, there is a trade-off between the PWM frequency and the produced torque since the current measurement takes a certain amount of time and must be conducted when the low-side transistors are on.

The second implication is that issues may occur if the inductance of the motor coils is too low. Each phase can be represented by a resistor and an inductor connected in series, i.e., an RL-loop. To demonstrate why low inductances can cause problems, consider an example where the DC voltage source is 12V, R = 1Ω, L = 1H, and the desired phase voltage is 6V, corresponding to a 50% duty cycle. With a PWM frequency of 20kHz, the current through the system follows the trajectory shown in figure 4.3. To illustrate how the current through the system changes between PWM periods, an exaggerated exemplification is shown in figure 4.4a, where the parameters for resistance and inductance remain the same, but the PWM frequency is changed to 1Hz. As can be seen in the figure, the current through the system changes relatively slowly due to the high inductance. If the inductance is lowered to L = 0.1H, the exaggerated behavior of the current through the system changes to what is shown in figure 4.4b. As expected, the current through the system changes much more rapidly when the inductance is lower.

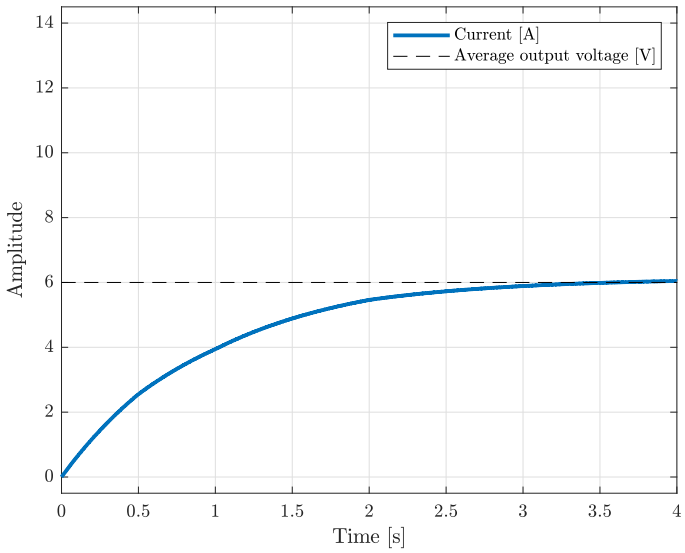
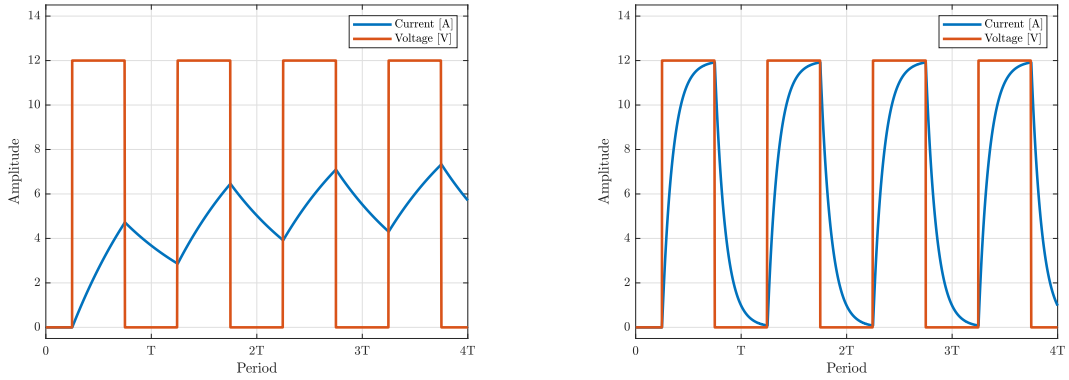


Figure 4.3: Current through the RL loop at R = 1Ω, L = 1H, 50% duty cycle at 20kHz frequency

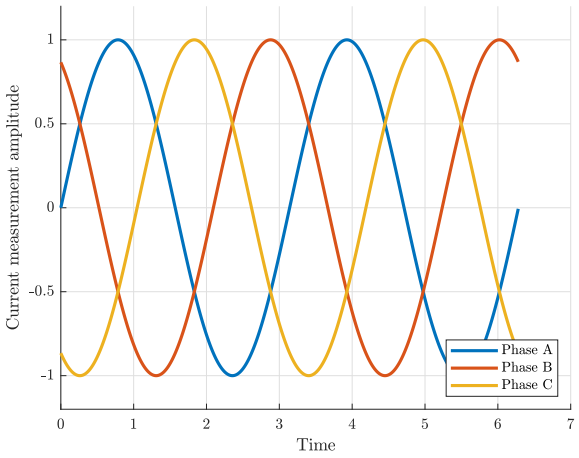


(a) High inductance, L = 1H

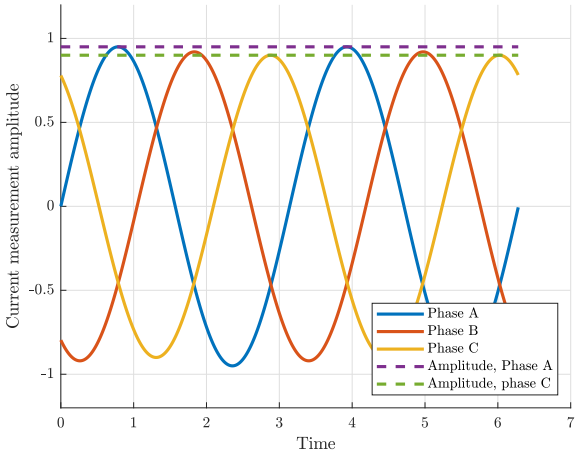
(b) Low inductance, L = 0.1H

Figure 4.4: Exaggerated illustration of how the current changes during and between PWM periods

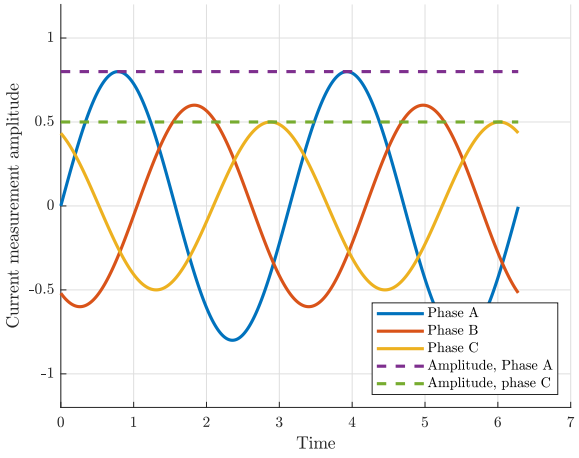
As previously explained, the current measurement is done sequentially, with approximately 5 μs between the end of the sampling of two currents. If the inductance is too low, the current will change significantly during this time, which makes the measurements inaccurate. To highlight the effects of this, assume that the motor is rotated by supplying sinusoidal voltages with the same amplitude and frequency at each phase, phase shifted by 120° each. The ideal phase current measurements should, therefore, also be perfectly sinusoidal with the same amplitude, phase shifted by 120° each. Figure 4.5 shows the comparison between the current measurements results, where figure 4.5a shows the ideal current measurements, figure 4.5b shows the resulting measurements when the inductance is high, and figure 4.5c shows the resulting measurements when the inductance is low. As can be seen, there is a significant difference in the amplitude of the sine waves when the inductance is low. The assumption behind the resulting figures is that the currents are being measured sequentially in chronological order from Phase A to Phase C. The problem is somewhat mitigated when measuring two currents and calculating the third according to (2.20), but it is only a slight improvement as the first two measurements are still inaccurate if the inductance is too low.



(a) Ideal measurements



(b) High inductance



(c) Low inductance

Figure 4.5: Comparison of how the inductance affects the current measurement results

In the firmware, SAADC samples are triggered on falling edges of the fourth PWM channel. The fourth PWM channel is configured to always have a longer duty cycle than the other three PWM channels. This ensures that the low-side transistors that drive the motor phases are on. The phase currents, therefore, go through the shunt resistors, enabling accurate measurements of the phase currents.

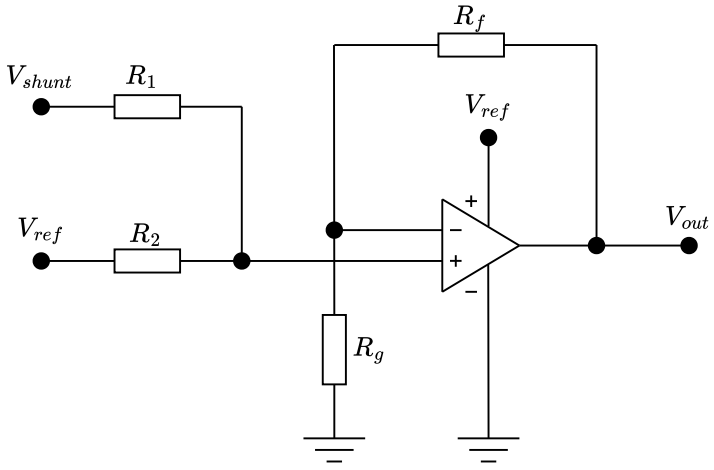


Figure 4.6: Current measurement amplifier circuit

Figure 4.6 illustrates the amplifier circuit for one of the three shunt resistors on the X-NUCLEO-IHM07M1. Here, $V_{ref} = 2.8V$, $R_1 = 680\Omega$, $R_2 = 2200\Omega$, $R_f = 2200\Omega$, $R_g = 2200\Omega$ and V_{shunt} is the voltage drop over the shunt resistor R_{shunt}. It is important to have knowledge of this circuit to be able to compute the phase currents correctly [24]. It is also possible to acquire the correct scaling of the current measurements through calibration.

When the SAADC has finished sampling and converting, the corresponding phase currents are computed according to

$$i = \frac{V_{shunt}}{R_{shunt} \cdot G_{OP}} = \frac{V_{meas} - V_{offset}}{R_{shunt} \cdot G_{OP}} \tag{4.7}$$

where $R_{shunt} = 0.33\Omega$ and G_{OP} is the operational amplifier gain of the current measurement circuit on the X-NUCLEO-IHM07M1 [18] computed as

$$G_{OP} = \frac{(R_2 \cdot R_f)/R_g + R_2}{R_1 + R_2} \approx 1.53 \tag{4.8}$$

V_{offset} is a voltage offset that is produced by the operational amplifier as

$$V_{offset} = \frac{V_{ref} \cdot R_1 \cdot G_{OP}}{R_2} \approx 1.32V \tag{4.9}$$

The reason for subtracting this offset from V_{meas} is to center the current measurements around 0A.

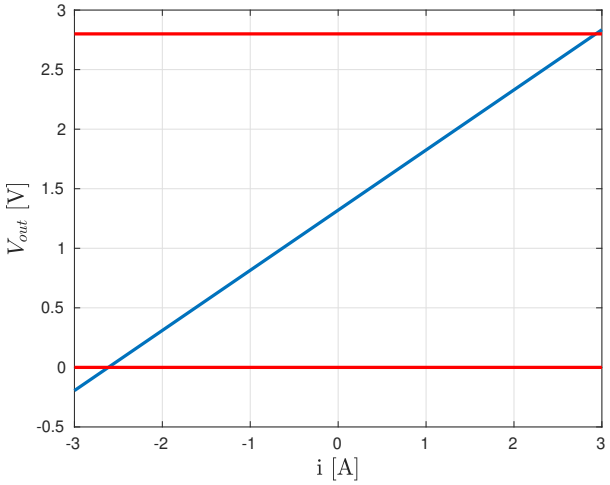


Figure 4.7: Voltage out in relation to actual current

The voltage output from the amplifier circuit, V_{out}, can then be described by the function

$$V_{out} = G_{OP} \cdot R_{shunt} \cdot i + V_{offset} \tag{4.10}$$

which is illustrated in figure 4.7.

V_{meas} is the measured voltage drop over the shunt resistor per

$$V_{meas} = \frac{S_{out} \cdot V_{ref,SAADC} \cdot G_{vref}}{RES_{SAADC}} \tag{4.11}$$

where S_{out} is the raw SAADC sample value of V_{out}, $V_{ref,SAADC}$ is the SAADC reference voltage, G_{vref} is the voltage reference gain and RES_{SAADC} is the SAADC resolution. In the firmware, $V_{ref,SAADC}$ was set to use the internal voltage reference of the nRF52832, which is 0.6V, and V_{gain} was set to 4. The internal voltage reference is more stable and accurate than the input voltage to the microcontroller. The SAADC resolution was set to 12 bits meaning $RES_{SAADC} = 2^{12} = 4096$.

To conclude this section, it is of utmost importance to be aware of the current measurement technique and time the measurement accordingly to provide accurate results. It is also vital to examine the specific hardware and to make sure that the inductance of the motor is high enough, such that the measured phase currents are accurate, especially if the measurements are conducted sequentially. This must be ensured for the FOC algorithm to work as intended. A good first step to make sure that the inductance is sufficiently high is to compare the phase current measurements and confirm that the amplitudes are similar to each other when applying a sinusoidal voltage at each phase, phase shifted by 120° each, which should yield similar results to what is shown in figure 4.5b. A second step to affirm that the measurements are

accurate is to compare them to the measurements using a current probe connected to an oscilloscope and measuring the phase currents manually.

4.1.4 Current control

The current control (FOC) module receives information from the modules described in sections 4.1.1, 4.1.2 and 4.1.3 as shown in figure 4.1. It is responsible for computing the appropriate phase voltages according to the reference or target currents that it receives.

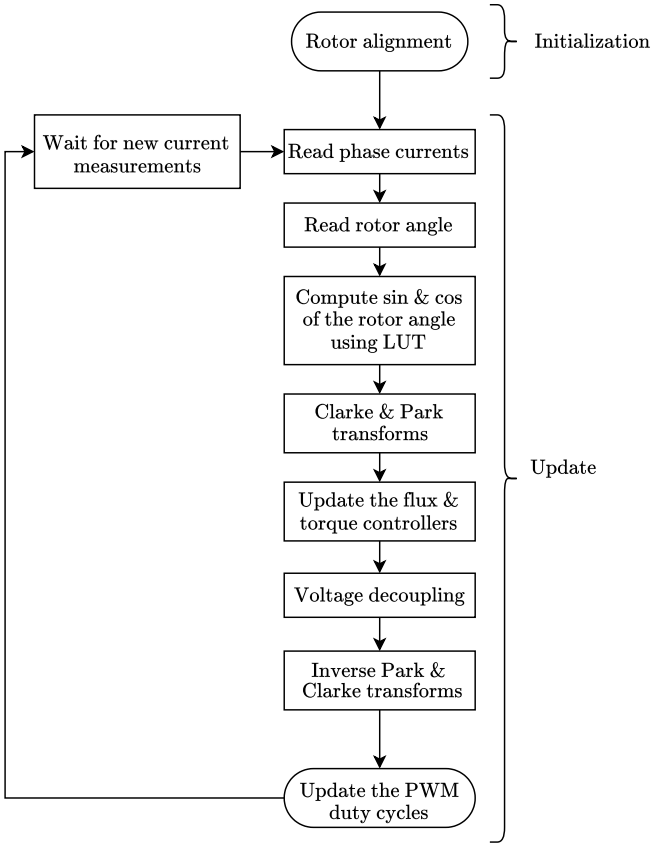


Figure 4.8: Flow diagram of the FOC code

Figure 4.8 illustrates the main flow of the current control module. The code performs an alignment routine at system boot to ensure that the rotor is positioned correctly. The alignment routine energizes the motor phases in a sequence that ends where one of the rotor's magnetic north pole is aligned with phase A. When the rotor is aligned, the firmware sets θ_m to 0 and the initialization is completed.

As mentioned in section 4.1.3, current measurements are triggered by the fourth PWM channel. When the sampling and conversions of the current measurements are done the firmware immediately performs an FOC update. It is crucial to use the

absolute latest and most relevant information in order to output the best control signals as fast as possible. The first steps of the update consist of reading the phase currents and the rotor angle as well as computing the sine and cosine of the angle. The sine and cosine values of the angle are used several times in the subsequent steps and in order to save some execution time, these are computed once and then reused. To reduce the execution time further, they are computed using a sinewave lookup table (LUT). Execution times are discussed in further detail in section 4.4. Then, the phase currents are transformed from the ABC reference frame into the *dq*-frame, as described in section 2.2. The final steps of the update include regulating the *dq*-currents using two PI controllers, one for the *d*-current (related to flux) and one for the *q*-current (related to torque). The outputs from the controllers are voltages in the *dq*-frame which need to be inverse transformed into the ABC-frame before being sent to the PWM module. Before transforming the *dq*-voltages however, voltage decoupling is performed as explained in section 2.4.2 such that individual control of the currents is achieved. The PWM module then makes sure that the computed phase voltages are applied to the motor phases. After this, one FOC update has finished and it will be executed again whenever new current measurements are available.

As mentioned previously in section 4.1.1, the PWM frequency is set to 20 kHz and current measurements are taken at 20 kHz, the current control code therefore runs at 20 kHz, given that code execution is fast enough. More on this in section 4.4.

The electrical dynamics of a PMSM can be described by a transfer function in the Laplace domain that relate the *dq*-voltages to their respective currents. Since $L_d \approx L_q$ for surface mount PMSMs, the transfer functions for both axes are given by

$$G_{V,i}(s) = \frac{1}{sL + R_s} \tag{4.12}$$

The motor's electrical time constant T_s is determined as $T_s = L/R_s$ according to (4.12).

The Direct Synthesis method was used to find suitable current controllers [25]. The ideal closed-loop transfer function for the current controllers is thus

$$L(s) = \frac{\frac{1}{s\tau}}{1 + \frac{1}{s\tau}} \tag{4.13}$$

where τ is the desired time constant of the closed loop transfer function.
This means that $F(s)G_{V,i}(s) = \frac{1}{s\tau}$, and the transfer function for the controllers is therefore given by

$$F(s) = \frac{1}{s\tau} G_{V,i}^{-1}(s) = \frac{1}{s\tau}(sL + R_s) = \frac{L}{\tau} + \frac{R_s}{s\tau} \tag{4.14}$$

which contains a proportional and an integral term, suggesting that PI-controllers are needed to regulate the currents. The controllers' theoretically optimal gains are

given by

$$K_p = \frac{L}{\tau}, \qquad K_i = \frac{R_s}{\tau} \tag{4.15}$$

The desired closed loop time constant τ was set to one-fourth of the motor's electrical time constant ($\tau = T_s/4$) to achieve a critically damped response. This choice ensures a balanced response between speed and stability. Using the motor parameters presented in section 3.3, the controller gains were computed to

$$K_p = 2.6, \qquad K_i = 2414 \tag{4.16}$$

4.1.5 Velocity control

As can be seen in figure 4.1, the firmware implements velocity control using a cascaded control structure. This configuration ensures that the velocity control loop commands the reference currents for the inner current control loops, thereby achieving precise velocity regulation. The velocity control loop reads the angular velocity from the QDEC module and computes appropriate current references using a PI controller. The velocity control loop was set up to run at 500 Hz, as the outer loop should be slower than the inner loop.

Unlike the current controllers, which were tuned without considering potential loads attached to the rotor, the velocity controller tuning needs to be adjusted based on the attached load. Therefore, the tuning for the velocity controller was determined empirically rather than through theoretical calculations since it would be difficult to determine the exact parameters needed to calculate theoretically optimal gains.

4.2 Code execution contexts

The motor control firmware presented in section 4.1 is just a small part of a larger code base. Naturally, the processor is often busy executing other functions with higher priorities. This section aims to give an insight into what sort of environment the motor control firmware has to fit into from a code execution time perspective.

To show what the different code execution contexts look like, a simple experiment was set up where a counter was incremented once every 50 μs using one of the nRF52832's timers. During normal code execution context, the counter increments linearly as in figure 4.9. Below the plot of the counter value, is a plot of the normalized change in counter value over time. As can be seen, the change is mostly around one which signifies that the counter value is updated at the expected periodicity of 20 kHz. There are a few points where the change is lower than one, indicating that some other process is taking up execution time. This scenario is considered to be the expected code execution context.

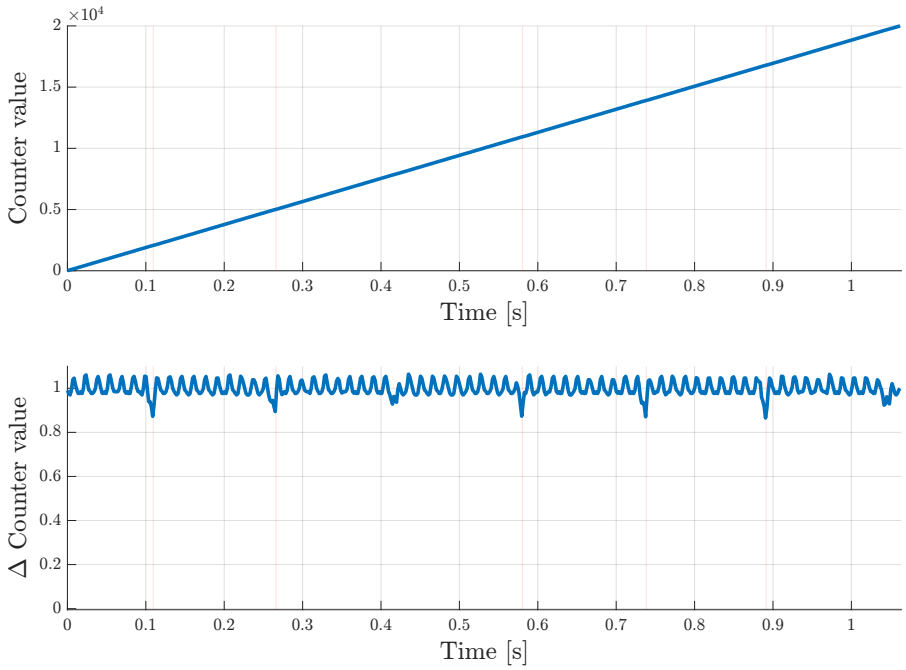


Figure 4.9: Normal context

In some cases however, the processor gets busy for longer periods as shown in figure 4.10. The regions where the normalized change in counter value goes below 0.9 are highlighted by the light red regions in the plots. In these regions, code execution is either fully or partially interrupted. Looking at the plot of the counter value it is clear that these interrupts affect the code execution since the counter is not incremented as expected. The longest interrupts endure for around 100 ms, which is considered the worst-case interrupt scenario for this thesis. Such long interrupts may result in serious consequences depending on the application, unless they are taken care of. Another thing about these longer interrupts is that it is not entirely

possible to predict when they occur or control them. Therefore, the application has to be ready to handle long interrupts at any time.

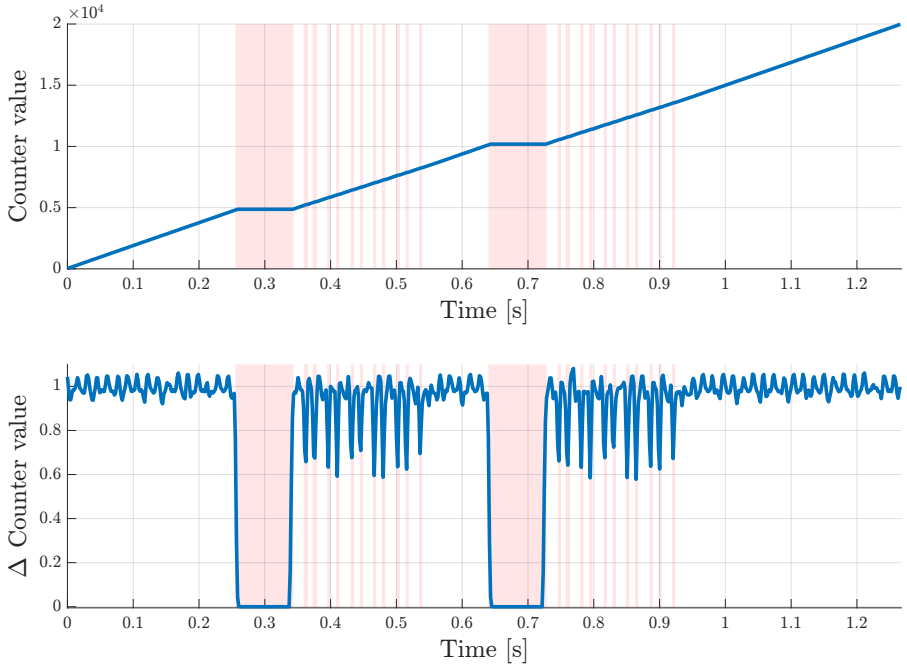


Figure 4.10: Interrupted context

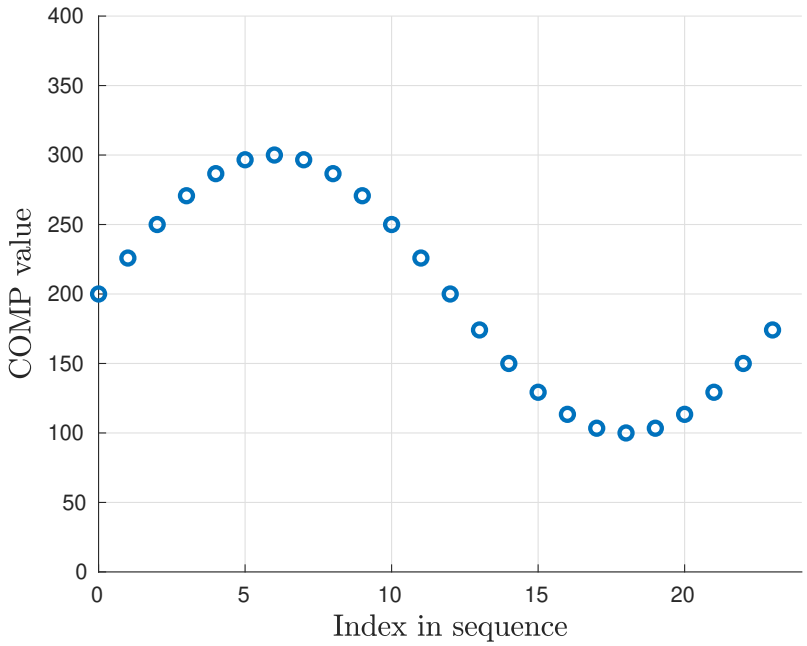
Regarding BLDC motor control, the motor will not rotate unless the phase voltages are updated according to the FOC algorithm. This is demonstrated in section 5.2.1. Since it is desired to maintain constant velocities at all times, including during long processor interrupts, the firmware needs to be modified and improved to handle these scenarios.

4.3 Implementation with interrupts

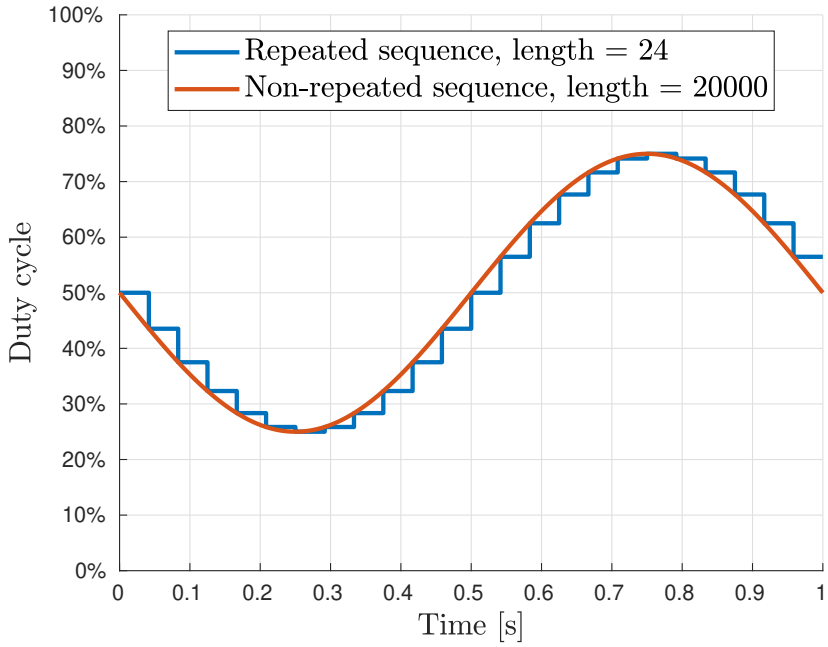
This section describes how the firmware implementation was modified to be able to handle interrupts of varying lengths that can occur at any time of operation, as discussed in section 4.2. More specifically, the implemented strategy aims to make sure that the rotor keeps rotating with the same angular velocity that it had at the very start of the interrupt until the CPU is available again. In order to achieve this, precomputed sine waves of the correct phase, period, and amplitude must be available in the PWM peripheral at all times, and for all three phases. One of the main features of the PWM module, described in detail in section 4.3.1, is that it provides ways to update the duty cycle values autonomously directly from memory, i.e. without needing CPU access. This is the fundamental principle that facilitates such strategies to work properly, but it should be noted that such strategies are open-loop control since no computations can take place during interrupts. The phase currents and angle can still be measured during interrupts, but the information can only be taken into consideration when the CPU is available. This means that no vital information is lost during the interrupts, but computations based on the information cannot take place until the CPU is available again.

4.3.1 PWM

The PWM module features a built-in decoder and EasyDMA capabilities which make it possible to update the PWM duty cycles without CPU intervention [17]. Arbitrary duty-cycle sequences are read from RAM by the PWM module, which is a key feature to output the desired phase voltages even when the CPU is busy with tasks of higher priority. A second important feature is that it is possible to configure the PWM module such that it repeats each value in a sequence a specific number of times before switching to the next value in the sequence. This is important since the PWM module has a predetermined and fixed base frequency, set to 20 kHz in this case. To highlight why this feature is important, consider the case where the desired output corresponds to a sine wave with a period of one second. If the values in the sequence are not repeated, the next value in the sequence will be outputted every 50μs, and a sequence length, n, of $\frac{1}{50 \cdot 10^{-6}} = 20000$ would be needed to be able to output the desired sine wave. By instead choosing an arbitrary number of evenly spaced discrete values in the same sine wave and repeating each value an appropriate number of times, the output will mimic the desired sine wave, albeit with a lower resolution. This is further illustrated in figure 4.11, where figure 4.11a shows the values of the shorter sequence, and figure 4.11b shows the corresponding duty cycle values for the two methods. The shorter sequence is of length 24, and each value is repeated $\frac{20000}{24} \approx 833$ times. As can be seen in the figure, there is a clear trade-off between n and the resolution of the outputted sine wave. As explained in section 4.1.3, the motor coils act as low-pass filters, mitigating the effects of the lower resolution. Therefore, the effects of the lower resolution are insignificant in comparison to the benefits of having to use vastly less memory.



(a) Discrete values in sequence of length 24



(b) Corresponding duty cycle output of the two methods

Figure 4.11: Duty cycle output comparison between repeated sequence of length 24 and non-repeated sequence of length 20000

Assuming a shorter sequence of fixed length is used to define the period of a sine wave, and then repeated the appropriate number of times to match a specific ω_m, there is a trade-off between the resolution of the sine wave and the range of different values ω_m corresponding to the same number of repeats. If n increases, the reso-

lution of the sine wave increases, but the granularity at higher velocities decreases, since a wider range of values of ω_m will have the same number of repeats, as shown in figure 4.12. This trade-off is due to the fact that the number of repeats must be an integer. The figure shows all corresponding numbers of repeats to all integers $\omega_m \in [1, 120]$ for a high and a low value of n. If $n = 240$, the number of repeats will be 4 for all $\omega_m \in [65, 86]$, highlighting the described issue. Therefore, it's important to make sure that n is large enough to provide an acceptable resolution of the sine wave, but small enough to provide short ranges of ω_m corresponding to a specific number of repeats for the range of ω_m at which the rotor will operate. Based on this, n was set to 24 for the implementation of the strategy, since the strategy was designed for rotor speeds in the range $\omega_m \in [0, 100]$ rad/s.

The number of repeats for a given ω_m is calculated as

$$repeats = \left\lfloor \frac{\omega_{m,max}}{\lfloor \omega_m \rfloor} \right\rfloor \tag{4.17}$$

where $\omega_{m,max}$ is given by

$$\omega_{m,max} = \frac{2\pi \cdot f_{PWM}}{n \cdot n_{pp}} \tag{4.18}$$

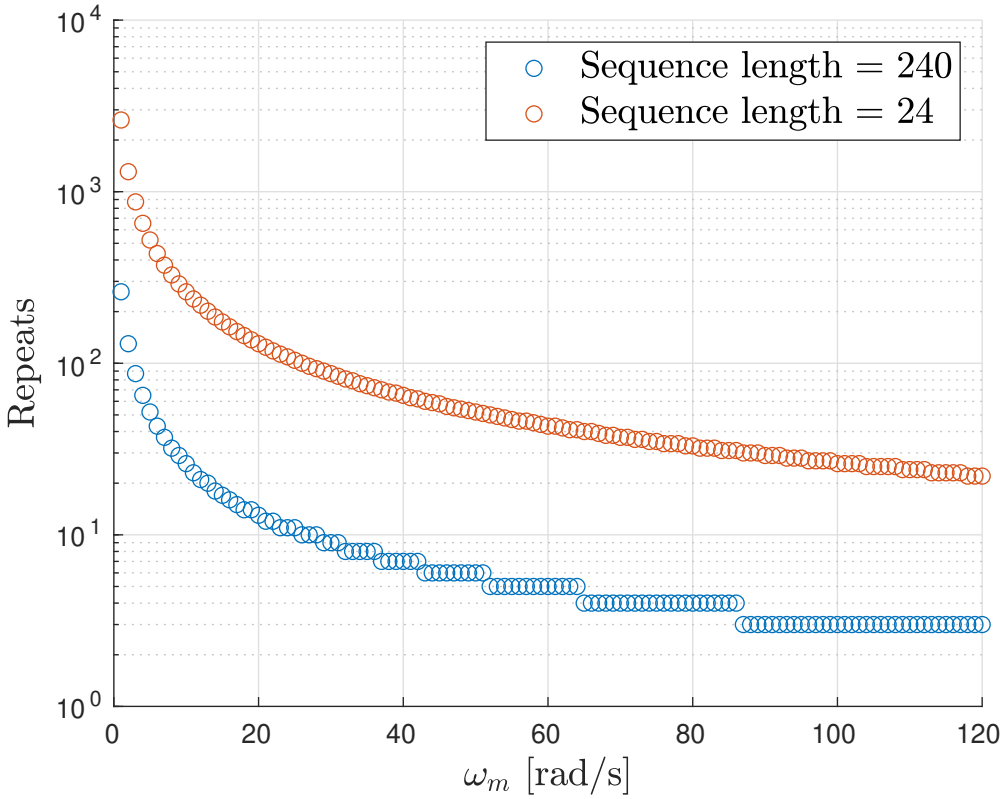


Figure 4.12: Corresponding number of repeats to all integers $\omega_m \in [1, 120]$ [rad/s] for a shorter and a longer sequence length, defining a period of a sine wave

A third important feature of the PWM module is that a specified sequence can be played through, with a specified number of repeats, and then played from the beginning of the sequence when the end of the sequence is reached. This will continue until the playback of another sequence is started. Defining a single period of a sine wave and playing back the sequence can create a continuous sine wave or any other desired sequence for as long as required. This means that there is no need to define multiple periods of the wave.

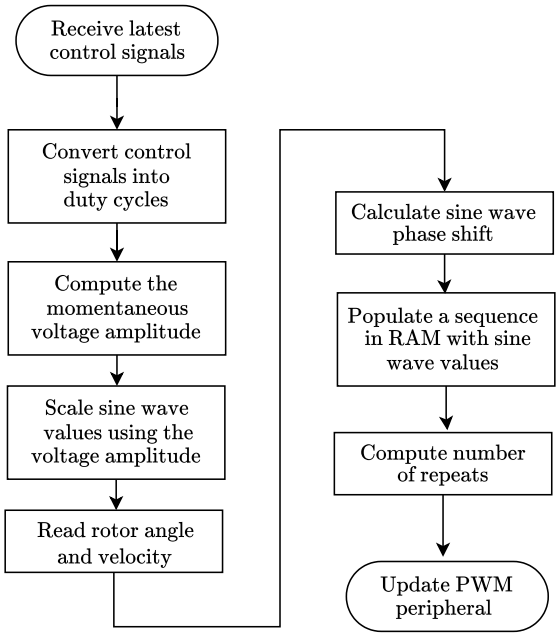


Figure 4.13: Strategy flow

Figure 4.13 illustrates the flow of the PWM module with the strategy. Before the strategy was implemented, all that was done in the PWM module was the conversion of the control signals into duty cycle values before updating the PWM peripheral.

4.3.2 Precomputed sine waves

As explained in section 4.3.1, it is enough to specify the sequence values of a period of a sine wave and playback the sequence until it is updated. The period of the outputted sine wave can be controlled by repeating the sequence values a specified number of times, as illustrated in figure 4.12, and the amplitude can be specified to approximately match the amplitude of the present output from the current controller, V_q as

$$A = \frac{|V_q|}{2 \cdot V_{max}} \tag{4.19}$$

which can be translated into the corresponding compare value, A_{COMP}, according to

$$A_{COMP} = A \cdot C_{top} \cdot DUTY_{max} \tag{4.20}$$

As explained in section 4.1.1, compare values are used to set a specific duty cycle value. The compare values, $sin_{COMP,i}$, that define the desired sine wave can be calculated according to

$$sin_{COMP,i} = \frac{C_{top}}{2} - A_{COMP} \cdot sin_{LUT,i}, \forall i \in (0, 1, ..., n-1) \tag{4.21}$$

where n is the sequence length, and sin_{LUT} is a lookup table calculated as

$$sin_{LUT,i} = sin(i \cdot \frac{2\pi}{n}), \forall i \in (0, 1, ..., n-1) \tag{4.22}$$

For example, figure 4.11a shows the resulting compare values $sin_{COMP,i}$ for $A_{COMP} = 100$ and $n = 24$.

To ensure that the FOC works as intended when the CPU is available, the output from the controller is placed at the very beginning of the sequence each time it is calculated, i.e. at the end of every PWM period. By placing the correct compare values, sin_{COMP}, directly after the output from the controller, the sequence will always consist of compare values defining a sine wave period. To ensure that the rotor keeps rotating with the same velocity during interrupts, no matter when they occur and their length, the compare values must have the proper amplitude and phase shift on each phase respectively. Subsequently, this means that the amplitude, A_{COMP}, and all compare values, sin_{COMP}, must also be calculated at the end of every PWM period, and inserted into the sequence with the correct phase shifts depending on the electrical angle (θ_e). Figure 4.14 shows the correct sequence for all three phases at an arbitrary angle θ_e, an amplitude of $A_{COMP} = 100$, and a sequence length of $n = 24$. When the CPU is available, the sequence will be updated at the end of each PWM period. Figure 4.15 further illustrates how the algorithm works. The figure shows the past output of the controller, the current time instance (dashed black line), and the future output of the sequence currently in the RAM buffer. If an interrupt occurs, the entire sequence will be repeated until the CPU is available again, meaning that the motor can be controlled open-loop indefinitely.

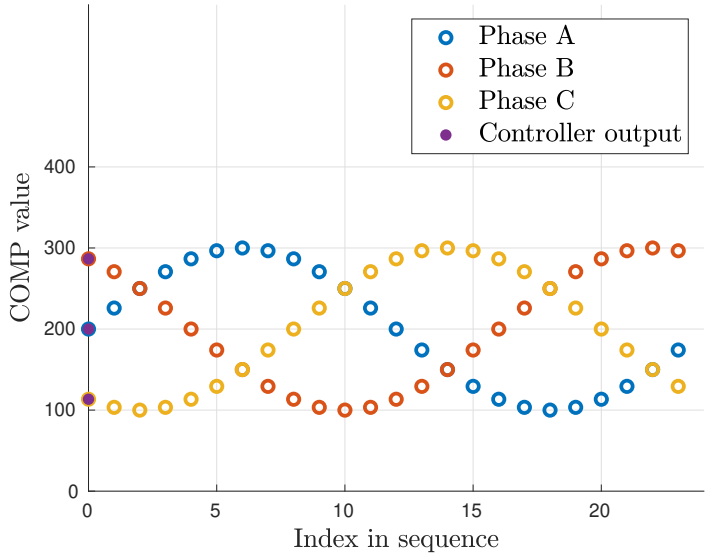


Figure 4.14: Sequence for all three phases at the time the PWM is updated

Figures 4.15 and 4.16 visualize how the strategy works in practice. The dashed vertical line in figure 4.15 illustrates the current time step. The waveforms before the line are of the previously outputted phase voltages and the dots after the line are the precomputed sequence stored in RAM. Figure 4.16 shows what voltages are outputted by the PWM peripheral whenever interrupts occur. The sequence freezes and is looped over creating discrete voltage steps for each phase.

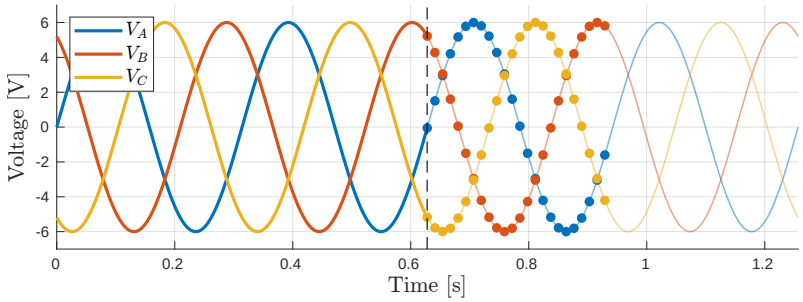


Figure 4.15: Visualization of previous voltages and the computed future voltages

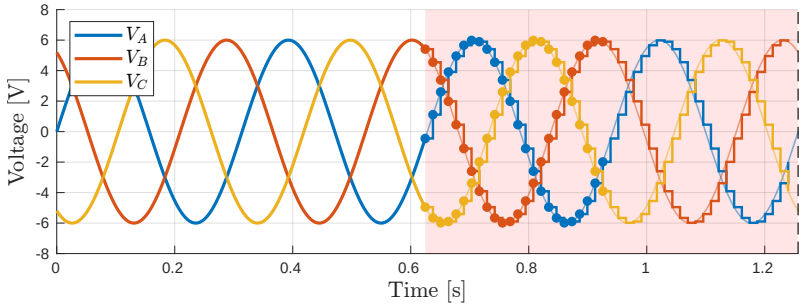


Figure 4.16: Visualization of the phase voltages during an interrupt

4.4 Code execution time

When implementing timing-critical firmware, it is often important to account for code execution time. Typically, the aim is to optimize code execution speed while maintaining acceptable accuracy. In some cases, it is also important to consider execution speed in contrast to ease of development, readability, and maintainability.

Arithmetic operations using different data types are typically not equally time-consuming and it is said that integer arithmetic is generally faster than floating-point arithmetic [26]. The trade-off when using integer arithmetic, however, is that numbers can not be represented with decimals. In order to get fine resolution on, for example, current readings, you would have to represent the currents in milli- or micro Ampere in the code. This should not be an issue in itself, though difficulties may arise when using trigonometric functions which typically have values in the range between -1 and 1. A popular method to handle such difficulties is called fixed-point arithmetic, which is a technique that utilizes integers to store representations of decimal values. There exist libraries that use this method, which can be used to simplify firmware development, e.g. "libfixmath" [27]. However, the nRF52832 has a built-in floating-point unit (FPU) which means that it has hardware support for floating-point arithmetic (numbers with decimals). If the microprocessor does not have an FPU it would have to emulate floating-point numbers (floats) to use them, otherwise, it is only possible to do integer arithmetic. Because of this, a few rudimentary tests were performed to investigate the differences in code execution time between integer and floating-point arithmetic.

The first tests consisted of performing addition, subtraction, multiplication, and division using integer arithmetic, software-emulated floating-point (soft float) arithmetic, and hardware-accelerated floating-point (hard float) arithmetic. The tests consisted of 10 operations that were looped 10000 times, resulting in a total of 100000 operations. The results from these tests are displayed in table 4.2. Note that the execution times are approximate measurements and that their purpose is to give some indication of the speed differences when choosing data types for the variables that were involved in most of the computations.

Table 4.2: Arithmetic operations execution time

	Execution time (ms)		
Operation	Integer	Hard float	Soft float
Addition	7.8125	9.375	141.09
Subtraction	7.8125	9.375	147.34
Multiplication	7.8125	9.375	108.28
Division	9.3751	29.688	97.362

From table 4.2 it is clear that the operations using integers is faster than floats. However, hard floats are only 1.2 times slower for addition, subtraction, and multiplication and 3.17 times slower for division. Soft floats on the other hand are

significantly slower. Considering these results, hard floats were deemed fast enough compared to integers and therefore used in the firmware. This meant that it was possible to represent variables for e.g. currents and voltages using SI-units without prefixes, which simplified development and readability of the code.

Another test of importance was to compare the time to compute the sine of an angle, which is used in the current control update. For this test, a comparison was made between utilizing the $sinf$-function from the standard C library (`math.h`) and a custom-made lookup table (LUT). The test was performed similarly to the artihmetic operations test, meaning 10 sine operations were performed in a loop with 10000 iterations resulting in a total of 100000 operations. The results from this test are shown in table 4.3.

Table 4.3: Sine computation execution times

	Execution time (ms)	
Operation	`math.h` function	LUT
Sine	168.91	54.531

Comparing the two methods of computing the sine of an angle it can be seen that the LUT method is about 3.1 times faster than using the $sinf$-function from the standard C library.

The results presented in tables 4.2 and 4.3 were taken into consideration when designing the firmware to make the code as fast as possible. The final version of the firmware was tested for execution time where it was found that one FOC update takes just under $14\mu s$ to perform without the strategy. One FOC update with the strategy takes approximately $32\mu s$. The total time from starting the current measurements to outputting new control signals is therefore around $24\mu s$ without the strategy and approximately $42\mu s$ with the strategy. This is a crucial aspect of the firmware since the control loop is run at 20 kHz, which means that new control signals are computed every $50\mu s$. Since the execution time of the FOC firmware is less than the periodicity of the loop, all computations complete in time and the most relevant control is guaranteed. If the execution time would be slower, the system would get out of sync and the control would not be as efficient.

5
Results

This chapter presents how the motor control strategy presented in this thesis performs under different circumstances. For all scenarios, the two FOC controllers and the cascaded velocity controller were active.

Data for each scenario was captured on a RIGOL MSO5074 oscilloscope with sensors that were attached to the motor. Micsig CP2100B current probes were clamped over the motor phase wires and another AMT-102V (with 384 PPR) was attached to the rotor. The oscilloscope captured the raw encoder pulses, pulse indications for when interrupts happened, pulse indications for when the rotor angle was 0 radians and the phase currents. Captured data was transferred from the oscilloscope to a computer where it was processed and condensed into the figures shown in this chapter. Using this setup, it was possible to acquire external measurements of the entire system.

All scenarios in the following sections benchmark the rotor's angular velocity, motor phase currents and corresponding dq-currents, the electrically produced torque and the magnetic field phase shift between the stator and rotor. The scenarios presented in sections 5.1, 5.2 and 5.3 were performed without a load attached to the rotor. The reason for this was mainly due to the somewhat weak electronics ($< 1.4A$ RMS) and the fact that it was more convenient to run the motor without a load attached during development. Appendix B contains more results, mostly of the same scenarios as in chapter 5 but for a lower velocity (20 rad/s). There are also results for how the motor performs when a small load is attached to the rotor under the same scenarios as in 5.2.

5.1 Uninterrupted motor control

This section demonstrates the motor control performance under uninterrupted code execution. The results in this section can be viewed as a baseline for how the motor should behave if nothing interrupts or disturbs it.

5.1.1 Constant velocity

Figures 5.1, 5.2, and 5.3 show the motor's performance when it was run uninterrupted at a constant velocity of 100 rad/s.

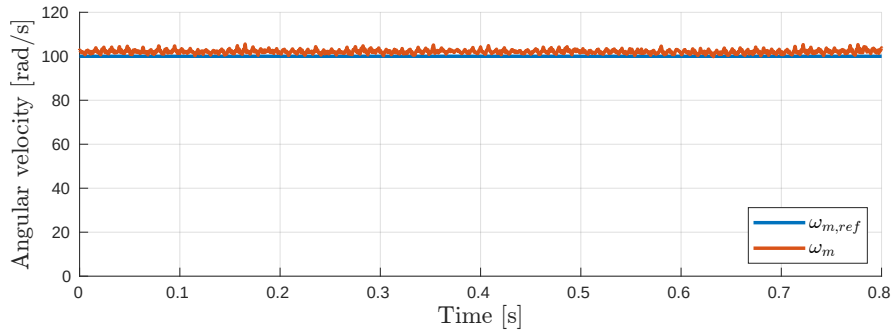


Figure 5.1: Rotor angular velocity, 100 rad/s uninterrupted.

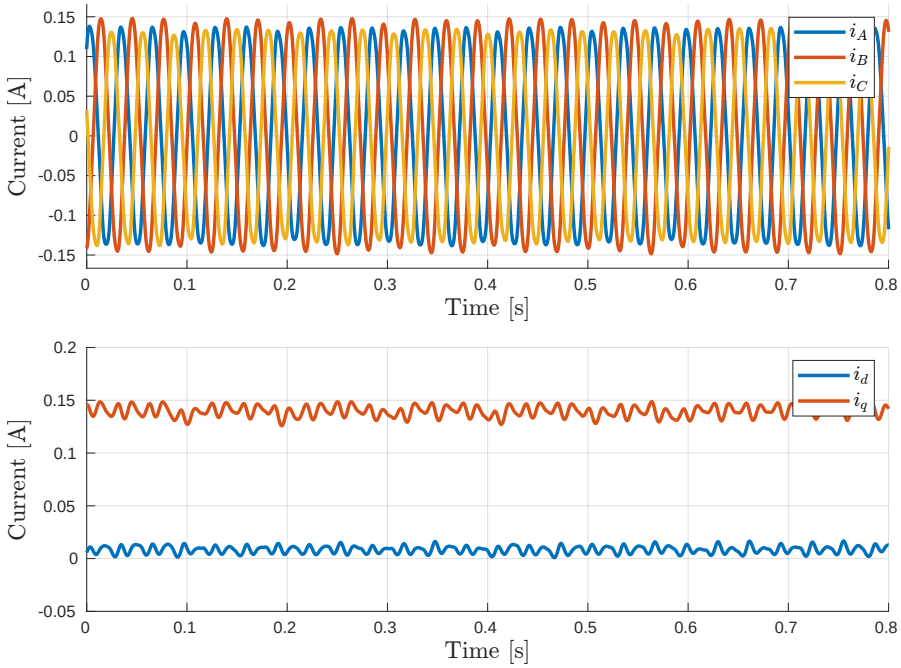


Figure 5.2: Motor phase currents and corresponding dq-currents, 100 rad/s uninterrupted.

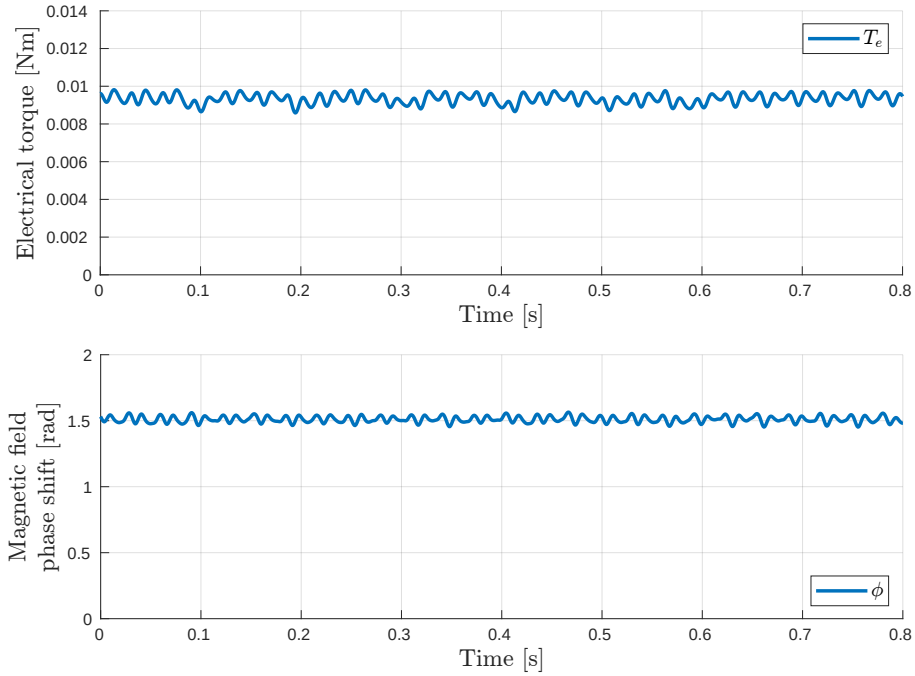


Figure 5.3: Electrical torque and magnetic field phase shift, 100 rad/s uninterrupted.

5.2 Motor control under 100 ms interrupts

This section presents the motor control performance under 100 ms interrupts, with and without the strategy. The light red regions in the figures symbolize interrupts, which means no code is executed during this time.

5.2.1 Constant velocity without strategy

Figures 5.4, 5.5, and 5.6 show the motor's performance during 100 ms interrupts when it was run at a constant velocity of 100 rad/s without the strategy.

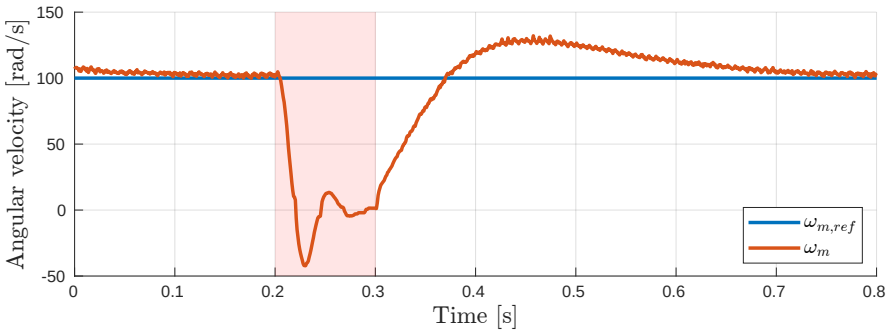


Figure 5.4: Rotor angular velocity, 100 rad/s, 100 ms interrupt, without strategy

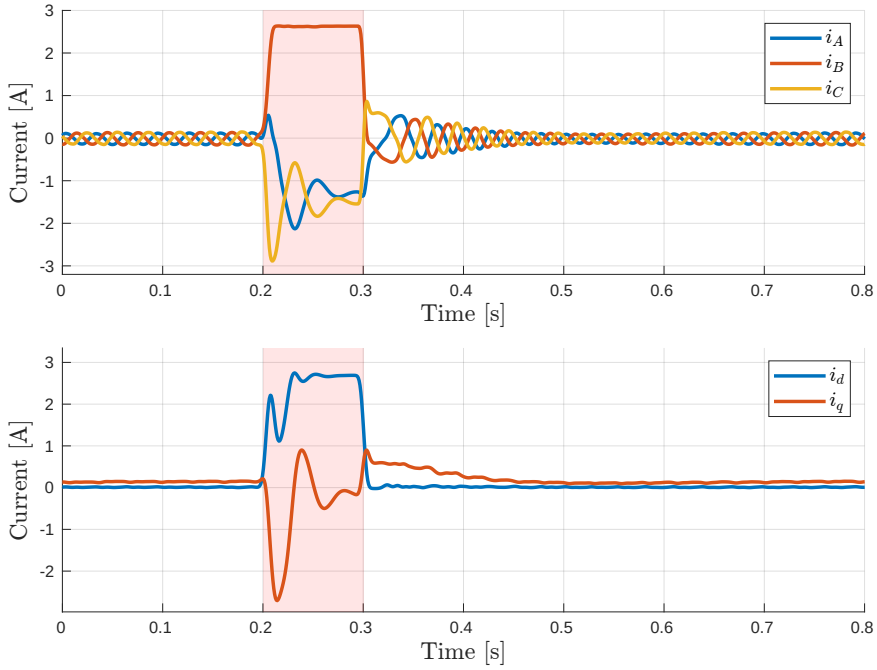


Figure 5.5: Motor phase currents and corresponding dq-currents, 100 rad/s, 100 ms interrupt, without strategy.

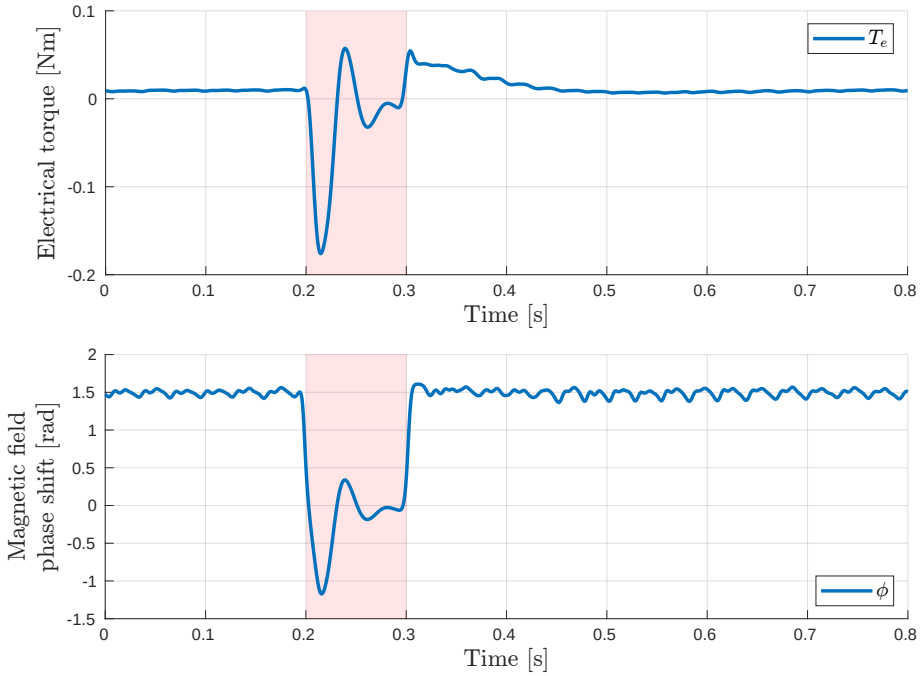


Figure 5.6: Electrical torque and magnetic field phase shift, 100 rad/s, 100 ms interrupt, without strategy.

5.2.2 Constant velocity with strategy

Figures 5.7, 5.8, and 5.9 show the motor's performance during 100 ms interrupts when it was run at a constant velocity of 100 rad/s with the strategy.

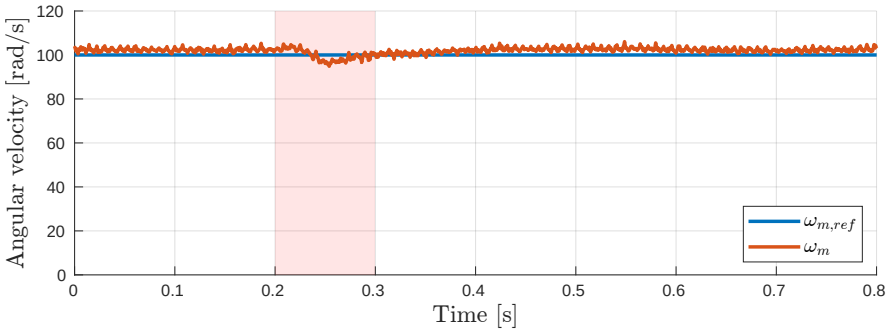


Figure 5.7: Rotor angular velocity, 100 rad/s, 100 ms interrupt, with strategy.

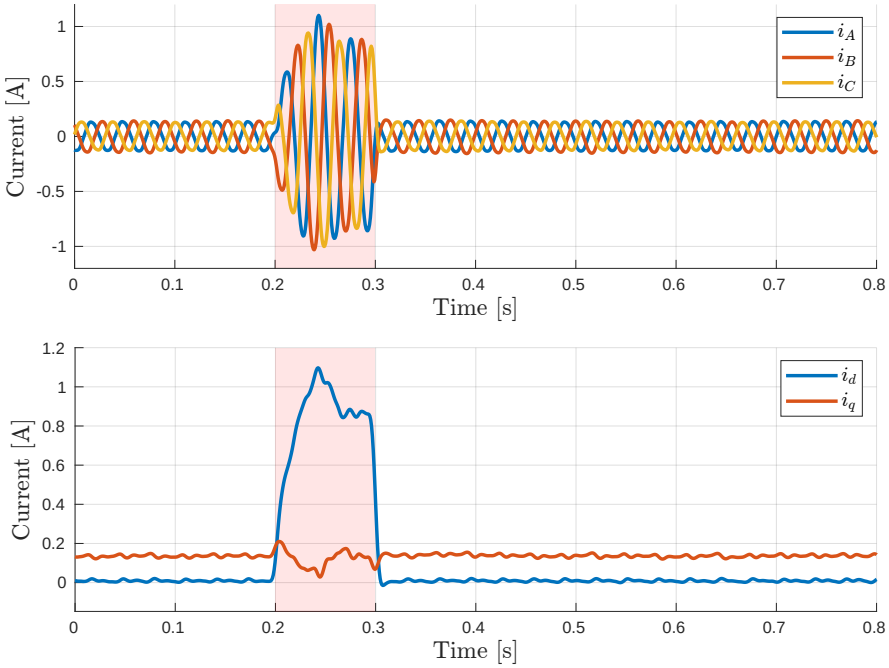


Figure 5.8: Motor phase currents and corresponding dq-currents, 100 rad/s, 100 ms interrupt, with strategy

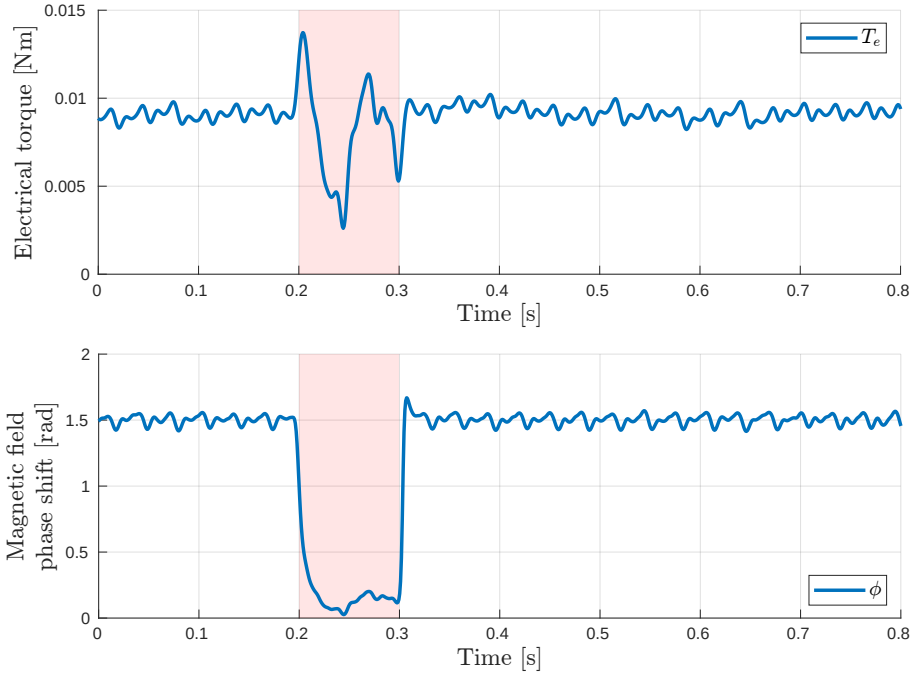


Figure 5.9: Electrical torque and magnetic field phase shift, 100 rad/s, 100 ms interrupt, with strategy.

5.2.3 Acceleration and deceleration with strategy

Figures 5.10, 5.11, and 5.12 show the motor's performance during 100 ms interrupts in an acceleration and a deceleration phase with the strategy. The motor was accelerated from 0 rad/s to 100 rad/s in 2 seconds, kept at a constant 100 rad/s for 1 second and then decelerated from 100 rad/s to 0 rad/s in 2 seconds.

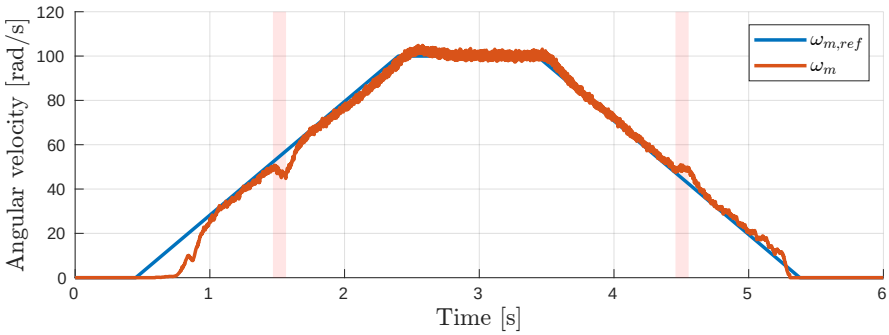


Figure 5.10: Rotor angular velocity, 0-100-0 rad/s, 100 ms interrupts, with strategy.

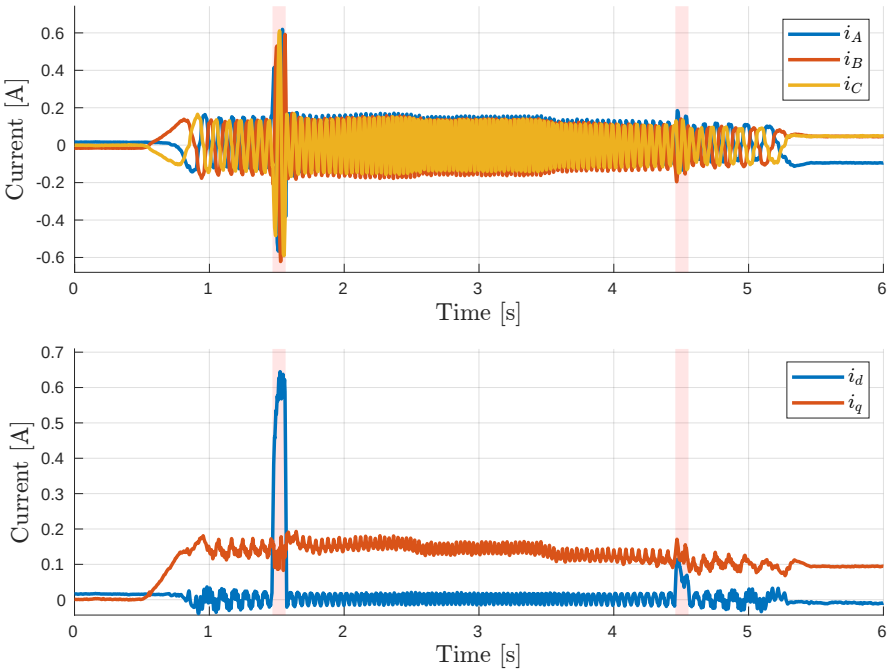


Figure 5.11: Motor phase currents and corresponding dq-currents, 0-100-0 rad/s, 100 ms interrupts, with strategy.

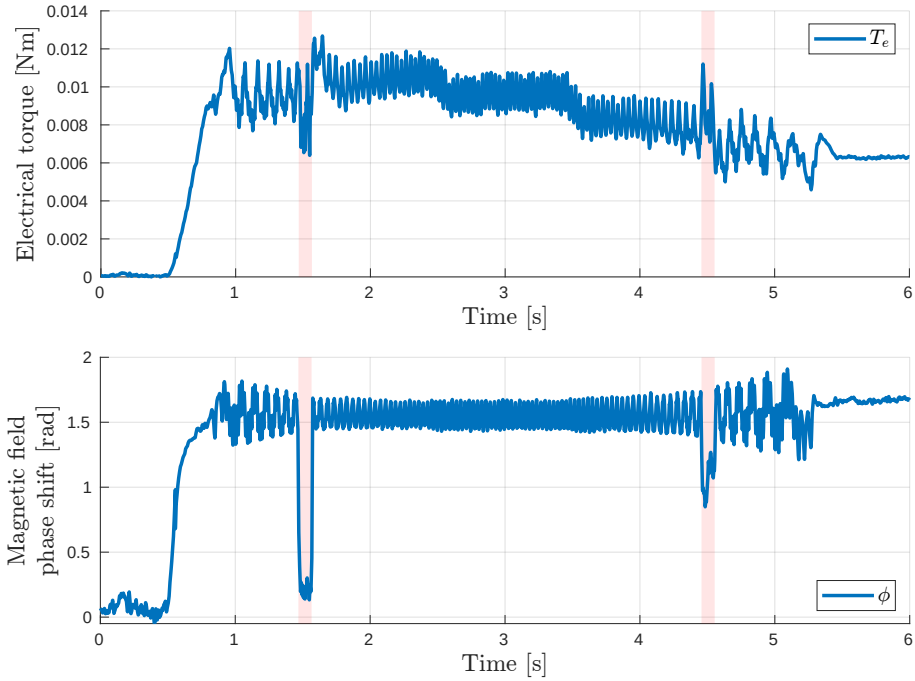


Figure 5.12: Electrical torque and magnetic field phase shift, 0-100-0 rad/s, 100 ms interrupts, with strategy.

5.3 Motor control under other interrupt scenarios

This section demonstrates how the motor control strategy performs under a few more interrupt scenarios, highlighting its capabilities.

5.3.1 Low interrupt frequency during constant velocity

This section demonstrates how the motor control performs with the strategy in a scenario with shorter and longer interrupts occurring at a relatively low frequency. Figures 5.13, 5.14, and 5.15 show the motor's performance in the low interrupt frequency scenario when it was run at a constant velocity of 100 rad/s with the strategy.

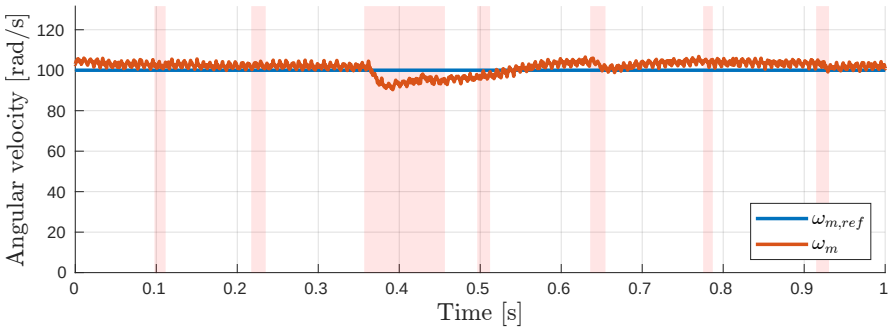


Figure 5.13: Rotor angular velocity, 100 rad/s, low interrupt frequency, with strategy.

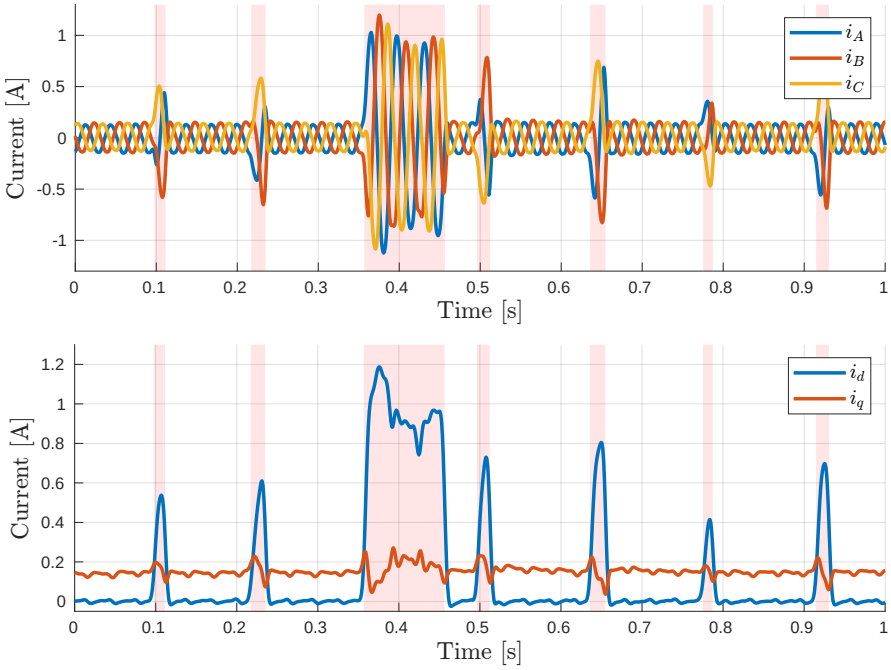


Figure 5.14: Motor phase currents and corresponding dq-currents, 100 rad/s, low interrupt frequency, with strategy.

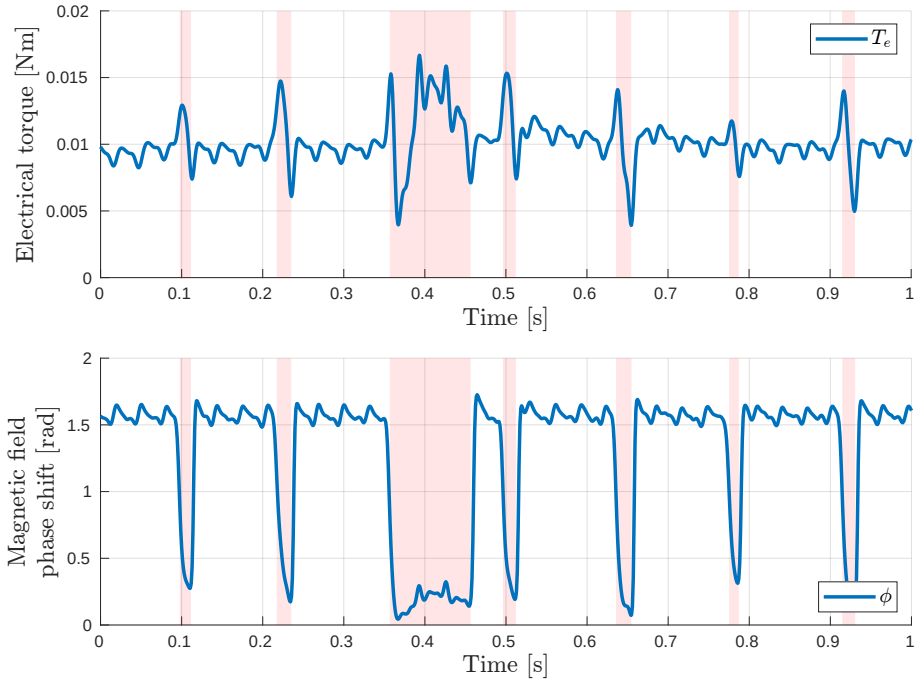


Figure 5.15: Electrical torque and magnetic field phase shift, 100 rad/s, low interrupt frequency, with strategy.

5.3.2 High interrupt frequency during constant velocity

This section presents results for how the motor control behaves with the strategy in a more extreme interrupt scenario which is never seen during normal context for the microcontroller's application. The purpose of this scenario, however, is to show how the motor control handles worse scenarios than are expected to occur.

Figures 5.16, 5.17, and 5.18 show the motor's performance in the high interrupt frequency scenario when it was run at a constant velocity of 20 rad/s with the strategy.

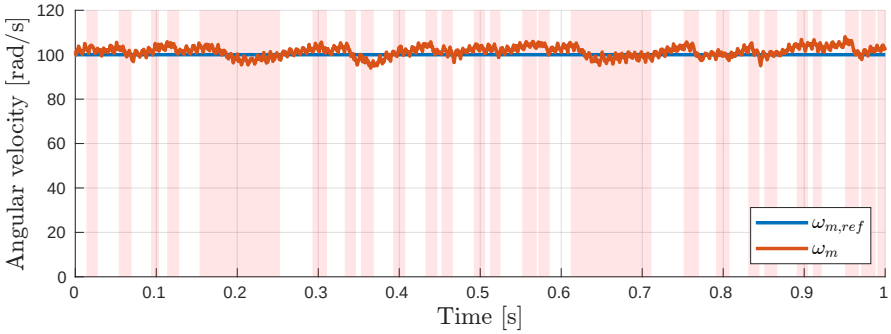


Figure 5.16: Rotor angular velocity, 100 rad/s, high interrupt frequency, with strategy.

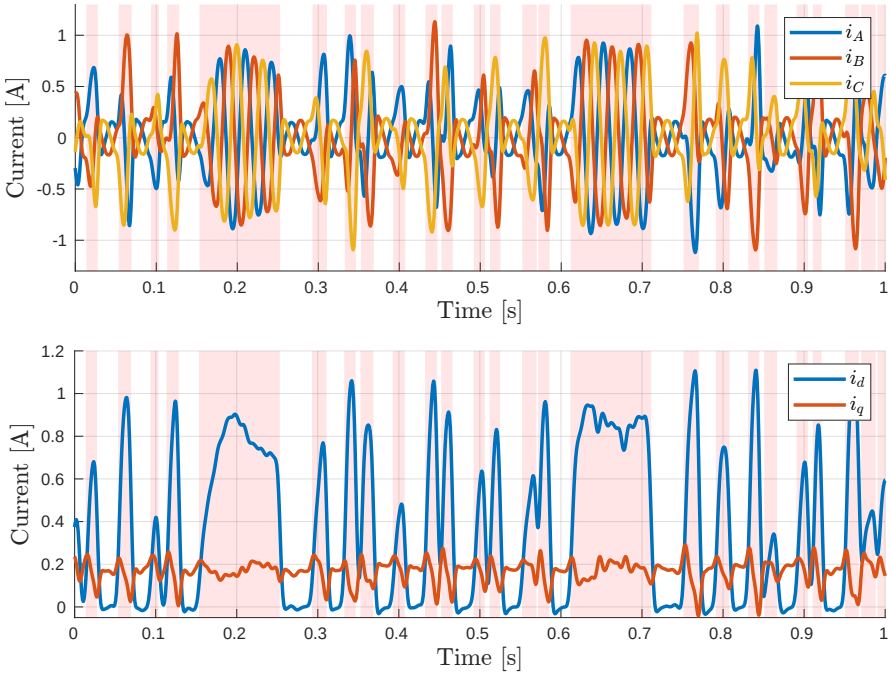


Figure 5.17: Motor phase currents and corresponding dq-currents, 100 rad/s, high interrupt frequency, with strategy.

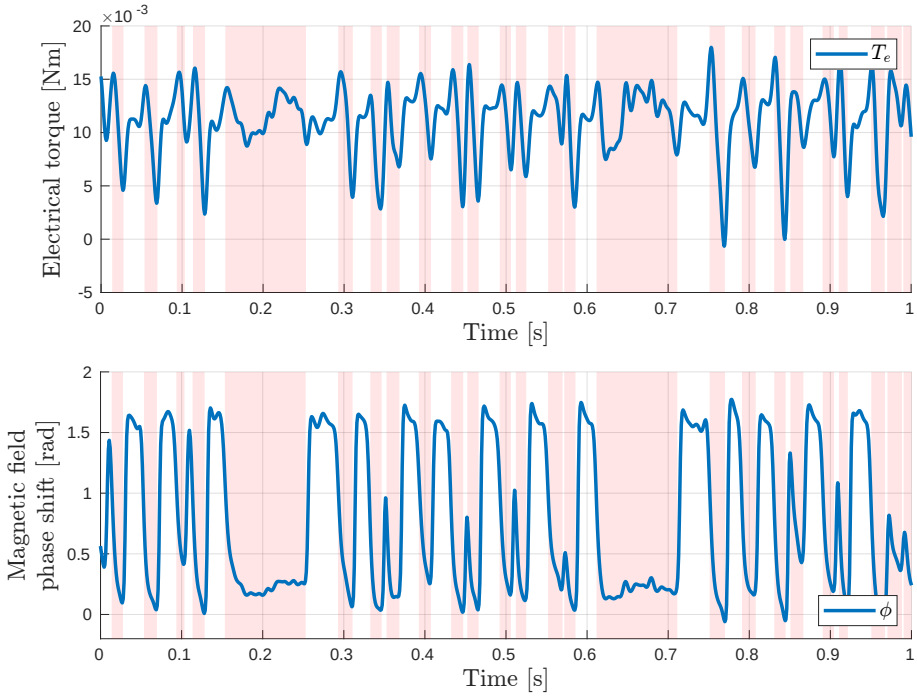


Figure 5.18: Electrical torque and magnetic field phase shift, 100 rad/s, high interrupt frequency, with strategy.

5.3.3 Infinite interrupt during constant velocity

This section presents results for what happens if the motor control is interrupted and never allowed to start running again.

Figures 5.19, 5.20, and 5.21 show the motor's performance during an infinitely long interrupt when it was run at a constant velocity of 100 rad/s with the strategy.

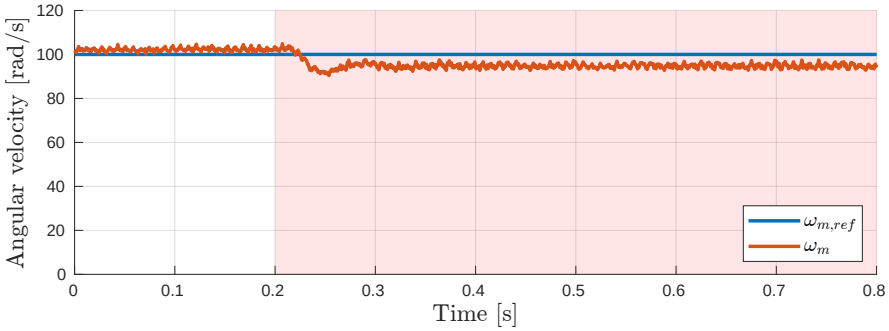


Figure 5.19: Rotor angular velocity, 100 rad/s, infinite interrupt, with strategy.

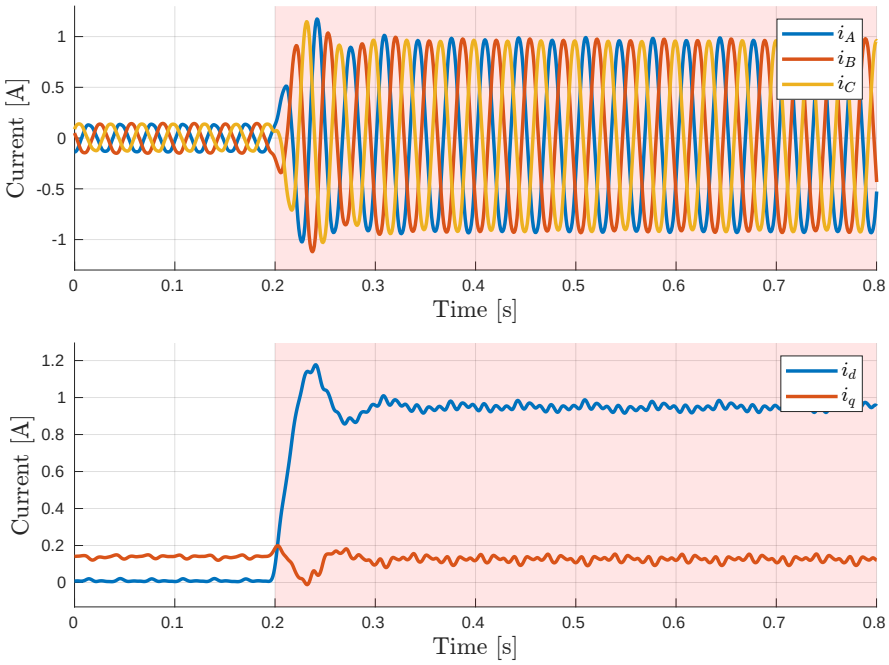


Figure 5.20: Motor phase currents and corresponding dq-currents, 100 rad/s, infinite interrupt, with strategy.

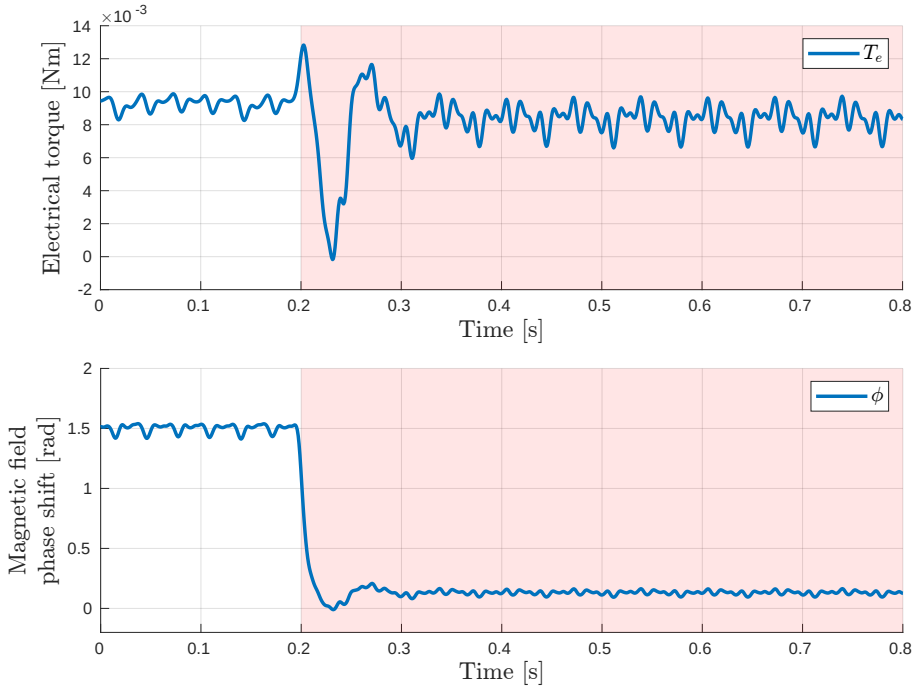


Figure 5.21: Electrical torque and magnetic field phase shift, 100 rad/s, infinite interrupt, with strategy.

6
Discussion and future work

Presented in this chapter is a discussion of the results presented in chapter 5.

The results shown in section 5.1 are the system output when there are no interrupts. As can be seen in figure 5.1, the externally measured angular velocity follows the reference well. The reason why the measured velocity is slightly higher than the reference is that the timer of the SoC on which the motor control is run is marginally faster than expected. This means that from the controller's perspective, the reference is followed precisely even though there is a minor error in practice. Figure 5.2 shows that the d-current hovers around 0 while the q-current oscillates lightly around a constant non-zero value. This is the essence of FOC since it means that nearly 100% of the total current input is translated into produced torque with minimal losses. This is further illustrated in figure 5.3, where it can be seen that the stator magnetic field leads the rotor magnetic field by close to 90° at all times. There is a minimal amount of torque ripples, and the motor rotates smoothly as a result. These results are regarded as the baseline for motor control, and the performance of the implemented control strategy with introduced interrupts will be compared to these results.

Section 5.2.1 shows the output when 100 ms interrupts are introduced when no strategy is used. As can be seen in figure 5.4, the angular velocity drops down to 0 rad/s as expected, since the phase voltage remains constant without the strategy, locking the rotor. In essence, this motivates why a strategy is needed to ensure that the rotor can keep rotating without CPU access. The system output for the same test case, but with the implemented strategy enabled is presented in section 5.2.2, where it can be seen that the actual velocity is able to follow the reference before, during, and after interrupts, albeit with slightly more variation during the interrupt. As shown in figure 5.8, the q-currents deviates somewhat from its constant value while the d-current is significantly increased during the interrupt. This is an effect of the rotor magnetic field aligning with the stator magnetic field, further illustrated in figure 5.9. This occurs since the stator magnetic field is rotated without feedback from the rotor magnetic field, causing torque ripples.

As the two magnetic fields align, the magnitude of the phase currents is significantly increased, even though the magnitude of the phase voltages remains constant during interrupts. This is clearly shown in figure 5.20, where the interrupt is infinitely long. 0.2s after the interrupt is introduced, the amplitude and frequency of the phase currents are approximately the same as before the interrupt, and the

magnetic field phase shift is roughly 60° indicating that the phase voltages do not change in amplitude nor frequency. However, as the magnetic field shift approaches 0° the amplitude of the phase currents is significantly increased, indicating that the increase is solely dependent on the angle between the rotor and stator magnetic fields. This may be an effect of what is known as magnetic saturation [28]. When the two magnetic fields align, the magnetization of the stator coil cores will be strengthened, which can saturate the material. Magnetic saturation of the core reduces the winding inductance and thus increases the phase currents for given phase voltages [29]. It is not proven in this thesis that this is the cause of the increased phase currents and needs to be further investigated.

Figure 5.2.3 shows that the velocity during interrupts differs from the velocity just before the interrupt as intended. Instead, it decreases slightly during acceleration and increases slightly during deceleration, a result of how the QDEC was implemented. As explained in 4.1.2, the angular velocity is computed according to (4.5), where ΔP is the accumulated valid transitions in the latest 200 QDEC samples. Dividing ΔP with the time since the last update means that the accumulated transitions are evened out over that time to reduce measurement noise. During acceleration, the measured velocity will thus be lower than the true velocity, and during deceleration the opposite holds. There is, thus, a trade-off between measurement noise and momentaneous accuracy. As a result, the implemented strategy decreases the angular velocity if an interrupt occurs during acceleration, and increases the velocity if it occurs during deceleration. These effects can be mitigated by estimating the angular velocity differently, for example by using shorter time spans to determine the rotor position change. Future work includes improved interrupt handling during acceleration and deceleration, such that the reference value is followed as opposed to keeping a constant velocity during interrupts.

Comparing figure 5.19, where the reference velocity was set to 100 rad/s, to the results in figure B.6, where the velocity was set to 20 rad/s, it can be seen that the measured velocity follows the reference better for the lower velocity reference. This is an effect of what is presented in section 4.3.1, namely that there is a minor rounding error between the theoretically correct number of repeats and the actual number of repeats for the higher of the two references. The theoretically correct number of repeats for the current angular velocity is rounded down to the nearest integer in the implemented strategy, resulting in the steady state error shown in figure 5.19. Note that both figures show the case where the interrupt is of infinite length, exaggerating the effect of the rounding error.

In summary, the torque production is relatively constant using closed-loop control, and the motor rotates silently. The results show that the transitions between closed- and open-loop control are reasonably seamless regarding angular velocity, but not electrical torque and power consumption. As described in section 2.3, an effect of the observed torque ripples is a slight increase in audible noise.

7
Conclusion

This thesis presents an implementation of a control strategy aimed at mitigating the effects of CPU interrupts during FOC of a 3-phase PMSM. More specifically, the objective of the implemented strategy was to ensure that the angular velocity is maintained during interrupts, independently of when interrupts occur and their longevity.

Without interrupts, the system's measured angular velocity closely followed the reference, demonstrating good closed-loop performance. FOC effectively minimized losses, resulting in smooth motor operation with minimal torque ripples. Introducing 100 ms interrupts without interrupt handling control caused the rotor to lock, highlighting the need for a strategy to keep the motor spinning. With the strategy implemented, the motor maintained its velocity during interrupts, although with increased torque ripples due to the alignment of rotor and stator magnetic fields.

The implementation of the interrupt handling strategy relies heavily on peripherals that run independently from the processor and without these peripherals, it would be much more difficult to perform the same type of PMSM control strategy.

In summary, closed-loop control resulted in stable torque production and silent motor operation. The implemented strategy provided seamless transitions between control modes in terms of angular velocity, regardless of when CPU interrupts occur and their longevity.

Bibliography

[1] T. M. Inc., "Field-oriented control of pmsm using quadrature encoder," Natick, Massachusetts, United States, 2024. [Online]. Available: https://se.mathworks.com/help/mcb/gs/foc-pmsm-using-quadrature-encoder.html

[2] A. Glumineau and J. De León-Morales, *Sensorless AC Electric Motor Control*. Springer Cham, 01 2015.

[3] S. Halder and X. Yang, "Extracting the maximum torque from a PMSM for a vehicular application by using different modulation techniques in the overmodulation region," Master's thesis, Chalmers University of Technology, 2019.

[4] S. Lee, T. Lemley, and G. Keohane, "A comparison study of the commutation methods for the three-phase permanent magnet brushless dc motor," in *Electrical Manufacturing Technical Conference 2009: Electrical Manufacturing and Coil Winding Expo, EMCWA 2009*, 2009, pp. 49–55.

[5] W. A. Salah, D. Ishak, and K. J. Hammadi, "Minimization of torque ripples in bldc motors due to phase commutation-a review," *Przeglad Elektrotechniczny*, vol. 87, no. 1, pp. 182–188, 2011.

[6] S. Basu, "Chapter 1 - general discussions on control systems," in *Plant Intelligent Automation and Digital Transformation*. Academic Press, 2023, vol. 1, pp. 1–56. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323902465000077

[7] N. Phung Quang and J.-A. Dittrich, *Vector Control of Three-Phase AC Machines - System Development in the Practice*. Springer Berlin, Heidelberg, 01 2015.

[8] S. K. Baral, "Closed loop control of pmsm motor : Field oriented control using hall sensors," Master's thesis, Linköping University, 2021.

[9] S. Ziegler, R. C. Woodward, H. H.-C. Iu, and L. J. Borle, "Current sensing techniques: A review," *IEEE Sensors Journal*, vol. 9, no. 4, pp. 354–376, 2009.

[10] J. Bridgmon and C. Andrews, "Current sensing for inline motor-control applications," Texas Instruments, Tech. Rep., 2016.

[11] SimpleFOC. (2020) Current sensing. [Online]. Available: https://docs.simplefoc.com/current_sense

[12] ——. (2020) In-line current sensing. [Online]. Available: https://docs.simplefoc.com/inline_current_sense

[13] A. Mehta, "Understand low-side vs. high-side current sensing," *Planet analog*, 2009.

[14] J. Oh, H. Bae, H. Jeong, K. Lee, and J.-H. Oh, "Bldc motor current control using filtered single dc link current based on adaptive extended kalman filter,"

in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2213–2218.
[15] M. Mesgenaw and I. Lara, "Differences between optical and magnetic incremental encoders," Texas Instruments, Tech. Rep., 2022.
[16] J. Smoot, "Capacitive, magnetic, and optical encoders – comparing the technologies." [Online]. Available: https://www.cuidevices.com/blog/capacitive-magnetic-and-optical-encoders-comparing-the-technologies
[17] *nRF52832 Product Specification*, Nordic Semiconductor, 12 2023, rev. 1.9.
[18] *Three-phase brushless DC motor driver expansion board based on L6230 for STM32 Nucleo*, STMicroelectronics, 11 2023, rev. 3.
[19] *RS Pro Brushless DC Motor datasheet, stock no: 536-6046*, RS Pro.
[20] *Modular incremental encoder datasheet, series AMT10*, CUI Devices, 5 2022.
[21] *STM32L4 - Timers*, STMicroelectronics, rev. 2.0.
[22] M. Marne, "Model Based Control of PMSM using Field Oriented Control," Master's thesis, Chalmers University of Technology, 2019.
[23] *User Manual - STM32F PMSM single/dual FOC SDK v. 4.3*, STMicroelectronics, 9 2016.
[24] B. Carter, "Designing gain and offset in thirty seconds," Texas Instruments, Tech. Rep., 2002.
[25] R. Helsing and T. Sanchez, "Modeling and control of a pmsm operating in low speeds," Master's thesis, Linköping University, 2022.
[26] J. B. Gosling, *Floating Point Operation*. New York, NY: Springer New York, 1980, pp. 74–104. [Online]. Available: https://doi.org/10.1007/978-1-4757-4938-0_6
[27] B. Brewer, "libfixmath." [Online]. Available: https://code.google.com/archive/p/libfixmath/
[28] T. Lubin, H. Razik, and A. Rezzoug, "Magnetic saturation effects on the control of a synchronous reluctance machine," *IEEE Transactions on Energy Conversion*, vol. 17, no. 3, pp. 356–362, 2002.
[29] W. Dafang, Q. Ji, Z. Cheng, L. Jiangmin, and Y. Yechen, "Strategy of starting sensorless bldcm with inductance method and emf integration," *Mathematical Problems in Engineering*, Nov 2013. [Online]. Available: https://doi.org/10.1155/2013/146058

A

Effects of stator voltage decoupling

Figure A.1 shows the results of using stator voltage decoupling, and figure A.2 shows the results when not used. More specifically, what is shown in the figures is the step response of the system when the rotor is stalled and the i_q-reference is changed from 0 to 1 A. Stator voltage decoupling is explained in more detail in section 2.4.2. Comparing the two figures, it is clear that the d- and q-currents follow their respective reference better when stator voltage decoupling is used. Furthermore, the angle between the stator and rotor magnetic field is closer to 90° when stator voltage decoupling is used, suggesting that using stator voltage decoupling is more efficient in terms of energy consumption.

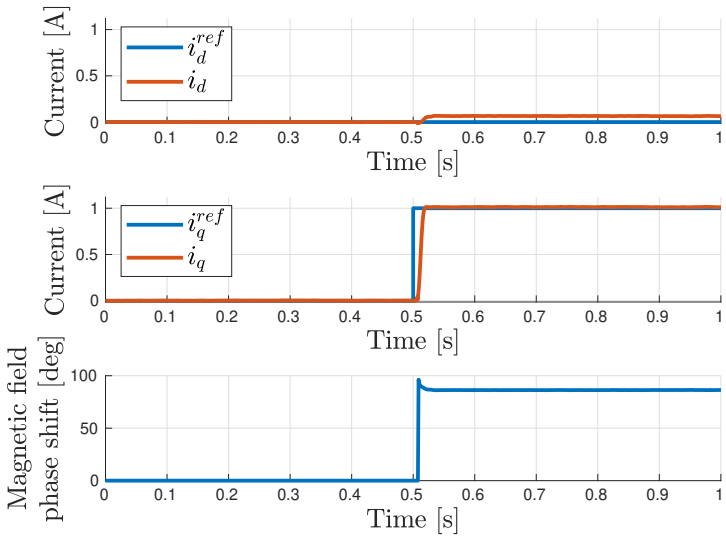


Figure A.1: Step response when the rotor is stalled, with stator voltage decoupling

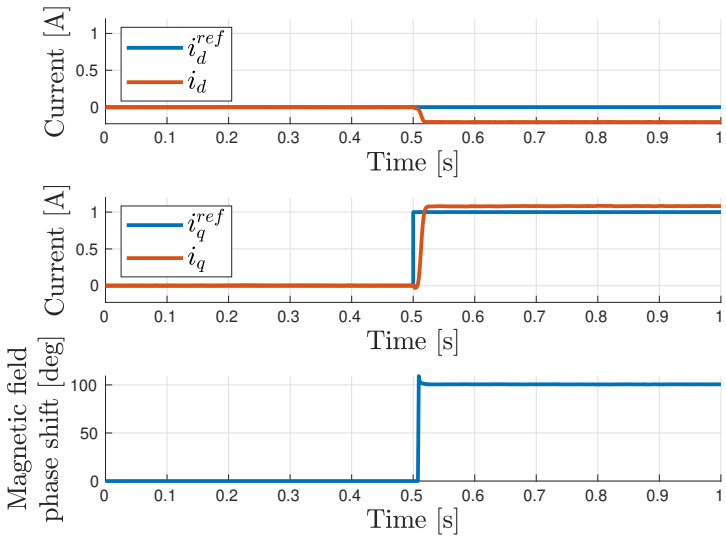


Figure A.2: Step response when the rotor is stalled, without stator voltage decoupling

B

Additional results

Section B.1 contains additional results from principally the same scenarios as in chapter 5, but using a lower velocity (20 rad/s). The plots are condensed into fewer figures to save paper, but their explanations are similar to the ones in chapter 5. Section B.2 contains results from attaching some load to the rotor. The exact load was not determined but it was enough to make the motor draw more current than without a load attached. The purpose of those scenarios was to investigate if the motor control strategy would perform differently with a load attached than without a load attached.

B.1 Constant lower velocity under various operating scenarios

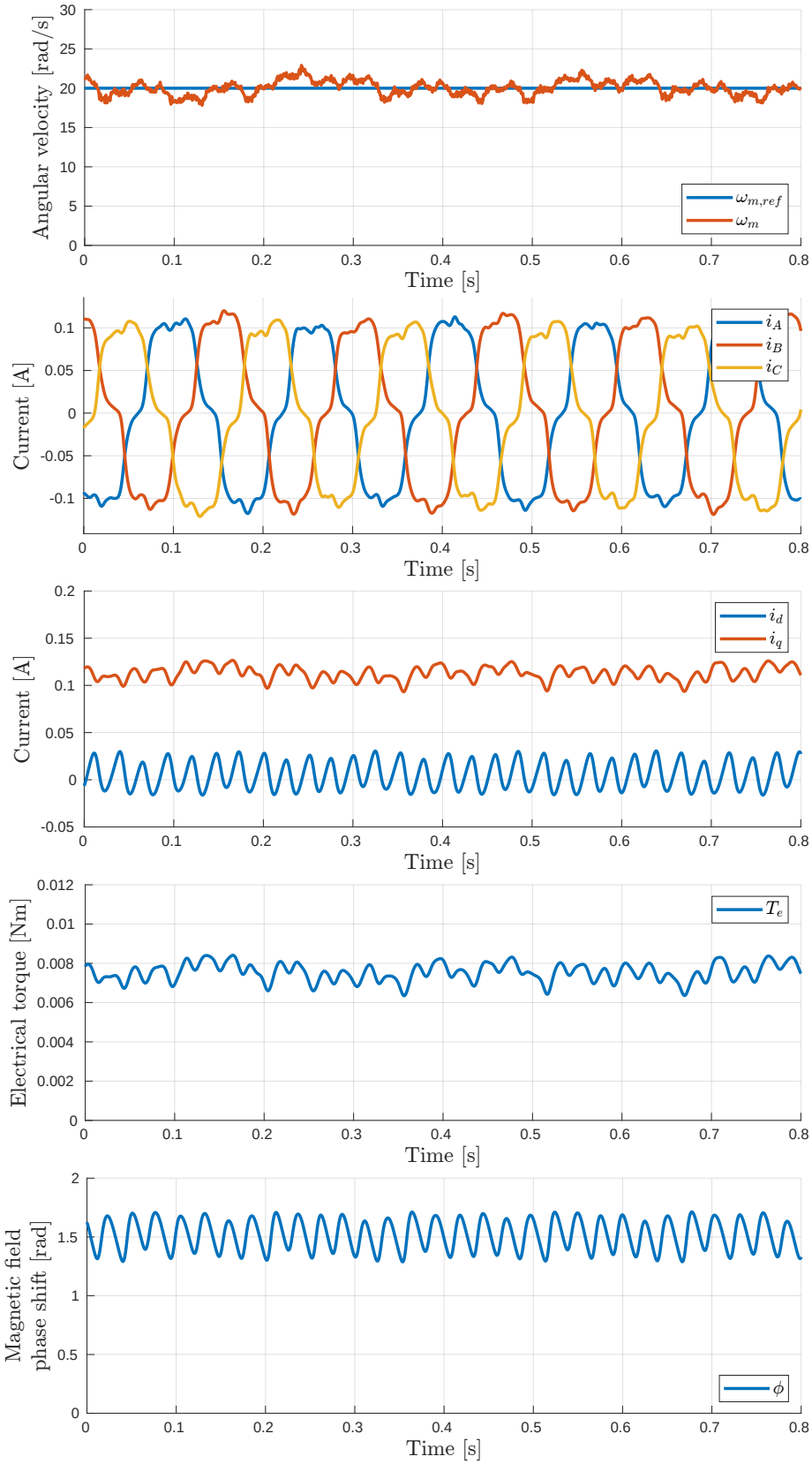


Figure B.1: 20 rad/s uninterrupted

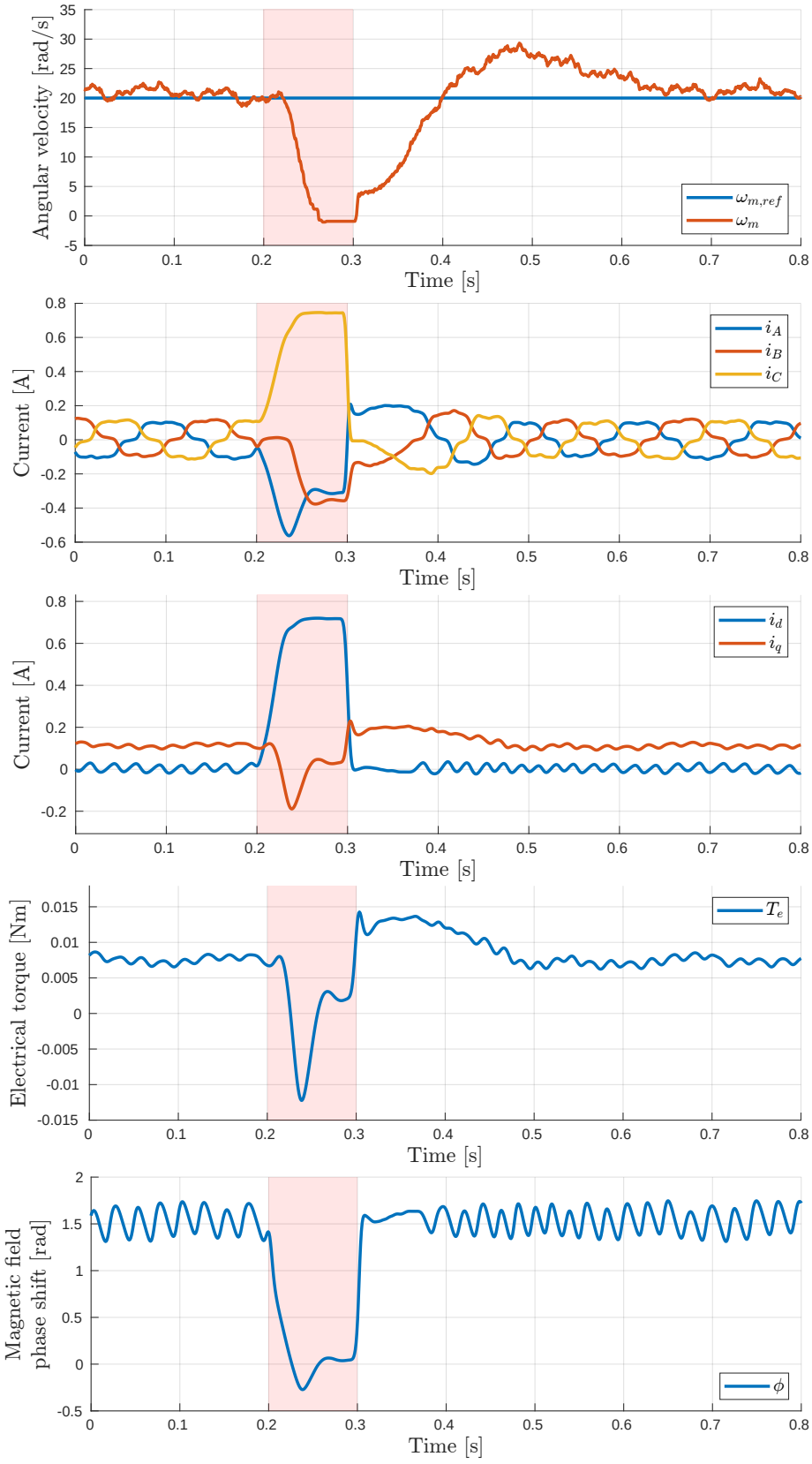


Figure B.2: 20 rad/s, 100 ms interrupt, without strategy

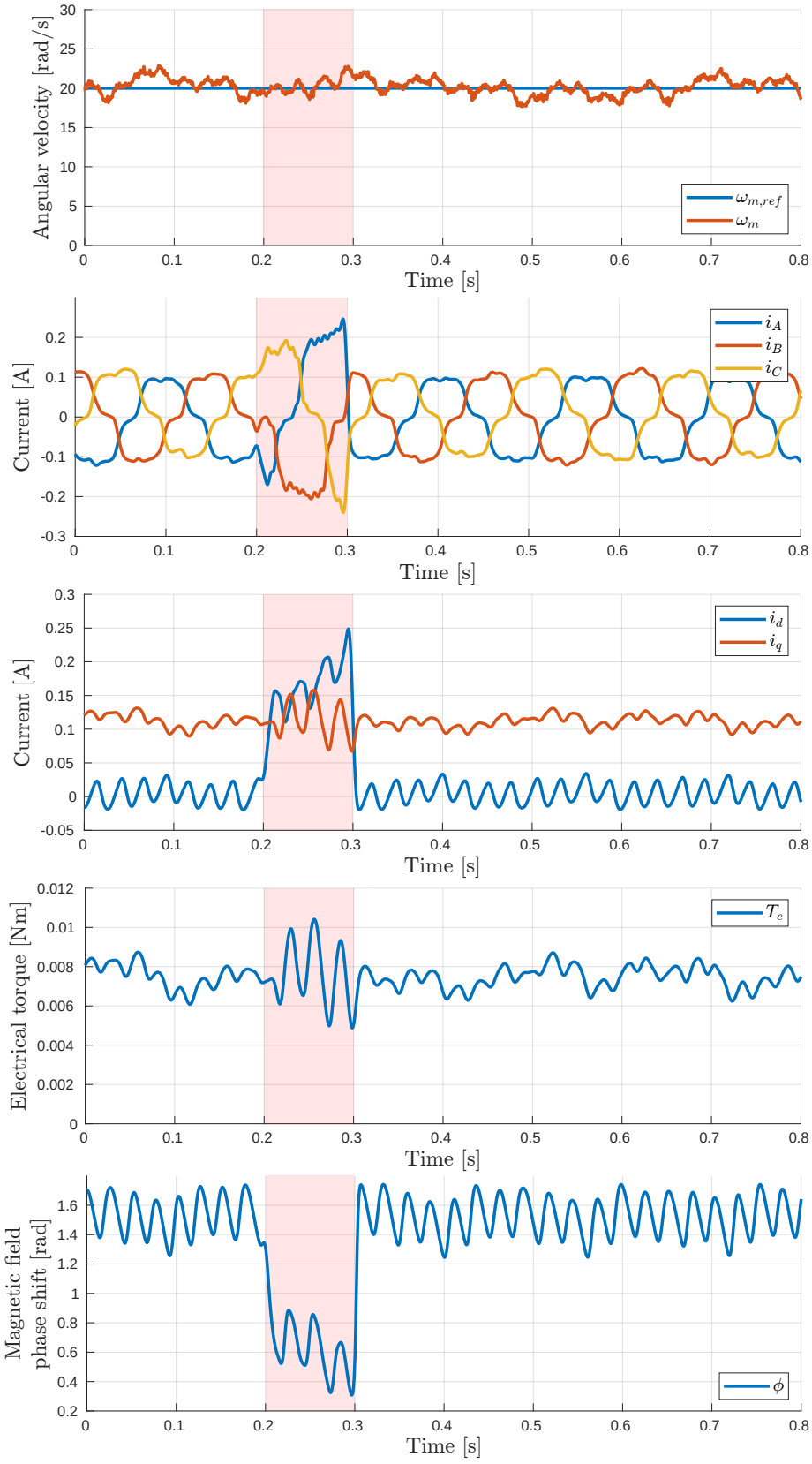


Figure B.3: 20 rad/s, 100 ms interrupt, with strategy

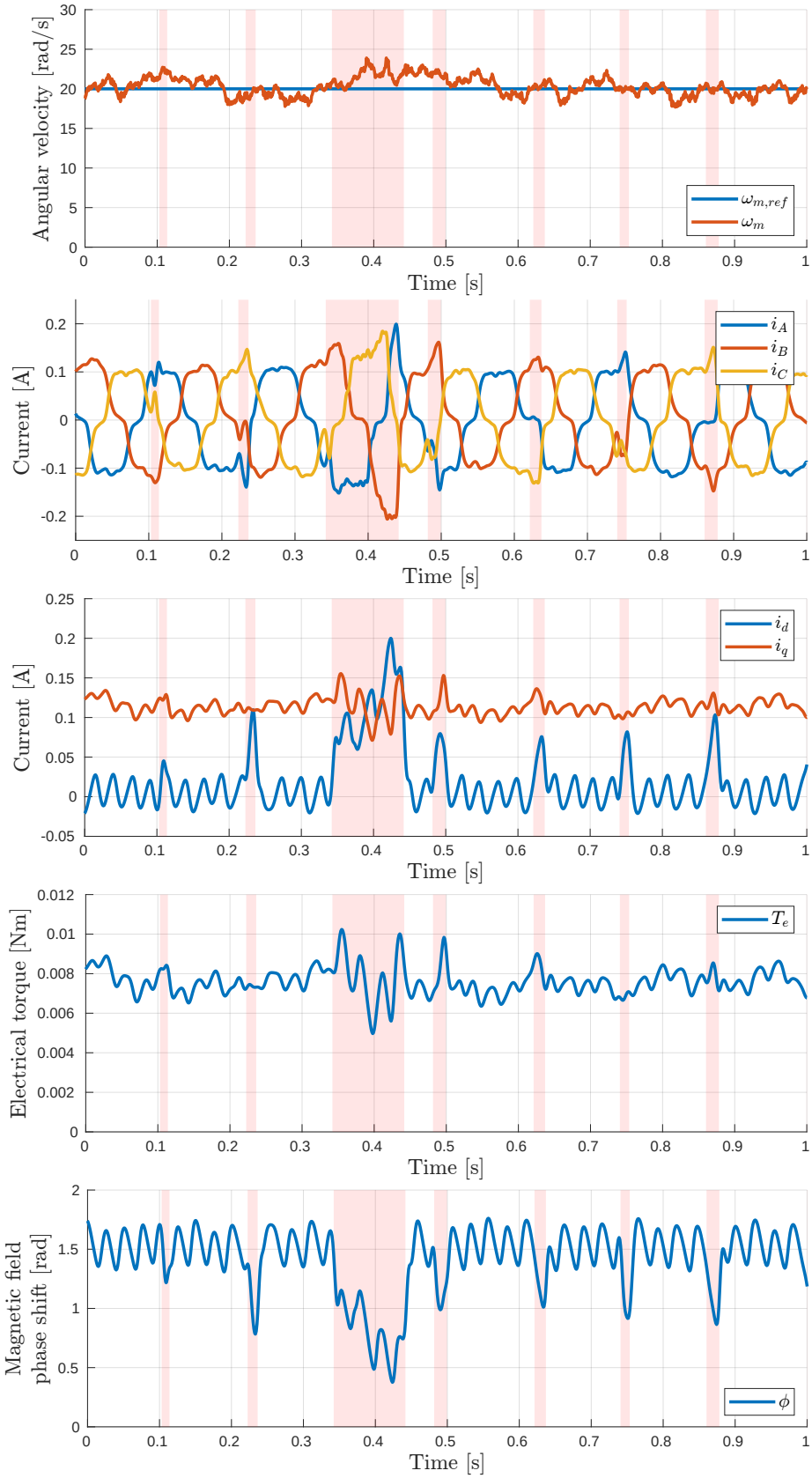


Figure B.4: 20 rad/s, low interrupt frequency, with strategy

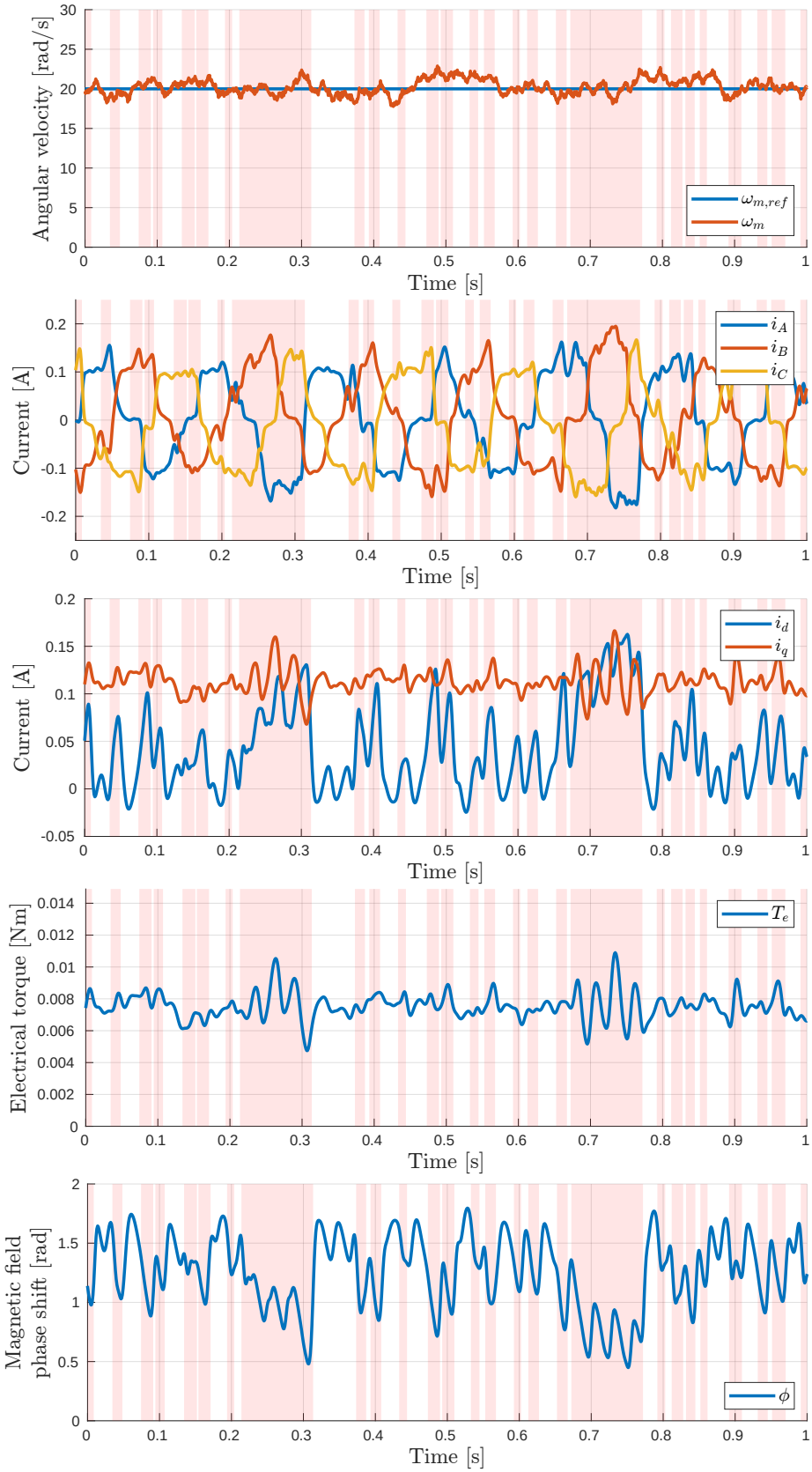


Figure B.5: 20 rad/s, high interrupt frequency, with strategy

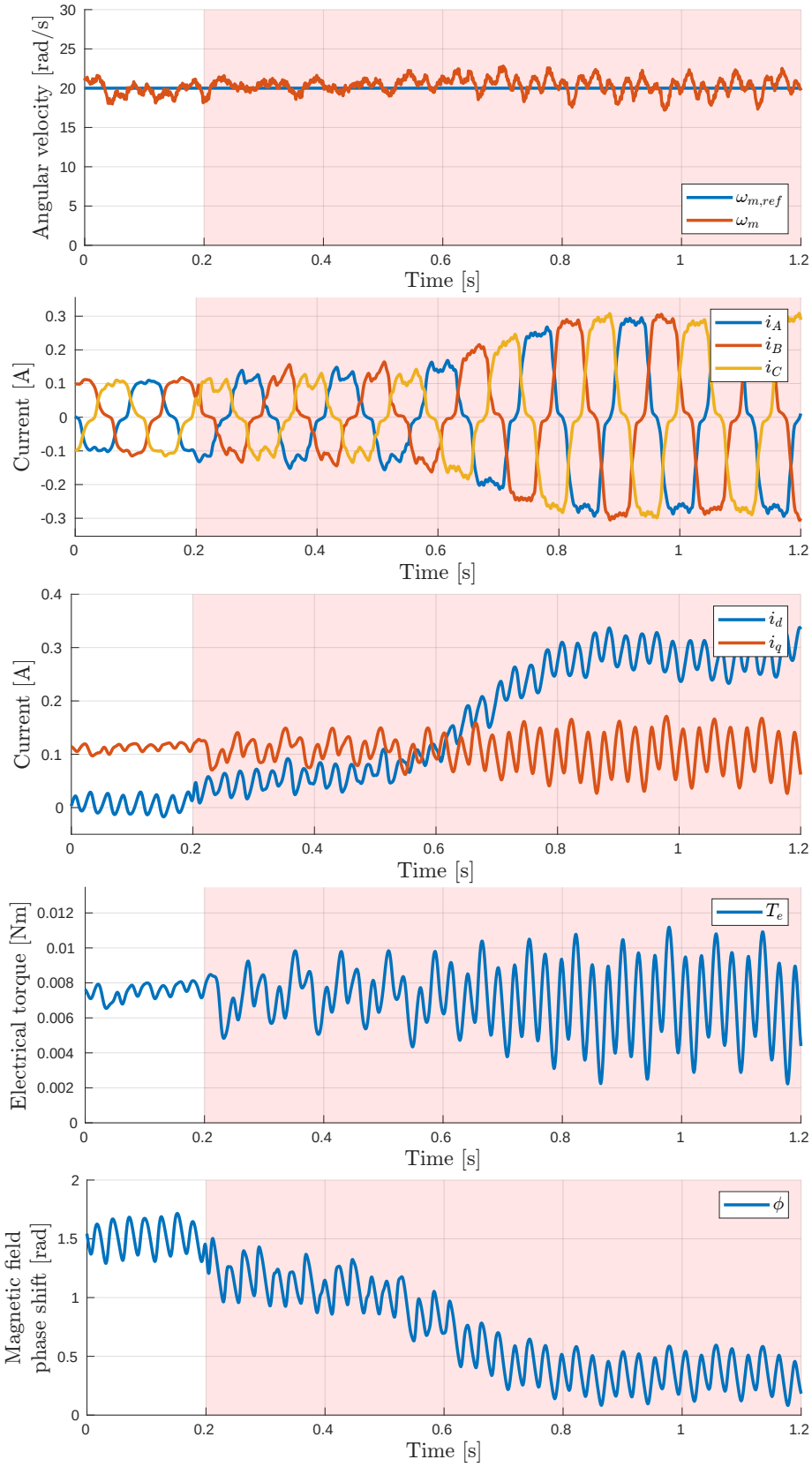


Figure B.6: 20 rad/s, infinite interrupt, with strategy

B.2 Motor control under 100 ms interrupts with external load

This section provides results for how the motor behaves when it has a load attached to the rotor. The purpose of these results is to show how the motor behaves with an attached load during interrupts with and without the strategy.

B.2.1 Constant velocity without strategy

Figures B.7, B.8, and B.9 show the motor's performance during 100 ms interrupts when it was run at a constant velocity of 20 rad/s without the strategy.

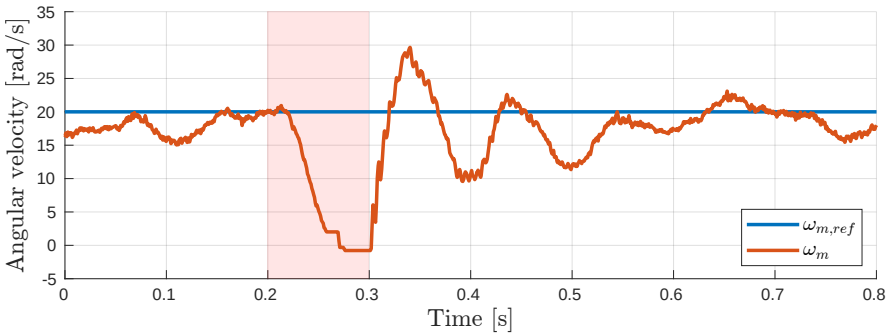


Figure B.7: Rotor angular velocity, 20 rad/s, 100 ms interrupt, without strategy, with load attached

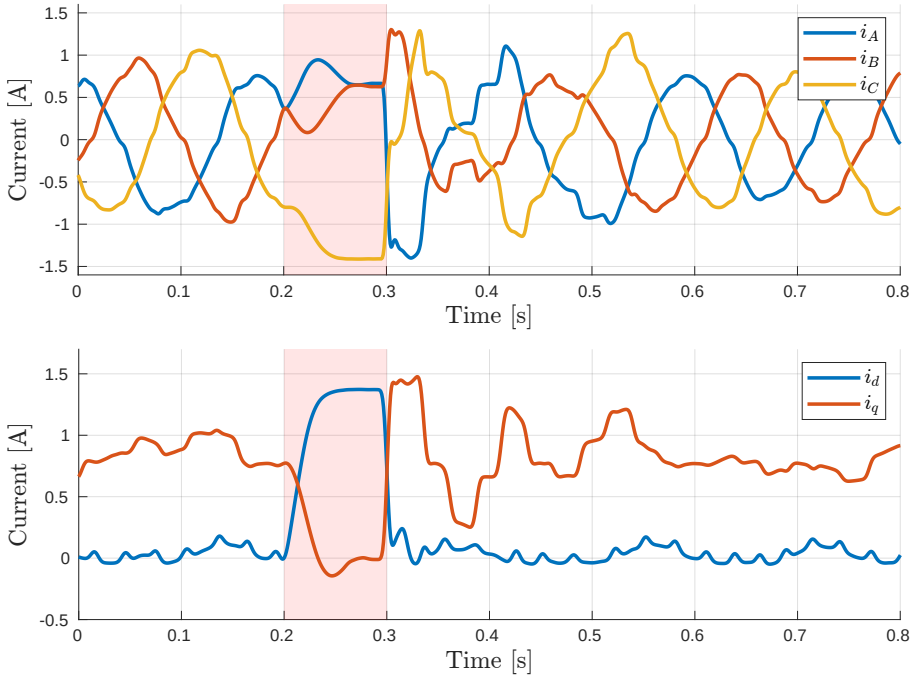


Figure B.8: Motor phase currents and corresponding dq-currents, 20 rad/s, 100 ms interrupt, without strategy, with load attached

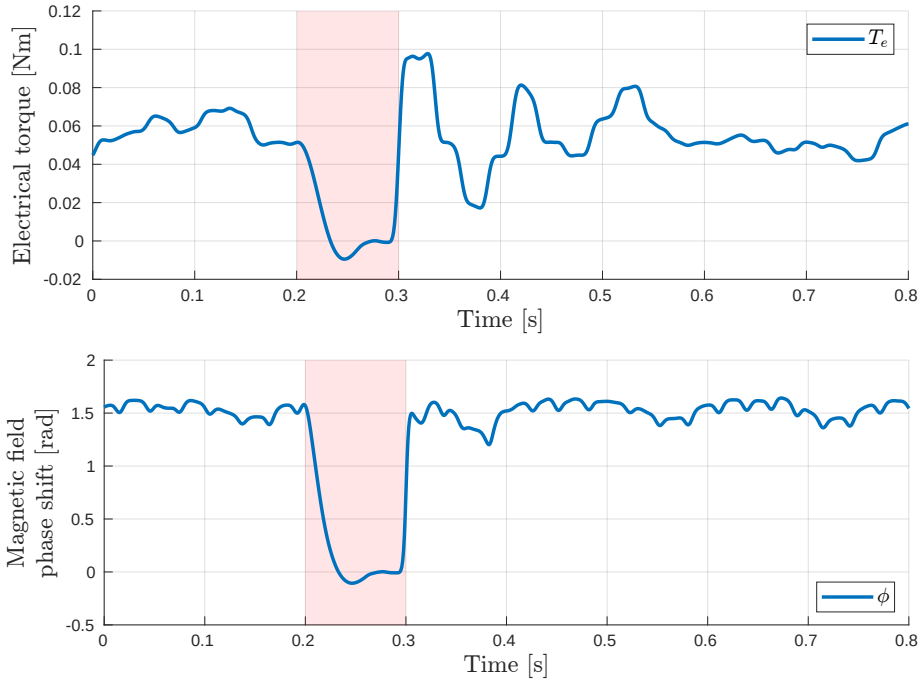


Figure B.9: Electrical torque and magnetic field phase shift, 20 rad/s, 100 ms interrupt, without strategy, with load attached

Figures B.10, B.11, and B.12 show the motor's performance during 100 ms interrupts when it was run at a constant velocity of 100 rad/s without the strategy.

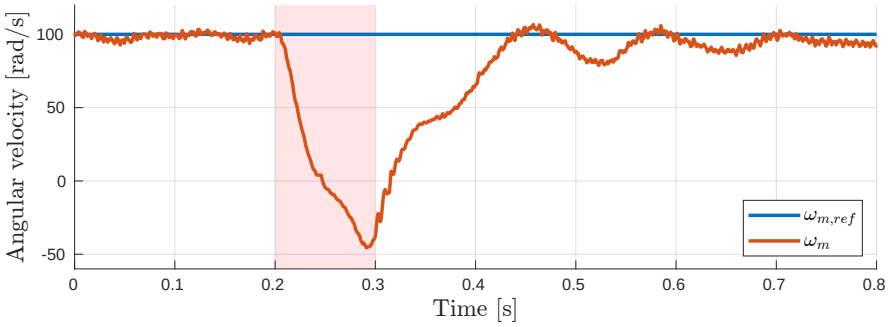


Figure B.10: Rotor angular velocity, 100 rad/s, 100 ms interrupt, without strategy, with load attached

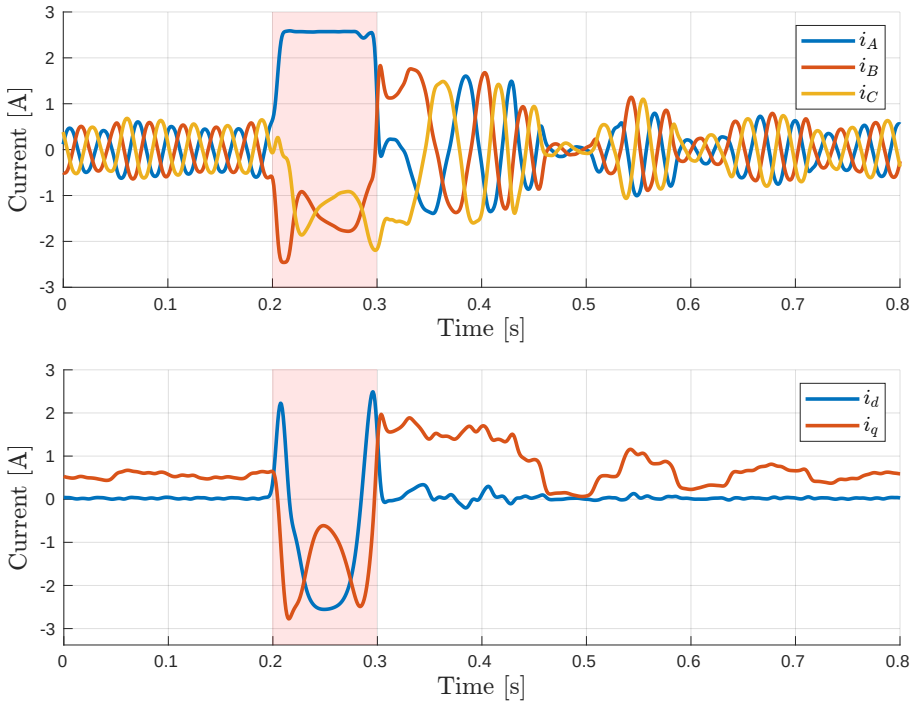


Figure B.11: Motor phase currents and corresponding dq-currents, 100 rad/s, 100 ms interrupt, without strategy, with load attached

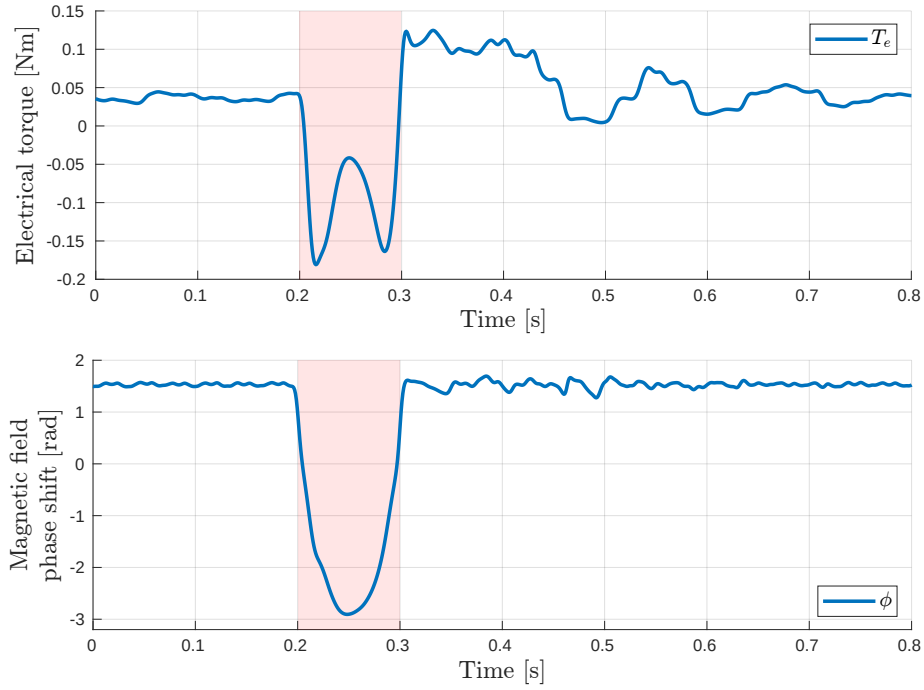


Figure B.12: Electrical torque and magnetic field phase shift, 100 rad/s, 100 ms interrupt, without strategy, with load attached

B.2.2 Constant velocity with strategy

Figures B.13, B.14, and B.15 show the motor's performance during 100 ms interrupts when it was run at a constant velocity of 20 rad/s with the strategy.

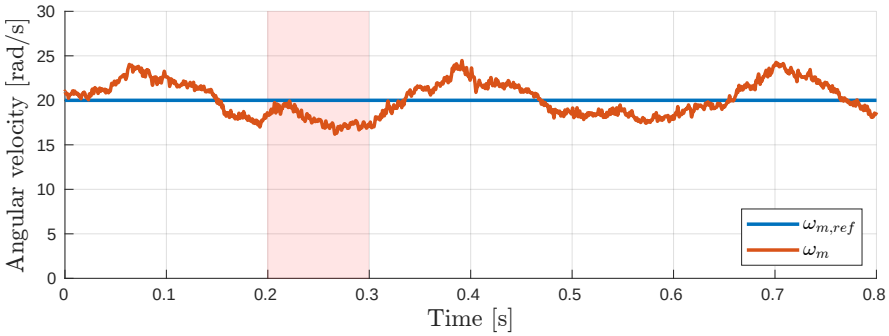


Figure B.13: Rotor angular velocity, 20 rad/s, 100 ms interrupt, with strategy, with load attached

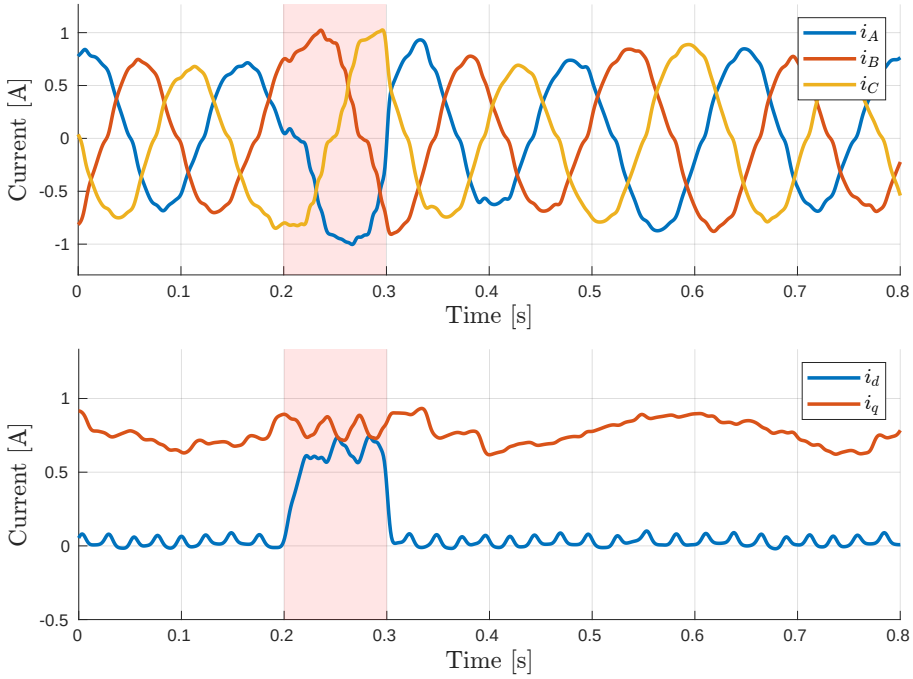


Figure B.14: Motor phase currents and corresponding dq-currents, 20 rad/s, 100 ms interrupt, with strategy, with load attached

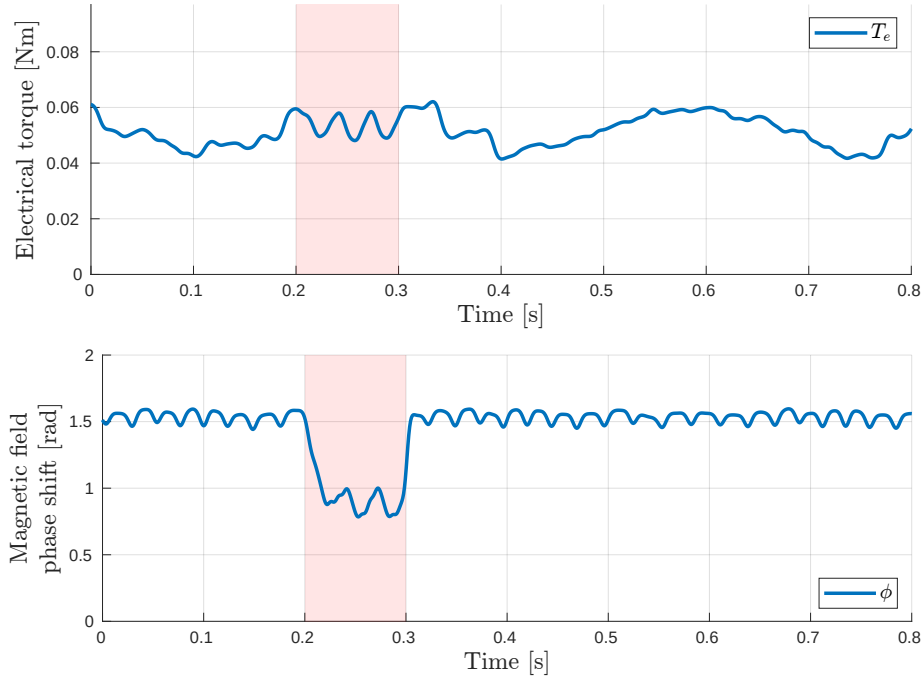


Figure B.15: Electrical torque and magnetic field phase shift, 20 rad/s, 100 ms interrupt, with strategy, with load attached

Figures B.16, B.17, and B.18 show the motor's performance during 100 ms interrupts when it was run at a constant velocity of 100 rad/s with the strategy.

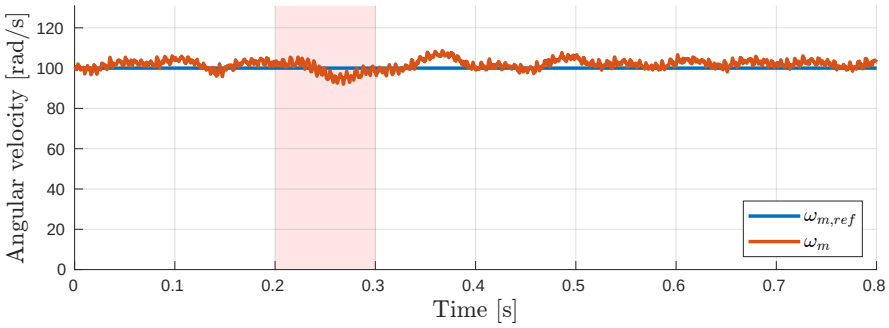


Figure B.16: Rotor angular velocity, 100 rad/s, 100 ms interrupt, with strategy, with load attached

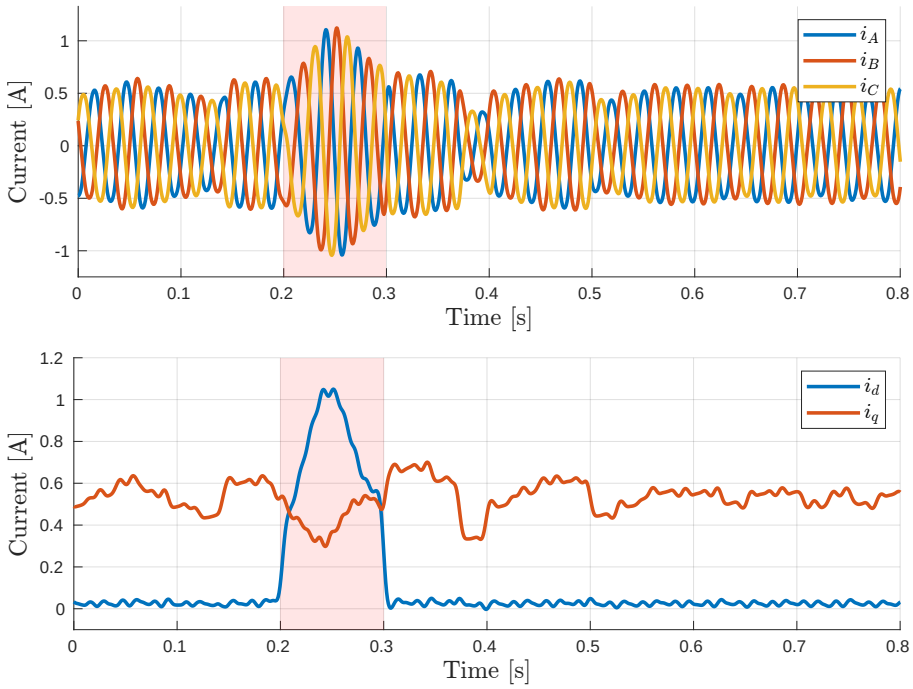


Figure B.17: Motor phase currents and corresponding dq-currents, 100 rad/s, 100 ms interrupt, with strategy, with load attached

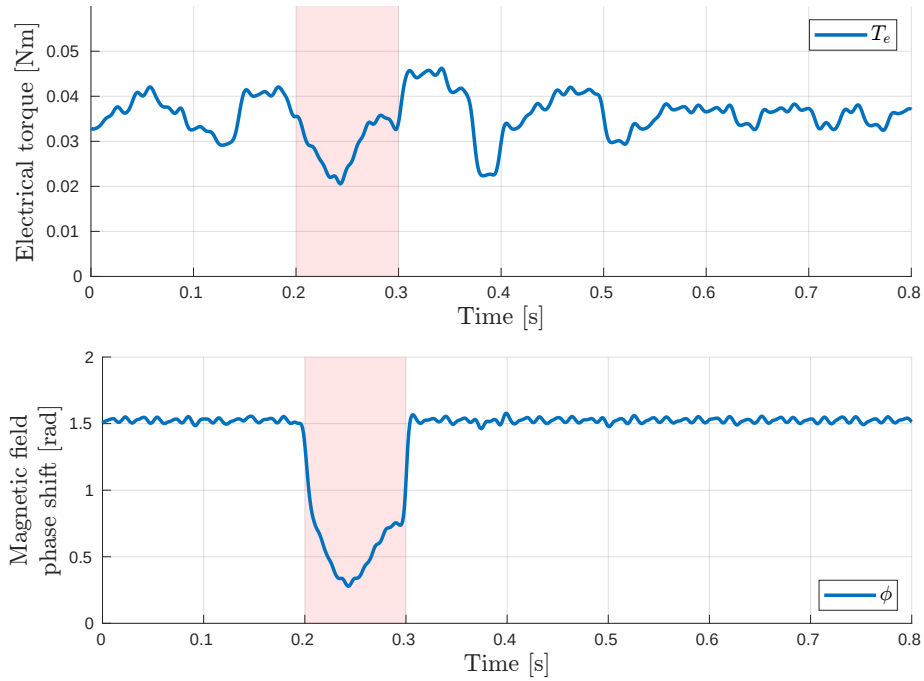


Figure B.18: Electrical torque and magnetic field phase shift, 100 rad/s, 100 ms interrupt, with strategy, with load attached

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY