Lehrstuhl für Elektrische Antriebssysteme und Leistungselektronik
der Technischen Universität München

Advanced Finite-Set Model Predictive Control for Power Electronics and Electrical Drives

Peter Johannes Stolze

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. Hans-Georg Herzog
Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Ralph Kennel
2. Univ.-Prof. Dr.-Ing. Joachim Holtz, em.
   (Bergische Universität Wuppertal)

Die Dissertation wurde am 01.10.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 14.01.2014 angenommen.

The hardest thing is to go to sleep at night, when there are so many urgent things needing to be done. A huge gap exists between what we know is possible with today's machines and what we have so far been able to finish.

Donald Knuth

Preface

I have written this thesis when I was working as a scientific assistant at the Institute for Electrical Drive Systems and Power Electronics at Technische Universität München.

First of all, I would like to thank Prof. Ralph Kennel for all his support during my time at his institute. I am very thankful for your confidence in me and I really appreciated the freedom during all periods of my work. You gave me the chance to present the results of my research at many different conferences all over the world and I also got the opportunity to collaborate with Stellenbosch University in South Africa. My stay in Stellenbosch from 2009 to 2010 was one of the best times in my life.

I want to thank Prof. Joachim Holtz for being the second supervisor of this thesis.

I would also like to thank Prof. Toit Mouton from Stellenbosch University—you supported me in many ways and taught me many things about engineering design, especially regarding power electronics. I always appreciated your comments and constructive suggestions which helped to improve the quality of this work. Without your support I would not have been able to develop and to set up the test benches for three-level inverters.

During my time at the Institute for Electrical Drive Systems and Power Electronics I also had very fruitful discussions and collaborations with many of my colleagues. I would especially like to thank Peter Landsmann with whom I stayed in South Africa several times and my office colleagues Jean-François Stumper, Dirk Paulus, Sascha Kühl and Alexander Dötlinger. I also want to thank Petros Karamanakos with whom I collaborated very successfully, too.

For my work it was necessary to set up a test bench with a three-level inverter completely from scratch, together with a suitable real-time computer system. This would not have been possible at all without the help of very talented and motivated students. Namely, I especially want to thank Janos Jung and Mathias Kramkowski—you did an outstanding job while doing your Diploma and Bachelor theses! I also want to thank Mauricio Trincado, my first student at university: You helped me a lot to quickly pick up all the necessary knowledge about electrical machines, power electronics and predictive control.

I also want to thank Prof. Christian Endisch who encouraged me to come back to university. Without you, this thesis would never have been written! Thank you for all the helpful suggestions, discussions and everything else since we know each other!

Finally, I want to thank my family for all their support—not just during this work but also during my entire life. Thanks to my parents who always supported me and who also encouraged me to go back to university. It is very sad that my father had to pass away far too early and that

he cannot read this thesis anymore. I do hope that you are proud of me—wherever you may be now. I also want to say thank you to my brother in law, Stefan Schenk, for proofreading the final version of this thesis. Thank you to my wife Barbara, you had so many evenings and also weekends alone when I was still working on my computer to get done all of this work and the things beyond it. Thanks that you always stood by my side although it was not always easy and thank you for reminding me that there's a life beyond work and computers. You also gave birth to our wonderful son Raphael who is—besides you—the greatest joy in my life.

Munich, in June 2014
Peter Stolze

Contents

CHAPTER 1

Introduction

In industry controlled electrical drives are used for many different applications. Nowadays, DC machines are mostly replaced by digitally controlled AC machines due to the huge progress which was made in the (power) electronics field. Compared to DC machines which have separate windings for the field- and torque-producing currents, the control of AC machines requires more complicated control loops. Furthermore, multilevel inverters also become more and more interesting for industrial applications since these topologies offer several benefits compared to simpler two-level inverters (e.g. better voltage and current quality). However, these topologies also require more sophisticated control strategies. Additionally, in order to increase efficiency and to reduce power losses, switching mode power supplies are widely used today which also have to be controlled actively.

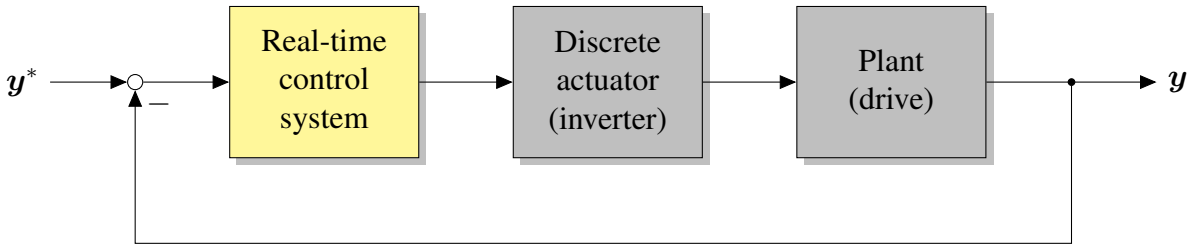


Figure 1.1: Typical control structure in the field of power electronics and electrical drives

Figure 1.1 shows a simplified and basic control structure which is very common in the field of power electronics and electrical drive systems: In general, the system which should be controlled (the plant, e.g. a drive) is fed by an actuator. This actuator is normally an inverter or any other power electronics circuit with *switching* components, i.e. it is an actuator which can only produce certain *discrete-valued* outputs. The real-time control system takes measurements of the system's controlled variables $\boldsymbol{y}$ at every sampling instant and calculates reference values for

the actuator such that the system follows its references $\boldsymbol{y}^*$. This control system has to execute the algorithm in real-time, i.e. the execution of the program has to be finished before the next sample starts. The sampling time, i.e. the time interval between two samples, can be chosen by the user.

In general, the following things must be considered:

- Typical control tasks (e.g. speed and position control of AC drives) lead to higher-order differential equations with different nonlinearities (e.g. saturation effects).
- Constraints of the plant (e.g. current and speed limitation) and of the actuator (maximum voltage, current etc.) must be taken into account.
- The actuator can only produce certain discrete values for the actuating variables (switching behavior).
- In some cases multivariable control, i.e. control of more than only one variable is necessary.

In nearly all industrial applications conventional (linear) PID controllers are used. These linear controllers consist of a proportional (P), an integral (I) and a differential (D) part. In order to fulfill the requirements stated above, several extensions have to be made:

- *Pulse Width Modulation* (PWM) is necessary to "synthesize" discrete values from a continous-valued output variable which is necessary because PID controllers cannot handle discrete-valued variables.
- *Output limitation* and *anti-windup* strategies have to be implemented in a PID controller.
- *Multivariable control* (more than only one variable has to be controlled, e.g. flux and torque control) is performed by single PID controllers. Mutual impact of the controlled variables (e.g. cross coupling effects) is considered as "external disturbance" in the respective PID controller tuning.
- For higher-order systems like speed and position control of drives *cascaded control loops* are used.

Output limitation, anti-windup, PWM and especially the mutual impact of controlled variables add further nonlinearities and lead to a non-optimal control result. In addition, a cascaded control structure introduces another important disadvantage: For stability reasons the inner control loop has to be *significantly* (about 7–10 times) faster than the outer one which deteriorates the performance of the outer loop [1].

These considerations make clear that an optimal control result for the *whole* operation range cannot be easily achieved with a single controller: Almost all industrial controllers today make use of additional feedforward controllers, reference value filtering and adaptive schemes. Nevertheless, nowadays the conventional methods are still sufficient to fulfill their demands. In the future this might not be true anymore: More sophisticated control strategies are already gaining more and more interest in the field of power electronics and electrical drives because of higher dynamic requirements and the use of more advanced power electronics circuits (e.g. multilevel inverters).

A very promising alternative to conventional controllers is Model Predictive Control (MPC) which offers several advantages compared to linear control schemes (e.g. the possibility to resolve cascaded control loops). Especially Finite-Set MPC (FS-MPC) seems to be a very promising strategy in this field: This control principle takes the discrete nature of the actuator directly into account and does not need a modulator. Furthermore, it is a very simple and intuitive method which is also capable of considering all kinds of nonlinearities and constraints.

However, FS-MPC methods currently still suffer from two major drawbacks:

1. FS-MPC allows switchings to take place *only* at the beginning of a sample while PWM can distribute the switchings over the whole sampling interval. This leads to high ripples on the controlled variables if FS-MPC methods are compared to PWM-based methods.

2. The calculation effort rises exponentially with the prediction horizon. Calculating more than one prediction step for multilevel inverters can already be too demanding for a successful real-time implementation.

These two problems demonstrate that the currently known FS-MPC methods have to be improved until these algorithms—despite of all their advantages—can compete with traditional PID controllers with subsequent PWM.

In order to solve these problems, the following work extends the existing FS-MPC methods. Thus, the three main points of this thesis are:

1. The development of a heuristic method to reduce the calculation effort, especially for multilevel inverters and higher prediction horizons.

2. Methods to reduce ripples on the controlled variables by increasing the time resolution of FS-MPC algorithms.

3. A combination of both methods to achieve a better control result (less ripples) with reduced calculation effort.

This work is organized as follows: The next chapter gives an overview of the used systems and control algorithms in order to supply the reader with the necessary theoretical background. In chapter 3 important remarks for a successful real-time implementation of control algorithms are given and the used control platforms are introduced. Chapter 4 deals with Predictive Torque Control, an FS-MPC alternative to Direct Torque Control. In chapter 5 Predictive Current Control, an FS-MPC alternative to Field Oriented Control is described and analyzed and optimizations are presented for this approach. Chapter 6 presents a different approach where an oversampled control algorithm is implemented on programmable hardware. Finally, in chapter 7 the conclusions are made and an outlook to further work is given.

CHAPTER 2

Background and motivation

This chapter gives a brief overview of the used physical systems, their working principles and explains all necessary details which are needed to understand the systems which are used within this work. Furthermore, this chapter gives a short overview of the state of the art in control of electrical drive systems and highlights known problems in these methods. It is followed by a general introduction to Model Predictive Control (MPC) techniques with special focus on Finite-Set MPC (FS-MPC). Finally, the two major drawbacks of FS-MPC methods, the high calculation effort and the low time resolution compared to modulation-based methods, are explained in detail in order to clarify the motivation for this work.

2.1 Mathematical symbols and definitions

In this work the following conventions for mathematical symbols and definitions are used:

- Scalars are written in italic letters: x
- Vectors are bold lower case letters: $\boldsymbol{x}$
- Matrices are bold upper case letters: $\boldsymbol{X}$
- References are marked with a star superscript: x^*

Another definition is made regarding the switching frequency: Two transitions per semiconductor device (one *on*- and one *off*-transition) are counted as one switching event.

Further information about the used mathematical symbols, definitions and also abbreviations can be found in appendix A.

2.2 Physical values and units

All given physical values are *not* normalized. Normalization leads to the fact that the relatedness of the given values to the real physical values is lost. Because of this and as additional calculations are necessary for the normalization, this can lead to severe errors during the implementation process. Furthermore, for today's control platforms it is not necessary anymore to use normalized values for the implementation of control algorithms: Today's rapid prototyping systems can handle 32 bit floating point numbers which provide enough accuracy, even for non-normalized values. In former times this was different: Older processors had much less computational power and could only handle 8 or 16 bit (fixed point) numbers. For such systems normalization was essential in order to achieve the needed accuracy.

All used physical quantities are given in SI units. The only exception is the rotational speed which is given in revolutions per minute (rpm). This choice was made as rpm is more common than revolutions per second (s^{-1}).

2.3 Discretization of differential equations

2.3.1 Continous-time state-space system representation

For a linear time-invariant (LTI) system the continuous-time differential equations can be written in state-space form [2]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \tag{2.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t), \tag{2.2}$$

where t is the time, $\boldsymbol{x}$ the state vector, $\boldsymbol{u}$ the input vector and $\boldsymbol{y}$ is the output vector. $\boldsymbol{A}$ is the state, $\boldsymbol{B}$ the input, $\boldsymbol{C}$ the output and $\boldsymbol{D}$ the feedthrough matrix.

2.3.2 Discrete-time state-space system representation

For implementing control algorithms on digital control systems it is necessary to discretize the continuous-time system equations. If a linear system is discretized with the sampling time T_s, the general formulation for the corresponding discrete-time system equations in state space form [3] can be obtained:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_\mathrm{d}\boldsymbol{x}(k) + \boldsymbol{B}_\mathrm{d}\boldsymbol{u}(k), \tag{2.3}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}_\mathrm{d}\boldsymbol{x}(k) + \boldsymbol{D}_\mathrm{d}\boldsymbol{u}(k), \tag{2.4}$$

where $k = \frac{t}{T_\mathrm{s}}$ is the current sampling cycle. All matrices for the discrete-time system representation are denoted in the form $(*)_\mathrm{d}$.

2.3.3 Exact discretization and approximations

A discrete-time system representation can be obtained from the continuous-time equations with several different methods. For a sake of brevity only the most important ones with respect to FS-MPC are mentioned in the following.

2.3.3.1 Exact discretization

Especially for linear systems it is possible to discretize the differential equations exactly for a given sampling time T_s. For nonlinear systems, however, it is not always possible to deliver an exact solution.

2.3.3.2 Euler-forward discretization

The Euler-forward discretization is the most simple approximation method. Assuming that the sampling time is T_s, the time differentiation of the state vector results to

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) \approx \frac{\boldsymbol{x}(k+1) - \boldsymbol{x}(k)}{T_\mathrm{s}}. \tag{2.5}$$

The discrete-time state-space matrices for the Euler-forward approximation are given by

$$\begin{aligned} \boldsymbol{A}_\mathrm{d} &= \mathbf{1} - \boldsymbol{A} \cdot T_\mathrm{s}, && (2.6)\\ \boldsymbol{B}_\mathrm{d} &= \boldsymbol{B} \cdot T_\mathrm{s}, && (2.7)\\ \boldsymbol{C}_\mathrm{d} &= \boldsymbol{C} \text{ and} && (2.8)\\ \boldsymbol{D}_\mathrm{d} &= \boldsymbol{D} && (2.9) \end{aligned}$$

where $\mathbf{1}$ is the unity matrix. The Euler-forward approximation mostly delivers results which are accurately enough for short prediction horizons. However, if the sampling interval T_s becomes too long, this approximation can become unstable (even if the continuous-time system is stable).

2.3.3.3 Euler-backwards discretization

The Euler-backwards discretization is very similar to the previously mentioned Euler-forward method. The approximation of the time differentiation is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x} \approx \frac{\boldsymbol{x}(k) - \boldsymbol{x}(k-1)}{T_\mathrm{s}}. \tag{2.10}$$

This approximation is numerically more stable than the Euler-forward method but in some cases it is not possible to calculate the discrete-time state-space matrices explicitly.

2.3.3.4 Higher-order discretization methods

Better approximations to the solution of the continuous-time differential equations can be obtained through higher-order and more sophisticated discretization methods, e.g. Runge-Kutta. For an overview of these methods, the interested reader is referred to [4].

2.4 Three-phase system and space vector notation

2.4.1 Clarke transformation

Most of the drives which are nowadays used are three-phase electrical machines and thus connected to three-phase sinusoidal voltage sources. The phase shift between the voltages is $120°$.

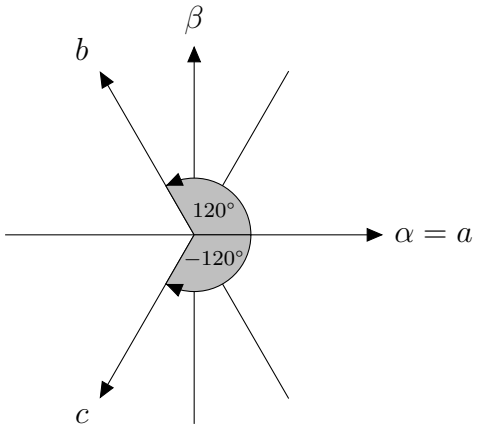


Figure 2.1: Space-vector notation of three-phase systems

The basic idea behind the space vector notation is that a three-phase system can be visualized by a two-dimensional three-phase coordinate system. The three axes a, b and c which represent the three phases are shown in Figure 2.1. All three axes together are linearly dependent on each other. In order to simplify the system, the two axes α and β which are linearly independent are used to describe a point in the space vector plane. By using the Clarke transformation [5]

$$\begin{pmatrix} \alpha \\ \beta \\ \delta \end{pmatrix} = \frac{2}{3} \begin{pmatrix} 1 & -0.5 & -0.5 \\ 0 & 0.5\sqrt{3} & -0.5\sqrt{3} \\ 0.5 & 0.5 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \tag{2.11}$$

it is possible to transform from the three-phase coordinate system to an equivalent two-phase system. The additional component, δ, is only necessary for unbalanced circuits, i.e. if the condition

$$a + b + c = 0 \tag{2.12}$$

is not fulfilled. In this work for the Clarke transformation of currents equation (2.12) is valid for all three-phase systems. This assumption is not true for the transformation of inverter output voltages to the $\alpha\beta$ plane but as long as no back-transformation of the voltages to the abc plane is necessary, the δ component can still be omitted.

For balanced circuits equation (2.11) can be further simplified as only two of the three phases have to be measured:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{3}\sqrt{3} & \frac{2}{3}\sqrt{3} \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} \tag{2.13}$$

The inverse transformation from the $\alpha\beta$ coordinate system to abc coordinates can be done as follows:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ -0.5 & 0.5\sqrt{3} & 1 \\ -0.5 & -0.5\sqrt{3} & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \\ \delta \end{pmatrix} \tag{2.14}$$

2.4.2 Park transformation

The Park transformation [6, 7] is used to transform from a stator-fixed $\alpha\beta$ coordinate system to a rotating dq system with the same origin. As it can be seen in Figure 2.2, the dq coordinate system is rotated by the angle φ against the $\alpha\beta$ system. Because of this the Park transformation is a simple rotation. The angle φ is the rotor flux angle which can be obtained by $\varphi = \arctan\left(\frac{\psi_{\mathrm{r}\beta}}{\psi_{\mathrm{r}\alpha}}\right)$ where $\psi_{\mathrm{r}\alpha}$ and $\psi_{\mathrm{r}\beta}$ are the α and β components of the rotor flux, respectively. Consequently, the

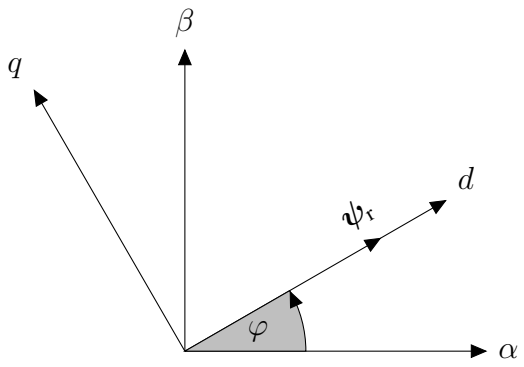


Figure 2.2: Park transformation

Park transformation (from $\alpha\beta$ to dq coordinates) is given by

$$\begin{pmatrix} d \\ q \end{pmatrix} = \begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{2.15}$$

The inverse Park transformation then results to

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix} \cdot \begin{pmatrix} d \\ q \end{pmatrix}. \tag{2.16}$$

2.5 Physical systems

This section briefly introduces the physical systems which should be controlled: At the beginning a short explanation of the used inverter topologies (two-level, three-level Neutral Point Clamped (NPC) and three-level Flying Capacitor (FC)) is given, together with remarks about voltage balancing. After this the inverter loads are introduced: First, the differential equations for a resistive-inductive load are given and then the second-order LC lowpass-filter is explained. Finally, the differential equations which are necessary to model an induction machine are presented.

2.5.1 Two-level inverter

2.5.1.1 Basic working principle

Figure 2.3 shows a single-leg two-level inverter with two semiconductor switches S_{11} and S_{12} and its output voltage v_{out}. NP is the neutral (zero) point, the DC link rails are marked with

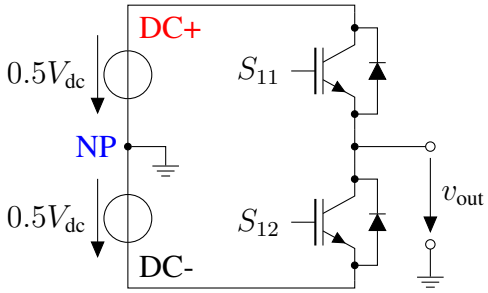


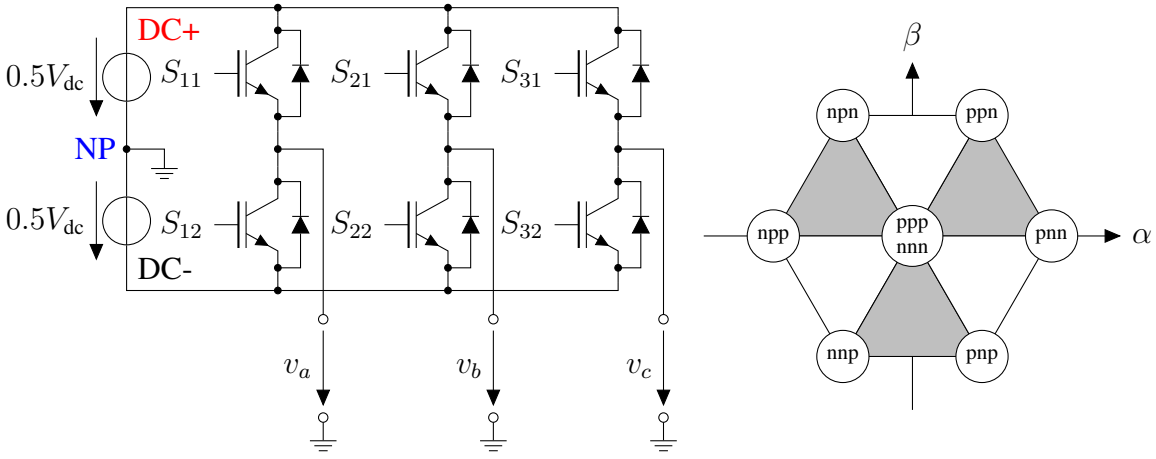
Figure 2.3: Single-leg two-level inverter

DC+ and DC-. Within this work the semiconductor switches always consist of an Insulated Gate Bipolar Transistor (IGBT) with an antiparallel freewheeling diode. The switch positions which can either be *on* or *off* are complementary: If both switches are *on*, this would lead to a short circuit whereas both switches in the *off* state lead to an undefined potential at the output clamp of the leg. Hence, two different switch positions are allowed which can be seen in Table 2.1, together with the produced output voltage at the leg's clamping terminal.

Table 2.1: Switching states and output voltages of a two-level inverter leg x

S_{x1}	S_{x2}	v_{out}
1	0	$0.5V_{\text{dc}}$
0	1	$-0.5V_{\text{dc}}$

2.5.1.2 Voltage vectors



(a) Three-phase two-level inverter (b) Voltage vectors

Figure 2.4: Three-phase two-level inverter and its produced voltage vectors

For three-phase operation three two-level inverter legs can be connected to the same DC bus as it is shown in Figure 2.4(a). As explained in the previous chapter, for every leg two switching

states are possible. This leads to $2^3 = 8$ different switching states. According to chapter 2.4, the Clarke transformation can be applied to these switching states leading to seven different voltage vectors (resulting from six active and two zero switching states) which are visualized in Figure 2.4(b).

There are two possibilities for coding the switching states: As it can be seen in Figure 2.4(b) and as it was explained in the previous chapter, there are two switching states per phase (named "n" for the negative and "p" for the positive one). The switching states in the three phases a, b and c are given by s_a, s_b and s_c. Then, the inverter output voltages in $\alpha\beta$ coordinates can be calculated as

$$\begin{pmatrix} v_\alpha \\ v_\beta \end{pmatrix} = \gamma V_{\text{dc}} \begin{pmatrix} 1 & -0.5 & -0.5 \\ 0 & 0.5\sqrt{3} & -0.5\sqrt{3} \end{pmatrix} \cdot \begin{pmatrix} s_a \\ s_b \\ s_c \end{pmatrix}. \tag{2.17}$$

If the switching states s_a, s_b and s_c are coded with -1 for the negative and 1 for the positive one, the multiplication factor has to be set to $\gamma = 0.5$. For two-level inverters, however, the switching states are often also coded with 0 for the negative and 1 for the positive one. In this case the correct multiplication factor is $\gamma = 1$.

In this thesis, however, the switching states are coded with -1 and 1 for all inverter types since this definition is consistent to the modelling of multilevel inverters later on.

2.5.2 Three-level Neutral Point Clamped inverter

2.5.2.1 Basic working principle

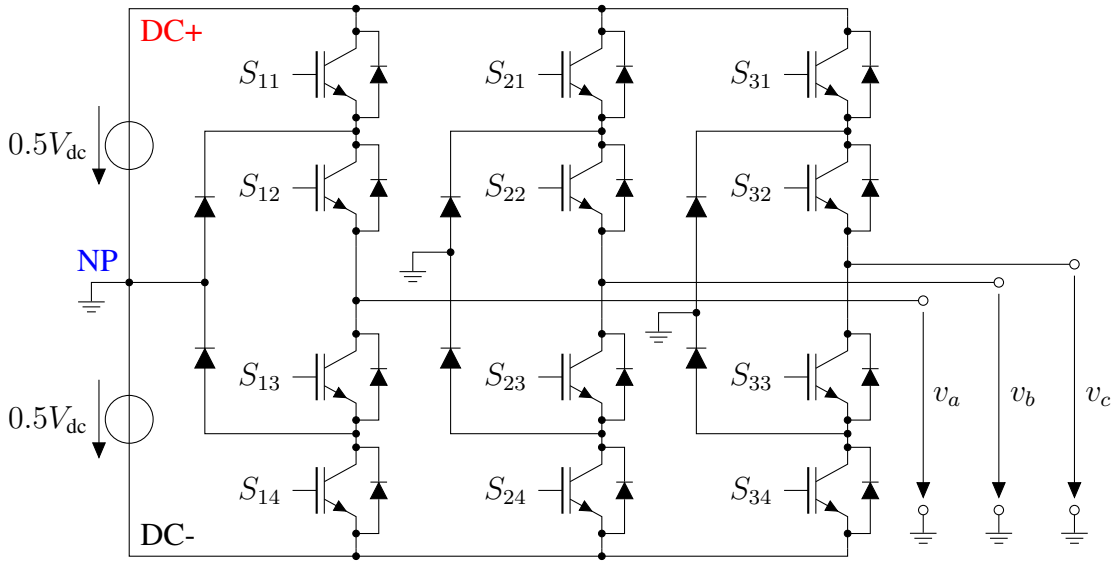


Figure 2.5: Three-phase three-level Neutral Point Clamped inverter

Figure 2.5 shows a three-phase three-level Neutral Point Clamped (NPC) inverter [8]. Every leg consists of four switches $S_{xi}, i = 1 \ldots 4$ where x denotes the phase leg and i the number of the switch in that phase. The switches S_{x1} and S_{x3} as well as the ones S_{x2} and S_{x4} are complementary. Additionally, the outer two switches in a phase (S_{x1} and S_{x4}) are not allowed

Table 2.2: Switching states and output voltages of a three-level NPC inverter leg x

S_{x1}	S_{x2}	v_{out}
0	0	$-0.5V_{\text{dc}}$
0	1	$0\,\text{V}$
1	1	$0.5V_{\text{dc}}$

to be *on* at the same time. This leads to three different switching states per phase which are given in Table 2.2.

2.5.2.2 Voltage vectors

If a three-phase application is considered, a three-level NPC inverter can produce $3^3 = 27$ different switching states. The resulting 19 voltage vectors (after the Clarke transformation has been applied to the switching states) can be seen in Figure 2.6.

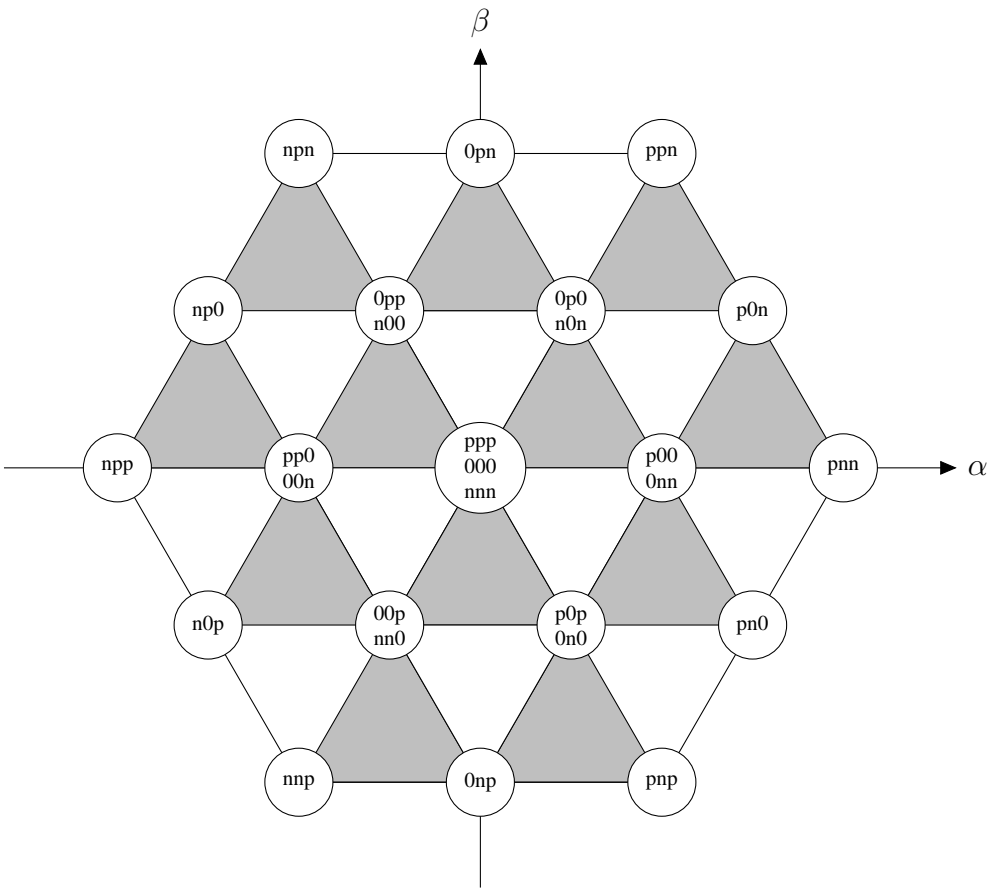


Figure 2.6: Voltage vectors of a three-level inverter

2.5.2.3 Voltage balancing

In contrast to Figure 2.5, normally only one voltage source is used in the DC link. In most applications the DC link is fed from a three-phase diode bridge rectifier which is shown in Figure 2.7.

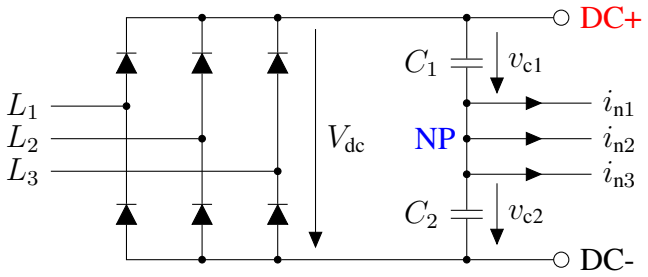


Figure 2.7: Three-phase diode bridge rectifier feeding the DC link

In order to buffer the pulsating voltage output of the rectifier, DC link capacitors are used to decrease the voltage ripple. For the case of an NPC inverter two capacitors in serial connection replace the two voltage sources. Figure 2.7 gives a more detailed insight into the voltage balancing mechanism of an NPC inverter: Ideally, both DC link capacitor voltages v_{c1} and v_{c2} should be equal, i.e. their reference values are

$$v_{c1}^* = v_{c2}^* = 0.5 V_{dc}. \tag{2.18}$$

In other words, the reference value for the capacitor voltage difference should be

$$\Delta v_c^* = v_{c1} - v_{c2} \stackrel{!}{=} 0\,\mathrm{V}. \tag{2.19}$$

The differential equation of a capacitor is given by

$$i_{cj} = C \frac{\mathrm{d}v_{cj}}{\mathrm{d}t}, \tag{2.20}$$

where i_{cj} is the capacitor current, C the capacitance and $j = 1, 2$. By taking a closer look at Figure 2.5, it is clear that only the zero switching state in the phase j leads to a neutral point current i_{nj}. If a positive or a negative switching state is selected, no current is drawn from the neutral point, i.e. these switching states do not affect the capacitor voltage balance. Then, the differential equation for the capacitor voltage difference can be calculated to

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\Delta v_c\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(v_{c1} - v_{c2}\right) = \frac{\mathrm{d}v_{c1}}{\mathrm{d}t} - \frac{\mathrm{d}v_{c2}}{\mathrm{d}t} = \frac{1}{C}\left(i_{c1} - i_{c2}\right). \tag{2.21}$$

According to the Kirchhoff current law, the neutral point current i_n is given by

$$i_n = i_{c1} - i_{c2}. \tag{2.22}$$

As i_n is the sum of the three neutral point currents in the phases (i_{nj}, $j = 1 \ldots 3$), the final equation which describes the NPC voltage balancing mechanism results to

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\Delta v_c\right) = \frac{1}{C}\left(\sum_{j=1}^{3} i_{nj}\right). \tag{2.23}$$

2.5.3 Three-level Flying Capacitor inverter

2.5.3.1 Basic working principle

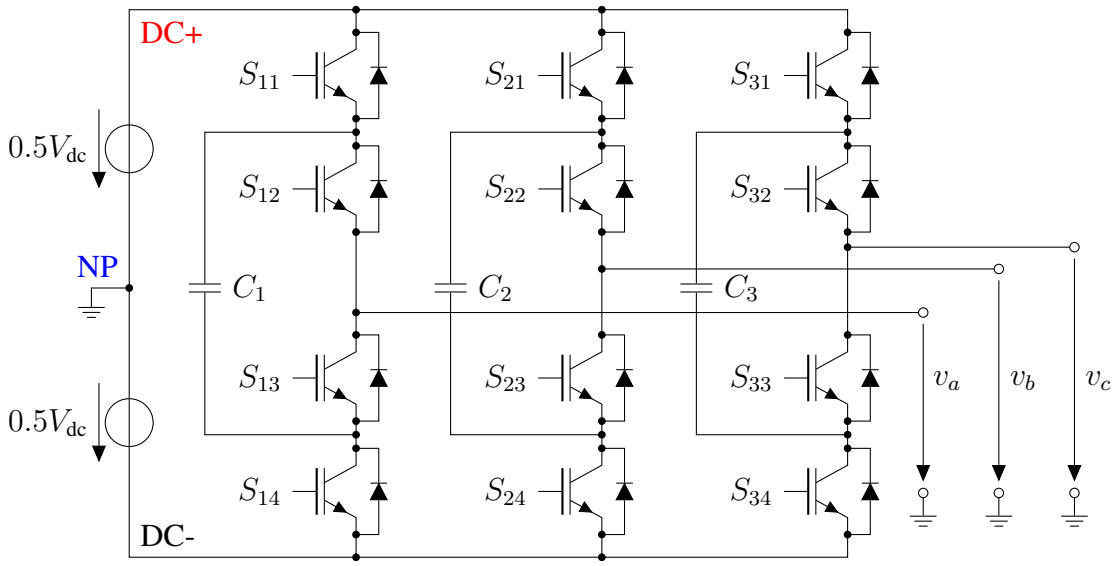


Figure 2.8: Three-phase three-level Flying Capacitor inverter

Figure 2.8 shows a three-level three-phase Flying Capacitor (FC) inverter [9]. Similar to the NPC inverter, every phase leg consists of four semiconductor switches S_{xi}, $i = 1 \ldots 4$, where x denotes the phase leg and i is the number of the switch in this phase. The switches S_{x1} and S_{x4} as well as the ones S_{x2} and S_{x3} are complementary. This leads to four different switching states per phase which are shown in Table 2.3.

Table 2.3: Switching states and output voltages of a three-level FC inverter leg x

S_{x1}	S_{x2}	v_{out}
0	0	$-0.5V_{\text{dc}}$
0	1	0 V
1	0	0 V
1	1	$0.5V_{\text{dc}}$

In contrast to the NPC inverter, the flying capacitor topology has two different switching states per phase which lead to 0 V in the output.

2.5.3.2 Voltage vectors

Considering a three-phase application, a three-level FC inverter has $4^3 = 64$ different switching states. However, as every phase can only produce three different output voltages, the number of voltage vectors is still 19 as for an NPC inverter. Thus, the resulting voltage vectors are the same as for the NPC inverter and can be seen in Figure 2.6. The FC inverter has a higher number of redundant switching states compared to the NPC topology.

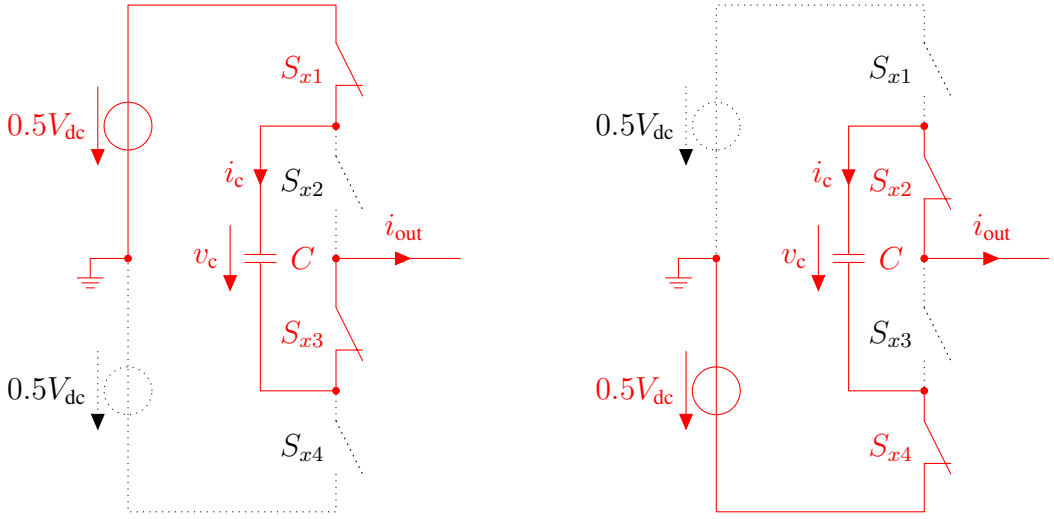
2.5.3.3 Voltage balancing

Similarly to the NPC inverter, the voltages across the flying capacitors in every phase leg have to be balanced and to be kept within a certain range around their reference values. In order to produce 0 V in the phase output, the flying capacitor voltage reference is

$$v_{c1}^* = v_{c2}^* = v_{c3}^* \stackrel{!}{=} 0.5V_{dc}. \tag{2.24}$$

The differential equation for a capacitor which is given in equation (2.20) is necessary for understanding the mechanism of the voltage balancing. Positive or negative switching states in a phase do not affect the flying capacitor voltage, similar to the NPC inverter. In contrast to the NPC inverter, the voltage balancing of every FC inverter phase is independent from the other phases.

In order to get a deeper insight into this mechanism, Figures 2.9(a) and 2.9(b) show the two different zero switching states of an FC inverter leg x, assuming that a certain current is drawn out of that leg. If the first zero state (Figure 2.9(a)) is selected, the output current i_{out} is equal to



(a) S_{x1} and S_{x3} *on*, S_{x2} and S_{x4} *off*

(b) S_{x2} and S_{x4} *on*, S_{x1} and S_{x3} *off*

Figure 2.9: Zero switching states of a three-level FC inverter affecting the capacitor voltage

the capacitor current i_c which leads to the differential equation

$$\frac{\mathrm{d}v_c}{\mathrm{d}t} = \frac{1}{C} i_{out}, \tag{2.25}$$

i.e. a positive output current will increase the capacitor voltage, while a negative one will decrease it.

By taking a closer look at the second zero state (Figure 2.9(b)), it can be easily seen that in this case i_{out} is directed opposite to i_c, and hence

$$\frac{\mathrm{d}v_c}{\mathrm{d}t} = -\frac{1}{C} i_{out}, \tag{2.26}$$

i.e. a positive output current will decrease the capacitor voltage and vice versa.

In conclusion, the flying capacitor voltage in every phase can be balanced by choosing the appropriate zero switching state.

2.5.4 Natural voltage balancing mechanisms

Since no circuit is ideal, every semiconductor switch has parasitic effects. Several other non-linearities (e.g. leakage currents) have to be considered for an accurate modelling of power electronics devices and circuits, too. Regarding multilevel inverters, these effects can lead to natural voltage balancing mechanisms [10]. This means that—at least in some special cases—no additional voltage balancing algorithms are necessary in order to keep the capacitor voltages close enough to their references. For the FS-MPC control methods introduced later in this section, however, these natural balancing mechanisms are not sufficient to keep the voltages around their references.

2.5.5 Resistive-inductive load

Figure 2.10 shows the circuit diagram of a resistive-inductive (RL) load. The voltage drop

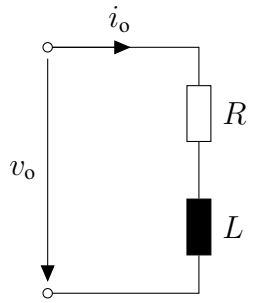


Figure 2.10: Resistive-inductive load

across a resistor can be calculated according to Ohm's law. The voltage drop across an inductor is proportional to the derivative of the current flowing through it. Then, the differential equation of an RL load can be easily set up. The state-space form of the RL load equations then results to

$$\frac{\mathrm{d}}{\mathrm{d}t} i_\mathrm{o} = -\frac{R}{L} \cdot i_\mathrm{o} + \frac{1}{L} \cdot v_\mathrm{o} \tag{2.27}$$

where v_o is the applied (inverter output) voltage, i_o the resulting (inverter output) current, R the resistance and L the inductance.

2.5.6 LC lowpass-filter

The circuit diagram of an LC lowpass-filter is shown in Figure 2.11. Such circuits are second-order filters: The LC filter output voltage v_l (which will be applied to the load) is the filtered LC filter input (inverter output) voltage v_o. L is the inductance and C the capacitance. The inductor current is the inverter output current i_o, the capacitor current is i_c. Furthermore, the subsequent circuit at the LC filter's output clamps can draw a certain (probably unknown) output (load) current i_l. The state variables for this circuit are i_o and v_l. The output current can either be assumed as another input or as a constant or slowly changing "dummy" state.

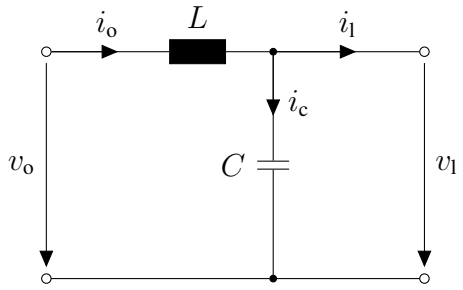


Figure 2.11: LC lowpass-filter

The differential equations for an LC lowpass-filter can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} i_\mathrm{o} \\ v_\mathrm{l} \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{pmatrix} \cdot \begin{pmatrix} i_\mathrm{o} \\ v_\mathrm{l} \end{pmatrix} + \begin{pmatrix} \frac{1}{L} & 0 \\ 0 & -\frac{1}{C} \end{pmatrix} \cdot \begin{pmatrix} v_\mathrm{o} \\ i_\mathrm{l} \end{pmatrix} \tag{2.28}$$

2.5.7 Induction machine

In contrast to separately excited DC machines which have different windings for the field- and for the torque-producing currents, AC machines only have a three-phase stator winding. AC machines can be separated into synchronous and asynchronous (induction) machines. This work only deals with induction machines (IMs); because of this only a brief introduction to IMs is given in the following.

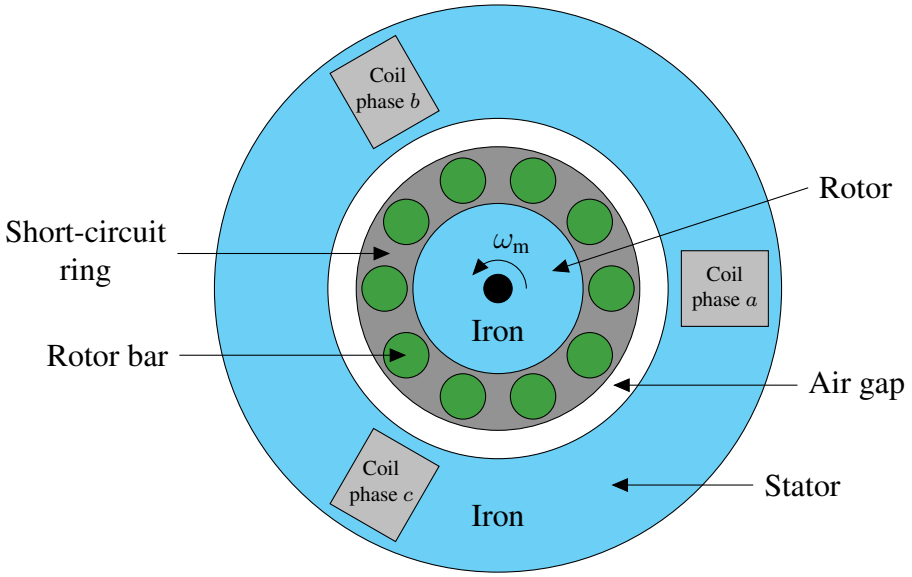


Figure 2.12: Squirrel-cage induction motor (simplified)

Figure 2.12 shows a simplified diagram of a squirrel-cage induction motor. In this case the distributed stator windings are replaced by concentrated windings. The three-phase windings have a spatial phase shift of 120°. If the stator coils are connected to a three-phase sinusoidal voltage source (with an electrical phase shift of 120° between the phases), a rotating magnetic

field is produced in the stator. As long as this magnetic field has a different rotational speed than the rotor (i.e. als long as they are *asynchronous*), voltages are induced in the rotor. The resulting short-circuit currents then produce torque. This torque is dependent on the relative rotational speed between the rotor and the magnetic field in the stator.

Since—as already mentioned—induction motors only have a three-phase stator winding, a separate control of both flux and torque is only possible with a deeper insight into the mechanisms of such motors. For this reason, the basic IM equations and their differential equations are explained in this chapter.

According to [11], the basic equations of an IM can be written in a coordinate system which rotates with an arbitrary angular velocity ω_k:

$$\boldsymbol{\psi}_\mathrm{s} = L_\mathrm{s}\boldsymbol{i}_\mathrm{s} + L_\mathrm{m}\boldsymbol{i}_\mathrm{r}, \tag{2.29}$$

$$\boldsymbol{\psi}_\mathrm{r} = L_\mathrm{r}\boldsymbol{i}_\mathrm{r} + L_\mathrm{m}\boldsymbol{i}_\mathrm{s}, \tag{2.30}$$

$$\boldsymbol{v}_\mathrm{s} = R_\mathrm{s}\boldsymbol{i}_\mathrm{s} + \frac{\mathrm{d}\boldsymbol{\psi}_\mathrm{s}}{\mathrm{d}t} + j\omega_\mathrm{k}\boldsymbol{\psi}_\mathrm{s} \text{ and} \tag{2.31}$$

$$\boldsymbol{v}_\mathrm{r} = R_\mathrm{r}\boldsymbol{i}_\mathrm{r} + \frac{\mathrm{d}\boldsymbol{\psi}_\mathrm{r}}{\mathrm{d}t} + j\left(\omega_\mathrm{k} - \omega_\mathrm{el}\right)\boldsymbol{\psi}_\mathrm{s}. \tag{2.32}$$

Stator variables are marked in the form $(*)_\mathrm{s}$ while rotor variables are denoted in the way $(*)_\mathrm{r}$. $\boldsymbol{\psi}_\mathrm{s}$ and $\boldsymbol{\psi}_\mathrm{r}$ are the fluxes, $\boldsymbol{i}_\mathrm{s}$ and $\boldsymbol{i}_\mathrm{r}$ the currents, R_s and R_r the resistances, L_s and L_r the inductances and L_m is the mutual inductance between stator and rotor. $\boldsymbol{v}_\mathrm{s}$ is the applied stator voltage and $\boldsymbol{v}_\mathrm{r}$ the rotor voltage ($\boldsymbol{v}_\mathrm{r} = (0\,\mathrm{V}, 0\,\mathrm{V})^T$ for a squirrel-cage induction motor). j is defined as $j = \sqrt{-1}$. ω_el is the electrical angular machine speed which is given by

$$\omega_\mathrm{el} = p \cdot \omega_\mathrm{m}, \tag{2.33}$$

where p is the number of pole pairs and ω_m the mechanical machine speed.

For $\omega_\mathrm{k} = 0$ the coordinate system is stator-fixed, i.e. the coordinates are given in the $\alpha\beta$ system. In order to obtain a more compact system representation, it is quite common to describe the $\alpha\beta$ system with complex numbers where the real part corresponds to the α axis and the imaginary one to the β axis.

According to [12] and [13], equations (2.29)–(2.32) can be rewritten in the form

$$\boldsymbol{i}_\mathrm{s} + \tau_\sigma\frac{\mathrm{d}\boldsymbol{i}_\mathrm{s}}{\mathrm{d}t} = \frac{1}{r_\sigma}\boldsymbol{v}_\mathrm{s} - j\omega_\mathrm{k}\tau_\sigma\boldsymbol{i}_\mathrm{s} + \frac{k_\mathrm{r}}{r_\sigma}\left(\frac{1}{\tau_\mathrm{r}} - j\omega_\mathrm{el}\right)\boldsymbol{\psi}_\mathrm{r}, \tag{2.34}$$

$$\boldsymbol{\psi}_\mathrm{r} + \tau_\mathrm{r}\frac{\mathrm{d}\boldsymbol{\psi}_\mathrm{r}}{\mathrm{d}t} = L_\mathrm{m}\boldsymbol{i}_\mathrm{s} - j\left(\omega_\mathrm{k} - \omega_\mathrm{el}\right)\tau_\mathrm{r}\boldsymbol{\psi}_\mathrm{r}, \tag{2.35}$$

where the coefficients are given by $\tau_\sigma = \frac{\sigma L_\mathrm{s}}{r_\sigma}$ and $r_\sigma = R_\mathrm{s} + k_\mathrm{r}^2 R_\mathrm{r}$ with $k_\mathrm{r} = \frac{L_\mathrm{m}}{L_\mathrm{r}}$, $\tau_\mathrm{r} = \frac{L_\mathrm{r}}{R_\mathrm{r}}$ and $\sigma = 1 - \frac{L_\mathrm{m}^2}{L_\mathrm{s}L_\mathrm{r}}$.

The mechanical machine torque is given by

$$T_\mathrm{m} = \frac{3}{2}p\left(\boldsymbol{\psi}_\mathrm{s} \times \boldsymbol{i}_\mathrm{s}\right) = \frac{3}{2}p\left(\boldsymbol{\psi}_\mathrm{r} \times \boldsymbol{i}_\mathrm{r}\right). \tag{2.36}$$

Finally, the mechanical differential equation can be stated:

$$\frac{\mathrm{d}\omega_\mathrm{m}}{\mathrm{d}t} = \frac{1}{J}\left(T_\mathrm{m} - T_\mathrm{l}\right) \tag{2.37}$$

where T_l is the mechanical load torque and J the inertia.

2.6 State of the art in control of electrical drive systems

The following section gives a brief overview of the state of the art in control of power inverters and electrical drives.

2.6.1 Conventional controllers and Pulse Width Modulation

Figure 2.13 shows the structure of a one-dimensional control system: The plant output y (which should be controlled) is measured by a sensor and compared to its reference value y^*. The control error $e = y^* - y$ is fed to the controller. The actuator will then produce an output corresponding to the controller output and apply it to the plant.

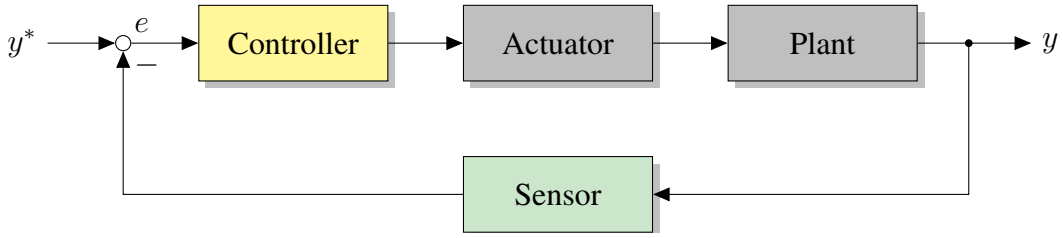


Figure 2.13: Control structure

Nearly all industrial applications use conventional PID controllers. These linear controllers consist of a proportional (P), an integral (I) and a differential (D) part. The continuous-time differential equation for the PID controller output is given by

$$y_{\text{pid}} = K_{\text{p}} e + K_{\text{i}} \int e \, \mathrm{d}t + K_{\text{d}} \frac{\mathrm{d}e}{\mathrm{d}t} \tag{2.38}$$

where e is the control error, y_{pid} is the controller output and K_{p}, K_{i} and K_{d} are the proportional, integral and differential gains, respectively. PID controllers can be implemented in digital as well as in analog control systems. Many improvements and adaptive schemes are known in order to optimize the controllers for specific tasks. For a sake of brevity, only the two most important extensions which are essential for real implementations of PID controllers are mentioned in the following:

1. As all actuators can only operate within certain boundaries, i.e. their output values are limited, this output limitation also has to be implemented in the PID controller.

2. If the output limitation of a PID controller is active, the integral of the control error can become very large and then lead to instabilities of the whole system. A so-called anti-windup strategy is normally implemented in order to avoid these problems: As soon as the output limitation is active, the integration will be stopped until the controller output is again within its limits.

These two extensions lead to additional nonlinearities.

By taking a closer look at equation (2.38), it is obvious that a PID controller produces a continuous-valued output. In the field of power electronics and electrical drives it is necessary

that the actuators deliver continuous-valued outputs. However, this requirement is only fulfilled for linear regulators which produce very high losses. All modern power inverters today consist of switching elements which means that their outputs are discrete-valued. Thus, an additional element to "synthesize" a continuous-valued output is necessary.

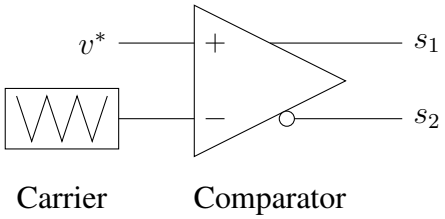


Figure 2.14: Technical realization of PWM for a two-level inverter leg

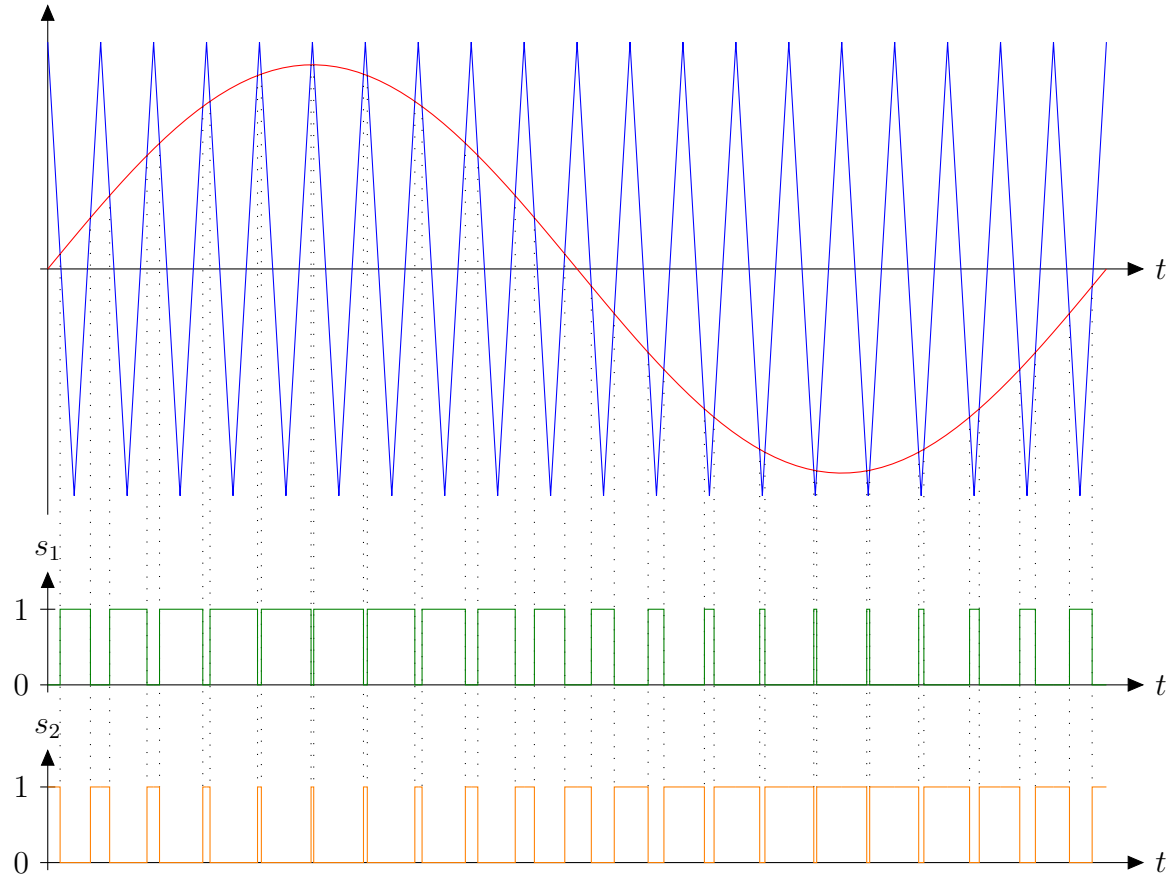


Figure 2.15: PWM example for a sinusoidal reference and a triangular carrier

The method to synthesize a continuous-valued output with an actuator that can only deliver a certain number of discrete output values is called Pulse Width Modulation (PWM). PWM is widely used in the area of power electronics and electrical drives. The basic idea is that a commanded value at the input leads to a corresponding proportionally related *on*-time of a switch. If the minimum value is commanded, the switch will be *off* for the whole time, whereas

it will be always *on* if the maximum value is demanded. Assuming that a PWM cycle is short enough, i.e. the switching frequency is sufficiently high, it can be assumed that the *average* value at the output of the actuator, driven by the PWM, is equal to the commanded one.

For a simple two-level inverter leg, as shown in Figure 2.3, a typical technical realization of PWM is shown in Figure 2.14. The basic principle of a technical PWM realization is the comparison of the reference voltage v^* to a sawtooth or triangular carrier signal. As soon as the reference is greater than the carrier, the comparator delivers a positive output, i.e. the corresponding switch is turned *on*. For the case of a two-level inverter, the carrier signal alternates between the boundaries $-0.5V_{\mathrm{dc}}$ and $0.5V_{\mathrm{dc}}$. If the reference voltage is $v^* = 0\,\mathrm{V}$, this results in an *on*-time for S_1 of half the carrier period or in a 50% duty cycle.

Figure 2.15 shows an example of PWM for a sinusoidal reference and with a triangular carrier for a system like in Figure 2.3. As it can be seen, the gating signals s_1 and s_2 for both semiconductor switches S_1 and S_2 are complementary.

For further and more detailed information about PWM, especially regarding three-phase (multilevel) inverters, the interested reader is referred to [14].

2.6.2 Field Oriented Control

2.6.2.1 Basic principle

The basic idea of Field Oriented Control (FOC) [15] is to describe an induction machine in a dq coordinate system whose d axis is aligned with the rotor flux. In this coordinate system the rotor flux is given by

$$\psi_{\mathrm{r}d} = |\boldsymbol{\psi}_{\mathrm{r}}| \text{ and} \tag{2.39}$$

$$\psi_{\mathrm{r}q} = 0\,\mathrm{Wb}, \tag{2.40}$$

i.e. the q component of the rotor flux is zero. By substituting this result into the IM equations (2.29) – (2.32) and after some further calculations, the rotor flux magnitude results to

$$\psi_{\mathrm{r}d} + \tau_{\mathrm{r}}\frac{\mathrm{d}\psi_{\mathrm{r}d}}{\mathrm{d}t} = L_{\mathrm{m}} i_{\mathrm{s}d}. \tag{2.41}$$

This means that in the rotor flux aligned dq coordinate system the rotor flux magnitude is only dependent on the d component of the stator current, $i_{\mathrm{s}d}$.

If equation (2.36) is written in the dq coordinate system, the following expression can be obtained:

$$T_{\mathrm{m}} = \frac{3}{2}p\left(\psi_{\mathrm{r}d} i_{\mathrm{r}q} - \psi_{\mathrm{r}q} i_{\mathrm{r}d}\right) = \frac{3}{2}p\left(\psi_{\mathrm{r}d} i_{\mathrm{r}q}\right). \tag{2.42}$$

The machine torque T_{m}, if calculated in dq coordinates, then only depends on the rotor flux magnitude and on the q component of the rotor current. Furthermore, by rewriting equation (2.30) in dq coordinates and considering equation (2.40), the q component of the rotor current results to

$$i_{\mathrm{r}q} = -\frac{L_{\mathrm{m}}}{L_{\mathrm{r}}} i_{\mathrm{s}q}. \tag{2.43}$$

By inserting equation (2.43) into equation (2.42), the final expression for the machine torque can be calculated:

$$T_{\mathrm{m}} = \frac{3}{2}\frac{L_{\mathrm{m}}}{L_{\mathrm{r}}} p\psi_{\mathrm{r}d} i_{\mathrm{s}q}. \tag{2.44}$$

From equations (2.41) and (2.44) it becomes clear that a transformation of the IM equations to a rotor flux aligned dq coordinate system makes it possible to control both flux and torque independently, similarly to a separately excited DC machine.

2.6.2.2 Rotor flux estimation

Synchronous machines rotate with the same frequency as the applied stator voltage. For these machines the rotor flux angle is equal to the electrical machine angle. Thus, for synchronous machines the rotor flux angle can simply be measured with an encoder or a resolver.

In contrast to that, in an induction motor the electrical machine speed is (in motor operation) always lower than the frequency of the applied stator voltage, i.e. there is a *slip* between these two speeds. Consequently, the rotor flux angle is not equal to the electrical machine angle. Because of this induction machines are also called *asynchronous* machines and it is mandatory to implement a rotor flux estimator. Otherwise, a decoupled control of both flux and torque is not possible. Basically, two different types of estimators have been proposed [16]:

1. Open-loop estimators: Estimators without feedback and
2. closed-loop estimators with feedback.

Although the latter ones provide a better and more robust flux estimation, mostly open-loop estimators are used as these can be implemented easier. In the following only the two most important open-loop estimators [17] are briefly introduced and explained.

One very common estimator is the so-called *voltage model*. According to equation (2.31), the stator flux $\boldsymbol{\psi}_\mathrm{s}$ can be calculated for $\omega_\mathrm{k} = 0$:

$$\frac{\mathrm{d}\boldsymbol{\psi}_\mathrm{s}}{\mathrm{d}t} = \boldsymbol{v}_\mathrm{s} - R_\mathrm{s}\boldsymbol{i}_\mathrm{s} \tag{2.45}$$

The rotor flux $\boldsymbol{\psi}_\mathrm{r}$ can be calculated by substituting equation (2.29) into equation (2.30):

$$\boldsymbol{\psi}_\mathrm{r} = \frac{L_\mathrm{r}}{L_\mathrm{m}} \cdot \boldsymbol{\psi}_\mathrm{s} - \left(\frac{L_\mathrm{r} L_\mathrm{s}}{L_\mathrm{m}} + L_\mathrm{m} \right) \cdot \boldsymbol{i}_\mathrm{s} \tag{2.46}$$

For the implementation of the voltage model on a real-time computer system, equations (2.45) and (2.46) have to be discretized with the sampling time T_s. The two main problems of the voltage model are the open-loop integration (without feedback) which can lead to convergence problems (especially at lower speeds) and the stator resistance R_s which changes its value during the operation of the machine: Due to the temperature rise its real value can differ more than 50% from its nominal value.

Another possibility to estimate the rotor flux is the so-called *current model*. For this estimator equation (2.35) is used with $\omega_\mathrm{k} = 0$:

$$\boldsymbol{\psi}_\mathrm{r} + \tau_\mathrm{r} \frac{\mathrm{d}\boldsymbol{\psi}_\mathrm{r}}{\mathrm{d}t} = L_\mathrm{m}\boldsymbol{i}_\mathrm{s} + j\omega_\mathrm{el}\tau_\mathrm{r}\boldsymbol{\psi}_\mathrm{r} \tag{2.47}$$

After discretization with the sampling time T_s, equation (2.47) can be used for estimating the rotor flux. In contrast to the voltage model there is no open-loop integration which can lead to stability problems. For this equation the electrical machine speed, ω_el, is also needed. However,

even this estimator is still parameter-dependent: The rotor time constant $\tau_\mathrm{r} = \frac{L_\mathrm{r}}{R_\mathrm{r}}$ is dependent on the rotor resistance R_r. If the machine is operated, its temperature rises which increases R_r.

In FOC mostly the current model is used for the rotor flux estimation.

When the rotor flux is determined, the rotor flux angle can then be calculated easily:

$$\varphi = \arctan\left(\frac{\psi_{\mathrm{r}\beta}}{\psi_{\mathrm{r}\alpha}}\right) \quad [1] \tag{2.48}$$

2.6.2.3 FOC block diagram

A block diagram of the basic FOC structure is shown in Figure 2.16. As for normal machine

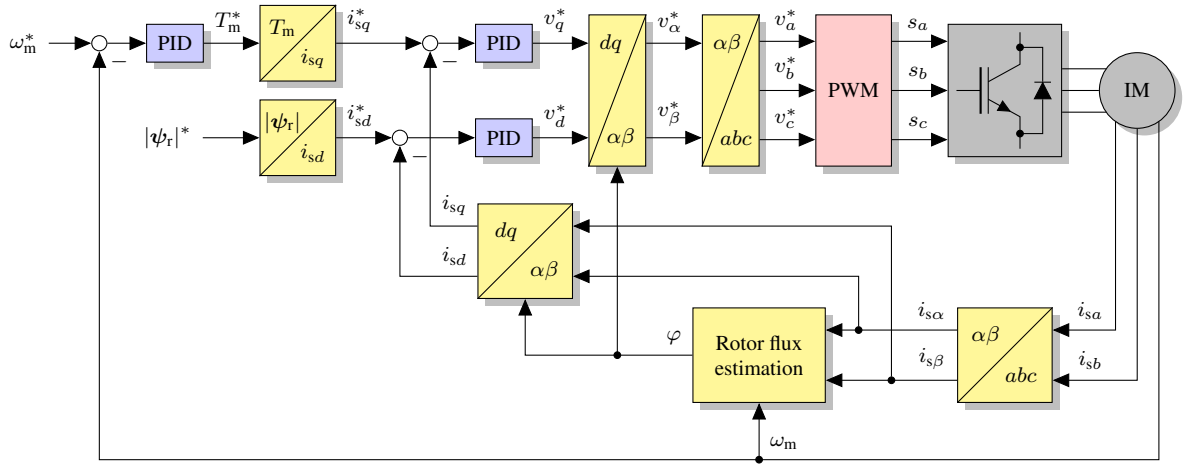


Figure 2.16: Basic FOC scheme

operation the sum of all three phase currents is equal to zero (equation (2.12)), it is enough to measure only two phase currents (in this case i_{sa} and i_{sb}). After applying the Clarke transformation, the currents $i_{s\alpha}$ and $i_{s\beta}$ can be obtained. Then, the rotor flux can be estimated according to chapter 2.6.2.2 and the rotor flux angle φ can be calculated. After that the measured currents are transformed to dq coordinates. The torque reference is calculated by a speed PID controller; usually, the reference for the rotor flux magnitude is set to a fixed value which is decreased when field-weakening operation is desired. From the rotor flux magnitude reference and the torque reference the current references i_{sd}^* and i_{sq}^* are calculated according to equations (2.41) and (2.44). The differences between the reference and measured d and q components of the stator current are then fed to current PID controllers. These deliver the reference voltages v_d^* and v_q^*. After two further transformations, from dq to $\alpha\beta$ and then to abc coordinates, the reference voltages for all three phases are fed to a modulator which then produces the switching states s_a, s_b and s_c that are fed to the inverter.

[1] As $\arctan(x/y)$ gives no information in which quadrant φ lies, for computer implementations normally $\mathrm{atan2}(x, y)$ is used.

2.6.3 Direct Torque Control

Another very well-known method which allows separate control of both flux and torque is called Direct Torque Control (DTC) [18, 19]. In contrast to FOC, the control is completely done in stator coordinates. Furthermore, it is a *direct* control strategy, i.e. it generates the inverter gating signals directly and a modulator is not necessary.

2.6.3.1 Basic principle

Figure 2.17 shows the basic DTC scheme. Equal to FOC, only two stator currents have to be

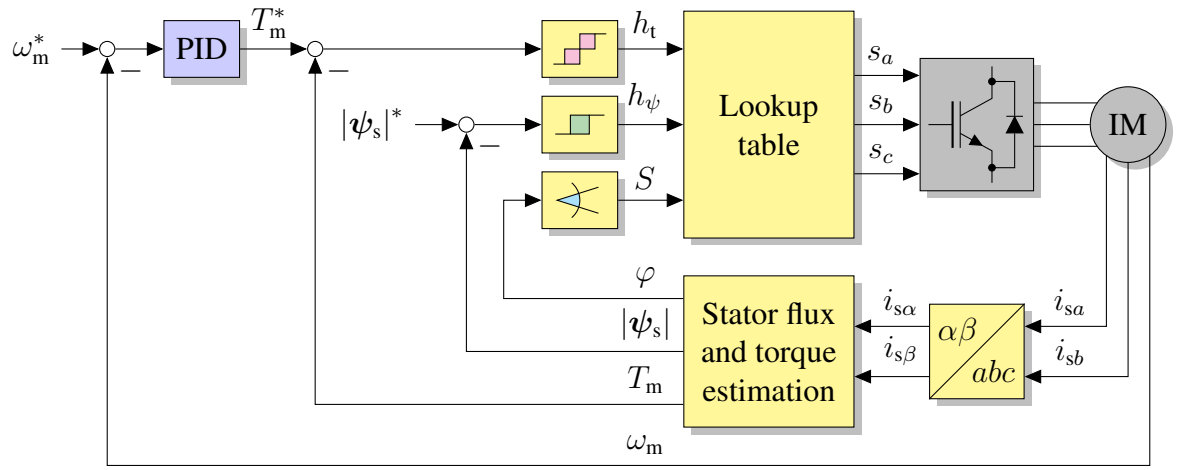


Figure 2.17: Basic DTC scheme

measured as the three-phase current system is balanced. When the Clarke transformation is applied to the measured phase currents i_{sa} and i_{sb}, the currents $i_{s\alpha}$ and $i_{s\beta}$ can be obtained. As the whole controller is implemented in stator coordinates, no further coordinate transformation is necessary. The stator flux $\boldsymbol{\psi}_s$ is calculated using the voltage model (equation (2.45)). The stator flux magnitude can be calculated as

$$|\boldsymbol{\psi}_s| = \sqrt{\psi_{s\alpha}^2 + \psi_{s\beta}^2}. \tag{2.49}$$

The torque reference T_m^* is generated with a PID speed controller. The reference value for the stator flux magnitude, $|\boldsymbol{\psi}_s|^*$, can be set to a constant or—if field-weakening is desired—to a value smaller than its nominal one. The basic DTC controller only consists of a three-dimensional lookup table for the torque (h_t), the stator flux magnitude (h_ψ) and for the sector (S) in which the stator flux angle is located.

For the operation of DTC two hysteresis bands have to be set up: One for the torque and one for the stator flux magnitude. For the torque a double-band hysteresis controller with three different output values is used:

- -1: The machine torque has to be decreased.
- 0: The torque should be kept constant.
- 1: The torque must be increased.

For the stator flux magnitude only a single-band hysteresis controller is used which has two possible output values:

- -1: The stator flux magnitude should be decreased.
- 1: The stator flux magnitude has to be increased.

Finally, the stator flux angle is divided into six different sectors I...VI (every sector covers $60°$) which are shown in Figure 2.18.

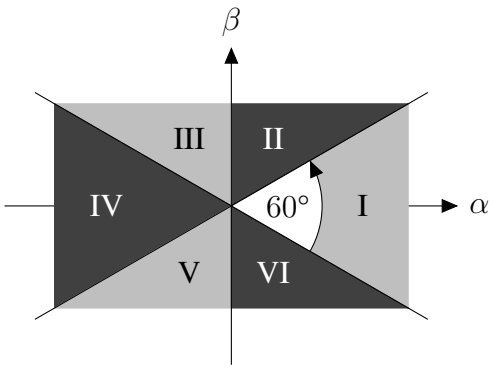


Figure 2.18: DTC stator flux angle sector distribution

2.6.3.2 Lookup table for Direct Torque Control

In the following a short explanation of the DTC lookup table is given. As already mentioned, the three-dimensional lookup table distinguishes between six different sectors. For every sector an optimum switching state must be determined if the stator flux magnitude should be increased or decreased and if the machine torque should be increased, decreased or be kept constant. In general, torque regulation has priority over flux regulation.

Setting $\omega_\mathrm{k} = 0$ in equation (2.31) and applying the Euler-forward discretization with the sampling time T_s, the following expression for the stator flux can be obtained:

$$\boldsymbol{\psi}_\mathrm{s}(k+1) = \boldsymbol{\psi}_\mathrm{s}(k) + T_\mathrm{s} \cdot \boldsymbol{v}_\mathrm{s} - \underbrace{R_\mathrm{s} T_\mathrm{s} \cdot \boldsymbol{i}_\mathrm{s}}_{\approx 0} \tag{2.50}$$

As the stator resistance R_s is usually small, it can be neglected for the following considerations. Then, the stator flux which will be obtained in the next sample is only dependent on its current value and on the applied stator voltage.

Another assumption can be made for the machine torque: If equation (2.46) is solved for $\boldsymbol{i}_\mathrm{s}$ and replaced into equation (2.42), the machine torque T_m can be calculated as

$$T_\mathrm{m} = \frac{3}{2} p \cdot \frac{L_\mathrm{m}}{\sigma L_\mathrm{s} L_\mathrm{r}} \cdot (\boldsymbol{\psi}_\mathrm{r} \times \boldsymbol{\psi}_\mathrm{s}) = \frac{3}{2} p \cdot \frac{L_\mathrm{m}}{\sigma L_\mathrm{s} L_\mathrm{r}} \cdot |\boldsymbol{\psi}_\mathrm{r}|\,|\boldsymbol{\psi}_\mathrm{s}| \sin(\epsilon) \tag{2.51}$$

where ϵ is the angle between the rotor and the stator flux. Simplified, it can be stated that

$$T_\mathrm{m} \propto \sphericalangle(\boldsymbol{\psi}_\mathrm{r}, \boldsymbol{\psi}_\mathrm{s})\,. \tag{2.52}$$

As the rotor flux $\boldsymbol{\psi}_\mathrm{r}$ changes slowly compared to the stator flux $\boldsymbol{\psi}_\mathrm{s}$, it can be seen as constant. Then, from equation (2.52) the following assumption can be made:

$$T_\mathrm{m} \propto \delta = \sphericalangle\left(\boldsymbol{\psi}_\mathrm{s}(k), \boldsymbol{\psi}_\mathrm{s}(k+1)\right) \tag{2.53}$$

If the machine torque T_m should be increased, the angle δ between $\boldsymbol{\psi}_\mathrm{s}(k)$ and $\boldsymbol{\psi}_\mathrm{s}(k+1)$ has to be increased and vice versa. This principle is visualized in Figure 2.19 for sector I. If e.g.

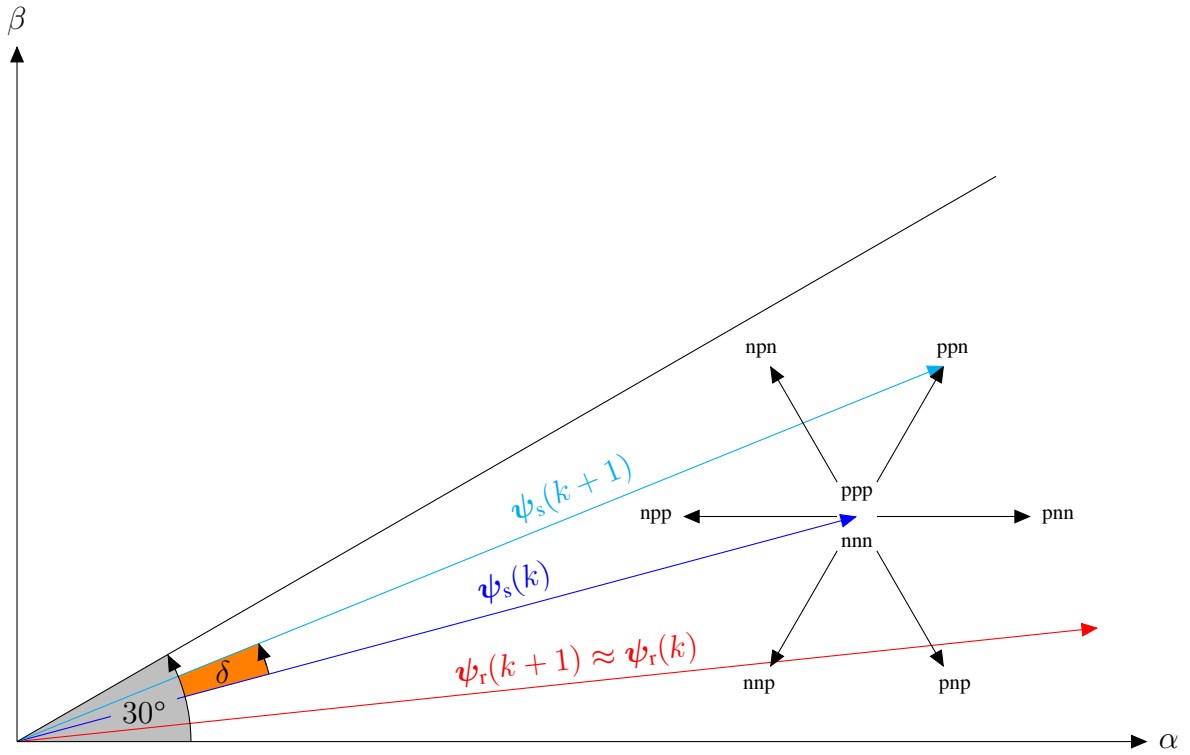


Figure 2.19: Determination of the DTC lookup table for sector 1

the stator flux magnitude should be increased ($h_\psi = 1$), i.e. the length of the vector $\boldsymbol{\psi}_\mathrm{s}(k+1)$ should be greater than the length of $\boldsymbol{\psi}_\mathrm{s}(k)$, the switching states ppn, pnn or pnp could be used. If now also the torque has to be increased ($h_\mathrm{t} = 1$), the angle δ should additionally become as large as possible. For this reason, for $h_\psi = h_\mathrm{t} = 1$ the switching state ppn is selected. If the torque should be kept constant, i.e. $h_\mathrm{t} = 0$, a zero switching state is the best choice. The decision between nnn and ppp is made under the assumption that h_t has changed its value from 1 to 0 or from -1 to 0. Then, the zero switching state leading to less transitions is chosen. For all other sectors and hysteresis controller outputs analog considerations can be made.

These brief explanations should just give a rough overview of the DTC operating principle. Many improvements and different DTC algorithms have been proposed, e.g. Direct Mean Torque Control (DMTC) [20] or Model Predictive Direct Torque Control (MPDTC) [21, 22].

2.6.4 Discussion and evaluation

Both FOC and DTC are the two mostly used control schemes for variable-speed induction motors. With some adjustments these methods can also be applied to synchronous machines.

Table 2.4: Lookup table for DTC

h_ψ	h_t	S_I	S_II	S_III	S_IV	S_V	S_VI
1	1	ppn	npn	npp	nnp	pnp	pnn
1	0	ppp	nnn	ppp	nnn	ppp	nnn
1	−1	pnp	pnn	ppn	npn	npp	nnp
−1	1	npn	npp	nnp	pnp	pnn	ppn
−1	0	nnn	ppp	nnn	ppp	nnn	ppp
−1	−1	nnp	pnp	pnn	ppn	npn	npp

In general, FOC is known to deliver very good control results in steady-state. Because of the modulator the controlled variables show less ripples compared to the results that can be obtained by DTC. In contrast to that, DTC is known for its better transient behavior since it is a direct switching strategy.

FOC is mostly used for low-voltage drives: As a modulator is used, the inverter switching frequency is constant and usually much higher than for direct switching strategies. For medium- and high-voltage drives special adjustments must be made in order to reduce inverter switching losses. In contrast to that, DTC is mostly used for medium- and high-voltage drives. For these applications low switching frequencies are mandatory and less important than higher ripples on the controlled variables. Due to the variable switching frequency, DTC also produces an undesired audible noise compared to FOC.

Another remark has to be made regarding the implementation of both strategies: For a successful implementation of hysteresis controllers either analog controllers or digital controllers with very high sampling rates are mandatory. Implementing DTC on a software-based control platform (e.g. a microcontroller) can become complicated because sampling rates of about $40\,\mathrm{kHz}$ and above are necessary. Otherwise, the time delay which is inherent to software-based real-time implementations, has to be compensated in order to achieve an acceptable control result. Thus, DTC normally has to be implemented directly in hardware. This issue will be explained more detailed in chapter 3. In contrast to that, FOC can be implemented in hardware as well as in software. Furthermore, digital implementations of hysteresis controllers suffer from another drawback: As measurements and switchings can only take place at the sampling instances, a digitally implemented hysteresis controller can only react if its boundaries are already violated. The smaller the sampling time T_s is, the smaller are these boundary violations.

Nowadays, these two main control concepts are in most cases still sufficient to fulfill their static and dynamic requirements. However, as especially the field of nonlinear control is becoming more and more popular and since the available computational power of microprocessors and computers rises, more and more research is done regarding general improvements of these basic control concepts. Especially MPC concepts are becoming more and more popular and might lead to a change to more sophisticated controllers in this field.

2.7 Model Predictive Control

MPC methods belong to the wide class of predictive or precalculating controllers. MPCs can be considered as a sub-class of Predictive Control (PC) in general. A good overview of PC for electrical drive systems can be found in [23] and [1]. A detailed overview of MPC algorithms is given in [24] and [25].

2.7.1 Basic idea

MPC is a wide class of different control algorithms which share the following basic concepts:

- An MPC uses a system model in order to calculate optimum values for the actuating variables.
- An MPC calculates the system's behavior for a sequence of values for the actuating variables up to a certain *prediction horizon*.
- An MPC determines the optimum sequence of values for the actuating variables by minimizing a so-called *cost function*.

The cost function usually contains the control deviation. Additionally, other terms, e.g. for penalizing high switching frequencies or large changes of the actuating variables, can also be included. In general, a higher prediction horizon leads to a better control result but also increases the number of calculations for finding the optimum sequence of values for the actuating variables.

If prediction horizons of more than one sampling cycle are used, the *receding horizon principle* [26] is applied: When the control algorithm is executed, the optimum sequence of values for the actuating variables is calculated for the whole prediction horizon. However, only the optimum values for the next sample are forwarded to the actuator (the other ones are still in the future).

In contrast to that, (linear) PID controllers calculate the reference value for the actuator based on an amplification of the control error. Although a system model is normally necessary for the controller tuning, i.e. tuning of the proportional (P), integral (I) and differential (D) gains, the controller can operate without any knowledge of the system itself.

In general, MPC has several advantages over conventional control methods. The most important ones are

- Control of multidimensional systems, i.e. of systems with more than one actuating variable, is possible with a single controller.
- Constraints can be considered within the model and in the control design.
- The cost function can be designed according to the user's needs.
- The control of linear as well as nonlinear systems is possible.

All these points clearly show that MPC offers more possibilities and flexibility compared to classical controllers.

2.7.2 Historical background

The first MPC algorithms have been developed in the 1960s [27]. From the beginning the high calculation effort which is necessary to execute these controllers has been a problem for real-time implementations of MPC algorithms.

Since the 1980s, however, MPC methods are already widely used in chemical and process engineering [24]: The plants in these industries have time constants which are in the range of seconds, minutes and sometimes even hours. Even in the 1980s the available calculation power was already enough for a real-time execution of MPC algorithms for such systems.

2.7.3 MPC for power electronics and electrical drive systems

Compared to chemical and process engineering, the time constants for power electronics and electrical drive systems are in the range of milli- and microseconds. For this reason much higher sampling rates are necessary for controllers in this field which also leads to much higher hardware requirements. However, according to Moore's law [28], the number of components in integrated circuits doubles every two years and with it the available computational power in microprocessors. This means that MPC, even in the field of power electronics and electrical drives, will become feasible on standard industrial controllers. Because of this, MPC for electrical drive systems has become more popular in the research community in the last years.

2.7.4 MPC with continuous-valued input sets

Most MPCs use a continuous-valued control set, i.e. it is assumed that the actuator can, at least within certain boundaries, deliver a continuous-valued output. One very famous MPC method which utilizes a continous-valued input set is Generalized Predictive Control (GPC) [29, 30].

However, as already mentioned in chapter 2.6.1, nearly all actuators in the field of power electronics and electrical drive systems have a discrete-valued nature which means that, in the same way as for conventional controllers, PWM is still necessary.

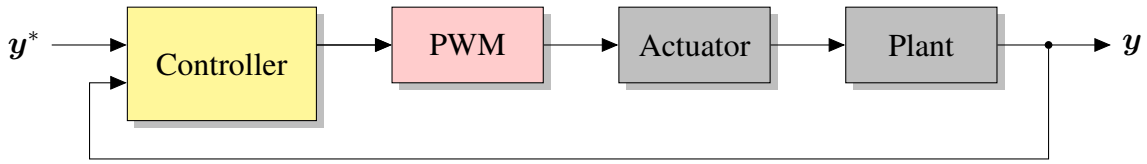
2.7.5 MPC with finite input sets

In contrast to the previously mentioned MPC methods for continuous-valued input sets, there are also MPC algorithms for finite input sets, i.e. methods which explicitly consider the discrete nature of the actuator. In technical literature these methods are often referred to as Finite-Set MPC (FS-MPC).

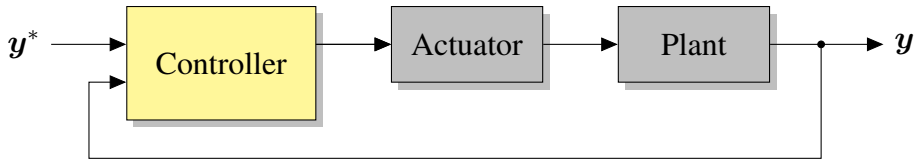
2.7.5.1 Basic principle

As already mentioned, the basic principle of FS-MPC methods is to explicitly consider the discrete nature of the actuator. As it can be seen in the block diagram in Figure 2.20, a modulator is not necessary for these methods: In contrast to continuous-valued input sets, FS-MPC delivers an optimum switching state which can then directly be forwarded to the inverter.

For finite input sets an analytical minimization of the cost function, compared to continuous-valued input sets, is in general not possible. This means that the cost function has to be calculated for every possible sequence of values for the actuating variables and then the sequence that



(a) PWM control structure



(b) Direct (finite-set) control structure

Figure 2.20: PWM and direct control structures

leads to the smallest value for this cost function has to be chosen. Assuming a simple two-level inverter leg and a prediction horizon of one sample, the cost function (for reasons of simplicity assumed to be only the control deviation) has to be calculated for both possible switching states. The switch position that leads to a smaller cost function value will be chosen and applied to the inverter.

This generic and very simple method can be easily extended to a higher prediction horizon: If the system's behavior should be optimized for two steps in advance, $2 \cdot 2 = 4$ possible combinations of switching states have to be evaluated. It is also possible to extend this principle to more switching possibilities (e.g. seven different voltage vectors for a three-phase two-level inverter). The number of possible trajectories that have to be evaluated for an actuator with m different switching states and for a prediction horizon of n sampling cycles is given by

$$N_\mathrm{t} = m^n \tag{2.54}$$

where N_t is the number of possible trajectories which have to be predicted. The calculation effort for FS-MPC algorithms rises exponentially with the prediction horizon. This means that a real-time implementation of FS-MPC methods, especially for multilevel inverters and for higher prediction horizons, needs very powerful and fast controllers or more sophisticated strategies to find the optimal trajectory.

2.7.5.2 General algorithm flow graph

In order to illustrate the basic principle which was already roughly described in chapter 2.7.5.1, Figure 2.21 shows a general flow graph of an FS-MPC algorithm. Without loss of generality it is only shown for one prediction step.

At the beginning of a new sampling cycle all measurements are done. In the whole i_sw switching states have to be tested. At the beginning the counter i for the switching state is initialized with 0, the optimal switching state that minimizes the cost function is set to the invalid value $s_\mathrm{opt} = 0$ and the minimum of the cost function is set to $j_\mathrm{min} = \infty$. Then, the enumeration starts: The first switching state $i = 1$ is tested, i.e. the system model is predicted assuming that s_i is applied and the corresponding cost function value j_i is calculated. If j_i is smaller than the current cost function value j_min, it is stored as the new minimum and the

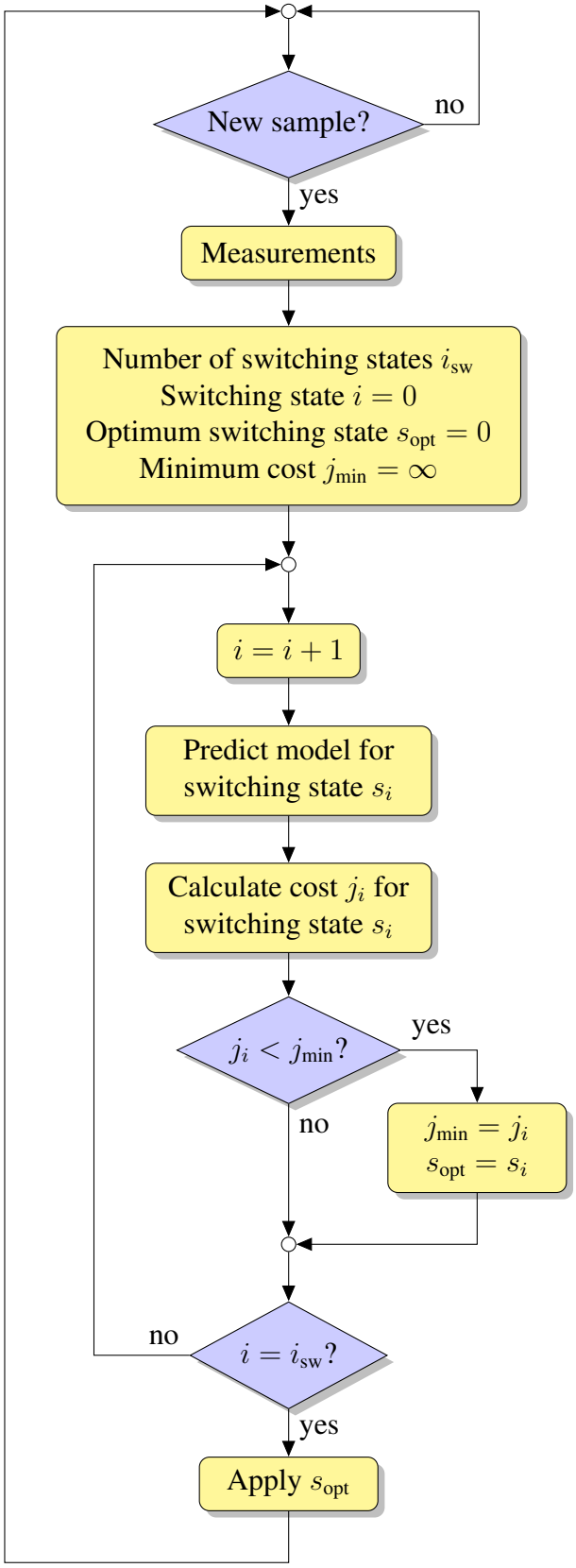


Figure 2.21: General FS-MPC algorithm flow graph

corresponding switching state s_i is saved as the optimum switching state s_{opt}. Afterwards, the counter for the possible switching states, i, is increased and the same procedure is repeated until i is equal to the number of possible switching states, i_{sw}. Then, the optimization process is finished and the optimal switching state s_{opt} is applied to the system. When a new sample starts, the whole algorithm is repeated for the new measurements.

2.7.5.3 Advantages

FS-MPC methods offer several advantages which can—if at all—only hardly be achieved with conventional control schemes. The most important advantages are

1. The control of nonlinear systems is possible as long as the nonlinearities can be modeled; linearization (e.g. at certain operating points) or nonlinear optimization is not necessary.

2. FS-MPC controllers allow multivariable control, e.g. one controller can be used to control both flux and torque or two currents simultaneously.

3. Constraints (e.g. of the inverter output voltages or current limitations) can be easily included into the cost function.

4. A modulator which leads to further nonlinearities is not necessary.

These four points have to be considered in the field of power electronics and electrical drives. Especially for linear schemes these effects lead to non-optimal control results. Thus, FS-MPC methods are very promising candidates for such applications.

2.7.5.4 Problems

As already mentioned in chapter 1, the two major drawbacks of FS-MPC methods are the calculation effort which rises exponentially with the prediction horizon and the low time resolution as switchings can only take place at the sampling instances.

As mentioned before, for an actuator with m different switching states and a prediction horizon of n samples, the number of possible trajectories which need to be evaluated can be calculated according to equation (2.54).

Especially for multilevel inverters the calculation effort rises very quickly with the prediction horizon because of the higher number of switching states m. For a three-phase two-level inverter eight different switching states are possible, a three-level NPC inverter already has 27 different switching states and in case of a three-level FC inverter 64 combinations are possible. Table 2.5 shows the number of trajectories N_{t} for these three different types of three-phase inverters, depending on the prediction horizon n. Usual controller frequencies are in the range of $5\,\text{kHz}$ to $40\,\text{kHz}$, i.e. the corresponding sampling times are in the range of $25\,\mu\text{s}$ to $200\,\mu\text{s}$. The evaluation of a trajectory requires several arithmetic operations (depending on the system and on the model). Thus, with currently available controllers only for a two-level three-phase inverter more than one prediction step is feasible in real-time. If higher prediction horizons are necessary, the calculation effort needs to be drastically reduced.

Of course, compared to a simple *complete enumeration*, several strategies to reduce the calculation effort can be applied:

Table 2.5: Number of trajectories for different inverter types and prediction horizons

Inverter type	Prediction horizon		
	1	2	3
Two-level inverter	8	64	512
Three-level NPC inverter	27	729	19,683
Three-level FC inverter	64	4,096	262,144

1. From operations research strategies like *branch and bound* [31] or *branch and cut* [32] are known. Such strategies can significantly reduce the *average* necessary calculation effort. In the worst case the calculation effort cannot be reduced for an iteration. For real-time applications, however, this *worst case* has to be considered since the algorithm has to be finished within one sampling cycle. Thus, these strategies cannot be used for such applications.

2. With the help of *Multiparametric Programming* [33–35] it is possible to solve the optimization problem *offline* with the state vector as a parameter. In this way the calculation effort can be significantly reduced. However, for practical realizations with multilevel inverters and for prediction horizons greater than one sample, the resulting offline calculation effort also increases exponentially with the prediction horizon: Even for a simple three-level inverter and for two prediction steps a standard personal computer needs several hours up to two or three days to calculate one controller partition. This dramatically reduces the applicability of Multiparametric Programming for FS-MPC.

3. Natural switching constraints (i.e. not all possible transitions are allowed if the inverter is in a certain switching state) can be used to significantly reduce the number of necessary calculations. If such natural constraints are considered, higher prediction horizons can be realized. However, even in this case the *worst case* has to be considered for a real-time application. Another problem arises from the fact that natural switching constraints are only valid for one certain type and size of inverter. Thus, no general simplifications can be made.

Additionally to the calculation effort, FS-MPC methods suffer from another important drawback which is—in simplified form—visualized in Figure 2.22: In this case a discrete actuator with two different switching states is assumed. Furthermore, it is assumed that these two different switching states produce two different slopes of the controlled variable y. Figure 2.22 shows the number of samples T_s on its horizontal axis whereas the controlled variable is drawn vertically. If an FS-MPC method is applied, the switching state can only be changed at the beginning of a sampling interval. The switching state which leads to a high positive slope (from now on referred to as *active* switching state) has to be applied for the whole sample T_s. Assuming steady-state operation, the other switching state which leads to a small negative slope (*zero* switching state) has to be applied until y has become small enough again. This leads to high ripples on the controlled variables. Although for FS-MPC the theoretical maximum switching frequency would be half the sampling frequency, it is in fact far lower than this value. In

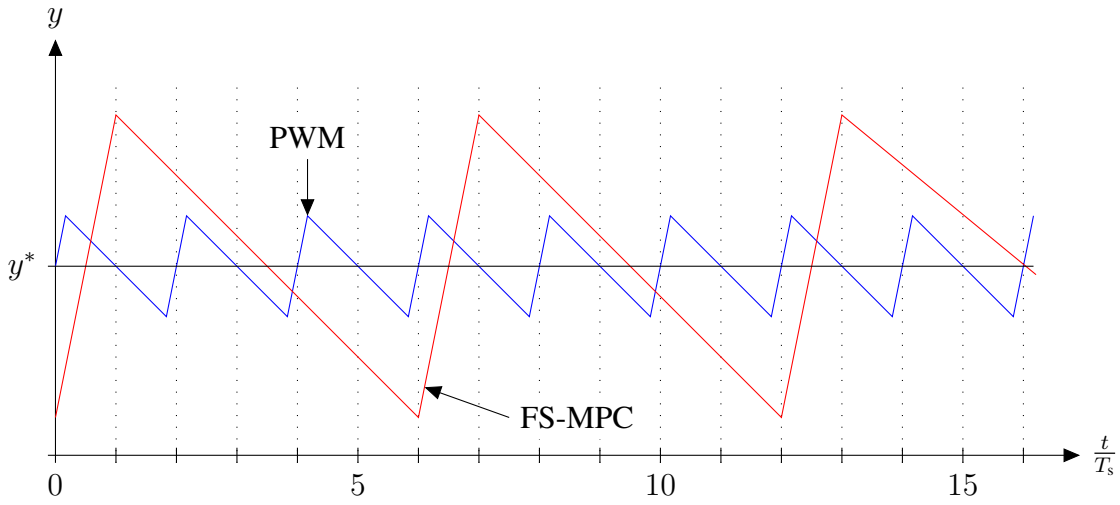


Figure 2.22: Time resolution of FS-MPC methods compared to PWM

contrast, PWM has a *much* higher time resolution and allows switching events to take place *in between* the samples. Because of this it is easily possible to apply the active switching state for a much shorter time compared to FS-MPC. The reference value for the modulator is in most cases not saturated. This means that the carrier signal will cross it once in a sample (Figure 2.15) which leads to one switching event per semiconductor device and sample. Thus, the switching frequency of PWM-based methods is constant and equal to half the sampling frequency.

Furthermore, for three-phase inverters FS-MPC leads to a *variable* switching frequency which produces an undesired audible noise—besides the high ripples on the controlled variables.

2.8 Goals of this work

As already stated, despite of all the advantages mentioned in chapter 2.7.5.3, the two main drawbacks of FS-MPC methods which were explained in detail in chapter 2.7.5.4 still limit their applicability to industrial control systems. For medium- and high-voltage drives low switching frequencies are necessary because of the high switching losses whereas for low-power systems and applications a good quality of the controlled variables and small ripples are desired. Nevertheless, even for high-power applications a reduction of ripples can lead to significant improvements.

The most important drawback of FS-MPC methods, however, is the tremendous calculation effort, especially for the case of higher prediction horizons and for multilevel inverters. If the calculation effort can be reduced, this can lead to a wider acceptance of FS-MPC methods.

The main goal of this work is to present solutions for these two problems, with special attention to three-level NPC and FC inverters. The three main goals of this work are:

1. The development of a heuristic method which significantly reduces the calculation effort: The calculation effort for this method is independent of the number of voltage levels in the inverter which is promising especially for the application to multilevel inverters.

Although the calculation effort still rises exponentially with the prediction horizon, this method allows prediction horizons of up to three sampling cycles on the used real-time computer hardware, even for three-level inverters. The basic principle of this heuristic method is described and introduced in chapter 5.2.

2. Strategies to reduce ripples on the controlled variables by increasing the time resolution of FS-MPC algorithms: Methods to calculate variable switching points (VSPs) are introduced in chapters 4.2 and 5.3. Furthermore, an oversampling approach with a higher time resolution is presented and evaluated in chapter 6.

3. A combination of the previous two points, i.e. a method to reduce ripples on the controlled variables with less calculation effort which is introduced and explained in chapter 5.4.

CHAPTER 3

Real-time implementation of control algorithms

Control algorithms can be implemented on different control hardware. The first controllers were analog implementations of PID controllers with operational amplifiers and also PWM had to be implemented with many discrete components. Nowadays, nearly all control systems are implemented digitally and analog hardware is only used when absolutely necessary.

Basically, even regarding today's digital control platforms, controllers can be divided into hardware- and software-based implementations with special requirements, advantages and drawbacks. The aim of this chapter is to compare both possible implementations and to explain which strategies are better suited for implementation in hardware and which can be easier realized in software.

3.1 Software-based real-time implementation

Beginning in the 1970s, more and more control algorithms were implemented digitally with the help of microprocessors. Generally speaking, these processors are either triggered by an external interrupt which signalizes the processor that a new sampling cycle has started and that the control algorithm has to be executed, or by an internal timer that generates an interrupt. All measurements are sampled, the reference values can either be created internally in the controller or also be sampled from an external reference. Then, the processor executes the control algorithm, i.e. it does all necessary calculations. After that the processor outputs the updated references for the actuator and waits for the next interrupt signal.

This basic scheme describes the principle of a software-based implementation of a control algorithm.

3.1.1 Time delay compensation

One very important remark about software-based implementations has to be made: All digital control systems need a certain time for their calculations. For real-time operation it is manda-

tory that the calculations in one sample are finished *before* the next one starts. This real-time condition must not be violated in any case—otherwise this can lead to unpredictable results. If the behavior of a real-time system is compared to that of a simulation program, some very important implementation issues need to be considered. In Figure 3.1 the behavior of a simulation at the current sampling step k is visualized: Without loss of generality in this case it is—for

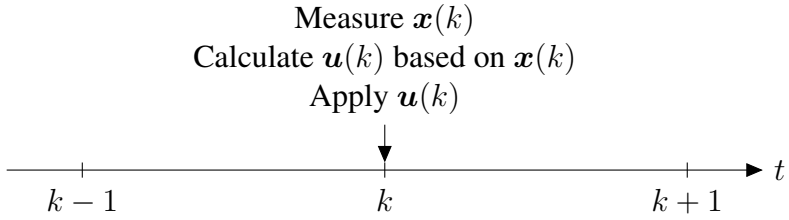


Figure 3.1: Behavior of a simulation, optimization for time step $k+1$

reasons of simplicity—assumed that the complete system state $\boldsymbol{x}$ is measured. As already mentioned, the actuating variable is denoted with $\boldsymbol{u}$. When the simulation time reaches sample k, the system state $\boldsymbol{x}(k)$ is measured. Then, updated values for the actuating variables $\boldsymbol{u}(k)$ are calculated at time step k as well. These updated values for $\boldsymbol{u}(k)$ are calcuated based on the measured system state $\boldsymbol{x}(k)$. The optimization is carried out for time step $k+1$. In contrast to real implementations, however, the simulation time is "stopped" until the calculation of $\boldsymbol{u}(k)$ has finished such that $\boldsymbol{u}(k)$ can be applied at step k. Thus, in simulations the calculation time is not considered.

The behavior of a real-time implementation is visualized in Figure 3.2: In contrast to a sim-

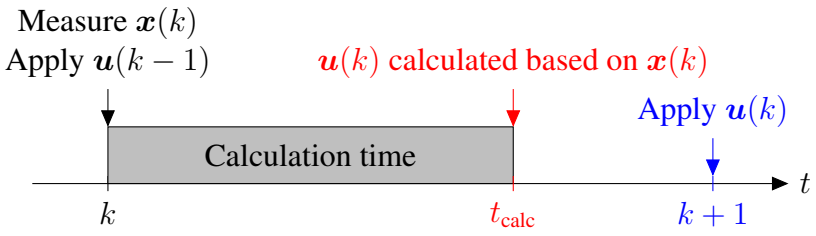


Figure 3.2: Behavior of a real-time system, optimization for time step $k+1$

ulation where the calculation time is not considered, the updated actuating variables, $\boldsymbol{u}(k)$, are delayed for one sample. At the beginning of the current sampling cycle k the system state $\boldsymbol{x}(k)$ is measured and the calculation of new values for the actuating variables is started, exactly as in a simulation. This calculation is finished at $k + t_{\text{calc}}$. As t_{calc} is usually not a fixed time, the updated values for the actuating variables which were calculated during the kth sample, are applied at time step $k+1$. Consequently, at step k the previously calculated values, $\boldsymbol{u}(k-1)$, are applied, i.e. the actuating variables are applied with a delay of *one* sample. This additional dead time has to be considered in the controller; otherwise, a non-optimal control or even instabilities can result from that.

Especially for MPC methods this time delay has to be considered explicitly in the controller. However, as in every step the values of the actuating variable $\boldsymbol{u}$ are known, this time delay

of one sample can be easily compensated by using a model-based approach. The principle is shown in Figure 3.3. In contrast to Figure 3.2, in a first step the system state at the next sample,

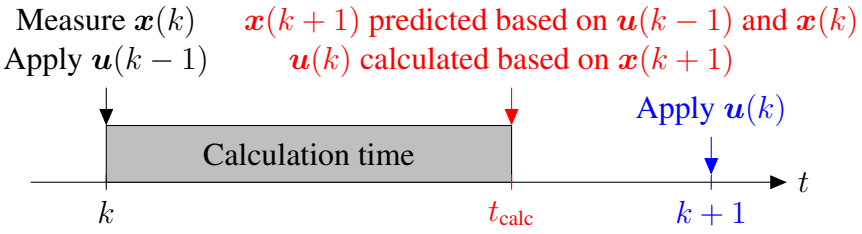


Figure 3.3: Model-based time delay compensation, optimization for time step $k+2$

$\boldsymbol{x}(k+1)$, is predicted for the currently applied values of the actuating variables, $\boldsymbol{u}(k-1)$ and for the measured system state $\boldsymbol{x}(k)$. $\boldsymbol{u}(k-1)$ is known since it was calculated in the previous sample. Thus, the system model has to be calculated once more before the actual optimization can be started. Then, in the optimization process, the actuating variable $\boldsymbol{u}(k)$ which will be applied at step $k+1$, is calculated. In contrast to Figure 3.2, however, it is calculated based on $\boldsymbol{x}(k+1)$ and not based on the actual system state $\boldsymbol{x}(k)$. Thus, the optimization is carried out for time step $k+2$. In this way the time delay which is present in every software-based digital control system can be compensated.

The described realistic behavior can be easily simulated with an additional delay after the controller output. Especially for hysteresis controllers this time delay can lead to huge violations of the hysteresis boundaries.

Further information about this principle can be found in [36].

3.1.2 Two-level inverter test bench

The two-level inverter test bench is shown in Figure 3.4. The real-time computer system used for this test stand is described in [37] with the only difference that a faster PC104 computer module with a 1.4 GHz Pentium M CPU is used. The details and parameters are given in appendix B.1. The test bench consists of two coupled 2.2 kW squirrel-cage induction motors. One of them is used as a load machine, driven by a Danfoss VLT FC-302 3.0 kW load inverter. The working machine is driven by a modified Seidel/Kollmorgen Servostar 600 14 kVA inverter which allows the user to command the IGBT gating signals directly from a suitable control system. In order to avoid frequent use of the break chopper resistor, the DC links of both inverters are connected to each other.

3.1.3 Three-level inverter test bench

The three-level inverter test bench is shown in Figure 3.5. The real-time computer system used for the control of the three-level inverter is a significantly improved version of the one for the two-level inverter [38]. It is based on an FPGA board which is shown in Figure 3.6. The setup consists of a 2.2 kW squirrel-cage induction motor and a self-developed three-level inverter which can be used either in NPC or in FC configuration. The DC link is powered by a 3 kW DC power supply. Further details and parameters are stated in appendix B.2.

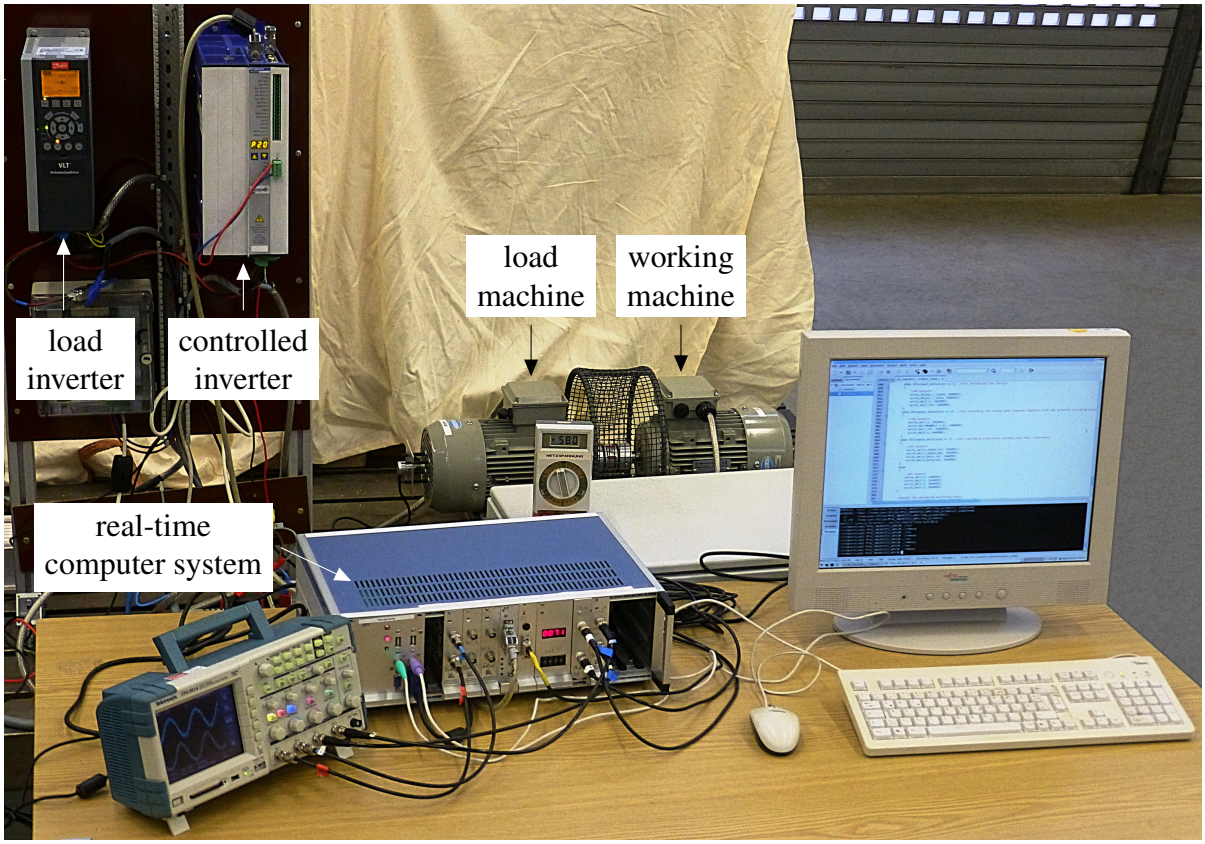


Figure 3.4: Two-level inverter test bench

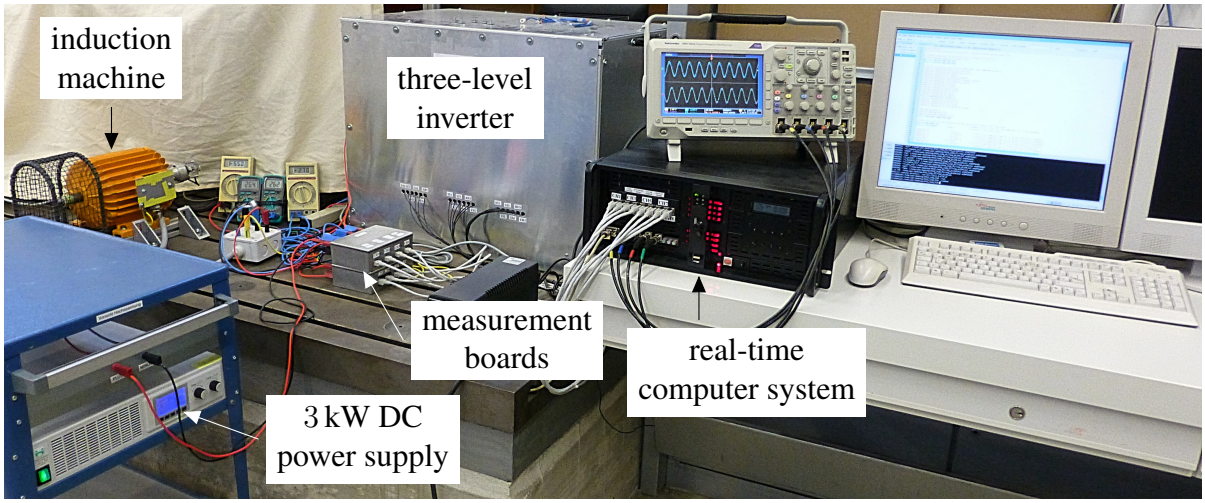


Figure 3.5: Three-level inverter test bench

3.2 Hardware-based real-time implementation

As already mentioned, in former times all controllers had to be realized with analog components, i.e. in hardware. Today nearly all controllers are implemented digitally.

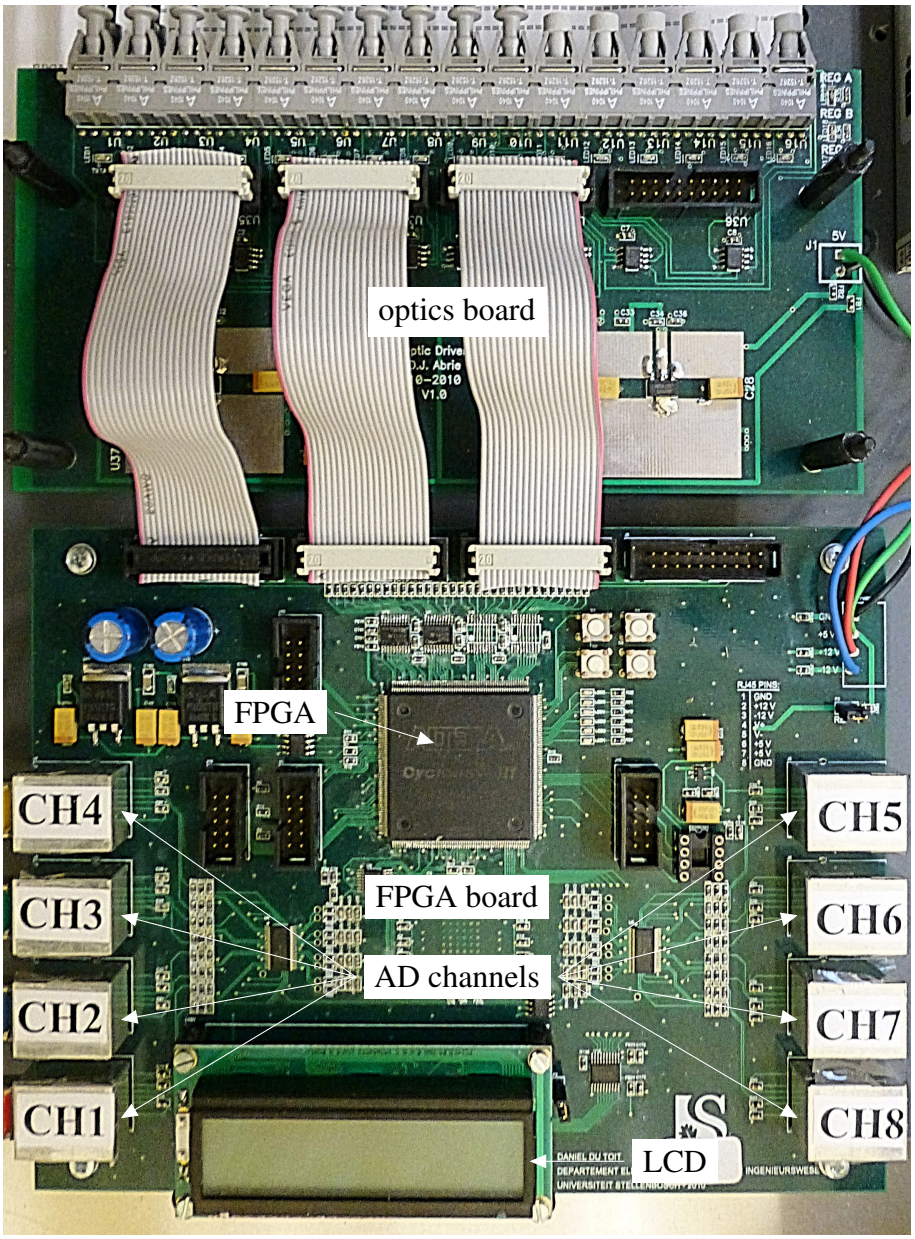


Figure 3.6: FPGA board for hardware-based real-time implementations

However, thanks to the advent of Complex Programmable Logic Devices (CPLDs) and Field Programmable Arrays (FPGAs), even with digital hardware a quasi-analog implementation of control algorithms is possible by having nearly the same flexibility as with software-based implementations.

Especially FPGAs which nowadays have lots of logic elements allow to implement many operations in parallel. Furthermore, as hardware-programmed commands are normally executed

within one clock cycle, even for some subsequent calculations (calculations which are based on other calculation results), only a few clock cycles are necessary for the calculation of new and optimized values for the actuating variables $\boldsymbol{u}$. Thus, compared to software-based implementations, the calculation time is normally negligibly short compared to the sampling time and does not need to be compensated. This also simplifies the calculations.

Figure 3.6 shows the FPGA board which is used for the hardware-based real-time implementation of control algorithms. The control board uses a Cyclone III FPGA from Altera with $40{,}000$ logic elements. Further information about this system is given in appendix B.3.

3.3 Comparison

The main difference between hardware- and software-based implementations is that microprocessors execute a control algorithm *sequentially*, i.e. all instructions are executed one after each other. However, as processors are normally clocked with a very high frequency, it is possible to execute many calculations during one sample. In hardware-based implementations the clock speed is normally much lower but most of the calculations are executed in *parallel*, i.e. it is also possible to do these calculations very quickly.

Another very important difference between hardware- and software-based implementations is that software-based applications have a certain interrupt delay, i.e. it takes some time until the processor can react to this interrupt. Especially for implementations where an operating system is running on the processor this time is not negligible. For this reason the sampling rates that can be achieved with software-based implementations are much lower than the ones of hardware-based applications. Software-based algorithms are usually limited to sampling frequencies of around $40\,\text{kHz}$, whereas hardware-based algorithms can be executed with frequencies up to several hundreds of kHz.

Processors are usually programmed with high-level programming languages like *C*. For CPLDs and FPGAs mostly *VHDL* (Very High Speed Integrated Circuit Hardware Description Language) and *Verilog* are used.

Further advantages of software-based compared to hardware-based implementations are:

- Floating-point operations can be easily programmed.
- The programming of control algorithms on processors is much easier than for CPLDs and FPGAs (e.g. trigonometric operations).
- The necessary compilation times are much shorter.

Thus, hardware-based implementations are mostly used in applications where high sampling rates are mandatory.

CHAPTER 4

Predictive Torque Control for induction machines

In this chapter the Predictive Torque Control (PTC) algorithm [39, 40] is explained and evaluated. It is a model-based alternative to DTC as explained in chapter 2.6.3. Compared to DTC, it uses an FS-MPC algorithm to determine the optimum switching state for the next sample. The basic algorithm will be explained in chapter 4.1 and is more intuitive and much easier to understand than DTC. The PTC algorithm is validated experimentally for a two-level inverter with special attention to problems resulting from this control approach.

After that an extension to PTC to reduce torque ripples, namely Variable Switching Point Predictive Torque Control (VSP2TC), is presented. Experimental results for a two-level inverter are shown in order to demonstrate the algorithm's capability to reduce torque ripples and to improve the control result.

Then, the PTC algorithm is extended to three-level NPC inverters with special attention to the DC link capacitor voltage balancing. Several experimental results are presented in order to verify the algorithm's capability to perform torque and flux control while simultaneously keeping the DC link capacitor voltages balanced. Furthermore, the VSP2TC algorithm is extended to three-level NPC inverters.

Thereafter, PTC is applied to an induction machine fed by a three-level FC inverter which has 64 different switching states. A strategy to reduce the calculation effort for PTC in combination with FC inverters is presented along with several experimental results.

4.1 Basic Predictive Torque Control algorithm

As already mentioned, PTC uses the same stator-fixed approach as DTC. This means that the whole control algorithm is performed in $\alpha\beta$ coordinates. In contrast to that, FOC uses a dq coordinate system which is aligned to the rotor flux. The Euler-forward method is utilized in order to discretize the differential equations. For a successful real-time implementation the computational delay has to be compensated as explained in chapter 3.1.1.

4.1.1 Prediction equations

First, the stator current has to be predicted with equation (2.34) for $\omega_\mathrm{k} = 0$. After discretization with the sampling time T_s, the stator current $\boldsymbol{i}_\mathrm{s}$ can be calculated to

$$\boldsymbol{i}_\mathrm{s}(k+1) = \left(1 - \frac{T_\mathrm{s}}{\tau_\sigma}\right)\boldsymbol{i}_\mathrm{s}(k) + \frac{T_\mathrm{s}}{r_\sigma \tau_\sigma}\boldsymbol{v}_\mathrm{s}(k) + \frac{k_\mathrm{r} T_\mathrm{s}}{r_\sigma \tau_\sigma}\left(\frac{1}{\tau_\mathrm{r}} - j\omega_\mathrm{el}\right)\boldsymbol{\psi}_\mathrm{r}(k). \tag{4.1}$$

The stator flux $\boldsymbol{\psi}_\mathrm{s}$ can be calculated by discretizing equation (2.31) and by setting $\omega_\mathrm{k} = 0$:

$$\boldsymbol{\psi}_\mathrm{s}(k+1) = \boldsymbol{\psi}_\mathrm{s}(k) + T_\mathrm{s}\boldsymbol{v}_\mathrm{s}(k) - R_\mathrm{s} T_\mathrm{s}\boldsymbol{i}_\mathrm{s}(k) \tag{4.2}$$

By substituting equation (2.29) into equation (2.30), the rotor flux $\boldsymbol{\psi}_\mathrm{r}$ results to

$$\boldsymbol{\psi}_\mathrm{r}(k+1) = \frac{L_\mathrm{r}}{L_\mathrm{m}}\boldsymbol{\psi}_\mathrm{s}(k+1) + \left(L_\mathrm{m} - \frac{L_\mathrm{r} L_\mathrm{s}}{L_\mathrm{m}}\right)\boldsymbol{i}_\mathrm{s}(k+1). \tag{4.3}$$

The rotor flux only needs to be calculated for the time delay compensation or for intermediate time steps when a higher prediction horizon than one sample is implemented. The machine torque can then be predicted according to equation (2.36):

$$T_\mathrm{m}(k+1) = \frac{3}{2} p \left(\boldsymbol{\psi}_\mathrm{s}(k+1) \times \boldsymbol{i}_\mathrm{s}(k+1)\right) \tag{4.4}$$

Finally, the stator flux magnitude can be easily calculated by

$$|\boldsymbol{\psi}_\mathrm{s}(k+1)| = \sqrt{\psi_{\mathrm{s}\alpha}^2(k+1) + \psi_{\mathrm{s}\beta}^2(k+1)}. \tag{4.5}$$

4.1.2 Control algorithm

4.1.2.1 Basic principle

Figure 4.1 shows a block diagram of the basic PTC control structure: Analog to FOC and DTC, if a balanced three-phase system is considered, only two phase currents (i_{sa} and i_{sb}) have to be measured. The stator and rotor flux can be estimated after the Clarke transformation has been applied to the measured phase currents. The currents in $\alpha\beta$ coordinates, together with the stator $\boldsymbol{\psi}_\mathrm{s}$ and rotor $\boldsymbol{\psi}_\mathrm{r}$ flux and the machine speed ω_m, are needed for the PTC algorithm. Furthermore, the reference values for the machine torque, T_m^*, and for the stator flux magnitude, $|\boldsymbol{\psi}_\mathrm{s}|^*$, are also needed for the cost function calculation. The torque reference value is generated by a conventional speed PID controller. Finally, after the optimization process, the controller outputs the optimum switching states which are directly fed to the inverter.

4.1.2.2 Cost function

As every MPC scheme, the PTC algorithm is based on a minimization of the cost function. This cost function contains the control error together with additional terms (if necessary). For

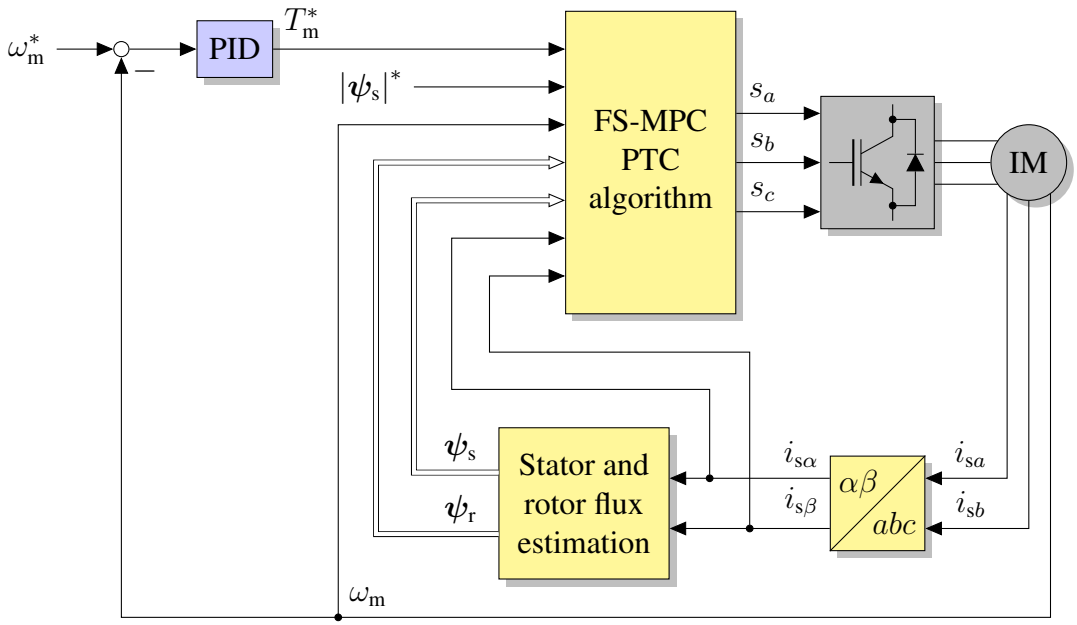


Figure 4.1: Basic PTC scheme

PTC simultaneous control of both the stator flux magnitude and the machine torque is desired. Hence, an L_1-norm (linear) cost function[1] can be defined as

$$j_{\text{ptc, lin}} = |T_m^* - T_m(k+1)| + w_{\text{lin}} \cdot ||\psi_s|^* - |\psi_s(k+1)|| + i_{\text{lim}}. \tag{4.6}$$

w_{lin} is the weighting factor for the stator flux magnitude. Furthermore, it is also possible to define an L_2-norm (quadratic) cost function[1] of the form

$$j_{\text{ptc, quad}} = (T_m^* - T_m(k+1))^2 + w_{\text{quad}} \cdot (|\psi_s^*| - |\psi_s(k+1)|)^2 + i_{\text{lim}}. \tag{4.7}$$

In this case w_{quad} is the flux weighting factor. i_{lim} is a factor for limiting the stator current $\boldsymbol{i}_s$. It can be implemented as follows:

$$i_{\text{lim}} = \begin{cases} 0 & \text{if } |\boldsymbol{i}_s| \leq i_{\text{max}} \\ \infty & \text{if } |\boldsymbol{i}_s| > i_{\text{max}} \end{cases} \tag{4.8}$$

i_{max} is the maximum allowed stator current magnitude $|\boldsymbol{i}_s|$. Thus, the term i_{lim} ensures that no switching states are selected which would lead to a stator current greater than its allowed maximum.

4.1.2.3 Switching state selection

As it can be seen in Figure 2.4(b), a two-level inverter has eight different switching states which produce seven different voltage vectors. The zero switching states nnn and ppp are

[1] In literature these cost functions are often denoted as L_1- or L_2-norms. However, other terms (e.g. for current limitation) can also be added. The resulting expressions are then of course different to the mathematical definitions of L_1- and L_2-norms.

redundant. Hence, for the cost function optimization it is enough to calculate the cost function value for the seven different voltage vectors. If a zero switching state is optimal, a subsequent decision between the two possibilities has to be made. Since the inverter switching losses are proportional to the switching frequency, that zero switching state is selected which can be reached with less than two transitions (based on the currently applied switching state):

- If two or three legs are in the positive state, ppp is selected.
- If less than two legs are in the positive state, nnn is selected.

This principle should be demonstrated with two examples: Assuming that the previously applied switching state was npp, the optimum zero switching state is ppp; for npn it is nnn.

4.1.3 Experimental results

In order to evaluate the PTC algorithm and to use it as a reference for improved and more sophisticated algorithms, several experiments at different operating points were conducted. All experiments shown in this section were recorded on the two-level inverter test bench described in chapter 3.1.2. The algorithm was executed with a sampling time $T_\mathrm{s} = 61.44\,\mu\mathrm{s}$ which corresponds to a controller frequency of about $16\,\mathrm{kHz}$. The DC link voltage was set to $580\,\mathrm{V}$. The machine parameters are given in Table B.1. The nominal machine torque can be easily calculated to

$$T_{\mathrm{m,\,nom}} = \frac{P_{\mathrm{m,\,nom}}}{\omega_{\mathrm{m,\,nom}}} = \frac{2.2\,\mathrm{kW}}{2772\,\mathrm{rpm}} = 7.58\,\mathrm{Nm}. \tag{4.9}$$

$P_{\mathrm{m,\,nom}}$ is the rated mechanical machine power and $\omega_{\mathrm{m,\,nom}}$ the rated nominal speed. For the experimental results the cost function given in equation (4.7) was used with $w_{\mathrm{quad}} = 289$. i_{max} was set to $14\,\mathrm{A}$.

Figure 4.2 shows a torque reference step from almost zero to full nominal load. In order to operate the test bench safely, the PTC algorithm has to be implemented with an overlayed speed PID controller. Thus, the torque reference step was generated by changing the speed reference from $1000\,\mathrm{rpm}$ to $2000\,\mathrm{rpm}$. The speed controller then changes the torque reference from nearly zero to full nominal torque (speed PID controller output limit). During the short time that is shown ($2\,\mathrm{ms}$ after the step), the torque reference is constant as the acceleration process takes longer. In this way a torque reference step can be easily recorded. The torque reference T_m^* and the real machine torque T_m can be seen in the upper plot; the plot below shows the applied switching state where the number is coded binary in the form $[s_c\, s_b\, s_a]$. Thus, the switching state $s_c = 0$, $s_b = 0$, $s_a = 1$ leads to 1, whereas $s_c = 1$, $s_b = 0$, $s_a = 1$ leads to 5. It can clearly be seen that in the case of a commanded zero voltage vector always the one is selected which can be reached with less than two transitions. Furthermore, during the transient (from $2\,\mathrm{ms}$ to $2.5\,\mathrm{ms}$) only one switching state (number 6) is applied to the machine, i.e. no switching takes place until the torque has reached its reference. Thus, the torque follows its reference as fast as possible and PTC operates the machine at its physical limits.

Another experiment was performed in steady-state: The stator flux in the $\alpha\beta$ plane was recorded at $2000\,\mathrm{rpm}$ and $4\,\mathrm{Nm}$ load which can be seen in Figure 4.3. As expected, both components, $\psi_{\mathrm{s}\alpha}$ and $\psi_{\mathrm{s}\beta}$, form a circle.

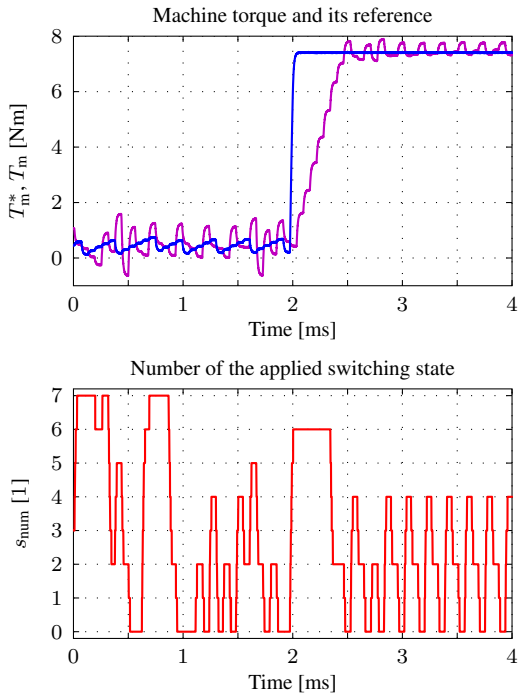


Figure 4.2: Torque reference step and applied switching state

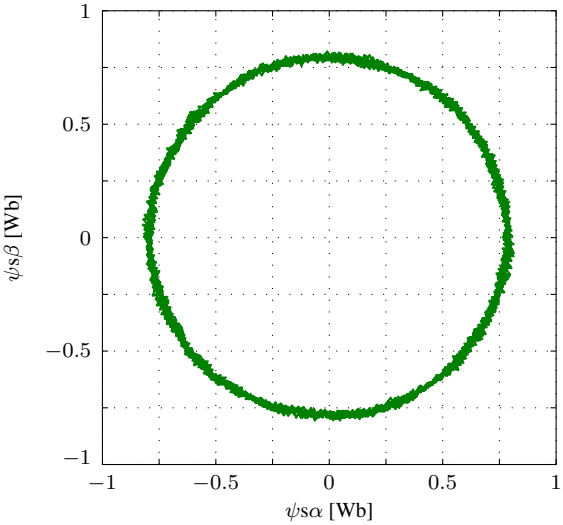


Figure 4.3: Steady-state stator flux in $\alpha\beta$ coordinates at 2000 rpm and 4 Nm load torque

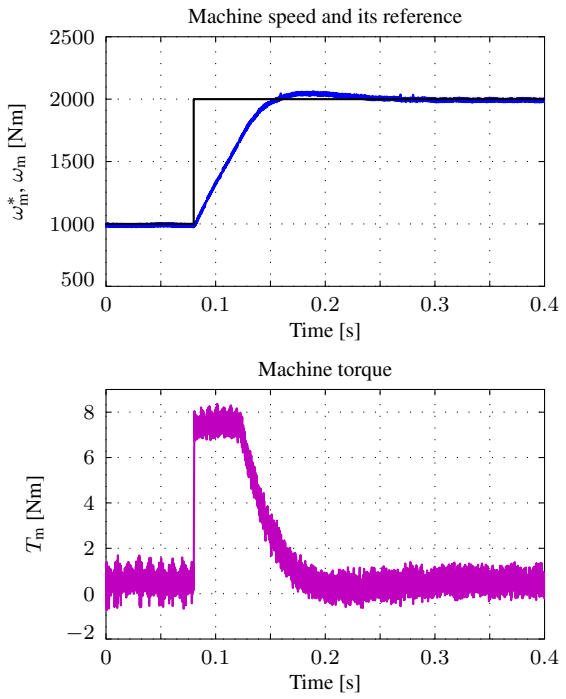


Figure 4.4: Speed reference step and produced machine torque

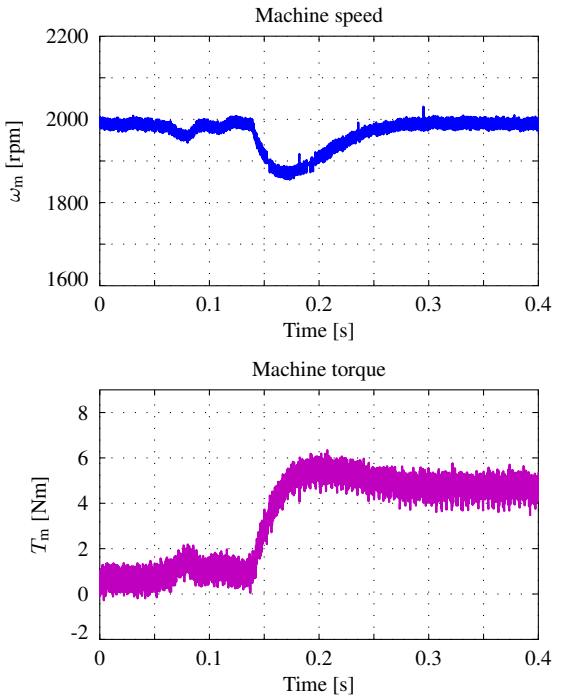
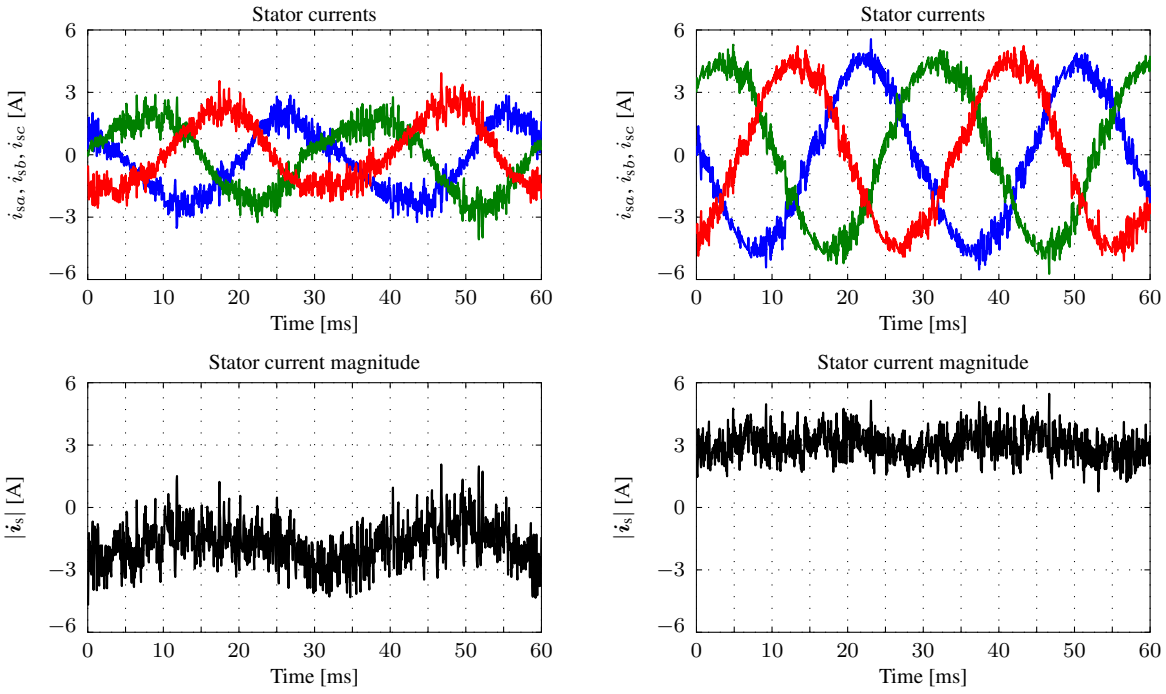


Figure 4.5: Load torque impact (4 Nm at 2000 rpm)

Figure 4.4 shows the machine speed ω_m, its reference ω_m^* and the applied machine torque T_m during a speed reference step from $1000\,\mathrm{rpm}$ to $2000\,\mathrm{rpm}$. In contrast to Figure 4.2, the recorded variables are shown for a longer time period since the time constant of the speed control loop is much larger than that for torque control. Although the maximum available dynamics are utilized due to the large speed reference change, the behavior after $0.13\,\mathrm{s}$ is not optimal: The PI controller does not operate the system at its physical limits and slowly decreases the machine torque. Furthermore, a slight overshoot can be noticed. Of course, a different speed PID controller tuning (and additional reference filtering and feedforward control) could lead to better results; in this case, however, the speed controller was adjusted such that the closed speed control loop has a bandwidth of $10\,\mathrm{Hz}$ and that good disturbance rejection can be achieved: Usually, a load torque can be applied to the machine and the speed controller has to react quickly to that disturbance.

The working machine is coupled to a load machine which can perform a sudden torque step. The results of such an experiment are shown in Figure 4.5. In this case a load torque step with $4\,\mathrm{Nm}$ was applied at $0.14\,\mathrm{s}$ when the machine was rotating at $2000\,\mathrm{rpm}$. The speed drops to about $1850\,\mathrm{rpm}$ and after about $150\,\mathrm{ms}$ it has reached its reference again. Although the speed controller can react quickly to such disturbances, it becomes obvious that in this case the machine is not operated at its physical limits: By applying a higher torque during the speed drop, a faster speed regulation would be possible. This also demonstrates the general problem of conventional controllers: A controller setting is always a tradeoff for different operating points and in order to operate a drive always at its limits, adaptive controllers are necessary. By using MPC such disadvantages can be overcome.



(a) No load torque, $f_\mathrm{sw} = 4.2\,\mathrm{kHz}$

(b) 4 Nm load torque, $f_\mathrm{sw} = 3.5\,\mathrm{kHz}$

Figure 4.6: Steady-state stator currents at $2000\,\mathrm{rpm}$

In another experiment the steady-state phase currents i_{sa}, i_{sb} and i_{sc} were recorded at $2000\,\text{rpm}$, without load (Figure 4.6(a)) and with $4\,\text{Nm}$ load torque (Figure 4.6(b)). Since the machine torque and the stator flux magnitude are controlled and *not* the machine currents, especially without load the stator currents are heavily distorted and show a significant deviation from their ideally sinusoidal waveform. The average switching frequency per IGBT is in this case about $4.2\,\text{kHz}$. Although the sinusoidal waveform is clearly visible when a load torque is present, the phase currents still show a high current ripple which is due to the low switching frequency of about $3.5\,\text{kHz}$.

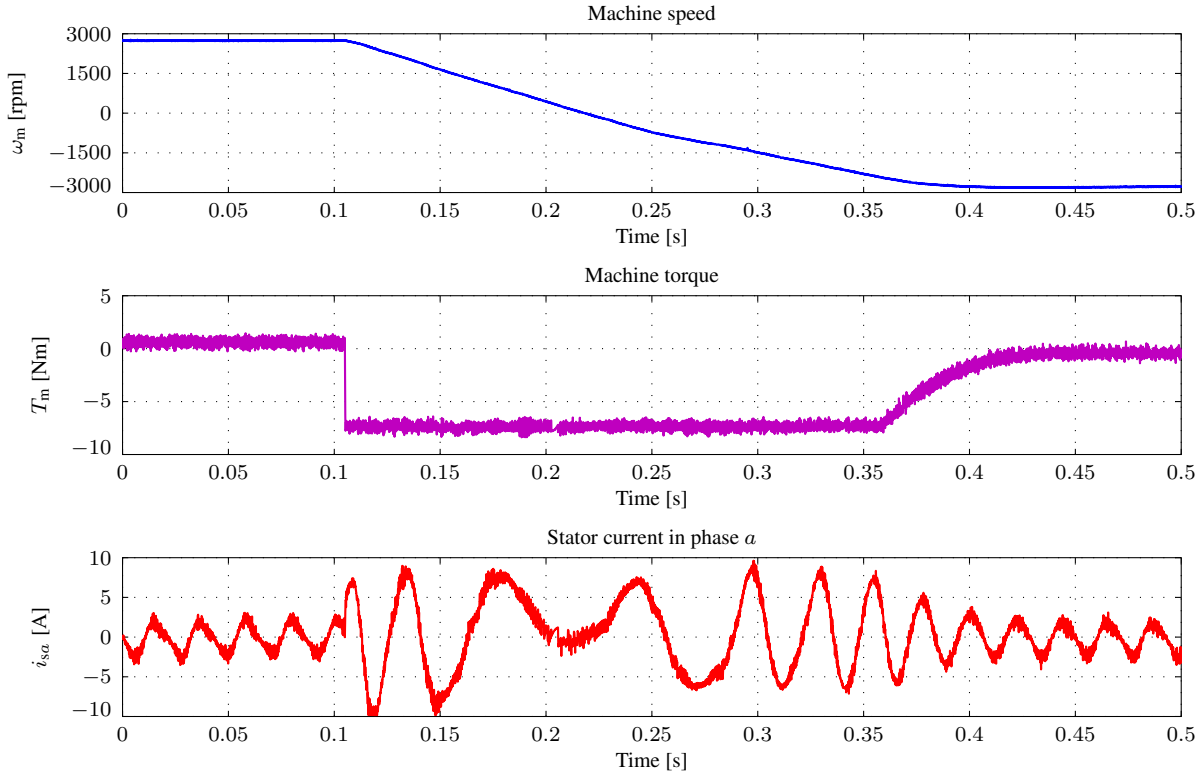


Figure 4.7: Speed reversal from positive nominal to negative nominal speed

A speed reversal from positive nominal to negative nominal speed was executed in order to show that the algorithm can operate at all speed levels and operation points. The results are shown in Figure 4.7. The upper plot shows the machine speed ω_m, in the middle the machine torque T_m can be seen and in the third graph the current in phase a, i_{sa}, is shown during the speed reversal. Similar to the speed reference change, at the beginning the PID controller operates the system at its physical limits. However, at about $0.36\,\text{s}$ the control result becomes suboptimal since the speed deviation has already become small. In this case a speed MPC could also improve the dynamics.

4.1.4 Evaluation

As already mentioned, PTC is a model-based alternative to DTC. It is much easier to understand and more intuitive compared to DTC. Furthermore, it is much better suited for software-based

implementations than DTC since hysteresis controllers usually have to be implemented in hardware or at least with sampling frequencies of $40\,\text{kHz}$ and above. Despite all advantages of PTC compared to DTC and those of FS-MPC methods in general, high torque, flux and current ripples can be seen in all experimental results. Furthermore, without load torque the phase currents which are only indirectly controlled show significant deviations from sinusoidal waveforms *additionally* to high ripples which result from the low switching frequency compared to PWM-based methods with the same sampling frequency. Furthermore, due to the non-constant switching frequency, PTC produces an undesired audible noise which is much more annoying than noise produced by PWM.

For high- and medium-voltage drives low switching frequencies are necessary, since for these applications the system's total losses are mainly dominated by the inverter switching losses whereas ripples on the controlled variables are less important. Hence, for these applications PTC can be a promising and more intuitive alternative to DTC. For low-voltage drives, however, inverter switching losses are less important than small ripples on the controlled variables. Therefore, PTC needs additional improvements in order to achieve better control results with less ripples which are comparable to PWM-based methods. In the following, an algorithm to reduce torque ripples in PTC will be presented.

4.2 Variable Switching Point Predictive Torque Control

VSP2TC [41] implements the basic PTC algorithm. The main difference to PTC is the calculation of a variable switching point (VSP) such that torque ripples are minimized.

4.2.1 Motivation and basic principle

As already mentioned in chapter 2.7.5.4, FS-MPC suffers from the fact that its time resolution is very low compared to modulation-based methods: The switching states which an inverter produces normally lead to *different* slopes of the controlled variables. Since the minimum duration for which a certain switching state has to be applied to the system is equal to one *whole* sample, this leads to high ripples on the controlled variables. Although the theoretical maximum switching frequency of FS-MPC methods is equal to half the sampling frequency, it is in reality far lower than this value which was also proven with the conducted PTC experiments: Although the theoretically maximum switching frequency per IGBT is $8\,\text{kHz}$ for a sampling frequency of $16\,\text{kHz}$, the average switching frequency at $2000\,\text{rpm}$ was only $4.2\,\text{kHz}$ without load and $3.5\,\text{kHz}$ with load. Since—primarily for software-based implementations—the maximum controller frequency is limited, the currently available FS-MPC methods need further improvements in order to reduce ripples and to improve the control result, especially for low-voltage drives and applications.

In order to achieve a better control result, the following approach is used: Since it is not possible to change the slope of the controlled variables, the only possibility to reduce ripples on the controlled variables is to apply voltage vectors for a *shorter* time than one whole sample T_s. Thus, the basic idea is—in contrast to normal FS-MPC—to apply the new switching state not necessarily at the *beginning* of a new sample; the previously applied switching state is kept until the VSP t_sw. Then, at t_sw, the new switching state is applied until the end of the sample at

T_s. In this way a switching state can be applied for less time than a whole sample which results in a higher switching frequency (which is, however, still limited to half the sampling frequency) and reduced ripples on the controlled variables.

4.2.2 Control algorithm

The prediction equations for VSP2TC are the same as for standard PTC; the only difference is that the "sampling time" for the predictions (T_s for standard PTC) is now variable (t_sw for the first interval, $T_\mathrm{s} - t_\mathrm{sw}$ for the second one). The control algorithm follows the same principles as standard PTC. For VSP2TC, however, the torque and stator flux magnitude have to be calculated at two points (at t_sw and at T_s) and a modified cost function is used:

$$\begin{aligned} j_\mathrm{vsp2tc} = &\left(T_\mathrm{m}^* - T_\mathrm{m}(t_\mathrm{sw})\right)^2 + \left(T_\mathrm{m}^* - T_\mathrm{m}(k+1)\right)^2 + \\ &+ w_\mathrm{quad} \cdot \left[\left(|\boldsymbol{\psi}_\mathrm{s}^*| - |\boldsymbol{\psi}_\mathrm{s}(t_\mathrm{sw})|\right)^2 + \left(|\boldsymbol{\psi}_\mathrm{s}^*| - |\boldsymbol{\psi}_\mathrm{s}(k+1)|\right)^2\right] \end{aligned} \tag{4.10}$$

w_quad is the weighting factor which is necessary to keep a good balance between the two control aims, i.e. control of the machine torque and of the stator flux magnitude. It can be tuned empirically, in the same way as for standard PTC.

4.2.3 Calculation of the variable switching time point

In order to keep the calculations simple, the machine torque $T_\mathrm{m}(t)$ is assumed to change *linearly* if a certain switching state is applied. Hence, the torque can be assumed to change linearly with the time t:

$$T_\mathrm{m}(t) \approx T_\mathrm{m0} + mt \tag{4.11}$$

where T_m0 is the torque at the beginning of the straight and m is the torque slope.

Figure 4.8 illustrates the basic principle of the VSP calculation for a tested switching state. The whole sampling cycle which has the duration T_s is divided in two parts: The intervals $k \ldots t_\mathrm{sw}$ and $t_\mathrm{sw} \ldots k+1$. Compared to standard PTC where a switching state is applied from the beginning until the end of a sampling cycle, for VSP2TC the previously applied switching state s_1 is kept until t_sw and then, in the second interval, the new switching state s_2 is applied. In order to minimize the torque, the VSP is calculated such that

$$T_\mathrm{m}(k+1) = T_\mathrm{m}^*, \tag{4.12}$$

i.e. the torque at the end of the sampling cycle should be equal to its reference. In order to simplify the calculations, it is assumed that every switching state leads to a certain torque slope which is constant over the whole sample. At the beginning the torque slopes for every possible voltage vector have to be calculated. For this it is assumed that the switching would take place at step k and that the voltage vector would be applied for the whole sample T_s. Thus, for the prediction of the torque slopes seven extra predictions are necessary. The torque slope for all voltage vectors can then be calculated to

$$m_i = \frac{T_{\mathrm{m}i}(k+1) - T_\mathrm{m}(k)}{T_\mathrm{s}}, i = 1 \ldots 7, \tag{4.13}$$

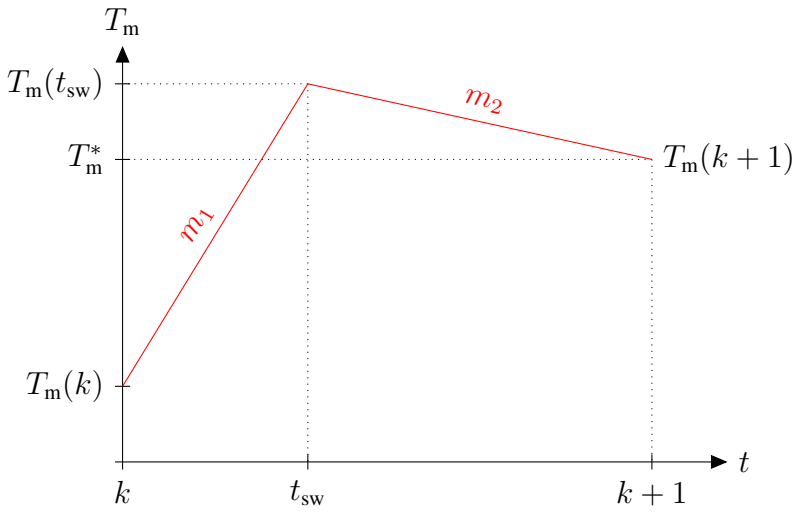


Figure 4.8: Basic principle of the VSP calculation for VSP2TC

where i is the number of the voltage vector. The machine torque at the intermediate time step can be calculated to

$$T_\mathrm{m}(t_\mathrm{sw}) = T_\mathrm{m}(k) + m_1 \cdot t_\mathrm{sw}. \tag{4.14}$$

The torque at the end of the sample, by taking equation (4.12) into account, results to

$$T_\mathrm{m}(k+1) = T_\mathrm{m}^* = T(t_\mathrm{sw}) + m_2 \cdot (T_\mathrm{s} - t_\mathrm{sw})\,. \tag{4.15}$$

Finally, the VSP can then be calculated to

$$t_\mathrm{sw} = \frac{T_\mathrm{m}^* - T_\mathrm{m}(k) - m_2 \cdot T_\mathrm{s}}{m_1 - m_2}. \tag{4.16}$$

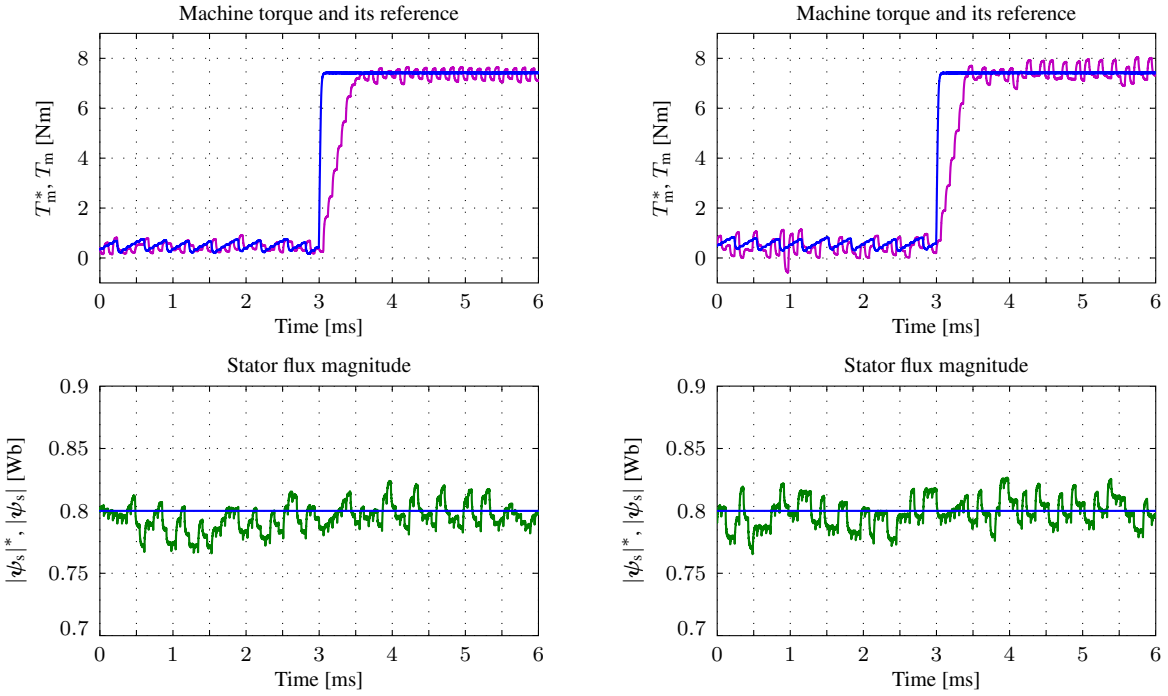
In this context m_1 is the torque slope of the first straight ($t \in [k \ldots t_\mathrm{sw}]$) and m_2 the one of the second torque straight ($t \in]t_\mathrm{sw} \ldots T_\mathrm{s}]$).

The calculated VSP can theoretically be in the range $t_\mathrm{sw} \in -\infty \ldots \infty$ for certain combinations of switching states and torque slopes. In these cases $t_\mathrm{sw} = 0$ is set which means that for those switching states the standard PTC algorithm is used.

4.2.4 Experimental results

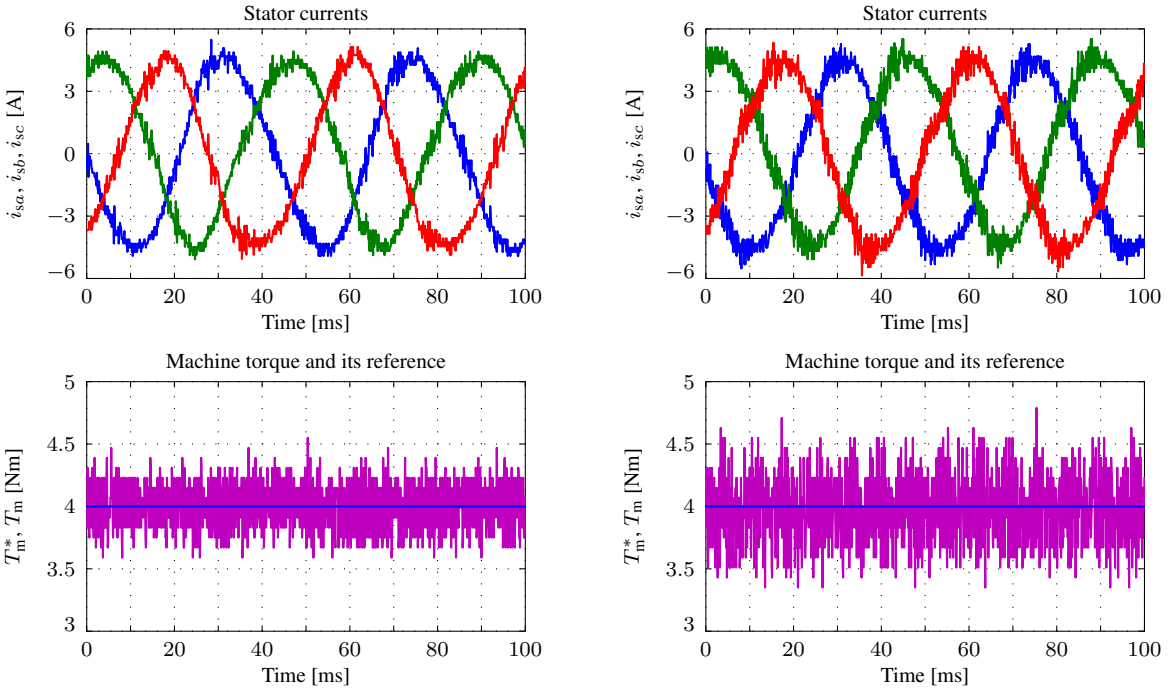
In order to verify the proposed VSP2TC algorithm, several experiments on the two-level inverter test bench (described in chapter 3.1.2) were conducted. For VSP2TC equation (4.10) was used as cost function, equation (4.7) was used for PTC. In both cases the weighting factor w_quad was set to 289 and the maximum allowed stator current magnitude was limited to $14\,\mathrm{A}$.

Figure 4.9(a) shows a step change of the machine torque reference for VSP2TC, T_m^*, from nearly $0\,\mathrm{Nm}$ to $7.58\,\mathrm{Nm}$ (nominal machine torque) at $3\,\mathrm{ms}$ and the corresponding stator flux magnitude $|\psi_\mathrm{s}|$ whose reference is $0.8\,\mathrm{Wb}$. This torque step was produced in closed-loop speed control during a speed reference change from $1000\,\mathrm{rpm}$ to $2000\,\mathrm{rpm}$. Compared to the results of PTC which are shown in Figure 4.9(b), it can clearly be seen that the torque ripple is reduced



(a) VSP2TC (b) PTC

Figure 4.9: Torque reference step



(a) VSP2TC, $f_{sw} \approx 2.9\,\text{kHz}$ (b) PTC, $f_{sw} \approx 3.2\,\text{kHz}$

Figure 4.10: Steady-state operation at $\omega_m = 1500\,\text{rpm}$

and that the dynamic behavior is not deteriorated. Furthermore, no influence of the torque change on the stator flux can be observed.

In order to examine the steady-state performance of the VSP2TC algorithm and its ability to reduce torque and current ripples, closed-loop speed control was performed with the Danfoss load inverter. The speed reference was set to $\omega_\mathrm{m} = 1500\,\mathrm{rpm}$ which is slightly higher than the machine's half nominal speed ($\omega_{\mathrm{m,\,nom}} = 2772\,\mathrm{rpm}$). Then, the controlled inverter was operated at constant torque, i.e. without a superimposed speed control loop. In this way it is possible to operate the machine with a constant torque reference and without exceeding the machine's speed limits. A comparison of the steady-state currents during such an experiment can be seen in Figure 4.10(a) (VSP2TC) and Figure 4.10(b) (PTC): The results were taken at $\omega_\mathrm{m} = 1500\,\mathrm{rpm}$ and the controlled inverter was then operated at a constant torque reference of $4\,\mathrm{Nm}$ which is slightly more than half the nominal machine torque. By taking a closer look at the recorded phase currents, it is clearly visible that VSP2TC also reduces current ripples at this operating point. The second plots show the machine torque whose ripple is significantly reduced by the use of VSP2TC. Of course, the average IGBT switching frequency for VSP2TC is slightly higher ($3.2\,\mathrm{kHz}$ compared to $2.9\,\mathrm{kHz}$ for PTC) which is due to strategies that utilize a VSP.

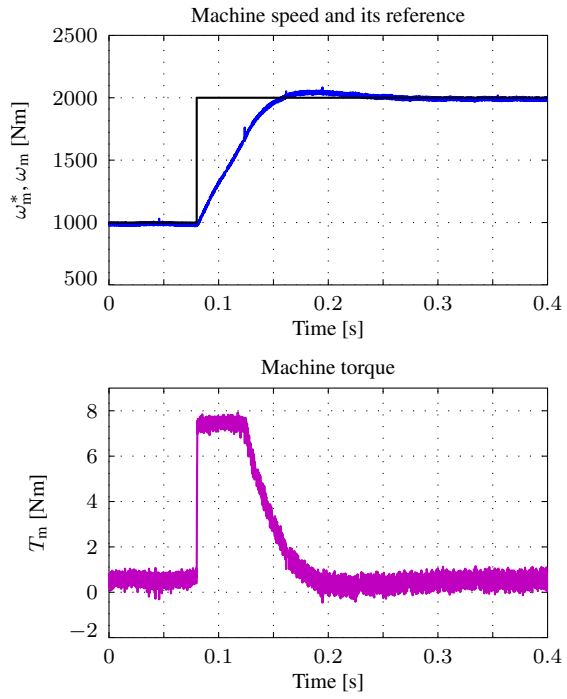


Figure 4.11: Speed reference step and produced machine torque

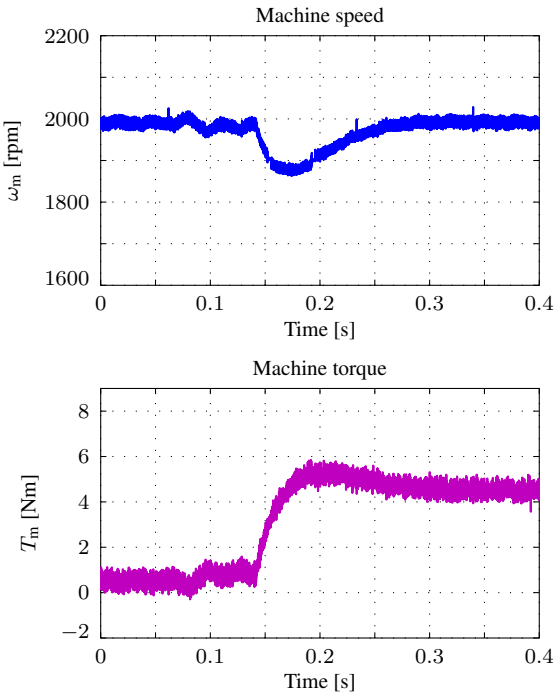


Figure 4.12: Load torque impact ($4\,\mathrm{Nm}$ at $2000\,\mathrm{rpm}$)

Figure 4.11 shows the machine speed ω_m and the corresponding machine torque T_m during a speed reference step from $1000\,\mathrm{rpm}$ to $2000\,\mathrm{rpm}$. It is obvious that VSP2TC does not deteriorate the speed control result compared to PTC which is shown in Figure 4.4. In Figure 4.12 the influence of a load torque impact ($4\,\mathrm{Nm}$ at $2000\,\mathrm{rpm}$) can be seen. This experiment also clearly verifies that the dynamic control result is not deteriorated compared to PTC (Figure 4.5).

4.2.5 Evaluation

The presented experimental results clearly demonstrate that VSP2TC leads to a significant reduction of torque ripples compared to PTC when it is executed with the same sampling frequency. Furthermore, at least for certain operating points, VSP2TC also leads to a reduction of current ripples. The control result for the stator flux magnitude is not deteriorated which makes VSP2TC a very promising approach.

However, for VSP2TC a much higher calculation effort compared to PTC is necessary:

1. In order to calculate the torque slopes, one extra set of predictions for all possible voltage vectors is necessary.
2. The VSP has to be calculated.
3. For the cost function calculation the model has to be calculated *twice* for every possible voltage vector.

Thus, more than *twice* as many calculations are necessary for VSP2TC compared to PTC. This limits the applicability of VSP2TC. However, for systems where high sampling frequencies are not possible but which have enough calculation power, VSP2TC is a promising alternative to PTC.

4.3 PTC for three-level NPC inverters

The generic PTC algorithm can be easily extended to three-level NPC inverters [42]. The basic algorithm and the prediction equations do not need to be changed. Compared to two-level inverters, 19 different voltage vectors and 27 different switching states have to be evaluated which leads to a much higher calculation effort. Furthermore, the DC link capacitor voltage balancing has to be ensured and thus, an additional term in the cost function is necessary.

4.3.1 Voltage balancing extension

The voltage balancing mechanism of an NPC inverter is described in chapter 2.5.2.3. In order to ensure that the DC link capacitor voltages stay close to their reference values $v_{\mathrm{c1}} = v_{\mathrm{c2}} = 0.5V_{\mathrm{dc}}$, equation (2.23) has to be discretized with the sampling time T_{s}. In this case the Euler forward approximation (equation (2.5)) was used. Then, the DC link capacitor voltage difference can be predicted by

$$\Delta v_{\mathrm{c}}(k+1) = \Delta v_{\mathrm{c}}(k) + \frac{T_{\mathrm{s}}}{C}\left(\sum_{j=1}^{3} i_{\mathrm{n}j}(k)\right). \tag{4.17}$$

$i_{\mathrm{n}j}$, $j = 1 \ldots 3$ are the neutral point currents for the three phases.

4.3.2 Control algorithm

As already mentioned, for three-level NPC inverters the prediction equations for the machine model which are described in chapter 4.1.1 are exactly the same as for two-level inverters. The

control algorithm is also the same, with the only difference that in this case more voltage vectors have to be evaluated.

In order to simplify the calculations for the voltage balancing, the machine is seen as a current source and it is assumed that the currents are changing slowly compared to the sampling time, i.e. $\boldsymbol{i}_\mathrm{s}(k+1) \approx \boldsymbol{i}_\mathrm{s}(k)$. Then, the predicted neutral point currents at time step $k+1$ can be used in equation (4.17) in order to model the direct influence of the selected switching state on the capacitor voltage difference.

After the machine model is calculated, the inverse Clarke transformation is applied to the currents $i_{\mathrm{s}\alpha}(k+1)$ and $i_{\mathrm{s}\beta}(k+1)$ in order to get the predicted phase currents $i_{\mathrm{s}a}(k+1)$, $i_{\mathrm{s}b}(k+1)$ and $i_{\mathrm{s}c}(k+1)$. From these phase currents the predicted neutral point currents $i_{\mathrm{n}j}(k+1)$, $j = 1 \ldots 3$ can be easily calculated: The neutral point current in a phase is zero if a positive or a negative switching state is applied, for the case of a zero switching state it is equal to the phase current. Then, the DC link capacitor voltage unbalance for the tested switching state can be calculated according to equation (4.17).

4.3.3 Cost function

For PTC of three-level NPC inverters a modified cost function has to be used: The voltage balancing must be considered in order to keep the DC link capacitor voltages close to their reference values. If an L_1-norm (linear) cost function is used, equation (4.6) can be extended as follows:

$$j_{\text{ptc, lin, npc}} = j_{\text{ptc, lin}} + w_{\text{lin, npc}} \cdot |\Delta v_\mathrm{c}(k+1)| \tag{4.18}$$

For the case of an L_2-norm or quadratic cost function, the following extension of equation (4.7) is proposed:

$$j_{\text{ptc, quad, npc}} = j_{\text{ptc, quad}} + w_{\text{quad, npc}} \cdot (\Delta v_\mathrm{c}(k+1))^2 \tag{4.19}$$

where $w_{\text{lin, npc}}$ and $w_{\text{quad, npc}}$ are the weighting factors for the voltage balancing which can be tuned empirically.

4.3.4 Experimental results

In order to evaluate the PTC algorithm for three-level NPC inverters, several experiments were conducted. All experiments which are shown in this section were recorded on the three-level inverter test bench which is described in chapter 3.1.3. The control algorithm was executed with a frequency of $12\,\mathrm{kHz}$ which corresponds to a sampling time $T_\mathrm{s} = 83.33\,\mu\mathrm{s}$. The DC link was powered with $550\,\mathrm{V}$ by a $3\,\mathrm{kW}$ DC power supply. In this case a different induction machine was used whose parameters are given in Table B.2. For this machine the nominal torque can be calculated according to equation (4.9) and results to $T_{\mathrm{m, nom}} = 7.42\,\mathrm{Nm}$. In this case the L_2-norm cost function given in equation (4.19) was used with $w_{\text{quad, npc}} = 0.01$. The flux weighting factor was set to $w_{\text{quad}} = 156$. The stator current magnitude was in this case limited to $10\,\mathrm{A}$.

Figure 4.13 shows the steady-state machine torque and stator flux at $2830\,\mathrm{rpm}$ (nominal speed). The experiment was conducted by performing closed-loop speed control. The torque reference which is generated by the overlaying speed PI controller is not constant; thus, the visible torque ripple is higher than for a constant torque reference. In the lower graph the stator

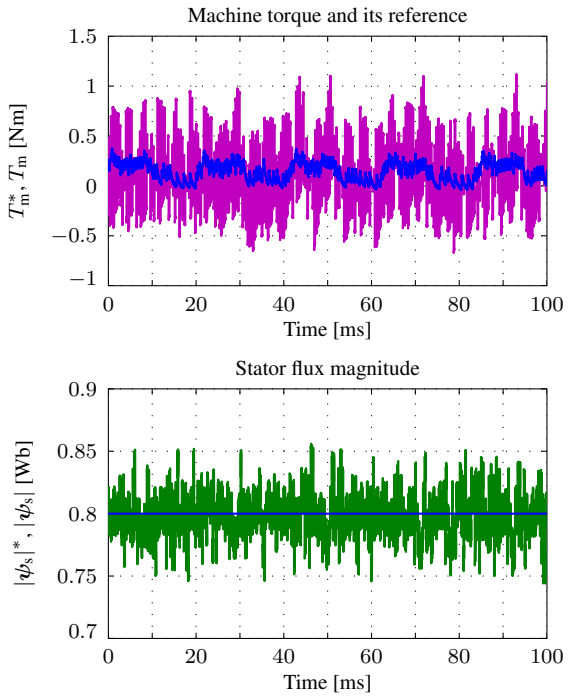


Figure 4.13: Steady-state machine torque and stator flux magnitude at 2830 rpm

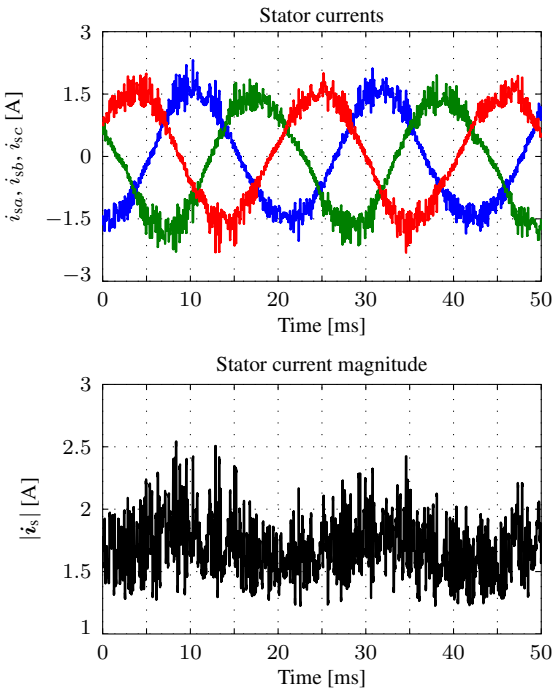


Figure 4.14: Steady-state stator currents at 2830 rpm and no load

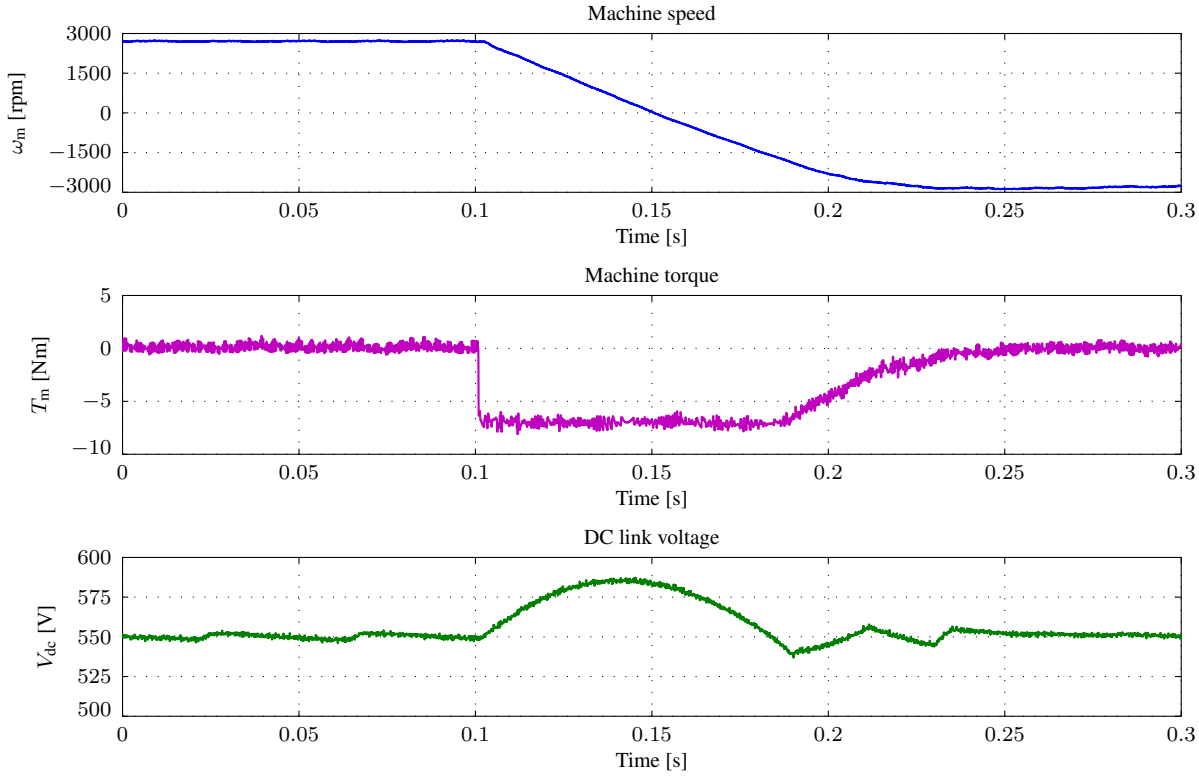


Figure 4.15: Speed reversal from positive nominal to negative nominal speed

flux magnitude and its reference can be seen. This experiment clearly demonstrates that PTC can also be successfully implemented for three-level NPC inverters.

In Figure 4.14 the stator currents and their magnitude at 2830 rpm are shown. Once again, as expected, the current ripples are less than those produced by the two-level inverter. In this case the average switching frequency per IGBT was around 1.9 kHz.

In order to verify that the algorithm is able to operate the drive at different speeds, a speed reversal from positive nominal to negative nominal speed was conducted. The results can be seen in Figure 4.15. It is clearly visible that the controller has no problems to invert the machine speed. Compared to the results on the two-level inverter test bench, the machine torque ripple is less. The lower graph shows the measured DC link voltage during the speed reversal which was delivered from the DC power supply. During the reversal, until the drive has reached zero speed, the drive operates in generator mode. Because of this the DC link voltage rises up to more than 585 V. When the speed becomes negative, the DC link voltage drops since the drive operates again as a motor. The speed reversal occurs during a very short time (about 0.15 s) during which the power supply cannot regulate the DC link voltage.

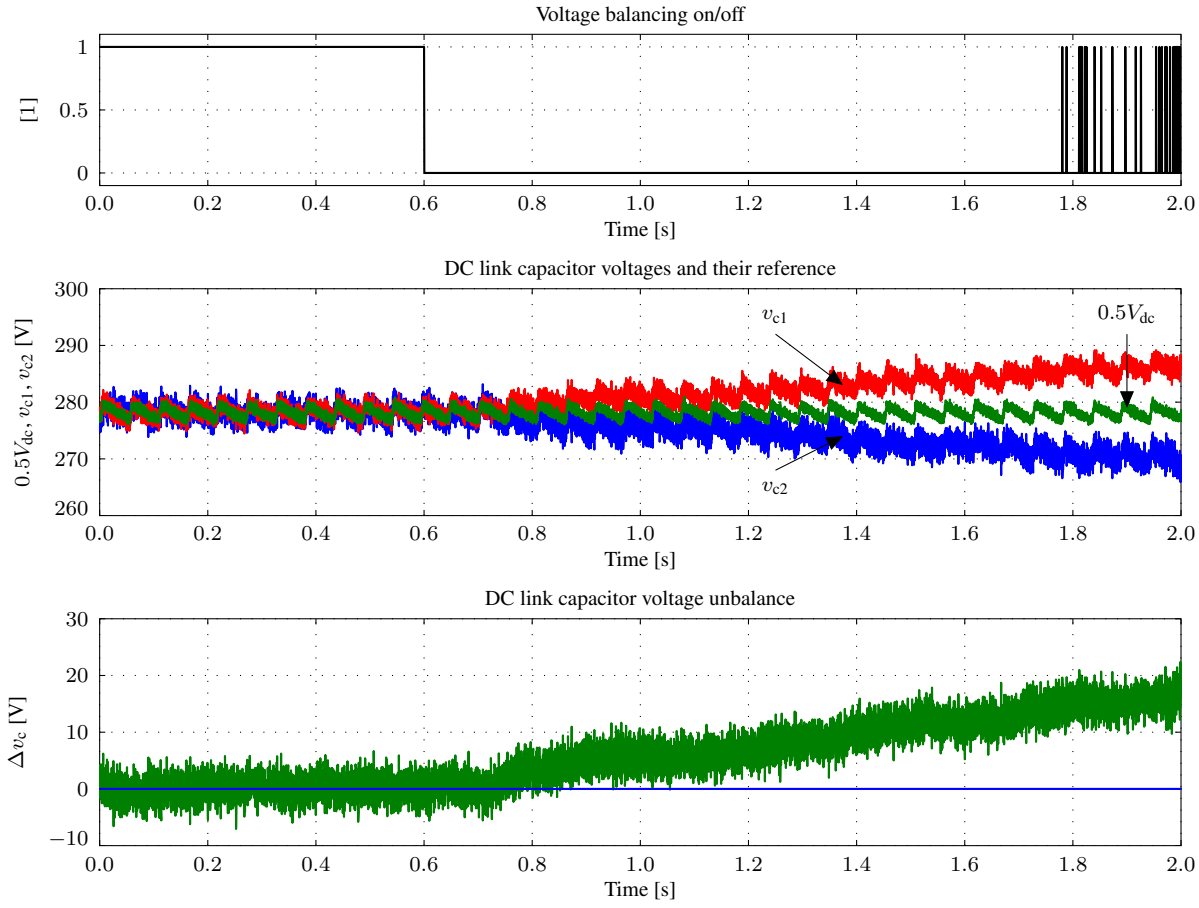


Figure 4.16: Voltage balancing test at 2830 rpm and no load

Finally, the capability of the voltage balancing algorithm was tested. Figure 4.16 shows the DC link capacitor voltages, their reference ($0.5V_{\mathrm{dc}}$) and the resulting voltage unbalance Δv_{c}. At 0.6 s the weighting factor $w_{\text{quad, npc}}$ was set from its initial value 0.01 to 0, i.e. after 0.6 s the

voltage balancing algorithm was manually disabled. For safety reasons it was automatically enabled again as long as Δv_c exceeded a limit of $20\,\mathrm{V}$. It is clearly visible that the DC link capacitor voltages $v_{\mathrm{c}1}$ and $v_{\mathrm{c}2}$ drift away from their references when the voltage balancing algorithm is not effective anymore. Thus, the implementation of a voltage balancing algorithm is absolutely necessary for a safe operation of the drive and the inverter.

4.3.5 Evaluation

The shown experimental results clearly verify that PTC can be successfully implemented for three-level NPC inverters. The voltage balancing problem of the two DC link capacitors can be easily solved by adding a corresponding term to the cost function. The effectiveness of the voltage balancing algorithm was proven experimentally.

The use of a three-level NPC inverter instead of a simple two-level inverter leads to reduced current, torque and flux ripples. However, the calculation effort which is necessary for FS-MPC of an NPC inverter is *significantly* higher than for a two-level inverter: Instead of 7 different voltage vectors and 8 switching states in this case 19 voltage vectors and 27 switching states must be evaluated. The necessary calculation effort is more than *twice* as high than for a simple two-level inverter (for only one prediction step). Additionally to that, further calculations are necessary to balance the DC link capacitor voltages.

Because of the increased calculation effort the sampling time probably has to be increased if a three-level inverter is used instead of a two-level inverter. Then, the benefit of smaller ripples on the controlled variables might be less than expected. Thus, especially for multilevel inverters techniques to reduce the calculation effort are absolutely necessary.

4.4 VSP2TC for three-level NPC inverters

Analogously to PTC, the VSP2TC algorithm which was described in chapter 4.2 can also be extended to three-level NPC inverters [43].

4.4.1 Control algorithm

The VSP2TC algorithm for three-level NPC inverters implements the basic VSP2TC algorithm. In this case 19 voltage vectors and 27 switching states have to be evaluated. Furthermore, the DC link capacitor voltage balancing has to be done as well. Analogously to PTC, the capacitor voltage unbalance can be predicted with equation (4.17). The DC link capacitor voltage difference for the first prediction interval (from 0 to t_sw) is given by

$$\Delta v_\mathrm{c}(k + t_\mathrm{sw}) = \Delta v_\mathrm{c}(k) + \frac{t_\mathrm{sw}}{C} \left(\sum_{i=1}^{3} i_{\mathrm{n}j}(k) \right). \tag{4.20}$$

Then, for the second interval (from t_sw to T_s) the DC link capacitor voltage difference can be calculated to

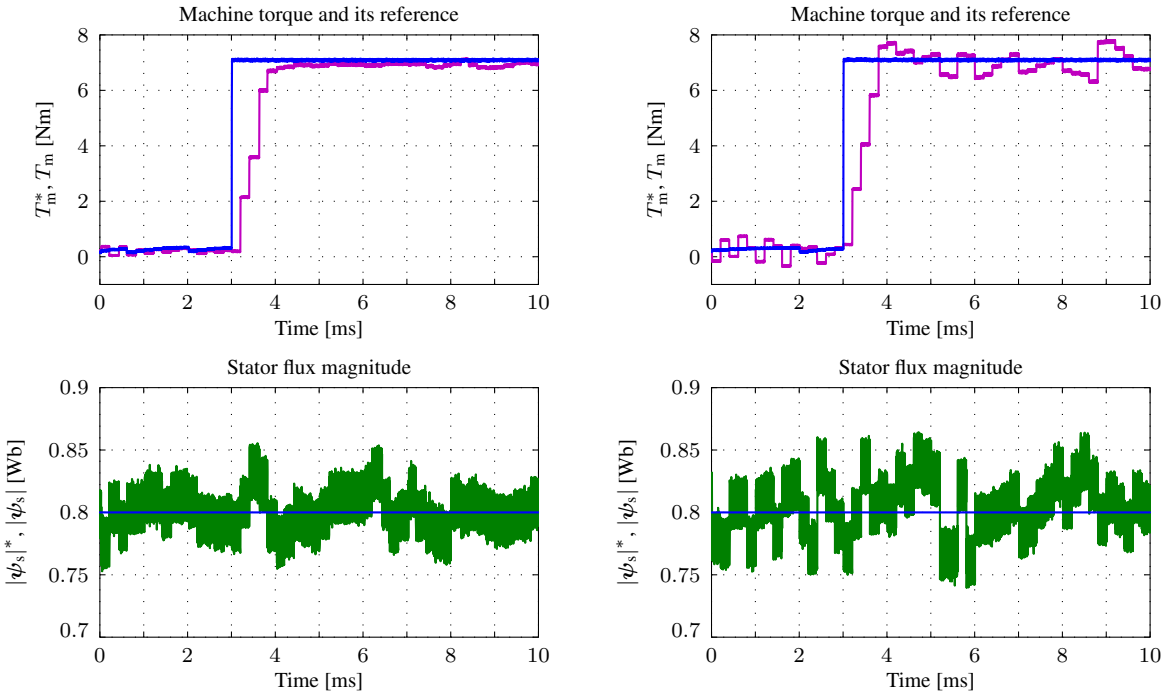
$$\Delta v_\mathrm{c}(k + 1) = \Delta v_\mathrm{c}(k + t_\mathrm{sw}) + \frac{T_\mathrm{s} - t_\mathrm{sw}}{C} \left(\sum_{i=1}^{3} i_{\mathrm{n}j}(k + t_\mathrm{sw}) \right). \tag{4.21}$$

Furthermore, the VSP2TC cost function which is given by equation (4.10) has to be extended for the voltage balancing:

$$j_{\text{vsp2tc, npc}} = j_{\text{vsp2tc}} + w_{\text{quad, npc}} \cdot \left(\Delta v_{\text{c}}(k + t_{\text{sw}})\right)^2 + \left(\Delta v_{\text{c}}(k+1)\right)^2 \tag{4.22}$$

4.4.2 Experimental results

In order to verify the VSP2TC algorithm for three-level NPC inverters, it was implemented on the three-level inverter test bench. As already mentioned, the calculation effort for VSP2TC is significantly higher than for PTC. Additionally, more voltage vectors and switching states have to be evaluated compared to a two-level inverter. Furthermore, the reduction of ripples is expected to be lower compared to two-level inverters because three-level inverters can produce more voltage vectors. In order to execute the control algorithm in real-time, the controller frequency had to be reduced to $5\,\text{kHz}$ which corresponds to a sampling time of $200\,\mu\text{s}$. As for PTC, in this case $w_{\text{quad, npc}}$ was set to 156 and the maximum stator current magnitude was limited to $10\,\text{A}$.



(a) VSP2TC (b) PTC

Figure 4.17: Torque reference step and stator flux magnitude

Figure 4.17(a) shows a torque reference step from almost zero to full nominal torque for VSP2TC which was produced by changing the speed reference from $1415\,\text{rpm}$ to $2830\,\text{rpm}$. The speed controller output was limited to the nominal machine torque. Figure 4.17(b) shows the same experiment for PTC. It is clearly visible that VSP2TC can reduce torque ripples and that the dynamic behavior is not deteriorated. Furthermore, it can also clearly be seen that VSP2TC does not deteriorate the control result for the stator flux magnitude.

4.4.3 Evaluation

Although VSP2TC for three-level NPC inverters effectively reduces torque ripples, a very high calculation effort is necessary for a successful real-time implementation of VSP2TC for three-level inverters. For this reason the controller frequency had to be decreased from $12\,\text{kHz}$ for normal PTC to $5\,\text{kHz}$ for VSP2TC which also increases ripples on the controlled variables. Because of this the applicability of VSP2TC for three-level inverters is limited—a similar control result could probably also be obtained for PTC with a higher sampling frequency. Thus, for VSP methods a reduction of the necessary calculation effort can lead to huge improvements of the control result because then these algorithms can be executed with a higher sampling rate.

4.5 PTC for three-level FC inverters

As shown in [44], it is also possible to apply the PTC algorithm to three-level FC inverters driving an IM. Similar as for the NPC inverter, no changes of the basic algorithm and of the prediction equations are necessary. Of course, for the FC inverter a different voltage balancing algorithm has to be implemented than for NPC inverters. In this case the 19 voltage vectors can be produced by 64 different switching states, i.e. more redundant switching states are available. Although at first sight the necessary calculation effort seems to be higher than for a three-level NPC inverter, the voltage vector selection can be decoupled from the selection of the optimum switching state and thus a very efficient implementation in real-time is possible.

4.5.1 Decoupling of voltage vector and switching state selection

As explained in chapter 2.5.3, in every phase of an FC inverter four different switching states are possible which leads to 64 possible switching states. Since a zero output voltage in a phase can be produced by *two* different switching states and because every phase has its own flying capacitor, the voltage balancing for an FC leg is independent from the other legs. Furthermore, as mentioned in chapter 2.5.3.3, one of the two zero switching states in a phase increases the FC voltage in this phase while the other one will decrease it, depending on the sign of the current drawn from that inverter leg. A positive or a negative switching state does not affect the FC voltage. Consequently, it is possible to generate one of the three available output voltages in a leg ($0.5V_{\text{dc}}$, $0\,\text{V}$ or $-0.5V_{\text{dc}}$) while at the same time maintaining the FC voltage balance, simply by choosing the appropriate zero switching state if $0\,\text{V}$ should be delivered at the phase leg's output clamp. Thus, for an FC inverter it is possible to determine in a first step the optimum voltage vector and in a second one the optimum switching state (regarding the voltage balance) which produces this voltage vector.

Figure 4.18 shows this decoupling principle for an FS-MPC algorithm in a general and simplified way: First, the optimum voltage vector is determined where $\boldsymbol{y}$ are the controlled variables and $\boldsymbol{y}^*$ their references. When the optimum voltage vector $\boldsymbol{v} = (v_\alpha\, v_\beta)^T$ is found, the optimum switching state can be determined. Therefore, the phase currents $\boldsymbol{i} = (i_a\, i_b\, i_c)^T$, the flying capacitor voltages $\boldsymbol{v}_\text{c} = (v_{\text{c}1}\, v_{\text{c}2}\, v_{\text{c}3})^T$ and the switching state $\boldsymbol{s}_\text{old}$ from the previous sample have to be known, too.

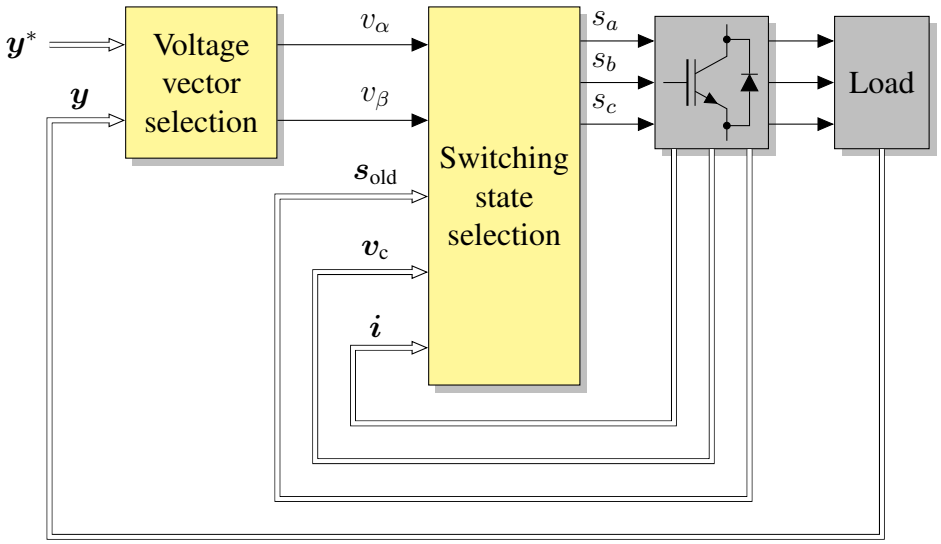


Figure 4.18: FC inverter, decoupling of voltage vector and switching state selection

4.5.2 Control algorithm

As already mentioned, the control algorithm can be split into the voltage vector selection and into a subsequent selection of the optimum switching state. The prediction equations for the voltage vector selection are exactly the same as for a simple two-level inverter but in this case 19 instead of only 7 voltage vectors have to be tested. Thus, the cost function for the first part of the algorithm is also the same as described in chapter 4.1.2.2.

After that the best switching state which produces the optimum voltage vector has to be determined. In this case the control task is to keep the flying capacitor voltages in all three phases as close as possible to their reference values $0.5V_{\text{dc}}$. The basic FC voltage balancing mechanism is explained in chapter 2.5.3.3. For the case that a positive or a negative switching state is applied to phase j, the corresponding FC voltage will not change, i.e.

$$v_{\text{c}j}(k+1) = v_{\text{c}j}(k). \tag{4.23}$$

If the switches S_{j1} and S_{j3} are *on*, the FC voltage change can be calculated by applying the Euler-forward approximation to equation (2.25):

$$v_{\text{c}j}(k+1) = v_{\text{c}j}(k) + \frac{T_{\text{s}}}{C} i_j(k) \tag{4.24}$$

where T_{s} is the sampling time, C the capacitance and i_j is the current drawn from phase j. Analogously, if S_{j2} and S_{j4} are *on*, the FC voltage change results to

$$v_{\text{c}j}(k+1) = v_{\text{c}j}(k) - \frac{T_{\text{s}}}{C} i_j(k). \tag{4.25}$$

In the same way as for the NPC inverter, the load is seen as a current source for the voltage balancing, i.e. the phase currents $i_j(k)$ are assumed to be approximately constant during one sample. During the experiments this approximation has been proven to provide an acceptable voltage balancing result. If necessary, a more accurate model could also be used.

Finally, the cost function for the voltage balancing can be set up:

$$j_{\text{vb, lin, fc}} = \sum_{i=1}^{3} |0.5V_{\text{dc}} - v_{ci}(k+1)| \tag{4.26}$$

for a linear (L_1-norm) and

$$j_{\text{vb, quad, fc}} = \sum_{i=1}^{3} (0.5V_{\text{dc}} - v_{ci}(k+1))^2 \tag{4.27}$$

for a quadractic (L_2-norm) cost function. As the selection of the switching state is made after the optimum voltage vector has been found, no additional weighting factor is necessary which is an important advantage.

4.5.3 Experimental results

The previously described PTC algorithm was experimentally tested on the three-level inverter test bench. The algorithm was executed with a sampling time of $83.33\,\mu\text{s}$ which corresponds to $12\,\text{kHz}$. As in the previous experiments, the DC link was powered with $550\,\text{V}$ by a $3\,\text{kW}$ DC power supply. The parameters of the used induction machine are given in table B.2. For the implementation quadratic cost functions were used. The flux weighting factor was set to $w_{\text{quad}} = 156$. As previously explained, for the voltage balancing no additional weighting factor is necessary because of the decoupling strategy.

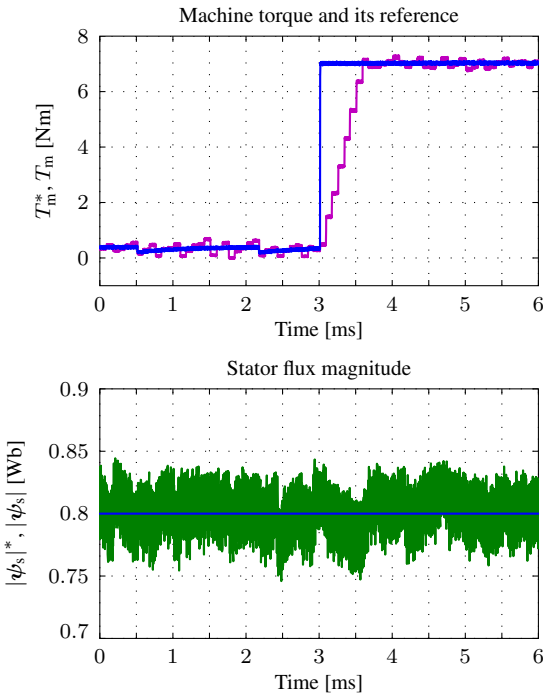


Figure 4.19: Torque reference step and stator flux magnitude

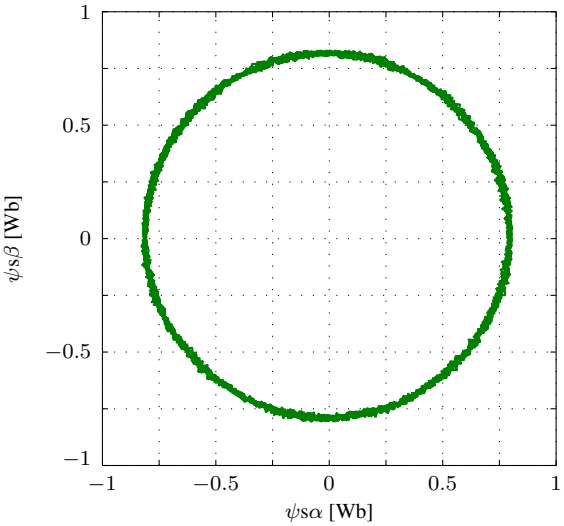


Figure 4.20: Steady-state stator flux in $\alpha\beta$ coordinates at $2830\,\text{rpm}$, no load

In the first conducted experiment a torque reference step from almost zero to full nominal torque (7.42 Nm) was conducted which can be seen in Figure 4.19. In order to operate the test bench safely, it was conducted by changing the speed reference from half nominal (1415 rpm) to full nominal speed (2830 rpm). In the upper plot the machine torque T_m and its reference T_m^* can be seen. The torque follows its reference quickly and without any problems. The plot below shows the stator flux magnitude $|\boldsymbol{\psi}_\mathrm{s}|$ and its reference $|\boldsymbol{\psi}_\mathrm{s}|^*$ which was set to 0.8 Wb. It is clearly visible that the voltage vector selection works without any problems and that the control of both the stator flux magnitude and the machine torque is possible.

Another experiment is shown in Figure 4.20: In this case the stator flux $\boldsymbol{\psi}_\mathrm{s}$ is shown in $\alpha\beta$ coordinates. The flux was recorded at 2830 rpm. As expected, both components form a circle.

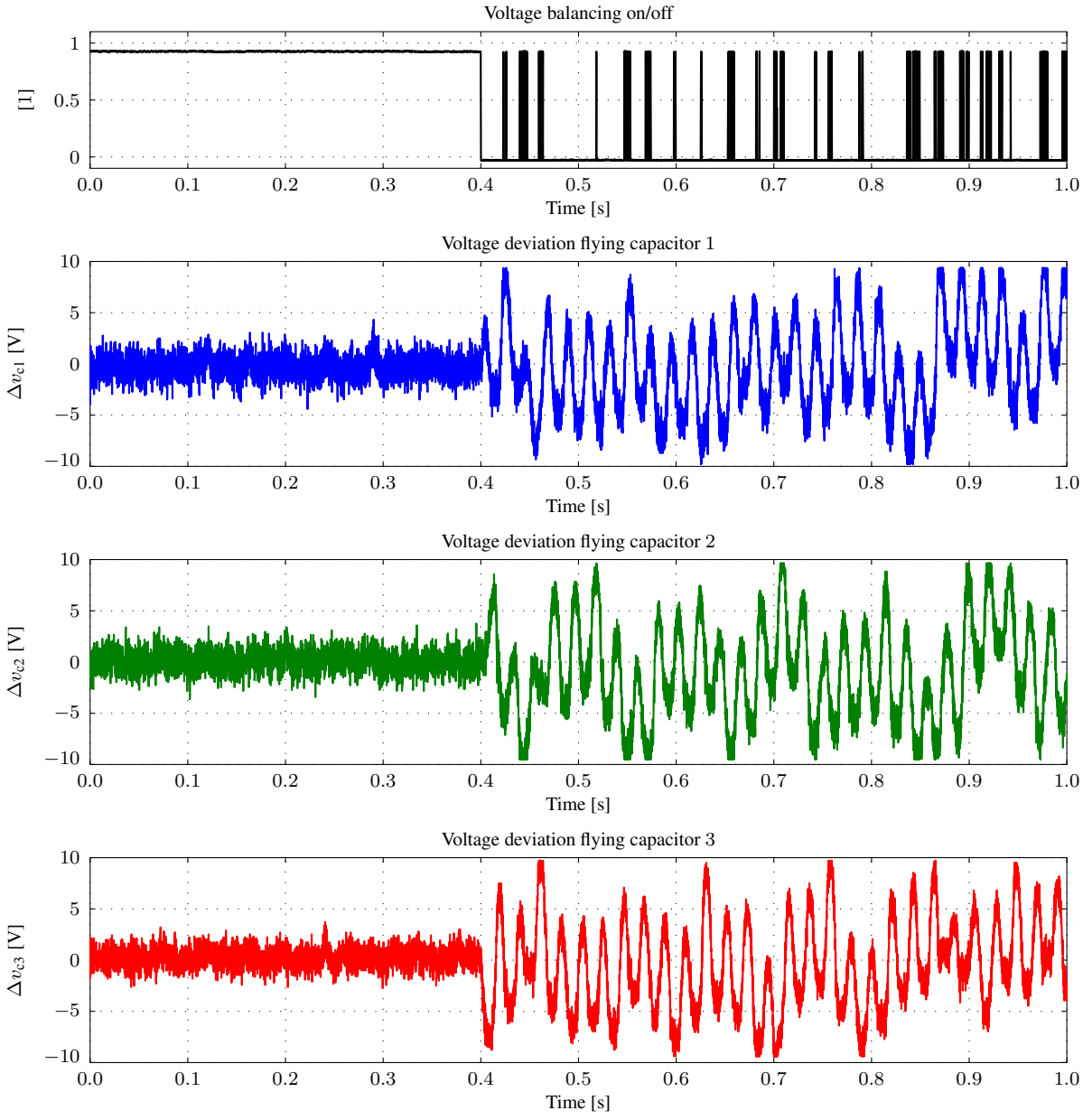


Figure 4.21: Voltage balancing test at 2830 rpm and no load

The last shown experiment in Figure 4.21 was conducted in order to prove that the proposed decoupled control algorithm works as expected. For this reason the machine was operated at full nominal speed and at $0.4\,\mathrm{s}$ the voltage balancing algorithm was manually disabled, i.e. the cost function (equation (4.27)) was set to zero for *all* possible switching states. Then, the first one of all possible switching states is chosen and no voltage balancing is performed. In order to operate the test bench safely, the voltage balancing algorithm was enabled automatically as long as any flying capacitor voltage differed more than $\pm 10\,\mathrm{V}$ from its reference ($275\,\mathrm{V}$). In order to visualize the effectiveness of the proposed voltage balancing algorithm, only the deviations of the flying capacitor voltages from their references (Δv_{ci}, $i = 1 \ldots 3$) are shown in Figure 4.21. From $0\,\mathrm{s}$ to $0.4\,\mathrm{s}$, i.e. before the voltage balancing was manually disabled, all three FC voltages only differ about $\pm 3\,\mathrm{V}$ from their references. However, as soon as the balancing is disabled manually, much larger deviations of the FC voltages can be seen. If the balancing algorithm would not be enabled automatically for a safe operation of the test bench, these deviations would become larger and could even destroy the inverter. This experiment clearly demonstrates that the proposed control algorithm works as expected.

4.5.4 Evaluation

The shown experimental results clearly verify that the PTC algorithm can be successfully applied to FC inverters driving an induction motor. By using the proposed strategy to decouple the voltage vector selection from the determination of the optimum switching state, the calculation effort can be reduced: Instead of having to evaluate all 64 different switching states at once, in a first step only 19 different voltage vectors have to be tested. Then, for the subsequent switching state selection in the worst case a decision between ten possibilities has to be made. Thus, in the worst case only 29 possible combinations need to checked, i.e. the resulting calculation effort for PTC of FC inverters is in the same range as for PTC of NPC inverters. Furthermore, because of the higher number of redundant switching states, a slightly better control result can be expected than for an NPC inverter.

CHAPTER 5

Predictive Current Control for induction machines

This chapter presents the Predictive Current Control (PCC) algorithm for induction machines, an FS-MPC alternative to FOC which is introduced and explained in chapter 2.6.2. Compared to FOC, it uses an FS-MPC algorithm to determine the optimum switching state for the next sampling cycle. The basic algorithm will be explained in chapter 5.1. It is followed by a comparison to PTC, especially regarding the number of necessary calculations.

After that a heuristic extension is presented which *significantly* reduces the necessary calculation effort, especially for higher prediction horizons and for multilevel inverters. The basic algorithm for the heuristic voltage vector selection is presented and explained in chapter 5.2. The algorithm is verified experimentally and compared to a full enumeration of all possible switching states.

Thereafter, an extension to PCC with a variable switching point, namely Variable Switching Point Predictive Current Control (VSP2CC) is introduced and explained. Experimental results demonstrate the algorithm's capability to *significantly* reduce current ripples. Since FS-MPC strategies, unlike PWM-based methods, do in general not lead to constant switching frequencies, it is experimentally verified that PCC leads to very low switching frequencies at low speeds. These low switching frequencies lead to high current ripples. In contrast to that, VSP methods lead to a strong improvement of the control result at lower speeds which is also verified experimentally.

As the VSP2CC strategy comes with an increased calculation effort compared to PCC, VSP2CC is combined with a heuristic voltage vector preselection which is also verified experimentally.

Afterwards, PCC, PCC with heuristic voltage vector preselection, VSP2CC and VSP2CC with heuristic voltage vector preselection are extended to three-level NPC and FC inverters with additional voltage balancing terms in the cost function. Furthermore, for FC inverters the same voltage balancing strategy as presented in chapter 4.3.1 is applied. Experimental results are presented to verify the algorithms and to demonstrate their capability.

5.1 Basic Predictive Current Control algorithm

FS-PCC, compared to FS-PTC, utilizes the FOC principle: Instead of performing the control in stator fixed coordinates, a coordinate transformation from stator fixed $\alpha\beta$ to rotor flux oriented dq coordinates is done such that an independent control of both the flux-producing current i_d and the torque-producing current i_q is possible.

5.1.1 Basic equations

For $\omega_\mathrm{k} = 0$ equation (2.34) can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{i}_\mathrm{s} = -\frac{1}{\tau_\sigma}\boldsymbol{i}_\mathrm{s} + \frac{1}{r_\sigma \tau_\sigma}\left(\boldsymbol{v}_\mathrm{s} + \frac{k_\mathrm{r}}{\tau_\mathrm{r}}\boldsymbol{\psi}_\mathrm{r} - j\omega_\mathrm{el}\boldsymbol{\psi}_\mathrm{r}\right). \tag{5.1}$$

Simplified, this results to

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{i}_\mathrm{s} = -\frac{1}{\tau_\sigma}\boldsymbol{i}_\mathrm{s} + \frac{1}{r_\sigma \tau_\sigma}\left(\boldsymbol{v}_\mathrm{s} - \boldsymbol{v}_\mathrm{emf}\right) \tag{5.2}$$

with the back-electromotive force (back-EMF) voltage

$$\boldsymbol{v}_\mathrm{emf} = -\frac{k_\mathrm{r}}{\tau_\mathrm{r}}\boldsymbol{\psi}_\mathrm{r} + jk_\mathrm{r}\omega_\mathrm{el}\boldsymbol{\psi}_\mathrm{r}. \tag{5.3}$$

For the rotor flux estimation equation (2.35) is used ($\omega_\mathrm{k} = 0$):

$$\tau_\mathrm{r}\frac{\mathrm{d}\boldsymbol{\psi}_\mathrm{r}}{\mathrm{d}t} + \boldsymbol{\psi}_\mathrm{r} = L_\mathrm{m}\boldsymbol{i}_\mathrm{s} + j\omega_\mathrm{el}\tau_\mathrm{r}\boldsymbol{\psi}_\mathrm{r} \tag{5.4}$$

The discrete-time equations (sampling time T_s) can be easily obained by applying the Euler-forward transformation. The current for the next sample can be predicted with

$$\boldsymbol{i}_\mathrm{s}(k+1) = \left(1 - \frac{T_\mathrm{s}}{\tau_\sigma}\right)\cdot \boldsymbol{i}_\mathrm{s}(k) + \frac{T_\mathrm{s}}{r_\sigma \tau_\sigma}\left(\boldsymbol{v}_\mathrm{s}(k) - \boldsymbol{v}_\mathrm{emf}(k)\right). \tag{5.5}$$

The back-EMF can be estimated with

$$\boldsymbol{v}_\mathrm{emf}(k) = -\frac{k_\mathrm{r}}{\tau_\mathrm{r}}\boldsymbol{\psi}_\mathrm{r}(k) + jk_\mathrm{r}\omega_\mathrm{el}\boldsymbol{\psi}_\mathrm{r}(k). \tag{5.6}$$

For the rotor flux estimation the equation

$$\boldsymbol{\psi}_\mathrm{r}(k+1) = \left(1 - \frac{T_\mathrm{s}}{\tau_\mathrm{r}} + j\omega_\mathrm{el}T_\mathrm{s}\right)\boldsymbol{\psi}_\mathrm{r}(k) + \frac{L_\mathrm{m}T_\mathrm{s}}{\tau_\mathrm{r}}\cdot \boldsymbol{i}_\mathrm{s}(k) \tag{5.7}$$

can be used.

For speed control, in the same way as for FOC, DTC and PTC, a simple overlayed PI or PID controller is used.

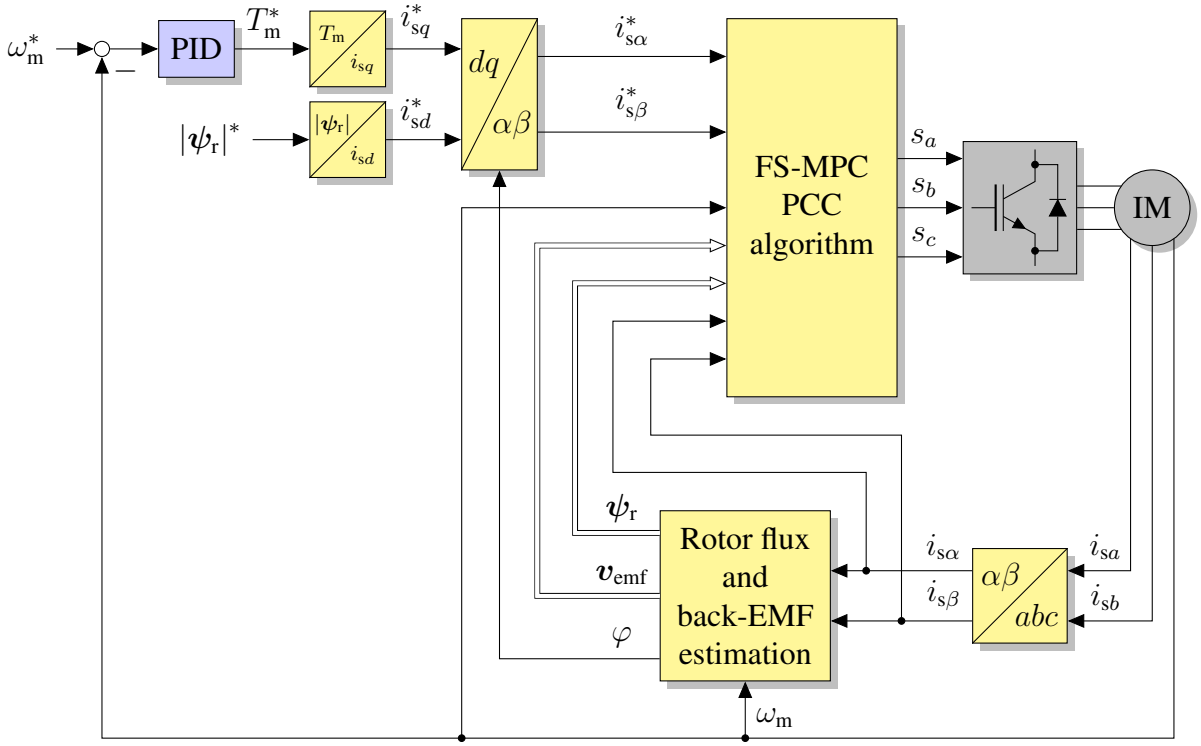


Figure 5.1: Basic PCC scheme

5.1.2 Control algorithm

As it can be clearly seen from equation (5.2), the current control loop can be modeled as a linear first-order system with external disturbance. The back-EMF voltage v_{emf} is changing slowly compared to the sampling frequency and hence, it can be assumed to be constant for the whole prediction horizon. Figure 5.1 shows the basic control scheme.

As already mentioned, the torque reference is generated by a conventional speed PID controller. The reference value for the rotor flux magnitude is set to a constant value which can be decreased if field-weakening operation is desired. The corresponding reference values for the field- and torque-producing currents i_{sd} and i_{sq} are given by

$$i_{sd}^* = \frac{|\psi_r|^*}{L_m} \text{ and} \tag{5.8}$$

$$i_{sq}^* = \frac{T^*}{\frac{3}{2} \cdot \frac{L_m}{L_r} |\psi_r|^*}. \tag{5.9}$$

In FOC the PID controllers for the currents are operating in rotor flux-oriented dq coordinates and the transformation to $\alpha\beta$ coordinates is done afterwards. In contrast to that, for PCC the current references, i_{sd}^* and i_{sq}^*, are transformed to $\alpha\beta$ current references, $i_{s\alpha}^*$ and $i_{s\beta}^*$, and the controller operates in $\alpha\beta$ coordinates. This choice is made as it is computationally more efficient to transform the references (one operation) and then to do the control in stationary coordinates. Otherwise, every voltage vector that is produced by the inverter would have to be transformed to dq coordinates (seven operations for a two-level inverter). The basic principle is the same as

for PTC but in this case either a linear cost function,

$$j_{\text{pcc, lin}} = |i^*_{\text{s}\alpha} - i_{\text{s}\alpha}(k+1)| + \left|i^*_{\text{s}\beta} - i_{\text{s}\beta}(k+1)\right|, \tag{5.10}$$

or a quadratic cost function,

$$j_{\text{pcc, quad}} = \left(i^*_{\text{s}\alpha} - i_{\text{s}\alpha}(k+1)\right)^2 + \left(i^*_{\text{s}\beta} - i_{\text{s}\beta}(k+1)\right)^2, \tag{5.11}$$

has to be minimized. Of course, as for PTC, additional terms, e.g. for reducing the switching frequency or to penalize large changes of the actuating variables could be used as well. It has to be noted that for PCC the current limitation does not necessarily have to be included into the cost function: As the currents are controlled directly, it is enough to limit the current references. The minimization process is the same as for PTC, i.e. the cost function is calculated for every possible voltage vector which can be produced by the inverter and then the optimum switching state which produces this voltage vector is chosen and applied in the next sample.

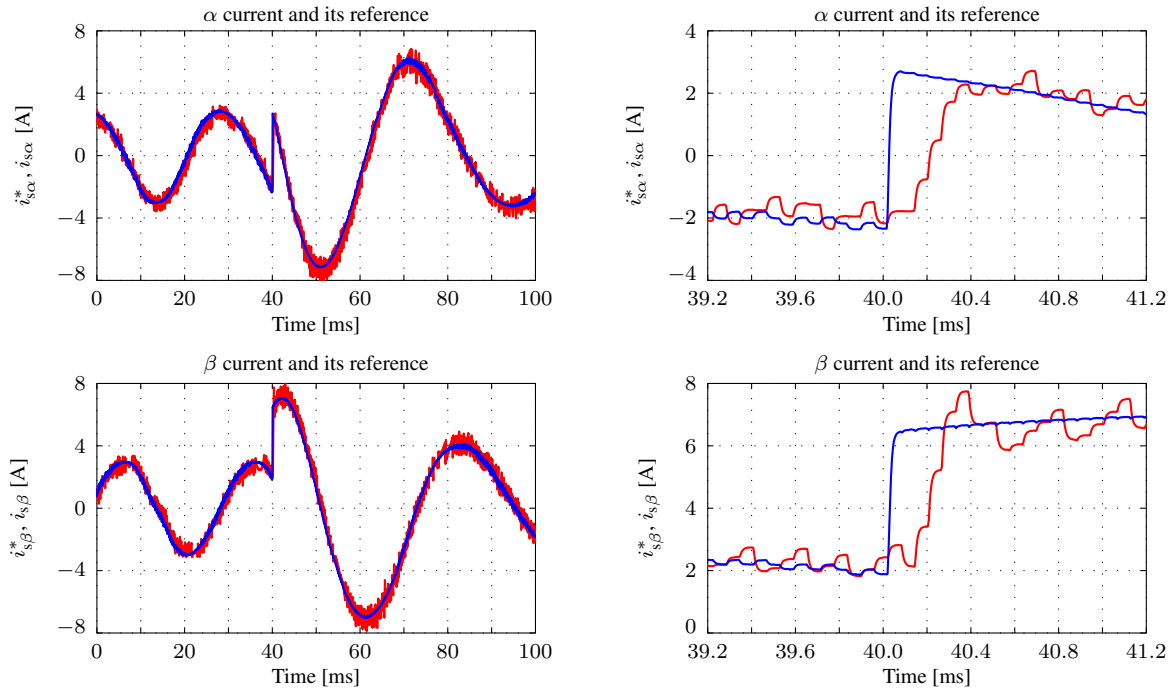
5.1.3 Experimental results

The PCC algorithm was evaluated on the two-level inverter test bench. Several experiments at different operating points were conducted in order to use it as a reference for improved and more sophisticated control algorithms. All experiments shown in this chapter were made using the quadratic cost function given in equation (5.11). The algorithm was executed with a sampling time $T_\text{s} = 61.44\,\mu\text{s}$ which corresponds to a controller frequency of about $16\,\text{kHz}$. The DC link voltage was $580\,\text{V}$. All machine parameters can be found in Table B.1.

Figure 5.2 shows the current control result when the machine was operated: The reference value for the rotor flux magnitude was set to $|\boldsymbol{\psi}_\text{r}|^* = 0.8\,\text{Wb}$ and the torque reference was produced by the speed PI controller. The current references $\boldsymbol{i}^*_\text{s}$ were then calculated as described in the PCC algorithm. The sinusoidal current reference steps were produced by changing the speed reference from $2000\,\text{rpm}$ to $1000\,\text{rpm}$. It is clearly visible that the current controller has absolutely no problems to track its references. In order to show the dynamic behavior during transients, Figure 5.2(b) shows a zoom of the current reference steps during the same experiment. After the normal delay of two samples (because of the sample and hold operation and the calculation time) the currents quickly follow their references and after about three samples they are already very close to their references. Of course, as usual for FS-MPC methods, quite high current ripples can be noticed. In the lower graph of Figure 5.2(b) the problem explained in chapter 2.7.5.4 can be observed: In this case a switching state leading to a high negative slope of $i_{\text{s}\beta}$ is selected and afterwards one which leads to a small positive slope which leads to high ripples.

In Figure 5.3 a speed reference change from $1000\,\text{rpm}$ to $2000\,\text{rpm}$ was conducted. It is clearly visible that the speed PI controller has no problems to track its reference. Figure 5.4 shows the effect of a load torque impact on the speed: At about $0.13\,\text{s}$ a $4\,\text{Nm}$ load torque was applied to the machine which was rotating at $2000\,\text{rpm}$. The experiment verifies that also in this case a good control result can be obtained.

In Figure 5.5(a) the stator currents at $2000\,\text{rpm}$ without load torque are presented, Figure 5.5(b) shows the stator currents when a load torque of $4\,\text{Nm}$ is present. Compared to PTC



(a) larger time scale

(b) zoom during reference steps

Figure 5.2: $\alpha\beta$ current reference steps

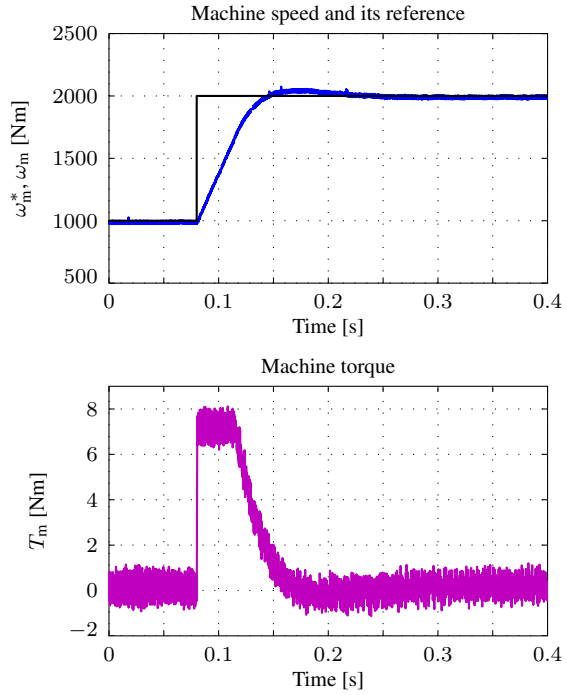


Figure 5.3: Speed reference step and produced machine torque

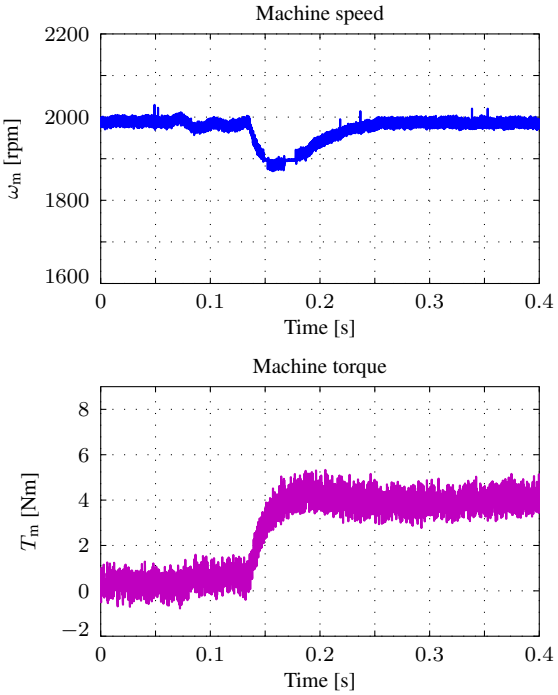
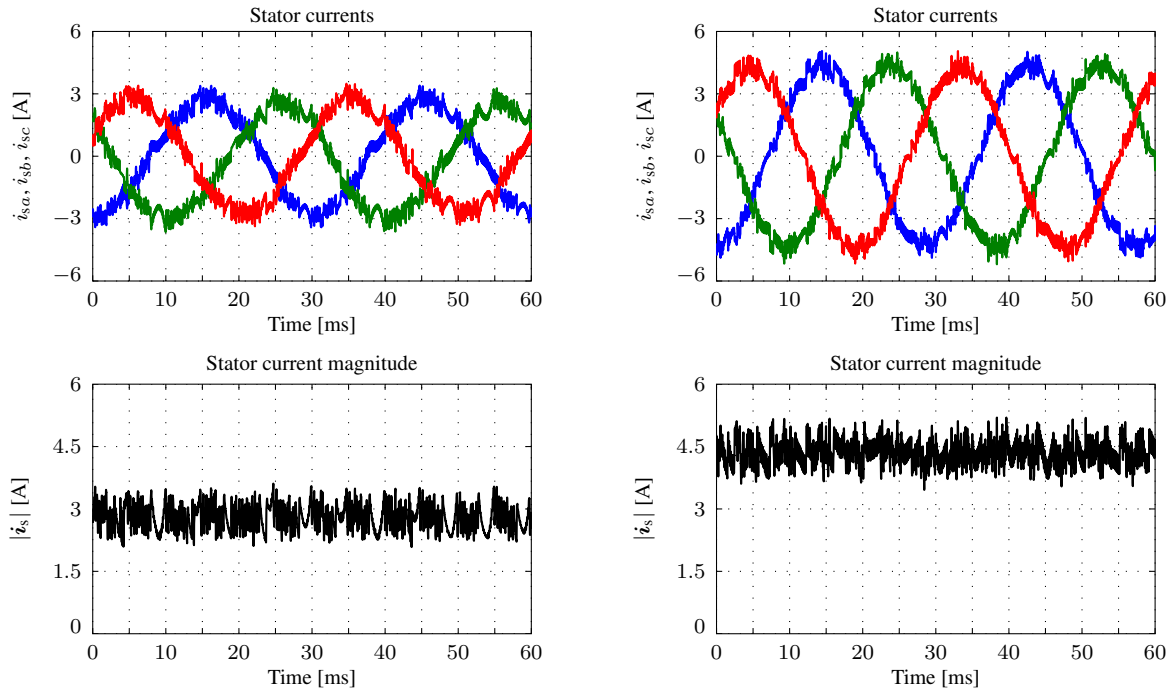


Figure 5.4: Load torque impact (4 Nm at 2000 rpm)



(a) No load torque

(b) 4 Nm load torque

Figure 5.5: Steady-state stator currents at 2000 rpm

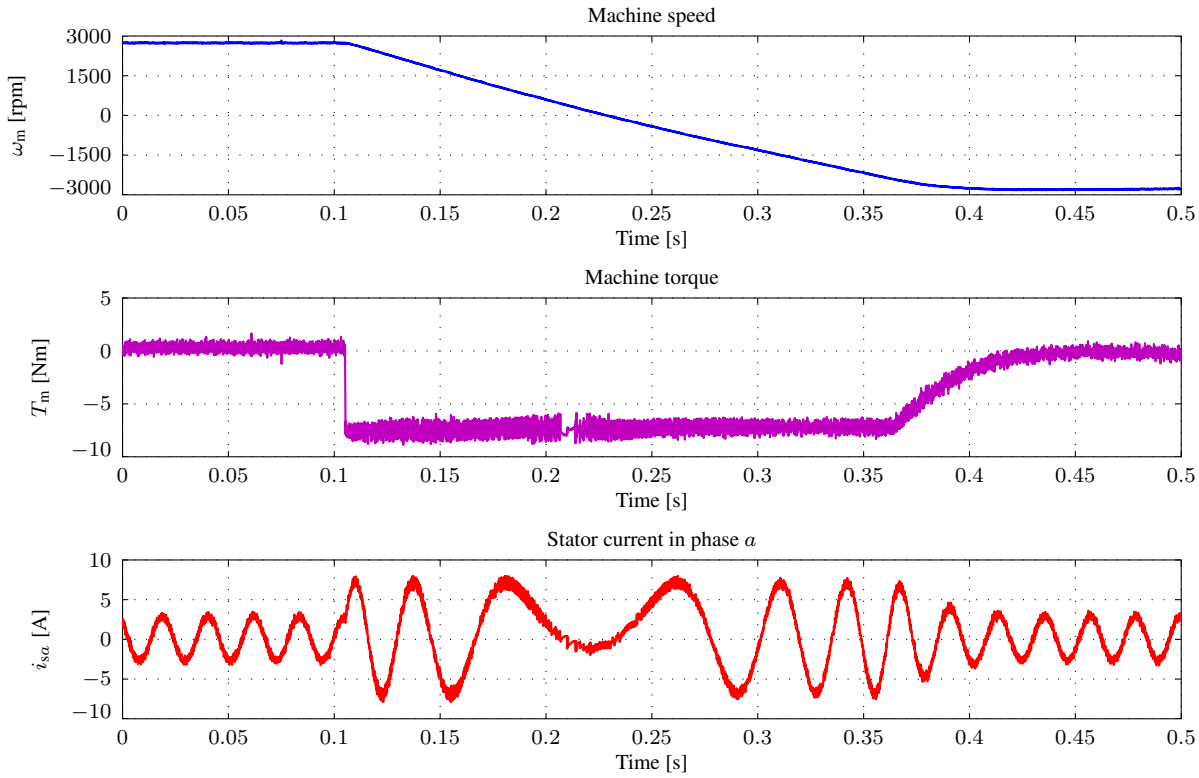


Figure 5.6: Speed reversal from positive nominal to negative nominal speed

(Figure 4.6) it is obvious—especially without load—that the stator currents have a better quality. This is due to the fact that PTC controls the currents *indirectly* via the machine torque and stator flux magnitude. The average switching frequency per IGBT was about 3.3 kHz.

In order to verify that the algorithm can control the drive at different operating points, a speed reversal from positive nominal to negative nominal speed was conducted. The results can be seen in Figure 5.6. Although the produced machine torque shows a slightly higher ripple compared to PTC (Figure 4.7), the phase currents are *significantly* improved. This is due to the fact—in contrast to PTC—that the stator currents are controlled.

5.1.4 Evaluation

Compared to PTC, PCC has three main advantages:

1. As the back-EMF and rotor flux estimations only have to be executed once in a sample (they can even be assumed to be constant over the whole prediction horizon) and as only two equations have to be evaluated for both currents, the calculation effort for PCC is less than for PTC. Compared to the number of voltage vectors which have to be evaluated, the calculation effort for the Park transformation is negligible.

2. In contrast to PTC, two currents have to be controlled. Thus, no weighting factor is necessary. Although PTC also requires only one weighting factor, a non-optimal weighting factor leads to deteriorated control results.

3. In contrast to PTC, PCC offers possibilties for further simplifications: If the back-EMF is considered to be constant over the prediction horizon, only a linear first-order system with the back-EMF as "external" disturbance has to be controlled.

By comparing the experimental results of PTC to those of PCC, another advantage of PCC compared to PTC becomes obvious: As PTC only controls the stator flux magnitude and the machine torque but not the currents, PCC produces much better currents compared to the ones delivered by PTC.

Since PCC offers several advantages compared to PTC, it should be preferred.

5.2 Heuristic Finite-Set Model Predictive Current Control

5.2.1 Motivation

As already explained in chapter 2.7.5.4, the necessary calculation effort for FS-MPC methods rises exponentially with the prediction horizon. For simple systems a prediction horizon of only one sample might be enough; however, for more complex systems and if a higher prediction horizon is desired, a successful real-time implementation of the control algorithms is normally not feasible. For this reason it is necessary to find methods to reduce the calculation effort. If possible, these methods should also be applicable to multilevel inverters. Since all algorithms for integer optimization which are known from operations research can only reduce the *average* calculation effort, a heuristic method to reduce the calculation effort [45] is proposed.

5.2.2 Basic principle

It is well-known and obvious that the discrete-valued optimum is not necessarily the closest point to the continuous-valued optimum. Because of this it is *not* enough to solve the continuous (relaxed) optimization problem and then to determine the closest discrete-valued point to it. However, especially for first-order and not very complex systems, the following assumptions can be made:

1. In most cases the discrete-valued optimum lies close to the continuous-valued optimum although it is not necessarily the point which is closest to the continuous-valued optimum.

2. The more discrete points are available and the closer these are to each other, the more accurate is the above assumption, i.e. the presented heuristic method is supposed to work better for inverters which have more voltage levels.

It should be noted that these two assumptions only present a general *heuristic* and are not mathematically proven. By taking a look at the voltage vectors produced by a two-level inverter (Figure 5.7), all feasible voltage vectors (i.e. voltage vectors which can by synthesized by the inverter) are within the shown hexagon. Voltage vectors outside the hexagon exceed the constraint of the available DC link voltage and cannot be produced by the inverter. Similar to

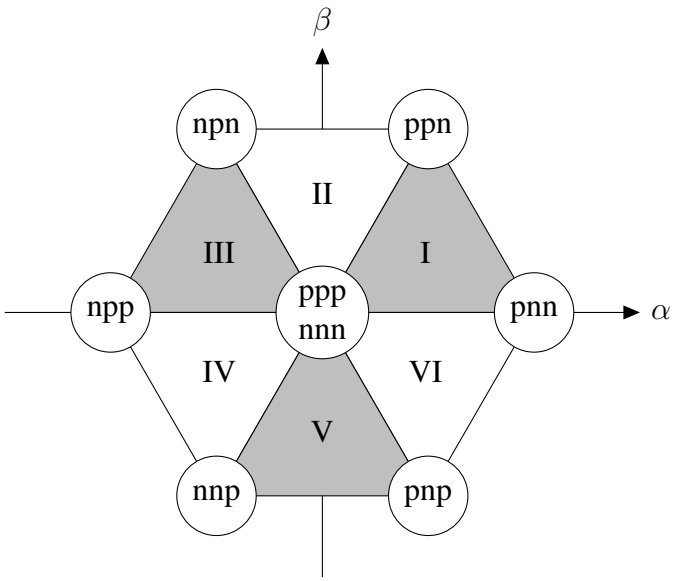


Figure 5.7: Voltage vectors produced by a two-level inverter and division into sectors

Space Vector Modulation (SVM) [14], the hexagon spanned by the seven voltage vectors can be divided into six sectors I...VI which are shown in Figure 5.7. It can be clearly seen that in all sectors the three closest points to every continuous-valued point within a sector are located at the corners of this sector, e.g. the three voltage vectors closest to any continuous-valued point in sector III are the zero voltage vector (ppp or nnn), npn and npp.

Now, by taking into account the first point stated before and assuming that the discrete-valued optimum is in most cases at least the third closest discrete point with respect to the continuous-valued optimum, a heuristic method can be formulated as follows:

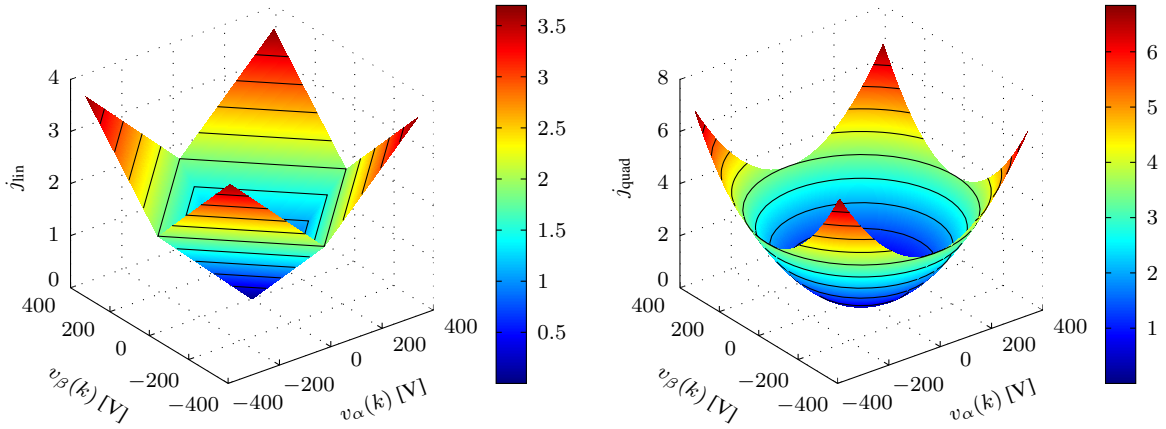
1. Determine the continuous-valued optimum.
2. Determine the sector in which this continuous-valued optimum is located.
3. Only the three points closest to this continuous-valued optimum are candidates for the discrete optimum.
4. Determine the discrete optimum out of the three closest points via FS-MPC.

For higher prediction horizons this principle can be applied as well: Then, for every predicted step the continuous-valued optimum has to be determined and for the subsequent FS-MPC optimization only the three points closest to these continuous optimum points are chosen as candidates.

5.2.3 Validation of the heuristic preselection principle

In order to illustrate the described preselection principle for the voltage vectors, both the linear (equation (5.10)) and the quadratic (equation (5.11)) cost function values for the current control loop are visualized in Figure 5.8. Equation (5.5) was used for the calculation of the predicted current. Furthermore, for these plots a sampling time $T_\mathrm{s} = 100\,\mu\mathrm{s}$, $r_\sigma = 4.2779\,\Omega$ and $\tau_\sigma = 4.635\,\mathrm{ms}$ (machine data for the three-level inverter test bench) was used.

Although these plots only show one single prediction step and were made under the assumption that the back-EMF voltage $\boldsymbol{v}_\mathrm{emf}(k)$, the measured stator current $\boldsymbol{i}_\mathrm{s}(k)$ and the stator current reference $\boldsymbol{i}_\mathrm{s}^*(k)$ are zero, it is important to note that these values do *not* change the general shape of the cost function.



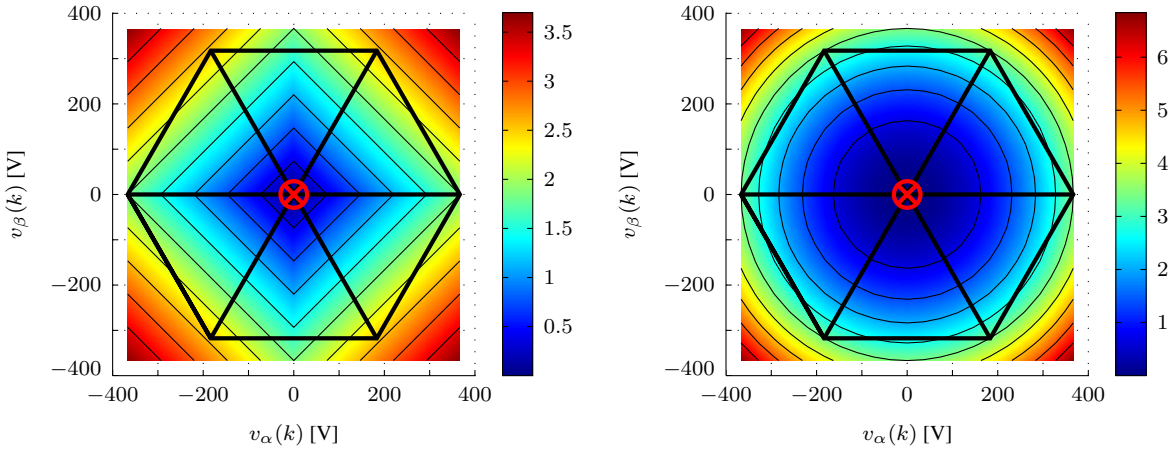
(a) Linear cost function

(b) Quadratic cost function

Figure 5.8: Three-dimensional visualization of the cost functions for one prediction step ($\boldsymbol{v}_\mathrm{emf}(k) = (0\,\mathrm{V}, 0\,\mathrm{V})^T$, $\boldsymbol{i}_\mathrm{s}(k) = (0\,\mathrm{A}, 0\,\mathrm{A})^T$, $\boldsymbol{i}_\mathrm{s}^*(k) = (0\,\mathrm{A}, 0\,\mathrm{A})^T$)

Figure 5.9 shows two-dimensional plots of the cost functions for the same conditions as in Figure 5.8. Furthermore, all feasible voltage vectors are marked by the overlayed black hexagon. All voltage values lying within this black hexagon (formed by the discrete switching states) can be commanded by the inverter if a continuous-valued optimization is considered.

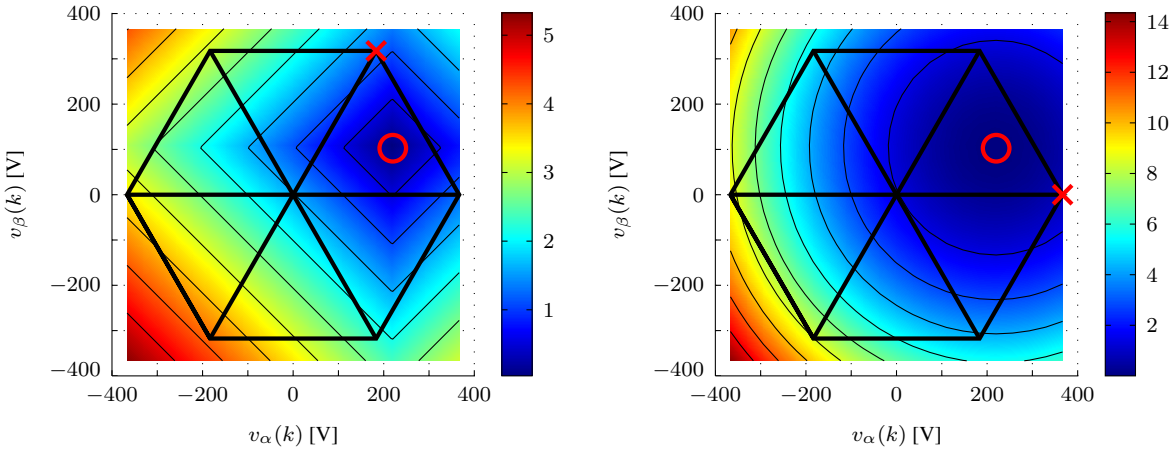
The continuous-valued optimum is marked with a red circle, the discrete optimum with a red x. As expected, when no current is flowing through the machine, no back-EMF is existing and when the current references are equal to zero, the voltage vector leading to a minimum cost function value lies at the origin.



(a) Linear cost function (b) Quadratic cost function

Figure 5.9: Cost functions and discrete-valued optimization points for one prediction step ($\boldsymbol{v}_{\text{emf}}(k) = (0\,\text{V}, 0\,\text{V})^T$, $\boldsymbol{i}_{\text{s}}(k) = (0\,\text{A}, 0\,\text{A})^T$, $\boldsymbol{i}_{\text{s}}^*(k) = (0\,\text{A}, 0\,\text{A})^T$)

Finally, the cost function is visualized for a different operating condition in Figure 5.10: In this case the back-EMF voltage, the measured stator currents and their references are non-zero. From Figure 5.10(a) it is clearly visible that the discrete optimum is not necessarily the discrete point which is closest to the continuous-valued one.



(a) Linear cost function (b) Quadratic cost function

Figure 5.10: Cost functions and discrete-valued optimization points for one prediction step ($\boldsymbol{v}_{\text{emf}}(k) = (55\,\text{V}, 275\,\text{V})^T$, $\boldsymbol{i}_{\text{s}}(k) = (1.2\,\text{A}, 1.7\,\text{A})^T$, $\boldsymbol{i}_{\text{s}}^*(k) = (2.0\,\text{A}, 0.8\,\text{A})^T$)

These plots only show one single prediction step (more prediction steps would result in a higher dimension). However, the cost function plots will in general look very similar for further

prediction steps (then, of course, the predicted stator currents instead of the measured ones have to be used). Thus, the same heuristic principle can also be used for higher prediction horizons. However, since in every prediction step a certain error is made it can be expected that the accuracy of the proposed heuristic preselection is lower for higher prediction horizons. Nevertheless, the proposed algorithm and its accuracy have been verified in simulations for two- and three-level inverters [45, 46]. Even for three and four prediction steps in more than 96% of all cases the correct voltage vector was found. In all conducted simulations no deterioration of the control result, compared to a full enumeration of all possible voltage vectors, could be noticed.

5.2.4 Determination of the continuous-valued optimum

Before the heuristic FS-MPC optimization process can be started, the continuous-valued optimum must be found. Of course, the proposed heuristic method is only useful if the determination of the continuous-valued optimum and the subsequent discrete optimization with the heuristically reduced input set needs less calculation effort compared to a full enumeration of all voltage vectors or switching states.

By taking a closer look at the prediction equation for PCC, equation (5.5), it can be seen that the current control loop is a linear first-order system with an external disturbance, $\boldsymbol{v}_{\text{emf}}$. As already mentioned, the back-EMF is considered to change slowly compared to the sampling time T_{s} and hence, it can be assumed to be constant for the whole prediction horizon.

If now a cost function as in equation (5.10) or (5.11) has to be minimized ($\boldsymbol{v}_{\text{emf}}$ is known), the continuous optimization problem can be solved using either linear (LP) or quadratic programming (QP).

An n-dimensional LP in *inequality form* with m inequality and m_{eq} equality constraints is given as follows [47]:

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T\boldsymbol{x} \tag{5.12}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \tag{5.13}$$

$$\boldsymbol{A}_{\text{eq}}\boldsymbol{x} = \boldsymbol{b}_{\text{eq}}. \tag{5.14}$$

An n-dimensional QP with m inequality and m_{eq} equality constraints can be stated as [47]

$$\min_{\boldsymbol{x}} \quad \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \tag{5.15}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \tag{5.16}$$

$$\boldsymbol{A}_{\text{eq}}\boldsymbol{x} = \boldsymbol{b}_{\text{eq}} \tag{5.17}$$

where $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{A}_{\text{eq}} \in \mathbb{R}^{m_{\text{eq}}\times n}$ and $\boldsymbol{b}_{\text{eq}} \in \mathbb{R}^{m_{\text{eq}}}$.

These LPs and QPs can either be solved with Simplex or Interior Point methods [48, 49].

LPs and QPs can also be solved for discrete input sets; then, these problems are referred to as *Mixed Integer* LPs and QPs (MILPs and MIQPs). Compared to continuous-valued optimization problems, these programs need much more calculation time. Nearly all techniques to reduce the calculation effort necessary for such discrete optimization problems (e.g. Branch and Bound or Branch and Cut methods) can only reduce the *average* calculation time. Compared to the classical application field, for FS-MPC algorithms the optimization problems are normally

much smaller but have to be solved in real-time. For this reason it is *mandatory* to reduce the worst-case calculation effort which is in general not possible. Even for conventional optimization problems heuristics are used in order to decrease the number of necessary calculations. The main goal of these heuristic methods is to find the discrete optimum in most cases or at least a good suboptimal point.

Another problem arises as these LPs and QPs normally need quite much calculation time. If they are solved online, a real-time execution might not be possible. Fortunately, it is possible to solve these optimization problems offline via *Multiparametric Programming* [33,50]: If a linear system given in state-space form (equation (2.1)) should be controlled using a cost function as stated in equation (5.10) or (5.11), an *explicit* solution of this optimization problem can be calculated *offline* with the state vector $\boldsymbol{x}$ as parameter. However, since a controller defined in such a way can only regulate the controlled variables, i.e. the states $\boldsymbol{x}$, back to the origin (zero), the reference values for the states, $\boldsymbol{x}^*$, as well as the back-EMF voltages $\boldsymbol{v}_{\text{emf}}$ have to be considered as "additional states" or parameters. When the multiparametric LP or QP (mpLP or mpQP) is solved offline, the parameter space will be divided into several convex polytopes; in every polytope a different piecewise affine (linear plus offset) control law is valid, i.e. in every polytope i the optimum values for the actuating variables, $\boldsymbol{u}_{\text{opt}}$, can be calculated from the state vector $\boldsymbol{x}$ (which in this case also contains the reference values and the back-EMF voltages) with

$$\boldsymbol{u}_{\text{opt}} = \boldsymbol{H}_i \cdot \boldsymbol{x} + \boldsymbol{k}_i \tag{5.18}$$

where $\boldsymbol{H}_i$ and $\boldsymbol{k}_i$ are obtained for every polytope i when the offline solution is calculated.

The remaining task which has to be solved online is the determination of the correct polytope in state space.

The implementation of algorithms to solve such mpLPs and mpQPs is a quite complex issue. Fortunately, at the Swiss Federal Institute of Technology the Multiparametric Toolbox (MPT) for Matlab® was developed [35,51,52]. With the help of this toolbox the user only has to define the state-space matrices, the cost function, prediction horizon and some additional parameters (if desired) and then, the explicit controller is automatically calculated. Furthermore, the generated controller partitions can also be exported to *C* code which allows a quick implementation on a test bench.

Of course, with the MPT toolbox it is also possible to solve multiparametric MILPs and MIQPs. However, for practical applications, especially if prediction horizons of two and more samples and if multilevel inverters are considered, the offline calculation time on currently available computer hardware quickly reaches several hours and even days. Furthermore, the resulting controller partitions of such integer optimization tasks easily reach numbers of $5{,}000$ to $10{,}000$ polytopes and more. The offline calculation time and the number of polytopes also rise exponentially with the prediction horizon. Thus, for higher prediction horizons and especially for multilevel inverters an offline solution of MILPs and MIQPs is not practical anymore. Of course, the offline calculation time and the number of polytopes also rise with the prediction horizon for LPs and QPs but not as fast as for the integer optimization.

5.2.5 System description for the continuous-valued optimization

In order to obtain the continuous-valued solution of the optimization problem, equation (5.5) has to be rewritten in state-space form. The back-EMF voltages $\boldsymbol{v}_{\text{emf}}$ are assumed to be a

"dummy" state, i.e. that they are changing slowly compared to the sampling time. Furthermore, the discrete-valued optimum has to be found for the three continuous-valued "switching states" $\boldsymbol{s}$ (in abc coordinates) whose values are limited to the range $[-1 \dots 1]$—in this way the voltage constraints can be set very easily. Then, the prediction equation for the next sample results to:

$$\begin{pmatrix} \boldsymbol{i}_\mathrm{s}(k+1) \\ \boldsymbol{v}_\mathrm{emf}(k+1) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{i}_\mathrm{s}(k) \\ \boldsymbol{v}_\mathrm{emf}(k) \end{pmatrix} + \boldsymbol{B}_1 \cdot \boldsymbol{s}(k) \tag{5.19}$$

with $\boldsymbol{A}_1 = \begin{pmatrix} 1 - \frac{T_\mathrm{s}}{\tau_\sigma} & 0 \\ 0 & 1 - \frac{T_\mathrm{s}}{\tau_\sigma} \end{pmatrix}$ and $\boldsymbol{A}_2 = \begin{pmatrix} -\frac{T_\mathrm{s}}{r_\sigma \tau_\sigma} & 0 \\ 0 & -\frac{T_\mathrm{s}}{r_\sigma \tau_\sigma} \end{pmatrix}$. In order to calculate the input matrix $\boldsymbol{B}_1$, the Clarke transformation has to be applied to the three switching states and finally $\boldsymbol{B}_1 = \frac{1}{3} \frac{V_\mathrm{dc} T_\mathrm{s}}{r_\sigma \tau_\sigma} \begin{pmatrix} 1 & -0.5 & -0.5 \\ 0 & 0.5\sqrt{3} & -0.5\sqrt{3} \end{pmatrix}$ can be obtained.

Since such a system representation would only allow to regulate the currents to their origin (0 A in both α and β direction), an extended system representation with the current references $\boldsymbol{i}_\mathrm{s}^*(k+1)$ has to be set up:

$$\begin{pmatrix} \boldsymbol{i}_\mathrm{s}(k+1) \\ \boldsymbol{i}_\mathrm{s}^*(k+1) \\ \boldsymbol{v}_\mathrm{emf}(k+1) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{0} & \boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{i}_\mathrm{s}(k) \\ \boldsymbol{i}_\mathrm{s}^*(k) \\ \boldsymbol{v}_\mathrm{emf}(k) \end{pmatrix} + \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \cdot \boldsymbol{s}(k) \tag{5.20}$$

The L_1-norm (linear) cost function can then be formulated as

$$j_\mathrm{lin} = \left| \boldsymbol{Q} \cdot \begin{pmatrix} \boldsymbol{i}_\mathrm{s}(k+1) \\ \boldsymbol{i}_\mathrm{s}^*(k+1) \\ \boldsymbol{v}_\mathrm{emf}(k+1) \end{pmatrix} \right| = \left| \begin{pmatrix} \boldsymbol{-1} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{i}_\mathrm{s}(k+1) \\ \boldsymbol{i}_\mathrm{s}^*(k+1) \\ \boldsymbol{v}_\mathrm{emf}(k+1) \end{pmatrix} \right|. \tag{5.21}$$

5.2.6 Binary search tree

In order to determine the polytope in which the current state $\boldsymbol{x}$ lies, in the worst case every polytope has to be tested. Unfortunately, as for real control problems several parameters are necessary and as the number of polytopes (especially for higher prediction horizons) quickly reaches more than 500, an efficient online-evaluation in real-time is also not possible. Basically, several algorithms for merging neighboring regions which contain the same control law (e.g. [53]) can be applied and are also implemented in the MPT toolbox. However, even these optimizations are often not enough for a successful real-time implementation.

One possibility that drastically reduces the necessary calculation time for finding the correct polytope in state space is described in [54]: If a binary search tree is created, the complexity for the online region search reduces to $\log_2(n)$ where n is the number of polytopes. The basic principle of a binary search tree is very simple and, for two dimensions, visualized in Figure 5.11. Assuming that the two-dimensional state space with the parameters x_1 and x_2 is divided into five regions $1 \dots 5$, in every iteration the system state $\boldsymbol{x}$ is checked if it is *above* or *below* a hyperplane. For the given example the hyperplane is a simple line. The hyperplane against which the current system state is checked is marked by a dashed line. If the system state is

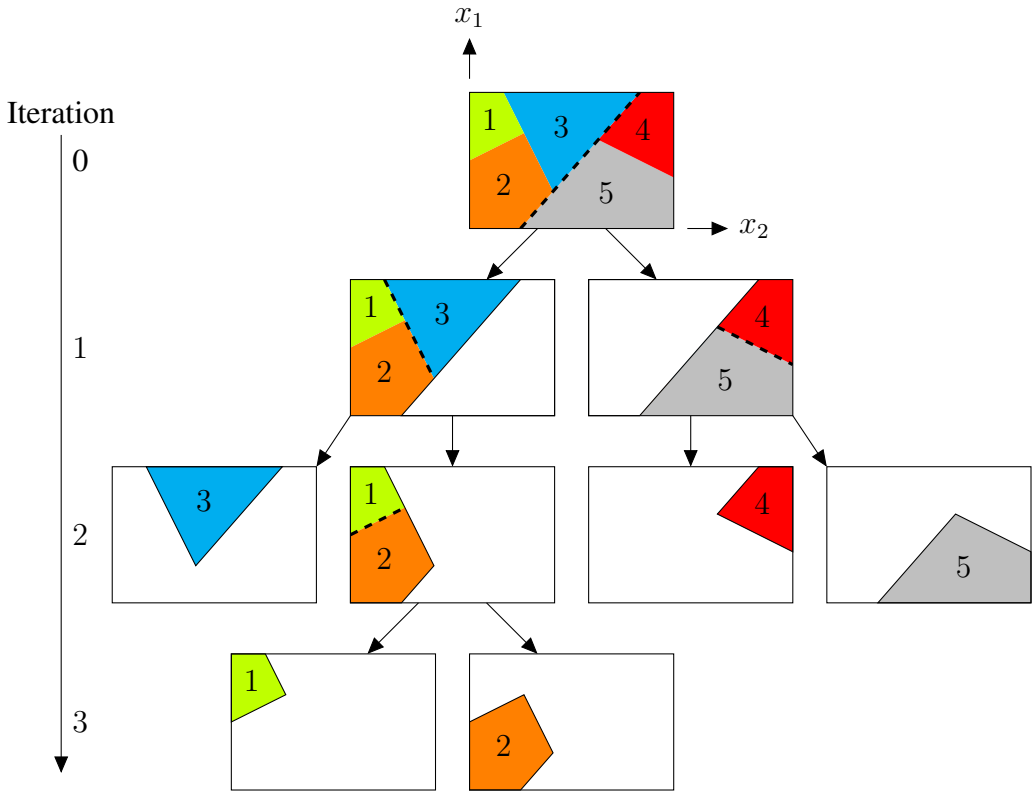


Figure 5.11: Example of a two-dimensional binary search tree

above the hyperplane, all polytopes *below* this line do not have to be searched and can be cut away. For the given example with five regions in the worst case only $\log_2(5) = 2.32$, i.e. three iterations are necessary. For real examples with more polytopes this method drastically reduces the calculation effort for the online region search.

In this way it is possible to determine the continuous-valued optimum in real-time.

5.2.7 Sector determination

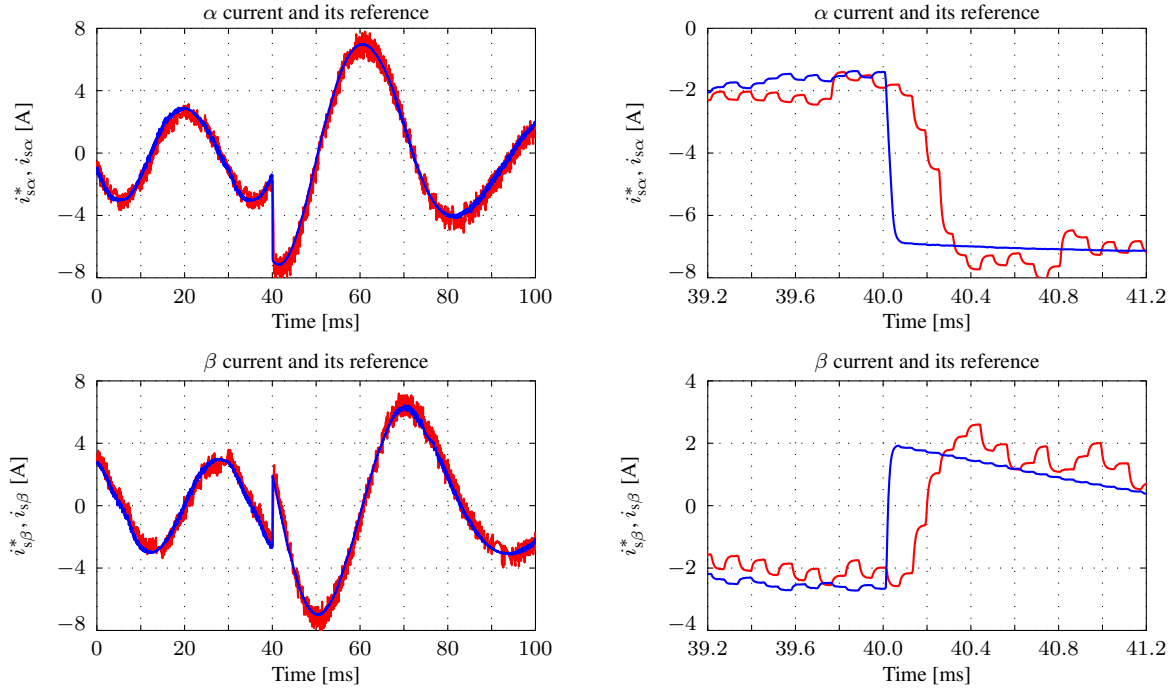
As already mentioned before, when the continuous-valued optimum has been found, the three closest points to it are used for a subsequent FS-MPC optimization. In order to find these three points, the sector (Figure 5.7) in which the continuous-valued optimum is located, has to be found. Basically, the sector can be determined in the same way as for Space Vector Modulation (SVM): For a simple two-level inverter it can be easily done by calculating the angle between the optimal voltages v_α and v_β and then the closest points can be determined via a lookup table.

For simple two-level inverters the sector determination is very easy. However, for multilevel inverters it can become a quite complex issue to find the correct sector in which the optimum continuous voltage vector lies. Basically, the same strategies to detect the sector which are used for SVM can be applied as well. By taking a closer look at the voltage vectors in Figure 5.7, it can be seen that all sectors are triangles which also holds true for multilevel inverters. Triangles are nothing else than convex polytopes. Because of this the sector determination can also be done using a binary search tree. Of course, as a two-level inverter only leads to six sectors, in

the worst case the continuous-valued solution has to be checked against three hyperplanes (in this case lines) which might not be more efficient than calculating the angle between v_α and v_β. For three- and five-level inverters, however, this possibility offers an approach which is very simple to implement and which is computationally more efficient compared to other solutions.

5.2.8 Experimental results

In order to validate the proposed algorithm, it was implemented on the two-level inverter test bench with a sampling time $T_\mathrm{s} = 61.44\,\mu\mathrm{s}$ and three prediction steps. For a full enumeration $7^3 = 343$ possible combinations of switching states have to be evaluated. It is obvious that a successful real-time implementation with the used sampling frequency of about 16 kHz is not feasible on the given computer hardware. With the proposed heuristic voltage vector preselection only $3^3 = 27$ trajectories need to be evaluated and a small overhead for the detection of the offline calculated continuous-valued optimum is necessary. Thus, three prediction steps can be realized in real-time. For finding the continuous-valued optimum an L_1-norm cost function was used since an mpLP can be solved easier than an mpQP. For the following discrete-valued optimization, an L_2-norm cost function was used.



(a) larger time scale

(b) zoom during reference steps

Figure 5.12: $\alpha\beta$ current reference steps

Figure 5.12 shows the current control result with sinusoidal current reference steps: The measured machine currents in $\alpha\beta$ coordinates are plotted together with their references. The recorded experiment was conducted with an overlayed speed PI controller. The sinusoidal current reference steps were obtained by changing the speed reference from 2000 rpm to 1000 rpm. It is obvious that the FS-MPC current controller with heuristic voltage vector preselection has

absolutely no problems to track its references. In order to take a closer look at the transients, Figure 5.12(b) shows a zoom of the reference and measured currents during the current reference steps at the same experiment. After the normal delay of two samples the currents reach values close to their references in three steps. This experiment clearly verifies that the proposed method shows excellent behavior during transients and in steady state.

Further experiments (e.g. speed reference step, load torque impact, speed reversal) are not shown in this case since the results do not really differ significantly from the results which were obtained for the classical PCC algorithm. The interested reader is referred to [55] for further experimental results.

5.2.9 Evaluation

The proposed heuristic voltage vector preselection for PCC of induction machines shows excellent behavior during transients and in steady-state. The number of trajectories which have to be evaluated could in this case be reduced to $\frac{27}{343} \approx 7.87\%$ which is a reduction of more than 92%. Thus, the proposed heuristic method offers a promising approach to reduce the calculation effort for FS-MPC methods. Furthermore, for multilevel inverters the calculation effort is *not* increased since the determination of the continuous-valued optimum is independent from that and still only three voltage vectors need to be tested per prediction step. Of course, a certain overhead for the voltage balancing algorithm is still necessary.

Despite all this, by comparing the results of PCC with three prediction steps with those where only one prediction step was implemented, no significant improvement of the control result can be seen. On the other hand, if certain switching constraints have to be applied or if higher-order systems need to be controlled, a higher prediction horizon can *significantly* improve the static and dynamic behavior. Another disadvantage of the proposed heuristic method is that it only allows the control of linear systems with linear or quadratic cost functions. Thus, one of the major advantages of FS-MPC methods is lost—the ability to easily deal with nonlinear systems and any arbitrary cost function terms.

5.3 Variable Switching Point Predictive Current Control

5.3.1 Basic idea

Similar to VSP2TC, VSP2CC implements the basic PCC algorithm with the difference that a VSP is calculated such that the root mean square (RMS) error of *both* α and β components of the stator current is minimized [56]. The principle is similar to the one described in [57, 58]. Analog to VSP2TC, the prediction equations are the same ones as for standard PCC; the only difference is that the "sampling time" for the predictions (T_s for standard PCC) is now variable (t_sw for the first interval, $T_\mathrm{s} - t_\mathrm{sw}$ for the second one), whereas t_sw is the variable switching time point.

For FS-MPC methods it is assumed that the switching states can only be changed at the beginning of the sampling intervals. In the same way as for VSP2TC, for VSP2CC the new switching state is in every sample applied at t_sw; before that time, the old switching state still remains the same. t_sw can be in the range $0 \ldots T_\mathrm{s}$.

5.3.2 Calculation of the variable switching time point

Figure 5.13 illustrates the basic principle of the VSP calculation for PCC. The squared RMS

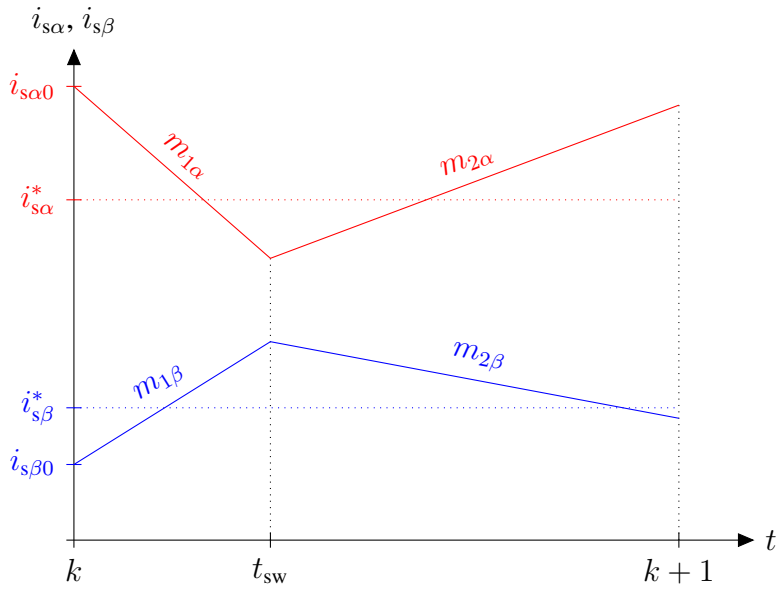


Figure 5.13: Basic principle of the VSP calculation for PCC

current error is given by

$$e_{\text{rms}^2} = \frac{1}{T_\text{s}} \left(\int_0^{t_\text{sw}} \left(\boldsymbol{i}_\text{s}^* - \boldsymbol{i}_\text{s}(t)\right)^2 \,\text{d}t + \int_{t_\text{sw}}^{T_\text{s}} \left(\boldsymbol{i}_\text{s}^* - \boldsymbol{i}_\text{s}(t)\right)^2 \,\text{d}t \right). \tag{5.22}$$

$\boldsymbol{i}_\text{s}^*$ are the current references. In order to derive a suitable equation for the calculation of t_sw, the currents are approximated by straight lines, analogously to the predicted torque for VSP2TC and it is assumed that the current slopes do not change during the sampling interval. The principle for the slope calculation of the currents is the same as for the torque slope calculation for VSP2TC. The stator currents are given by

$$\boldsymbol{i}_\text{s}(t) \approx \boldsymbol{m}_1 \cdot t + \boldsymbol{i}_{\text{s}0} \tag{5.23}$$

for the previously applied switching state which will be kept until t_sw and by

$$\boldsymbol{i}_\text{s}(t) \approx \boldsymbol{m}_2 \cdot t + \boldsymbol{i}_{\text{s},t_\text{sw}} \tag{5.24}$$

for the time between t_sw and T_s. $\boldsymbol{m}_i = \begin{pmatrix} m_{i\alpha} \\ m_{i\beta} \end{pmatrix}$, $i = 1, 2$ are the current slopes, $\boldsymbol{i}_{\text{s}0}$ the currents at the beginning of the sample and $\boldsymbol{i}_{\text{s},t_\text{sw}}$ are the currents at t_sw. As already mentioned, in order to further simplify the calculations, it is assumed that the current slopes are *constant* over one sample. Then, equation (5.22) can be written as

$$e_{\text{rms}^2} = \frac{1}{T_\text{s}} \left(\int_0^{t_\text{sw}} \left(\boldsymbol{i}_{\text{s}0} + \boldsymbol{m}_1 \cdot t - \boldsymbol{i}_\text{s}^*\right)^2 \,\text{d}t + \int_{t_\text{sw}}^{T_\text{s}} \left(\boldsymbol{i}_{\text{s},t_\text{sw}} + \boldsymbol{m}_2 \cdot t - \boldsymbol{i}_\text{s}^*\right)^2 \,\text{d}t \right). \tag{5.25}$$

In order to calculate t_{sw} such that e_{rms^2} is minimized, the derivation of the squared RMS current error is set equal to zero:

$$\frac{\mathrm{d}}{\mathrm{d}t} e_{\mathrm{rms}^2} \stackrel{!}{=} 0 \tag{5.26}$$

After some further calculations, the final equation for t_{sw} results to

$$\begin{aligned} t_{\mathrm{sw}} = \quad & \frac{(m_{2\alpha} - m_{1\alpha})(2i_{0\alpha} - 2i^*_\alpha + T_{\mathrm{s}} m_{2\alpha})}{(m_{1\alpha} - m_{2\alpha})(2m_{1\alpha} - m_{2\alpha}) + (m_{1\beta} - m_{2\beta})(2m_{1\beta} - m_{2\beta})} + \\ & + \frac{(m_{2\beta} - m_{1\beta})(2i_{0\beta} - 2i^*_\beta + T_{\mathrm{s}} m_{2\beta})}{(m_{1\alpha} - m_{2\alpha})(2m_{1\alpha} - m_{2\alpha}) + (m_{1\beta} - m_{2\beta})(2m_{1\beta} - m_{2\beta})}. \end{aligned} \tag{5.27}$$

Of course, equation (5.27) can be either a maximum or a minimum for e_{rms^2} and the resulting t_{sw} does not necessarily have to be in the range $0 \ldots T_{\mathrm{s}}$. Thus, if the resulting t_{sw} is outside its allowed range, t_{sw} is set to 0.

5.3.3 Control algorithm

Analogously to VSP2TC, VSP2CC implements the basic PCC algorithm which is described in chapter 5.1.2. The prediction equations are exactly the same ones as for the normal PCC algorithm. The only difference is that the "sampling time" for the predictions (T_{s} for standard PCC) is now variable (t_{sw} for the first interval during which the old voltage vector is applied and $T_{\mathrm{s}} - t_{\mathrm{sw}}$ for the second one in which the new voltage vector is commanded to the inverter).

After the calculation of the VSP the cost function is calculated for the current deviations at t_{sw} and T_{s}:

$$j_{\mathrm{vsp2cc}} = (\boldsymbol{i}_{\mathrm{s},t_{\mathrm{sw}}} - \boldsymbol{i}^*_{\mathrm{s}})^2 + (\boldsymbol{i}_{\mathrm{s},T_{\mathrm{s}}} - \boldsymbol{i}^*_{\mathrm{s}})^2 \tag{5.28}$$

If necessary, other cost function terms can also be added. The only difference to the PCC algorithm is that the VSP has to be considered in the time delay compensation and for the optimization, i.e. two predictions, one from 0 to t_{sw}, the other one from t_{sw} to T_{s}, are necessary.

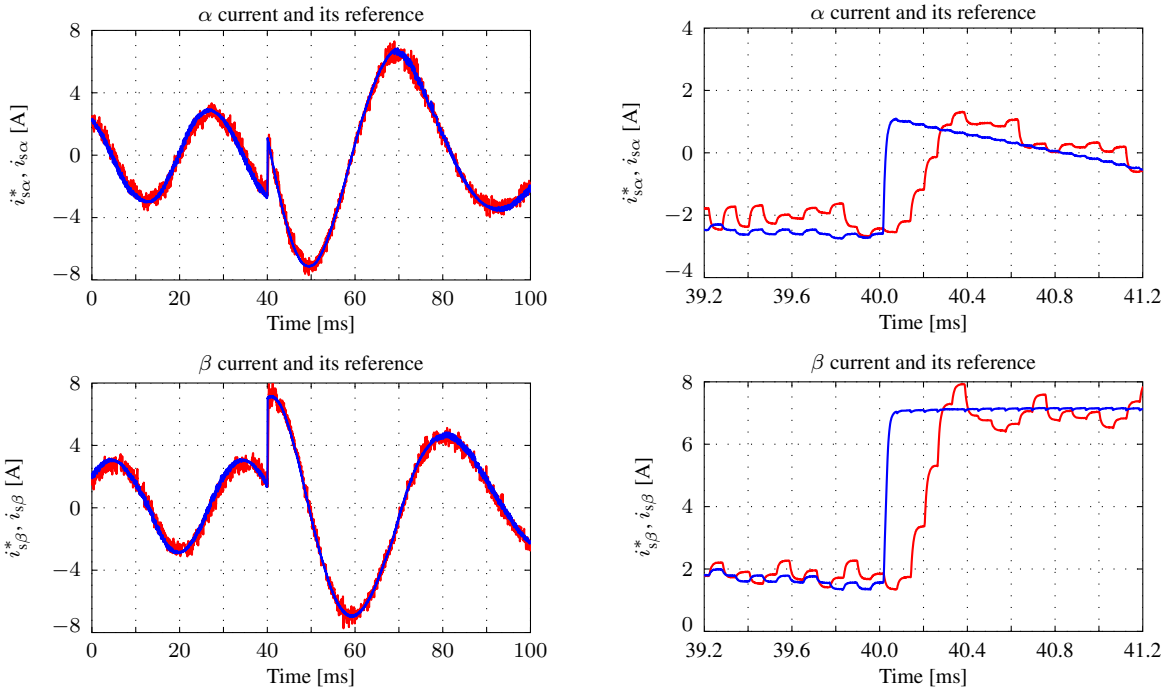
5.3.4 Experimental results

In order to verify the VSP2CC algorithm, it was implemented on the two-level inverter test bench with the sampling time $T_{\mathrm{s}} = 61.44\,\mu\mathrm{s}$.

The Figures 5.14(a) and 5.14(b) show the transient behavior of the VSP2CC algorithm. A comparison to the results of the conventional PCC algorithm shows that VSP2CC can also track the current references very well and that the VSP strategy has no negative effect on the transient behavior.

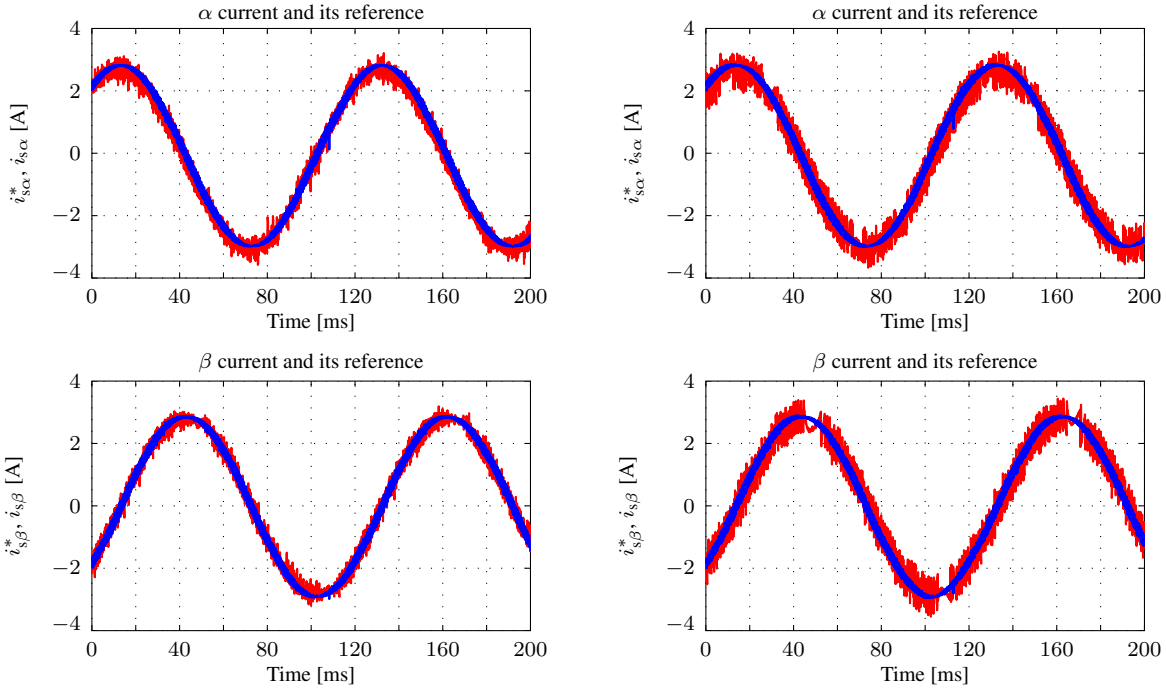
Figure 5.15(a) shows the steady-state machine currents at $500\,\mathrm{rpm}$ produced by the VSP2CC algorithm, in Figure 5.15(b) the currents produced by conventional PCC can be seen. It is clearly visible that the VSP2CC algorithm effectively reduces current ripples. As already mentioned, this comes with the cost of an increased switching frequency. For the presented operating point at $500\,\mathrm{rpm}$ the average switching frequency per IGBT is nearly doubled—however, compared to the theoretically maximum switching frequency of $8\,\mathrm{kHz}$ this value is still very low.

In practical experiments it was observed that the effect of VSP2CC on current ripples is strongly dependent on the operating point. For very low machine speeds a huge improvement



(a) larger time scale (b) zoom during reference steps

Figure 5.14: $\alpha\beta$ current reference steps



(a) VSP2CC, $f_{sw} \approx 2.7\,\text{kHz}$ (b) PCC, $f_{sw} \approx 1.5\,\text{kHz}$

Figure 5.15: $\alpha\beta$ steady-state currents at $500\,\text{rpm}$

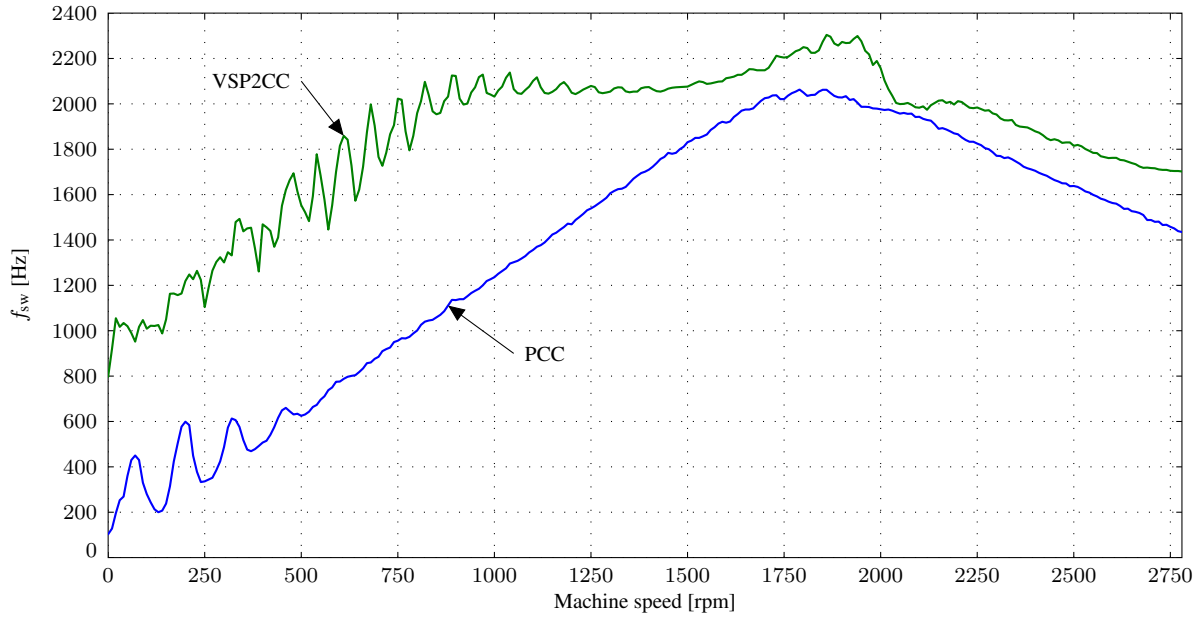


Figure 5.16: Average switching frequency of VSP2CC and PCC ($T_s = 100\,\mu s$)

of the current quality could be observed, whereas it was much smaller at nominal machine speed. In order to investigate this phenomenon more deeply, the average IGBT switching frequency was measured in steps of 10 rpm from zero to full nominal speed (2772 rpm). The speed value was increased every 10 s and the average switching frequency was recorded for both methods (PCC and VSP2CC) with a sampling frequency of 10 kHz. The result of this comparison can be seen in Figure 5.16. This experiment clearly verifies that the VSP2CC strategy has *significant* advantages over PCC at low speeds (0 rpm to about 1000 rpm). For zero speed the average switching frequency goes down to about 100 Hz for PCC whereas it is always higher than 800 Hz for VSP2CC.

5.3.5 Evaluation

The shown experimental results clearly verify the ability of the VSP2CC algorithm to reduce current ripples without decreasing the dynamic performance of the PCC algorithm. Furthermore, it needs no additional tuning parameters (no weighting factor) which means that no tuning at all is necessary. In the same way as for PCC, the algorithm just needs to be implemented and as long as the machine parameters are known it will deliver good control results without further tuning.

On the other hand—similar to VSP2TC—VSP2CC comes with a calculation effort which is more than twice as high as for the normal PCC algorithm: First, for all possible voltage vectors the current slopes have the be calculated which is an extra set of predictions. Then, for every voltage vector the VSP has to be calculated. And, finally, two sets of predictions (from 0 to t_{sw} and from t_{sw} to T_s) are necessary. Thus, in the whole the calculation effort for VSP2CC is more than *three* times higher compared to PCC. It is obvious—especially for multilevel inverters—that strategies to reduce the number of calculations for VSP2CC would be a great benefit.

5.4 VSP2CC with heuristic voltage vector preselection

Since VSP2CC needs much more calculations than PCC, a way to reduce the necessary calculation effort is needed. Especially if VSP2CC should be applied to multilevel inverters, a real-time implementation of the control algorithm can become feasible for higher sampling frequencies when only a subset of all possible voltage vectors or switching states has to be evaluated. So, the main idea is to combine the heuristic voltage vector preselection which is described in chapter 5.2 with VSP2CC and to use only the reduced subset of *three* instead of seven voltage vectors as candidates for the VSP calculation and for the following FS-MPC optimization.

5.4.1 Control algorithm

The control algorithm is exactly the same one as for VSP2CC with a complete enumeration of all possible switching states. In a previous step the offline calculated continuous-valued solution is determined in the same way as for PCC with heuristic voltage vector preselection. In this case, however, a prediction horizon of only one step is used since VSP2CC is also implemented for only one predicted sample.

Again, the continuous-valued solution is determined for the sampling time T_s and for VSP2CC the "new" voltage vector is only applied for the time $T_\mathrm{s} - t_\mathrm{sw}$.

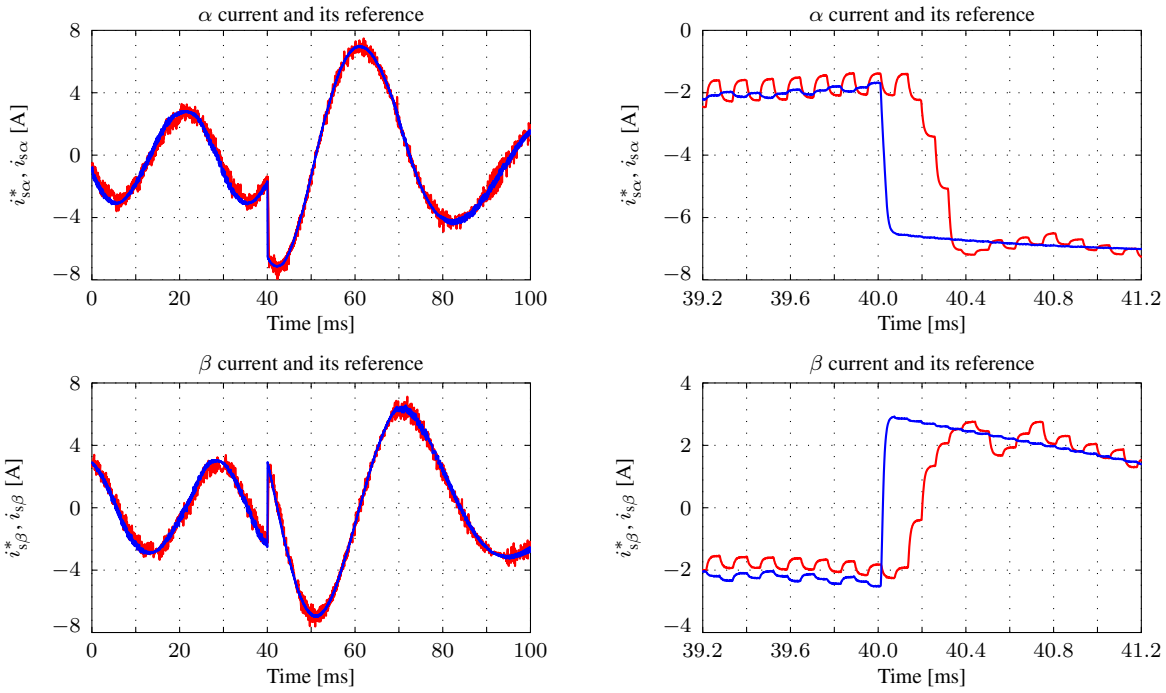
5.4.2 Experimental results

In order to verify the proposed VSP2CC algorithm with heuristic voltage vector preselection, several experiments were conducted on the two-level inverter test bench. During all experiments no deterioration of the control result could be observed compared to VSP2CC with a complete enumeration of all possible voltage vectors. For a sake of brevity only current control results are shown in the following. The experiment was conducted with the sampling time $T_\mathrm{s} = 61.44\,\mu\mathrm{s}$.

Figure 5.17 shows the control result for both $\alpha\beta$ currents with sinusoidal reference steps. It is clearly visible that the controller has no problems to track its references. Furthermore, compared to Figure 5.14, no deterioration of the control result because of the heuristically reduced number of voltage vectors can be observed. Figure 5.17(b) shows a zoom of the recorded currents and their references during the step change of the current references which were produced by changing the speed reference of the overlaying speed PI controller from $2000\,\mathrm{rpm}$ to $1000\,\mathrm{rpm}$. Once again, if the dynamic behavior is compared to that of VSP2CC with full enumeration (Figure 5.14(b)), no deterioration of the control result and of the dynamic behavior can be seen.

5.4.3 Evaluation

The shown experimental results clearly prove that a successful combination of VSP2CC with a heuristic voltage vector preselection is possible. The proposed method significantly reduces the calculation effort without deteriorating the control result. Thus, the first two goals of this work (reduction of the calculation effort for FS-MPC and increasing the time resolution) are successfully combined. The described method is very promising especially for multilevel inverters (which will be examined later on) since even in that case only three voltage vectors need



(a) larger time scale (b) zoom during reference steps

Figure 5.17: $\alpha\beta$ current reference steps

to be evaluated. A certain additional overhead for the voltage balancing is necessary although—compared to a full enumeration of all voltage vectors—a significant reduction of the calculation effort can be expected.

5.5 PCC for three-level NPC inverters

The four PCC algorithms which were described previously (PCC, PCC with heuristic voltage vector preselection, VSP2CC and VSP2CC with heuristic voltage vector preselection) can be easily extended to three-level NPC inverters. Of course, analogously to PTC, in this case more voltage vectors and switching states have to be evaluated. Furthermore, an algorithm for the DC link capacitor voltage balancing has to be implemented.

5.5.1 DC link capacitor voltage balancing

The DC link capacitor voltage balancing extension for FS-MPC of NPC inverters is described in chapter 4.3.1. The resulting equation (4.17) can also be used for PCC applied to three-level NPC inverters.

5.5.2 Control algorithms

Analogously to PTC for three-level NPC inverters, the prediction equations for the machine model, the rotor flux and back-EMF estimation are the same as for the two-level inverter. For

the optimization, however, the voltage balancing has to be included in the cost function and the cost function has to be evaluated for 27 different voltage vectors.

5.5.2.1 Basic PCC algorithm for three-level NPC inverters

As already mentioned, for PCC of three-level NPC inverters the capacitor voltage unbalance has to be considered in the cost function. Thus, the cost function for PCC of three-level NPC inverters is given by

$$j_{\text{pcc, lin, npc}} = j_{\text{pcc, lin}} + w_{\text{lin, npc}} \cdot |\Delta v_{\text{c}}(k)| \tag{5.29}$$

in case of an L_1-norm and by

$$j_{\text{pcc, quad, npc}} = j_{\text{pcc, quad}} + w_{\text{quad, npc}} \cdot (\Delta v_{\text{c}}(k))^2 \tag{5.30}$$

for an L_2-norm. $w_{\text{lin, npc}}$ and $w_{\text{quad, npc}}$ are the weighting factors for the voltage balancing.

5.5.2.2 Heuristic voltage vector preselection

For the case of a three-level NPC inverter the heuristic voltage vector preselection follows the same principles as for two-level inverters. Since the DC link voltage is the same, the range of the applied voltages in abc and thus also in $\alpha\beta$ coordinates is the same as for two-level inverters. This means that the complexity of the offline calculated continuous-valued solution is *not* increased compared to a simple two-level inverter.

For the following enumeration of all candidate sequences, however, the $\alpha\beta$ voltage plane can be divided into 24 different sectors (triangles) instead of only 6 ones (Figure 5.18). As already mentioned, the sector determination for three-level inverters can be done in the same way as for SVM of three-level inverters. However, a binary search tree to determine the sector in which the continuous-valued optimum voltage vector lies can be used as well.

Since the DC link capacitor voltages have to be balanced and since the heuristic preselection reduces the number of candidate voltage vectors per prediction step from 19 to only 3, it needs to be investigated if the voltage balancing can still be done with the reduced input set.

Since an induction machine is a balanced system, equation (2.12) is valid. Furthermore, only a zero switching state in a phase changes the DC link capacitor voltages. Therefore, all three zero switching states (ppp, 000 and nnn) have *no* influence on the voltage balancing. If the zero voltage vector has to be applied to the system, the choice about the switching state can be based on the previously applied one, similar to a two-level inverter.

By taking a closer look at the voltage vectors on the inner hexagon, it can be seen that every voltage vector can be produced by two different switching states. For everyone of these six voltage vectors it can also be stated that the voltage balancing can be done by choosing the appropriate switching state: One of the two choices applies one zero switching state in the three phases, the other choice applies two zero switching states but in the *other* two phases. Thus, as long as equation (2.12) is valid, the voltage balancing can be ensured by choosing an appropriate switching state. If the outer hexagon is considered, it can be seen that all switching states which can also be produced by a two-level inverter (pnn, ppn, npn, npp, nnp and pnp) have no influence on the voltage balancing since no zero switching states are used in any phase. However, by taking a look at the intermediate points (p0n, 0pn, np0, n0p, 0np and pn0), it can be seen that these voltage vectors have *no* redundancy, i.e. if such a voltage vector is commanded,

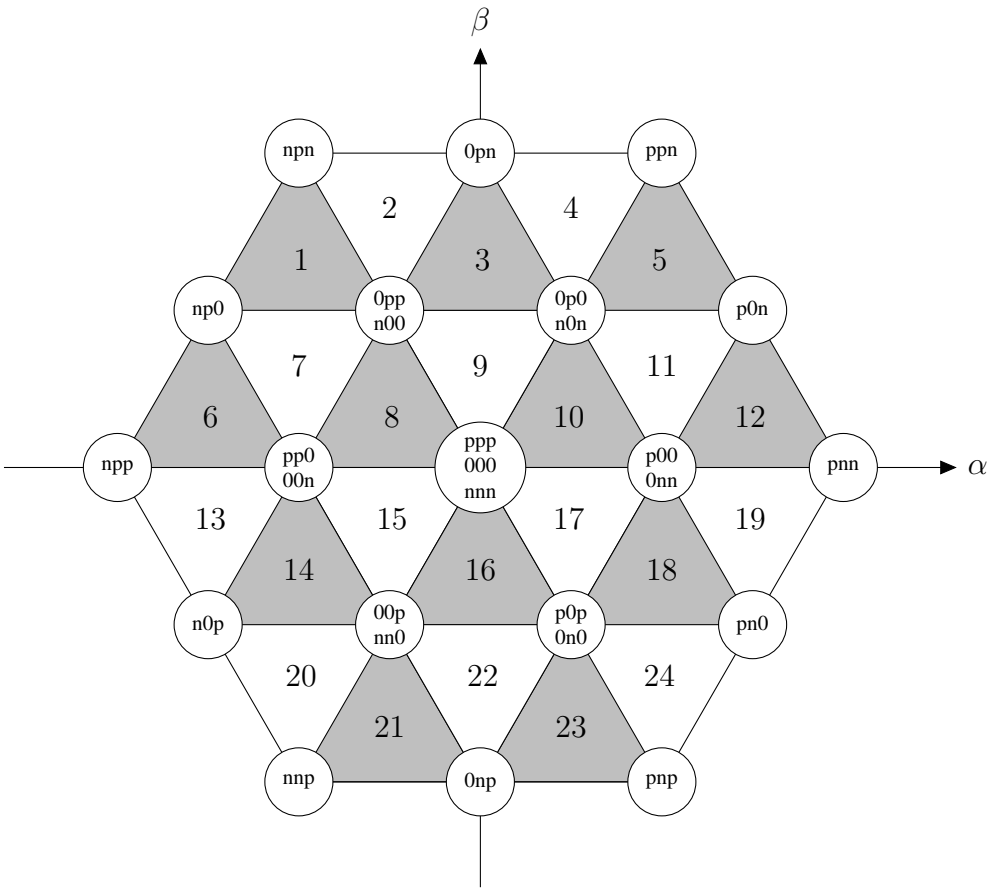


Figure 5.18: Voltage vectors of a three-level inverter and division into sectors

no influence on the voltage balance can be taken. This, however, is also the case for a full enumeration of *all* possible switching states. If the heuristic voltage vector preselection is taken into consideration, it can be seen that all these intermediate points are in the corners of sectors which also have *at least* one voltage vector which belongs to the inner hexagon, i.e. a voltage vector which allows balancing of the DC link capacitor voltages. Thus, even for the heuristically reduced set of candidate voltage vectors in *every* case a voltage vector can be found which ensures balanced DC link voltages. Because of this the heuristic preselection can also be applied to three-level NPC inverters.

5.5.2.3 VSP2CC for three-level NPC inverters

Similar to VSP2TC, the VSP2CC algorithm can also be extended to three-level NPC inverters [56]. Again, the DC link capacitor voltage balancing has to be included in the cost function.

Since for a three-level inverter already 19 different voltage vectors have to be evaluated and since the calculation effort for VSP2CC is significantly increased compared to PCC, this high number of voltage vectors limits the applicability of VSP2CC for three-level inverters: Because of the high number of calculations the sampling rate has to be decreased by a factor of two or even more in order to execute the algorithm in real-time.

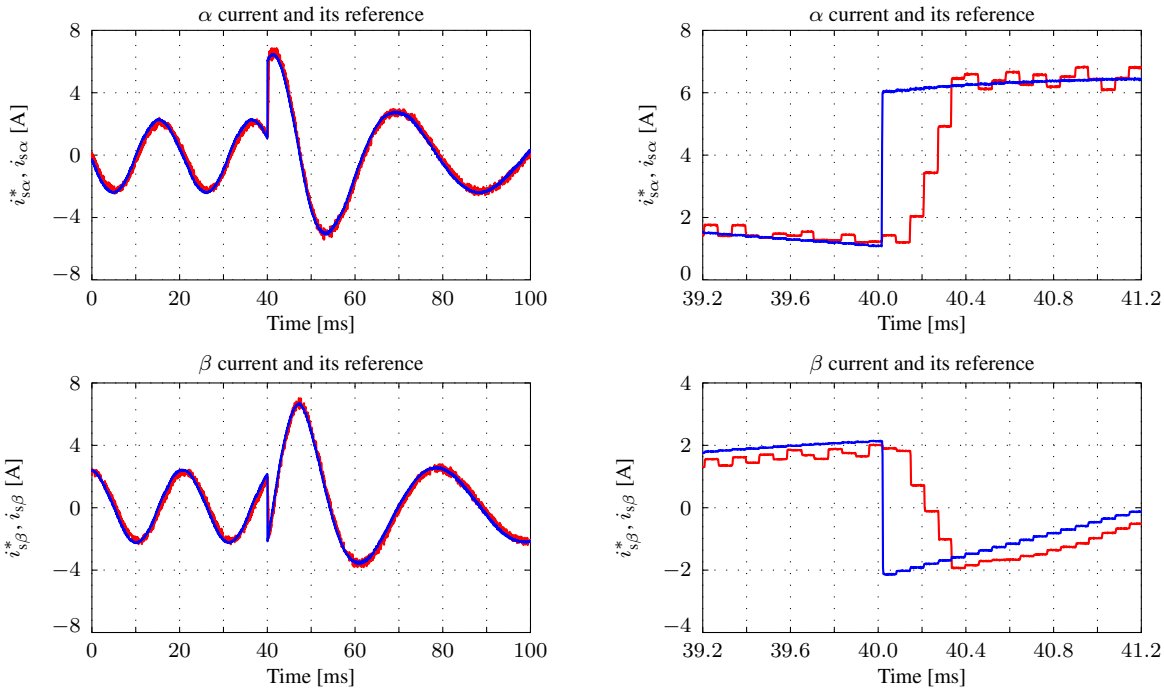
5.5.2.4 VSP2CC with heuristic voltage vector preselection

In order to overcome the previously mentioned problems regarding a real-time implementation of VSP2CC for three-level inverters, the VSP2CC algorithm for three-level NPC inverters can also be combined with the heuristic voltage vector preselection. In this case only three different voltage vectors have to be evaluated and the calculation effort is significantly reduced compared to VSP2CC with a full enumeration of all possibilities. Then, a much higher sampling frequency can be used which means that a better control result can be expected compared to VSP2CC without preselection (assuming that it is executed on the same hardware).

5.5.3 Experimental results

5.5.3.1 PCC

In order to verify the applicability of the PCC algorithm for three-level NPC inverters, several experiments were conducted on the three-level inverter test bench with a sampling frequency of $16\,\mathrm{kHz}$ which corresponds to $T_\mathrm{s} = 62.5\,\mu\mathrm{s}$. In this case a quadratic cost function was used and $w_{\text{quad, npc}}$ was set to 0.01.



(a) larger time scale

(b) zoom during reference steps

Figure 5.19: $\alpha\beta$ current reference steps

Figure 5.19 shows the results for sinusoidal current reference steps which were obtained with an overlaying speed PI controller: In the shown case the speed reference was changed from $2830\,\mathrm{rpm}$ to $1415\,\mathrm{rpm}$. Figure 5.19(b) shows a zoom of these currents during the reference step change. Both graphs clearly show that the proposed algorithm can track its current references without any problems in steady state as well as during transients. If these experimental results

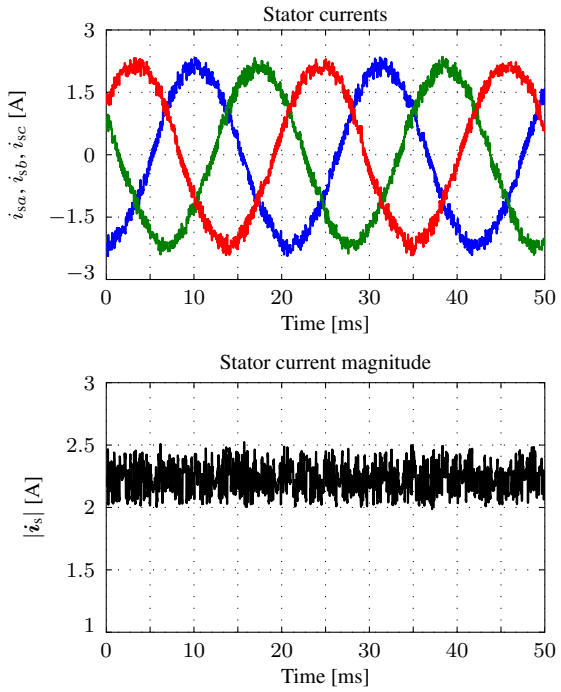


Figure 5.20: Steady-state phase currents at 2830 rpm and no load

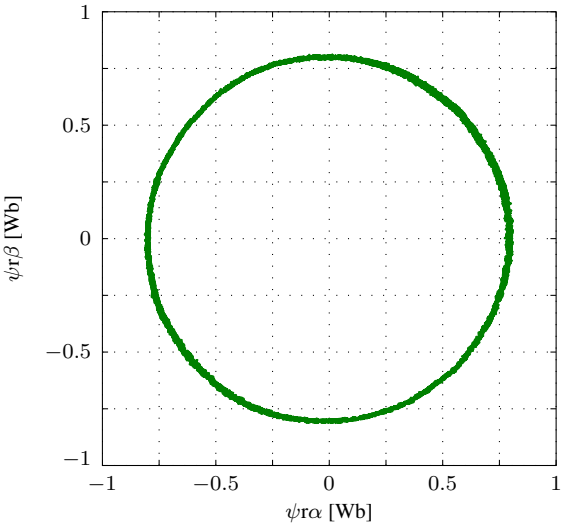


Figure 5.21: Steady-state rotor flux in $\alpha\beta$ coordinates at 2830 rpm and no load

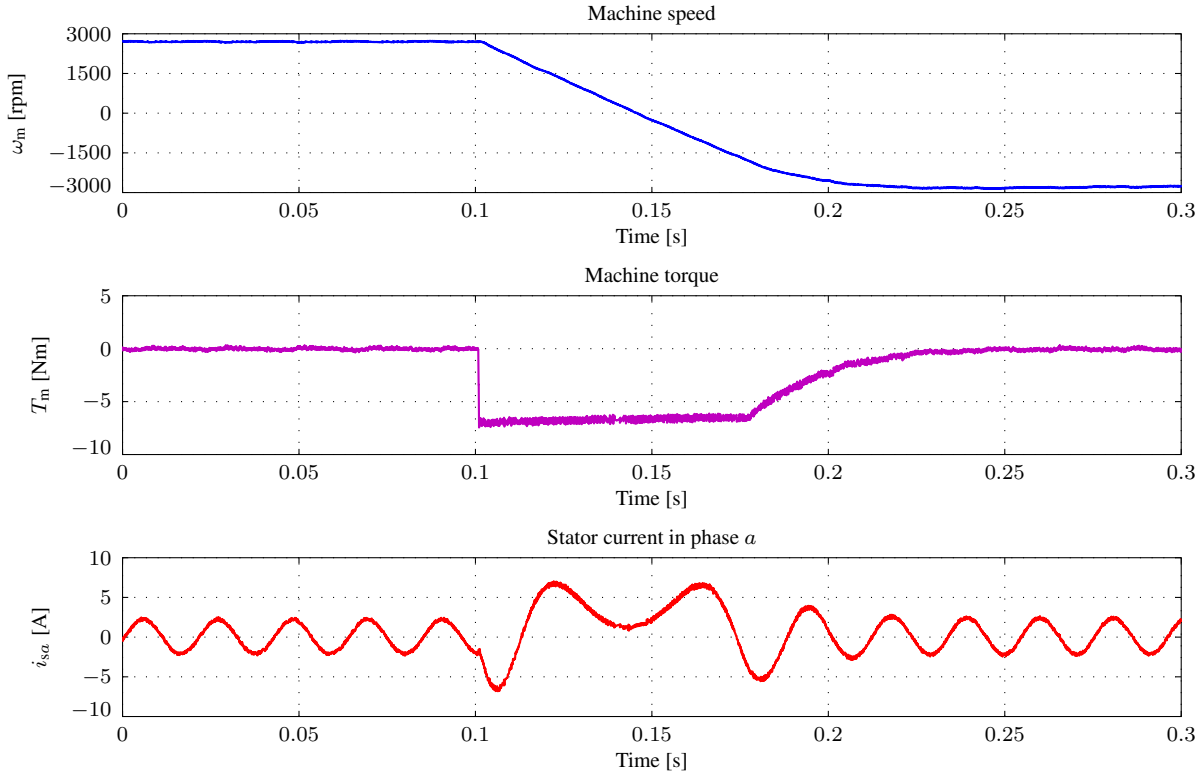


Figure 5.22: Speed reversal from positive nominal to negative nominal speed

are compared to the results obtained with the two-level inverter (Figure 5.2), it is clearly visible that the current ripples are significantly reduced. However, these results were obtained on different test benches and the test bench for the three-level inverter has much better and more accurate AD converters and current sensors which has considerable impact on the control result.

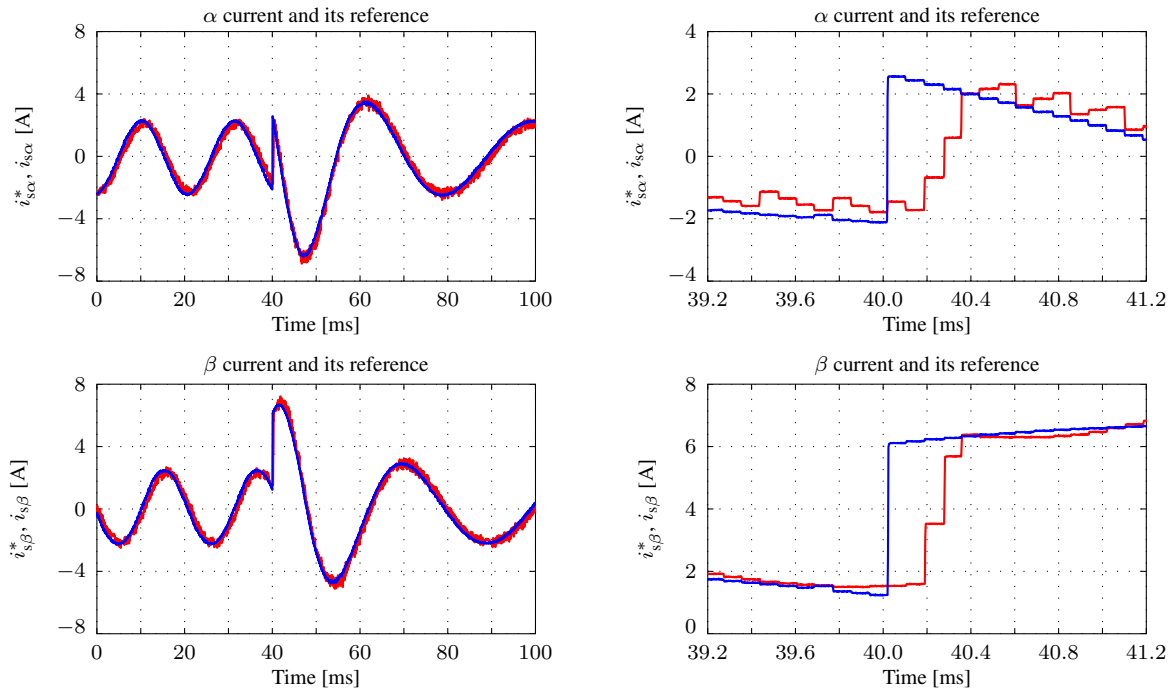
The steady-state phase currents and their magnitude at 2830 rpm can be seen in Figure 5.20. In this case the use of a three-level inverter also shows a significant improvement in terms of current ripples compared to the results which were obtained on a two-level inverter. The average switching frequency per IGBT was about 1.65 kHz.

Figure 5.21 shows the rotor flux in $\alpha\beta$ coordinates in steady-state at 2830 rpm. As expected, both α and β components of the rotor flux form a circle.

Finally, in order to prove that the proposed algorithm works in the whole speed range, a speed reversal from positive nominal to negative nominal speed was conducted which can be seen in Figure 5.22. Similar to the results obtained on the two-level inverter, the controller has no problems to reverse the machine speed. Of course, the ripples on the machine torque and especially on the currents are significantly lower compared to the results from the two-level inverter.

5.5.3.2 PCC with heuristic voltage vector preselection

The PCC algorithm with heuristic voltage vector preselection was also experimentally verified. The results were obtained at a sampling frequency of 12 kHz which corresponds to $T_\mathrm{s} = 83.33\,\mu\mathrm{s}$ and with two prediction steps. For a full enumeration of all voltage vectors



(a) larger time scale

(b) zoom during reference steps

Figure 5.23: $\alpha\beta$ current reference steps

$19^2 = 361$ possible trajectories would have to be evaluated but if the heuristic preselection is used, only $3^2 = 9$ different combinations of voltage vectors have to be considered. Thus, the number of evaluated trajectories can be reduced to only $\frac{9}{361} \approx 2.5\%$ which is a tremendous reduction of the necessary calculation effort.

Since the obtained experimental results for a speed reversal, steady-state currents etc. do not differ significantly from the results obtained for PCC with full enumeration of all switching states and with only one prediction step, for a sake of brevity only the results for sinusoidal current reference steps are shown.

Figure 5.23 shows the recorded $\alpha\beta$ currents and their references. The sinusoidal reference steps were obtained with an overlaying speed PI controller whose reference was changed from $2830\,\text{rpm}$ to $1415\,\text{rpm}$. It is clearly visible that the FS-MPC controller with heuristic preselection has absolutely no problems to track its references. As it can also be seen in Figure 5.23(b) which shows a zoom of the same experiment during the step change of the references, the dynamic behavior is not deteriorated by the use of the heuristic method. Within three steps (after the delay of two samples due to the calculation delay and the fact that the control error is minimized at the end of a sample) the measured currents have reached their references again. If these results are compared to the ones for PCC with a prediction horizon of only one sample (Figure 5.19), a higher current ripple can be noticed. This higher ripple is due to the fact that for one prediction step a higher sampling frequency ($16\,\text{kHz}$) could be used.

5.5.3.3 VSP2CC

The VSP2CC algorithm for three-level NPC inverters was tested with a sampling frequency of $10\,\text{kHz}$ and the results were compared to PCC which was executed with the same sampling time $T_\text{s} = 100\,\mu\text{s}$.

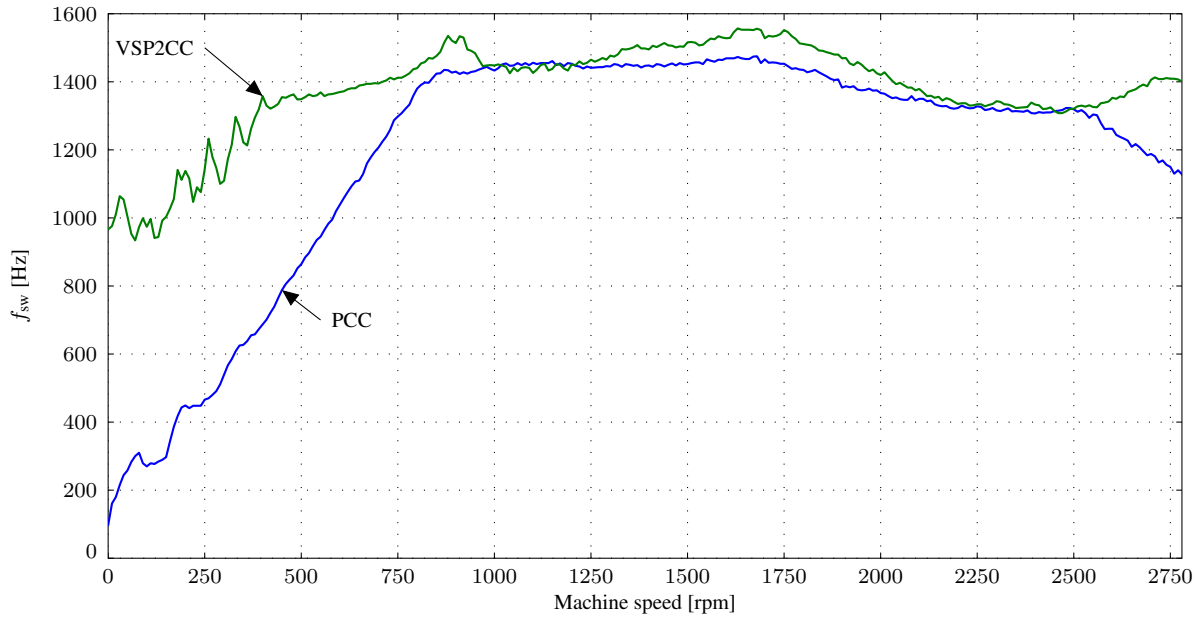
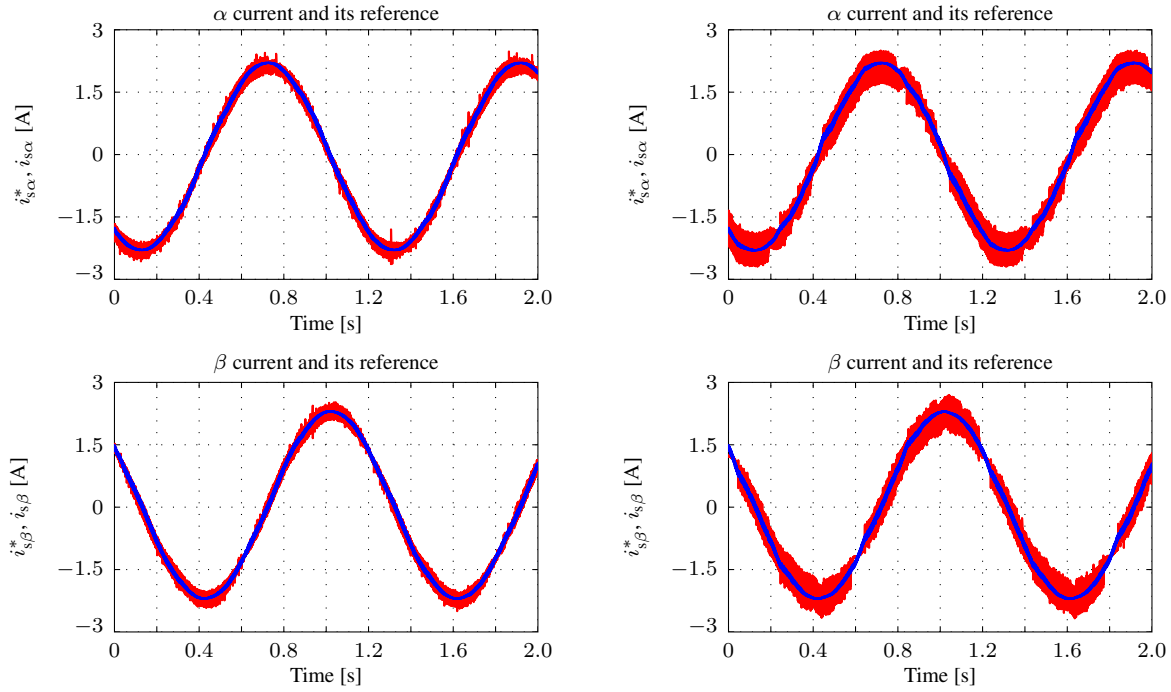
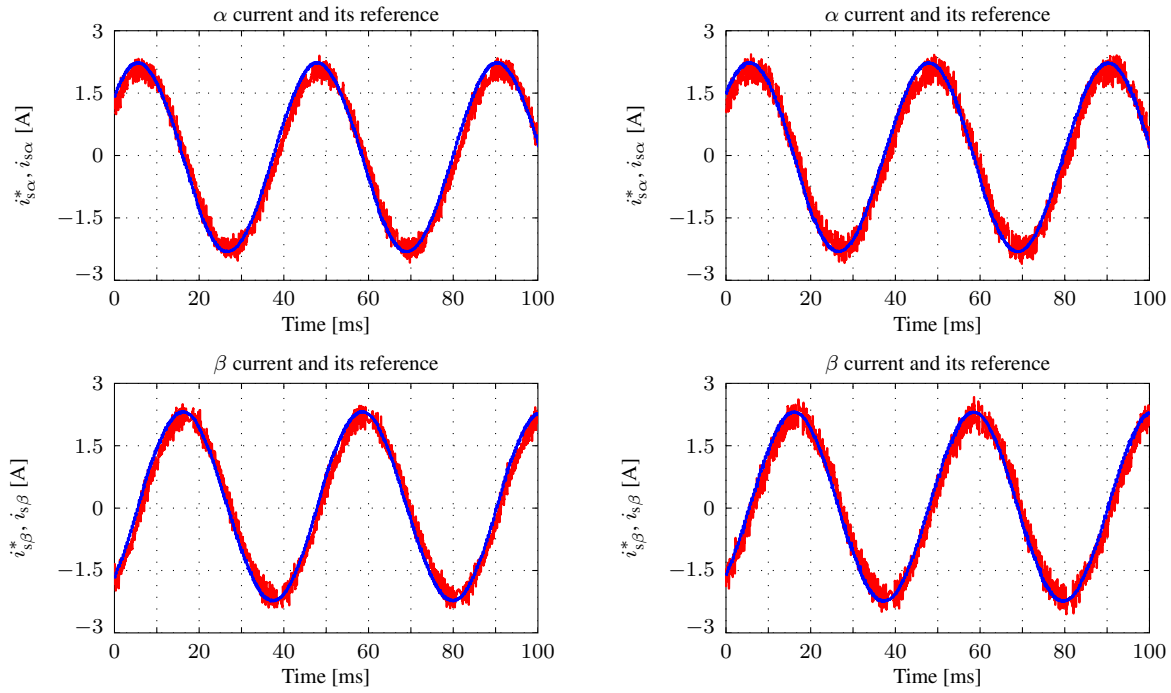


Figure 5.24: Average switching frequency of VSP2CC and PCC ($T_\text{s} = 100\,\mu\text{s}$)



(a) VSP2CC, $f_{sw} \approx 1.0\,\text{kHz}$ (b) PCC, $f_{sw} \approx 0.25\,\text{kHz}$

Figure 5.25: $\alpha\beta$ steady-state currents at $50\,\text{rpm}$



(a) VSP2CC, $f_{sw} \approx 1.5\,\text{kHz}$ (b) PCC, $f_{sw} \approx 1.45\,\text{kHz}$

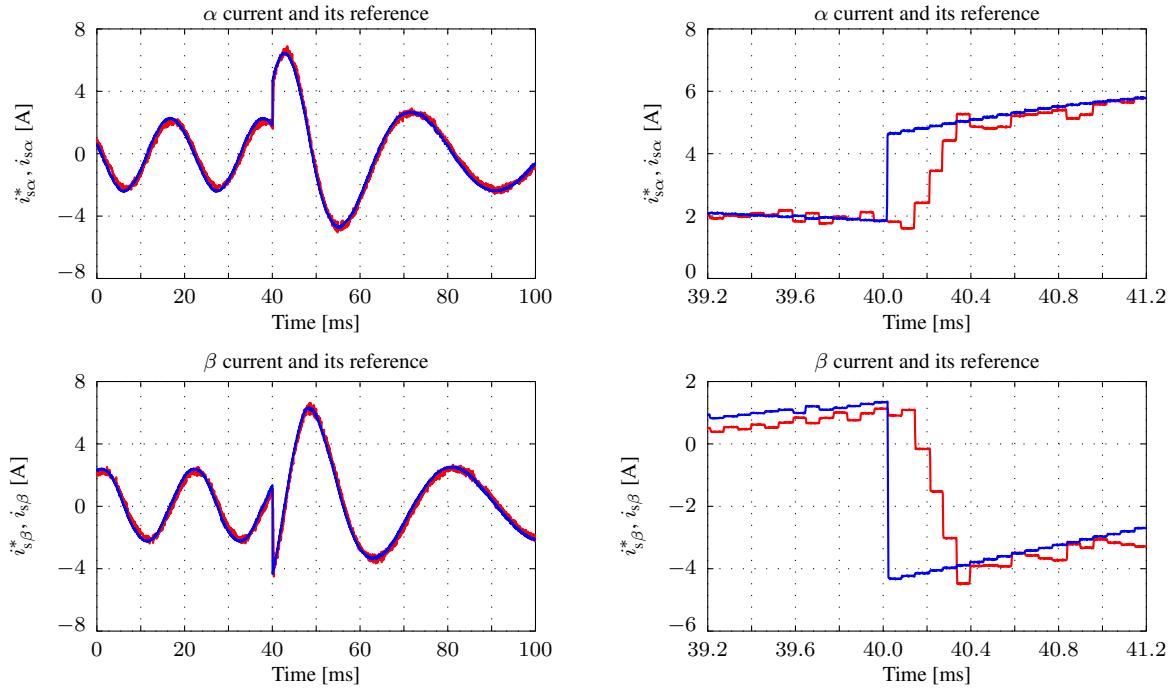
Figure 5.26: $\alpha\beta$ steady-state currents at $1415\,\text{rpm}$

In order to investigate the influence of the machine speed, the average IGBT switching frequency was measured in steps of 10 rpm from zero to full nominal speed (2830 rpm). The speed value was increased every 10 s and the average switching frequency was recorded for both methods (PCC and VSP2CC) with a sampling frequency of 10 kHz. The result of this comparison can be seen in Figure 5.24. This experiment clearly verifies that the VSP2CC strategy has *significant* advantages over PCC at low speeds (0 rpm to about 800 rpm). For zero speed the average switching frequency goes down to about 150 Hz for PCC whereas it is always higher than 950 Hz if VSP2CC is used.

The Figures 5.25(a) and 5.25(b) show the steady-state $\alpha\beta$ currents at 50 rpm. It is clearly visible that the VSP2CC strategy leads to a significant reduction of current ripples at lower speeds. The Figures 5.26(a) and 5.26(b) show the stator currents at half nominal speed (1415 rpm). Although the VSP2CC strategy still leads to a slight improvement of the currents, it is definitely less than that which was obtained at 50 rpm. These two experiments in steady-state prove that the VSP2CC strategy does not lead to a huge improvement for medium and higher speeds.

5.5.3.4 VSP2CC with heuristic voltage vector preselection

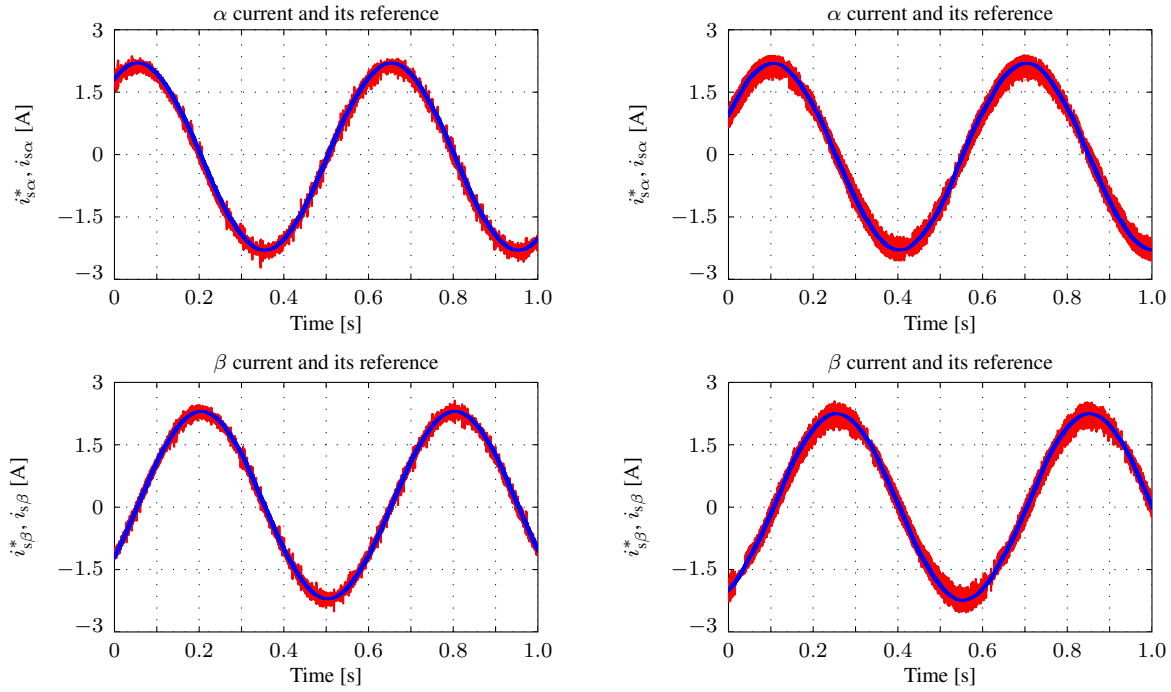
The VSP2CC algorithm with heuristic voltage vector preselection for three-level NPC inverters was tested with a sampling frequency of 16 kHz and the results were compared to PCC which was executed with the same sampling time $T_\mathrm{s} = 62.5\,\mu\mathrm{s}$.



(a) larger time scale (b) zoom during reference steps

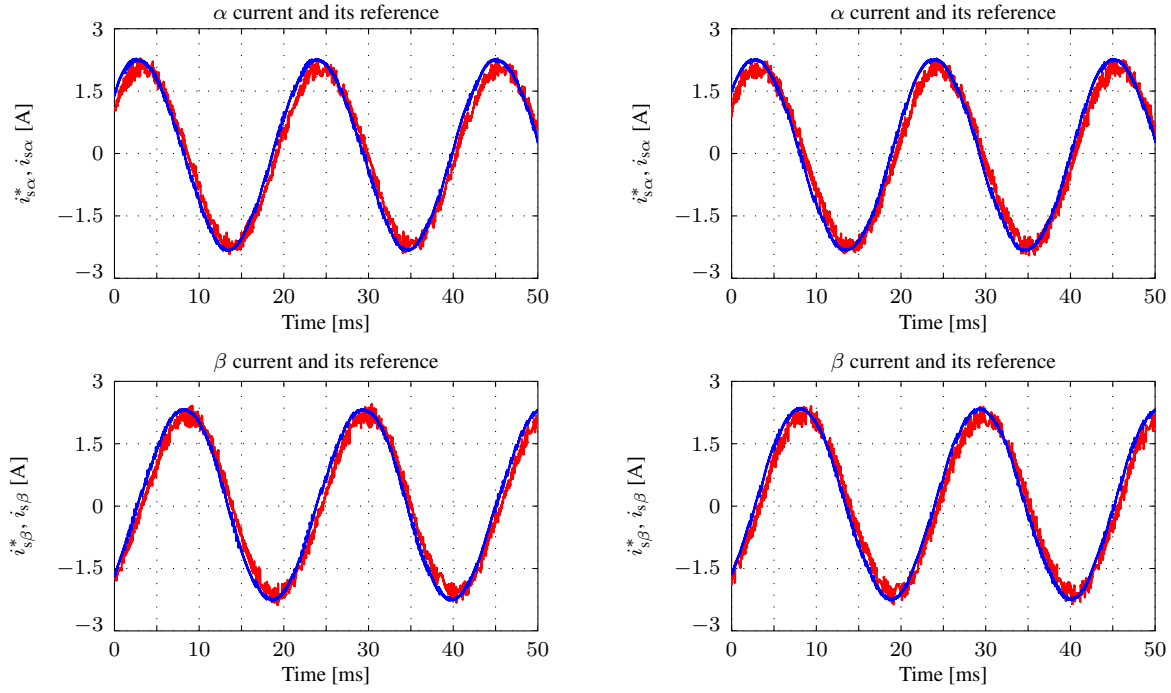
Figure 5.27: $\alpha\beta$ current reference steps

The first experiment was conducted in order to verify that the heuristic preselection for VSP2CC does not deteriorate the dynamic performance. Thus, current reference steps were



(a) VSP2CC, $f_{sw} \approx 1.9\,\text{kHz}$ (b) PCC, $f_{sw} \approx 0.85\,\text{kHz}$

Figure 5.28: $\alpha\beta$ steady-state currents at 100 rpm



(a) VSP2CC, $f_{sw} \approx 2.1\,\text{kHz}$ (b) PCC, $f_{sw} \approx 1.65\,\text{kHz}$

Figure 5.29: $\alpha\beta$ steady-state currents at 2830 rpm

commanded. As in the previous experiments, these were produced by changing the speed reference from 2830 rpm to 1415 rpm. The result can be seen in Figure 5.27. It is clearly visible that the current ripples are very small and that the strategy is able to accurately track the references. In order to take a closer look at the currents and their references during this reference step, Figure 5.27(b) shows a zoom of the same experiment during the reference step. After the normal delay of two samples the currents reach values around their references within three to four sampling cycles.

Another conducted experiment shows the steady-state currents at 100 rpm in the Figures 5.28(a) and 5.28(b). As for the previous experiments, the quality of the controlled currents is again significantly improved; however, due to the higher machine speed (100 rpm instead of 50 rpm) and the fact that a higher sampling rate (16 kHz instead of 10 kHz) was used, the effect of the VSP strategy cannot be expected to be in the same range. The Figures 5.29(a) and 5.29(b) show the currents at full nominal speed (2830 rpm). For this operating point the improvement of the current quality is very small.

5.5.4 Evaluation

The shown experimental results clearly verify that a successful implementation of FS-MPC methods for three-level NPC inverters is possible. Of course, due to the high number of possible voltage vectors and switching states, the calculation effort is significantly increased compared to two-level inverters. However, as expected, the experimental results show less ripples than the ones which were obtained on the two-level inverter test bench. If the number of calculations is reduced, especially with the proposed heuristic voltage vector preselection strategy, the ripples could be even further reduced as with a higher sampling rate.

Finally, it should be noted that all advantages that the PCC strategy offers compared to PTC also hold true for three-level NPC inverters.

5.6 PCC for three-level FC inverters

Analogously to PTC, the PCC algorithm can also be applied to three-level FC inverters driving an IM. The strategy to decouple the selection of the optimum voltage vector from the subsequent selection of the best switching state (chapter 4.5.1) can also be applied to PCC. As for PTC, in this way a reduction of the calculation effort can be achieved and the necessary calculation effort is comparable to that for FS-MPC of NPC inverters.

5.6.1 Control algorithms

The PCC control algorithms which were described in this chapter (basic PCC, PCC with heuristic voltage vector preselection, VSP2CC and VSP2CC with heuristic voltage vector preselection) can be easily adjusted to a three-level FC inverter driving an IM. Since all necessary parts regarding the PCC algorithm and its derivatives (voltage vector selection) have already been described before, the following descriptions of the algorithms are kept short.

5.6.1.1 Basic PCC algorithm for three-level FC inverters

The voltage vector selection for the basic PCC algorithm can be done in the same way as for a simple two-level inverter. Of course, instead of seven voltage vectors in this case the optimum out of 19 different possibilities has to be found. The prediction equations are described in chapter 5.1; the cost function can be calculated with equation (5.10) (L_1-norm) or with equation (5.11) if a quadratic norm should be used.

The following switching state selection is described in chapter 4.5.2. In this case the same cost function can be used as for PTC.

5.6.1.2 Heuristic voltage vector preselection

If PCC with heuristic voltage vector preselection should be applied to three-level FC inverters, the decoupling of the voltage vector from the switching state selection can be exploited such that an even greater reduction of the calculation effort is possible: The heuristic preselection of the optimum voltage vector can be applied to the voltage vector selection without any problems, even for higher prediction horizons. Then, for the following selection of the optimum switching state a prediction horizon of only *one* sample can be used, although a longer horizon is used to determine the best voltage vector. With this strategy only three voltage vectors have to be tested for every prediction step. Furthermore, despite the higher prediction horizon, no additional calculation effort is necessary for the voltage balancing compared to PCC with a horizon of only one sample. The basic principle of this algorithm is, for a simplified control task and a different voltage balancing algorithm, described in [46].

5.6.1.3 VSP2CC

Even VSP2CC can be applied to three-level FC inverters driving an IM: The algorithm to find the optimum voltage vector does not have to be changed and can be implemented as described in chapter 5.3: The previously applied voltage vector is kept until the VSP t_{sw} ($0 \leq t_{\mathrm{sw}} \leq T_{\mathrm{s}}$) is reached. Then, the new voltage vector is applied. The following switching state selection of course has to be implemented in a slightly different way: As both voltage vectors are already known, in a first step the optimum switching state for the first interval, i.e. for the time $0 \ldots t_{\mathrm{sw}}$ has to be determined as described in chapter 4.5.2. Of course, in this case the "sampling time" is not constant and t_{sw} has to be used for the prediction equations instead of T_{s}. Then, in a next step, the optimum switching state for the second interval, i.e. for the time $t_{\mathrm{sw}} \ldots T_{\mathrm{s}}$ has to be found. It can be done exactly in the same way as for the first interval but in this case the "sampling time" $T_{\mathrm{s}} - t_{\mathrm{sw}}$ has to be used instead of T_{s} for the prediction equations. It should be noted that the switching state selection for the second interval is independent of that for the first one.

One possibility to reduce the calculation effort and also the switching frequency in a very simple way is to keep not only the previously applied voltage vector during the first interval but also the previously applied switching state. Then, the voltage balancing algorithm has to be executed only once in a sample. Of course, the user has to be aware that in some cases only a suboptimal voltage balancing result might be obtained. Thus, this proposed strategy has to be applied with care.

5.6.1.4 VSP2CC with heuristic voltage vector preselection

As already mentioned, the VSP2CC algorithm comes with an increased calculation effort compared to PCC. Because of this it might be necessary to execute the algorithm with a reduced sampling rate which again leads to higher current ripples. In order to overcome this drawback, VSP2CC can also be combined with a heuristic preselection of the optimum voltage vector. In this case the heuristic preselection is implemented for a prediction horizon of only one sample.

5.6.2 Penalties on the control action

All algorithms presented so far do not include any penalties on the control action in their cost function, i.e. the effort to change the actuating variables is not considered. Since this work only deals with FS-MPC where a reduction of ripples is the major goal, penalties on the control action are contradictory for selecting the optimum voltage vector: By penalizing the control action, a voltage vector will in general be kept longer and the dynamic behavior will be deteriorated. Thus, such penalties can lead to deteriorated control results. However, for continuous-valued MPC, i.e. MPC with subsequent PWM, ripples on the controlled variables are not the major problem. Then, without penalties on the control action, a resulting continous-valued MPC behaves like a dead beat control scheme. If the measurements are noisy, this can result in undesired oscillations and ripples which means that the control result is also deteriorated. In order to avoid such oscillations and ripples, a certain penalty can be put on the control action. Then, the controller will become slower and will not immediately react to smaller control errors. The additional weighting factors for the penalties on the control action then have to be tuned by the user in order to achieve the desired control result. Furthermore, for medium- and high-voltage drive systems where ripples are less important than a very low switching frequency, penalties on the control action are also useful.

However, for the switching state selection penalties on the control action (on the number of switches which have to change their state) can also have a benefit: Then, the switching frequency can be reduced for the cost of a little higher voltage unbalance. Since the voltage vector selection is implemented without such penalties and as long as the resulting voltage unbalances are still small, the same control result in terms of ripples can be expected but it can be achieved with a lower switching frequency.

In order to implement penalties on the control action in the switching state selection, the algorithm which is described in chapter 4.5.2 only needs two minor modifications:

1. The previously applied switching state has to be saved and for every tested switching state the number of transitions has to be calculated.

2. The penalty on the number of transitions has to be included in the cost function (one additional weighting factor).

If a linear (L_1-norm) cost function is used, equation (5.10) must be modified as follows:

$$j_{\text{vb, lin, fc}} = \sum_{i=1}^{3} |0.5V_{\text{dc}} - v_{\text{c}i}(k+1)| + w_{\text{penalty, lin}} \cdot \Delta S \tag{5.31}$$

For the case of an L_2-norm, equation (4.27) has to be extended to

$$j_{\text{vb, quad, fc}} = \sum_{i=1}^{3} \left(0.5V_{\text{dc}} - v_{ci}(k+1)\right)^2 + w_{\text{penalty, quad}} \cdot \Delta S. \tag{5.32}$$

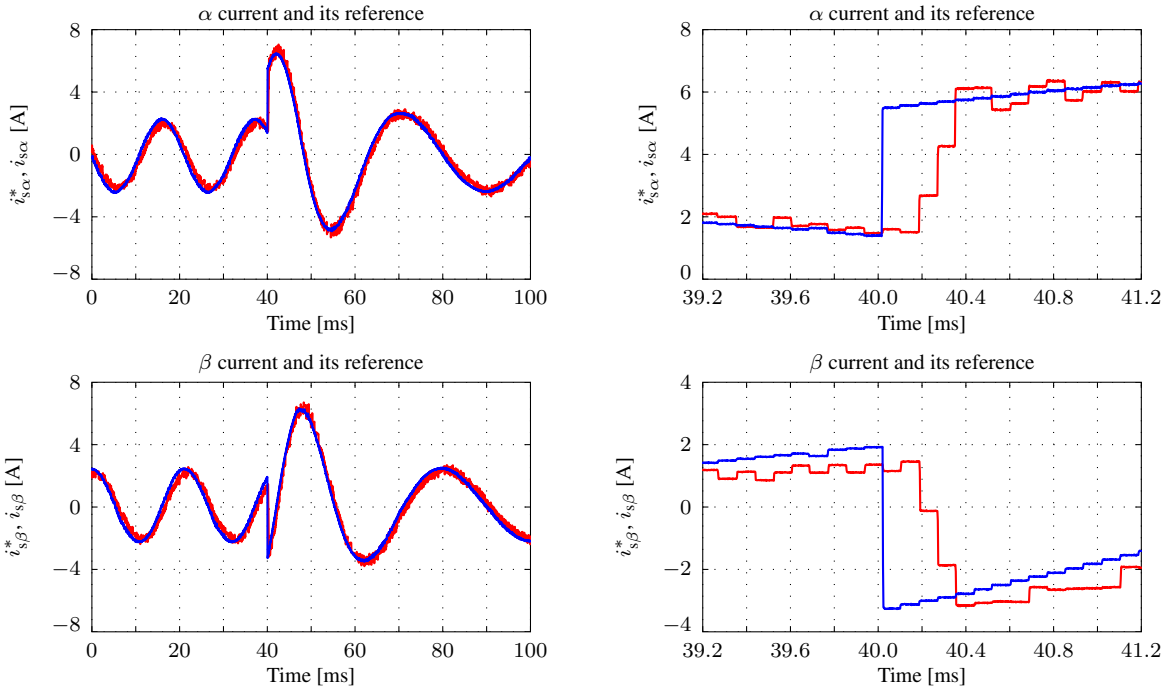
$w_{\text{penalty, lin}}$ and $w_{\text{penalty, quad}}$ are the weighting factors and ΔS is the number of transitions which are necessary to change the switching state from the previously applied to the tested one.

5.6.3 Experimental results

For all shown experimental results quadratic cost function terms were used, except for the heuristic voltage vector preselection: In this case a linear cost function was used to determine the continuous-valued optimum. The following discrete-valued optimization, however, was again performed with an L_2-norm.

5.6.3.1 PCC

First, the PCC algorithm was verified experimentally. It was implemented on the three-level inverter test bench with a sampling time of $83.33\,\mu$s which corresponds to a sampling rate of $12\,$kHz.



(a) larger time scale (b) zoom during reference steps

Figure 5.30: $\alpha\beta$ current reference steps

Figure 5.30 shows the control result for $\alpha\beta$ current reference steps. These were produced in closed-loop speed control by changing the speed reference from full nominal speed (2830 rpm)

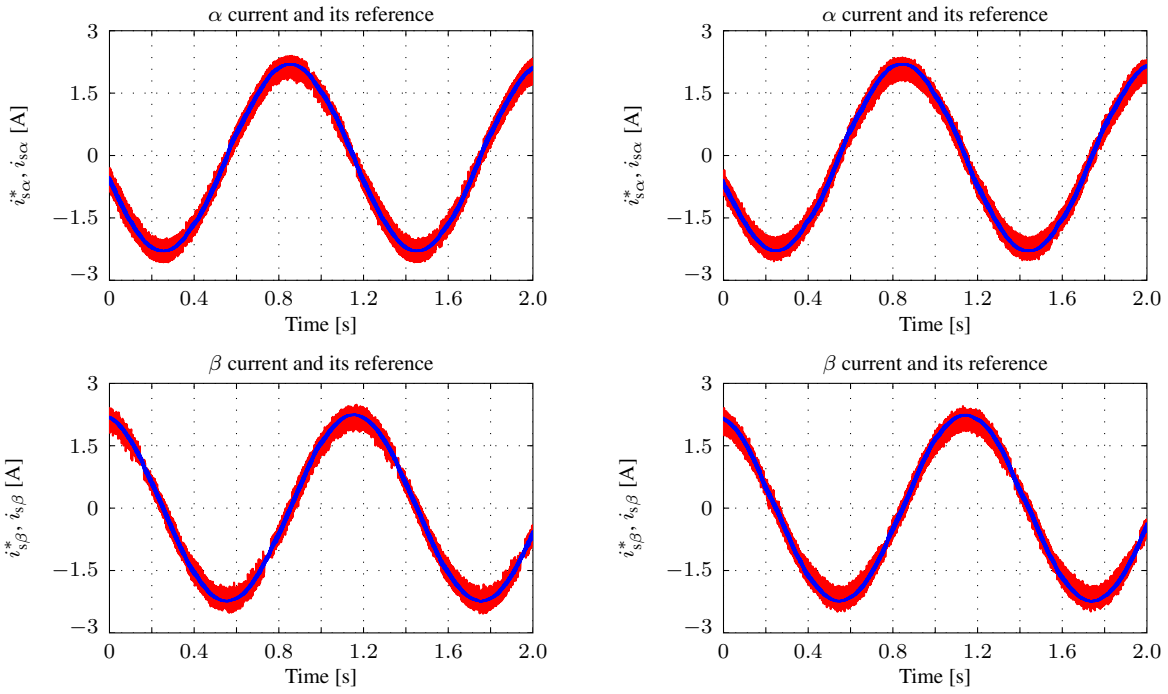
to half nominal speed (1415 rpm). It is clearly visible that the algorithm has no problems to follow the references. During the reference step the normal delay of two samples can be seen but the currents reach values around their references within three to four samples again. This performed experiment verifies that the PCC algorithm can also be successfully applied to three-level FC inverters and that a real-time implementation of the proposed control algorithm is possible. Further results are omitted for a sake of brevity.

5.6.3.2 PCC with heuristic voltage vector preselection

In order to prove that the heuristic voltage vector preselection can also be applied to FC inverters driving an IM, experimental results are presented for one, two and three prediction steps for the voltage vector selection. As explained in chapter 5.6.1.2, for the subsequent voltage balancing algorithm a prediction horizon of only one sample is enough.

One prediction step

The heuristic voltage vector preselection can of course also be applied if only a single prediction



(a) $w_{\text{penalty, quad}} = 5$, $f_{\text{sw}} \approx 0.5\,\text{kHz}$

(b) $w_{\text{penalty, quad}} = 0$, $f_{\text{sw}} \approx 4.4\,\text{kHz}$

Figure 5.31: Influence of $w_{\text{penalty, quad}}$ on the current control result ($\omega_{\text{m}} = 50\,\text{rpm}$)

step is implemented. As the calculation effort is reduced, a higher sampling frequency can be used compared to a full enumeration of all voltage vectors. In this case a sampling time of 62.5 µs was used.

In order to demonstrate that the weighting factor $w_{\text{penalty, quad}}$ for the voltage vector selection has no influence on the current control result, both α and β components of the stator currents were recorded at 50 rpm. The control result with penalty on the control action can be seen in

Figure 5.31(a) while the one in Figure 5.31(b) was recorded with $w_{\text{penalty, quad}} = 0$. By comparing the plots with each other, it is clearly visible that the penalty on the control action for the switching state selection only has very little influence on the current control result. Without penalty an average switching frequency of 4.4 kHz was measured while the penalty could reduce it to only 500 Hz. Despite the high switching frequency no better control result in terms of current ripples could be achieved.

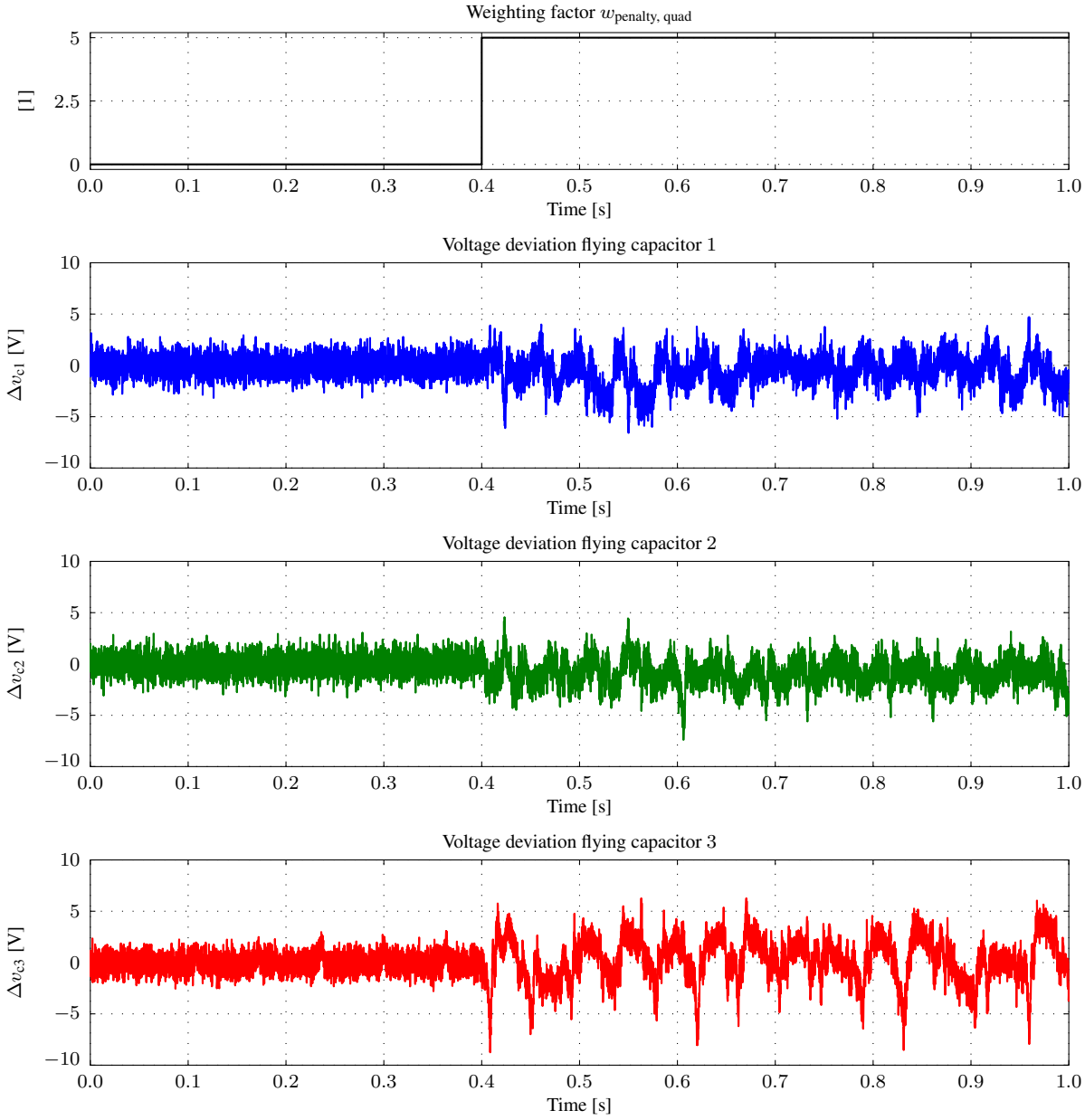


Figure 5.32: Penalty on the control action (switching state selection) and its influence on the voltage balancing (at 1415 rpm)

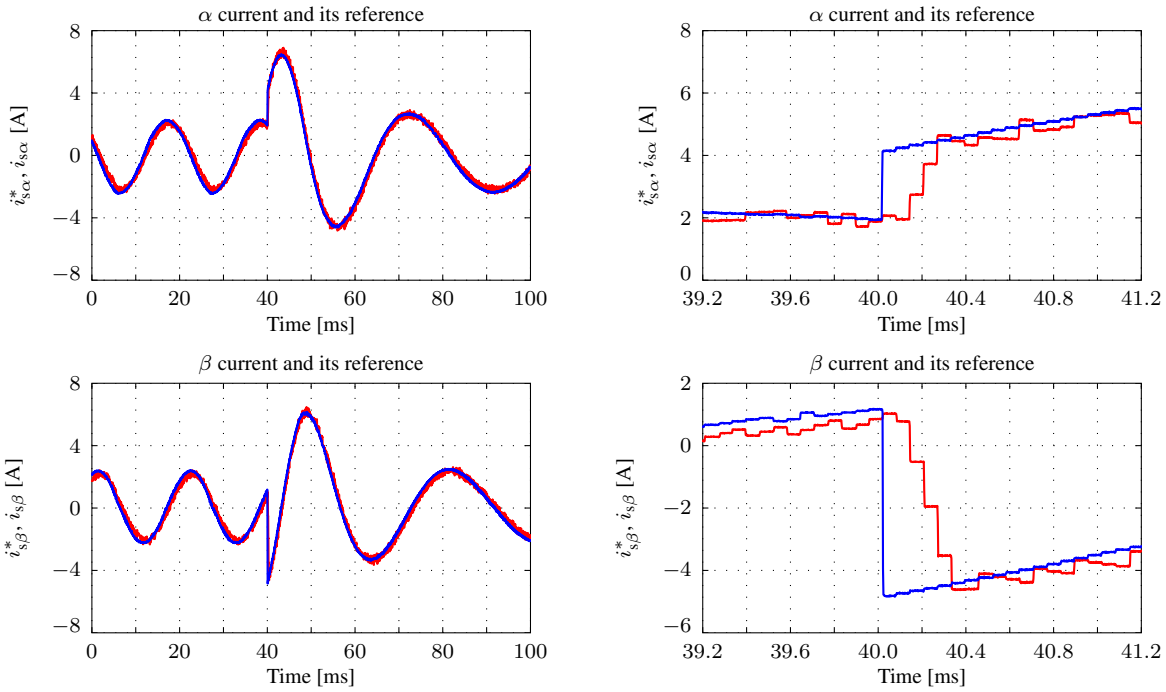
Figure 5.32 shows the influence of penalties on the control action on the voltage balancing. The experiment was conducted at half nominal speed (1415 rpm). At 0.4 s the weighting factor

$w_{\text{penalty, quad}}$ was changed from 0 to 5, i.e. after $0.4\,\text{s}$ the number of transitions was penalized. As expected, larger deviations of the FC voltages from their references can be seen. However, if these are compared to Figure 4.21, it is clearly visible that the deviations are much smaller. At $1415\,\text{rpm}$ and without penalties on the control action the average switching frequency per IGBT was around $3.8\,\text{kHz}$. If the weighting factor for these penalties was set to 5, the switching frequency could be reduced to only $1.6\,\text{kHz}$.

These experiments clearly verify that it is useful to include penalties on the control action in the switching state selection. Without these penalties this results in a much higher switching frequency.

Two prediction steps

For the selection of the optimum voltage vector only three possibilities have to be tested per prediction step. If the heuristic preselection is used, only nine different combinations have to be tested in order to determine the best voltage vector. This number is even less than a full enumeration (basic PCC algorithm) with only one single prediction step. Thus, it was possible to implement the proposed algorithm in real-time with $16\,\text{kHz}$ (corresponding to $62.5\,\mu\text{s}$). Considering the voltage vectors, in this way only 2.5% of all possible combinations have to be evaluated which is already a tremendous reduction of the calculation effort.



(a) larger time scale (b) zoom during reference steps

Figure 5.33: $\alpha\beta$ current reference steps

Figure 5.33 shows the current control result for both stator current components $i_{s\alpha}$ and $i_{s\beta}$. As in the previous cases, closed-loop speed control was performed and the reference steps were produced by changing the speed reference from full to half nominal speed. It is clearly visible that the currents follow their references without any problems. Due to the fact that a higher

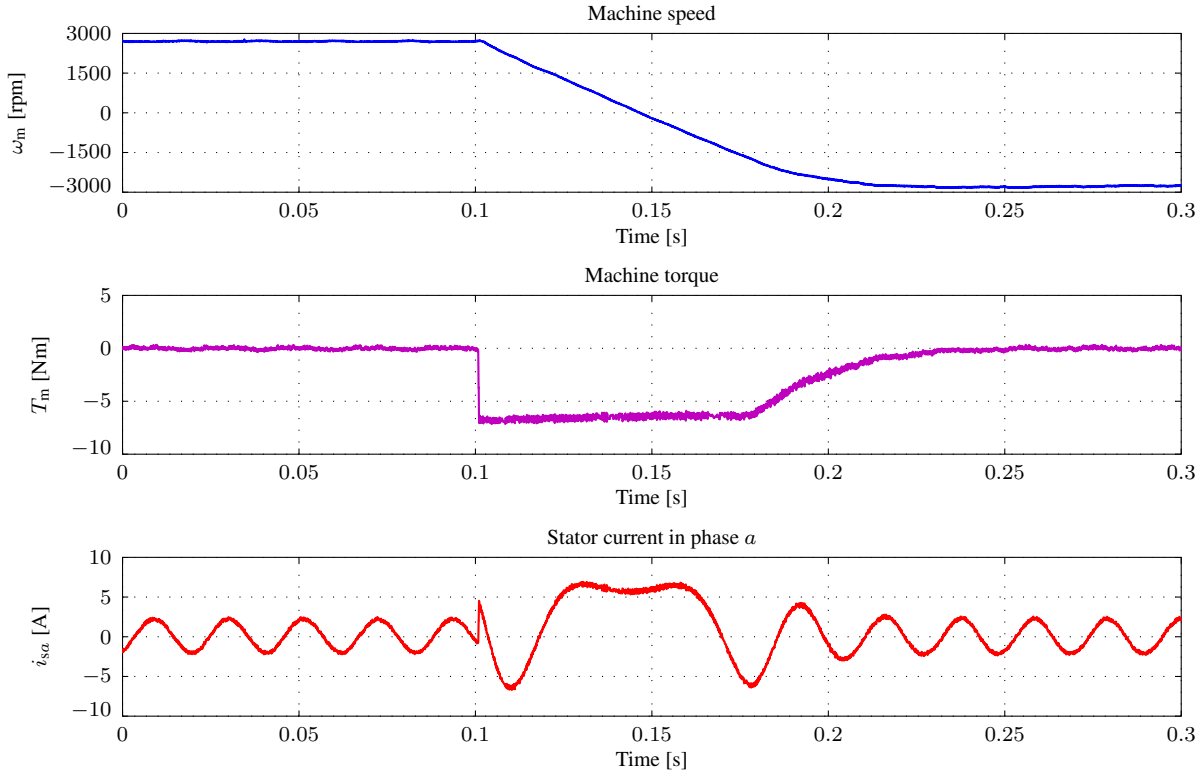


Figure 5.34: Speed reversal from positive nominal to negative nominal speed

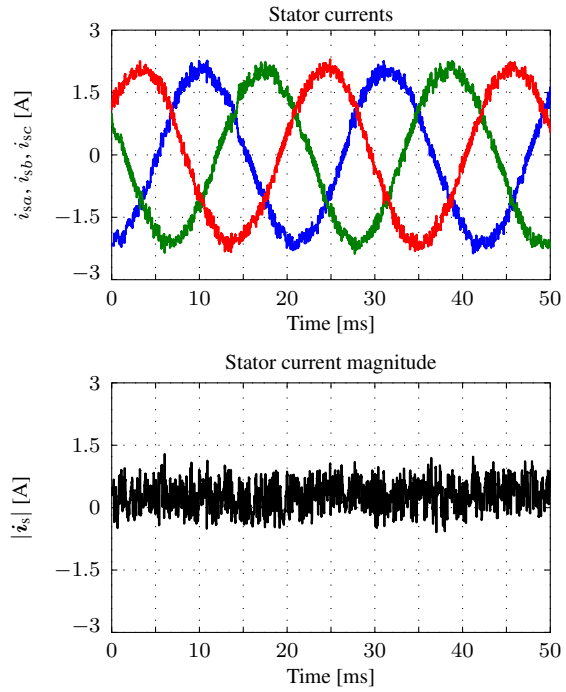


Figure 5.35: Steady-state phase currents at 2830 rpm and no load

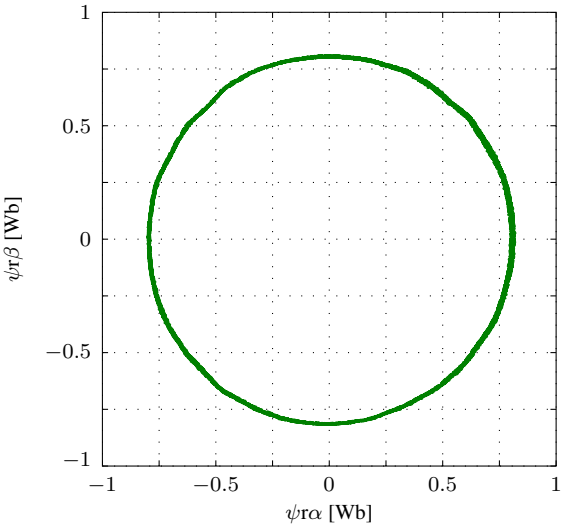


Figure 5.36: Steady-state rotor flux in $\alpha\beta$ coordinates at 2830 rpm and no load

sampling rate and two prediction steps instead of only one were implemented, the control result is slightly improved compared to Figure 5.30.

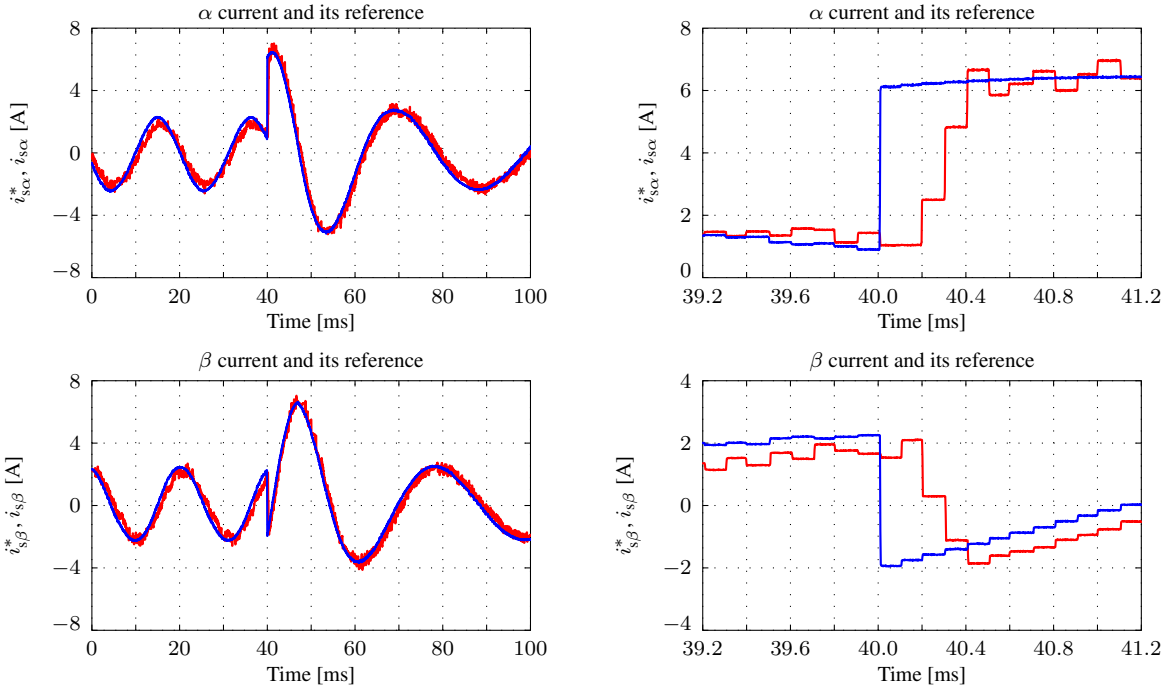
In order to prove that the algorithm works in the whole speed range, a speed reversal from positive to negative nominal speed was conducted which can be seen in Figure 5.34. It is clearly visible that this control task can be fulfilled very well. Due to the fact that the machine speed ω_m is controlled with a simple PI controller, the system is not operated at its physical limits when the speed is already very close to its reference again which can clearly be seen from the machine torque T_m in the second graph.

Figure 5.35 shows the three phase currents i_{sa}, i_{sb} and i_{sc} at $2830\,\mathrm{rpm}$ in the upper plot, below the stator current magnitude $|\boldsymbol{i}_\mathrm{s}|$ can be seen. The waveforms are more or less perfectly sinusoidal and even the current ripples are quite small.

Finally, the estimated rotor flux $\boldsymbol{\psi}_\mathrm{r}$ was recorded at full nominal speed and plotted in $\alpha\beta$ coordinates which can be seen in Figure 5.36. The reference value was set to $0.8\,\mathrm{Wb}$. As expected, both components of the rotor flux form a circle.

Three prediction steps

In order to prove that even three prediction steps can be successfully implemented in real-time, several experiments were conducted with three prediction steps for the voltage vector selection. Compared to a full enumeration, in this case only 27 different combinations of voltage vectors



(a) larger time scale

(b) zoom during reference steps

Figure 5.37: $\alpha\beta$ current reference steps

have to be evaluated which is 0.4% of all possible combinations. For the subsequent voltage balancing again a prediction horizon of only one sample was used. Due to the fact that in this

case the calculation effort is much higher than for two prediction steps, the algorithm could only be implemented with a sampling frequency of 12 kHz.

First, the current control result (reference steps) is shown in Figure 5.37. The control result is comparable to the one shown in Figure 5.30.

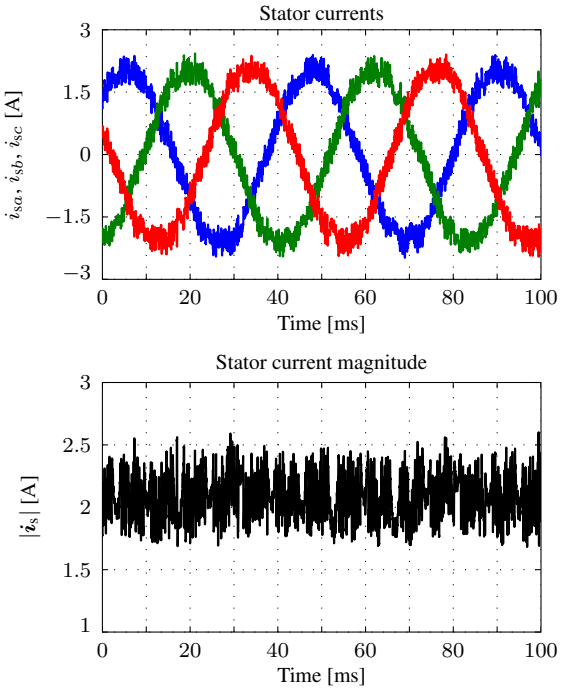


Figure 5.38: Steady-state phase currents at 1415 rpm, no load

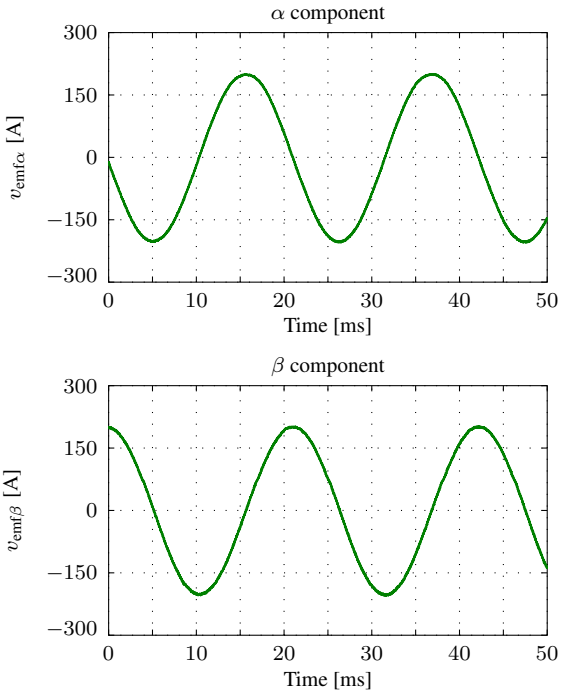


Figure 5.39: Estimated back-EMF voltage in steady state at 2830 rpm and no load

Another experiment shows the phase currents and their magnitude (Figure 5.38) at half nominal speed (1415 rpm). Although the sinusoidal waveform is clearly visible, the current ripples are significantly higher compared to the ones in Figure 5.35: This is mostly due to the fact that a lower sampling rate was used (12 kHz instead of 16 kHz). Furthermore, the currents are recorded at a different operating point.

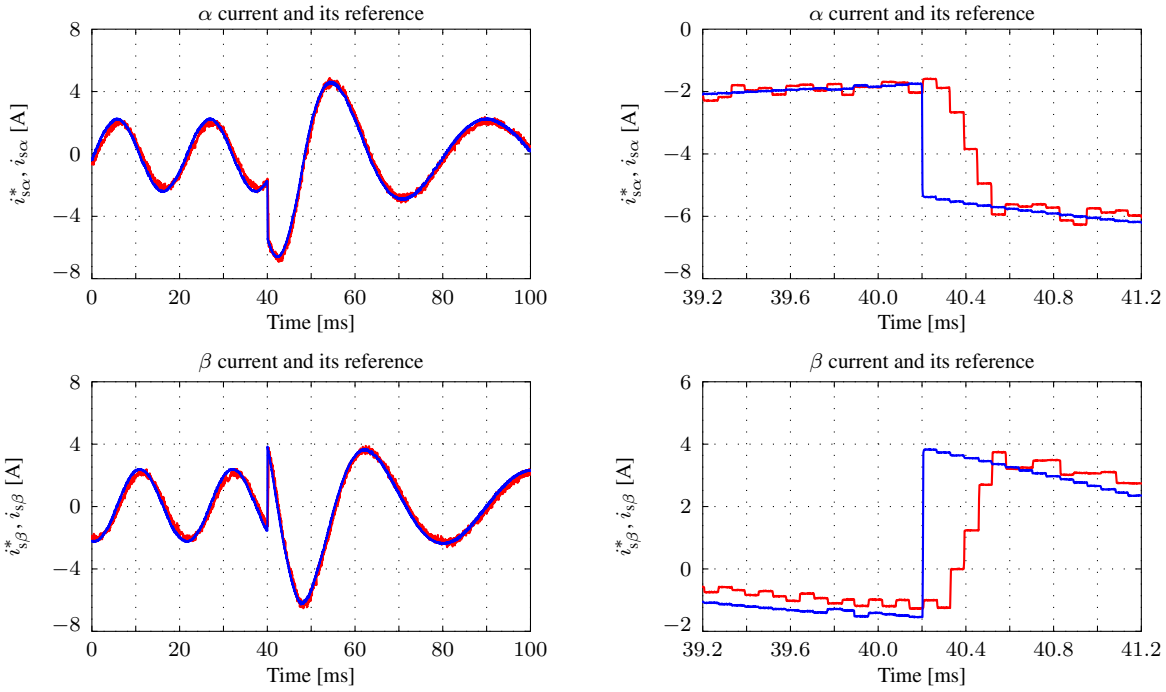
Finally, the last experiment is shown in Figure 5.39. In this case both α and β components of the estimated back-EMF voltages are shown. The experiment was conducted at full nominal speed. As expected, the voltages are sinusoidal and show a phase shift of 90°.

5.6.3.3 VSP2CC

Since the experimental results for VSP2CC applied to an FC inverter driving an IM do not significantly change from the ones obtained for VSP2CC with heuristic voltage vector preselection, no experimental results are shown at this point. Of course, since VSP2CC requires a significantly higher calculation effort than VSP2CC with heuristic voltage vector preselection, the VSP2CC algorithm without preselection can only be executed with a lower sampling rate.

5.6.3.4 VSP2CC with heuristic voltage vector preselection

In order to prove that the combination of VSP2CC with the heuristic preselection strategy can also be applied to three-level FC inverters driving an IM, several experimental results are shown. All results were obtained with a sampling frequency of 16 kHz and with $w_{\text{penalty, quad}} = 5$.



(a) larger time scale

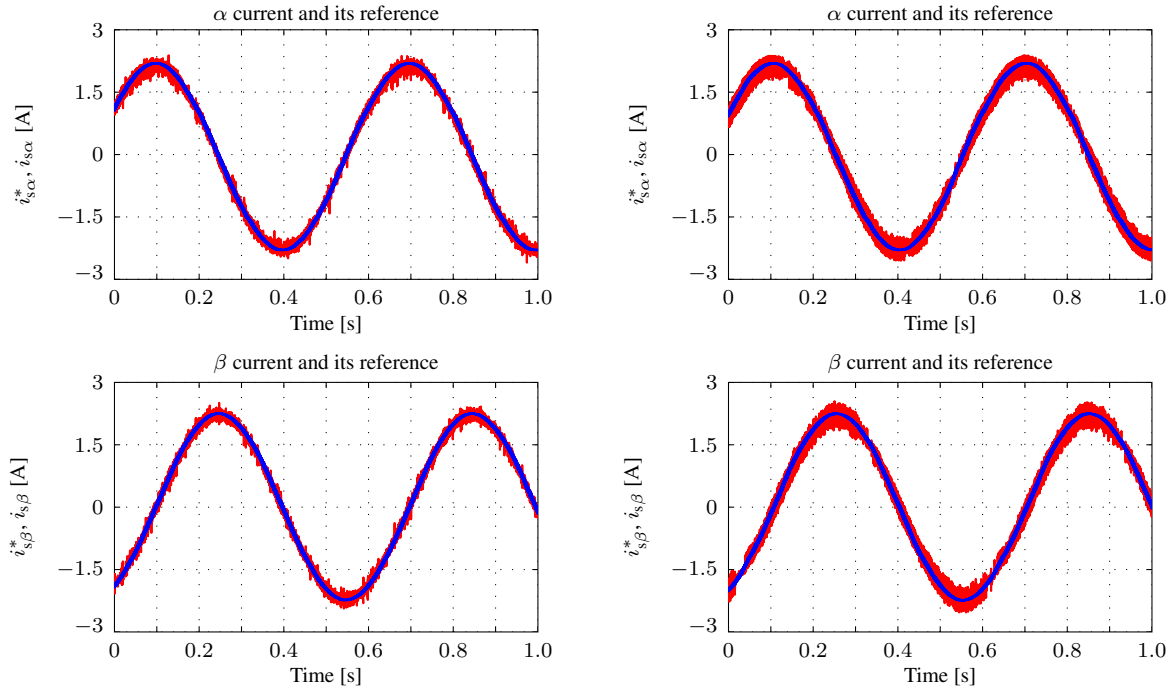
(b) zoom during reference steps

Figure 5.40: $\alpha\beta$ current reference steps

The first experiment which is shown in Figure 5.40 was conducted in order to prove that the transient response in not deteriorated compared to a full enumeration and a fixed switching point. The stator currents $i_{s\alpha}$ and $i_{s\beta}$ were recorded when the speed reference was changed from 2830 rpm to 1415 rpm. After the normal delay of two samples the currents reach values around their references within three to four samples.

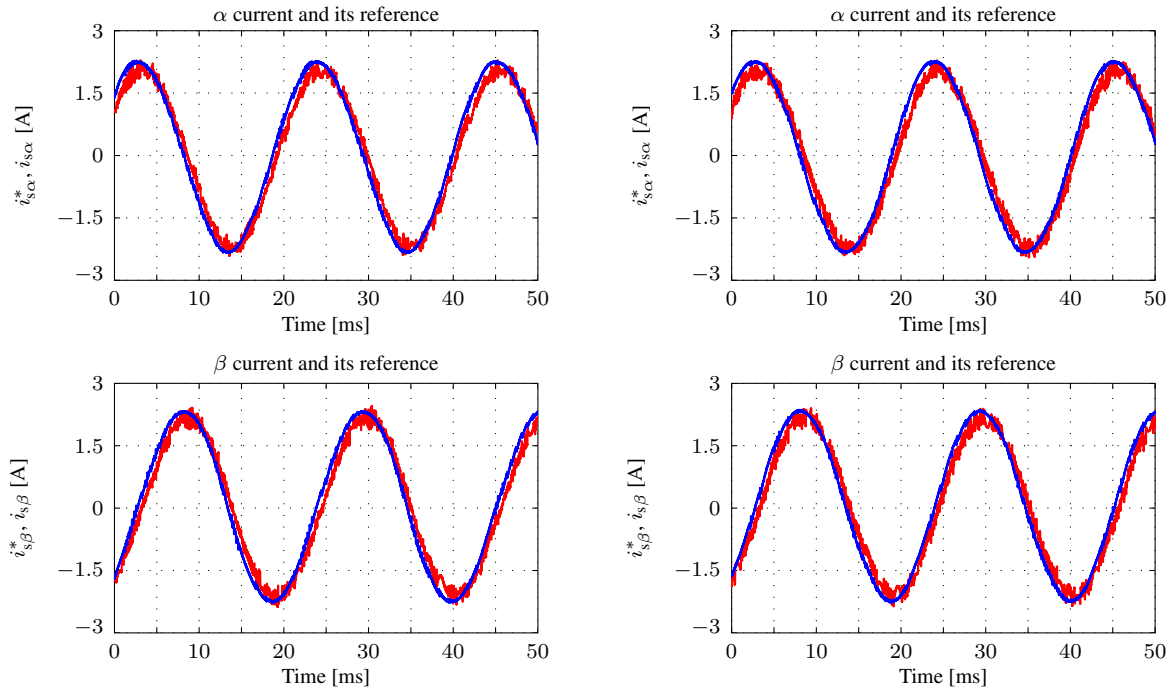
In order to prove that the proposed VSP2CC stategy with heuristic voltage vector preselection can successfully reduce current ripples, the steady-state stator currents were recorded when the machine was rotating with 100 rpm for both VSP2CC with heuristic voltage vector preselection (Figure 5.41(a)) and for PCC with heuristic preselection of the optimum voltage vector (Figure 5.41(b)). It is obvious that the VSP2CC strategy can also significantly reduce current ripples at lower speeds.

The same experiment was conducted at full nominal speed (2830 rpm). In Figure 5.42(a) the results for VSP2CC can be seen, Figure 5.42(b) shows the results for PCC. In this case no improvement of the control result can be seen. The measured switching frequencies per IGBT were in both cases about 1.65 kHz. As for a two-level and for a three-level NPC inverter, VSP strategies are more effective at lower speeds.



(a) VSP2CC, $f_{sw} \approx 1.1\,\text{kHz}$ (b) PCC, $f_{sw} \approx 0.8\,\text{kHz}$

Figure 5.41: $\alpha\beta$ steady-state currents at 100 rpm



(a) VSP2CC, $f_{sw} \approx 1.65\,\text{kHz}$ (b) PCC, $f_{sw} \approx 1.65\,\text{kHz}$

Figure 5.42: $\alpha\beta$ steady-state currents at 2830 rpm

5.6.4 Evaluation

The shown experimental results verify that FS-MPC can also be successfully applied to three-level FC inverters. Although a three-phase three-level FC inverter has 64 different switching states, the proposed decoupling and heuristic voltage vector preselection strategy allows to reduce the calculation effort such that even three prediction steps are possible in real-time. Furthermore, the VSP2CC strategy can also be successfully applied to three-level FC inverters and it is also possible to combine this strategy with a heuristic preselection of the optimum voltage vector. Thus, the two main goals of this work (reduction of the calcuation effort and reduction of ripples on the controlled variables for FS-MPC) could also be successfully achieved for this inverter topology. In order to reduce the switching frequency for the cost of a slightly increased FC voltage unbalance, a penalty on the number of IGBT transitions can be added to the switching state selection. In this way it is possible to reduce the average switching frequency while the quality of the delivered currents is not deteriorated.

CHAPTER 6

Oversampling Finite-Set Model Predictive Control

As already mentioned in chapter 3, control algorithms can be implemented in software or in hardware. Software-based implementations in general offer more flexibility and the programming is in most cases much easier than for hardware-based implementations. The main drawback of software-based solutions is that the sampling times are limited to much higher values than for hardware-based implementations. Because of this a higher time-resolution for software implementations can only be achieved if

1. a modulator is used or
2. a VSP is calculated.

It should be noted that a modulator also has to be implemented in hardware and that for a VSP realization a hardware structure similar to a modulator has to be used which means that a high time resolution can *only* be realized in hardware.

By taking into account that for hardware implementations the calculations can normally be done within a few very short clock cycles, the sampling frequency of the controller can be significantly increased. Furthermore, as no time delay compensation is necessary because of the fast calculation time (while the sampling frequency is still significantly lower than the hardware clock frequency), the control algorithm can be further simplified. Thus, if the sampling frequency of the controller is increased, the time resolution of the FS-MPC strategy can also be significantly increased compared to a software-based implementation on a conventional test bench.

6.1 Basic idea

Figure 6.1 shows the basic strategy of the oversampling approach [59]. In this case—without loss of generality—an actuator with only two different switching states is assumed, one which leads to a high positive slope of the controlled variable y (active switching state) and one which

leads to a small negative slope (zero switching state). The positive slope is *four* times greater than the negative one. As already explained in chapter 2.7.5.4, a high sampling time T_s leads to high ripples on the controlled variable y because the active switching state has to be applied for at least one whole sample T_s. Since there is no other switching state with a smaller positive torque slope, the ripple cannot be reduced. Theoretically, one switching event can happen per sample T_s. Thus, for every FS-MPC strategy the theoretical maximum switching frequency per IGBT is equal to half the sampling frequency (assuming that two transitions are counted as one switching event). In practical implementations, however, the switching frequency per IGBT is *significantly* lower than this value which can be seen in Figure 6.1: Only two transitions per five samples can happen because of the given slopes. If now one sample T_s is divided into five equal samples $T_{\mathrm{s,\,os}}$, the switching frequency can be increased by a factor of five. Then, with the high sampling time $T_{\mathrm{s,\,os}}$, a switching state still has to be applied for a whole sample. In contrast to normal FS-MPC, however, the minimum time for which a switching state has to be applied, is just a *fifth* of T_s. Because of this ripples on the controlled variable y can be significantly reduced. Of course, $T_{\mathrm{s,\,os}}$ does not necessarily have to be a fifth of T_s. In general, the smaller $T_{\mathrm{s,\,os}}$ is, the smaller are the ripples on the controlled variables and as higher is the theoretically maximum switching frequency.

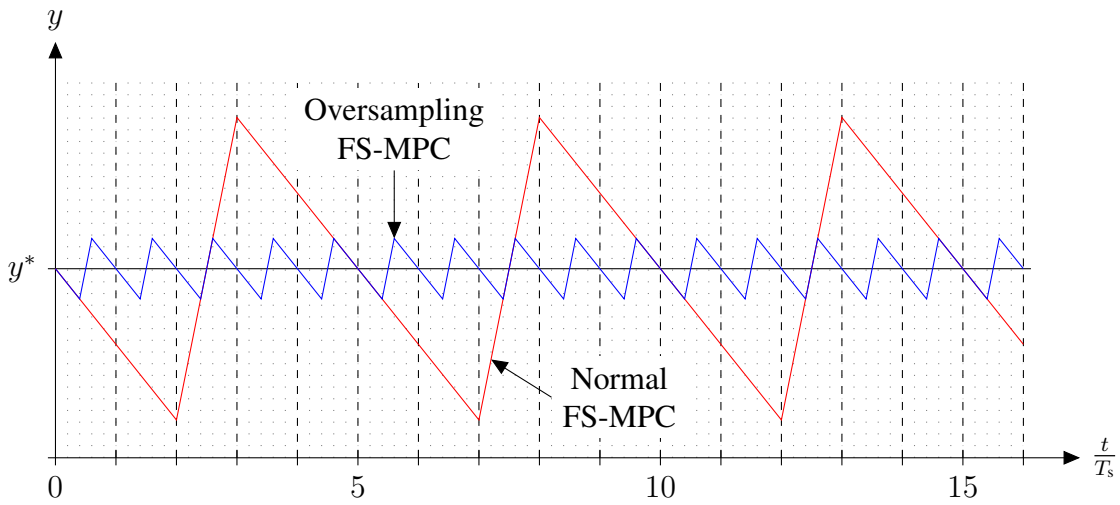


Figure 6.1: Oversampling-based FS-MPC compared to normal FS-MPC

If, however, $T_{\mathrm{s,\,os}}$ is *significantly* smaller than T_s, the maximum switching frequency per IGBT can be exceeded. As already explained and as it can be seen in Figure 6.1, the highest achievable switching frequency for $T_{\mathrm{s,\,os}}$ and for the given slopes is $\frac{1}{T_\mathrm{s}}$. Now, assuming that the maximum switching frequency per IGBT is limited to $\frac{0.5}{T_\mathrm{s}}$, an additional hard constraint has to be imposed on the number of transitions in order not to exceed the physical limits of the inverter. Then, however, a prediction horizon of only *one* sample would not be enough: In this case the controller would allow the inverter to switch when the hard constraint is not active and as soon as it is possible. This would again lead to high ripples on the controlled variable y which can be seen in Figure 6.2: In this case it is again assumed that only one switch exists and that T_s is *five* times greater than $T_{\mathrm{s,\,os}}$. The maximum switching frequency shall be limited to $\frac{0.5}{T_\mathrm{s}}$, i.e. only *two* switchings are allowed to happen during *ten* samples $T_{\mathrm{s,\,os}}$. Furthermore, in this case

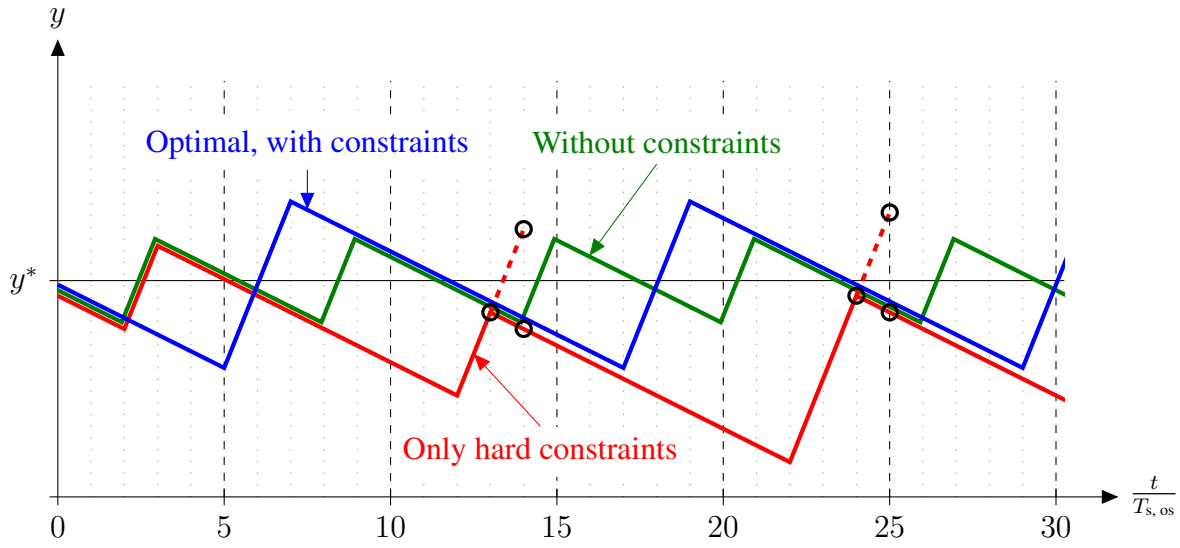


Figure 6.2: Oversampling-based FS-MPC with constraints on the switching frequency

the active switching state leads to a positive slope whose magnitude is *five* times higher than the one of the zero switching state. The green line shows the trajectory if the constraint on the number of transitions is not considered which leads to ten switching events. In contrast to that, the red line shows the trajectory if *only* the hard constraint on the transitions is considered: At the beginning, when the real value y is slightly less than its reference y^*, it is assumed that no switchings have happened before. Then, the first transition will occur as soon as possible (at time step 2, i.e. at the same time as without constraints). Since another transition is still possible, the actuator will switch back to the zero state at time step 3. During the steps 8 until 11 the hard constraint is still active and hence, the zero state is kept. Then, at step 12, the following problem occurs: The prediction horizon is only *one* sampling cycle and hence, the controller will choose the zero switching state because it leads to a smaller error. The corresponding points are marked with circles in Figure 6.2. Then, because of the active hard constraint, the zero state has to be kept until sample 22. At that time, however, the deviation of the controlled variable y from its reference y^* is already *twice* as high as it would be in the ideal case which is shown by the blue line. The same problem occurs at time step 24. Although a hard constraint on the switching frequency is used, the resulting trajectory is not optimal. The optimal trajectory needs also just *five* transitions but the resulting ripple on the controlled variable is significantly less compared to the one of the red trajectory. Although the higher sampling frequency and hard constraints on the number of transitions still lead to less ripples compared to FS-MPC without oversampling, the control result is not optimal.

The best solution for this problem would of course be to increase the prediction horizon to ten or more samples—then, the optimal trajectory (blue line in Figure 6.1) could easily be found. However, as shown in chapter 2.7.5.4, an increased prediction horizon quickly leads to optimization tasks which are not feasible in real-time.

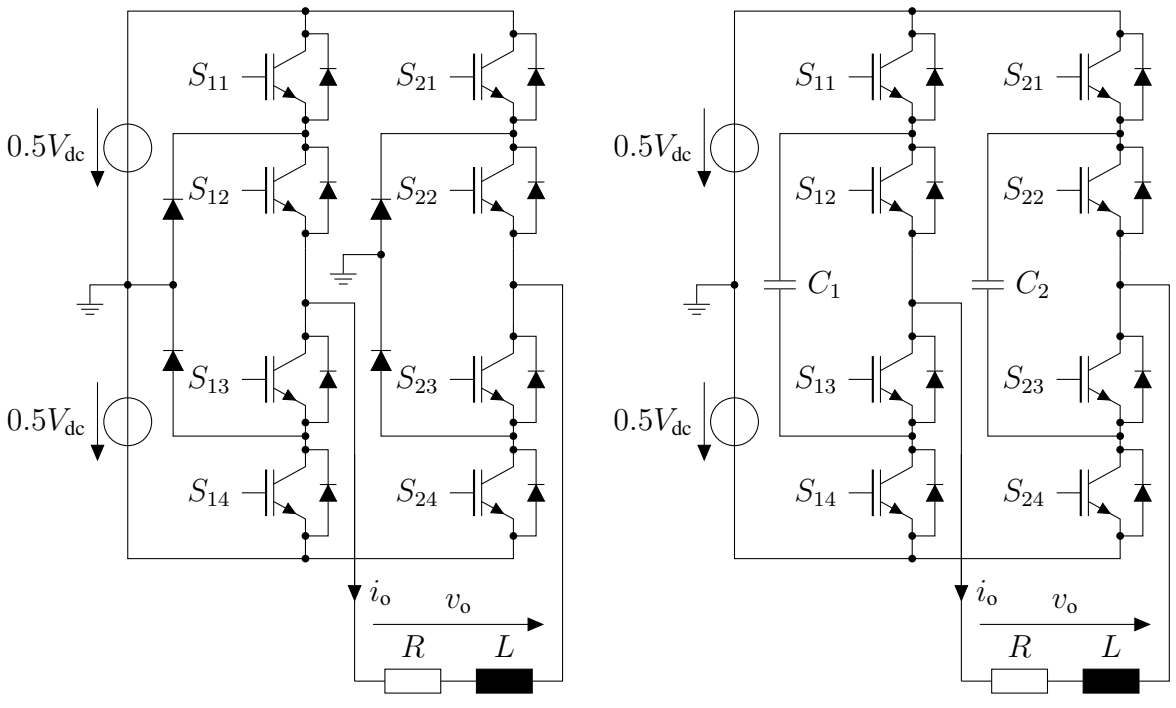
Thus, if only one prediction step can be realized, other strategies have to be found which lead to control results comparable to those with a higher prediction horizon. In order to come closer to the optimal behavior, two modifications are proposed:

1. Since the switching frequency is limited to $\frac{0.5}{T_\mathrm{s}} = \frac{1}{10 \cdot T_{\mathrm{s,\,os}}}$, the *average* time for which a switching state is applied is equal to $T_\mathrm{s} = 5T_{\mathrm{s,\,os}}$. Thus, the prediction can be done assuming the *original* sampling time T_s although in fact the sampling time is $T_{\mathrm{s,\,os}}$. Such an extrapolation does not lead to a higher calculation effort but allows to look farther into the future. This strategy which comes with *no* additional calculation effort can already *significantly* reduce ripples on the controlled variables.

2. By taking a closer look at the time instances when the switchings happen in Figure 6.2, it can be observed that the switchings leading to the optimal trajectory are more equally distributed over the time compared to the red line. Thus, additional *soft constraints* on the number of transitions can be used: If a switching event occurs, this leads to an additional penalty and thus to a higher cost. Because of this a switching event will only happen if the control deviation has exceeded a certain value, i.e. it will happen *later* as without soft constraints. This leads to the fact that switching states which produce *high* slopes are applied for a longer time which then results in a more equal distribution of the switching events. Of course, the corresponding weighting factor for these soft constraints needs to be well-tuned to get the desired result for all operating points.

6.2 Practical realization

As already mentioned, for a practical realization of the proposed Oversampling FS-MPC method the algorithms need to be implemented in hardware.



(a) NPC inverter (b) FC inverter

Figure 6.3: Two-leg three-level inverters with resistive-inductive load

Compared to a software-based implementation, several considerations have to be taken into account: For conventional test benches, for rapid prototyping systems but also for microcontrollers and digital signal processors (DSPs) the control algorithm can mostly be implemented in high-level programming languages like *C*. These controllers are clocked at a high frequency and execute all commands in an algorithm *sequentially*. Furthermore, it is usually also possible to implement the controllers in floating point notation and several debugging interfaces are available. This means that a successful implementation can normally be obtained quickly. For hardware-based implementations on FPGAs or CPLDs a successful implementation of controllers cannot be achieved that easily: Common hardware description languages (HDLs) like VHDL or Verilog of course also offer the possibility to program arithmetic operations and thus controllers can also be realized with HDLs. However, the controller normally has to be implemented in fixed point notation and, compared to software-implementations, less debugging possibilities are available. Furthermore, for operations which need to be executed *sequentially*, i.e. one result is calculated from the result of a previous computation, signal propagation times and delays have to be taken into account. Of course, the number of available hardware elements (logic elements, multipliers etc.) is also limited. Then, further techniques like pipelining have to be used in order to successfully implement a control algorithm.

These mentioned issues make clear that the controller complexity has a *considerable* influence on the technical feasibility of hardware implementations and especially on the time which is necessary for a successful implementation.

6.3 Control algorithms

6.3.1 Resistive-inductive load

Figure 6.3 shows a two-leg NPC inverter (left) and a two-leg FC inverter (right) connected to a resistive-inductive load. The system equations of a resistive-inductive load were derived in chapter 2.5.5. For an implementation of the control algorithm equation (2.27) has to be discretized with the sampling time. As already mentioned in chapter 6.1, although the Oversampling FS-MPC method was in fact implemented with the sampling time $T_{\mathrm{s,\,os}}$, better results can be obtained when the *original* sampling time T_{s} is used for the predictions. In this case an exact discretization is applied instead of a simple Euler-forward approximation: According to [60], the complete solution of a linear system as stated in equation (2.1) is given by

$$\boldsymbol{x} = e^{\boldsymbol{A}t}\boldsymbol{k} + \boldsymbol{x}_{\mathrm{p}} \tag{6.1}$$

where $\boldsymbol{k}$ is a constant vector and $\boldsymbol{x}_{\mathrm{p}}$ is a particular solution of equation (2.1) which can be determined through inspection by calculating the values of the state variables in steady state. $\boldsymbol{k}$ can be calculated from the initial conditions. For obtaining a discrete-time system representation with the sampling time T_{s}, $\boldsymbol{k}$ has to be determined first, in this case for $t = 0$:

$$\boldsymbol{k} = \boldsymbol{x}(0) - \boldsymbol{x}_{\mathrm{p}} \tag{6.2}$$

After that the state vector can be determined for $t = T_{\mathrm{s}}$:

$$\boldsymbol{x}(T_{\mathrm{s}}) = e^{\boldsymbol{A}T_{\mathrm{s}}}\boldsymbol{k} + \boldsymbol{x}_{\mathrm{p}} \tag{6.3}$$

In this way the exact discrete-time system matrices $\boldsymbol{A}_\mathrm{d}$ and $\boldsymbol{B}_\mathrm{d}$ can be obtained and with these the discrete-time system equation:

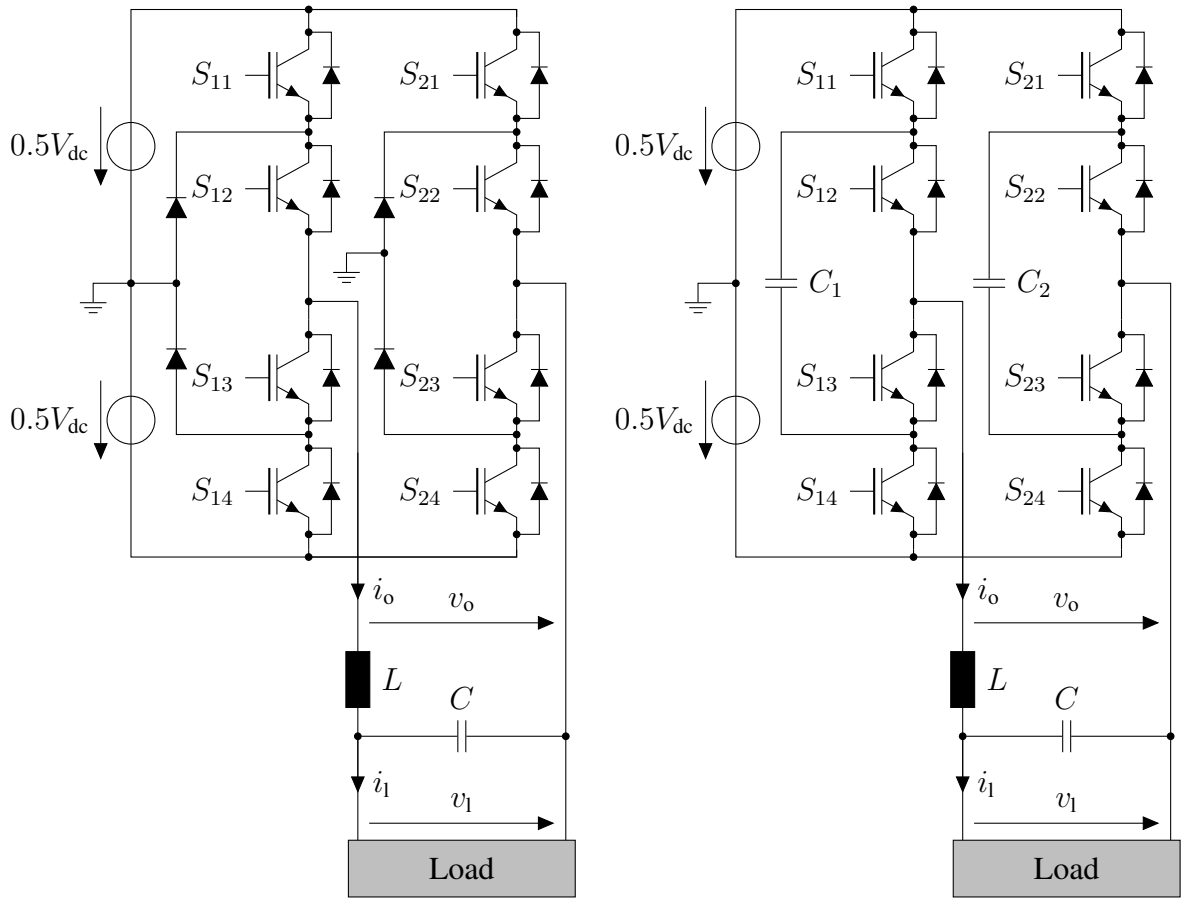
$$i_\mathrm{o}(k+1) = \boldsymbol{A}_\mathrm{d} \cdot i_\mathrm{o}(k) + \boldsymbol{B}_\mathrm{d} \cdot v_\mathrm{o}(k) \tag{6.4}$$

For the given system the inverter output current i_o should be controlled. Thus, the first term of the cost function, the control deviation, is given by

$$j_\mathrm{rl1} = \left(i_\mathrm{o}^* - i_\mathrm{o}(k+1)\right)^2 \tag{6.5}$$

where k is the current sample.

6.3.2 UPS application



(a) NPC inverter (b) FC inverter

Figure 6.4: Two-leg three-level inverters with LC lowpass-filter and unknown load (UPS)

Figure 6.4 shows a two-leg NPC (left) and a two-leg FC inverter (right) connected to an LC lowpass-filter with a subsequent (unknown) load. Such a circuit can be used as Uninterruptible

Power Supply (UPS). The system equations of an LC lowpass-filter are explained in chapter 2.5.6. The discretization is done in the same way as described in chapter 6.3.1. Then, the exact discrete-time system representation (sampling time T_s) results to

$$\begin{pmatrix} i_\mathrm{o}(k+1) \\ v_\mathrm{l}(k+1) \end{pmatrix} = \boldsymbol{A}_d \cdot \begin{pmatrix} i_\mathrm{o}(k) \\ v_\mathrm{l}(k) \end{pmatrix} + \boldsymbol{B}_\mathrm{d} \cdot \begin{pmatrix} v_\mathrm{o}(k) \\ i_\mathrm{l}(k) \end{pmatrix}. \tag{6.6}$$

For the given system the LC filter output voltage v_l should be controlled. Thus, the first term of the cost function, the control deviation, is given by

$$j_\mathrm{lc1} = \left(v_\mathrm{l}^* - v_\mathrm{l}(k+1)\right)^2 \tag{6.7}$$

where k is the current sample. Furthermore, since the inverter can only operate up to a certain current i_max (which can also be the maximum inductor current of the LC filter), a hard constraint is used to consider this limit:

$$i_\mathrm{lim} = \begin{cases} 0 & \text{if } |i_\mathrm{o}| \leq i_\mathrm{max} \\ \infty & \text{if } |i_\mathrm{o}| > i_\mathrm{max} \end{cases} \tag{6.8}$$

For an accurate prediction of the UPS output voltage it is essential to have an accurate value of the load current i_l. One possibility is of course to measure it. In order to reduce costs it is also possible to estimate the load current with a Luenberger observer or Kalman filter [61]. Since the load current is assumed to change slowly compared to the sampling frequency, i.e. $i_\mathrm{l}(k+1) \approx i_\mathrm{l}(k)$, a simpler solution is to calculate it from the prediction equation of the last sample.

6.3.3 Voltage balancing

The basic voltage balancing mechanisms were described in chapter 2.5.2.3 for an NPC and in chapter 2.5.3.3 for an FC inverter. As in this case only two legs are used, the equations can be simplified: First, only two legs and thus only the currents i_1 and i_2 have to be considered. Second, since also in this case a balanced system is given, the current of the first leg is given by

$$i_1 = i_\mathrm{o}. \tag{6.9}$$

The current through the second one then results to

$$i_2 = -i_\mathrm{o}. \tag{6.10}$$

Thus, the resulting cost function term which is necessary for the NPC inverter is given by

$$j_\mathrm{vb,\,npc} = w_\mathrm{vb,\,npc} \cdot \left(\Delta v_\mathrm{c}(k+1)\right)^2. \tag{6.11}$$

For the FC inverter the term

$$j_\mathrm{vb,\,fc} = w_\mathrm{vb,\,fc} \cdot \left(0.5V_\mathrm{dc} - v_\mathrm{c1}(k+1)\right)^2 + \left(0.5V_\mathrm{dc} - v_\mathrm{c2}(k+1)\right)^2 \tag{6.12}$$

has to be added to the cost function. v_c1 and v_c2 are the voltage drops across the flying capacitors of the first and second leg, respectively.

6.3.4 Limitation of the switching frequency

The most important and key point of the proposed Oversampling FS-MPC algorithm is the limitation of the switching frequency: For the proposed method the sampling time $T_{\text{s, os}} = 10\,\mu\text{s}$ should be used and the maximum switching frequency per IGBT should be limited to $10\,\text{kHz}$. Thus, only *two* transitions per IGBT and per *ten* samples are allowed. For this reason the switching states of all IGBTs are stored in a first in first out (FIFO) buffer. Then, the transitions for every IGBT from the last 10 samples up to the current time step can be calculated. In order to penalize the current switching transition for every IGBT depending on the number of switchings during the previous 9 sampling cycles, the soft constraint

$$j_{\text{os}} = w_{\text{os}} \cdot \left(\sum_{i=1}^{2} \sum_{j=1}^{2} \left(\sum_{l=-8}^{1} s_{ij}(k+l) \right)^2 \right) \tag{6.13}$$

is used. $s_{ij}(k)$ denotes the transitions of switch j in leg i at time step k. It is important to mention that in this case it is desired to penalize a switching event more heavily if there have already been transitions of that IGBT in the past (i.e. in the last 10 samples). This effect can be achieved with a quadratic cost function term but not with a linear one.

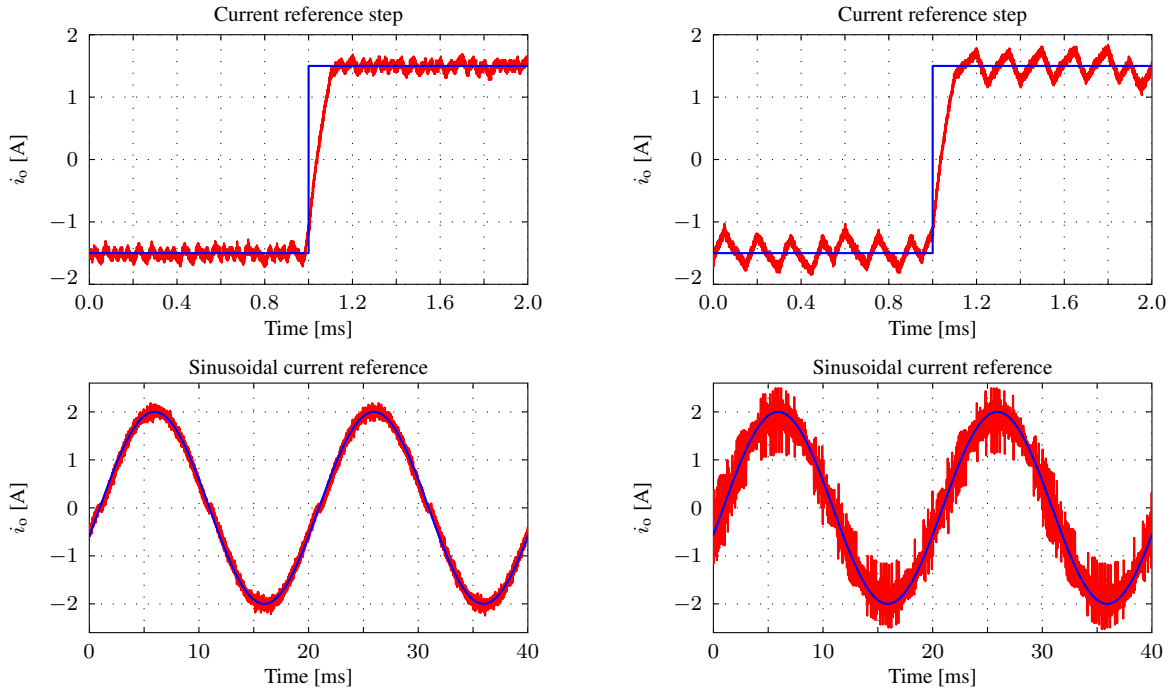
Further terms with penalties on the number of transitions (but only for the predicted time step) are added as well. Additionally, hard constraints on the number of transitions are implemented (similarly to the output current limitation in equation (6.8)). If necessary, other factors can be added to the cost function, too.

6.4 Experimental results

In order to verify the proposed Oversampling FS-MPC strategy, the control algorithm was applied to two-leg NPC and FC inverters, for current control of a resistive-inductive load and for a UPS application. The controllers were implemented on the test bench described in chapter 3.2. For all experiments a sampling frequency $f_{\text{s, os}} = 100\,\text{kHz}$ was used which corresponds to $T_{\text{s, os}} = 10\,\mu\text{s}$. The maximum switching frequency per IGBT was limited to $f_{\text{sw, max}} = 10\,\text{kHz}$ which means that every IGBT is only allowed to switch *twice* within 10 samples. Then, the proposed strategy was compared to a conventional FS-MPC implementation with the same maximum switching frequency, i.e. the controller frequency was set to $f_{\text{s}} = 20\,\text{kHz}$. All voltages and currents were measured and recorded with an oscilloscope.

6.4.1 Three-level Neutral Point Clamped inverter

Figure 6.5(a) shows the control result for a resistive-inductive load for the proposed Oversampling FS-MPC method; in Figure 6.5(b) the control result for FS-MPC without oversampling can be seen. The results were obtained with a DC link voltage $V_{\text{dc}} = 60\,\text{V}$. The load parameters are $R = 10\,\Omega$ and $L = 3\,\text{mH}$. The upper plots show an experiment where a current reference step from $-1.5\,\text{A}$ to $1.5\,\text{A}$ was commanded. It is clearly visible that the current ripple could be significantly reduced by applying the proposed method compared to conventional FS-MPC. It is also obvious that the dynamic behavior is not deteriorated. The lower plots show the result



(a) With oversampling

(b) Without oversampling

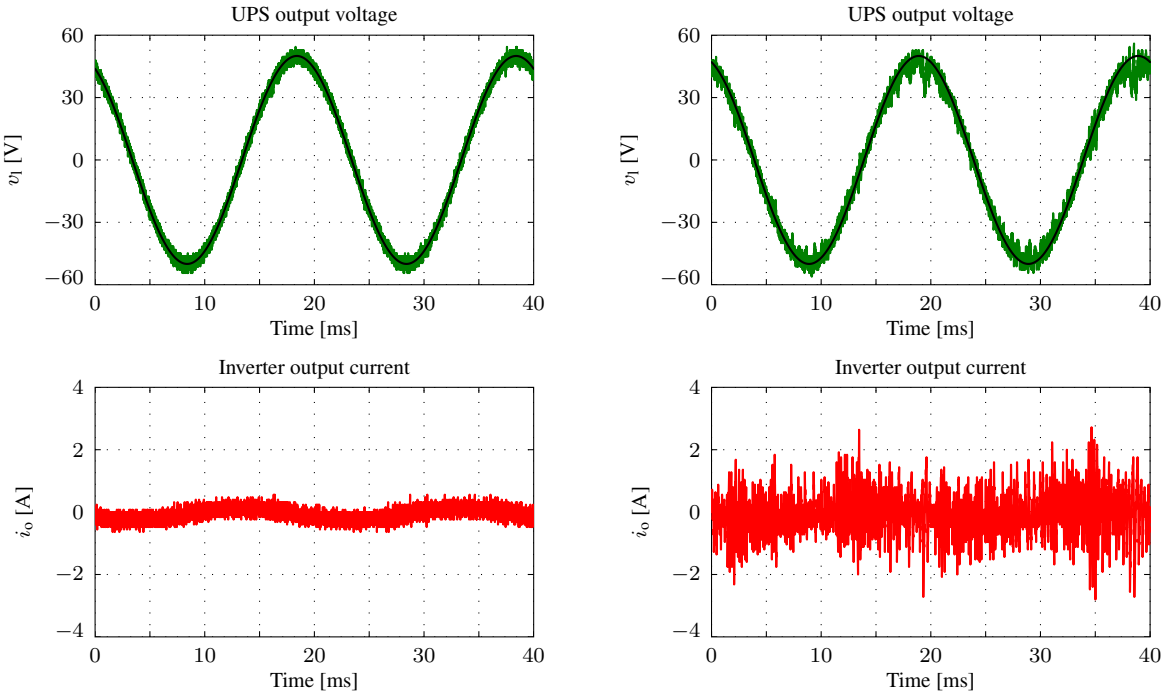
Figure 6.5: Current control of a resistive-inductive load

if a sinusoidal current reference with an amplitude of $2\,\mathrm{A}$ and a frequency of $50\,\mathrm{Hz}$ is commanded. It is clearly visible that the oversampling strategy produces much better currents with less ripples compared to normal FS-MPC.

Further experiments were conducted in order to prove that the proposed algorithm is also able perform control of a UPS application. The experiments were conducted with $V_{\mathrm{dc}} = 60\,\mathrm{V}$, the parameters of the used LC lowpass-filter were $C = 10\,\mu\mathrm{F}$ and $L = 3\,\mathrm{mH}$. For the voltage reference a sine with an amplitude of $50\,\mathrm{V}$ and $50\,\mathrm{Hz}$ was used.

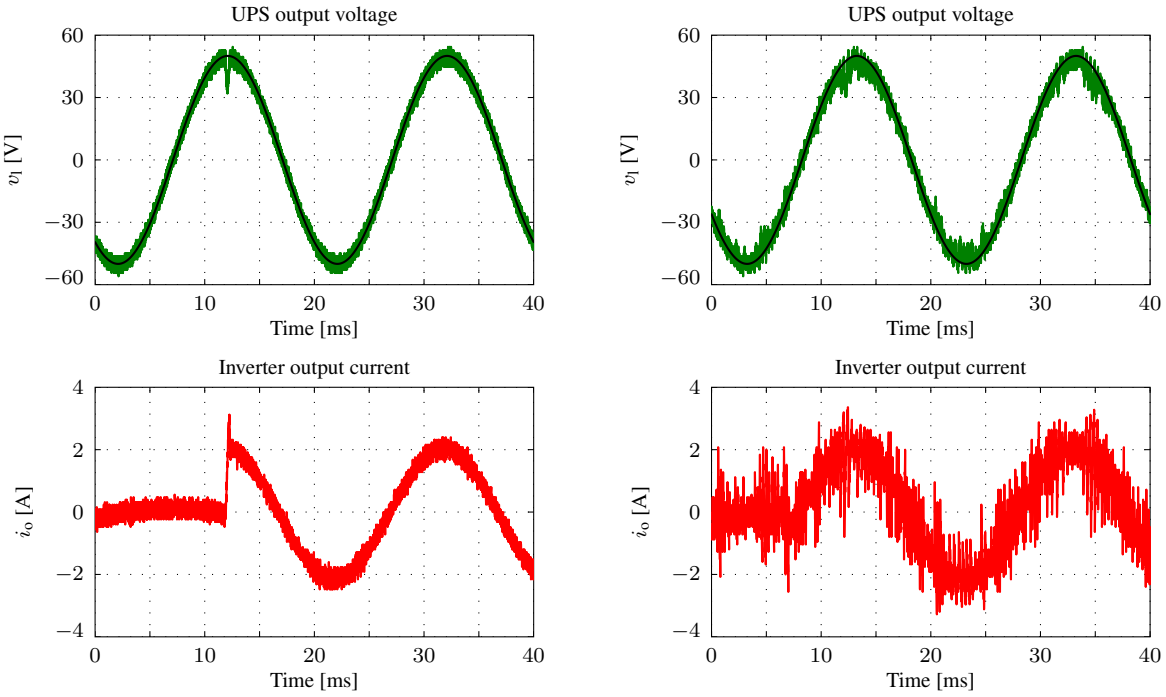
Figure 6.6 shows a comparison of the results with and without oversampling in idle operation, i.e. no load was connected to the LC filter. In the upper graphs the UPS output voltage can be seen, below the recorded inverter output current is shown. Of course, since the LC filter is a second-order filter, the difference in the output voltage regarding ripples is not that high, but even in this case an improvement is still clearly noticeable. Again, the biggest difference can be seen in the inverter output current: If the oversampling method is used, the current ripples are tremendously reduced. In idle operation only a very small current flow can be seen (reactive current for charging and discharging the inductor and capacitor).

Another experiment was made in order to prove that the proposed algorithm also works during transients, i.e. when a load is connected to the output of the UPS. In Figure 6.7 the results for a resistive load step are presented. Once again, as already mentioned for the previous experiment, the current ripple can be significantly reduced but also in the output voltage an improvement is clearly visible. Only during the load step a small voltage drop can be seen which is due to the inverter output current limitation of $3\,\mathrm{A}$.



(a) With oversampling (b) Without oversampling

Figure 6.6: UPS application, without load



(a) With oversampling (b) Without oversampling

Figure 6.7: UPS application, resistive load step

The last experiments shown for the three-level NPC inverter were made in order to investigate if it is also possible to operate the UPS when a highly nonlinear load is connected to its clamps. As for the previously shown results, the DC link voltage was set to $V_{\mathrm{dc}} = 60\,\mathrm{V}$ and the LC filter parameters were $C = 10\,\mu\mathrm{F}$ and $L = 3\,\mathrm{mH}$. The inverter output current was again limited to 3 A. A typical and highly nonlinear load is a B2 rectifier with a subsequent filter capacitor (in this case $C_{\mathrm{l}} = 470\,\mu\mathrm{F}$) for the DC link stabilization. Furthermore, a resistor with $R_{\mathrm{l}} = 150\,\Omega$ was connected to it. Such a load leads to high current spikes when the capacitor is charged. The circuit diagram is shown in Figure 6.8.

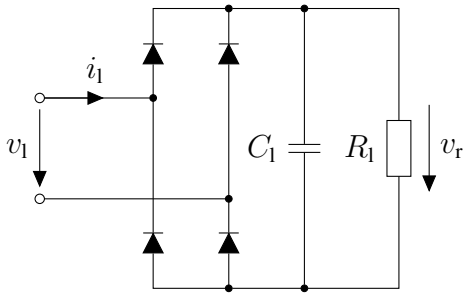
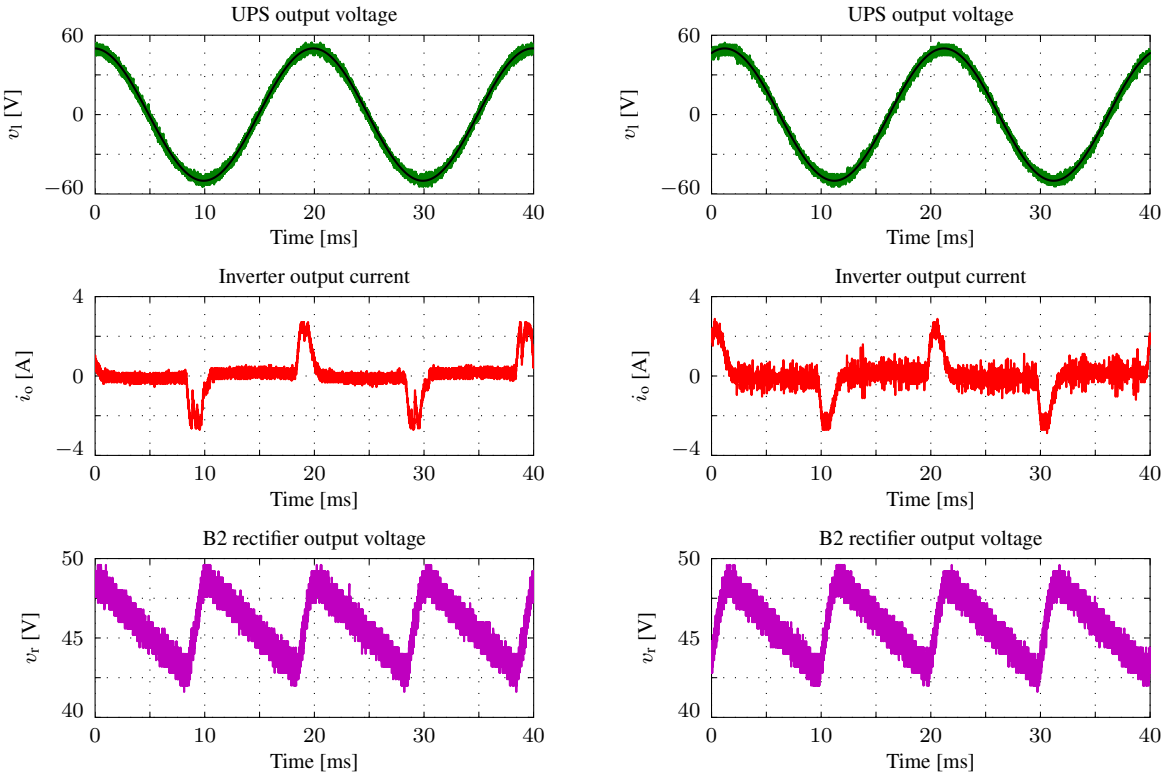


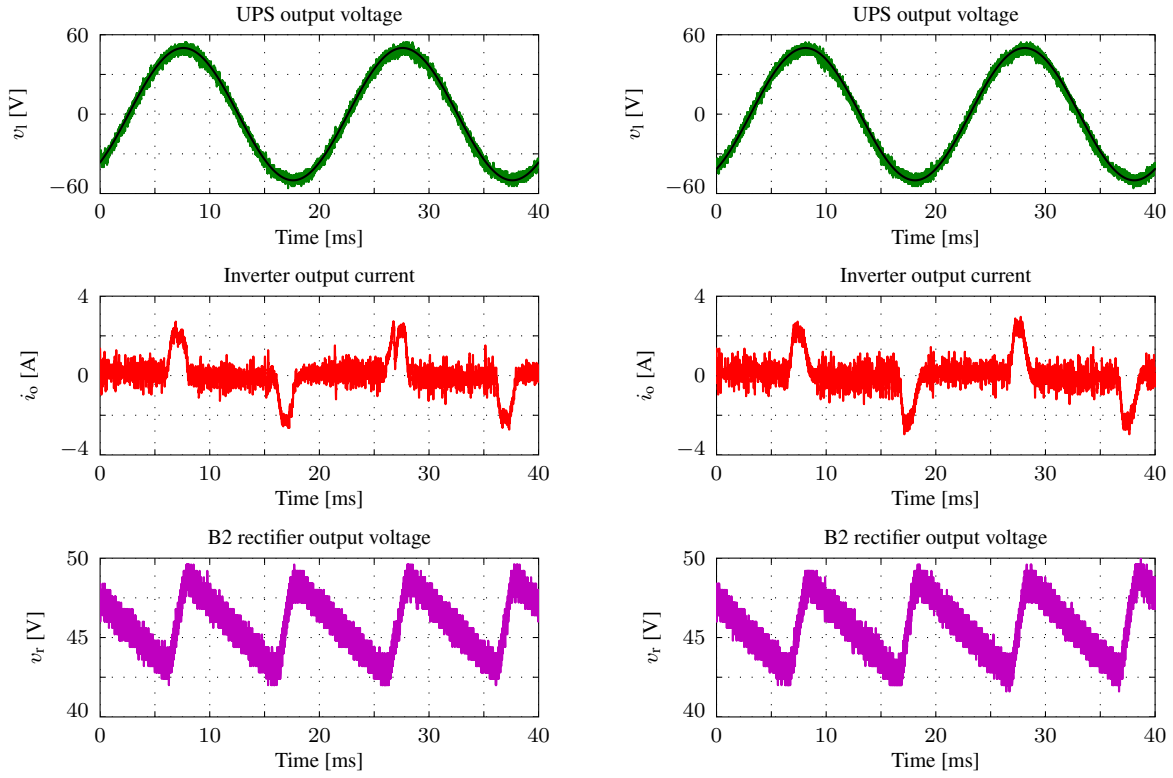
Figure 6.8: Typical nonlinear load: B2 rectifier with resistive-capacitive load



(a) With oversampling

(b) Without oversampling

Figure 6.9: UPS application, B2 rectifier with RC load, load current i_{l} measured



(a) With oversampling

(b) Without oversampling

Figure 6.10: UPS application, B2 rectifier with RC load, load current i_l estimated

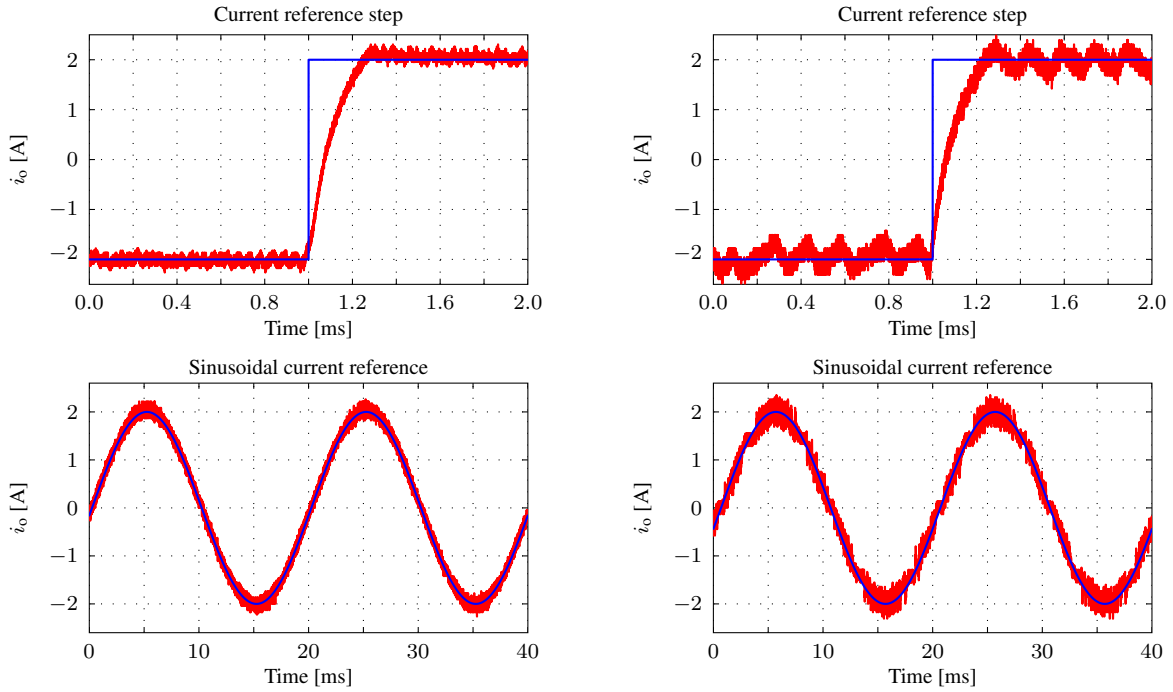
As already mentioned in chapter 6.3.2, it is also possible to estimate the LC filter output current i_l instead of measuring it. This is very interesting since less measurements lead to reduced costs and to a cheaper system. Figure 6.9 shows the control results with and without oversampling when the load current i_l is measured, whereas in Figure 6.10 the results for an estimated load current can be seen. The plots show the UPS output voltage v_l, the inverter output current i_o and the voltage drop v_r across the load resistor R_l.

A B2 rectifier normally leads to high current spikes when the capacitor is charged. However, since the inverter output current was limited to 3 A, these spikes can be reduced to the maximum allowed current. It is clearly visible that the current limitation works well for all four cases that are shown.

Considering the case with measured load currents, the inverter output current ripple can again be significantly reduced with the proposed oversampling strategy. However, if i_l is estimated with the proposed method, it can be seen that only a small improvement in terms of ripple is possible when oversampling is used. This is due to the fact that for the voltage measurements isolation amplifiers with a low bandwidth of only 20 kHz were used and that these amplifiers show a considerable ripple in their output. If better voltage measurements are used, this disadvantage could be overcome. Furthermore, if a more sophisticated output current estimator is used (e.g. a Kalman filter [61, 62]), the control result is also expected to become better.

6.4.2 Three-level Flying Capacitor inverter

Finally, experiments were conducted with the inverter in FC configuration. Figure 6.11 shows the results for current control of a resistive-inductive load. In this case the DC link voltage was set to $40\,\mathrm{V}$. The upper plots show a current reference step from $-2\,\mathrm{A}$ to $2\,\mathrm{A}$ at $1\,\mathrm{ms}$. Again, it is clearly visible that the proposed Oversampling FS-MPC method does not deteriorate the dynamic performance but the current ripple is significantly reduced. The other two plots show the steady-state control result for a sinusoidal current reference with an amplitude of $2\,\mathrm{A}$ and a frequency of $50\,\mathrm{Hz}$. In this case a huge improvement of the control result can be seen as well. The current ripple is smaller than the one in Figure 6.5 but this is due to the fact that a lower DC link voltage ($40\,\mathrm{V}$ instead of $60\,\mathrm{V}$) was used.



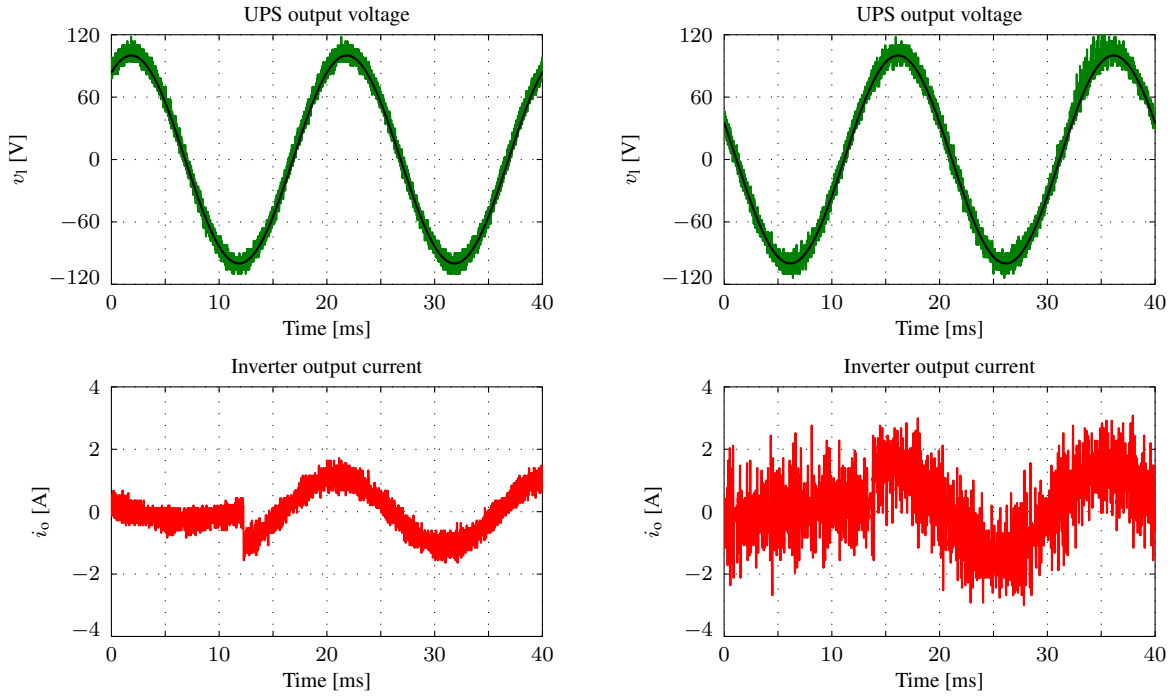
(a) With oversampling

(b) Without oversampling

Figure 6.11: Current control of a resistive-inductive load

The last experiment for the FC configuration shows a resistive load step applied to the LC filter (UPS application). In this case the DC link voltage was set to $V_{\mathrm{dc}} = 150\,\mathrm{V}$. The LC filter output voltage reference is a sine with an amplitude of $100\,\mathrm{V}$ and a frequency of $50\,\mathrm{Hz}$. As in the previous cases, the LC filter parameters were $C = 10\,\mu\mathrm{s}$ and $L = 3\,\mathrm{mH}$ and the inverter output current was limited to $3\,\mathrm{A}$.

The experimental results are shown in Figure 6.12. As for the NPC inverter, the proposed Oversampling FS-MPC method shows a huge improvement regarding the inverter output current ripple. Although the LC filter output voltage v_{l} is already quite good for normal FS-MPC, the oversampling method still leads to slight improvements of its quality.



(a) With oversampling

(b) Without oversampling

Figure 6.12: UPS application, resistive load step

6.5 Evaluation

The shown experimental results clearly verify that the proposed Oversampling FS-MPC method can effectively reduce ripples on the controlled variables. Especially for systems where a higher switching frequency (in the range of several kHz) can be used (usually for smaller-power systems), the shown strategy can significantly improve the control result without deteriorating the dynamic performance. The proposed strategy to take also switchings into account which have happened in the previous samples and to penalize switchings of an IGBT more heavily if that IGBT has already switched before leads to the fact that still a prediction horizon of only one sample is enough to achieve a good control result, i.e. the applied strategy is able to distribute the switchings more evenly as if only hard constraints are used.

For the UPS application the ripple reduction on the controlled output voltage v_l is not very high. This, however, is due to the LC lowpass-filter. If the proposed strategy is used, it is possible to reduce the size of the LC filter and thus also to reduce costs. Furthermore, due to the simple one-step prediction a quick implementation in hardware is easily possible.

CHAPTER 7

Conclusion

7.1 Summary

The goals of this work were

1. to find solutions to reduce the calculation effort for FS-MPC methods,

2. to increase the time resolution of FS-MPC methods in order to reduce ripples on the controlled variables and

3. a combination of the two last points, i.e. to find methods to reduce ripples on the controlled variables with less calculation effort.

Within this work solutions for all three items have been proposed and were proven experimentally.

It was also verified that FS-MPC methods offer several advantages over conventional PID controllers:

1. Multivariable control is easily possible (control of two currents, both flux and torque, and also the voltage balancing can be performed by one single FS-MPC controller).

2. Constraints can be considered without problems and nonlinearities can also be included.

3. FS-MPC controllers can easily operate the system at its physical limits. Conventional controllers mostly need additional (adaptive) schemes and feed forward controllers to achieve the same or similar dynamics.

4. FS-MPC controllers do normally not produce an overshoot which is usual for conventional controllers.

7.2 Final evaluation

As already mentioned, the applicability of direct switching strategies is highly dependent on the power range of the system: For medium- and high-voltage systems the system losses are dominated by the inverter switching losses. In this case switching frequencies of only a few hundred Hz per device are desired. For that reason industrial applications of FS-MPC methods have been reported mainly for high-power systems (MPDTC which was developed by ABB). Compared to classical DTC, MPDTC can lead to a further reduction of the switching frequency while maintaining at the same time the same quality of the control result. Sophisticated FS-MPC methods can partly even outperform Optimized Pulse Patterns for these types of drive systems [63].

In contrast to this, the presented work deals with low-voltage and smaller systems which are in the range of a few kW. For these applications a good quality of the controlled variables is usually much more important than a low switching frequency as in this case the inverter losses are less dominant. For these applications switching frequencies of 10–$20\,\mathrm{kHz}$ per device can be easily handled. The conventional FS-MPC approach only allows to change an inverter switching state at the *beginning* of a sample which is the reason for undesired high ripples. Another very important drawback of FS-MPC is the high calculation effort which rises exponentially with the prediction horizon. Thus, in this work several extensions to FS-MPC in order to reduce the calculation effort and to reduce ripples on the controlled variables were presented. As the shown experimental results clearly verify, the proposed extensions can effectively reduce these two drawbacks of FS-MPC methods. These extensions could even be successfully implemented for more sophisticated inverter topologies (three-level NPC and FC) where several tasks have to be performed by one single FS-MPC (e.g. control of two currents and three FC voltages). Even despite the high number of possible switching states (27 for an NPC and 64 for an FC inverter), up to three prediction steps could be realized in real-time with sampling rates up to $16\,\mathrm{kHz}$. For the proposed methods only one or two weighting factors have to be tuned (if any at all). Compared to linear controllers where parameter tuning is a work-intensive and crucial task, the proposed algorithms just need to be implemented and the weighting factors can be tuned quickly.

Although the methods presented within this work can enable FS-MPC strategies to become more attractive also for smaller and low-power (drive) systems, it is still questionable whether FS-MPC can outperform PWM-based MPC methods: For continuous-valued optimization tasks and linear systems the optimization problem can be solved analytically (e.g. with the MPT toolbox) which drastically reduces the calculation effort. PWM distributes the switching time points over the whole sample which leads to excellent control results in terms of ripples. Compared to the calculation of a VSP or to the implementation of an oversampled FS-MPC in hardware, the basic idea of PWM is ingeniously simple and has been proven to work well within the last decades. For multilevel inverters it is also possible to include a voltage balancing algorithm into the PWM which means that the overlaying controller only needs to calculate the voltages which should be applied—then it is not necessary to handle the voltage balancing within the control algorithm itself. Another drawback of FS-MPC is the varying switching frequency: Compared to PWM, FS-MPC methods produce an undesired audible noise which is much more annoying than the sound of PWM. Of course, it is also possible to modify the cost function such that a more or less constant switching frequency per device can be obtained. However, this can only be achieved at the expense of a deteriorated result regarding the main control objective (to mini-

mize the control deviation). Thus, in order to achieve the same control result in terms of ripples as without forcing a constant switching frequency, the sampling frequency and with it the time resolution of FS-MPC has to be drastically increased.

7.3 Outlook

There are several possibilities to extend and to modify the strategies which were presented in this work: One promising extension could be a method to calculate not only one but two or even more VSPs. If e.g. only one IGBT is allowed to switch at a time and if two VSPs are calculated within one sample, "online optimized" pulse patterns and a constant switching frequency could be obtained. Such an FS-MPC method would then be fully comparable to PWM in terms of ripples on the controlled variables. Another possibility would also be to increase the prediction horizon for VSP methods.

Another very promising application for (FS-)MPC is to perform direct speed or even position control for electrical drives. In this way all disadvantages which result from cascaded control loops could be overcome. Furthermore, it would then also be possible to operate the drive at its physical limits while still keeping all controlled variables within their allowed range.

APPENDIX A

List of symbols and abbreviations

A.1 List of symbols

General remark:
The following convention was used for variables:

Scalars are italic letters:	x
Vectors are bold lower case letters:	$\boldsymbol{x}$
Matrices are bold upper case letters:	$\boldsymbol{X}$
References are marked with a star superscript:	x^*

Used symbols:
In the following the most important symbols are listed which are used within this work.

General symbols:

$\boldsymbol{x}$	State vector
$\boldsymbol{u}$	Input vector
$\boldsymbol{y}$	Output vector
$\boldsymbol{A}$	State matrix
$\boldsymbol{B}$	Input matrix
$\boldsymbol{C}$	Output matrix
$\boldsymbol{D}$	Feedthrough matrix
t	Time (continuous)
k	Time (discrete, current sample)
$\frac{\mathrm{d}}{\mathrm{d}t}$	Time derivation
T_s	Sampling time
t_sw	Variable switching time point (VSP)
Δ	Difference
J	Inertia

General electrical variables:

a, b, c	Phases
α, β	Equivalent two-phase coordinates
j	$\sqrt{-1}$
v	Voltage
i	Current
R	Resistor
C	Capacitor
L	Inductor

Induction machine parameters:

$\boldsymbol{v}_\mathrm{s}$, $\boldsymbol{v}_\mathrm{r}$	Stator and rotor voltage
$\boldsymbol{i}_\mathrm{s}$, $\boldsymbol{i}_\mathrm{r}$	Stator and rotor current
$\boldsymbol{\psi}_\mathrm{s}$, $\boldsymbol{\psi}_\mathrm{r}$	Stator and rotor flux
ω_m	Mechanical machine speed
ω_el	Electrical machine speed
T_m	Mechanical machine torque
T_l	Mechanical load torque
P	Machine power
p	Number of pole pairs
R_s, R_r	Stator and rotor resistance
L_s, L_r	Stator and rotor inductance
L_m	Mutual inductance

Further variables and parameters:

S_{xi}	Switch i in phase x
s_{xi}	Gating signal for switch i in phase x
j	Cost function value
w	Weighting factor
v_o, i_o	Inverter output voltage and current (UPS)
v_l, i_l	LC lowpass-filter output voltage and current (UPS)

A.2 List of abbreviations

AC	Alternating Current
AD	Analog to Digital (converter)
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
DA	Digital to Analog (converter)
DC	Direct Current
DMTC	Direct Mean Torque Control
DSC	Direct Self Control
DSP	Digital Signal Processor
DTC	Direct Torque Control
EMI	Electromagnetic Interference
FC	Flying Capacitor
FIFO	First In First Out (buffer)
FOC	Field Oriented Control
FPGA	Field Programmable Gate Array
FS	Finite-Set
FS-MPC	Finite-Set Model Predictive Control
GPC	Generalized Predictive Control
HDL	Hardware Description Language
IGBT	Insulated Gate Bipolar Transistor
IM	Induction Machine, Induction Motor
ISA	Industry Standard Architecture (bus)
LCD	Liquid Crystal Display
LP	Linear Program
LTI	Linear Time-Invariant
MILP	Mixed Integer Linear Program
MIQP	Mixed Integer Quadratic Program
MPC	Model Predictive Control
MPDTC	Model Predictive Direct Torque Control
mpLP	Multiparametric Linear Program
mpQP	Multiparametric Quadratic Program
MPT	Multiparametric Toolbox
NP	Neutral Point
NPC	Neutral Point Clamped
PCC	Predictive Current Control
PTC	Predictive Torque Control
PWM	Pulse Width Modulation
QP	Quadratic Program
RAM	Random Access Memory
RMS	Root Mean Square
RPM	Revolutions Per Minute
RTAI	Real-Time Application Interface

SI	International System of Units
SVM	Space Vector Modulation
THD	Total Harmonic Distortion
UPS	Uninterruptible Power Supply
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VSP	Variable Switching Point
VSP2CC	Variable Switching Point Predictive Current Control
VSP2TC	Variable Switching Point Predictive Torque Control

APPENDIX B

Test bench data

B.1 Two-level inverter test bench

A quick overview of the two-level inverter test bench has already been given in chapter 3.1.2. The complete test bench consists of a real-time computer system, two squirrel-cage induction motors, two two-level inverters and measurement devices. It is to be noted that for this test bench no DC link voltage measurement is possible. A picture of the test bench can be seen in Figure 3.4.

B.1.1 Real-time computer system

The real-time computer system consists of a PC104 module with a 1.4 GHz Pentium M CPU, 1 GB RAM and a 60 GB hard disk. All components are mounted into a 19 inch rack. The system is running an Arch Linux distribution with an RTAI (real-time application interface) kernel patch. This RTAI kernel patch allows to program kernel modules which can be executed in real-time. The real-time control algorithm can be conveniently programmed in *C*.

The necessary peripheral hardware for analog and digital in- and outputs is connected via the 16 bit ISA bus. The used 19 inch rack has space for up to twelve extension boards. In order to measure the signal from the current transducers, an AD card with two channels is used. The encoder signal can be read via a special encoder board. The most important extension card is responsible for the inverter gating (PWM) signals: In order to synchronize the generation of these gating signals with the control algorithm which is running on the real-time computer, this card also generates an interrupt for the real-time computer. Every time when such an interrupt occurs, the control algorithm is executed. Thus, the whole control algorithm is triggered by this extension card. As this board contains an FPGA, it is possible to modify the existing implementations for the generation of the gating signals according to the user's needs. The current implementation allows of course the generation of PWM signals but direct switching and switching at a VSP is also possible. Furthermore, the interrupt for the control algorithm also

triggers the measurements of the AD converters such that it is possible to trigger the measurements at the beginning of a sample. In order to conveniently output measured values, DA cards can be inserted into the system as well: Then, variables can be easily visualized and recorded with an oscilloscope. Another extension board with a four digit hexadecimal (16 bit) display and four hexadecimal switches can be used for status notifications and user interaction (start and stop of the control algorithm, reference value changes etc.). Further information about this system can be found in [37].

B.1.2 Inverters

As already mentioned, the test bench consists of two inverters. Both inverters are supplied from a three-phase voltage source with an RMS phase to phase voltage of $400\,\mathrm{V}$. Since the inverters cannot feed back energy to the three-phase grid, the DC link voltage will rise if a connected machine is operated in generator mode. In order to avoid damages of the system, a break chopper resistor can be connected to discharge the DC link such that the voltage level does not become critical. Since both machines are connected to each other, one drive is normally operated as motor while the other one works as generator. Thus, in order to avoid a frequent use of the break chopper resistor, the DC links of both drives are coupled together.

The controlled inverter is a modified Seidel/Kollmorgen Servostar 600 $14\,\mathrm{kVA}$ inverter. It allows the user to directly command the IGBT gating signals from the real-time computer system. This inverter is connected to the working machine which is also controlled by the user. Consequently, the load inverter (Danfoss VLT FC-302 $3.0\,\mathrm{kW}$) is connected to the load machine. This inverter allows to perform speed and torque control of different machines. Furthermore, it can also be used to measure machine parameters.

B.1.3 Induction machines

Table B.1: Parameters for the working machine of the two-level inverter test bench

Parameter	Value
Nominal power P_{nom}	$2.2\,\mathrm{kW}$
Synchronous frequency f_{syn}	$50\,\mathrm{Hz}$
Nominal current $\lvert\boldsymbol{i}_{\mathrm{s,\,nom}}\rvert$	$8.02\,\mathrm{A}$
Power factor $\cos(\varphi)$	0.85
Nominal speed ω_{nom}	$2772\,\mathrm{rpm}$
Number of pole pairs p	1
Stator resistance R_{s}	$2.6827\,\Omega$
Rotor resistance R_{r}	$2.1290\,\Omega$
Stator inductance L_{s}	$283.4\,\mathrm{mH}$
Rotor inductance L_{r}	$283.4\,\mathrm{mH}$
Mutual inductance L_{m}	$275.1\,\mathrm{mH}$
Inertia J	$0.005\,\mathrm{kg\,m^2}$

The two-level inverter test bench consists of two $2.2\,\mathrm{kW}$ squirrel-cage induction machines which are coupled to each other. The parameters of the working machine (driven by the controlled inverter) are given in Table B.1. The load machine is completeley operated by the load inverter and hence, its parameters are not shown. The parameters were measured with the Danfoss load inverter. On both machines incremental encoders with 1024 points are mounted.

B.2 Three-level inverter test bench

In chapter 3.1.3 the three-level inverter test bench has already been introduced. It consists of a significantly improved version of the real-time computer system that is used for the two-level inverter test bench, one induction motor ($2.2\,\mathrm{kW}$) and a self-developed three-level inverter which can be used either in NPC or in FC configuration. The DC link is powered by a $3\,\mathrm{kW}$ DC power supply. This test bench also allows to measure the DC link voltages.

B.2.1 Real-time computer system

As already mentioned, the real-time computer system which is used for this test stand is a significantly improved version of the one that is used for the two-level inverter test bench. It consists of a PICMG 1.0 mainboard from Advantech which is equipped with $2\,\mathrm{GB}$ RAM and an Intel Pentium 4 CPU with $3.4\,\mathrm{GHz}$ and a $500\,\mathrm{GB}$ hard disk. The system is running a Debian Linux distribution with an RTAI kernel patch such that the control algorithms can be executed as kernel modules in real-time.

The PICMG 1.0 mainboard is connected to the same FPGA board which is described in appendix B.3 via the 16 bit ISA bus. The LCD which is connected to the FPGA board is used to display status notifications and error messages. In this case the FPGA board is used for the AD conversions of the measurements. Furthermore, an extension board for reading the measured encoder angle and for performing the DA outputs is connected to it. The firmware for the control system (communication with the AD and DA converters, for reading the encoder angle, IGBT gating signals etc.) is implemented in VHDL on the FPGA. Furthermore, additional safety functions for the inverter (overvoltage and overcurrent protection) are also implemented on the FPGA. If the control algorithm which is running on the real-time computer system is faulty, the execution of the program is automatically stopped by the safety routines in the FPGA. Thus, a safe operation of the test bench can be ensured in nearly all cases.

The basic operation of this improved real-time system is the same as for the other one which is used for the two-level inverter: The real-time algorithms are programmed in *C* as Linux kernel modules which are triggered by an external interrupt from the FPGA. One highlight of this system is that it can also monitor the real-time condition: For a successful real-time implementation it is necessary that the control algorithm is finished before the next interrupt signal is detected. This can be easily realized: The FPGA starts a counter when it has set an interrupt. Then, the computer will execute its control algorithm. When the algorithm is finished, the computer sends an "acknowledge signal" to the FPGA via the ISA bus. If the FPGA does not receive this signal before it starts the next interrupt, the real-time condition is violated and then the control algorithm is stopped by the FPGA. This principle is also used to monitor the "real-time load" which is displayed in percent in the LCD status display. In this way the effect

of code modifications on the algorithm's execution speed can be easily monitored by the user.

Furthermore, this real-time computer system has optical outputs for the inverter gating signals and uses analog differential measurements in order to reduce sensitivity to electromagnetic interference (EMI) and to provide better results.

Further information about this system can be found in [38].

B.2.2 Three-level inverter

The three-level inverter consists of three phase legs. In every leg an SK20MLI066 IGBT module from Semikron is used. These modules consist of four IGBTs with freewheeling diodes in a row. Furthermore, the two NPC diodes which are necessary for NPC operation are also included within this module. Since the FC voltages are always positive, it is also possible to use these modules for an FC inverter. The IGBT modules are rated for $20\,\mathrm{A}$. The complete DC link capacitance is $1700\,\mu\mathrm{F}$. Every flying capacitor has a capacitance of $440\,\mu\mathrm{F}$.

On top of the IGBT modules gate driver boards are mounted which provide desaturation monitoring and undervoltage lockout. A $12\,\mathrm{V}$ DC power supply is used for the optical interface for the gating signals and to provide auxiliary voltages for the gate drivers.

B.2.3 Induction machine

Table B.2: Parameters for the induction machine of the three-level inverter test bench

Parameter	Value
Nominal power P_{nom}	$2.2\,\mathrm{kW}$
Synchronous frequency f_{syn}	$50\,\mathrm{Hz}$
Nominal current $\lvert\boldsymbol{i}_{\mathrm{s,\,nom}}\rvert$	$8.5\,\mathrm{A}$
Power factor $\cos(\varphi)$	0.86
Nominal speed ω_{nom}	$2830\,\mathrm{rpm}$
Number of pole pairs p	1
Stator resistance R_{s}	$2.1294\,\Omega$
Rotor resistance R_{r}	$2.2773\,\Omega$
Stator inductance L_{s}	$350.47\,\mathrm{mH}$
Rotor inductance L_{r}	$350.47\,\mathrm{mH}$
Mutual inductance L_{m}	$340.42\,\mathrm{mH}$
Inertia J	$0.002\,\mathrm{kg\,m^2}$

The three-level inverter test bench uses of a $2.2\,\mathrm{kW}$ squirrel-cage induction motor. Its parameters are shown in Table B.2. The machine parameters were measured with the Danfoss load inverter of the two-level inverter test bench. An incremental encoder with 1024 points for position and speed measurements is mounted to this machine.

B.3 FPGA-based test bench

In chapter 3.2 a quick overview of the FPGA-based test bench has already been given. It consists of the FPGA board which is shown in Figure 3.6. The FPGA board is connected to an optics board which allows to transmit the IGBT gating signals optically to the inverter. Furthermore, one current measurement board and one board for voltage measurements are connected to the FPGA board. The two-phase three-level inverter just consists of two phase legs.

B.3.1 FPGA board

The FPGA board which is shown in Figure 3.6 is used for the real-time computer system for the three-level inverter test bench. As the board uses an Altera Cyclone III FPGA with $40{,}000$ logic elements, it is also possible to directly implement control algorithms on the FPGA which is clocked with 20 MHz. The board also has a very fast 12 bit AD converter. It allows to measure all eight different channels simultaneously with up to 65 megasamples per second. Because of this it is also possible to implement highly oversampled control algorithms and safety routines. The measurement boards can be connected to the FPGA board with RJ45 plugs. In order to deliver good measurement results and in order to have less EMI sensitivity, analog differential signalling is used for the measurements.

B.3.2 Two-leg three-level inverter

As already mentioned, the two-leg three-level inverter uses the same design as the version with three phase legs. Another difference is that in this case the complete DC link capacitance is 500 µF and the two flying capacitors both have a size of 500 µF, too. For this inverter also a 12 V power supply is used for the optical interface for the gating signals and to provide auxiliary voltages for the gate drivers.

B.3.3 Loads

As mentioned in chapter 6.4, experimental results were conducted with a resistive-inductive load and for a UPS application. The loads were simply made of discrete components (resistors, inductors, capacitors and diodes for the nonlinear load in UPS configuration).

List of Figures

List of Tables

Bibliography

[1] A. Linder, R. Kanchan, P. Stolze, and R. Kennel, *Model-Based Predictive Control of Electric Drives*. Göttingen: Cuvillier Verlag, 2010.

[2] K. Ogata, *Modern Control Engineering*, 5th ed. Prentice Hall, September 2009.

[3] ——, *Discrete-Time Control Systems*, 2nd ed. Prentice Hall, January 1995.

[4] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 2nd ed. Wiley, June 2008.

[5] E. Clarke, *Circuit Analysis of AC Power Systems*. J. Wiley & sons, 1943.

[6] R. H. Park, "Two-reaction Theory of Synchronous Machines – Generalized Method of Analysis – Part I," *Transactions of the American Institute of Electrical Engineers*, vol. 48, no. 3, pp. 716–727, July 1929.

[7] ——, "Two-reaction Theory of Synchronous Machines – II," *Transactions of the American Institute of Electrical Engineers*, vol. 52, no. 2, pp. 352–354, June 1933.

[8] A. Nabae, I. Takahashi, and H. Akagi, "A New Neutral-Point-Clamped PWM Inverter," *IEEE Transactions on Industry Applications*, vol. IA-17, no. 5, pp. 518–523, September 1981.

[9] T. Meynard and H. Foch, "Multi-Level Conversion: High Voltage Choppers and Voltage-Source Inverters," in *23rd Annual IEEE Power Electronics Specialists Conference (PESC '92)*, Toledo, Spain, June/July 1992, pp. 397–403.

[10] R. H. Wilkinson, T. A. Meynard, and H. du T. Mouton, "Natural Balance of Multicell Converters: The General Case," *IEEE Transactions on Power Electronics*, vol. 21, no. 6, pp. 1658–1666, November 2006.

[11] P. Kovács, *Transient Phenomena in Electrical Machines*. Elsevier, 1984.

[12] J. Holtz, "The Dynamic Representation of AC Drive Systems by Complex Signal Flow Graphs," in *1994 IEEE International Symposium on Industrial Electronics (ISIE '94)*, Santiago de Chile, Chile, May 1994, pp. 1–6.

[13] ——, "The Induction Motor – A Dynamic System," in *20th International Conference on Industrial Electronics, Control and Instrumentation (IECON '94)*, vol. 1, Bologna, Italy, September 1994, pp. 1–6.

[14] D. G. Holmes and T. A. Lipo, *Pulse Width Modulation for Power Converters: Principles and Practice (IEEE Press Series on Power Engineering)*, 1st ed. Wiley-IEEE Press, October 2003.

[15] W. Leonhard, *Control of Electrical Drives*, 3rd ed. Springer, September 2001.

[16] C. G. Verghese and S. R. Sanders, "Observers for Flux Estimation in Induction Machines," *IEEE Transactions on Power Electronics*, vol. 35, no. 1, pp. 85–94, February 1988.

[17] B. Bose, *Modern Power Electronics and AC Drives*, 1st ed. Prentice Hall, October 2001.

[18] I. Takahashi and T. Noguchi, "A New Quick-Response and High-Efficiency Control Strategy of an Induction Motor," *IEEE Transactions on Industry Applications*, vol. IA-22, no. 5, pp. 820–827, September 1986.

[19] M. Depenbrock, "Direct Self-Control (DSC) of Inverter-Fed Induction Machines," *IEEE Transactions on Power Electronics*, vol. 3, no. 4, pp. 420–429, October 1988.

[20] E. Flach, R. Hoffmann, and P. Mutschler, "Direct Mean Torque Control of an Induction Motor," in *European Conference on Power Electronics and Applications (EPE '97)*, vol. 7, Trondheim, Norway, September 1997, pp. 3672–3677.

[21] T. Geyer, G. Papafotiou, and M. Morari, "Model Predictive Direct Torque Control - Part I: Concept, Algorithm, and Analysis," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1894–1985, June 2009.

[22] G. Papafotiou, J. Kley, K. Papadopoulos, P. Bohren, and M. Morari, "Model Predictive Direct Torque Control - Part II: Implementation and Experimental Evaluation," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1906–1915, June 2009.

[23] R. Kennel and A. Linder, "Predictive Control of Inverter Supplied Electrical Drives," in *IEEE 31st Annual Power Electronics Specialists Conference (PESC '00)*, Galway, Ireland, June 2000, pp. 761–766.

[24] M. Morari and J. H. Lee, "Model Predictive Control: Past, Present and Future," *Computers and Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, May 1999.

[25] J. B. Rawlings, "Tutorial Overview of Model Predictive Control," *IEEE Control Systems Magazine*, vol. 20, no. 3, pp. 38–52, June 2000.

[26] J. M. Maciejowski, *Predictive Control With Constraints*. Prentice Hall, 2001.

[27] E. F. Camacho and C. Bordons, *Model Predictive Control (Advanced Textbooks in Control and Signal Processing)*, 2nd ed. Springer, February 2008.

[28] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, April 1965.

[29] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized Predictive Control - Part I. The Algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, March 1987.

[30] ——, "Generalized Predictive Control - Part II. Extensions and Interpretations," *Automatica*, vol. 23, no. 2, pp. 149–160, March 1987.

[31] R. J. Dakin, "A Tree-Search Algorithm for Mixed Integer Programming Problems," *The Computer Journal*, vol. 8, no. 3, pp. 250–255, January 1965.

[32] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*, 1st ed. Wiley-Interscience, November 1999.

[33] E. Pistikopoulos, M. Georgiadis, and V. Dua, Eds., *Process Systems Engineering: Volume 1: Multi-Parametric Programming*, 1st ed. Wiley-VCH, April 2007.

[34] E. Pistikopoulos, A. Galido, and V. Dua, Eds., *Process Systems Engineering, Volume 2: Multi-Parametric Model-Based Control*, 1st ed. Wiley-VCH, April 2007.

[35] M. Kvasnica, *Real-Time Model Predictive Control via Multi-Parametric Programming: Theory and Tools*. VDM Verlag, October 2009.

[36] P. Cortés, J. Rodríguez, C. Silva, and A. Flores, "Delay Compensation in Model Predictive Current Control of a Three-Phase Inverter," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 1323–1325, February 2012.

[37] N. A. Ameen, A. A. Naassani, and R. Kennel, "Design of a Digital System Dedicated for Electrical Drive Applications," *EPE Journal*, vol. 20, no. 4, pp. 37–44, December 2010.

[38] P. Stolze, J. Jung, W. Ebert, and R. Kennel, "FPGA-basiertes Echtzeitrechner-System mit RTAI-Linux für Antriebssysteme," in *SPS/IPC/Drives Congress*, Nuremberg, Germany, November 2013.

[39] P. Correa, M. Pacas, and J. Rodríguez, "A Predictive Torque Control for Inverter-fed Induction Machines," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 1073–1079, April 2007.

[40] H. Miranda, P. Cortés, J. I. Yuz, and J. Rodríguez, "Predictive Torque Control of Induction Machines Based on State-Space Models," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1916–1924, February 2009.

[41] P. Karamanakos, P. Stolze, R. Kennel, S. Manias, and T. Mouton, "Variable Switching Point Predictive Torque Control," in *2013 IEEE International Conference on Industrial Technology (ICIT)*, Cape Town, South Africa, February 2013, pp. 422–427.

[42] P. Stolze, F. Bauer, P. Landsmann, R. Kennel, and T. Mouton, "Predictive Torque Control of an Induction Machine Fed by a Neutral Point Clamped Inverter," in *2011 Workshop on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*, Munich, Germany, October 2011, pp. 24–29.

[43] P. Stolze, P. Karamanakos, R. Kennel, S. Manias, and T. Mouton, "Variable Switching Point Predictive Torque Control for Three-Level NPC Inverters," in *EPE 2013 ECCE Europe, 15th European Conference on Power Electronics and Applications*, Lille, France, September 2013, pp. 1–10.

[44] P. Stolze, M. Tomlinson, D. du Toit, R. Kennel, and T. Mouton, "Predictive Torque Control of an Induction Machine Fed by a Flying Capacitor Converter," in *Africon 2011*, Livingstone, Zambia, September 2011, pp. 1–6.

[45] P. Stolze, P. Landsmann, R. Kennel, and T. Mouton, "Finite-Set Model Predictive Control With Heuristic Voltage Vector Preselection for Higher Prediction Horizons," in *14th European Conference on Power Electronics and Applications (EPE 2011)*, Birmingham, UK, August/September 2011, pp. 1–9.

[46] ——, "Finite-Set Model Predictive Control of a Flying Capacitor Converter with Heuristic Voltage Vector Preselection," in *8th IEEE International Conference on Power Electronics and ECCE Asia (ICPE & ECCE)*, Jeju, South Korea, May/June 2011, pp. 210–217.

[47] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, July 2006.

[48] G. B. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton University Press, 1963.

[49] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, December 1984.

[50] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The Explicit Solution of Model Predictive Control via Multiparametric Quadratic Programming," in *American Control Conference (ACC 2000)*, vol. 2, Chicago, IL, USA, June 2000, pp. 872–876.

[51] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari, "Multi-Parametric Toolbox (MPT)," *Hybrid Systems: Lecture Notes in Computer Science*, vol. 2993, pp. 448–462, 2004.

[52] M. Kvasnica, P. Grieder, and M. Baotić, "Multi-Parametric Toolbox (MPT)," 2004. [Online]. Available: http://control.ee.ethz.ch/~mpt/

[53] T. Geyer, F. D. Torrisi, and M. Morari, "Optimal Complexity Reduction of Polyhedral Piecewise Affine Systems," *Automatica*, vol. 44, no. 7, pp. 1728–1740, July 2008.

[54] P. Tøndel, T. A. Johansen, and A. Bemporad, "Evaluation of Piecewise Affine Control via Binary Search Tree," *Automatica*, vol. 39, no. 5, pp. 945–950, May 2003.

[55] P. Stolze, M. Tomlinson, R. Kennel, and T. Mouton, "Heuristic Finite-Set Model Predictive Current Control for Induction Machines," in *2013 IEEE ECCE Asia Downunder (ECCE Asia)*, Melbourne, Australia, June 2013, pp. 1221–1226.

[56] P. Stolze, P. Karamanakos, M. Tomlinson, R. Kennel, T. Mouton, and S. Manias, "Heuristic Variable Switching Point Predictive Current Control for the Three-Level Neutral Point Clamped Inverter," in *IEEE International Symposium on Sensorless Control for*

Electrical Drives and Predictive Control of Electrical Drives and Power Electronics (SLED/PRECEDE 2013), Munich, Germany, October 2013.

[57] P. Landsmann, P. Stolze, and R. Kennel, "Optimal Switching Time Calculation in Predictive Torque Control," in *8th IEEE International Conference on Power Electronics and ECCE Asia (ICPE & ECCE)*, Jeju, Korea, May/June 2011, pp. 923–930.

[58] P. Landsmann and R. Kennel, "Saliency-Based Sensorless Predictive Torque Control with Reduced Torque Ripple," *IEEE Transactions on Power Electronics*, vol. 27, no. 10, pp. 4311–4320, October 2012.

[59] P. Stolze, M. Kramkowski, T. Mouton, M. Tomlinson, and R. Kennel, "Increasing the Performance of Finite-Set Model Predictive Control by Oversampling," in *2013 IEEE International Conference on Industrial Technology (ICIT)*, Cape Town, South Africa, February 2013, pp. 551–556.

[60] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, May 1974.

[61] B. D. O. Anderson and J. Moore, *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, 1979.

[62] P. Cortés, G. Ortiz, J. I. Yuz, J. Rodríguez, S. Vasquez, and L. G. Franquelo, "Model Predictive Control of an Inverter With Output LC Filter for UPS Applications," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1875–1883, June 2009.

[63] T. Geyer, "A Comparison of Control and Modulation Schemes for Medium-Voltage Drives: Emerging Predictive Control Concepts Versus PWM-Based Schemes," *IEEE Transactions on Industry Applications*, vol. 47, no. 3, pp. 1380–1389, May/June 2011.