Practical Implementation of Long-Horizon Direct Model Predictive Control

by

Martinus David Dorfling

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electrical) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof H. du T. Mouton

March 2018

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Tinus Dorfling

March 2018

Abstract

The use of model predictive control in power electronics has increased significantly in recent years. More specifically, the so-called direct model predictive control methods are primarily considered for power electronic converters due to their switching nature. In direct control methods the output of the controller directly manipulates the converter inputs, which are restricted to integers, without the use of a modulator. However, predominantly, only a short horizon of one prediction step is considered. This can be attributed to two reasons. Firstly, it has been previously regarded that longer horizons do not provide any performance benefits in power electronics. Secondly, the computational burden associated with prediction horizon increases exponentially, discouraging practical consideration.

Recently it was shown that the stigma that longer horizons do not provide performance benefits is false, and that long horizons do indeed increase the harmonic performance of a converter. In fact, if the prediction horizon is long enough, model predictive control can compete with the highly regarded optimised pulse patterns in terms of harmonic distortion. Furthermore, it was shown that the optimization problem of direct model predictive control with long horizons can be reformulated as an integer least-squares. A branch-and-bound method, known as sphere decoding, can solve the reformulated optimization problem in a time-efficient manner, enabling practical considerations.

The primary contribution of this thesis is the practical implementation of long-horizon direct model predictive control. A detailed description of the implementation of the controller within a field programmable gate-array is given. It is shown that, for almost $90\,\%$ of the cases, only $8.4\,\mu s$ are required to calculate the optimal inputs for a three-phase neutral-point-clamped inverter when using a prediction horizon of 5 with a sampling interval of $25\,\mu s$.

Continuing on the practical implementation of long-horizon direct model predictive control, experimental results are captured and analysed for prediction horizons one to five. The claim that longer horizons do provide a performance increase is validated through experimental results. A decrease of roughly $8.5\,\%$ in total total harmonic distortion at a switching frequency of $250\,\mathrm{Hz}$ is achieved when adopting a prediction horizon of five instead of one.

The secondary contribution of this thesis is the proposal of a method to selectively suppress selected harmonics. The formulation of the method is explained, and simulations are used to verify the suppression of harmonics.

Opsomming

Die toepassing van modelvoorspellendebeheer vir drywsingselektronika het aansienlik toegeneem in die afgelope paar jaar. Die sogenaamde direkte voorspellende beheer tegnieke is veral van toepassing tot die veld van drywingselektronika as gevolg van die skakelnatuur van die toerusting. 'n Modulator is afwesig wanneer direkte beheer metodes gebruik word, omdat die beheersein direk aan die intree van die omsetter gekoppel word. Die beheersein is daarom beperk tot heelgetalle. Oor die algemeen word 'n kort voorspellings horison van een gebruik. Daar is hoofsaaklik twee redes hiervoor. Eerstens, in die verlede was daar verneem dat langer horison geen voordelige bydra tot die optrede van 'n drywsingelektroniese toestel in hou nie. Tweedens, berekeninge wat verband hou met die voorspellings horison verhoog eksponensieel en ontmoedig daarom die gebruik daarvan.

Dit was redelik onlangs bewys dat langer horisonne wel 'n beduidende positiewe bydrae tot die werking van drywingselektroniese toestelle maak. Indien die horison lank genoeg is, kan voorspellende beheer redelik goed kompeteer met die hoog aangeskrewe optimalepulspatrone. Verder was dit bewys dat die direkte-voorspellendebeheer optimeringsprobleem herformuleer kan word as 'n heelgetal-kwadratiese optimeringsprobleem. 'n Metode wat bekend staan as sfeerdekodering kan gebruik word om die herformuleerde optimeringsprobleem effektief op te los, wat die praktiese gebruik daarvan bemoedig.

Die primêre bydrae van die tesis is die praktiese implementering van direkte modelvoorspellendebeheer met lang horisonne. 'n Gedetailleerde beskrywing om die beheerder binne 'n veldprogrammeerbare hekskikking te implementeer word aangebied. Dit word bewys dat in byna 90 % van die gevalle slegs 8.4 µs benodig word om die optimale intree te bereken vir 'n driefase neutralepuntgeklampte omsetter, wanneer 'n voorspellings horison van vyf gebruik word met 'n monsterperiode van 25 µs.

Verder word praktiese resultate aangebied vir voorspellendehorisonne vanaf een tot vyf. Dit word bewys dat langer horisonne verbetering rakende die harmoniese gedrag van drywingselektroniese toestelle teweeg bring. 'n Afname van 8.5 % in stroomharmoniesedistorsie teen 'n skakelfrekwensie van 250 Hz word waargeneem, wanneer 'n voorspellendehorison van vyf in plaas van een gebruik word.

Die sekondêre bydrae van die tesis is die bekendstelling van 'n metode om harmonieke selektief te onderdruk. Die formulering van die metode word verduidelik en simulasies word uitgevoer om te bevestig dat harmoieke wel onderdruk word.

Acknowledgements

First and foremost, I would like to thank Prof Toit Mouton for allowing me the opportunity to be his student. The knowledge and problem solving abilities that I gained are invaluable. Academics aside, I enjoyed the humorous meetings, croquettes, and ice cream.

Throughout my masters, I had the opportunity to work with Tobias Geyer and Petros Karamanakos, for which I am grateful. You both were always more than willing to answer questions and proofread my papers. Tobias, thank you for your wonderful and helpful textbook that guided me during the past two years. Petros, I appreciate you meticulously dissecting my papers.

Aanch, Adel, and Stef had the daunting task of proofreading this document. I am happy you are all still my friends (hopefully) after having to suffer through my verb agreement, and missing words that someone else must have removed before I sent the document. Thank you all.

As I am typing this sentence here in Switzerland, I have realised how easy and convenient my mom, Suna, has made the past few years (academic work aside). My meal selection has somehow reduced significantly, for some reason the dishes do not clean themselves, and my wardrobe is making clothes disappear instead of reappear. Baie dankie, Ma. I appreciate you for all of your hard work and motherly love.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ADC	analogue-to-digital converter
ALM	adaptive logic module
CB	carrier-based
CS	current sensor
DPC	direct power control
DSP	digital signal processing
DTC	direct torque control
FOC	field-orientated control
FPGA	field programmable gate-array
IGBT	insulated-gate bipolar transistor
ILS	integer least-squares
LTI	linear time-invariant
MIMO	multiple-input multiple-output
MPC	model predictive control
NP-hard	nondeterministic polynomial-time hard
NPC	neutral-point-clamped
OPP	optimised pulse pattern
PI	proportional-integral
PWA	piecewise affine

PWM	pulse-width modulation
QP	quadratic program
rms	root-mean-square
SDA	sphere decoding algorithm
SHE	selective harmonic elimination
SHS	selective harmonic suppression
SISO	single-input single-output
SVM	space-vector modulation
THD	total harmonic distortion
UART	universal asynchronous receiver/transmitter
VOC	voltage orientated control

Variables

$\boldsymbol{x} \in \mathbb{R}^n$	real-valued vector with dimension n
$\boldsymbol{M} \in \mathbb{R}^{n \times m}$	real-valued matrix with dimensions $n \times m$
$\mathcal{A}$	set

Symbols

$\mathbf{0}_{n \times m}$	zero matrix with dimensions $n \times m$
α	switching angle [rad]
λ_u	weighting factor on switching transitions
d	pulse number
f_{sw}	switching frequency [Hz]
$\boldsymbol{H}$	generator matrix
i, $\boldsymbol{i}$, I	ampere [A]
$\boldsymbol{I}_n$	identity matrix with dimension n
$\boldsymbol{K}$	Clarke transformation matrix
L	inductance [H]
m	modulation index

N_p	prediction horizon
n_x, n_y, n_u	size of state-, output-, and input variables
$\emptyset$	empty set
$\boldsymbol{Q}$	Hessian
R	resistance [Ω]
$\boldsymbol{R}$	penalty matrix
T_s	sampling interval [s]
u, $\boldsymbol{u}$	switch position (input variable)
$\Delta\boldsymbol{u}$	switching transition
$\boldsymbol{U}$	switching sequence over prediction horizon
$\mathcal{U}$	$\{-1, 0, 1\}$, set of admissible single-phase switch position
$\boldsymbol{\mathcal{U}}$	$\mathcal{U}^3$, set of admissible three-phase switch positions
$\mathbb{U}$	$\boldsymbol{\mathcal{U}}^{N_p}$, set of possible switching sequences over prediction horizon (feasible set of the optimization problem)
v, $\boldsymbol{v}$, V	voltage [V]
$\hat{x}_n$	amplitude of the n-th harmonic of x
$\boldsymbol{x}$	state variable
$\boldsymbol{y}$	output variable

Operators

$\dot{x}$	$\frac{dx}{dt}$, time derivative of x
$x \in \mathcal{X}$	x belongs to set $\mathcal{X}$
$\mathcal{A} \subseteq \mathcal{B}$	set $\mathcal{A}$ is a subset of (or included in) set $\mathcal{B}$
$\lfloor \boldsymbol{x} \rceil \in \mathcal{X}$	round x to nearest element in set $\mathcal{X}$
$\lVert \boldsymbol{x} \rVert_1$	$\sum_{i=1}^{n} \lvert x_i \rvert$, 1-norm (or Taxicab norm) of vector $\boldsymbol{x}$
$\lVert \boldsymbol{x} \rVert_2$	$\sqrt{\sum_{i=1}^{n} x_i^2}$, 2-norm (or Euclidean norm) of vector $\boldsymbol{x}$
$\lVert \boldsymbol{x} \rVert_{\boldsymbol{M}}^2$	$\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}$, 2-norm squared of vector $\boldsymbol{x}$ weighted with matrix $\boldsymbol{M}$

Clarification of Terms

prediction horizon	N_p denotes the *number* of predicted discrete-time steps. This forms the so-called prediction horizon with length N_pT_s in time, where T_s is the sampling interval, over which the prediction occurs. However, throughout this thesis the term *prediction horizon* will also be used when referring to N_p. The usage should be clear in context.

Chapter 1

Introduction

1.1 Background

In power electronics, there is a well-known fundamental trade-off between harmonic distortion and switching losses. If one of these two objectives is optimised, the other one will regress. Instead, both of these objectives should be marginally optimised as illustrated in Figure 1.1. Ideally, the optimal trade-off point should be as close to the origin as possible.

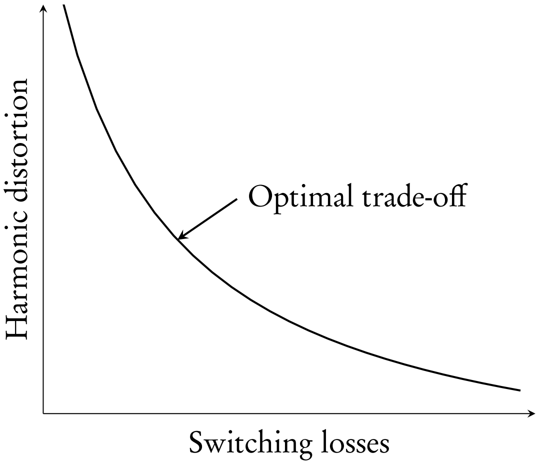


Figure 1.1: The fundamental trade-off. Replicated from [1].

Harmonic distortions are of great concern in electrical machines and grid-connected inverters. In machines, harmonic torque distortion leads to mechanical stress and wear on the shaft, while harmonic current distortion leads to iron and copper losses (thus leading to thermal losses) [1]. For grid-connected inverters, there are standards imposed on both the current and voltage harmonics. These standards are quite stringent, with certain harmonics, such as the even-order and third-order-odd harmonics, having extremely strict requirements [1].

In medium-voltage drives (i.e., power ratings above 1 MVA), the switching losses are of significant concern. Due to the high currents and voltages (in the range of kiloamperes and kilovolts), thyristors or gate-commuted thyristors are used. Due to their high-power applications, the switching frequency of these devices is limited to a few hundred hertz. Even at low switching frequencies, the switching losses are still significantly greater than the conduction losses [1].

Finding a control scheme that satisfies both of the above mentioned requirements, i.e., low harmonic distortion at low switching frequencies, can be challenging. Volts per hertz (V/Hz) control with optimised pulse patterns (OPPs) [2, 3] fulfils these requirements by having low harmonic distortion at low switching frequencies [1]. Unfortunately, this control scheme fails to satisfy another requirement: fast response time of the controller. A control scheme must be able to react quickly to any changes in operating set-points or faults, in other words, have a fast transient response. Field-orientated control (FOC) [4, 5] and voltage orientated control (VOC) [6], with space-vector modulation (SVM) [7], have fast response times but consequently high harmonic distortion at low switching frequencies [1]. Hysteresis control schemes, such as speed direct torque control (DTC) [8] and direct power control (DPC) [9], have high controller speeds, but unfortunately give rise to profound harmonic distortion [1].

Over the past few years, interest in model predictive control (MPC) within the power electronics community has increased significantly [10, 11, 12, 13, 14] due to the increased computational power of microprocessors and field programmable gate-arrays (FPGAs). As shown in Figure 1.2, MPC combines the best characteristics of the aforementioned control schemes (that is, fast transient response and low harmonic distortion) along with other advantages that will be mentioned.

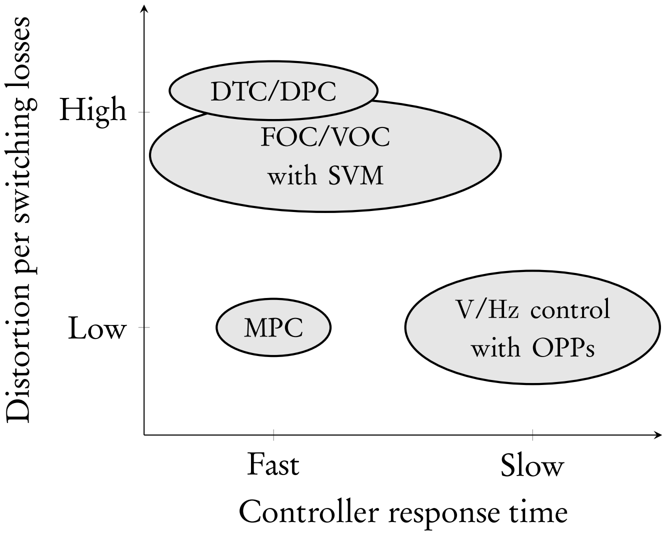


Figure 1.2: Comparison between control schemes. Replicated from [1].

To achieve harmonic distortion levels that can compete with OPPs, which is the modulation technique that offers the lowest harmonic distortion, the MPC controller must predict a fair amount of discrete-time steps into the future. This method of MPC is known as *long-horizon* (or *multi-step*) MPC. Unfortunately, the computational burden associated with the optimization problem underlying long-horizon MPC increases exponentially with the number of predicted discrete-time steps. Fortunately, a branch-and-bound method known as sphere decoding has been demonstrated to solve the optimization problem in a time efficient manner [15], enabling practical implementation of long horizons to be considered.

1.2 Thesis objectives

To date, papers that advocate for long horizons are based on simulation results. During simulations, the practical implementation of the controller and real-time requirements are ignored. The longest horizon found to be practically implemented in literature is a 2-step prediction in [16].

This thesis aims to achieve the following results. Firstly to execute the optimization problem of long horizons with short sampling intervals in real-time on an FPGA, without relying on heuristics (i.e., only concerned with *exact* solutions).

Secondly, MPC with long horizons should be practically implemented on a low-cost FPGA. Solving the optimization problem in real-time is only one half of the problem. Since FPGAs only have finite amount of resources, questions remain on whether long horizons can be implemented without severely sacrificing the computational performance, sampling interval, or achievable number of prediction steps.

Thirdly, the practical use of long horizons should offer a performance benefit over shorter horizons, even though non-ideal characteristics are present in practice (e.g., model uncertainties and saturation effects).

After investigating the practical implementation of long-horizon MPC, this thesis will propose a method to selectively suppress targeted harmonics present in the current. The theory of the method will be introduced and simulations will be used to verify that the method does indeed suppress harmonics.

To summarise, the objectives of this thesis are:

- Practically implement long-horizon MPC on a low-cost FPGA.
- With experimental results, validate that long horizons provide a performance benefit.
- Introduce a method for selective suppression of harmonics in the current spectrum.

1.3 Thesis summary

This thesis contains six chapters, with the following content.

Chapter 1: Introduction introduces the fundamental trade-off between harmonic distortion and switching losses in power electronics. Control schemes that are currently used in the industry, and their associated disadvantages, are mentioned. The concept of model predictive control with long horizons and the benefits thereof are stated, specifically with reference to low harmonic distortions with quick response time at low switching frequencies. The computational burden of the optimization problem underlying long-horizon MPC is mentioned to increase exponentially along with the number of prediction steps.

The objectives of this thesis are presented, of which the primary objective is to investigate the practical implementation of long-horizon MPC on an FPGA.

Chapter 2: Theoretical background lays the foundation of the theoretical knowledge required for this thesis. The chapter starts by introducing a multi-level inverter topology known as the neutral-point-clamped inverter. MPC is formally introduced and its advantages are discussed. This includes a description of the components that form MPC, namely, an internal dynamic model of a plant, addition of constraints, an optimal control problem, and a receding horizon policy. The reformulation of the original optimization problem, that makes use of an exhaustive search to find the minimum, to an integer least-squares problem is presented. The notation of sphere decoding is introduced, which is effective at solving the reformulated optimization problem underlying long-horizon MPC. The chapter concludes by reviewing optimised pulse patterns and space-vector modulation that are used as benchmarks during experimental evaluation. Relevant examples are given throughout this chapter.

Chapter 3: FPGA implementation discusses the implementation of long-horizon MPC on an FPGA. First, the chapter describes the modelling of a neutral-point-clamped inverter with a resistive-inductive load. The formulation of the optimal control problem is given. Some preliminary information on FPGAs are presented. The computational delay present in the practical controller is discussed and addressed. Thereafter, the implementation of the controller algorithm, including a reformulated sphere decoding algorithm that can be implemented within an FPGA, is explained. The resource usage of the FPGA for prediction steps 1 to 5 is shown. The chapter concludes with simulations that verify the implementation of the practical controller.

The computational burden of the sphere decoder is presented, which can solve the optimization problem for a 5-step prediction well within a sampling interval of 25 µs.

Chapter 4: Performance evaluation of long horizons presents the experimental results of long-horizon MPC. The testing framework for the practical evaluation is explained. The results for 1-step to 5-step predictions are given, with an emphasis on the trade-off between harmonic distortions and switching frequency.

It is shown that at a switching frequency of 250 Hz a prediction of 5 steps lowers the harmonic distortions by roughly 8.5 % when compared to a 1-step prediction. A comparison with optimised pulse patterns and space-vector modulation shows that MPC performs exceptionally well at low switching frequencies. Under the selected conditions for this thesis, MPC even outperformed the optimised pulse patterns. However, reasons regarding the apparent outperformance of optimised pulse patterns are given. The response time of MPC is demonstrated, showing a quick response when multiple reference changes are applied.

Chapter 5: Selective harmonic suppression for long horizons proposes a method to selectively suppress desired harmonics. Methods previously used to obtain similar outcomes are briefly discussed. The formulation of the suppression method is presented, which includes the design of an appropriate filter and an augmented state-space representation. The simulation framework is also explained.

Simulations are conducted which demonstrate the ability of the proposed method to suppress harmonics at selected frequencies. The proposed method is shown to benefit from long horizons.

Chapter 6: Conclusions and recommendations for future work concludes this thesis. Key observations and results of relevant chapters are summarised. Improvements and recommendations for future work are proposed and made, respectively.

Chapter 2

Theoretical background

2.1 Introduction

This chapter presents the relevant theory required for this thesis. The chapter starts by introducing a multilevel inverter topology that is used throughout the project. Model predictive control is formally introduced, along with its fundamental principles and components. The integer least-squares reformulation of the optimization problem underlying direct long-horizon model predictive control is presented. The solver for the integer least-squares problem, sphere decoding, is explained in detail. This chapter concludes with an overview of alternative modulation schemes that model predictive control will be measured against in terms of harmonic distortions. Relevant examples are given.

2.2 The neutral-point-clamped inverter

2.2.1 Introduction to topology

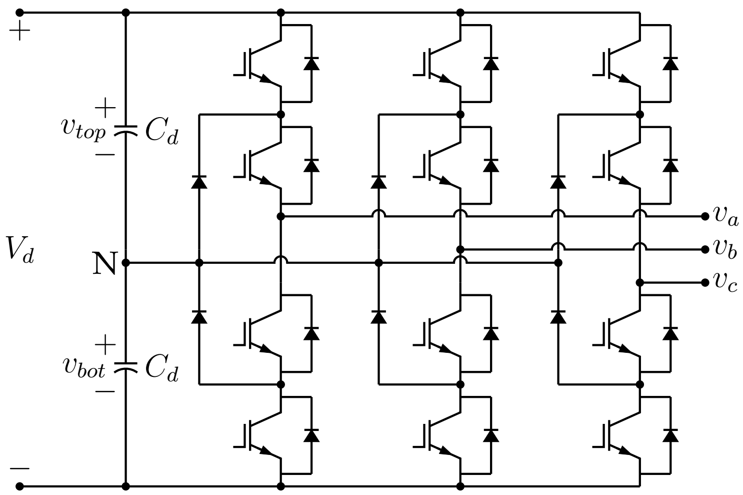


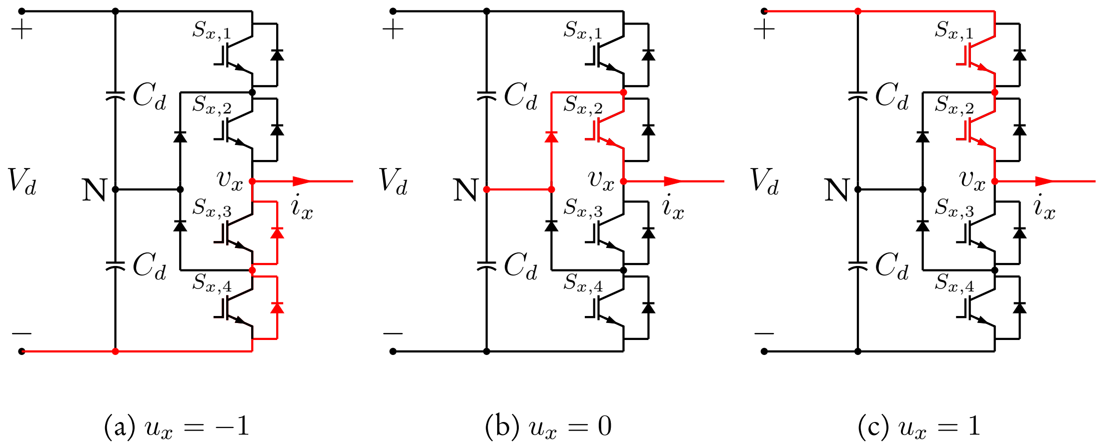
Figure 2.1: The neutral-point-clamped inverter, using insulated-gate bipolar transistors (IGBTs) semiconductor switches with their associated freewheeling diodes.

In 1981, a three-level inverter topology, known as the neutral-point-clamped (NPC) inverter, was introduced [17]. The inverter is shown in Figure 2.1, where V_d denotes the dc supply voltage. Two identical capacitors C_d are placed in parallel with the dc supply, where the point between them forms the neutral point N. The voltage across the top and bottom capacitors are denoted by v_{top} and v_{bot}, respectively.

A given phase arm can produce three different voltage levels with respect to the neutral point N: v_{top}, 0 V, and $-v_{bot}$. If fluctuations on the neutral point potential are neglected and the capacitor voltages v_{top} and v_{low} are equal, the output for a given phase arm is

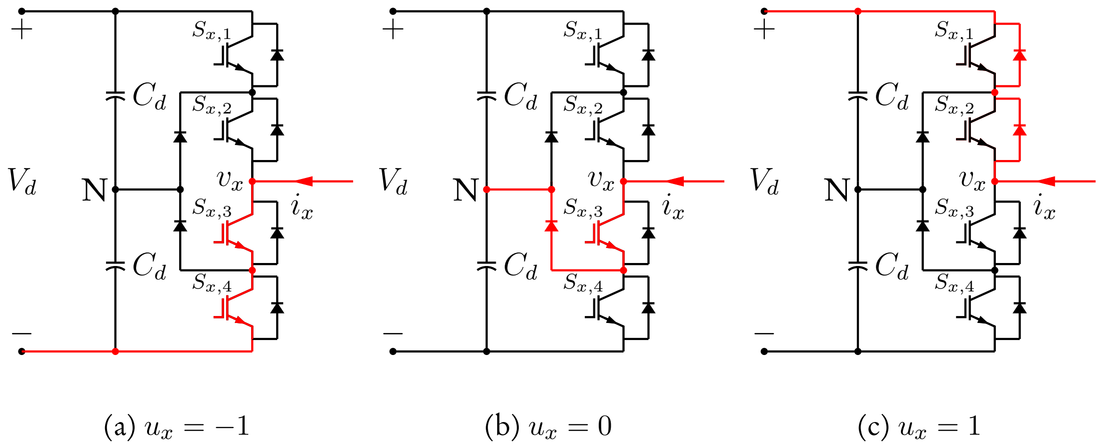
$$v_x = \frac{V_d}{2} u_x, \tag{2.1}$$

where $x \in \{a, b, c\}$ denotes the phase and $u_x \in \{-1, 0, 1\}$ represents the switch position for a given phase arm. Considering a phase arm of the inverter with switch position u_x, the positive and negative phase current paths are shown in Figure 2.2 and Figure 2.3, respectively. The output voltage and semiconductor states for a given phase with switch position u_x are summarised in Table 2.1.



(a) $u_x = -1$ (b) $u_x = 0$ (c) $u_x = 1$

Figure 2.2: Current paths for positive phase current.



(a) $u_x = -1$ (b) $u_x = 0$ (c) $u_x = 1$

Figure 2.3: Current paths for negative phase current.

Table 2.1: Switching states for phase arm of NPC inverter.

u_x	$S_{x,1}$	$S_{x,2}$	$S_{x,3}$	$S_{x,4}$	v_x
-1	0	0	1	1	$-\frac{V_d}{2}$
0	0	1	1	0	0
1	1	1	0	0	$\frac{V_d}{2}$

2.2.2 Clarke transformation and switching vectors

To simplify the control formulation in Section 3.2, consider the Clarke transformation

$$\boldsymbol{\xi}_{\alpha\beta0} = \boldsymbol{K}\boldsymbol{\xi}_{abc}, \tag{2.2}$$

and inverse Clarke transformation

$$\boldsymbol{\xi}_{abc} = \boldsymbol{K}^{-1}\boldsymbol{\xi}_{\alpha\beta0}, \tag{2.3}$$

where

$$\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \text{ and } \boldsymbol{K}^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix}. \tag{2.4}$$

Since only non-grounded star-connected loads are considered, the 0-component of the orthogonal reference frame is not required [1], as will be explained shortly. Therefore, the transformations are reduced to

$$\boldsymbol{\xi}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{\xi}_{abc} \tag{2.5}$$

and

$$\boldsymbol{\xi}_{abc} = \boldsymbol{K}^{-1}\boldsymbol{\xi}_{\alpha\beta}, \tag{2.6}$$

where the transformation matrices are redefined as

$$\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \text{ and } \boldsymbol{K}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}. \tag{2.7}$$

There exist $3^3 = 27$ different switching states that the inverter switch positions can assume in the form of $\boldsymbol{u}_{abc} = \begin{bmatrix} u_a & u_b & u_c \end{bmatrix}^T$. After applying the Clarke transformation to all of the different switching states

$$\boldsymbol{u}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{u}_{abc}, \tag{2.8}$$

19 unique switching vectors in the form $\boldsymbol{u}_{\alpha\beta} = \begin{bmatrix} u_\alpha & u_\beta \end{bmatrix}^T$ are produced, as illustrated in Figure 2.4. Note that there are 27 different switching states, but only 19 switching vectors. This is because 6 pairs of switching states produce identical short switching vectors, and 3 switching states produce the zero switching vectors, as shown in Table 2.2. These vectors are known as *redundant vectors*.

The inverter output voltage in the orthogonal reference frame is given by

$$\boldsymbol{v}_{\alpha\beta} = \frac{V_d}{2}\boldsymbol{K}\boldsymbol{u}_{abc}, \tag{2.9}$$

and it is apparent that the voltage vectors produced have the same form as the switching vectors shown in Figure 2.4. The (neglected) 0-component v_0 of the voltage vector represents the *common-mode voltages*, which do not drive phase currents if the star connection of the load floats [1]. The $\alpha\beta$-components form the *differential-mode voltages* that drive phase currents.

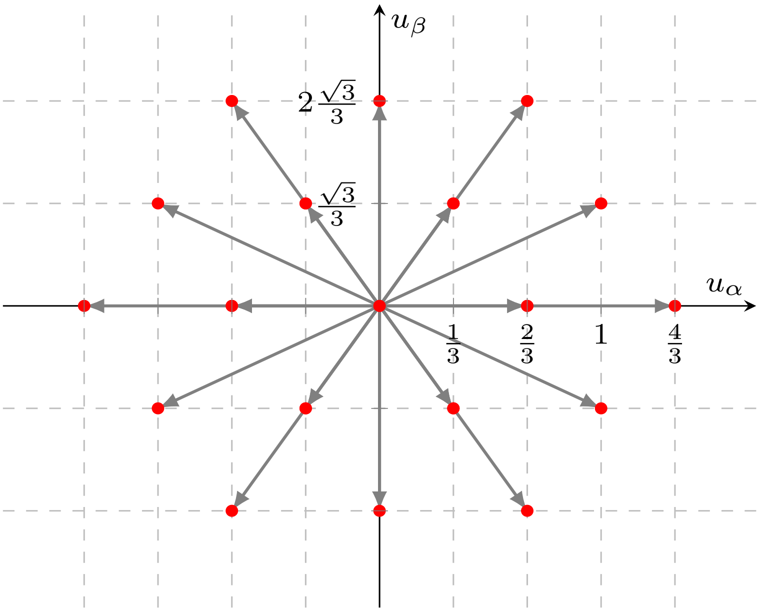


Figure 2.4: The switching vectors produced by a three-level NPC inverter.

Table 2.2: Switching vector table.

Classification	u_a	u_b	u_c	u_α	u_β
Zero	-1	-1	-1	0	0
	0	0	0		
	1	1	1		
Short	1	0	1	$\frac{1}{3}$	$-\frac{\sqrt{3}}{3}$
	0	-1	0		
	0	0	1	$-\frac{1}{3}$	$-\frac{\sqrt{3}}{3}$
	-1	-1	0		
	1	1	0	$\frac{1}{3}$	$\frac{\sqrt{3}}{3}$
	0	0	-1		
	0	1	0	$-\frac{1}{3}$	$\frac{\sqrt{3}}{3}$
	-1	0	-1		
	1	0	0	$\frac{2}{3}$	0
	0	-1	-1		
	0	1	1	$-\frac{2}{3}$	0
	-1	0	0		
Medium	0	1	-1	0	$2\frac{\sqrt{3}}{2}$
	0	-1	1	0	$-2\frac{\sqrt{3}}{2}$
	1	0	-1	1	$\frac{\sqrt{3}}{2}$
	1	-1	0	1	$-\frac{\sqrt{3}}{2}$
	-1	1	0	-1	$\frac{\sqrt{3}}{2}$
	-1	0	1	-1	$-\frac{\sqrt{3}}{2}$
Long	1	-1	-1	$\frac{4}{3}$	0
	-1	1	1	$-\frac{4}{3}$	0
	1	1	-1	$\frac{2}{3}$	$2\frac{\sqrt{3}}{3}$
	-1	-1	1	$-\frac{2}{3}$	$-2\frac{\sqrt{3}}{3}$
	-1	1	-1	$-\frac{2}{3}$	$2\frac{\sqrt{3}}{3}$
	1	-1	1	$\frac{2}{3}$	$-2\frac{\sqrt{3}}{3}$

2.3 Model predictive control

2.3.1 Introduction to MPC

History and fundamental principles

Model predictive control (MPC) is rooted in optimal control theory [18]. Since its inception in the 1970s, MPC has been widely used in the process industry [19], where plant dynamics are relatively slow. MPC has only recently been readily adopted in power electronics due to the increased computational power of microprocessors and FPGAs.

By using the internal dynamic model of a system, MPC predicts the evolution of the sampled system state over a prediction horizon. A constrained optimal control problem is formulated over the prediction horizon, with the control objectives as a cost function. By solving an optimization problem, an optimal control sequence is obtained that minimises the cost function, resulting in the optimal behaviour of the system. In order to provide feedback, the *receding horizon* policy is adopted. This implies that only the first control action is applied to the system, and at the next sampling instant the state is updated with new measurements and the optimization procedure is repeated.

Advantages of MPC

If plant is nonlinear, multiple-input multiple-output (MIMO), or has constraints, the design effort increases when classic control methods are considered. MPC can easily cope with the aforementioned challenges.

Firstly, any nonlinearities can be included (or approximated) in the internal dynamic model of the system. Nonlinearities range from the switching behaviour of the power electronics system to the saturation effects of an inductor. With induction machines, the electromagnetic torque or stator flux magnitude, if directly controlled, are nonlinear functions of currents or flux linkages [1]. Note that when the dynamic model is nonlinear, nonlinear MPC arises; see [20] for more details on nonlinear MPC.

Secondly, constraints can be imposed on the inputs, states, and outputs. For example, with proportional-integral (PI) controllers, anti-windup schemes have to be implemented in order to prevent the integrator from saturating. With MPC, a constraint can simply be imposed on any of the variables and requires no augmentation to the control loop.

Thirdly, MPC is well-suited for MIMO systems, requiring only a single control loop. With classical controllers, significant effort is required when designing for MIMO systems. The MIMO system is first broken down into multiple single-input single-output (SISO) cascaded control loops. Then, for every SISO control loop an individual controller is designed. However, when realising these control loops in practice, they tend to interact with each other in an adverse manner, especially during transients [1].

Continuous-control set MPC versus direct MPC

A power electronics system usually consists of a modulation stage and a converter. A modulator modulates the continuous input signal into a pulse-width modulated signal, that is used as gating signals for semiconductor switches. When using pulse-width modulation (PWM) with MPC, the method is known as *continuous-control set* MPC. The modulation stage is nonlinear and must be included in the internal dynamic model to account for the switching behaviour of the system. Unfortunately, nonlinear MPC requires solving a non-convex optimization problem [20]. Finding the global minimum for most non-convex optimization problems can prove to be intractable in real-time.

By using averaging techniques, such as the method of Middlebrook and Cuk [21], the internal dynamic model is simplified to an averaged state-space representation. The modulator is simplified to a gain, thus ignoring the baseband-, carrier multiple-, and sideband harmonics of PWM. However, the switching between sampling instants is ignored and not addressed by the controller. As a result, at low switching frequencies, averaging should be avoided [1].

An alternative is to model the power electronics system, which is sometimes referred to as a *hybrid system* [22, 23], as a piecewise affine (PWA) system [24]. The optimization problem is formulated as a mixed-integer quadratic program. The (explicit) state-feedback control law is solved offline by using multi-parametric programming, and stored in a lookup table [25]. The state-space is partitioned into polyhedra, where an affine control law is assigned to each polyhedron [25]. The problem then amounts to identifying which polyhedron the state vector belongs to. After identifying the polyhedron, the affine control law is read from the lookup table and the optimal control law is computed [25]. This is referred to as *explicit* MPC. This approach is well-suited for short sampling intervals and low dimensional problems. However, the memory requirements increase significantly as the dimension of the problem grows [1], e.g., many state variables. Interested readers are referred to [22] for more information on hybrid systems and their modelling, multi-parametric programming, and explicit MPC. For a summary on explicit MPC, see [25].

When removing the modulator and applying the outputs of the controller directly to the converter, *direct* MPC arises. The main advantage of direct MPC is that the internal dynamic model is linear, while the switching nature of the system is taken into consideration by the controller. Thus, direct MPC is suitable for low switching frequencies. The direct problem is convex up to the constraints, where the output of the controller is restricted to a set of integers. Hence, this method is also referred to as *finite-control set* MPC. Even though the optimization problem is non-convex, it can be solved efficiently by a branch-and-bound method that will be introduced in Section 2.5. From here on in, only direct MPC will be considered.

2.3.2 Internal dynamic model

Consider the controller and plant in Figure 2.5. The plant has an output vector $\boldsymbol{y} \in \mathbb{R}^{n_y}$ that is regulated by the controller along the reference $\boldsymbol{y}_{ref}$, and an input vector (also known as the *manipulated variable*) $\boldsymbol{u} \in \mathbb{R}^{n_u}$ that influences the state vector $\boldsymbol{x} \in \mathbb{R}^{n_x}$ of the system.

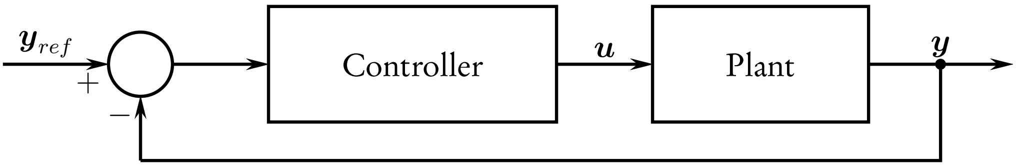


Figure 2.5: Block diagram consisting of a controller and plant.

Assuming that the plant in Figure 2.5 is linear time-invariant (LTI), it can be represented by a continuous-time state-space representation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{u}(t) \tag{2.10a}$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \tag{2.10b}$$

where $\boldsymbol{F} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{G} \in \mathbb{R}^{n_x \times n_u}$, and $\boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$ represent the state, input, and output matrix, respectively. The dimension of the state, input, and output vector are denoted by n_x, n_y, and n_u, respectively. The representation in (2.10) can be discretised to represent a discrete-time state-space representation [1],

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{2.11a}$$
$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k), \tag{2.11b}$$

where

$$\boldsymbol{A} = \boldsymbol{e}^{\boldsymbol{F}T_s} \tag{2.12a}$$
$$\boldsymbol{B} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_{n_x} - \boldsymbol{A})\boldsymbol{G}, \tag{2.12b}$$

with T_s being the sampling period, $\boldsymbol{e}$ as the matrix exponential, and $\boldsymbol{I}_{n_x}$ the identity matrix with dimension n_x.[1] The discretisation method in (2.12) is known as exact discretisation.

2.3.3 Constraints

Constraints can be imposed on the output, state, and input variables,

$$\boldsymbol{y}(k) \in \boldsymbol{\mathcal{Y}} \subseteq \mathbb{R}^{n_y} \tag{2.13a}$$

$$\boldsymbol{x}(k) \in \boldsymbol{\mathcal{X}} \subseteq \mathbb{R}^{n_x} \tag{2.13b}$$

$$\boldsymbol{u}(k) \in \boldsymbol{\mathcal{U}} \subseteq \mathbb{R}^{n_u}, \tag{2.13c}$$

during the optimization procedure. There are two types of constraints: *soft* constraints and *hard* constraints. Soft constraints, which are imposed by the user, can be violated during the optimization procedure to avoid infeasibility issues. Hard constraints are usually characteristics of the system, and thus cannot be relaxed.

Using direct MPC methods automatically imposes a non-convex constraint on the manipulated variable, meaning that the optimization problem will be non-convex. In the case of a three-phase three-level NPC inverter, the constraint is

$$\boldsymbol{\mathcal{U}} = \mathcal{U}^3, \tag{2.14}$$

with

$$\mathcal{U} = \{-1, 0, 1\}, \tag{2.15}$$

where $\mathcal{U}^3$ is the three-times Cartesian product of $\mathcal{U}$, that is, $\boldsymbol{\mathcal{U}} = \mathcal{U} \times \mathcal{U} \times \mathcal{U}$. Note that this constraint is physical in nature (i.e., the only switch positions the inverter can assume; see Table 2.1), and thus hard.

2.3.4 Optimal control problem

Cost function formulation

An optimal control problem involves constructing a cost function from the control objectives. A general cost function over N_p-time steps is given by

$$J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \sum_{l=k}^{k+N_p-1} V(\boldsymbol{x}(l), \boldsymbol{u}(l)), \tag{2.16}$$

where $V(\cdot,\cdot)$ denotes the *weighting functions* over the *prediction horizon*. For simplicity, N_p will be referred to as the prediction horizon from here on.[2] The weighting functions penalise the control objectives individually. The derivations henceforth will be for a three-phase three-level NPC inverter. See Appendix A for the general dimensions of the matrices and vectors. The *switching sequence* $\boldsymbol{U}(k) \in \boldsymbol{\mathcal{U}}^{N_p}$ is introduced as the control commands (i.e., the switch positions for the inverter) at every time step over the prediction horizon

$$\boldsymbol{U}(k) = \begin{bmatrix} \boldsymbol{u}^T(k) & \boldsymbol{u}^T(k+1) & \boldsymbol{u}^T(k+2) & \cdots & \boldsymbol{u}^T(k+N_p-1) \end{bmatrix}^T. \tag{2.17}$$

[1] If $\boldsymbol{F}$ is singular, then $\boldsymbol{B} = \int_0^{T_s} \boldsymbol{e}^{\boldsymbol{F}\tau} d\tau\, \boldsymbol{G}$

[2] If unclear, see clarification of terms in the nomenclature.

Recall from Section 1.1 that the system must have low harmonic distortions and low switching losses. These are two control objectives, and they directly relate to reference tracking and switching effort (i.e., the number of switching transitions). Formulating a quadratic cost function with the aforementioned objectives results in

$$J(\boldsymbol{x}(k),\boldsymbol{U}(k)) = \sum_{l=k}^{k+N_p-1} \left\|\boldsymbol{y}_{ref}(l+1) - \boldsymbol{y}(l+1)\right\|_{\boldsymbol{R}}^2 + \lambda_u \left\|\boldsymbol{u}(l) - \boldsymbol{u}(l-1)\right\|_2^2, \tag{2.18}$$

with

$$\boldsymbol{y}(l+1) = \boldsymbol{C}\boldsymbol{x}(l+1) \tag{2.19a}$$
$$\boldsymbol{x}(l+1) = \boldsymbol{A}\boldsymbol{x}(l) + \boldsymbol{B}\boldsymbol{u}(l), \tag{2.19b}$$

where $\boldsymbol{y}_{ref}$ is the current reference, $\left\|\boldsymbol{\xi}\right\|_2^2$ is the 2-norm (or Euclidean norm) squared of vector $\boldsymbol{\xi}$. The penalty on the tracking error is represented by $\left\|\boldsymbol{\xi}\right\|_{\boldsymbol{R}}^2$, which is the 2-norm squared of vector $\boldsymbol{\xi}$ weighted with the penalty matrix $\boldsymbol{R} \in \mathbb{R}^{n_y \times n_y}$. The weighting factor on the switching effort is denoted by $\lambda_u \in \mathbb{R}$.

The diagonal penalty matrix $\boldsymbol{R}$ can be used to prioritise certain references above others. For example, with an inductive-capacitive (LC) filter-connected induction machine, the stator current can be prioritised above inverter current. However, for the majority of this thesis, the systems considered will not require certain references to be prioritised above others. For simplicity, the penalty matrix can simply be set to $\boldsymbol{R} = \boldsymbol{I}_{n_y}$. By tuning the weighting factor λ_u, the trade-off between tracking error and switching effort (that is, between harmonic distortions and switching losses) is adjusted. As $\lambda_u \to 0$, the tracking error is prioritised and the controller allows switching transitions more freely. As $\lambda_u \to \infty$, switching losses are prioritised and the controller allows less switching transitions. Thus, λ_u is used to adjust the (average) switching frequency f_{sw} of the system. Note that direct MPC has a variable switching frequency.

The cost function (2.18) can be written in a vector form [10]

$$J(\boldsymbol{x}(k),\boldsymbol{U}(k)) = \left\|\boldsymbol{Y}_{ref}(k) - \boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{\Upsilon}\boldsymbol{U}(k)\right\|_2^2 + \lambda_u \left\|\boldsymbol{S}\boldsymbol{U}(k) - \boldsymbol{E}\boldsymbol{u}(k-1)\right\|_2^2, \tag{2.20}$$

with

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^{N_p} \end{bmatrix}, \boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{CB} & 0_{n_y\times 3} & \cdots & 0_{n_y\times 3} \\ \boldsymbol{CAB} & \boldsymbol{CB} & \cdots & 0_{n_y\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA}^{N_p-1}\boldsymbol{B} & \boldsymbol{CA}^{N_p-2}\boldsymbol{B} & \cdots & \boldsymbol{CB} \end{bmatrix},$$

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{I}_3 & 0_{3\times 3} & \cdots & 0_{3\times 3} \\ -\boldsymbol{I}_3 & \boldsymbol{I}_3 & \cdots & 0_{3\times 3} \\ 0_{3\times 3} & -\boldsymbol{I}_3 & \cdots & 0_{3\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{3\times 3} & 0_{3\times 3} & \cdots & \boldsymbol{I}_3 \end{bmatrix} \text{ and } \boldsymbol{E} = \begin{bmatrix} \boldsymbol{I}_3 \\ 0_{3\times 3} \\ 0_{3\times 3} \\ \vdots \\ 0_{3\times 3} \end{bmatrix},$$

where $\boldsymbol{0}_{n\times n}$ is the zero matrix with appropriate dimensions and $\boldsymbol{Y}_{ref}(k) \in \mathbb{R}^{n_y N_p}$ denotes the reference over the prediction horizon

$$\boldsymbol{Y}_{ref}(k) = \begin{bmatrix} \boldsymbol{y}_{ref}^T(k+1) & \boldsymbol{y}_{ref}^T(k+2) & \boldsymbol{y}_{ref}^T(k+3) & \cdots & \boldsymbol{y}_{ref}^T(k+N_p) \end{bmatrix}^T. \tag{2.21}$$

As with (2.18), the first term in (2.20) penalises the tracking error and the second term penalises the switching effort. The derivation of (2.20) is presented in Appendix A.

Optimization stage

The minimisation of $J(\boldsymbol{x}(k), \boldsymbol{U}(k))$, subjected to constraints on the input variable, and with $\boldsymbol{U}(k)$ as the *decision variable* of the problem, results in the (open-loop) *optimal solution* $\boldsymbol{U}_{opt}(k)$ (also known as the *optimizer*) that results in the ideal behaviour of the system. The optimization problem can be stated as

$$\boldsymbol{U}_{opt}(k) = \arg\min_{\boldsymbol{U}(k)} \quad J(\boldsymbol{x}(k), \boldsymbol{U}(k)) \tag{2.22a}$$

$$\text{subject to } \boldsymbol{U}(k) \in \mathbb{U}, \tag{2.22b}$$

where $\mathbb{U}$ is the N_p-times Cartesian product of $\boldsymbol{\mathcal{U}}$, that is, $\mathbb{U} = \boldsymbol{\mathcal{U}}^{N_p}$. The *feasible set* $\mathbb{U}$ for the optimization problem represents all of the possible switching sequences (also known as *candidate solutions*) that $\boldsymbol{U}(k)$ can assume.

The optimization problem is most often solved via the enumeration of all possible switching sequences with Algorithm 1. This is known as *exhaustive search*. This approach, however, is only suitable for very short horizons of $N_p = 1$ or $N_p = 2$. The number of candidate solutions for (2.22) is given by 3^{3N_p}, where the base value represents the number of single-phase switch positions ($|\mathcal{U}| = 3$) and the constant in the exponent represents the number of inputs ($n_u = 3$). It should be apparent that an increase in the prediction horizon causes an exponential increase in candidate solutions for (2.22), e.g., $N_p = 1$ has 27 candidate solutions, $N_p = 2$ has 729 candidate solutions, and $N_p = 3$ has 19 683 candidate solutions.

Algorithm 1 Exhaustive search

```
Initialize:
$J = \infty$, $\boldsymbol{U}_{opt}(k) = \emptyset$, $\mathbb{U} = \boldsymbol{\mathcal{U}}^{N_p}$
for each $\boldsymbol{U}(k) \in \mathbb{U}$ do
    $J_{temp} = \left\|\boldsymbol{Y}_{ref}(k) - \boldsymbol{\Gamma x}(k) - \boldsymbol{\Upsilon U}(k)\right\|_2^2 + \lambda_u \left\|\boldsymbol{SU}(k) - \boldsymbol{Eu}(k-1)\right\|_2^2$
    if $J_{temp} \leq J$ then
        $J = J_{temp}$
        $\boldsymbol{U}_{opt}(k) = \boldsymbol{U}(k)$
    end if
end for
```

2.3.5 The receding horizon policy

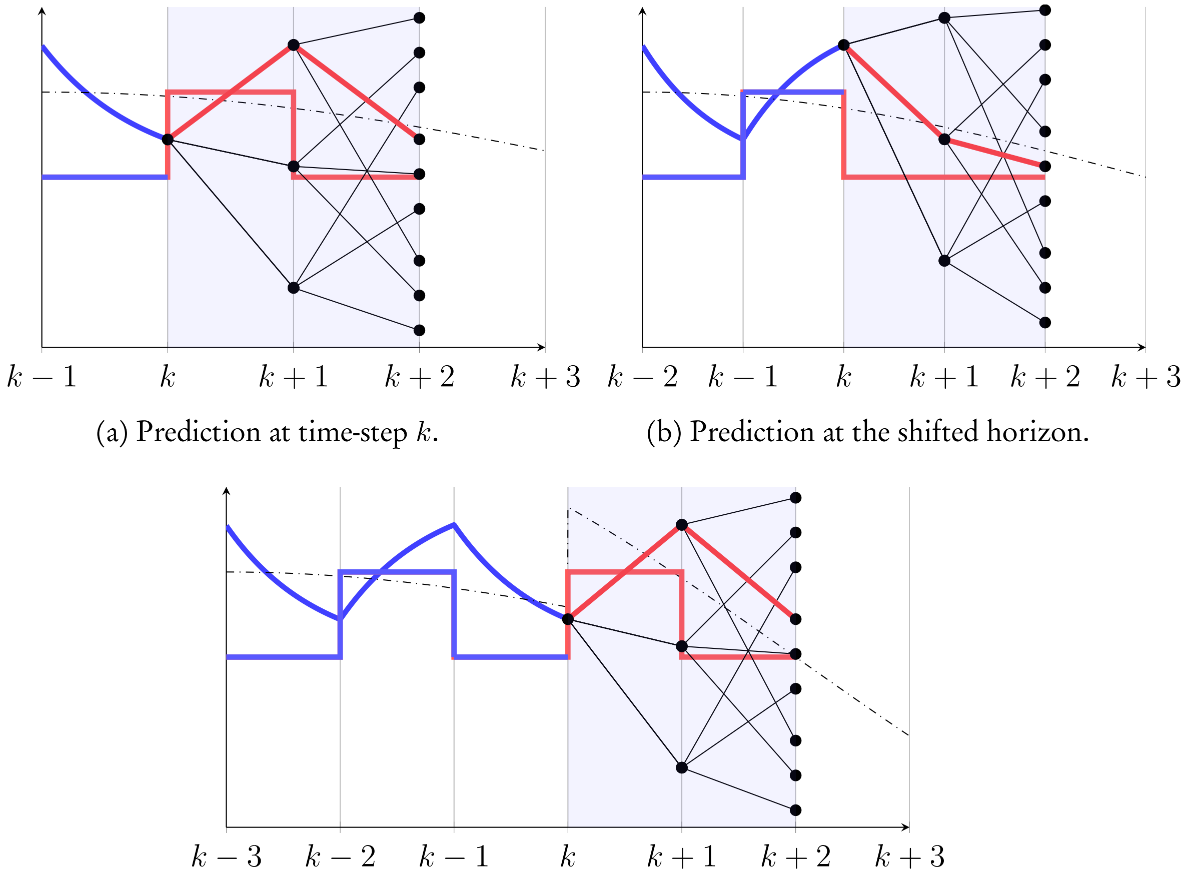
As mentioned in Section 2.3.4, the optimization problem (2.22) results in an open-loop switching sequence. To provide feedback and to make the system more robust to uncertainties and disturbances, only the first element in $\boldsymbol{U}_{opt}(k)$ (i.e., $\boldsymbol{u}_{opt}(k)$) is applied to the system at time-step k. At the next time-step $k+1$, new measurements are taken and the optimization procedure is repeated. An exhaustive search based controller is described below.

1. Obtain measurements for $\boldsymbol{x}(k)$.
2. Execute Algorithm 1.
3. Apply $\boldsymbol{u}_{opt}(k)$ to system.
4. Shift the prediction horizon with one time-step, $k = k + 1$.
5. Return to step 1.

2.3.6 Illustrative example of direct MPC

To illustrate direct MPC, consider Figure 2.6, where the phase current, switching sequence, and predicated current are shown. The system is a single-phase three-level NPC with an RL load. The prediction horizon is set to $N_p = 2$.

In Figure 2.6a, the phase current is sampled at time-step k. By using Algorithm 1, the current trajectories (black lines) for all of the possible switching sequences are evaluated. The trajectory (red line) that results in the minimisation of the cost function (2.20) yields the optimal switching sequence (red line). In Figure 2.6b, the prediction horizon is shifted forward with one time-step ($k = k+1$). The first element of the previous optimal switching sequence has been applied to the system (now $u_{opt}(k-1)$), and the entire optimization procedure is repeated. Figure 2.6c demonstrates the advantage and importance of the receding horizon principle. Although the optimal switch position calculated in Figure 2.6b was $u_{opt}(k|k-1) = 0$, the sudden reference step at k resulted in a different optimal switch position of $u_{opt}(k|k) = 1$.[3] Other disturbances include the quantization error of an analogue-to-digital converter (ADC), model uncertainties, assumptions that were made during modelling, and so forth.



(a) Prediction at time-step k.

(b) Prediction at the shifted horizon.

(c) Prediction at shifted horizon with a sudden reference step.

Figure 2.6: An example of a prediction horizon $N_p = 2$. The dash-dotted black line represents the reference current. The blue lines depict the phase current and switch positions up to the sample instant k. The black lines represent the predicted phase current for the 9 possible switching sequences. The red lines represent the optimal switching sequence and the associated current trajectory calculated at time-step k. The shaded area depicts the horizon.

[3] The notation $u_{opt}(k|k-1)$ implies that the optimal switch position at k (in Figure 2.6c) was calculated at the previous sample $k-1$ (now Figure 2.6b).

2.4 Integer quadratic programming formulation

As stated in Section 2.3.4, using a brute force approach to solve the optimization problem, via an exhaustive search, rapidly becomes computationally intractable as the prediction horizon increases. To achieve long horizons, the optimization problem in (2.22) will be reformulated as an integer quadratic program (QP) [10].

The cost function in (2.20) is first written in a more compact form of [10]

$$J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \boldsymbol{U}^T(k)\boldsymbol{Q}\boldsymbol{U}(k) + 2\boldsymbol{\Theta}^T(k)\boldsymbol{U}(k) + \theta(k) \tag{2.23}$$

where

$$\boldsymbol{Q} = \boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^T\boldsymbol{S} \tag{2.24}$$

$$\boldsymbol{\Theta}(k) = ([\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k)]^T\boldsymbol{\Upsilon} - \lambda_u[\boldsymbol{E}\boldsymbol{u}(k-1)]^T\boldsymbol{S})^T \tag{2.25}$$

$$\theta(k) = \left\|\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k)\right\|_2^2 + \lambda_u\left\|\boldsymbol{E}\boldsymbol{u}(k-1)\right\|_2^2. \tag{2.26}$$

The matrix $\boldsymbol{Q} \in \mathbb{R}^{3N_p \times 3N_p}$ is symmetrical and positive semidefinite for $\lambda_u \geq 0$ [1]. Note that $\theta(k)$ is a function of the sampled state $\boldsymbol{x}(k)$, the reference over the prediction horizon $\boldsymbol{Y}_{ref}(k)$, and the previous switch positions $\boldsymbol{u}(k-1)$, but is independent of the decision variable $\boldsymbol{U}(k)$. Therefore, $\theta(k)$ will be constant during optimization and can be omitted. The derivation of (2.23) can be found in Appendix B.

2.4.1 The unconstrained quadratic program

If the constraints are neglected, i.e., the feasible set becomes $\mathbb{R}^{3N_p}$, the underlying (convex) optimization problem is an *unconstrained* quadratic program (QP):

$$\boldsymbol{U}_{opt}(k) = \arg\min_{\boldsymbol{U}(k)} \quad \boldsymbol{U}^T(k)\boldsymbol{Q}\boldsymbol{U}(k) + 2\boldsymbol{\Theta}^T(k)\boldsymbol{U}(k). \tag{2.27}$$

For technical reasons that will be explained, $\boldsymbol{Q}$ (known as the *Hessian*) is required to be positive definite, and this only holds true when $\lambda_u > 0$ [10]. Since the feasible set is convex and the Hessian is positive definite, the QP can be solved in polynomial time [14]. This means that as the dimension of the problem is increased, the computational time increases polynomially. Since the optimization problem is convex, any local minimum (if there were minima) is also a global minimum. The minimum can be found by differentiating (2.23) and finding the point where the gradient is zero [1]. This is known as the *unconstrained solution* $\boldsymbol{U}_{unc}(k) \in \mathbb{R}^{3N_p}$ and after differentiating

$$\nabla J(\boldsymbol{U}_{unc}(k)) = 0, \tag{2.28}$$

is found as

$$\boldsymbol{U}_{unc}(k) = -\boldsymbol{Q}^{-1}\boldsymbol{\Theta}(k). \tag{2.29}$$

After completing the squares in (2.27), omitting any terms independent of $\boldsymbol{U}(k)$, and inserting (2.29), the (unconstrained) QP becomes [1]

$$\boldsymbol{U}_{opt}(k) = \arg\min_{\boldsymbol{U}(k)} \quad (\boldsymbol{U}(k) - \boldsymbol{U}_{unc}(k))^T\boldsymbol{Q}(\boldsymbol{U}(k) - \boldsymbol{U}_{unc}(k)). \tag{2.30}$$

2.4.2 The integer quadratic program

Although $\boldsymbol{U}_{unc}(k)$ minimises the cost function, it cannot be used as gating signals for the inverter due to the (hard) integer constraints. By reintroducing the constraints, the optimization problem can be stated as a (truncated) *integer* quadratic program:

$$\boldsymbol{U}_{opt}(k) = \arg\min_{\boldsymbol{U}(k)} \quad (\boldsymbol{U}(k) - \boldsymbol{U}_{unc}(k))^T \boldsymbol{Q} (\boldsymbol{U}(k) - \boldsymbol{U}_{unc}(k)) \tag{2.31a}$$

$$\text{subject to } \boldsymbol{U}(k) \in \mathbb{U}. \tag{2.31b}$$

Recall from Section 2.4.1 that $\boldsymbol{Q}$ is symmetric and positive definite. Therefore, there exists a unique invertible lower triangular matrix $\boldsymbol{H} \in \mathbb{R}^{3N_p \times 3N_p}$ such that

$$\boldsymbol{H}^T \boldsymbol{H} = \boldsymbol{Q}. \tag{2.32}$$

This is known as the Cholesky decomposition [26].[4] By applying (2.32) to (2.31), the optimization problem becomes an *integer least-squares* (ILS) problem:

$$\boldsymbol{U}_{opt}(k) = \arg\min_{\boldsymbol{U}(k)} \quad \left\|\boldsymbol{H}\boldsymbol{U}(k) - \boldsymbol{H}\boldsymbol{U}_{unc}(k)\right\|_2^2 \tag{2.33a}$$

$$\text{subject to } \boldsymbol{U}(k) \in \mathbb{U} \tag{2.33b}$$

The $\boldsymbol{H}$ matrix is known as the *generator* matrix, as it generates the (truncated) lattice Λ (i.e., a discrete space) of the ILS problem. The lattice points, representing transformed candidate solutions for the ILS problem, are generated from all of the possible switching sequences,

$$\Lambda = \{\boldsymbol{H}\boldsymbol{U}(k) | \boldsymbol{U}(k) \in \mathbb{U}\}. \tag{2.34}$$

Note that the columns of $\boldsymbol{H}$ form the basis for the new space, and the set of linear combinations of the basis spans the lattice, and thus can also be defined by

$$\Lambda = \{\sum_{i=1}^{3N_p} u_i \boldsymbol{h}_i | u_i \in \mathcal{U}\}. \tag{2.35}$$

Note that the lattice is truncated in the sense that the switching sequences only contain the integers from the set of single-phase switch positions $\mathcal{U}$ (and not $\mathbb{Z}$). Figure 2.7 illustrates the transformation from the original space (u_1-u_2) to the transformed space ($\bar{u}_1$-$\bar{u}_2$). Note that the example is 2-dimensional for illustration purposes. When considering a system with $n_u = 3$, the dimension increases by a factor of three for an increase in N_p. The general case has a dimension of $n = n_u N_p$

[4] It should be noted that, by definition, the Cholesky decomposition results in $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{A}$ or $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{A}$, where $\boldsymbol{L}$ and $\boldsymbol{U}$ are lower and upper triangular matrices, respectively. However, by taking the Cholesky decomposition of $\boldsymbol{A}^{-1}$, $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{A}^{-1}$, and inverting both sides results in a lower triangular matrix that satisfies $\boldsymbol{L}^{-T}\boldsymbol{L}^{-1} = \boldsymbol{A}$.

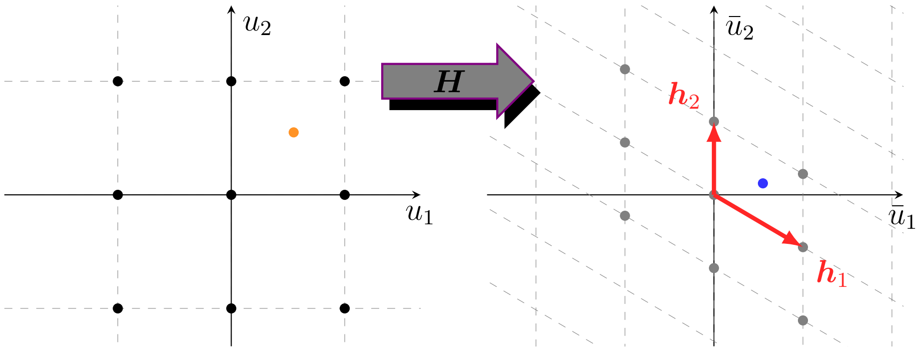


Figure 2.7: The $\boldsymbol{H}$ matrix transforms the integer switching sequences (black dots), in the orthogonal space, to transformed switching sequences (gray dots). Note that the transformed space is skew. The unconstrained solution (orange dot) is also transformed to the new space (blue dot). $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ represent the columns of $\boldsymbol{H}$, which are the basis vectors for the transformed space.

The ILS problem (2.33) can geometrically be interpreted as finding the nearest lattice point (a gray dot in Figure 2.7) to the transformed unconstrained solution (the blue dot in Figure 2.7). This is also known as the *closest vector problem* and is widely used in other fields such as communication theory and cryptography [27]. The closest vector problem is known to be *nondeterministic polynomial-time hard* (NP-hard) [27].[5] This means that it is highly unlikely that an algorithm will exist that can solve the problem in polynomial time; by increasing the dimension of the problem, the computation time increases exponentially. Note that when referring to the computational complexity of a problem, it is referred to as the worst-case scenario of that problem (in other words, the upper bound) and how the difficulty scales when significantly increasing the size of the problem (e.g., how much more difficult a 100-dimensional problem is to solve than a 10-dimensional problem). Note that unlike (2.33), (2.31) does not refer to the nearest neighbour of $\boldsymbol{U}_{unc}(k)$ in terms of the Euclidean distance.

2.5 Sphere decoding

2.5.1 Introduction to sphere decoding

Although the ILS problem is NP-hard, a branch-and-bound technique known as *sphere decoding*, in conjunction with a good initial candidate solution, solves the optimization problem in a time-efficient manner for the dimensions considered in power electronics [10]. Sphere decoding has been shown to solve the optimization problem for a prediction horizon of $N_p = 10$ (30-dimensional problem) extremely efficiently [15].

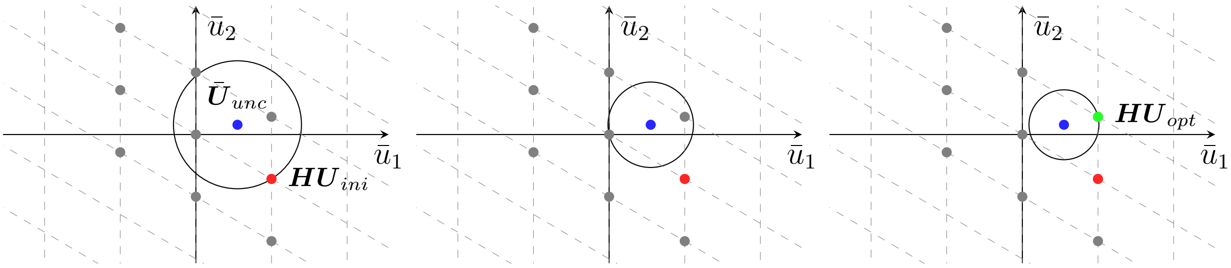
The principle of sphere decoding involves considering candidate solutions within a sphere with radius $\rho(k)$ that is centred at $\bar{\boldsymbol{U}}_{unc}(k) = \boldsymbol{H}\boldsymbol{U}_{unc}(k)$,

$$\rho(k) \geq \left\| \bar{\boldsymbol{U}}_{unc}(k) - \boldsymbol{H}\boldsymbol{U}(k) \right\|_2 . \tag{2.36}$$

The radius is tightened (decreasing the upper bound) until only one candidate solution lies within the sphere, which is then the *optimal solution* $\boldsymbol{U}_{opt}(k)$. Sphere decoding is illustrated

[5] NP has two definitions. The first is when given the solution of a problem, an algorithm exists that can *verify* that the solution is indeed optimal in polynomial time. The second definition states that a nondeterministic algorithm can *solve* the problem in polynomial time. The *hard* part in NP-hard means that if an algorithm is found to solve the NP-hard problem in polynomial time, all NP problems can be solved in polynomial time. The curious readers are referred to [28] for more details on computational complexity.

in Figure 2.8. The initial radius $\rho_{ini}(k)$, based on an initial candidate solution $\boldsymbol{U}_{ini}(k)$, should be chosen as small as possible so that the vast majority of the candidate solutions are not enclosed by the sphere. However, if the radius is too small the set of candidate solutions will be empty and feasibility issues will arise.[6]



(a) A radius is chosen based on an initial candidate solution.

(b) The tightening of the sphere around the transformed unconstrained solution.

(c) The sphere is tightened until only one point lies within the sphere, which is then the optimal solution.

Figure 2.8: Principle of sphere decoding.

To efficiently identify the candidate solutions that belong to the sphere, consider the lower triangular $\boldsymbol{H}$ matrix, which is a key property in sphere decoding. By exploiting the triangularity of $\boldsymbol{H}$, the squared radius of a candidate solution from the unconstrained solution is given by

$$\begin{aligned} d'^2(k) = {} & \underbrace{\overbrace{(\bar{u}_{unc,1} - h_{(1,1)}u_1)^2}^{d_1^2} + \overbrace{(\bar{u}_{unc,2} - h_{(2,1)}u_1 - h_{(2,2)}u_2)^2}^{d_2^2}}_{d_2'^2} + \dots \\ & + \overbrace{(\bar{u}_{unc,3N_p} - h_{(3N_p,1)}u_1 - h_{(3N_p,2)}u_2 - \dots - h_{(3N_p,3N_p)}u_{3N_p})^2}^{d_{3N_p}^2} \end{aligned} \tag{2.37}$$

where ξ_j denotes the j'th element of a vector, and $\zeta_{(i,j)}$ denotes the (i,j)'th entry of a matrix. The squared term of the j'th element is denoted by d_j^2, and $d_j'^2$ represents the intermediate squared radius up to the j'th element (i.e., the squared terms accumulated from levels 1 to j). The $\boldsymbol{H}$ matrix enables the multi-dimensional problem to be solved in a one-dimensional manner, by allowing a switching sequence to be assembled entry by entry and testing if the intermediate radius exceeds the sphere. From (2.36) and (2.37) it is apparent that for a candidate solution to be inside the sphere it must satisfy $\rho^2(k) \geq d_j'^2(k)$ at all instants during assembly. The computational burden associated with the calculation of the (intermediate) squared radius is alleviated by the fact that the lower levels (that is, levels 1 to $j-1$) do not have to be recomputed.

There are two methods that provide a good initial candidate solution: the *Babai estimate* [29] and the *educated guess* [10]. The Babai estimate involves rounding $\boldsymbol{U}_{unc}(k)$ to the nearest

[6] It might cause some confusion by stating the candidate solutions, in other words, switching sequences $\boldsymbol{U}(k)$ that belong to the feasible set $\mathbb{U}$, enclosed by the sphere are considered. Geometrically, the transformed candidate solutions $\boldsymbol{HU}(k)$ (i.e., the lattice points) are considered that belongs to the sphere. However, it should be obvious that if a lattice point is located outside of the sphere, then the associated candidate solution will violate the upper bound and considered a suboptimal solution. For simplicity, lattice points will also be referred to as candidate solutions.

switching sequence,

$$\boldsymbol{U}_{bab}(k) = \lfloor \boldsymbol{U}_{unc}(k) \rceil \in \mathbb{U}, \tag{2.38}$$

that is, the nearest black dot to the orange dot in Figure 2.7. The closer the basis vectors of $\boldsymbol{H}$ are to being orthogonal, and thus the lattice, the greater the probability that the Babai estimate is the optimal solution $\boldsymbol{U}_{opt}(k)$. The educated guess is based on the previous optimal solution $\boldsymbol{U}_{opt}(k-1)$, where the entries are shifted forward by one time step and repeating the last entry,

$$\boldsymbol{U}_{ed}(k) = \begin{bmatrix} 0_{3\times3} & \boldsymbol{I}_3 & 0_{3\times3} & \cdots & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & \boldsymbol{I}_3 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0_{3\times3} \\ 0_{3\times3} & \cdots & \cdots & 0_{3\times3} & \boldsymbol{I}_3 \\ 0_{3\times3} & \cdots & \cdots & 0_{3\times3} & \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{U}_{opt}(k-1). \tag{2.39}$$

The educated guess is based on the assumption that the switching transitions that were discarded in the receding horizon principle should, ideally, match those that are calculated at the next time-step. It will be explained in Section 3.3.4 that both the Babai estimate and educated guess should be investigated when calculating the initial sphere radius.

2.5.2 Sphere decoding algorithm

The recursive sphere decoding algorithm proposed in [10] is shown in Algorithm 2. The arguments are passed to the algorithm as $\boldsymbol{U}_{opt} = \textsc{SphDec}(\emptyset, 0, 1, \rho_{ini}^2, \bar{\boldsymbol{U}}_{unc})$.

Algorithm 2 Sphere decoder

```
function $\boldsymbol{U}_{opt} = \textsc{SphDec}(\boldsymbol{U}, d^2, j, \rho^2, \bar{\boldsymbol{U}}_{unc})$
    for each $u \in \mathcal{U}$ do
        $u_j = u$
        $d'^2 = \left\| \bar{u}_{unc,j} - \boldsymbol{H}_{(j,1:j)} \boldsymbol{U}_{1:j} \right\|_2^2 + d^2$
        if $\rho^2 \geq d'^2$ then
            if $j < 3N_p$ then
                $\textsc{SphDec}(\boldsymbol{U}, d'^2, j+1, \rho^2, \bar{\boldsymbol{U}}_{unc})$
            else
                $\boldsymbol{U}_{opt} = \boldsymbol{U}$
                $\rho^2 = d'^2$
            end if
        end if
    end for
end function
```

The algorithm can be visualised by traversing a search tree with a depth $n = 3N_p$ (see Figure 2.9 for an example of a search tree). The algorithm assembles the switching sequence $\boldsymbol{U}(k)$ entry by entry, where the levels of the search tree represent an entry u_j. The admissible single-phase switch positions $\mathcal{U} = \{-1, 0, 1\}$ are considered at every level and are represented by the branches of the search tree. When a branch is explored, a node is visited where the (intermediate) squared radius is calculated. If d'^2 exceeds the radius of the sphere, there is no need to explore the current branch any further. After the branch is pruned from the search tree, thereby reducing the candidate solutions that have to be considered, the adjacent branch is explored. Once the bottom node is reached, known as a *leaf node*, a full switching sequence has been assembled. If the candidate solution belongs to the sphere, the incumbent optimal solution is updated and the sphere is tightened. Note that a certificate for optimality (i.e., the incumbent optimal solution is optimal) is only obtained once all the possible branches have been investigated.

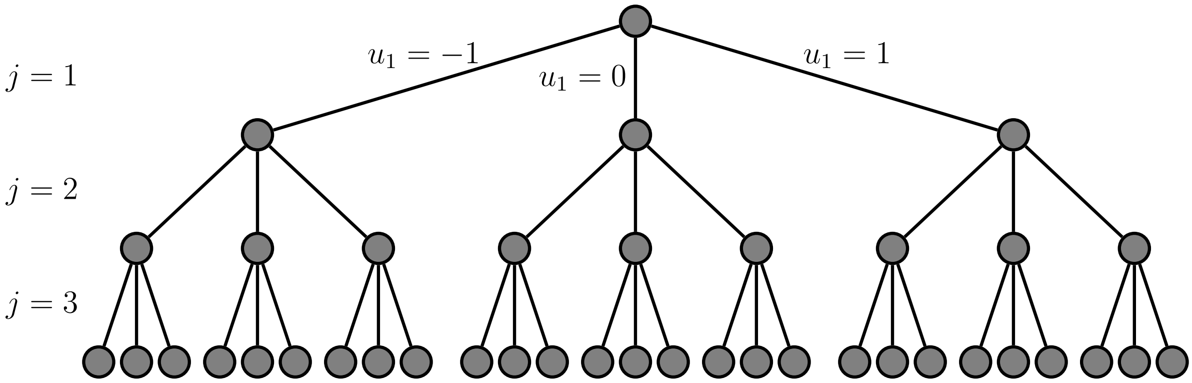


Figure 2.9: Search tree with depth $n = 3$. The branches represent single-phase switch positions $\mathcal{U}$. The level in the search tree is denoted by j and relates to the j'th entry in $\boldsymbol{U}(k)$

2.5.3 Illustrative example of sphere decoding

Consider a three-phase three-level NPC inverter with a prediction horizon of $N_p = 1$. An illustrative example on how the sphere decoder traverses a search tree is given in Figure 2.10.

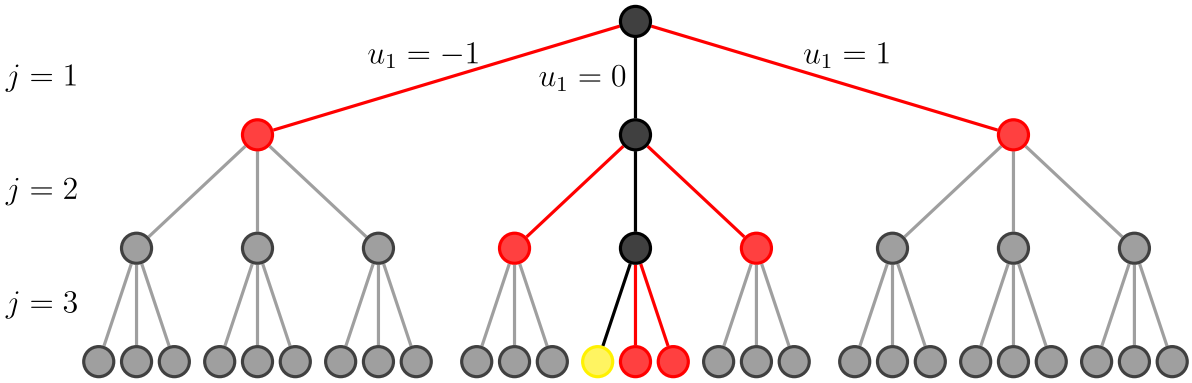


Figure 2.10: Example of the traversal of a search tree for $N_p = 1$. The gray branches and nodes represents those that were not explored during the sphere decoding. The red branches and nodes are those that exceeded the radius of sphere and are pruned from the tree. The black branches and nodes were traversed and evaluated, and the yellow node represents the leaf node corresponding to the optimal solution $\boldsymbol{U}_{opt}(k)$.

The algorithm considers the single-phase switch positions from -1 to 1, in other words, from the left to rightmost branch of any node (expect for the leaf nodes) in the search tree.[7] At the first level, the first element of $\boldsymbol{U}(k)$ is $u_1 = -1$. After computing the intermediate radius of the partial candidate solution, it is found that the intermediate radius already exceeds that of the sphere. The branch is pruned from the search tree and there is no need to consider any

[7] The order on how the sphere decoder considers the single-phase is not unique and is determined by the user.

switching sequences where the first entry is $u_1 = -1$. After testing $u_1 = 0$ for the first element in $\boldsymbol{U}(k)$, it is found that the partial candidate solution resides within the sphere and the current branch is explored further. With the second element $u_2 = -1$, the partial switching sequence is now $\boldsymbol{U}(k) = \begin{bmatrix} 0 & -1 \end{bmatrix}^T$. It is found that the intermediate radius of the current partial candidate solution exceeds that of the sphere, and the branch is pruned. The second element is now $u_2 = 0$, with the switching sequence $\boldsymbol{U}(k) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, and it is calculated that the intermediate radius of partial candidate solution is less than that of the sphere. After branching further down the tree, and setting the final element $u_3 = -1$, the switching sequence $\boldsymbol{U}(k) = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T$ is now complete. After calculating the radius of the candidate solution, it is found to belong to the sphere. The incumbent optimal solution is updated and the sphere is tightened. The remaining two branches of the last level (i.e., $u_3 = 0$ and $u_3 = -1$) are investigated and it is found that both candidate solutions are located outside the sphere. The sphere decoder backtracks to investigate the final branch of the second level, $u_2 = 1$. After calculating that the intermediate radius exceeds that of the sphere, the sphere decoder again backtracks to explore the final branch of the root node, $u_1 = 1$. The intermediate radius is calculated, where it exceeds that of the sphere, and the final remaining branch is pruned. The incumbent optimal solution $\boldsymbol{U}_{opt}(k)$ now has a certificate for optimality.

2.6 Alternative modulation techniques

2.6.1 Space-vector modulation

Carried-based pulse-width modulation

A very popular modulation technique, that is synonymous with power electronics, is pulse-width modulation (PWM). The concept of PWM is to modulate a real-valued reference signal $\boldsymbol{u}_{ref}$ into a discrete-valued output $\boldsymbol{u}_{abc}$ that is used as gating signals for semiconductor devices. In other words, the pulse-width modulated signals directly translate to the switch positions of the converter. The harmonic content of the pulse-width modulated signals $\boldsymbol{u}_{abc}$ include a fundamental component that is ideally $\boldsymbol{u}_{ref}$, and the baseband-, carrier multiple-, and sideband harmonics that arise due to the switching nature of PWM. This section will exclusively be devoted to PWM for a three-level NPC inverter. For information on pulse-width modulation for power electronics, the reader is referred to [30].

Carrier-based pulse-width modulation (CB-PWM) for an NPC inverter is depicted in Figure 2.11, where carrier signals are used to achieve modulation. In order to achieve three output levels (i.e., $\{-1, 0, 1\}$), two carrier signals are required. The carrier signals are in phase (known as *phase disposition* [1]), where one signal from ranges -1 to 0 and while the other signal ranges from 0 to 1, as shown in Figure 2.11a. The carrier signals have a frequency of f_c, which is proportional to the switching frequency of the inverter. For a three-phase system, three references are required

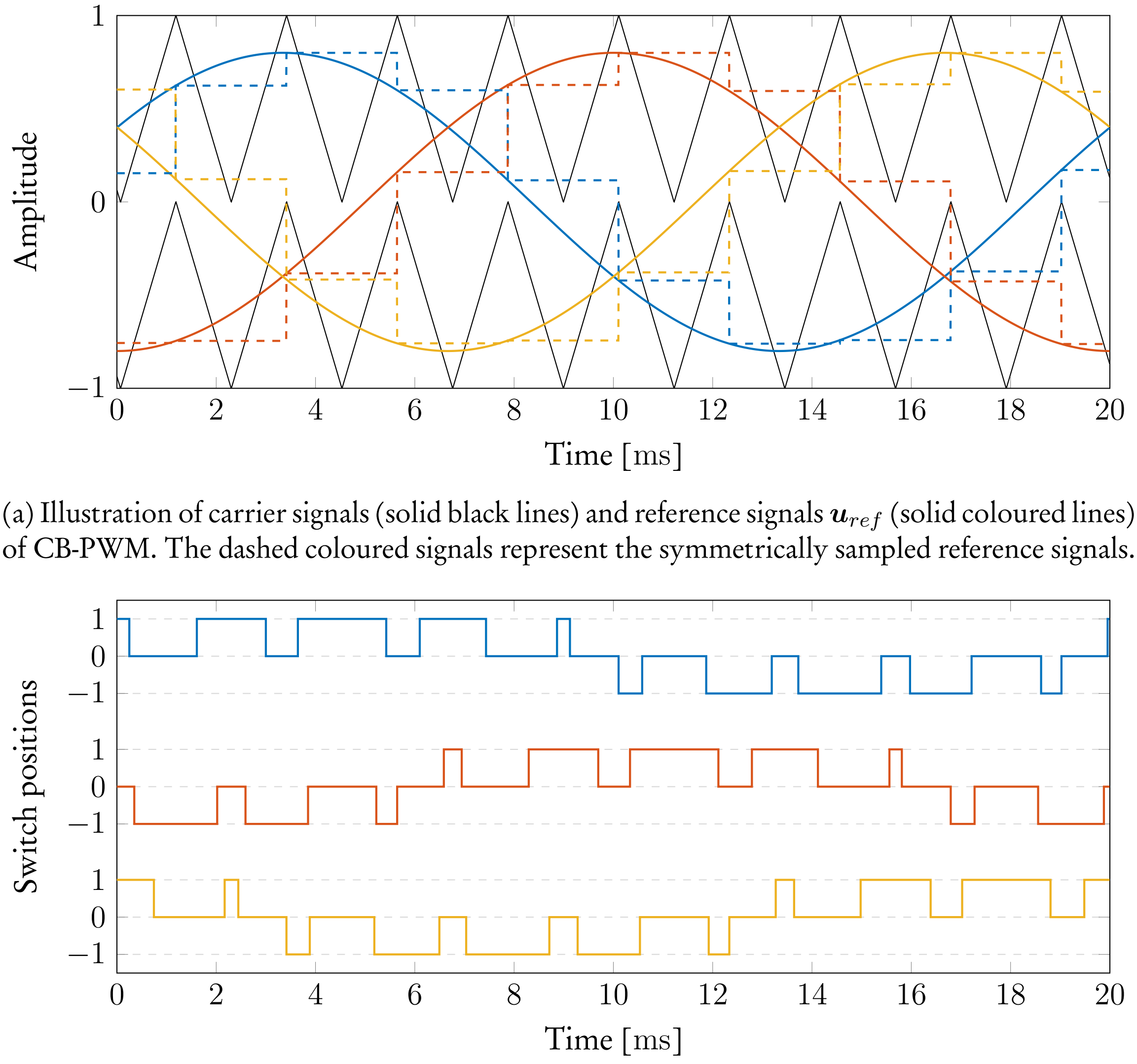
$$\boldsymbol{u}_{ref}(t) = m \begin{bmatrix} \sin(\omega_1 t) \\ \sin(\omega_1 t - \frac{2\pi}{3}) \\ \sin(\omega_1 t + \frac{2\pi}{3}) \end{bmatrix}, \tag{2.40}$$

where m is the *modulation index*. When digitally implementing CB-PWM, the reference $\boldsymbol{u}_{ref}(t)$ is a sampled signal. Two sampling techniques are usually used: *symmetric sampling* or *asymmetric sampling* [1]. In the former, the reference signal is sampled once during a carrier interval ($T_c = \frac{1}{f_c}$), for example, only at the upper triangular peaks. In the latter, the reference signal is sampled twice during the carrier interval, that is, at both of the upper and lower triangular peaks. In Figure 2.11a, symmetric sampling is illustrated.

Let c_1 and c_2 denote the top and bottom carriers, respectively. The modulation of the (sampled) reference signal happens as follows:

- if $u_{ref,x} \geq c_1$, then $u_x = 1$
- if $c_1 > u_{ref,x} > c_2$, then $u_x = 0$
- if $u_{ref,x} \leq c_2$, then $u_x = -1$,

where $x \in \{a, b, c\}$ denotes a phase. The resulting pulse-width modulated signals $\boldsymbol{u}_{abc}$, that are used as gating signals, are shown in Figure 2.11b.



(a) Illustration of carrier signals (solid black lines) and reference signals $\boldsymbol{u}_{ref}$ (solid coloured lines) of CB-PWM. The dashed coloured signals represent the symmetrically sampled reference signals.

(b) Illustration of the pulse-width modulated signals, which correspond to the switch positions $\boldsymbol{u}_{abc}$ of the inverter.

Figure 2.11: Example of pulse-width modulation for a NPC inverter where $m = 0.8$ and $f_c = 450\,\text{Hz}$.

Space-vector modulation equivalent

An attractive alternative to CB-PWM is space-vector modulation (SVM). The advantages of SVM over PWM include lower harmonic current distortion and a larger fundamental voltage [31]. The reader is referred to [7, 30] for more details on SVM. In [32], it is shown that by adding an appropriate common-mode term u_0 to the reference signal $\boldsymbol{u}_{ref}$ used during CB-PWM, equivalent switching patterns to SVM are achieved [1].

The common-mode term u_0 is given by [1]

$$u_0 = \bar{u}_0 + \frac{1}{2} - \frac{1}{2}(\min(\bar{\boldsymbol{u}}_{ref}) + \max(\bar{\boldsymbol{u}}_{ref})), \tag{2.41}$$

where

$$\bar{u}_0 = -\frac{1}{2}(\min(\boldsymbol{u}_{ref}) + \max(\boldsymbol{u}_{ref})) \tag{2.42a}$$

$$\bar{\boldsymbol{u}}_{ref} = (\boldsymbol{u}_{ref} + \bar{u}_0 + 1) \mod 1. \tag{2.42b}$$

This common-mode term and new reference signals for $m = 0.8$ are depicted in Figure 2.12. An example of the equivalent SVM is illustrated in Figure 2.13. The ease of implementation makes this technique attractive, while also offering adequate harmonic performance if the carrier frequency is not significantly less than 20 times the fundamental frequency [1].

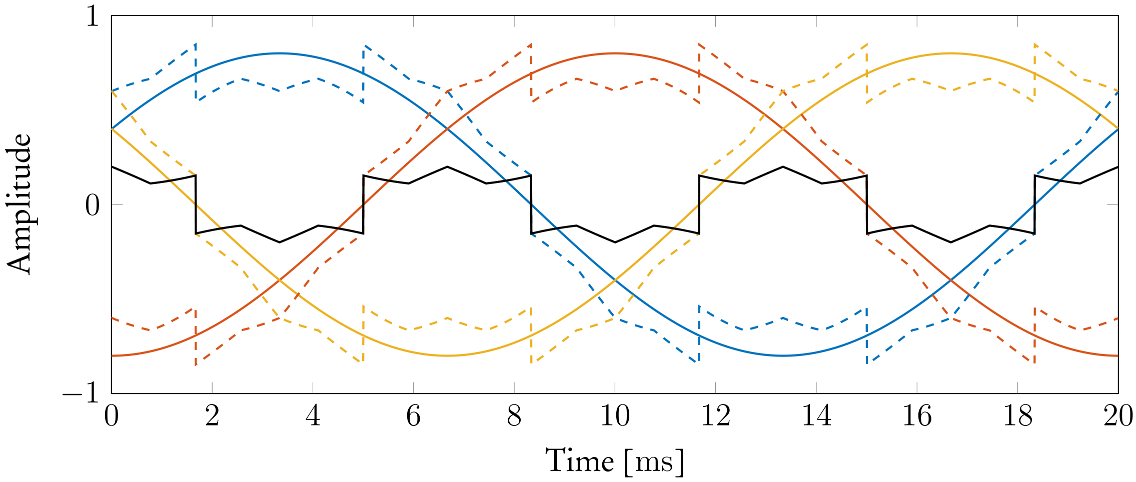
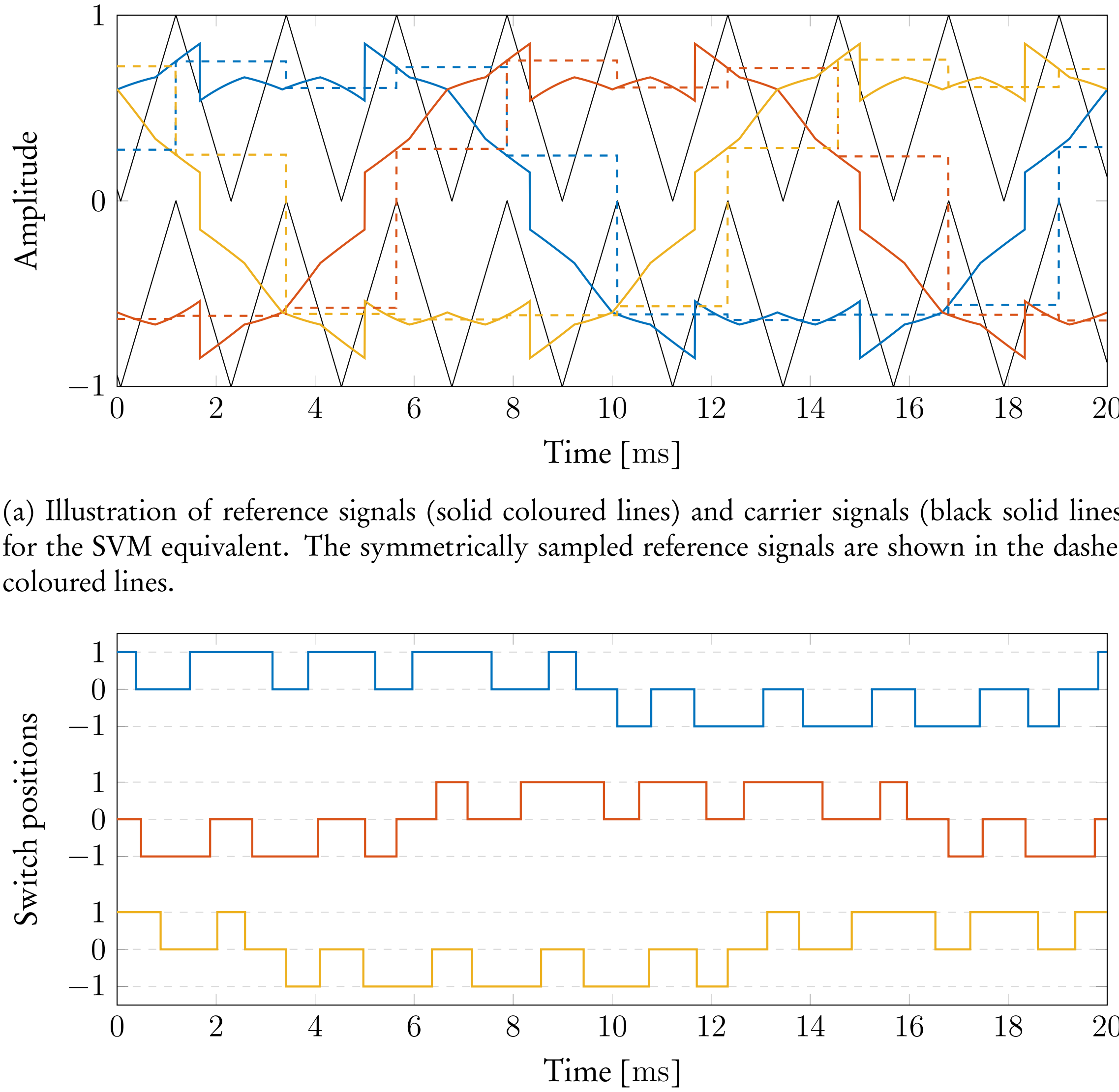


Figure 2.12: Example of the addition of the common-mode term u_0 (black solid line) to the reference signals $\boldsymbol{u}_{ref}$ (coloured solid lines), resulting in the new reference signals (coloured dashed lines) used to convert CB-PWM into SVM. The modulation index is $m = 0.8$.



(a) Illustration of reference signals (solid coloured lines) and carrier signals (black solid lines) for the SVM equivalent. The symmetrically sampled reference signals are shown in the dashed coloured lines.

(b) The resulting switching patterns equivalent to SVM.

Figure 2.13: Example of SVM equivalent for an NPC inverter where $m = 0.8$ and $f_c = 450\,\mathrm{Hz}$.

2.6.2 Optimised pulse patterns

Formulation of total harmonic distortion for a pulse pattern

For low switching frequencies well under 1 kHz, a modulation technique known as optimised pulse patterns (OPPs) is the definitive option in terms of harmonic distortions [2, 3]. The technique considers a single-phase pulse pattern (i.e., the switching pattern over a fundamental period), as shown in Figure 2.14. The idea is to calculate the angles where switching transitions should occur that will minimise the harmonic current distortion for a given modulation index and switching frequency. For a more in-depth (and well-written) analysis on OPPs, the reader is referred to [1].

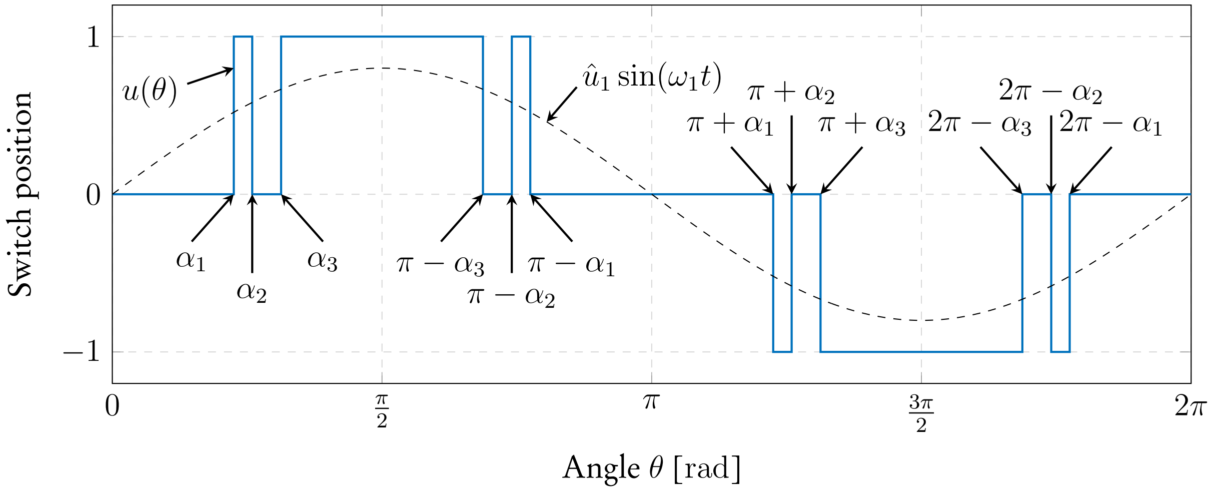


Figure 2.14: Illustration of an optimised pulse pattern. The angles α_j refer to where switching transitions occur. The pulse pattern is denoted by $u(\theta)$. The fundamental component of the pattern is given by $\hat{u}_1 \sin(\omega_1 t)$, where $\hat{u}_1$ relates to the amplitude of the fundamental component, which is desired to be equal to the modulation index m. In this illustration, the modulation index is $m = 0.8$. Adapted from [1].

In order to simplify the optimization problem underlying OPPs and to impose some harmonic characteristics, it is standard practice to enforce quarter-wave symmetry on the pulse pattern. The benefit of quarter-wave symmetry is that the even-order harmonics are zero. Furthermore, it is only required to calculate the switching angles up to a quarter of the fundamental period, i.e., up to $\frac{\pi}{2}$. The number of switching transitions in a quarter wave is determined by the *pulse number* d. Thus, the number of angles required to be calculated is equal to d, that is, angles α_j for $j \in \{1, 2, \ldots, d\}$. The pulse number d directly relates to the switching frequency, which is given by $f_{sw} = df_1$, where f_1 represents the fundamental frequency. At an angle α_j, a switching transition from u_{j-1} to u_j occurs, where $u_j \in \mathcal{U}$ denotes the switch position. Note that there are $d + 1$ switch positions u_j, since the initial switch position is included, in other words, $j \in \{0, 1, 2, \ldots, d\}$. In order to preserve the quarter-wave symmetry, the initial switch position is set to $u_0 = 0$.

Since the pulse pattern $u(\theta)$ is periodic, it can be represented by a Fourier series

$$u(\theta) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(n\theta) + b_n \sin(n\theta) \right), \tag{2.43}$$

where a_n and b_n represent the Fourier coefficients. Due to the quarter-wave symmetry, a_n coefficients are zero for all n, and b_n coefficients are nonzero only for odd n. This leads to the Fourier series being reduced to [1]

$$u(\theta) = \sum_{n=1}^{\infty} \hat{u}_n \sin(n\theta), \tag{2.44}$$

with

$$\hat{u}_n = b_n = \frac{4}{n\pi} \sum_{j=1}^{d} \Delta u_j \cos(n\alpha_j), \quad \text{if } n = 1, 3, 5, \ldots, \tag{2.45}$$

where $\hat{u}_n$ relates to the peak value of the n-th harmonic. A switching transition is denoted by $\Delta u_j = u_j - u_{j-1}$, and can be represented by $\Delta u_j = (-1)^{j+1}$ [1]. The modulation index

m determines the fundamental component $\hat{u}_1 \sin(\omega_1 t)$ (see Figure 2.14) of the pulse pattern $u(\theta)$, where $\hat{u}_1$ represents the amplitude of the fundamental component, which is desired to be equal to the modulation index ($\hat{u}_1 = m$).

When considering a three-phase inverter with switch positions $\boldsymbol{u}_{abc}(\theta)$, the pulse pattern of the thee phases are identical, but shifted by ±120°

$$\boldsymbol{u}_{abc}(\theta) = \begin{bmatrix} u(\theta) \\ u(\theta - \frac{2\pi}{3}) \\ u(\theta + \frac{2\pi}{3}) \end{bmatrix}. \tag{2.46}$$

The ±120° phase shift causes the third-order harmonics (i.e., $n = 3, 6, 9, \ldots$) to be in phase [1]. The third-order harmonics establish the common-mode voltage harmonics mentioned in Section 2.2.2. When the star connection of the load floats, the common-mode voltages do not drive phase currents.

Consider the output voltage of a single-phase NPC inverter, which is given by $v = \frac{V_d}{2}u$ when neglecting the fluctuations on the neutral-point potential. The output voltage amplitude for n-th harmonic is given by

$$\hat{v}_n = \frac{V_d}{2}\hat{u}_n. \tag{2.47}$$

When considering a resistive-inductive (RL) load, the amplitude of the n-th current harmonic is given by

$$\hat{i}_n = \frac{\hat{v}_n}{\sqrt{R^2 + (n\omega_1 L)^2}}. \tag{2.48}$$

Usually the reactance of the inductor dominates resistance, and it is customary to neglect the resistance.[8] Therefore, (2.48) is reduced to

$$\hat{i}_n = \frac{\hat{v}_n}{n\omega_1 L}. \tag{2.49}$$

The total harmonic distortion (THD) of the current is given by [1]

$$I_{THD} = \frac{1}{\hat{i}_1}\sqrt{\sum_{n\neq 1} \hat{i}_n^2} \tag{2.50}$$

where $\hat{i}_1$ is the amplitude of the fundamental component of the current.[9] By inserting (2.45), (2.47), and (2.49) into (2.50), the current THD can now be given by

$$I_{THD} = \frac{2V_d}{\pi\omega_1 L\hat{i}_1}\sqrt{\sum_{n=5,7,11,\ldots}\left(\frac{1}{n^2}\sum_{j=1}^{d}\Delta u_j \cos(n\alpha_j)\right)^2}. \tag{2.51}$$

Note that only non-third-order-odd harmonics are considered in (2.51). This is due to the quarter-wave symmetry of the pulse pattern (leading to no even-order harmonics), and ±120° phase shift between phase pulse patterns (resulting in third-order harmonics to cancel between phases, i.e., in differential-mode).

[8] Especially true for induction machines and grid-connected inverters [1].

[9] THD is most commonly defined by the root-mean-square (rms) values of the harmonics. However, as long as *all* of the harmonics in (2.50) relate to either their peak or rms value, there will be no difference in terms of THD.

Optimization problem for OPPs

It should be apparent that the minimisation of the term inside the square in (2.51) will minimise the current THD. To achieve this, an optimization problem can be stated by

$$\alpha_{j,opt} = \arg\min_{\alpha_j} \sum_{n=5,7,11,\ldots} \left(\frac{1}{n^2} \sum_{j=1}^{d} \Delta u_j \cos(n\alpha_j) \right)^2 \tag{2.52a}$$

$$\text{subject to } m = \frac{4}{\pi} \sum_{j=1}^{d} \Delta u_j \cos(\alpha_j), \tag{2.52b}$$

$$0 \le \alpha_1 \le \alpha_2 \le \ldots \le \alpha_d \le \frac{\pi}{2}. \tag{2.52c}$$

The optimization problem yields an optimal set of switching angles that minimises the cost function, and therefore current THD, in (2.52a). The constraint in (2.52b) enforces the fundamental component of the pulse pattern equal to the modulation index m. The constraint in (2.52c) imposes an ascending order on the switching angles. The optimization problem in (2.52) is a non-convex and has local minima. This makes it particularity challenging to find the global minimum for high pulse numbers d, and requires significant computational effort. Furthermore, there is no way to guarantee that a global minimum has been found. An example of the switching angles for a pulse number of $d = 3$, and for modulation indices $m \in [0, \frac{4}{\pi}]$, is shown in Figure 2.15. Interested readers are referred to [3] for a guide on how to calculate OPPs.

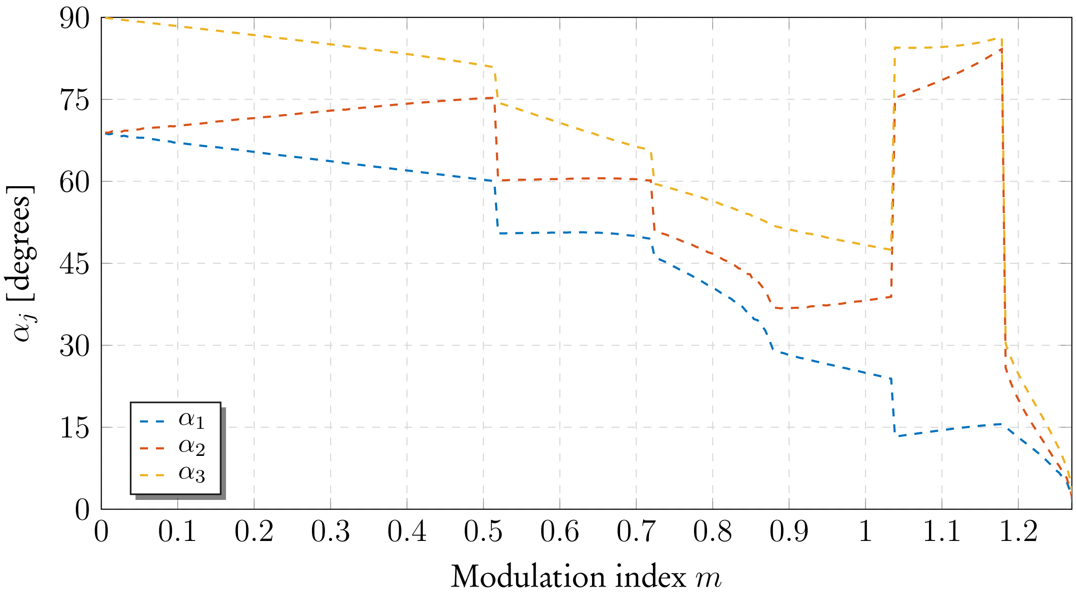


Figure 2.15: Switching angles α_j for $d = 3$ over $m \in [0, \frac{4}{\pi}]$.

Chapter 3

FPGA implementation

3.1 Introduction

This chapter discusses the implementation of long-horizon direct MPC within an FPGA. For convenience, some of the concepts and symbols introduced in Chapter 2 are repeated. The modelling and optimal control formulation for a three-phase three-level NPC inverter with an RL load are presented. The FPGA implementation of the controller is discussed in detail. A non-recursive sphere decoding algorithm, that is well-suited for FPGA implementation, is presented. The computational burden and resource usage of the implemented controller are given. The verification of the FPGA implementation is demonstrated via simulations. This chapter concludes with a summary.

3.2 Modelling and optimal control formulation

To evaluate the practical implementation of long-horizon direct MPC, a three-phase three-level NPC inverter with an RL load, as shown in Figure 3.1, will be used as the plant.

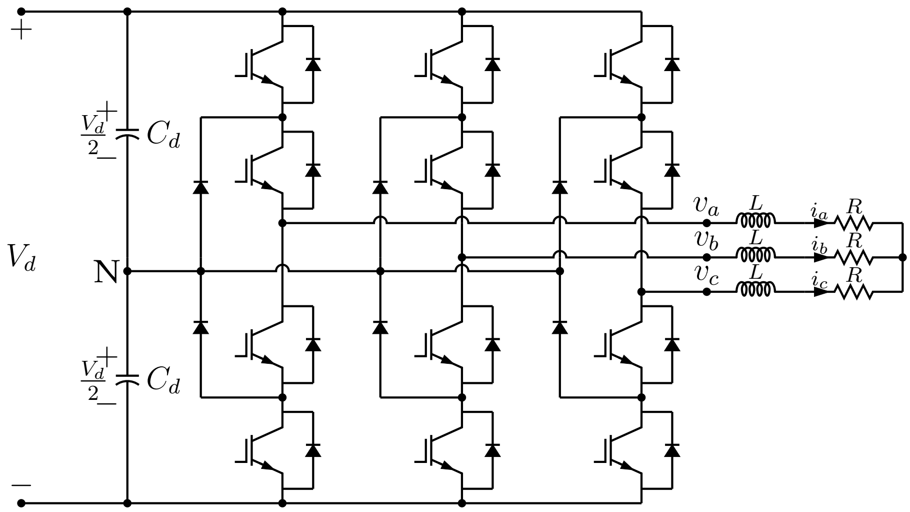


Figure 3.1: NPC inverter with an RL load.

During modelling, fluctuations on the neutral point voltage are ignored and it is assumed that the bus voltage V_d divides evenly between the two bus capacitors. Second-order effects, such as dead-time and saturation effects, are ignored.

Modelling is done by applying the Clarke transformation to the three-phase currents and using a continuous-time state-space representation of the plant. The exact discretisation is used to obtain the discrete-time state-space representation of the system. The state vector includes the $\alpha\beta$-currents

$$\boldsymbol{x}(k) = \begin{bmatrix} i_\alpha(k) & i_\beta(k) \end{bmatrix}^T, \tag{3.1}$$

and to apply reference tracking to the phase currents, the output vector is also taken as the $\alpha\beta$-currents

$$\boldsymbol{y}(k) = \boldsymbol{i}_{\alpha\beta}(k) = \begin{bmatrix} i_\alpha(k) & i_\beta(k) \end{bmatrix}^T. \tag{3.2}$$

The input vector denotes the three-phase switch positions

$$\boldsymbol{u}(k) = \begin{bmatrix} u_a(k) & u_b(k) & u_c(k) \end{bmatrix}^T, \tag{3.3}$$

where a single-phase switch position is restricted to $u_x \in \mathcal{U}$, with $x \in \{a, b, c\}$ and $\mathcal{U} = \{-1, 0, 1\}$. The input vector is restricted to $\boldsymbol{u}(k) \in \boldsymbol{\mathcal{U}}$, where $\boldsymbol{\mathcal{U}} = \mathcal{U}^3$. The discrete-time state-space representation is given by

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{3.4a}$$

$$\boldsymbol{i}_{\alpha\beta}(k+1) = \boldsymbol{C}\boldsymbol{x}(k), \tag{3.4b}$$

where

$$\boldsymbol{A} = \boldsymbol{e}^{\boldsymbol{F}T_s} \tag{3.5a}$$

$$\boldsymbol{B} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_2 - \boldsymbol{A})\boldsymbol{G} \tag{3.5b}$$

$$\boldsymbol{C} = \boldsymbol{I}_2, \tag{3.5c}$$

with T_s as the sampling period and $\boldsymbol{e}$ as the matrix exponential. The (continuous-time) state matrices are given by

$$\boldsymbol{F} = -\frac{R}{L}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{G} = \frac{V_d}{2L}\boldsymbol{K}, \tag{3.6}$$

where $\boldsymbol{K}$ is the (reduced) Clarke transformation as defined in Section 2.2.2. The full derivation of (3.4) is presented in Appendix C.

By applying reference tracking to the phase currents and penalising switching transitions, the cost function over a prediction horizon of N_p is formulated as[1]

$$J = \sum_{l=k}^{k+N_p-1} \left\| \boldsymbol{i}_{abc,ref}(l+1) - \boldsymbol{i}_{abc}(l+1) \right\|_2^2 + \lambda_u \left\| \Delta\boldsymbol{u}(l) \right\|_2^2, \tag{3.7}$$

where $\Delta\boldsymbol{u}(l) = \boldsymbol{u}(l) - \boldsymbol{u}(l-1)$, and λ_u is the weighting factor on the switching transitions. Knowing that $\boldsymbol{K}\boldsymbol{K}^{-1} = \boldsymbol{I}_2$, $\frac{3}{2}\boldsymbol{K} = (\boldsymbol{K}^{-1})^T$ and $\left\|\boldsymbol{\xi}\right\|_2^2 = \boldsymbol{\xi}^T\boldsymbol{\xi}$, the tracking error can be transformed to the orthogonal reference frame by

$$\begin{aligned} \left\| \boldsymbol{i}_{abc,e}(l+1) \right\|_2^2 &= (\boldsymbol{K}^{-1}\boldsymbol{i}_{\alpha\beta,e}(l+1))^T(\boldsymbol{K}^{-1}\boldsymbol{i}_{\alpha\beta,e}(l+1)) \\ &= \boldsymbol{i}_{\alpha\beta,e}(l+1)^T(\boldsymbol{K}^{-1})^T\boldsymbol{K}^{-1}\boldsymbol{i}_{\alpha\beta,e}(l+1) \\ &= \frac{3}{2}\left\| \boldsymbol{i}_{\alpha\beta,ref}(l+1) - \boldsymbol{i}_{\alpha\beta}(l+1) \right\|_2^2. \end{aligned} \tag{3.8}$$

[1] The argument of J, as used in Section 2.3.4, is dropped for convenience.

By omitting the constant factor $\frac{3}{2}$ in (3.8), the cost function now becomes[2]

$$J = \sum_{l=k}^{k+N_p-1} \left\| \boldsymbol{i}_{ref}(l+1) - \boldsymbol{i}(l+1) \right\|_2^2 + \lambda_u \left\| \Delta \boldsymbol{u}(l) \right\|_2^2. \tag{3.9}$$

With $\boldsymbol{U}(k) = \begin{bmatrix} \boldsymbol{u}^T(k) & \boldsymbol{u}^T(k+1) & \cdots & \boldsymbol{u}^T(k+N_p-1) \end{bmatrix}^T$ introduced as the switching sequence over the prediction horizon, the optimization problem is stated as

$$\boldsymbol{U}_{opt}(k) = \arg \min_{\boldsymbol{U}(k)} \quad J \tag{3.10a}$$

$$\text{subject to } \boldsymbol{U}(k) \in \mathbb{U}, \tag{3.10b}$$

where $\mathbb{U} = \boldsymbol{\mathcal{U}}^{N_p}$.

As explained in Section 2.4, (3.10) can be reformulated as an ILS problem

$$\boldsymbol{U}_{opt}(k) = \arg \min_{\boldsymbol{U}(k)} \quad \left\| \bar{\boldsymbol{U}}_{unc}(k) - \boldsymbol{H}\boldsymbol{U}(k) \right\|_2^2 \tag{3.11a}$$

$$\text{subject to } \boldsymbol{U}(k) \in \mathbb{U}, \tag{3.11b}$$

where

$$\bar{\boldsymbol{U}}_{unc}(k) = \boldsymbol{H}\boldsymbol{U}_{unc}(k) \tag{3.12}$$

$$\boldsymbol{U}_{unc}(k) = -\boldsymbol{Q}^{-1}\boldsymbol{\Theta}(k) \tag{3.13}$$

$$\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{Q}, \tag{3.14}$$

with

$$\boldsymbol{Q} = \boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^T\boldsymbol{S} \tag{3.15}$$

$$\boldsymbol{\Theta}(k) = ([\boldsymbol{\Gamma}\boldsymbol{i}(k) - \boldsymbol{Y}_{ref}(k)]^T\boldsymbol{\Upsilon} - \lambda_u[\boldsymbol{E}\boldsymbol{u}(k-1)]^T\boldsymbol{S})^T. \tag{3.16}$$

The reference current over the prediction horizon is denoted by $\boldsymbol{Y}_{ref}(k)$, and matrices $\boldsymbol{\Gamma} \in \mathbb{R}^{2N_p \times 2}$, $\boldsymbol{\Upsilon} \in \mathbb{R}^{2N_p \times 3N_p}$, $\boldsymbol{S} \in \mathbb{R}^{3N_p \times 3N_p}$, and $\boldsymbol{E} \in \mathbb{R}^{3N_p \times 3}$ are defined in Section 2.3.4.

3.3 VHDL implementation

An Altera Cyclone® V 5CSEMA5F31C6N FPGA is used to implement the controller. The FPGA has 32 070 adaptive logic modules (ALMs) and 87 variable-precision digital signal processing (DSP) blocks. The FPGA includes an ARM Cortex™ processor.[3] The Cyclone® series is described as "*. . . optimized for low-cost, high-volume systems*" [33]. Management of resources, especially multipliers, is of the utmost importance to achieve long horizons.[4] Moreover, the structure of the code has to be considered in order to achieve a clock frequency that can solve the optimization problem within the sampling period.

[2] From here on, the $\alpha\beta$ subscript is dropped for convenience.

[3] It is worth mentioning that the ARM processor use some of the available space on the die, and an FPGA-only solution will have more FPGA resources.

[4] Multipliers are provided by the DSP blocks.

3.3.1 FPGA preliminaries

By coding in **V**ery High Speed Integrated Circuit (VHSIC) **H**ardware **D**escription **L**anguage (VHDL), a behavioural description of a program is given. Quartus®, Altera software, synthesises the behavioural description into a netlist that is used to program the logic blocks of the FPGA. The code structure will determine how the logic blocks are used. Where possible, the implementation must be done in such a way that the same logic blocks (especially the multipliers) can be used for different equations. This will be demonstrated in Section 3.3.3 and 3.3.4. Recursion should be avoided in VHDL, as will be explained in Section 3.3.5.

Although VHDL do not natively support matrices, it does support the use of arrays. A matrix can be stored in an array from the first to the last column, for example,

$$\boldsymbol{M} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \Longleftrightarrow \begin{bmatrix} a & b & c & d & e & f & g & h & i \end{bmatrix}.$$

Accessing the element in the first row and second column in matrix indexing is indicated by $M_{(1,2)}$, while in the array index it will be M_4. For simplicity, matrix indexing will be used in this chapter.

Attention must be given to the timing requirements of the FPGA, especially as the program becomes more complex and arithmetic intensive. The timing requirements are due to delays that occur in the logic elements of the FPGA. Quartus® includes a timing analysis tool, known as TimeQuest Timing Analyzer, that determines conditions that should be satisfied. One of these conditions is the maximum clock frequency at which the FPGA can operate reliably. The critical path (i.e., the longest) determines the conditions that must be satisfied. If the timing requirements are not met, errors in the program could occur.

A real value can either be represented as a fixed-point or floating-point number. Fixed-point arithmetic has the advantage of being quicker than floating-point arithmetic, and is well-suited for most applications. When realising that quantization errors and model uncertainties are present in the practical system, the fixed-point package is more than sufficient for this purpose. The accuracy of the fractional part is determined by the number of bits assigned. A compromise is made between accuracy and resource usage. The maximum obtainable clock frequency is also influenced by the number of assigned bits.

3.3.2 Delay compensation

Ideally, once a measurement is taken at time-step k, the optimal solution $\boldsymbol{U}_{opt}(k)$ is instantaneously calculated and the optimal switch position $\boldsymbol{u}_{opt}(k)$ is applied at the current sampling instant, as illustrated in Figure 3.2. In simulation this is possible. However, in a practical scenario there will be a *variable* computational time t_c required to calculate the optimal switching sequence. The computational time t_c is dependent on how quick the sphere decoder traverses the search tree. Note, that once the optimal switching sequence is calculated, switching is not permitted between sampling instants. The prediction model does not account for any switching between sampling instants, and by applying a switch transition the prediction model becomes invalid. This is emphasised in Figure 3.3.

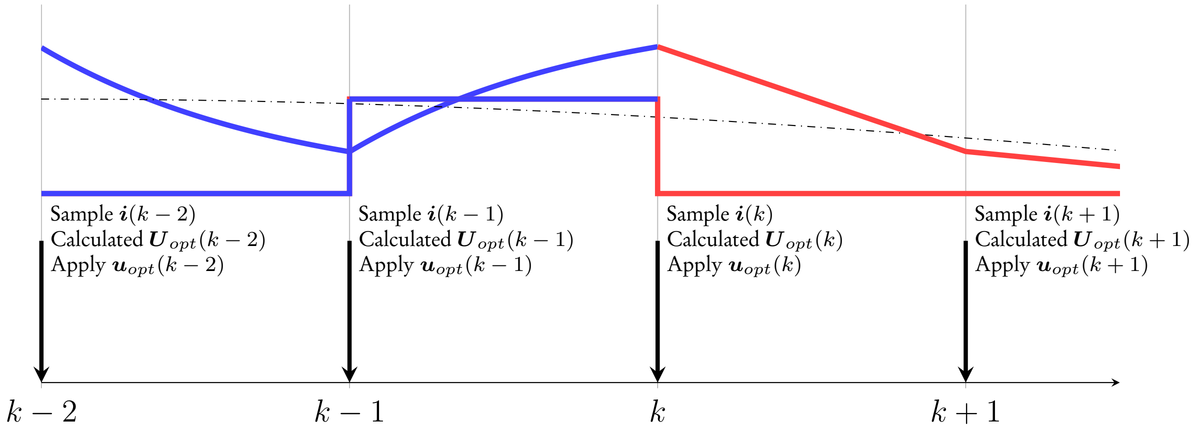

Sample $\boldsymbol{i}(k-2)$
Calculated $\boldsymbol{U}_{opt}(k-2)$
Apply $\boldsymbol{u}_{opt}(k-2)$
Sample $\boldsymbol{i}(k-1)$
Calculated $\boldsymbol{U}_{opt}(k-1)$
Apply $\boldsymbol{u}_{opt}(k-1)$
Sample $\boldsymbol{i}(k)$
Calculated $\boldsymbol{U}_{opt}(k)$
Apply $\boldsymbol{u}_{opt}(k)$
Sample $\boldsymbol{i}(k+1)$
Calculated $\boldsymbol{U}_{opt}(k+1)$
Apply $\boldsymbol{u}_{opt}(k+1)$
$k-2$
$k-1$
k
$k+1$

Figure 3.2: The idealised scenario: zero computational delay.

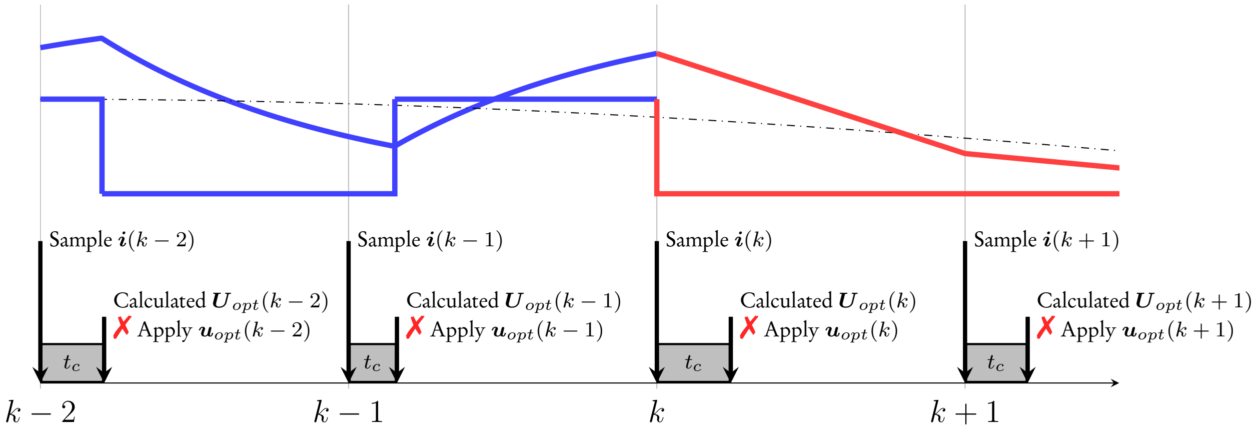

Sample $\boldsymbol{i}(k-2)$
Calculated $\boldsymbol{U}_{opt}(k-2)$
✗ Apply $\boldsymbol{u}_{opt}(k-2)$
Sample $\boldsymbol{i}(k-1)$
Calculated $\boldsymbol{U}_{opt}(k-1)$
✗ Apply $\boldsymbol{u}_{opt}(k-1)$
Sample $\boldsymbol{i}(k)$
Calculated $\boldsymbol{U}_{opt}(k)$
✗ Apply $\boldsymbol{u}_{opt}(k)$
Sample $\boldsymbol{i}(k+1)$
Calculated $\boldsymbol{U}_{opt}(k+1)$
✗ Apply $\boldsymbol{u}_{opt}(k+1)$
t_c
t_c
t_c
t_c
$k-2$
$k-1$
k
$k+1$

Figure 3.3: The practical scenario: computational delay of t_c.

Since the computational time t_c is variable, it cannot be directly addressed. Instead, a fixed delay of one sampling interval is assumed, $T_d = T_s$. Now, instead of solving the optimization problem and finding $\boldsymbol{U}_{opt}(k)$ for time-step k, the optimization problem is solved for the *next* time-step $k+1$, in other words, finding $\boldsymbol{U}_{opt}(k+1|k)$. At the sampling instant, the switch position that was calculated at the *previous* sampling instant $\boldsymbol{u}_{opt}(k|k-1)$ is applied. Therefore, the current at the next sampling instant can be extrapolated to[5]

$$\boldsymbol{i}(k+1|k) = \boldsymbol{A}\boldsymbol{i}(k) + \boldsymbol{B}\boldsymbol{u}_{opt}(k|k-1). \tag{3.17}$$

The optimization problem is now solved for the next time step based on the measurements at the current time-step,

$$\boldsymbol{U}_{opt}(k+1|k) = \arg \min_{\boldsymbol{U}(k+1)} \left\| \bar{\boldsymbol{U}}_{unc}(k+1|k) - \boldsymbol{H}\boldsymbol{U}(k+1) \right\|_2^2 \tag{3.18a}$$

$$\text{subject to } \boldsymbol{U}(k+1) \in \mathbb{U}. \tag{3.18b}$$

Once the optimal solution $\boldsymbol{U}_{opt}(k+1|k)$ is calculated within the sampling period, the optimal switch position $\boldsymbol{u}_{opt}(k+1|k)$ is only applied once the next sampling instant is reached. This procedure is illustrated in Figure 3.4.

[5] The notation $\boldsymbol{i}(k+m|k+m-1)$ denotes the value of the current at time-step $k+m$ calculated at $\boldsymbol{i}(k+m-1)$. $\boldsymbol{u}_{opt}(k+m-1|k+m-2)$ is the optimal switch position at time-step $k+m-1$ based on the measurements at $k+m-2$.

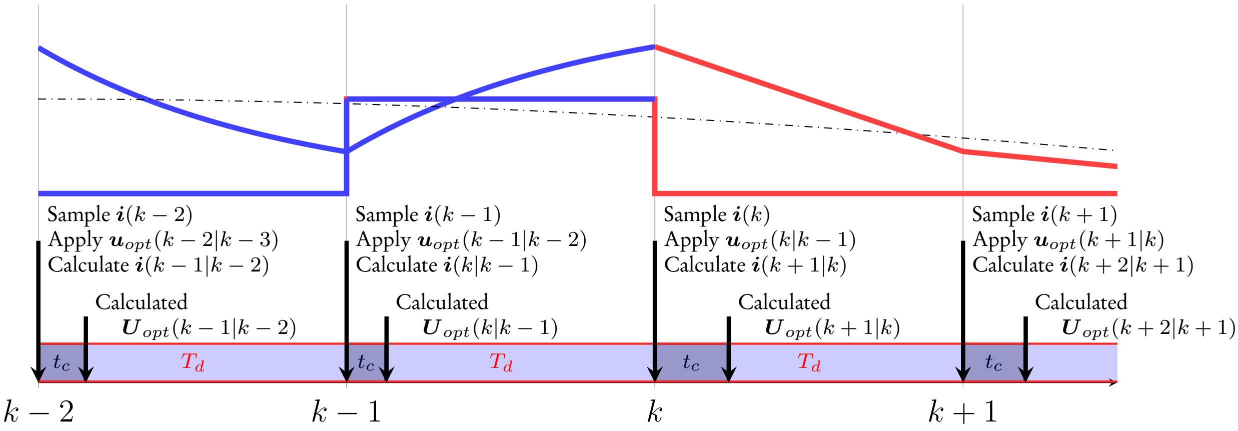

Sample $\boldsymbol{i}(k-2)$
Apply $\boldsymbol{u}_{opt}(k-2|k-3)$
Calculate $\boldsymbol{i}(k-1|k-2)$
Calculated $\boldsymbol{U}_{opt}(k-1|k-2)$
Sample $\boldsymbol{i}(k-1)$
Apply $\boldsymbol{u}_{opt}(k-1|k-2)$
Calculate $\boldsymbol{i}(k|k-1)$
Calculated $\boldsymbol{U}_{opt}(k|k-1)$
Sample $\boldsymbol{i}(k)$
Apply $\boldsymbol{u}_{opt}(k|k-1)$
Calculate $\boldsymbol{i}(k+1|k)$
Calculated $\boldsymbol{U}_{opt}(k+1|k)$
Sample $\boldsymbol{i}(k+1)$
Apply $\boldsymbol{u}_{opt}(k+1|k)$
Calculate $\boldsymbol{i}(k+2|k+1)$
Calculated $\boldsymbol{U}_{opt}(k+2|k+1)$
t_c
T_d
$k-2$
$k-1$
k
$k+1$

Figure 3.4: Illustration of delay compensation.

A second delay is introduced due to the acquisition and conversion time of the ADC. Sampling is done at a fixed time ΔT before the sampling instant, where ΔT is greater than the delay introduced by the ADC. By using a discrete-time state-space representation discretised with an interval of ΔT, the current value at the sample instant can be extrapolated to

$$\boldsymbol{i}(k) = \boldsymbol{A}_{\Delta T}\boldsymbol{i}(k-\Delta T) + \boldsymbol{B}_{\Delta T}\boldsymbol{u}_{opt}(k-1), \tag{3.19}$$

where

$$\boldsymbol{A}_{\Delta T} = \boldsymbol{e}^{\boldsymbol{F}\Delta T} \tag{3.20a}$$
$$\boldsymbol{B}_{\Delta T} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_2 - \boldsymbol{A}_{\Delta T})\boldsymbol{G}. \tag{3.20b}$$

3.3.3 Unconstrained solution

Efficiently calculating the unconstrained solution is cardinal to achieve long horizons. First $\boldsymbol{\Theta}(k)$ has to be calculated, and is given by (from Section 2.4)

$$\boldsymbol{\Theta}(k) = ([\boldsymbol{\Gamma}\boldsymbol{i}(k) - \boldsymbol{Y}_{ref}(k)]^T\boldsymbol{\Upsilon} - \lambda_u[\boldsymbol{E}\boldsymbol{u}(k-1)]^T\boldsymbol{S})^T. \tag{3.21}$$

By calculating all entries of $\boldsymbol{\Theta}(k)$ in one clock cycle, i.e., the linear equations in the rows, a significant amount of resources will be used. Moreover by doing so, it is highly likely that the maximum achievable clock frequency will be low due to the intensive arithmetic in one clock cycle. Instead, at every clock cycle one entry of $\boldsymbol{\Theta}(k)$ is calculated. Since every entry is an inner product with the same form, the coefficients can simply be changed at every clock cycle. This will significantly reduce the resource usage, since the same multipliers are used to calculate all the entries of $\boldsymbol{\Theta}(k)$.

After manipulating (3.21), the j'th entry of $\boldsymbol{\Theta}(k)$ is given by

$$\begin{aligned}\Theta_j = {} & \Upsilon^T_{(j,1)}(\Gamma_{(1,1)}i_1 + \Gamma_{(1,2)}i_2 - Y_{ref,1}) + \Upsilon^T_{(j,2)}(\Gamma_{(2,1)}i_1 + \Gamma_{(2,2)}i_2 - Y_{ref,2}) \\ & + \Upsilon^T_{(j,3)}(\Gamma_{(3,1)}i_1 + \Gamma_{(3,2)}i_2 - Y_{ref,3}) + \ldots + \Upsilon^T_{(j,2N_p)}(\Gamma_{(2N_p,1)}i_1 + \Gamma_{(2N_p,2)}i_2 - Y_{ref,2N_p}) \\ & - \lambda_u Z_j U(k-1)_j, \end{aligned} \tag{3.22}$$

with

$$\boldsymbol{Z} = \begin{bmatrix}1 & 1 & 1 & 0 & \cdots & 0\end{bmatrix}^T,$$

where ξ_j denotes the j'th element of a vector and $\zeta_{(i,j)}$ denotes the (i,j)'th entry of a matrix. $\boldsymbol{Z}$ is introduced by noting that $\boldsymbol{\Theta}(k)$ is a function of only the first element $\boldsymbol{u}(k-1)$ in the previous switching sequence $\boldsymbol{U}(k-1)$.

The unconstrained solution and transformed unconstrained solution are given by (from Section 2.4.1)

$$\boldsymbol{U}_{unc}(k) = -\boldsymbol{Q}^{-1}\boldsymbol{\Theta}(k) \tag{3.23}$$

$$\bar{\boldsymbol{U}}_{unc}(k) = \boldsymbol{H}\boldsymbol{U}_{unc}(k), \tag{3.24}$$

respectively. It is worth mentioning that if only the educated guess is used, there is no need to calculate $\boldsymbol{U}_{unc}(k)$, and $\bar{\boldsymbol{U}}_{unc}(k)$ can directly be calculated by pre-multiplying $\boldsymbol{H}$ and $-\boldsymbol{Q}^{-1}$. If the Babai estimate is used, the unconstrained solution $\boldsymbol{U}_{unc}(k)$ (in the original vector space) is required in order to round to the nearest switching vector in $\mathbb{U}$. The calculation of the unconstrained solution follows the same principles as calculating $\boldsymbol{\Theta}(k)$.

The j'th entry of $\boldsymbol{U}_{unc}(k)$ is given by

$$u_{unc,j} = -Q^{-1}_{(j,1)}\Theta_1 - Q^{-1}_{(j,2)}\Theta_2 - Q^{-1}_{(j,3)}\Theta_3 - \ldots - Q^{-1}_{(j,3N_p)}\Theta_{3N_p}. \tag{3.25}$$

By noting that $\boldsymbol{U}_{unc}(k)$ and $\bar{\boldsymbol{U}}_{unc}(k)$ has an inner product of the same form, only the coefficients have to be changed in (3.25). Therefore, the j'th entry of $\bar{\boldsymbol{U}}_{unc}(k)$ can be calculated by

$$\bar{u}_{unc,j} = -(HQ^{-1})_{(j,1)}\Theta_1 - (HQ^{-1})_{(j,2)}\Theta_2 - (HQ^{-1})_{(j,3)}\Theta_3 - \ldots - (HQ^{-1})_{(j,3N_p)}\Theta_{3N_p}, \tag{3.26}$$

with no extra expense to the multipliers. The coding implementation for the calculation of the unconstrained solution is shown in Algorithm 3. Although not shown, the Babai estimate is calculated as the entries of the unconstrained solution are calculated.

Algorithm 3 Unconstrained solution

```
for $j = 1$ to $3N_p$ do
    $\Theta_j = \Upsilon^T_{(j,1)}(\Gamma_{(1,1)}i_1 + \Gamma_{(1,2)}i_2 - Y_{ref,1}) + \ldots - \lambda_u Z_j U(k-1)_j$
end for
for $q = 1$ to $2$ do
    if $q = 1$ then
        $\boldsymbol{M} = -\boldsymbol{Q}^{-1}$
    else if $q = 2$ then
        $\boldsymbol{M} = -\boldsymbol{H}\boldsymbol{Q}^{-1}$
    end if
    for $j = 1$ to $3N_p$ do
        $u_{temp,j} = M_{(j,1)}\Theta_1 + M_{(j,2)}\Theta_2 + \ldots + M_{(j,3N_p)}\Theta_{3N_p}$
    end for
    if $q = 1$ then
        $\boldsymbol{U}_{unc} = \boldsymbol{U}_{temp}$
    else if $q = 2$ then
        $\bar{\boldsymbol{U}}_{unc} = \boldsymbol{U}_{temp}$
    end if
end for
```

3.3.4 Initial radius

Babai estimate versus educated guess

It is highly recommended that both the Babai estimate and the educated guess are evaluated when determining the initial radius. Under steady-state conditions, the educated guess proves to be an excellent initial guess [1]. As explained in Section 2.5, the educated guess is dependent on the previous optimal solution (in other words, information of the *previous* sampling instant). The guess relies on the assumption that the discarded entries of $\boldsymbol{U}_{opt}(k-1)$ should look similar to the new (now shifted) entries in $\boldsymbol{U}_{opt}(k)$. However, during transients, this does not always hold true and the new optimal solution can look substantially different. This will cause an enormous initial radius that encloses a great number of lattice points, as demonstrated in Figure 3.5.

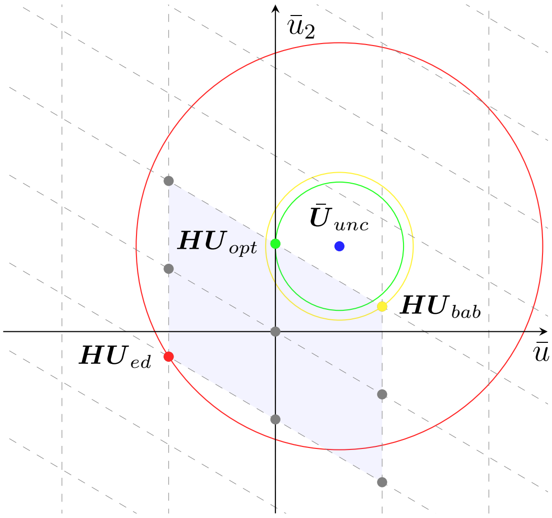


Figure 3.5: Demonstration of the initial radii for the Babai estimate $\boldsymbol{U}_{bab}(k)$ and the educated guess $\boldsymbol{U}_{ed}(k)$ during a transient. The shaded area represents the convex hull of the lattice.

The Babai estimate, however, is reliant on the unconstrained solution $\boldsymbol{U}_{unc}(k)$ which is calculated at the *current* sampling instant. Since the unconstrained solution contains information about where the optimal solution is located in the lattice, the Babai estimate will be in the vicinity of the optimal solution (see Figure 3.5). Unfortunately, in the case where the unconstrained solution is located far from the convex hull of the truncated lattice (see Section 2.4.2 for lattice description), the sphere decoder will visit many nodes of the search tree. It is worth stressing that aforementioned problems intensify at higher dimensions, where the geometry of the lattice is non-trivial. Note that the skewness of the lattice will also determine how quickly the sphere decoder will traverse the search tree. The interested readers are referred to [34] where a proposal is made to efficiently reduce the computational effort during transients.

Calculation of initial radius

The radii for the Babai estimate and the educated guess are given by

$$\rho(k) = \left\| \bar{\boldsymbol{U}}_{unc}(k) - \boldsymbol{HU}_{ini}(k) \right\|_2, \tag{3.27}$$

where $\boldsymbol{U}_{ini}(k) \in \{\boldsymbol{U}_{ed}(k), \boldsymbol{U}_{bab}(k)\}$. The initial radius will be the minimum of the Babai estimate and the educated guess

$$\rho_{ini}(k) = \min\{\rho_{bab}(k), \rho_{ed}(k)\}. \tag{3.28}$$

By squaring (3.27), the squared radius is given by

$$\rho^2(k) = \overbrace{(\bar{u}_{unc,1} - h_{(1,1)}u_{ini,1})^2}^{\rho_1^2} + \overbrace{(\bar{u}_{unc,2} - h_{(2,1)}u_{ini,1} - h_{(2,2)}u_{ini,2})^2}^{\rho_2^2} + \ldots$$

$$\overbrace{(\bar{u}_{unc,3N_p} - h_{(3N_p,1)}u_{ini,1} - h_{(3N_p,2)}u_{ini,2} - \ldots - h_{(3N_p,3N_p)}u_{ini,3N_p})^2}^{\rho_{3N_p}^2} \tag{3.29a}$$

$$= \sum_{j=1}^{3N_p}(\bar{u}_{unc,j} - h_{(j,1)}u_{ini,1} - h_{(j,2)}u_{ini,2} - \ldots - h_{(j,3N_p)}u_{ini,3N_p})^2. \tag{3.29b}$$

From (3.29a) it is apparent the polynomial of the squared term of the bottom level, that is, $\rho^2_{3N_P}$, can be used to calculate the squared term of any level by simply changing the coefficients. The squared radius is given by the squared terms accumulated up to the final level, as shown in (3.29b). As with the unconstrained solution, only one squared term is calculated per clock cycle. The code implementation for the initial radius is shown in Algorithm 4.

Algorithm 4 Initial radius

```
for $q = 1$ to $2$ do
    $\rho^2_{temp,prev} = 0$
    if $q = 1$ then
        $\boldsymbol{U}_{ini} = \boldsymbol{U}_{bab}$
    else if $q = 2$ then
        $\boldsymbol{U}_{ini} = \boldsymbol{U}_{ed}$
    end if
    for $j = 1$ to $3N_p$ do
        $\rho^2_{temp} = (\bar{u}_{unc,j} - h_{(j,1)}u_{ini,1} - h_{(j,2)}u_{ini,2} - \ldots - h_{(j,3N_p)}u_{ini,3N_p})^2 + \rho^2_{temp,prev}$
        $\rho^2_{temp,prev} = \rho^2_{temp}$
    end for
    if $q = 1$ then
        $\rho^2_{bab} = \rho^2_{temp}$
    else if $q = 2$ then
        $\rho^2_{ed} = \rho^2_{temp}$
    end if
end for
if $\rho^2_{ed} \leq \rho^2_{bab}$ then
    $\rho_{ini} = \rho_{ed}$
else
    $\rho^2_{ini} = \rho^2_{bab}$
end if
```

3.3.5 Sphere decoding algorithm

Non-recursive sphere decoding algorithm

Recall that the sphere decoding algorithm (SDA) proposed in [10] employs recursion (see Algorithm 2 in Section 2.5.2). Implementing recursion in VHDL is not as benign as in common programming languages. Whereas languages such as C/C++ use a stack to implement recursion, VHDL will synthesise the same algorithm n-times (the maximum recursion depth).

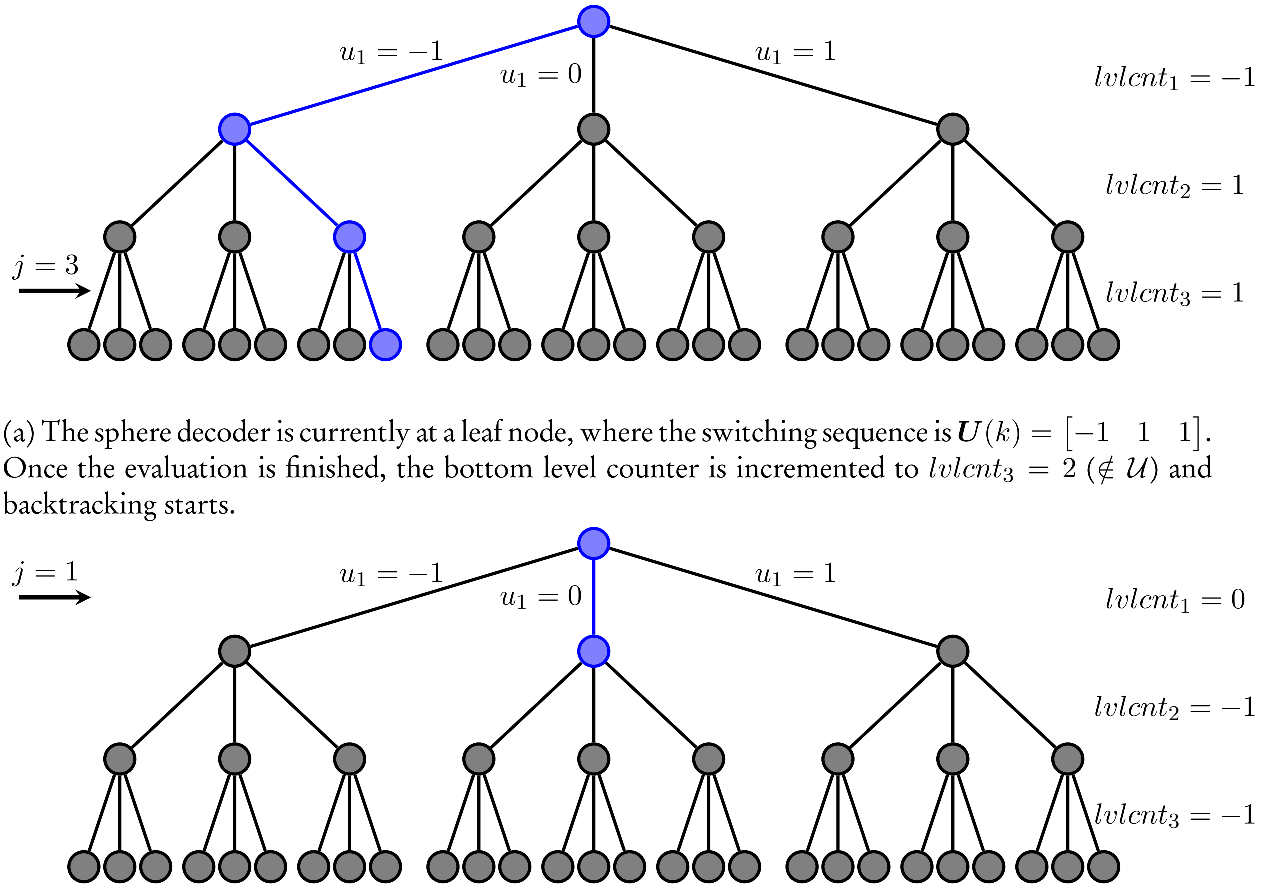
This results in n-times the amount of resources being used. Due to the complexity of the SDA, using recursion will come at a great expense of resources and should be avoided. A non-recursive SDA is proposed in Algorithm 5. The algorithm fundamentally functions the same as the recursive variant, but storage and backtracking methods are realised differently.

Algorithm 5 Non-Recursive Sphere Decoder

```
 1: function $\boldsymbol{U}_{opt} = \textsc{SphDec}(\boldsymbol{U}, 3N_p, \rho^2, \bar{\boldsymbol{U}}_{unc})$
 2:     Initialize:
 3:     $j = 1$
 4:     set each element in $\boldsymbol{lvlcnt} = -1$
 5:     set each element in $\boldsymbol{d}^2 = 0$
 6:     while solutionfound $= 0$ do
 7:         $u_j = lvlcnt_j$
 8:         $d'^2 = \left\|\bar{u}_{unc,j} - \boldsymbol{H}_{(j,1:j)}\boldsymbol{U}_{1:j}\right\|_2^2 + d_j^2$
 9:         if $\rho^2 \geq d'^2$ then
10:             if $j = 3N_p$ then
11:                 $\boldsymbol{U}_{opt} = \boldsymbol{U}$
12:                 $\rho^2 = d'^2$
13:                 $lvlcnt_j + +$
14:             else
15:                 $j + +$
16:                 $d_j^2 = d'^2$
17:             end if
18:         else
19:             $lvlcnt_j + +$
20:         end if
21:         for $q = 3N_p$ downto $2$ do
22:             if $lvlcnt_q > 1$ then
23:                 $lvlcnt_q = -1$
24:                 $j = q - 1$
25:                 $lvlcnt_j + +$
26:             end if
27:         end for
28:         if $lvlcnt_1 > 1$ then
29:             solutionfound = 1
30:         end if
31:     end while
32: end function
```

The squared terms of the ILS problem are stored in the array $\boldsymbol{d}^2$, where d_j^2 are the accumulated squared terms up to (not including) the j'th level. The algorithm traverses the search tree in the same order as the non-recursive algorithm, but uses two different pointers to keep track of the explored branches. The first pointer is known as the *level pointer*, denoted by j, and keeps track of which level of the search tree is being explored. The second pointer is known as the *level counter*, denoted by $lvlcnt_j$, and keeps track of what branches of the search tree is being explored. There are $3N_p$ of such pointers stored in an array $\boldsymbol{lvlcnt}$.

Up to line 20 in Algorithm 5, the two SDAs are relativity similar. The biggest difference is how backtracking is achieved. Instead of popping an element from a stack[6], the algorithm evaluates $3N_p - 1$ *if-statements* (line 21–26) in order to realise backtracking. Backtracking works as follow: if a branch of a node is more than 1, which means that the switch position does not belong to the single-phase set $\mathcal{U}$, the branch is set back to -1.[7] The current level is decreased and the next branch of the node is evaluated. This procedure is done from bottom to top. Figure 3.6 illustrates the backtracking procedure. Once the top-level counter exceeds $lvlcnt_1 > 1$, it means that the entire search tree has been evaluated and the sphere decoder terminates. A certificate for optimality is obtained.



(a) The sphere decoder is currently at a leaf node, where the switching sequence is $\boldsymbol{U}(k) = \begin{bmatrix} -1 & 1 & 1 \end{bmatrix}$. Once the evaluation is finished, the bottom level counter is incremented to $lvlcnt_3 = 2$ ($\notin \mathcal{U}$) and backtracking starts.

(b) The bottom-level counter is now set back to $lvlcnt_3 = -1$ and the current level is decreased to $j = 2$. After the second level counter is incremented, since the current branch has been explored, the level counter is now $lvlcnt_2 = 2$ ($\notin \mathcal{U}$). The sphere decoder sets $lvlcnt_2 = -1$ and backtracks to the top level, where the level counter is incremented to $lvlcnt_1 = 0$. Backtracking is finished, and the (partial) switching sequence now is $\boldsymbol{U}(k) = \begin{bmatrix} 0 \end{bmatrix}$.

Figure 3.6: Backtracking example. The blue lines represent the branches that are currently being explored. This relates to the switching sequence that is assembled.

The squared term calculation in line 8 is achieved in a similar way to that of line 9 in Algorithm 4. Note that the elements in $\boldsymbol{U}(k)$ will never be empty during practical implementation, but due to the triangularity of the $\boldsymbol{H}$ matrix, elements at lower levels of the search tree (i.e., the levels more than j) will not be considered.

[6] The term stack refers to a programming stack, and is accessed in the Last In, First Out method.

[7] For convenience, $\mathcal{U} = \{-1, 0, 1\}$.

At every clock cycle the SDA visits one node of the search tree, in other words, one (partial or full) candidate solution is evaluated.

3.3.6 Computational burden and resource usage

Offline calculations

All of the matrices are pre-calculated in advance and stored on the FPGA in arrays. This includes the state-space matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, as well as the prediction matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Upsilon}$. The Cholesky decomposition of $\boldsymbol{Q}$ is calculated via MATLAB® in order to obtain the generator matrix $\boldsymbol{H}$. The pre-multiplication of $\boldsymbol{Q}^{-1}$ and $\boldsymbol{H}$ are done offline. Note that matrices $\boldsymbol{S}$ and $\boldsymbol{E}$ are not required for practical implementation (see (3.22) in Section 3.3.3). The $\alpha\beta$-reference currents are calculated at discrete intervals of T_s and stored on the FPGA in arrays.

It is important to note that when the weighting factor λ_u is changed, $\boldsymbol{Q}$ and $\boldsymbol{H}$ have to be recalculated. This poses a problem as the switching frequency, via tuning of λ_u, cannot be adjusted online without recalculating the required matrices. When a reference change is given, the switching frequency could change to an undesired frequency. This can be overcome in two possible ways.

For multiple λ_u values, multiple $\boldsymbol{Q}$ and $\boldsymbol{H}$ matrices can be calculated offline and stored on the FPGA in lookup tables. This approach has significant memory requirements.

Another approach would be to calculate the matrices online, albeit not instantaneously. Using this approach would require significantly more DSP blocks, inhibiting the achievable horizon. Note that the matrices grow quadratically in size as the prediction horizon increases, and the Cholesky decomposition is non-trivial.

Online calculations

The number of clock cycles in which online calculations are permitted within a sampling period is given by $T_s f_{clk}$, where f_{clk} is the clock frequency of the controller algorithm. To guarantee optimality, the sphere decoder must terminate within the allocated clock cycles. Irrespective of the sampling interval and clock frequency, calculations for the unconstrained solution and initial radius are limited to one term per clock cycle.

Five clock cycles are used to calculate the sampled $\alpha\beta$-currents. This includes processing the ADC value, applying the Clarke transformation, and adding the delay compensation. The reference is also shifted accordingly. $5 \times 3N_p$ clock cycles are required to calculated the (transformed) unconstrained $\boldsymbol{U}_{unc}(k)$ ($\bar{\boldsymbol{U}}_{unc}(k)$), and the radii squared $\rho^2_{ed}(k)$ and $\rho^2_{bab}(k)$. One clock cycle is used to determine the initial radius $\rho^2_{ini}(k)$, and a final clock cycle is used to send $\bar{\boldsymbol{U}}_{unc}(k)$ and $\rho^2_{ini}(k)$ to the SDA. Taken as a whole, $7 + 5 \times 3N_p$ clock cycles are required before the sphere decoding starts.

The remaining clock cycles are used for sphere decoding. If the sphere decoder does not terminate within the sampling period, the Babai estimate is applied to the plant in order to avoid infeasibility issues. Ideally, the incumbent optimal solution found by the sphere decoder should be applied. However, the possibility for truncation does exist, since the variables are assigned a reasonable number of bits (i.e., the largest number a variable can represent) within the expected operating range. Transients can occur where the variables of the sphere radius or unconstrained solution truncates. This will lead to a compromised sphere decoder, since some candidate solutions will erroneously be considered to belong within the sphere.

FPGA resources

The resource usage of the FPGA for prediction horizons $N_p = 1$ to 5 is shown in Figure 3.7. Increasing the horizon leads to a linear increase in resource usage. Note that squaring is done in two steps: the term that should be squared is calculated, and then multiplied with itself. The number of bits assigned to variables range from 23 to 33. It is encouraging to see that digital signal processing (DSP) blocks, arguably the most valuable resource, increase by only 13 blocks as the horizon increases. The limiting factor for the prediction horizon was the amount of DSP blocks available, as 100 % are used, while only 21 % of the adaptive logic modules (ALMs) are used.

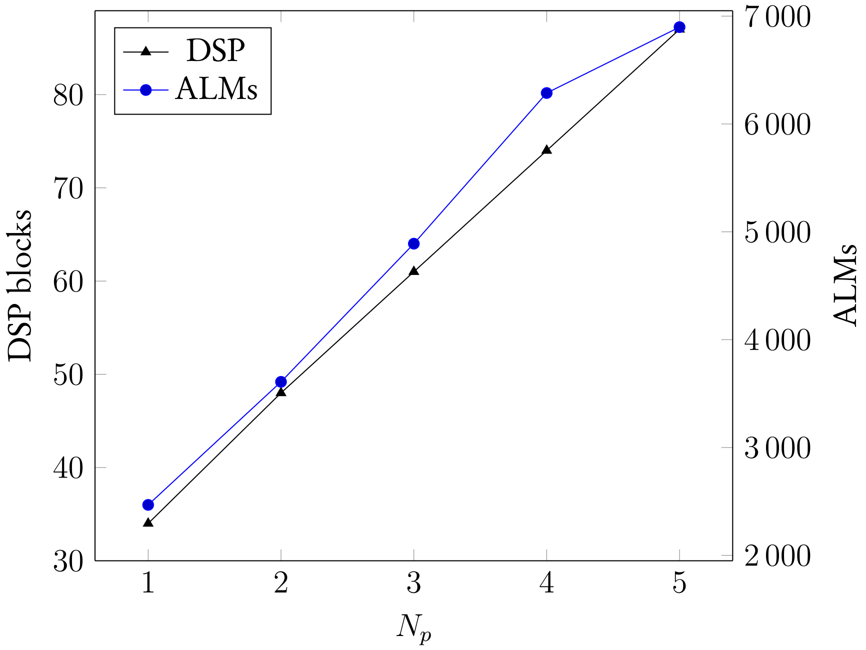


Figure 3.7: Resource usage of controller implementation. The FPGA has a total of 87 DSP blocks and 32 070 ALMs.

3.3.7 Verification

ILS verification

First and foremost, an exhaustive search (see Algorithm 1 in Section 2.3.4) and an ILS simulation are done in order to verify that the reformulated ILS problem is correctly implemented. Ideally, it is preferable to do a comparison for $N_p = 5$, as this was the longest horizon that could be implemented on the FPGA and has the most complexity. Unfortunately, for horizons beyond $N_p > 2$, the exhaustive search rapidly becomes intractable. Since all simulations are run for multiple fundamental periods to ensure steady-state conditions, only an exhaustive search simulation for $N_p = 3$ was done. MATLAB® was used for the simulation platform. The system parameters are shown in Table 3.1.

Figure 3.8 shows the comparison of phase currents for the exhaustive search and ILS simulations. A 50 Hz, 10 A (peak) reference current is given, and the weighting factor λ_u is adjusted so that the average switching frequency is 250 Hz. It can be seen that the exhaustive search and ILS problem have identical results, confirming the implementation of the reformulated problem. The switch positions of the simulation are shown in Figure 3.9.

Table 3.1: Simulation parameters.

Parameter	Description	Value
V_d	Bus voltage	100 V
R	Resistive load	3.5 Ω
L	Inductive load	2 mH
T_s	Sampling interval	25 µs

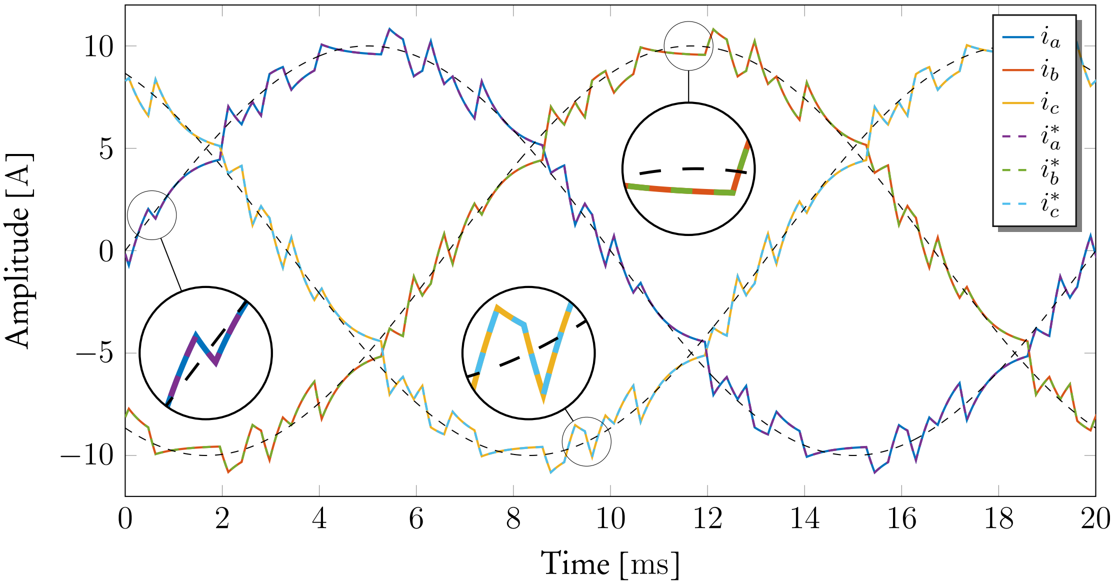


Figure 3.8: Exhaustive search and ILS simulation comparison for $N_p = 3$. The phase currents for the exhaustive search and the ILS simulations are denoted by i_x and i_x^*, respectively. The phase references are depicted by the black dash dotted lines.

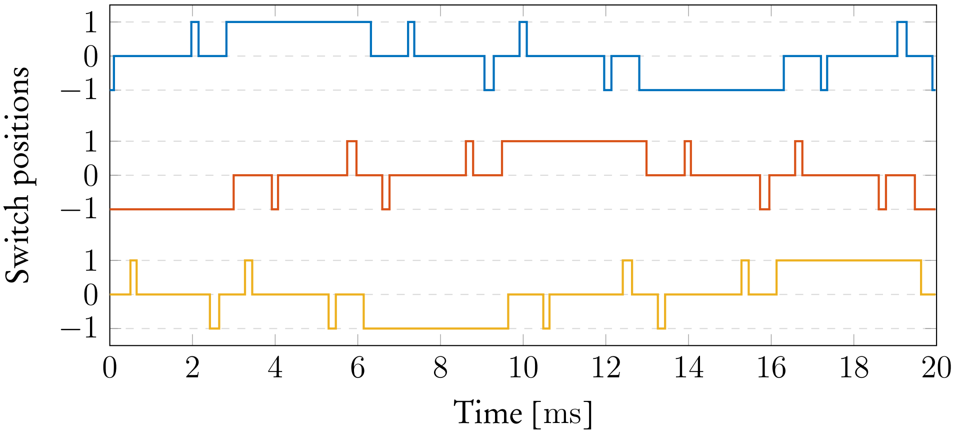


Figure 3.9: The switch positions for ILS verification simulation. The switch positions for phases a, b, and c are indicated by blue, red, and yellow, respectively.

VHDL verification

To verify the VHDL implementation of the controller, a simulation of the controller and the plant is conducted within the FPGA. The simulation and the practical controllers are near identical, with the only differences being that in the practical controller a delay compensation is added and the state variable is updated via the ADC. The sampling interval is chosen as $T_s = 25\,\mu\text{s}$ to demonstrate that the optimization problem can be solved well under $100\,\mu\text{s}$ in real-time. Although only the verification for a prediction horizon of $N_p = 5$ will be shown, as this is the horizon with the most complexity, prior horizons were also verified.

According to the timing analysis tool, the upper bound and lower bound of the maximum clock frequency are 16.1 MHz and 31.1 MHz, respectively.[8] A conservative approach is taken and the clock frequency is set to 15 MHz.

Figure 3.10 shows a comparison of the simulation executed on the FPGA and that of MATLAB®. A 50 Hz, 10 A (peak) reference current is given, and the weighting factor λ_u is adjusted so that the average switching frequency is 250 Hz. Figure 3.11 shows the switch positions of the simulation. It should be apparent that the FPGA implementation of the controller algorithm is correct, as Figure 3.10 indicates that both simulations have identical results.

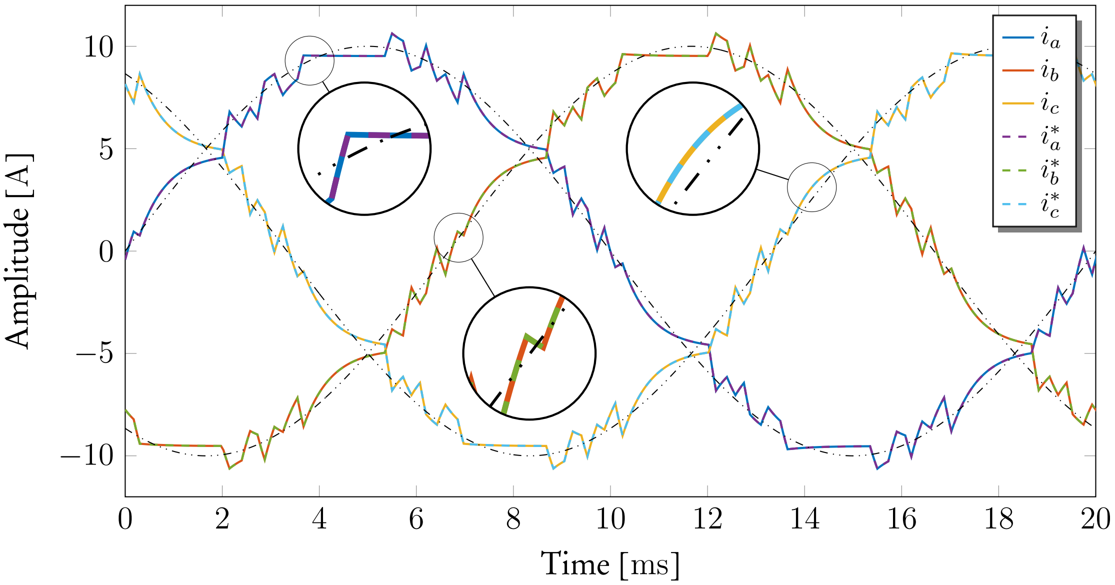


Figure 3.10: FPGA and MATLAB® simulations comparison for $N_p = 5$. The phase currents for the FPGA and MATLAB® simulations are denoted by i_x and i_x^*, respectively. The phase references are shown by the black dash dotted lines.

[8] The TimeQuest Timing Analyzer does an analysis that is subjected to a variety of conditions. This includes temperature, voltage, and manufacturing process conditions.

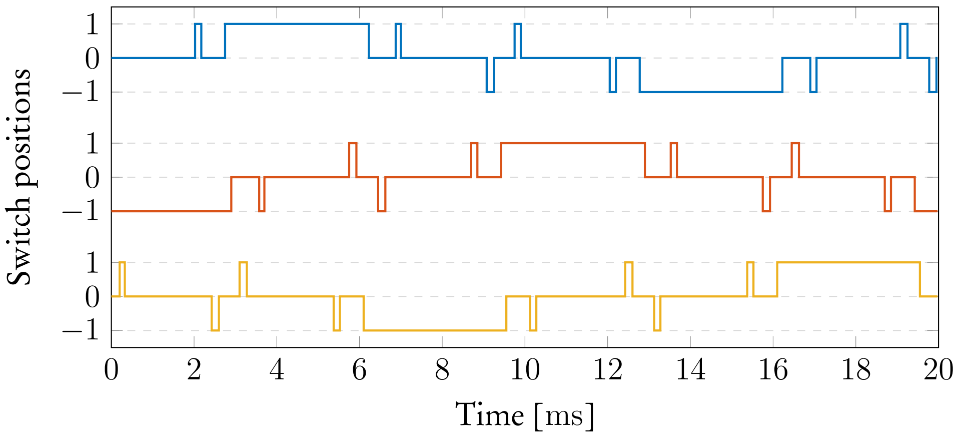


Figure 3.11: The switch positions for FPGA verification simulation. The switch positions for phases a, b, and c are indicated by blue, red, and yellow, respectively.

Figure 3.12 shows the distribution of the number of clock cycles required for sphere decoding within one fundamental period. The data is captured from the FPGA simulation. In 89.5 % of the occurrences, the sphere decoder only requires 45 clock cycles to receive a certificate for optimality. This translates to only 3 µs. The calculations during the first 82 clock cycles have a duration of 5.4 µs. Thus, in 89.5 % of the occurrences, the controller only requires 8.4 µs to find the optimal switch positions; well under 25 µs. The maximum number of clock cycles the sphere decoder required to solve the optimization problem was 120 cycles. This relates to 13.4 µs in total, still within the sampling interval. Considering that there are 3^{15} candidate solutions, this is rather promising.

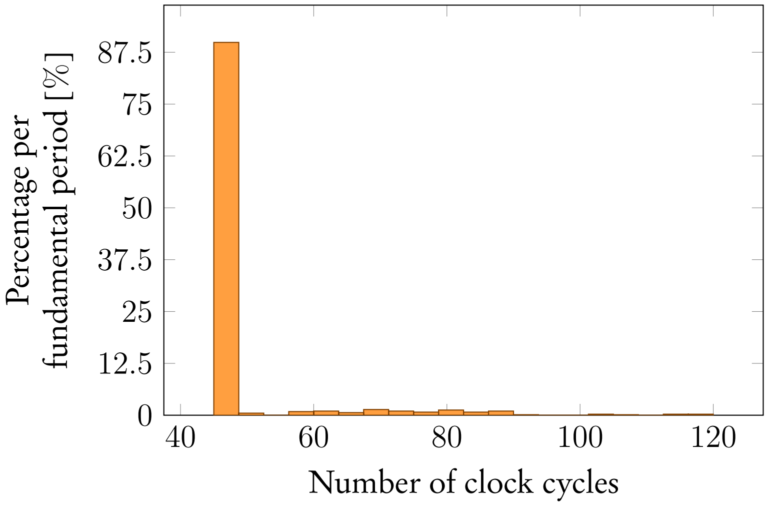


Figure 3.12: Histogram of number of clock cycles required for sphere decoding for $N_p = 5$.

3.4 Summary

With the processing power available today, long-horizon direct MPC can be practically implemented within a low cost FPGA. However, the structure of the code should not be considered trivial. By efficiently structuring the code and using a non-recursive SDA, the resource usage is drastically decreased, enabling the possibility of long horizons.

The optimization problem can be solved in real-time well within a sampling interval of $T_s = 25\,\mu\text{s}$ for a prediction horizon of $N_p = 5$. The computational burden associated with sphere decoding scales well with an increase in the prediction horizon. In [1], the average nodes visited by the sphere decoder roughly doubled when the horizon was increased from $N_p = 5$ to $N_p = 10$. It can be concluded that solving the optimization problem for prediction horizons up to $N_p = 10$ in real-time is possible, provided the FPGA has sufficient resources for the implementation.

Chapter 4

Performance evaluation of long horizons

4.1 Introduction

This chapter presents experimental results of long-horizon direct MPC. The experimental setup that is used during practical evaluation is explained, including the testing framework and conditions. The steady-state performance of direct MPC with horizons $N_p = 1$ to 5 are presented and discussed, with an emphasis on the trade-off between harmonic distortions and switching frequency. A comparison between MPC, OPPs, and SVM is given. The response time of a prediction horizon $N_p = 5$ is analysed via a reference step. This chapter concludes with a summary.

4.2 Testing procedure

4.2.1 Experimental setup

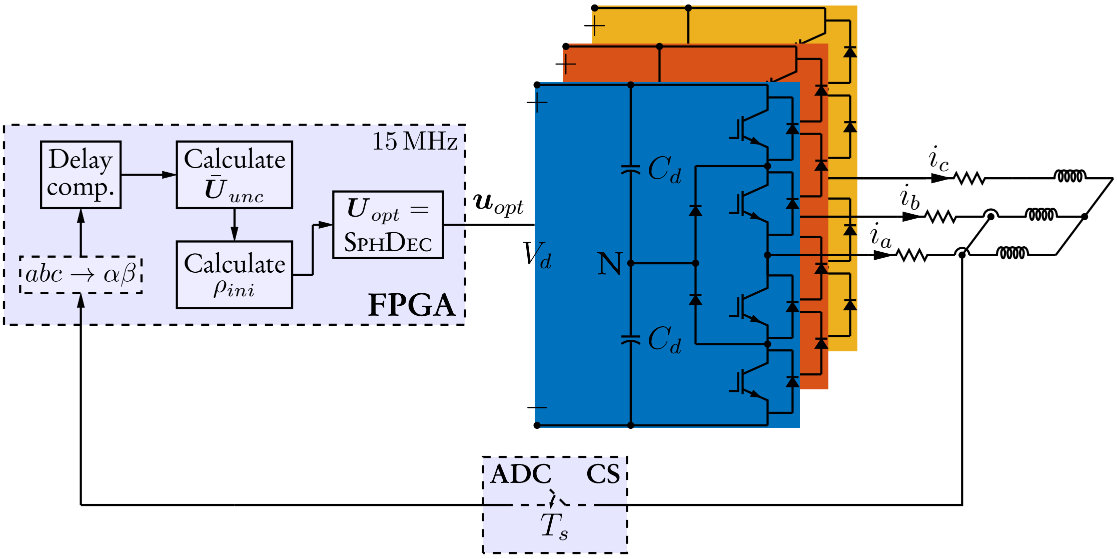


Figure 4.1: Experimental setup block diagram.

Figure 4.1 represents the experimental setup that is used to evaluate long-horizon direct MPC. The flow of the controller algorithm, which is clocked at 15 MHz, is illustrated.

The experimental setup consist of a Cyclone® V FPGA[1], an Infineon F3L030E07 evaluation board with an F3L75R07W2E3 three-level IGBT module, an RL load, an Allegro ACS714 Hall effect current sensor (CS), and a Texas Instruments ADS7864 ADC. The plant parameters are shown in Table 4.1. The parameters are identical to that of the simulations in Section 3.3.7, but with the sampling interval increased to $T_s = 100\,\mu\text{s}$. The bus voltage is realised by two power supplies; one across each of the top and bottom capacitors. The power supplies ensure that the neutral point potential has minimum fluctuations, which was assumed to be constant during modelling.

Table 4.1: Practical parameters.

Parameter	Description	Value
V_d	Bus voltage	100 V
R	Resistive load	3.5 Ω
L	Inductive load	2 mH
T_s	Sampling interval	100 µs

4.2.2 Testing framework

Long-horizon direct MPC

In order to complete a thorough analysis on the benefit of long horizons, a diverse set of weighting factors λ_u (relating to numerous switching frequencies) for all horizons must be considered. As stated in Section 3.3.6, changing λ_u requires certain matrices to be recomputed. Recomputing these matrices offline is the only viable option available for the FPGA at hand, since the implementation of the longest horizon ($N_p = 5$) causes all the resources to be utilised. After recalculating the new matrices offline, the FPGA has to be recompiled with the matrices for the new λ_u. This is inconvenient, as it takes an excess of 20 minutes to compile for long horizons.

To overcome this, serial communication via a universal asynchronous receiver/transmitter (UART) is established between the FPGA and personal computer. MATLAB® is used to manage the automatic testing procedure. This includes calculating the required matrices for λ_u values, converting them to a fixed-point representation, and then transmitting to the FPGA. The experimental data (e.g., the ADC readings and switch positions) stored in the memory of the FPGA is sent back to MATLAB®, where it is processed.

Data is sampled at the same sampling interval as the controller, that is, T_s. Due to the low (average) switching frequency, harmonic components of the current above the Nyquist frequency ($\frac{f_s}{2}$) should be insignificant. This allows for 80 fundamental periods of data per λ_u to be measured, resulting in an averaged representation of steady-state operation.

Alternative modulation schemes

OPPs are calculated offline and stored on the FPGA in lookup tables. The optimization problem is repeated multiples times in order to improve the probability that the global minimum, or at least a satisfactory local minimum, is found. The modulating signal (with the added common-mode term) of SVM is calculated offline for one fundamental period and stored on

[1] More specifically, a terasIC DE01-SoC development board is used that includes the FPGA.

the FPGA in lookup tables. Two carrier signals with phase disposition are generated on the FPGA that are used to modulate the input signals.

The modulation indices of both OPPs and SVM are tuned so that the fundamental component is nearly equivalent to that of MPC. The measurements of OPPs and SVM are sampled at 25 µs and recorded over 20 fundamental periods.

Harmonic distortion and switching frequency

The total harmonic distortion (THD) of the current is a measurement of the harmonic distortion present in the phase current, and is given by [30]

$$I_{THD} = \frac{\sqrt{\sum_{n\neq 1} I_{n,rms}^2}}{I_{1,rms}}, \tag{4.1}$$

where $I_{1,rms}$ is the root-mean-square (rms) value of the fundamental component, and $I_{n,rms}$ the rms value of the n-th harmonic. Note that due to the variable switching frequency of MPC, n is not restricted to an integer and all of the harmonics up to the Nyquist frequency of the Fourier transform are considered. The final THD value is comprised of the average THD of the three-phase currents.

Although direct MPC does not have a fixed switching frequency, the *average* switching frequency of a semiconductor switch can be calculated.[2] This is done by counting the number of switching transitions over an NT_s interval, and is given by [1]

$$f_{sw} = \lim_{N\to\infty} \frac{1}{12NT_s} \sum_{l=0}^{N-1} \left\|\boldsymbol{u}(l) - \boldsymbol{u}(l-1)\right\|_1, \tag{4.2}$$

where $\|\boldsymbol{\xi}\|_1$ is the 1-norm (or Taxicab norm) of vector $\boldsymbol{\xi}$, N is the number of samples, and the factor 12 represents the number of semiconductor switches in an NPC inverter.

4.3 Long horizons steady-state performance evaluation

4.3.1 Comparison of $N_p = 1$ to 5

The weighting factor λ_u is varied so that switching frequencies from $f_{sw} = 125$ Hz to 450 Hz (widely-used medium-voltage application switching frequencies) are obtained for horizons $N_p = 1$ to 5. In total, the results of over 1000 λ_u values are recorded and examined. In the figures, dots represent recorded values. Appropriate trend lines are plotted. Unless stated otherwise, all tests are conducted with a peak reference current of 8 A.

Figure 4.2 shows the trade-off between the switching frequency and current THD for horizons $N_p = 1$ to 5. It can be seen that, in general, longer horizons tend to improve current THD for a given switching frequency. The improvement in THD from $N_p = 2$ to 3 is substantial, especially between 200 Hz and 350 Hz. After increasing the horizon beyond $N_p > 3$, a decrease in current THD is still achieved, albeit gradually. Considering that dead-time, quantization errors (from the ADC and the fixed-point representation of variables), and model uncertainty are present, which all have adverse effects as the horizon increases,

[2] From here on when referring to the switching frequency of direct MPC, the average switching frequency is implied.

the results of $N_p = 5$ are encouraging.[3] This also highlights the importance of the receding horizon principle. Note that a longer horizon does not necessarily offer improved performance over all the switching frequencies. This was also observed in [15]. The benefits of long horizons are more profound for higher-order and more complex plants; see [35] for control of a quasi-Z-source inverter and [36] for control of an LC filter-connected induction motor.

The horizon $N_p = 2$ case had peculiar results. From 375 Hz to 425 Hz, it outperformed $N_p = 1$ noticeably. However, from 175 Hz to 240 Hz, $N_p = 2$ performed the worst of the tested horizons. Measurements were repeated and implementation was verified with a simulation comparison as discussed in Section 3.3.7. It is worth mentioning that with a sampling interval of $T_s = 100\,\mu\text{s}$ and clock frequency of $f_{clk} = 15\,\text{MHz}$, the sphere decoder always issues a certificate for optimality since the number of available clock cycles is greater than the nodes of the search tree. The search tree has 798 nodes, which is given by $\sum_{i=0}^{3N_p} 3^i$, and there are roughly 1500 clock cycles available for sphere decoding.

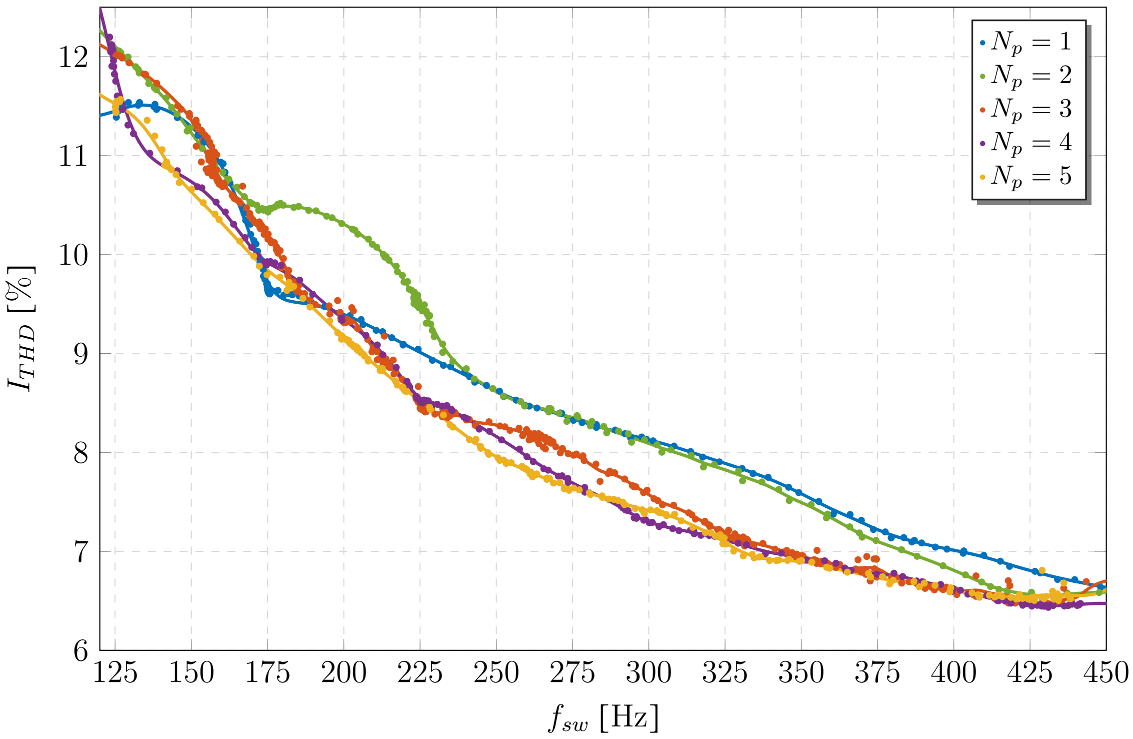


Figure 4.2: Trade-off between switching frequency and current THD for horizons $N_p = 1$ to 5.

Figure 4.3 shows the influence of the weighting factor λ_u on the switching frequency. As expected, the switching frequency decreases as λ_u increases. The switching frequency cannot be reliably increased beyond 450 Hz. As $\lambda_u \rightarrow 0$, the entries in $\boldsymbol{Q}^{-1}$ become larger and eventually truncate in the FPGA due to the number of integer bits assigned to the variables.[4] This causes the unconstrained solution $\boldsymbol{U}_{unc}(k)$, and therefore the Babai estimate, to become compromised. Furthermore, the conditioning of the matrices start to deteriorate until they become *ill-conditioned*.[5] This means that rounding and quantization errors will have a profound effect.

[3] A post analysis showed that the time constant of the RL load is 6 % lower than the model parameters.

[4] For convenience, $\boldsymbol{Q} = \boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^T\boldsymbol{S}$

[5] When a matrix $\boldsymbol{A}$ is ill-conditioned and a system of linear equations is given by $\boldsymbol{Ax} = \boldsymbol{b}$, any small changes to $\boldsymbol{b}$ will result in significant changes in $\boldsymbol{x}$.

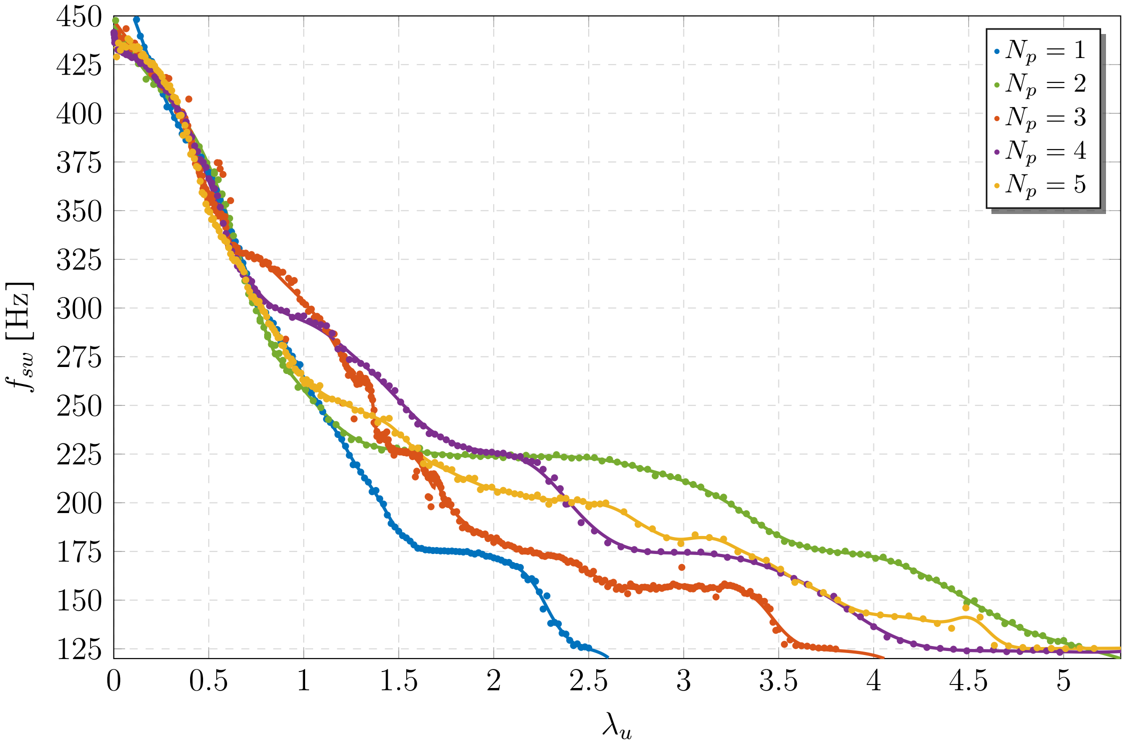


Figure 4.3: Trade-off between switching frequency and the weighting factor for horizons $N_p =$ 1 to 5.

Switching frequencies from 350 Hz to 450 Hz for all horizons have similar λ_u values. Below 350 Hz, longer horizons (generally) require a larger weighting factor than shorter horizons in order to achieve similar switching frequencies. It is interesting to note that all the horizons exhibit frequencies where increasing λ_u result in an identical switching frequency. It appears that direct MPC enters a limit cycle at certain frequencies. The $N_p = 2$ case had an extended period ($\lambda_u = 1.5$ to 2.5) where the switching frequency remained 225 Hz. Also, $N_p = 2$ required the largest weighting factor to achieve similar switching frequencies as longer horizons.

Figure 4.4 shows the trade-off between the current THD and the weighting factor. As can be seen, an increase in λ_u results in an increase in current THD, since the prioritisation of the reference tracking is decreasing. Note the sections where as increase in λ_u resulted in (more or less) the same THD. This translates to the plateaus in switching frequency that are referred to in Figure 4.3. These occurrences can be seen in Figure 4.2 at the concentrated dots, e.g., at $f_{sw} = 225$ Hz for $N_p = 2$ and $f_{sw} = 160$ Hz for $N_p = 3$.

Figure 4.5 and Figure 4.6 show the differential-mode switch positions u_α and u_β, respectively, for prediction horizons $N_p = 1$ and $N_p = 5$. Figure 4.7 shows the difference between the harmonic amplitudes of the differential-mode switch positions for $N_p = 1$ and $N_p = 5$, and illustrates how MPC with longer horizons achieves lower current THD. It can be seen that for harmonics under 650 Hz, the horizon $N_p = 5$ case has smaller differential-mode switch harmonics (and therefore voltage harmonics $\boldsymbol{v}_{\alpha\beta} = \frac{V_d}{2}\boldsymbol{u}_{\alpha\beta}$) than the $N_p = 1$ case, but larger harmonics at frequencies above 3000 Hz. It can be concluded that long horizons shift some of the energy in differential-mode voltage harmonics from lower frequencies to higher frequencies. This was also observed in [1]. This will cause the current THD to be lower, since at higher frequencies the current harmonics will be attenuated by the RL load, which has a cutoff frequency of 280 Hz. It is also stated in [1] that longer horizons can shift some of the harmonic energy into the common-mode voltage harmonics $v_0 = \frac{V_d}{2}u_0$, which do not drive phase currents and contribute to current THD.

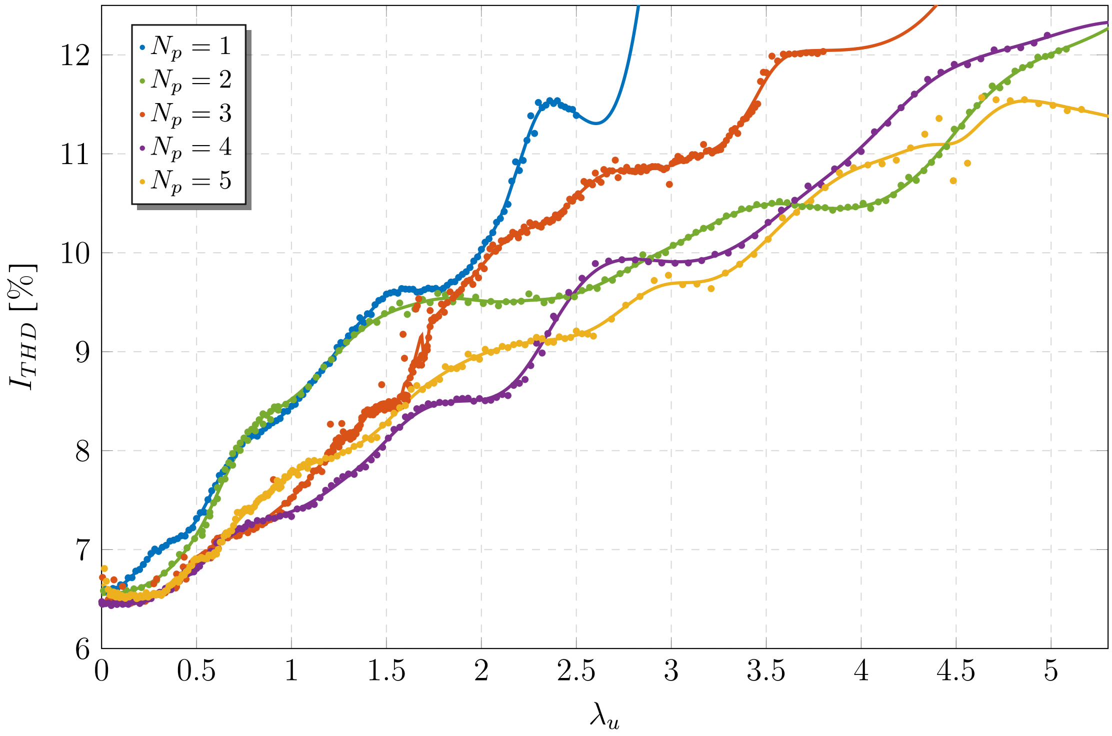


Figure 4.4: Trade-off between current THD and the weighting factor for horizon $N_p = 1$ to 5.

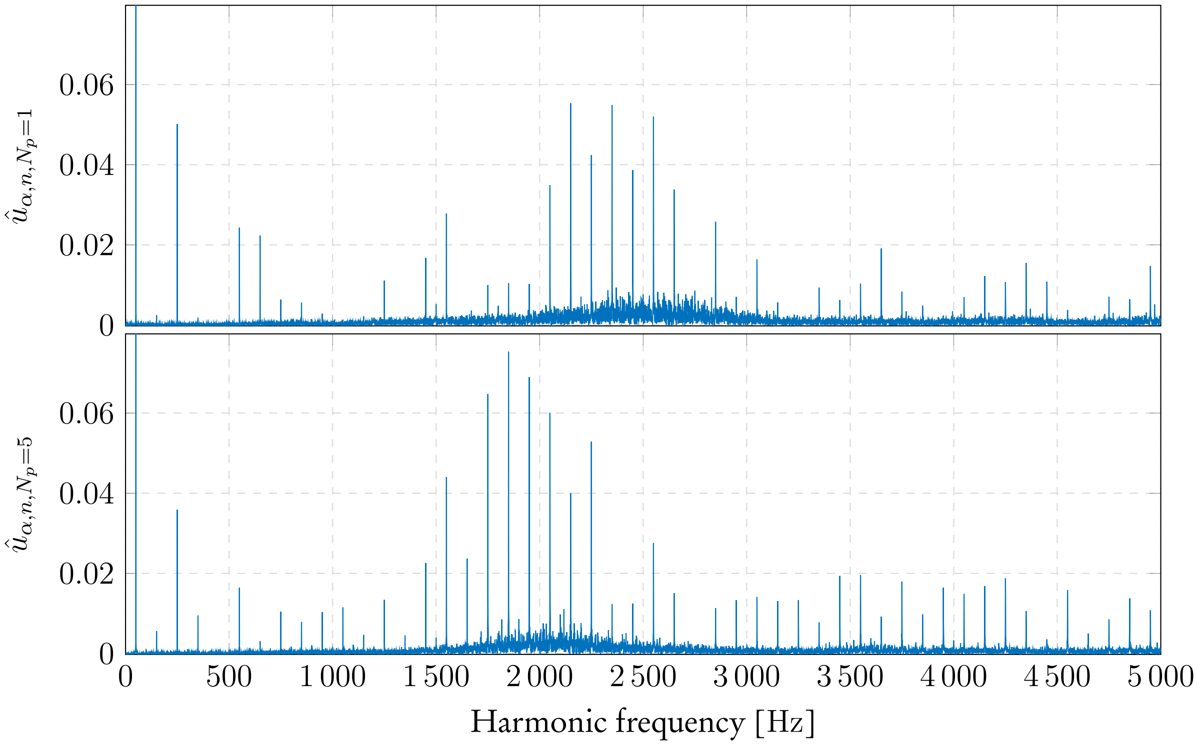


Figure 4.5: The harmonic amplitude spectra of the differential-mode switch position u_α for $N_p = 1$ and $N_p = 5$.

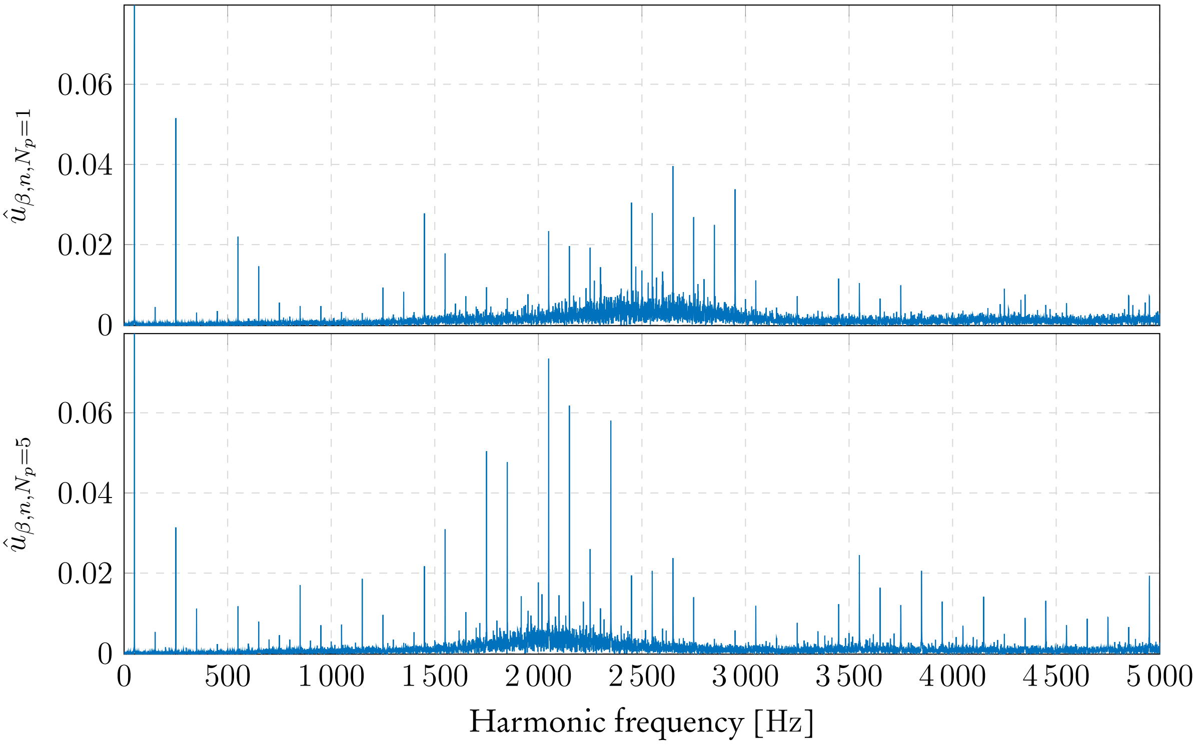


Figure 4.6: The harmonic amplitude spectra of the differential-mode switch position u_β for $N_p = 1$ and $N_p = 5$.

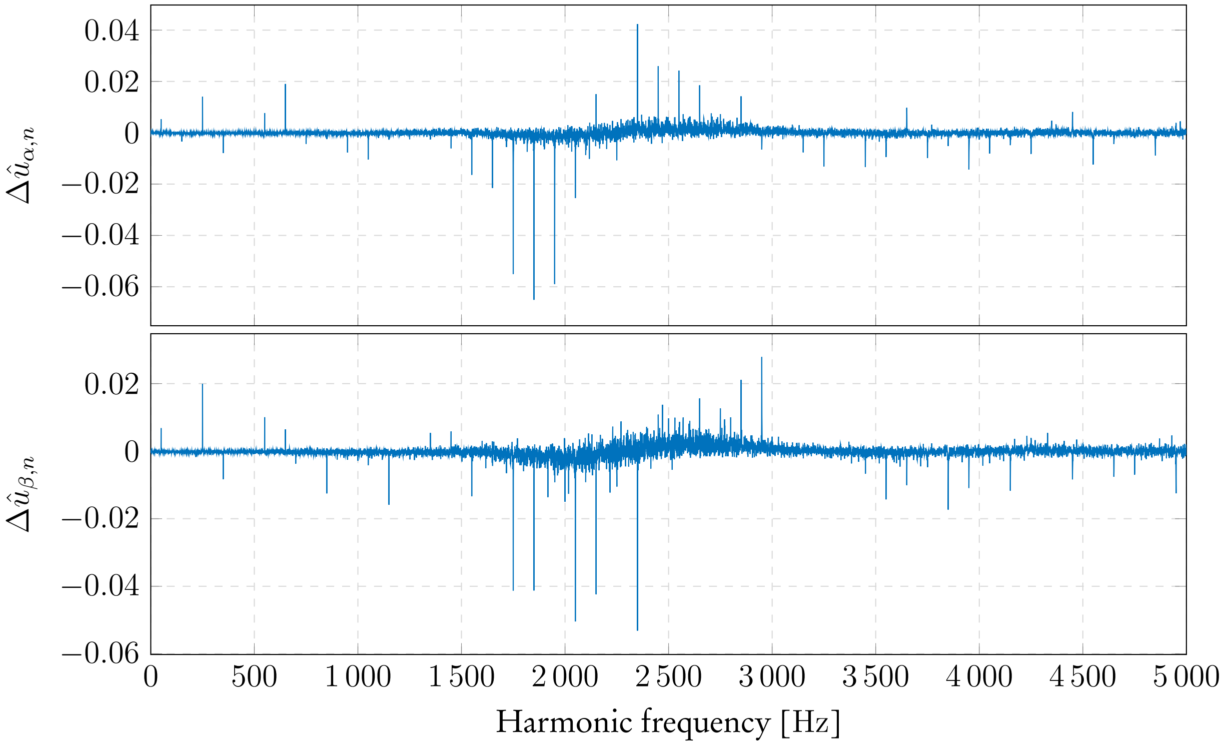


Figure 4.7: The harmonic amplitude spectra of the difference between differential-mode switch positions u_α and u_β, for $N_p = 1$ and $N_p = 5$. The difference in amplitude of the n-th harmonic for the differential-mode switch positions are represented by $\Delta\hat{u}_{\xi,n} = \hat{u}_{\xi,n,N_p=1} - \hat{u}_{\xi,n,N_p=5}$.

4.3.2 Comparison of MPC, OPPs, and SVM

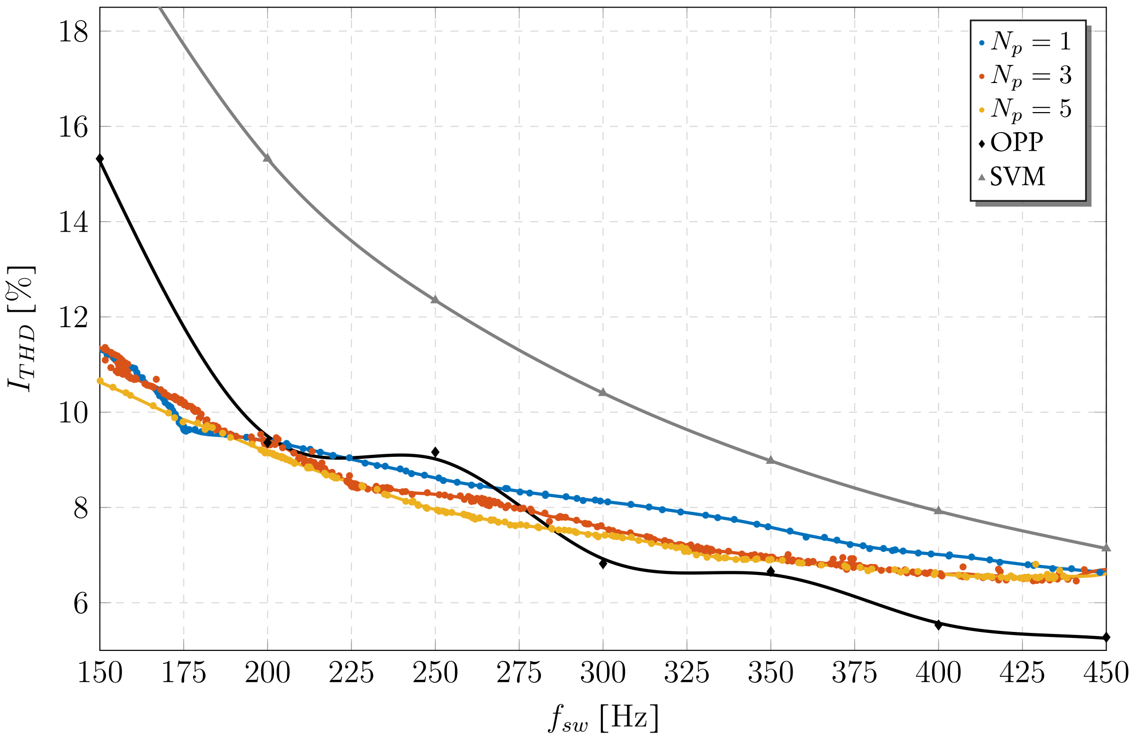


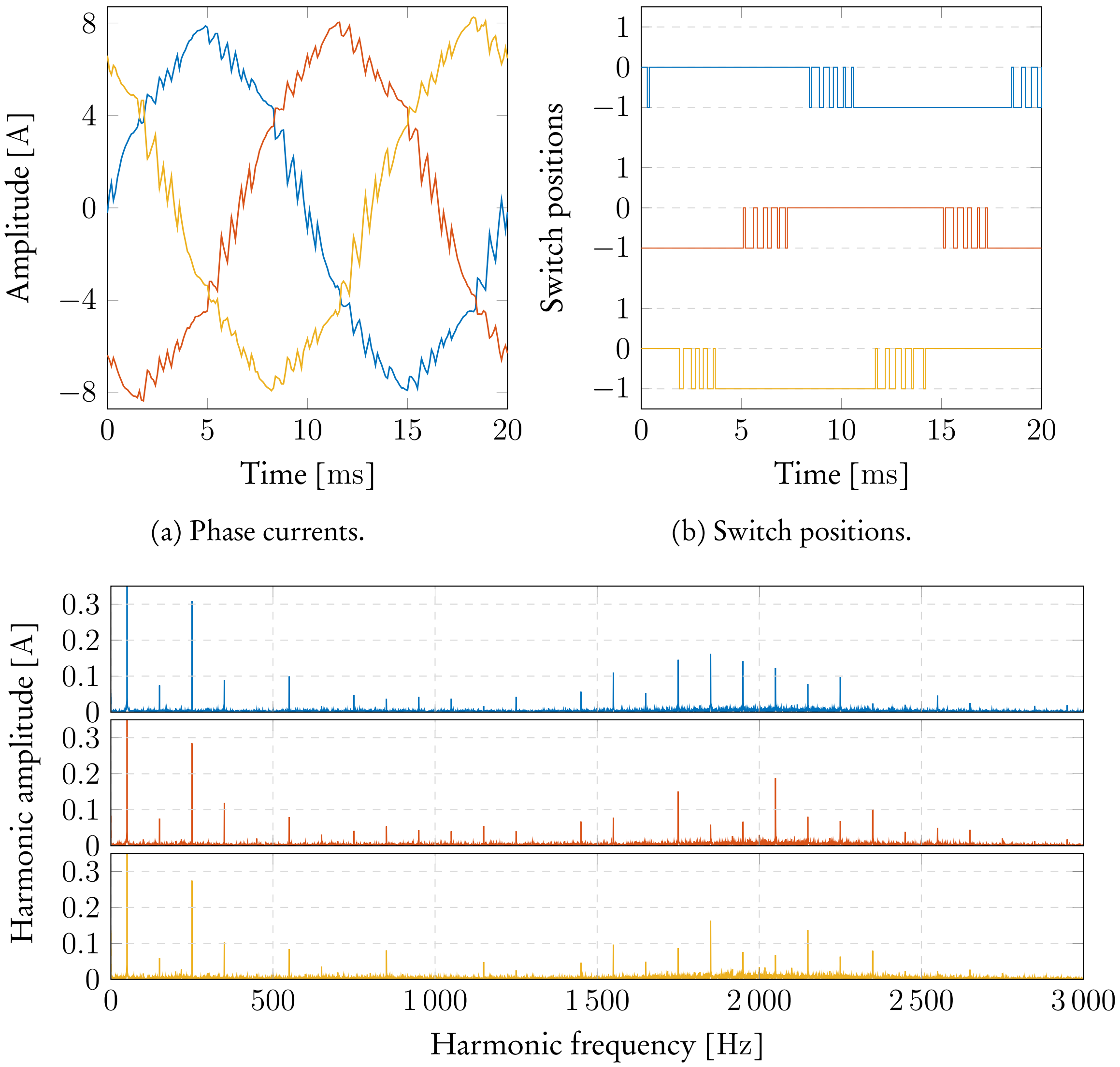
Figure 4.8: Trade-off between switching frequency and current THD for MPC, OPPs, and SVM.

Table 4.2: Comparison of MPC, OPPs, and SVM at $f_{sw} = 250\,\text{Hz}$.

Method	f_{sw} [Hz]	I_{THD} [%]
MPC, $N_p = 1$	251	8.61
MPC, $N_p = 3$	252	8.27
MPC, $N_p = 5$	250	7.94
SVM, $f_c = 450$	250	12.34
OPP, $d = 5$	250	9.16

Unless stated otherwise, all tests are conducted with a peak reference current of $8\,\text{A}$. For OPPs and SVM, the modulation index is set to $m = 0.61$ to achieve the same current fundamental amplitude as MPC. In Figure 4.8, the current THD of MPC with horizons $N_p = 1$, 3, and 5 are compared to OPPs and SVM for switching frequencies between $150\,\text{Hz}$ and $450\,\text{Hz}$. Table 4.2 shows a comparison at $f_{sw} = 250\,\text{Hz}$, a popular switching frequency for medium-voltage applications. As expected, the lower the switching frequency, the more pronounced the advantage of using OPPs instead of SVM becomes. At a switching frequency of $450\,\text{Hz}$, the reduction in THD is roughly $35\,\%$, while at $200\,\text{Hz}$ it is further reduced to $63\,\%$.

When comparing MPC to SVM, the reduction in THD is immediately clear for lower switching frequencies. Even the horizon $N_p = 1$ case outperforms SVM. However, as the switching frequency increases above $450\,\text{Hz}$, the performance of MPC and SVM starts to converge. For higher switching frequencies, SVM can outperform MPC depending on the horizon and sampling interval [15].



(a) Phase currents.

(b) Switch positions.

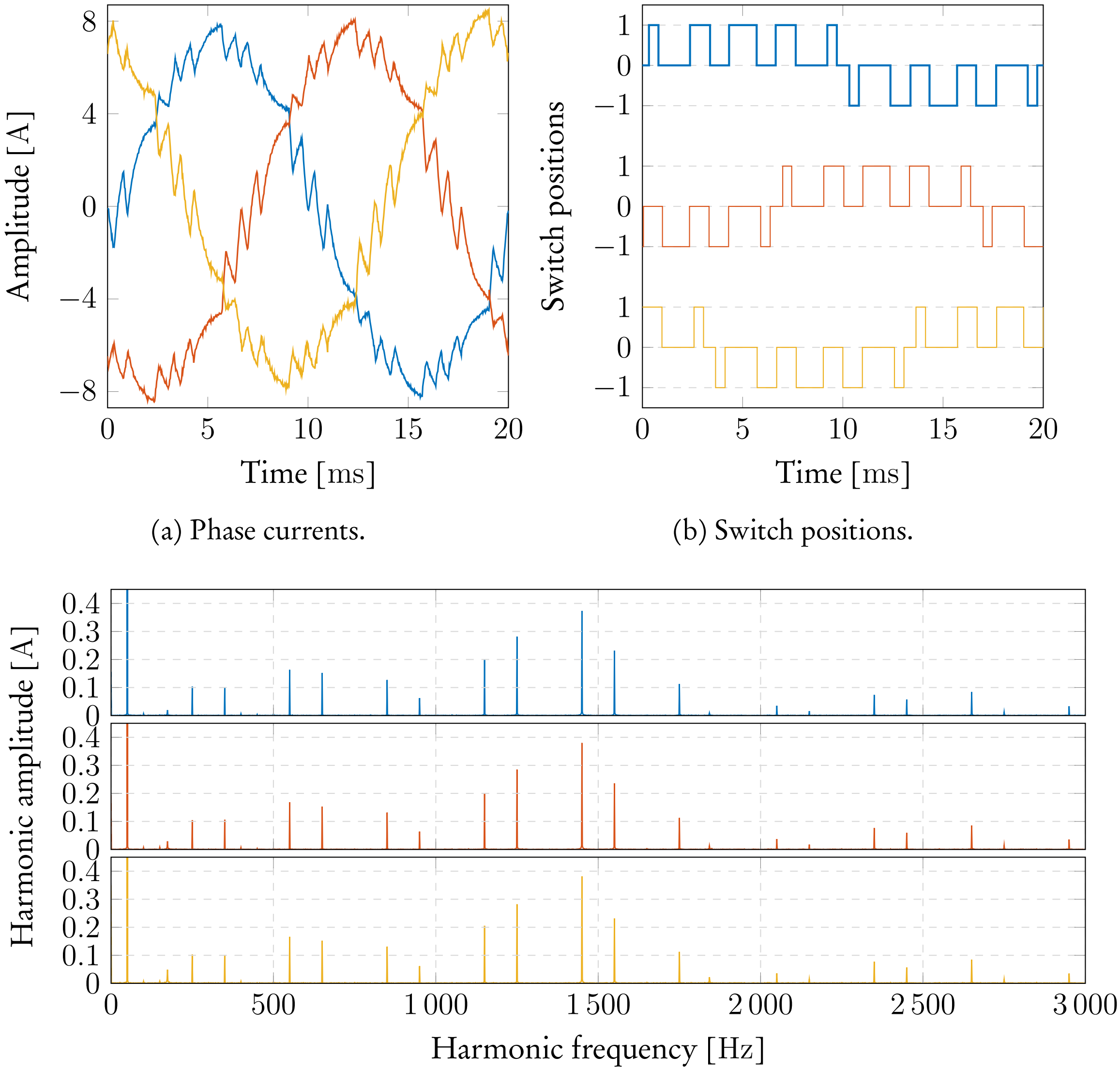
(c) Current spectrum.

Figure 4.9: Current waveforms, switch position, and current spectrum for MPC with $N_p = 5$ at $f_{sw} = 250$. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

The comparison between MPC and OPPs is rather interesting. For switching frequencies above 300 Hz, OPPs outperforms MPC. However, for frequencies below 300 Hz, MPC outperforms OPPs. For frequencies below 200 Hz, MPC presents a noticeable advantage over OPPs. In [15], OPPs always had the lowest THD. The reason MPC outperforms OPPs in the following conditions can be contributed to the rather unique switching pattern of MPC. Figure 4.9 and Figure 4.10 show the current waveforms, switching pattern, and current spectrum of MPC with horizon $N_p = 5$ and OPPs, respectively.[6] Figure 4.9b shows that MPC only uses one half of the bus voltage (i.e., the short voltage vectors in Table 2.2) due to the relatively low current reference in comparison with the bus voltage. This will lead to a smaller

[6] The dc-component seen in the current spectra is due to the offset in the current sensor. This was compensated for in the controller, but not during the measuring and recording of data.

ripple current (see Figure 4.9a and Figure 4.10a). The reader is reminded that characteristics are enforced on OPPs, i.e., quarter wave symmetry and full bus voltage utilisation. In terms of absolute THD, this might not necessarily be the lowest. Moreover, and very importantly, the optimization problem for OPPs (see Section 2.6.2) considers a purely inductive load, which is a valid assumption for frequencies where the reactance dominates the resistance of the load. For grid-connected inverters and induction machines, this statement hold true. However, when considering the RL load parameters that is used, the resistance is far greater than the inductance. At low frequencies the reactance will be significantly less than the resistance, and it can be seen that the advantage that MPC holds over OPPs only increases at lower switching frequencies.



(a) Phase currents.

(b) Switch positions.

(c) Current spectrum.

Figure 4.10: Current waveforms, switch position, and current spectrum for OPPs at $f_{sw} = 250$. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

It is interesting to note that the switching pattern of MPC does pose a form of half-wave symmetry, leading to small even-order harmonics. Some symmetry with appropriate phase shifts is present between the phase switch positions, but they are not identical. This does lead to a more noticeable third-order harmonic at 150 Hz when compared to that of OPPs. MPC has no constraint that enforces identical, 120° out-of-phase switching patterns, or any form of symmetry for that matter. Therefore, there is no guarantee that even-order or third-order harmonics will not be present in the current spectrum. The variable switching pattern of MPC can be observed in Figure 4.9c, where the spectrum does have noticeable inter-harmonics, i.e., harmonics at non-integer multiples of the fundamental. However, the switching pattern does have a form of periodicity, and harmonics are more profound at integer multiples of the fundamental frequency. In contrast, OPPs (as well as SVM) have a significantly more discrete harmonic spectrum, as can be observed in Figure 4.10c.

Table 4.3 shows the current THD of MPC and OPPs when the reference current is changed. In these conditions, OPPs outperform MPC at a switching frequency of 250 Hz. Furthermore, even at the lower reference current, the resulting switching pattern of MPC utilised the full bus voltage.

Table 4.3: Comparison of MPC and OPPs at different reference current.

Method	f_{sw} [Hz]	I_{THD} @ 6 A [%]	I_{THD} @ 9 A [%]
MPC, $N_p = 5$	250	16.67	8.83
OPP, $d = 5$	250	16.14	7.43

4.4 Response time of MPC

A benefit of using MPC is a fast response time during transients. In [15], it is shown that the prediction horizon does not influence the response time of the controller. However, solving the optimization problem during transients requires a greater computational effort than in steady-state operation (see Section 3.3.4). Therefore, practical evaluation is conducted with a horizon of $N_p = 5$, as this horizon has the highest computational burden out of all tested horizons. If the optimal solution is not found during a transient, the fast response of MPC could be inhibited if a suboptimal switch position is applied.

In Figure 4.11, the current is stepped from 8 A to 4 A, then to 10 A, then to 0 A, and then back to 8 A. The switch positions are shown in Figure 4.12. The reference step is suddenly given and not included in the prediction horizon, in other words, the controller cannot anticipate the reference step. Due to the addition of a delay compensation as explained in Section 3.3.2, a delay of T_s (i.e., 0.1 ms) is inherent. The step-down cases have a settling time of approximately 0.2 ms and 0.3 ms, while both the step-up cases take approximately 0.5 ms. The step-down cases are faster since the available voltage to manipulate across the inductor is greater due to the current through the load.

The maximum number of clock cycles required to find the optimal solution during the transients is 1030 out of the available 1500 clock cycles. This includes the calculation of the unconstrained solution and initial radius. For more aggressive reference steps to larger currents or if shorter sampling intervals are used, there is no guarantee that the optimal solution will be found within the sampling interval.

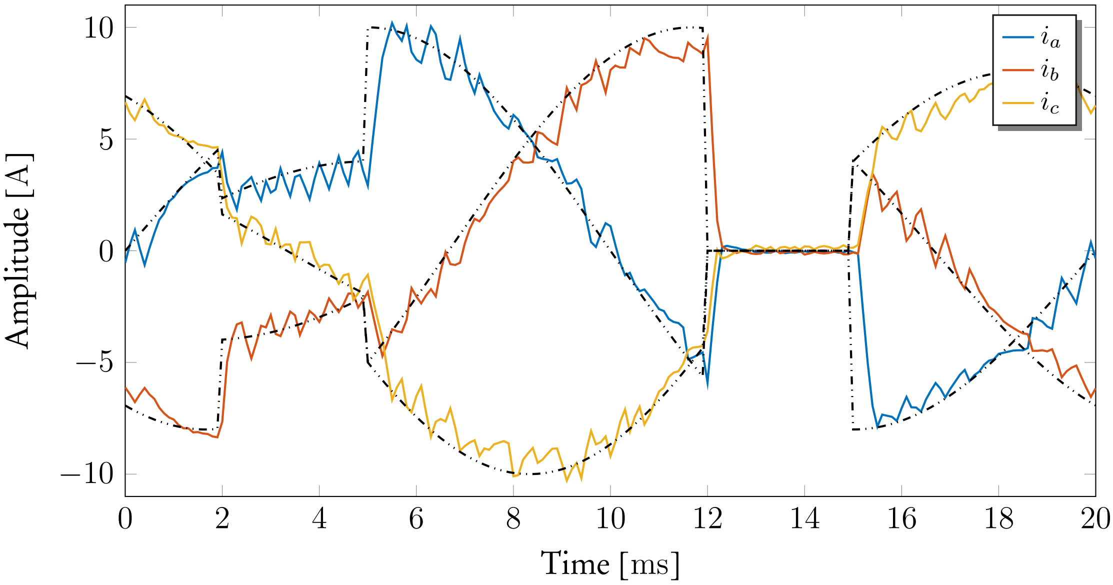


Figure 4.11: The current waveforms for reference step changes. The phase references are shown by the black dash dotted lines.

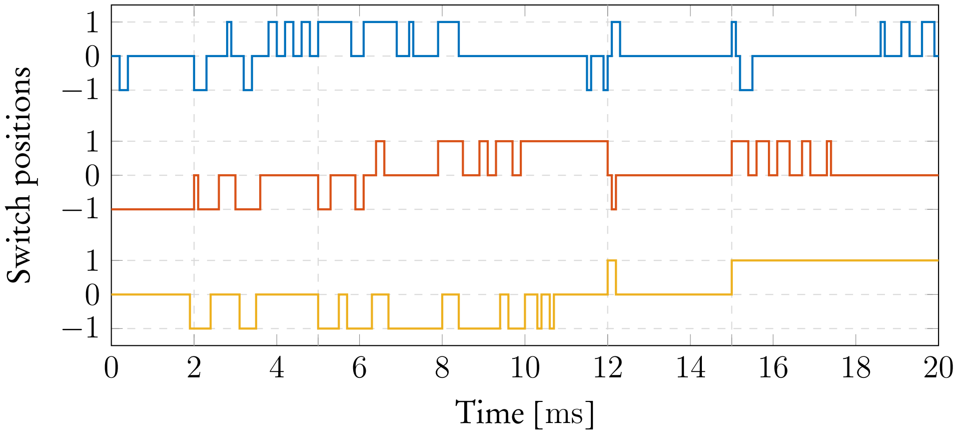


Figure 4.12: The switch positions for reference step changes. The switch positions for phases a, b, and c are indicated by blue, red, and yellow, respectively. The grid lines on the x-axis represent the reference step points.

4.5 Summary

The increased performance benefit that long horizons offer is verified with practical measurements. At a switching frequency of 250 Hz, the current total harmonic distortion is decreased by roughly 8.5 % when using a prediction horizon of $N_p = 5$ instead of $N_p = 1$. Considering that a primitive RL load is used and that modelling uncertainties are present, the benefits of long horizons are promising. The practical application of long horizons on more complex plants should definitely be considered and investigated. The fast response time of MPC is demonstrated, where it is shown to quickly regulate to new reference set-points.

Although MPC did outperform OPPs by a significant margin in some cases, caution should be exercised. For operating conditions that are different from those in this chapter, and if highly inductive loads are considered, it is expected that OPPs will result in the lowest current THD.

Although the performance benefit that longer horizons should offer does not always seem to hold true, only the weighting factor was adjusted to obtain the desired switching frequency. However, there is one parameter that was not adjusted that has an influence on the performance: the sampling interval T_s [15]. The sampling interval, along with the number of prediction steps N_p, determines the length of the horizon in time, N_pT_s. In [15], Monte Carlo simulations were performed, where the prediction horizon N_p, sampling interval T_s, and weighting factor λ_u are randomly chosen for simulations. In the Monte Carlo simulations, the performance benefits of longer horizons over all of the switching frequencies are much more pronounced and clear. This also highlights a disadvantage of long horizons; tuning of the weighting factor and sampling interval is required in order to find the optimal operating point for a given horizon and desired switching frequency.

Chapter 5

Selective harmonic suppression for long horizons

5.1 Introduction

One disadvantage of direct MPC is that little control is available over the resulting non-deterministic harmonic spectrum, which could contain undesired harmonics such as even-order harmonics or certain harmonics that exceed standards. This chapter proposes a method to selectively suppress selected harmonics, allowing some control over the spectrum.

This chapter will open with a brief discussion about some of the existing methods to alter the harmonic spectrum. The formulation of the new proposed suppression method, which is compatible with the integer least-squares (ILS) formulation discussed in Section 2.4, is presented. This includes the formulation of the filter and state-space representation of the entire system. Preliminary simulation results of steady-state performance and response time are presented. This chapter concludes with a summary.

5.2 Existing methods

In literature, methods already exist that are able to eliminate harmonics. However, most of these methods are limited to a prediction horizon of $N_p = 1$.

In [37], the phase currents are passed through a notch filter. The filtered current is included alongside the phase-current tracking error in the cost function. Note that the cost function does not penalise switching transitions. The filtered variable thus represents the frequency components that should be attenuated by the controller (i.e., all frequencies other than that of the notch filter). In order to achieve this, the controller focuses the switching energy into the harmonic that is filtered out. This results in a switching frequency around the frequency of the filter. Whether this method is suited for low switching frequencies well below 1 kHz remains to be seen.

A very interesting and promising method is proposed in [38]. In this method, selective harmonic elimination (SHE) PWM switching patterns are calculated, which is similar to the concept of optimised pulse patterns (OPPs). The cost function includes phase-current reference tracking and a term that penalises deviation of the switch positions from those of SHE-PWM. Thus, in steady state this method has a harmonic spectrum that represents that of the SHE-PWM, while combining the fast response time of MPC during transients. Note that this method depends on the offline calculation of the SHE-PWM patterns over a wide range of operating conditions.

5.3 Formulation of filter

The proposed harmonic suppression method, hereafter referred to as selective harmonic suppression (SHS) MPC, is done exclusively online. The principle of the scheme is as follows. By placing a resonant pole at a target frequency, that is, $s = \pm jw_n$, where w_n is the *natural frequency*, the frequency component of a signal at the target frequency will be significantly amplified. In a sense, the resonant term acts as a filter by amplifying the desired frequency component well above the rest of the spectrum. The filtered signal, consisting of a single frequency component[1], can then be included in the cost function so that it can be suppressed.

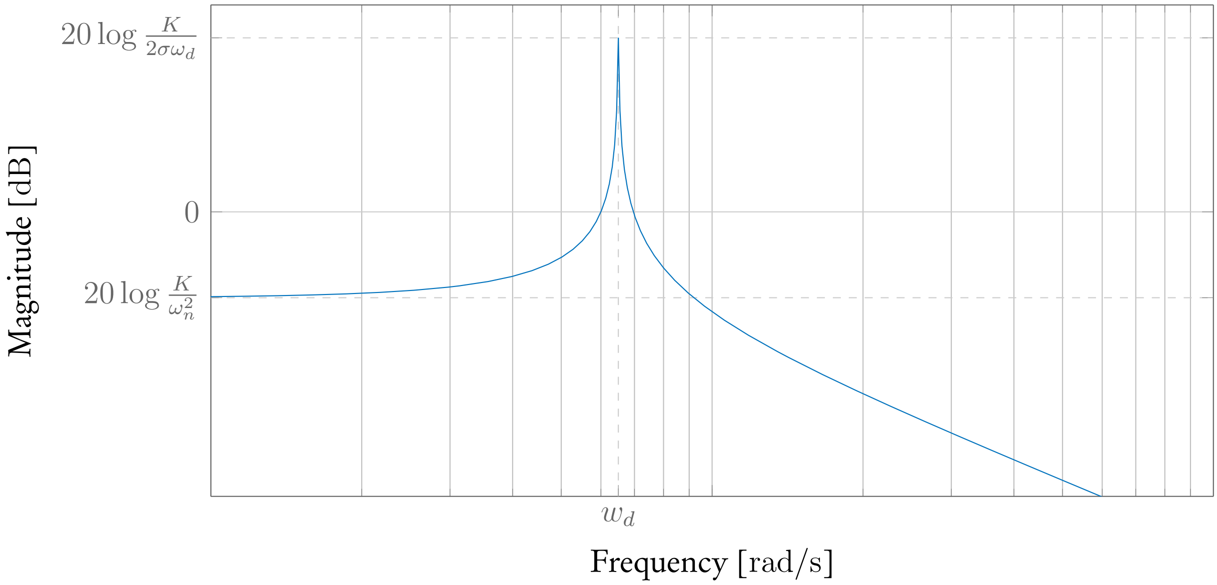


Figure 5.1: Magnitude response of the filter.

The first step in this method will be to design an appropriate filter. First, some slight damping will be added. This will cause a reduction in gain, but the bandwidth will be slightly higher than that of a resonant term. This is done to ensure that the controller does not simply shift the harmonic energy of the suppressed term into a neighbouring frequency (e.g., simply shifting the energy from 250 Hz to 253 Hz). Note that if harmonics near the fundamental frequency are to be suppressed, special attention must be given to the bandwidth. The transfer function of the filter is given by

$$\frac{I_f(s)}{I(s)} = H_f(s) = \frac{K}{s^2 + 2\sigma s + \omega_n^2}, \tag{5.1}$$

which has a complex pole pair at $s = -\sigma \pm j\omega_d$, where

$$\sigma = \zeta\omega_n \tag{5.2a}$$

$$\omega_d = \omega_n\sqrt{1-\zeta^2}, \tag{5.2b}$$

[1] Although the signal will contain other frequency components as well, they are assumed to be insignificant in comparison to that at the natural frequency.

with ζ as the *damping ratio* and ω_d the *damped natural frequency*. The maximum gain of the filter $H_f(s)$ is achieved at ω_d (see Figure 5.1). However, if $\zeta \ll 1$, the maximum gain of the filter will approximately be at the natural frequency, since $\omega_d \approx \omega_n$. The gain at ω_d will be

$$20 \log \left| H(j\omega_d) \right| = 20 \log \frac{K}{2\sigma\omega_d}. \tag{5.3}$$

For the formulation of SHS-MPC in Section 5.3.1 and Section 5.3.2, the filter can be described by a continuous-time state-space representation of

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}_f \boldsymbol{x}(t) + \boldsymbol{G}_f i(t), \tag{5.4}$$

where the state vector is

$$\boldsymbol{x}(t) = \begin{bmatrix} i_f(t) & \dot{i}_f(t) \end{bmatrix}^T, \tag{5.5}$$

and the state and input matrices are

$$\boldsymbol{F}_f = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\sigma \end{bmatrix} \tag{5.6a}$$

$$\boldsymbol{G}_f = \begin{bmatrix} 0 \\ K \end{bmatrix}. \tag{5.6b}$$

5.3.1 Selective harmonic suppression formulation for a single-phase inverter

The proposed elimination method is first evaluated on the single-phase three-level NPC with an RL load, as shown in Figure 5.2.

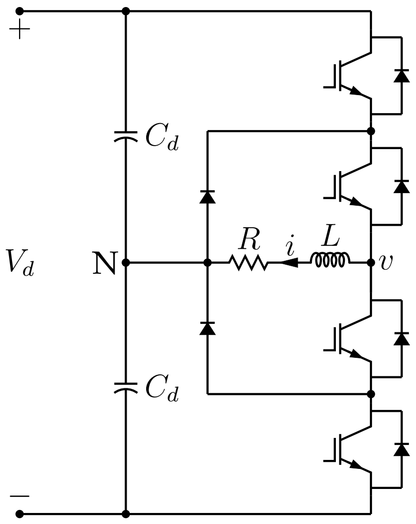


Figure 5.2: Single-phase NPC with an RL load.

The output voltage (with respect to N) is given by $v(t) = \frac{V_d}{2}u(t)$, where $u(t) \in \{-1, 0, 1\}$ denotes the switch position of the inverter. After analysing the circuit, the differential equation describing the current is given by

$$\dot{i}(t) = -\frac{R}{L}i(t) + \frac{V_d}{2L}u(t). \tag{5.7}$$

For a single-phase system, only one filter in the form of (5.1) is required per harmonic suppression. A state-space representation can now be derived that includes the inverter current and filter states,

$$\boldsymbol{x}(t) = \begin{bmatrix} i(t) & i_f(t) & \dot{i}_f(t) \end{bmatrix}^T. \tag{5.8}$$

The augmented state-space representation is given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}u(t), \tag{5.9}$$

where

$$\boldsymbol{F} = \begin{bmatrix} -\frac{R}{L} & \boldsymbol{0}_{1\times 2} \\ \boldsymbol{G}_f & \boldsymbol{F}_f \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & 0 & 0 \\ 0 & 0 & 1 \\ K & -\omega_n^2 & -2\sigma \end{bmatrix} \tag{5.10a}$$

$$\boldsymbol{G} = \begin{bmatrix} \frac{V_d}{2L} \\ \boldsymbol{0}_{2\times 1} \end{bmatrix} = \begin{bmatrix} \frac{V_d}{2L} \\ 0 \\ 0 \end{bmatrix}. \tag{5.10b}$$

Note that the input $i(t)$ in (5.4) is now a state in (5.9). If desired, multiple harmonics can be suppressed, but will require extra filters. The state vector will increase by a factor of two for every harmonic suppression, i.e, $\boldsymbol{x}(t) \in \mathbb{R}^{1+2m}$, where m represents the number of harmonic eliminations. For the general case of m harmonic eliminations, the state vector will be

$$\boldsymbol{x}(t) = \begin{bmatrix} i(t) & i_{f,1}(t) & \dot{i}_{f,1}(t) & \cdots & i_{f,m}(t) & \dot{i}_{f,m}(t) \end{bmatrix}^T, \tag{5.11}$$

where the state and input matrices are

$$\boldsymbol{F} = \begin{bmatrix} -\frac{R}{L} & \boldsymbol{0}_{1\times 2} & \boldsymbol{0}_{1\times 2} & \cdots & \boldsymbol{0}_{1\times 2} \\ \boldsymbol{G}_{f,1} & \boldsymbol{F}_{f,1} & \boldsymbol{0}_{2\times 2} & & \boldsymbol{0}_{2\times 2} \\ \boldsymbol{G}_{f,2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{F}_{f,2} & \ddots & \boldsymbol{0}_{2\times 2} \\ \vdots & & \ddots & \ddots & \vdots \\ \boldsymbol{G}_{f,m} & \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 2} & \cdots & \boldsymbol{F}_{f,m} \end{bmatrix} \tag{5.12a}$$

$$\boldsymbol{G} = \begin{bmatrix} \frac{V_d}{2L} \\ \boldsymbol{0}_{2\times 1} \\ \vdots \\ \boldsymbol{0}_{2\times 1} \end{bmatrix}. \tag{5.12b}$$

The discrete-time state-space representation can by obtained by the exact discretisation as described in Section 2.3.2:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}u(k) \tag{5.13a}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k), \tag{5.13b}$$

where the output vector includes the phase current and filter outputs

$$\boldsymbol{y}(k) = \begin{bmatrix} i(k) & i_{f,1}(k) & i_{f,2}(k) & \cdots & i_{f,m}(k) \end{bmatrix}^T, \tag{5.14}$$

meaning that the output matrix is given by

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & & 0 & 0 \\ \vdots & \vdots & & & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{5.15}$$

The cost function for N_p prediction steps is given by[2]

$$J = \sum_{l=k}^{k+N_p-1} \left\|\boldsymbol{y}_{ref}(l+1) - \boldsymbol{y}(l+1)\right\|_{\boldsymbol{R}}^2 + \lambda_u \left\|\Delta u(l)\right\|_2^2, \tag{5.16}$$

where

$$\boldsymbol{y}_{ref}(l+1) = \begin{bmatrix} i_{ref}(l+1) & i_{ref,f,1}(l+1) & i_{ref,f,2}(l+1) & \cdots & i_{ref,f,m}(l+1) \end{bmatrix}^T, \tag{5.17}$$

and $\Delta u(l) = u(l) - u(l-1)$. The weighting factor on the switching effort is represented by λ_u, and $\boldsymbol{R} \in \mathbb{R}^{(1+m)\times(1+m)}$ represents the penalty matrix where the diagonal elements represent the weight on the tracking errors. Naturally, since the filter outputs should be suppressed (i.e., the selected harmonics), the reference for the filters should be zero. Thus, $i_{ref,f,j} = 0$ for $j \in \{1, 2, \ldots, m\}$.

One of the benefits of using SHS-MPC is that no modifications to the ILS problem formulation (or more specifically, the derivation) are required. This allows for the use of sphere decoding, thus enabling the use of long horizons. With the addition of the penalty matrix $\tilde{\boldsymbol{R}} \in \mathbb{R}^{N_p(1+m)\times N_p(1+m)}$ over the prediction horizon[3] and by using a single-phase system, the constraints of the optimization problem and the calculation of the unconstrained solution $\boldsymbol{U}_{unc}(k)$ is slightly different to that in Section 2.4:

$$\boldsymbol{U}_{opt}(k) = \arg\min_{\boldsymbol{U}(k)} \left\|\bar{\boldsymbol{U}}_{unc}(k) - \boldsymbol{H}\boldsymbol{U}(k)\right\|_2^2 \tag{5.18a}$$

$$\text{subject to } \boldsymbol{U}(k) \in \{-1, 0, 1\}^{N_p}, \tag{5.18b}$$

where

$$\bar{\boldsymbol{U}}_{unc}(k) = \boldsymbol{H}\boldsymbol{U}_{unc}(k) \tag{5.19}$$

$$\boldsymbol{U}_{unc}(k) = -\boldsymbol{Q}^{-1}\boldsymbol{\Theta}(k) \tag{5.20}$$

$$\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{Q}, \tag{5.21}$$

with

$$\boldsymbol{Q} = \boldsymbol{\Upsilon}^T\tilde{\boldsymbol{R}}\boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^T\boldsymbol{S} \tag{5.22}$$

$$\boldsymbol{\Theta}(k) = ([\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k)]^T\tilde{\boldsymbol{R}}\boldsymbol{\Upsilon} - \lambda_u[\boldsymbol{E}\boldsymbol{u}(k-1)]^T\boldsymbol{S})^T. \tag{5.23}$$

[2] The argument of J, as used in Section 2.3.4, is dropped for convenience.

[3] The penalty matrix $\tilde{\boldsymbol{R}}$ over the prediction horizon contains $\boldsymbol{R}$ in the diagonal.

5.3.2 Selective harmonic formulation for three-phase inverters

For convenience, the three-phase three-level MPC with an RL load in Figure 3.1 is repeated in Figure 5.3.

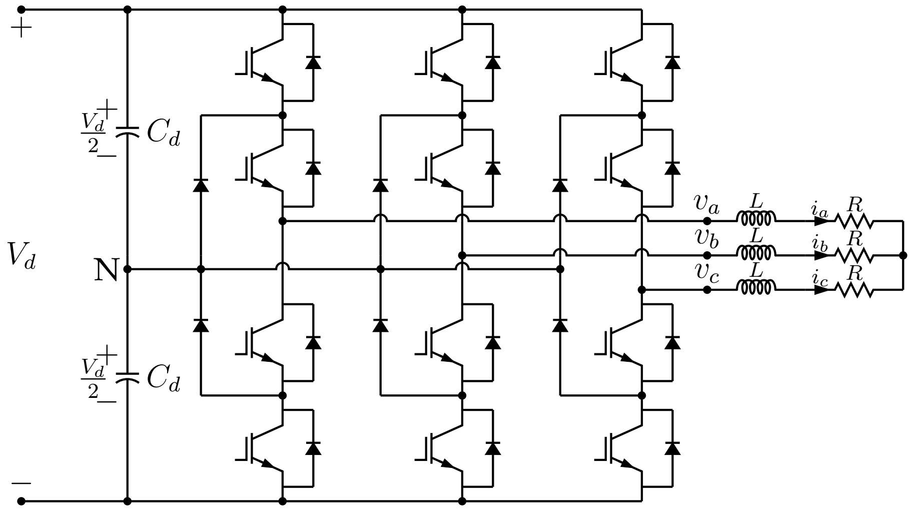


Figure 5.3: NPC inverter with an RL load.

The continuous-time state-space representation of the plant is given by

$$\dot{\boldsymbol{i}}_{\alpha\beta}(t) = \boldsymbol{F}_{RL}\boldsymbol{i}_{\alpha\beta}(t) + \boldsymbol{G}_{RL}\boldsymbol{u}(t), \tag{5.24}$$

where the state vector includes the phase currents in the orthogonal reference frame

$$\boldsymbol{i}_{\alpha\beta} = \begin{bmatrix} i_\alpha(t) & i_\beta(t) \end{bmatrix}^T, \tag{5.25}$$

and the input vector denotes the three-phase switch positions

$$\boldsymbol{u}(t) = \begin{bmatrix} u_a(t) & u_b(t) & u_c(t) \end{bmatrix}^T, \tag{5.26}$$

which are restricted to $\{-1, 0, 1\}^3$. The state and input matrices are given by

$$\boldsymbol{F}_{RL} = -\frac{R}{L}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.27a}$$

$$\boldsymbol{G}_{RL} = \frac{V_d}{2L}\boldsymbol{K}, \tag{5.27b}$$

where $\boldsymbol{K}$ is the (reduced) Clarke transformation as defined in Section 2.2.2. The derivation of (5.24) can be found in Appendix C.

The formulation of SHS-MPC for a three-phase system is similar to that of a single-phase system described in Section 5.3.1. The only difference is that two filters (instead of one) is required per harmonic suppression; one filter each for i_α and i_β. For one harmonic suppression, the state vector will include $\alpha\beta$-currents and four filter states,

$$\boldsymbol{x}(t) = \begin{bmatrix} i_\alpha(t) & i_\beta(t) & i_{\alpha f}(t) & \dot{i}_{\alpha f}(t) & i_{\beta f}(t) & \dot{i}_{\beta f}(t) \end{bmatrix}^T. \tag{5.28}$$

The augmented state-space representation is given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{u}(t), \tag{5.29}$$

where

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_{RL} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \boldsymbol{G}_{\alpha f} & \boldsymbol{F}_{\alpha f} & \mathbf{0}_{2\times2} \\ \boldsymbol{G}_{\beta f} & \mathbf{0}_{2\times2} & \boldsymbol{F}_{\beta f} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{R}{L} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ K_\alpha & 0 & -w_{\alpha n}^2 & -2\sigma_\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & K_\beta & 0 & 0 & -w_{\beta n}^2 & -2\sigma_\beta \end{bmatrix} \tag{5.30a}$$

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_{RL} \\ \mathbf{0}_{2\times3} \\ \mathbf{0}_{2\times3} \end{bmatrix} = \frac{V_d}{3L} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{5.30b}$$

Note that the input vectors of the filters in (5.6b) are now redefined to

$$\boldsymbol{G}_{\alpha f} = \begin{bmatrix} 0 & 0 \\ K_\alpha & 0 \end{bmatrix} \tag{5.31}$$

$$\boldsymbol{G}_{\beta f} = \begin{bmatrix} 0 & 0 \\ 0 & K_\beta \end{bmatrix}, \tag{5.32}$$

in order to simplify the representation in (5.30a).

For a three-phase system, the state vector will increase by a factor of four for each harmonic suppression, i.e., $\boldsymbol{x}(t) \in \mathbb{R}^{2+4m}$, where m is the number of harmonic eliminations. For the general case of m harmonic suppressions, the state vector will be

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{i}_{\alpha\beta}^2(t) & i_{\alpha f,1}(t) & \dot{i}_{\alpha f,1}(t) & i_{\beta f,1}(t) & \dot{i}_{\beta f,1}(t) & \cdots & i_{\alpha f,m}(t) & \dot{i}_{\alpha f,m}(t) & i_{\beta f,m}(t) & \dot{i}_{\beta f,m}(t) \end{bmatrix}^T, \tag{5.33}$$

while the state and input matrices are

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_{RL} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \cdots & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \boldsymbol{G}_{\alpha f,1} & \boldsymbol{F}_{\alpha f,1} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \cdots & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \boldsymbol{G}_{\beta f,1} & \mathbf{0}_{2\times2} & \boldsymbol{F}_{\beta f,1} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \boldsymbol{G}_{\alpha f,2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \boldsymbol{F}_{\alpha f,2} & \mathbf{0}_{2\times2} & \ddots & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \boldsymbol{G}_{\beta f,2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \boldsymbol{F}_{\beta f,2} & \ddots & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots & \vdots \\ \boldsymbol{G}_{\alpha f,m} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \cdots & \boldsymbol{F}_{\alpha f,m} & \mathbf{0}_{2\times2} \\ \boldsymbol{G}_{\beta f,m} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \cdots & \mathbf{0}_{2\times2} & \boldsymbol{F}_{\beta f,m} \end{bmatrix} \tag{5.34a}$$

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_{RL}^T & \mathbf{0}_{2\times3}^T & \mathbf{0}_{2\times3}^T & \mathbf{0}_{2\times3}^T & \mathbf{0}_{2\times3}^T & \cdots & \mathbf{0}_{2\times3}^T & \mathbf{0}_{2\times3}^T \end{bmatrix}^T. \tag{5.34b}$$

As described in Section 2.3.2, the discrete-time state-space representation can be obtained by using the exact discretisation, and will be in the form of (5.13). The output vector includes the orthogonal reference frame currents and the filter outputs

$$\boldsymbol{y}(k) = \begin{bmatrix} \boldsymbol{i}_{\alpha\beta}^T(t) & i_{\alpha f,1}(t) & i_{\beta f,1}(t) & \cdots & i_{\alpha f,m}(t) & i_{\beta f,m}(t) \end{bmatrix}^T, \tag{5.35}$$

where

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k), \tag{5.36}$$

with

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{5.37}$$

The cost function for N_p prediction steps is given by

$$J = \sum_{l=k}^{k+N_p-1} \left\| \boldsymbol{y}_{ref}(l+1) - \boldsymbol{y}(l+1) \right\|_{\tilde{\boldsymbol{R}}}^2 + \lambda_u \left\| \Delta \boldsymbol{u}(l) \right\|_2^2, \tag{5.38}$$

where

$$\boldsymbol{y}_{ref}(l+1) = \begin{bmatrix} i_{ref}(l+1) & i_{ref,f,1}(l+1) & i_{ref,f,2}(l+1) & \cdots & i_{ref,f,m}(l+1) \end{bmatrix} \tag{5.39}$$

and $\Delta \boldsymbol{u}(l) = \boldsymbol{u}(l) - \boldsymbol{u}(l-1)$. The weighting factor on the switching transitions is denoted by λ_u , and $\boldsymbol{R} \in \mathbb{R}^{(1+2m)\times(1+2m)}$ represents the diagonal penalty matrix. As mentioned in Section 5.3.1, the reference for the filters should be zero since the selected harmonics should be suppressed.

As already stated in Section 5.3.1, this harmonic elimination method requires no modification to the ILS problem described in Section 2.4. With the addition of the penalty matrix $\tilde{\boldsymbol{R}} \in \mathbb{R}^{N_p(1+2m)\times N_p(1+2m)}$, the calculation of the unconstrained solution $\boldsymbol{U}_{unc}(k)$ is slightly different to that in Section 2.4:

$$\boldsymbol{U}_{opt}(k) = \arg\min_{\boldsymbol{U}(k)} \left\| \bar{\boldsymbol{U}}_{unc}(k) - \boldsymbol{H}\boldsymbol{U}(k) \right\|_2^2 \tag{5.40a}$$

$$\text{subject to } \boldsymbol{U}(k) \in \mathbb{U}, \tag{5.40b}$$

where

$$\bar{\boldsymbol{U}}_{unc}(k) = \boldsymbol{H}\boldsymbol{U}_{unc}(k) \tag{5.41}$$

$$\boldsymbol{U}_{unc}(k) = -\boldsymbol{Q}^{-1}\boldsymbol{\Theta}(k) \tag{5.42}$$

$$\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{Q}, \tag{5.43}$$

with

$$\boldsymbol{Q} = \boldsymbol{\Upsilon}^T\tilde{\boldsymbol{R}}\boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^T\boldsymbol{S} \tag{5.44}$$

$$\boldsymbol{\Theta}(k) = ([\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k)]^T\tilde{\boldsymbol{R}}\boldsymbol{\Upsilon} - \lambda_u[\boldsymbol{E}\boldsymbol{u}(k-1)]^T\boldsymbol{S})^T. \tag{5.45}$$

It is worth mentioning that the computational burden associated with the calculation of the unconstrained solution $\boldsymbol{U}_{unc}(k)$ will increase with the number of suppressed terms. This is due to an increase in the number of state variables, and the influence can be seen in (5.45).

5.4 Simulation results

5.4.1 Simulation framework

MATLAB® will be used for simulation purposes. The optimization problem will be solved in the ILS form via sphere decoding, instead of the commonly-used exhaustive search based algorithm described in Section 2.3.4. The investigated plants are the single-phase and three-phase NPC inverter with an RL load shown in Figure 5.2 and Figure 5.3, respectively. Unless stated otherwise, the phase currents will have a reference current of 50 Hz, 12 A peak. Multiple fundamental periods are simulated and only the results during steady-state operation, in terms of THD and switching frequency, will be analysed. Only harmonic components up to 2 kHz will be shown for a clear and convenient presentation of the suppressed harmonics. Unless stated otherwise, the gain of the respective filters will be set to $K = 2\sigma\omega_n$ (see (5.1) for transfer function and (5.3) for gain at target frequency). The damping of the filter and weight on the suppressed terms will be adjusted until adequate suppression of the selected harmonics is achieved. The simulation parameters are shown in Table 5.1.

Table 5.1: Selective harmonic elimination simulation parameters.

Parameter	Description	Value
V_d	Bus voltage	100 V
R	Resistive load	2 Ω
L	Inductive load	2 mH
T_s	Sampling interval	25 µs

5.4.2 Evaluation of SHS-MPC for single-phase inverter

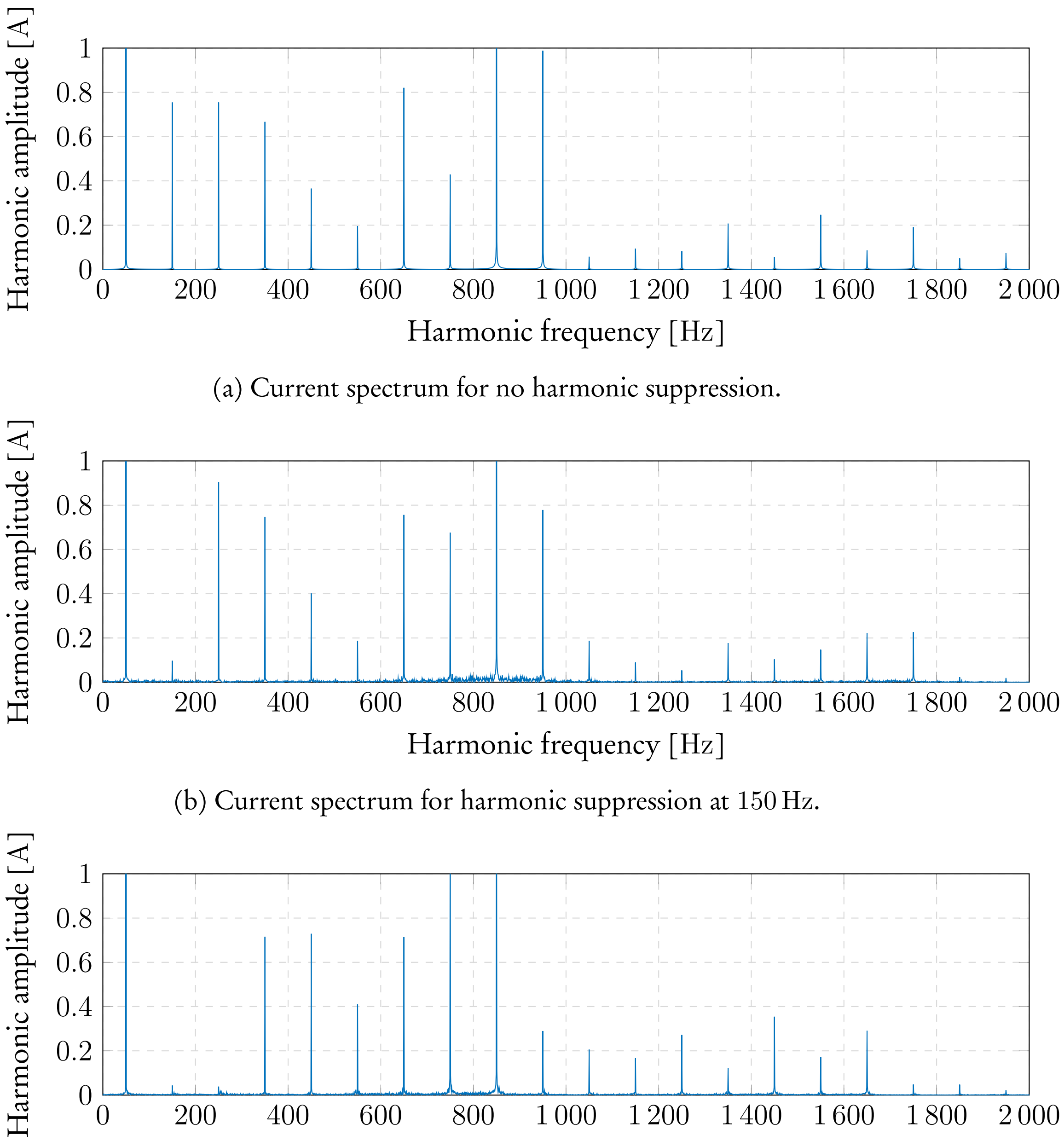
Before investigating SHS-MPC for a three-phase system, a straightforward single-phase NPC will be simulated to verify that SHS-MPC does indeed suppress harmonics. Simulations are done for a prediction horizon of $N_p = 1$.

Figure 5.4a shows the current spectrum for MPC.[4] The weighting factor is tuned to $\lambda_u = 4$, where a switching frequency of 300 Hz is obtained. Note that a near carrier-based discrete-like harmonic spectrum is obtained in Figure 5.4a. During simulations with ideal conditions, MPC tends to enter a near-periodic switching pattern, resulting in most of the harmonic energy to be focused into integer multiples of the fundamental frequency.

Figure 5.4b shows the current spectrum for SHS-MPC where the harmonic at 150 Hz is suppressed. The weighting factor is kept at $\lambda_u = 4$. From Figure 5.4b it is shown that the selected harmonic is significantly attenuated when compared to that of Figure 5.4a. The harmonic spectrum in Figure 5.4b has noticeable, although small and spread out, inter-harmonics between 600 Hz and 1050 Hz. The harmonic at 250 Hz is noticeably larger. The suppression method merely shifts the energy of the suppressed harmonic into other frequencies. It is interesting to observe that the switching frequency remains at 300 Hz. As stated in [1], MPC, and therefore reasonable to assume SHS-MPC as well, tend to lock into switching frequencies at integer multiples of the fundamental even if the weighting factor is varied.

In Figure 5.4c the current spectrum for SHS-MPC is shown, where harmonic suppressions at 150 Hz and 250 Hz are applied. The weighting factor is kept at $\lambda_u = 4$, and the resulting switching frequency still remains 300 Hz. It is seen that the selected harmonics are significantly attenuated. It can be observed that certain harmonics (e.g., at 450 Hz and 650 Hz) noticeably increase, while others decrease significantly (e.g., at 950 Hz).

[4] When just referring to MPC, it is implied that no harmonic suppression is active.



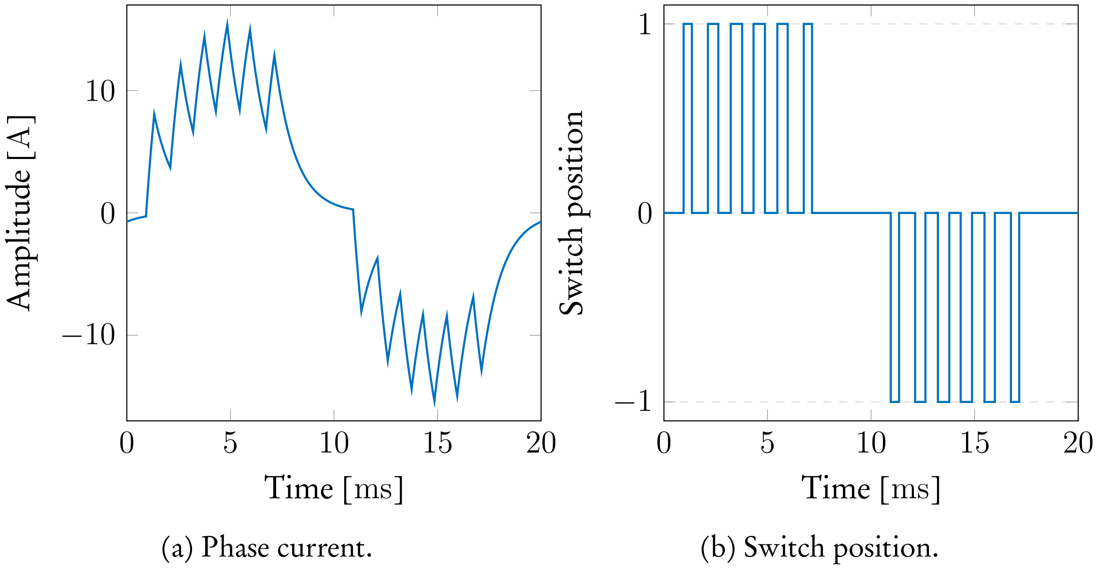
(a) Current spectrum for no harmonic suppression.

(b) Current spectrum for harmonic suppression at 150 Hz.

(c) Current spectrum for harmonic suppressions at 150 Hz and 250 Hz.

Figure 5.4: Current spectra for single-phase MPC and SHS-MPC simulations.

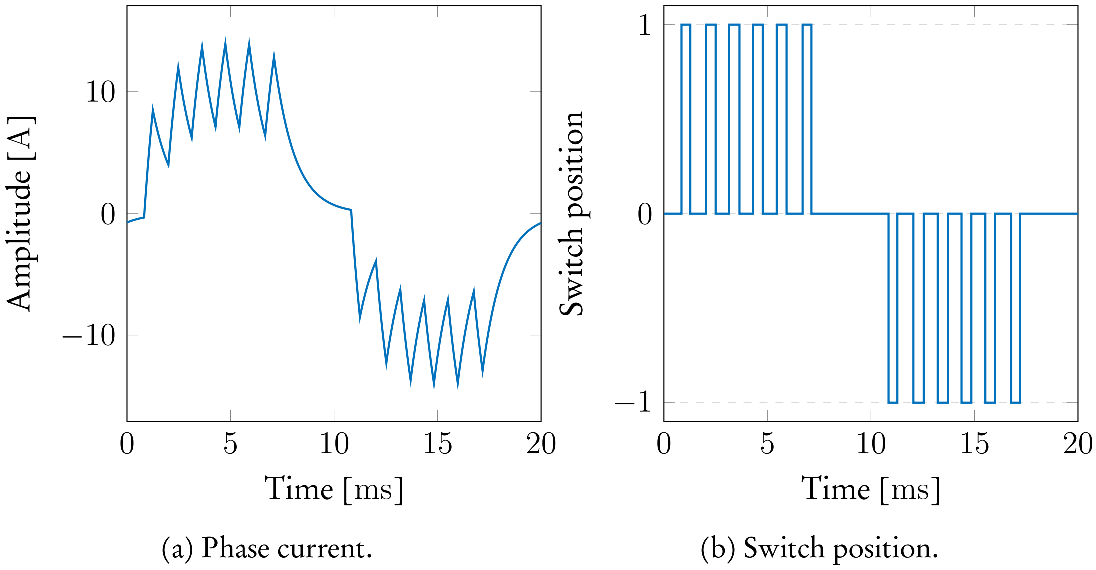
Figure 5.5, Figure 5.6, and Figure 5.7 show the current waveforms and switch positions for MPC, SHS-MPC with a term at 150 Hz, and SHS-MPC with terms at 150 Hz and 250 Hz, respectively. Although the current waveforms and switch positions seem similar, there is a key difference. The switch positions shown in Figure 5.6b and Figure 5.7b have varying pulse widths over time, leading to the harmonics at non-integer multiples of the fundamental frequency (see Figure 5.4b and Figure 5.4c). The pulse widths of Figure 5.5a are identical for all fundamental periods in steady-state, leading to the before mentioned discrete spectrum in Figure5.4a.



(a) Phase current.

(b) Switch position.

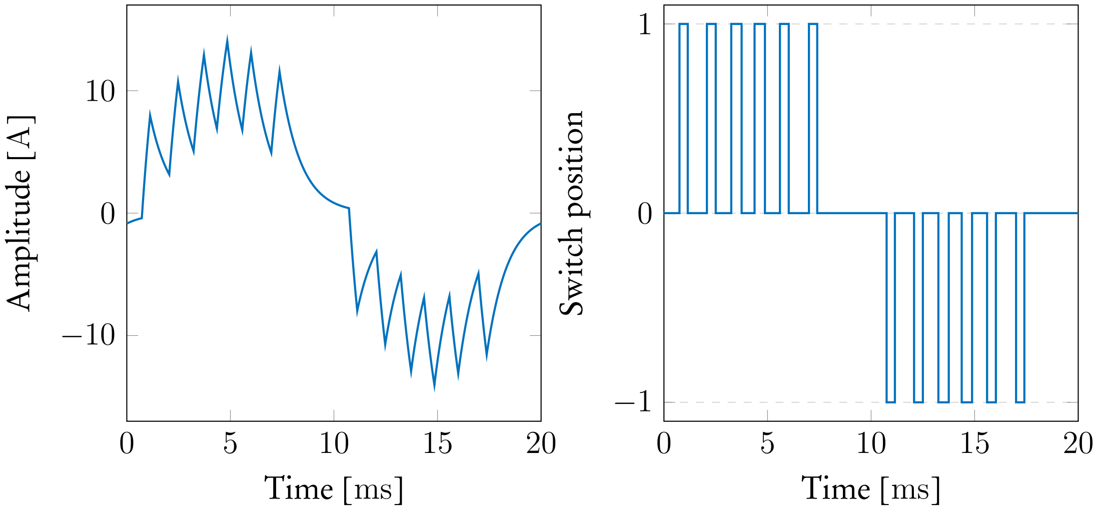
Figure 5.5: Current waveform and switch position of single-phase simulation for MPC.



(a) Phase current.

(b) Switch position.

Figure 5.6: Current waveform and switch position of single-phase simulation for SHS-MPC with a term at 150 Hz.



(a) Phase current.

(b) Switch position.

Figure 5.7: Current waveform and switch position of single-phase simulation for SHS-MPC with terms at 150 Hz and 250 Hz.

The filter parameters, penalties, and harmonic distortion for the simulations are shown in Table 5.2. As seen, a disadvantage of using SHS-MPC is a reduction in the current fundamental component, contributing to an increased harmonic distortion. This can be attributed to the inclusion of the 50 Hz component, which is relatively large in comparison to the other harmonics, in the filter output (see Figure 5.1 for magnitude response). Furthermore, the weight on the suppressed terms causes a trade-off between suppression and reference tracking of the phase currents.

Table 5.2: Comparison of single-phase SHS-MPC simulations. P_1 and P_2 denote the penalties on the suppressed frequencies. I_1 represents the amplitude for the fundamental component. The weighting factor and switching frequency for all of the simulations are $\lambda_u = 4$ and $f_{sw} =$ 300 Hz, respectively.

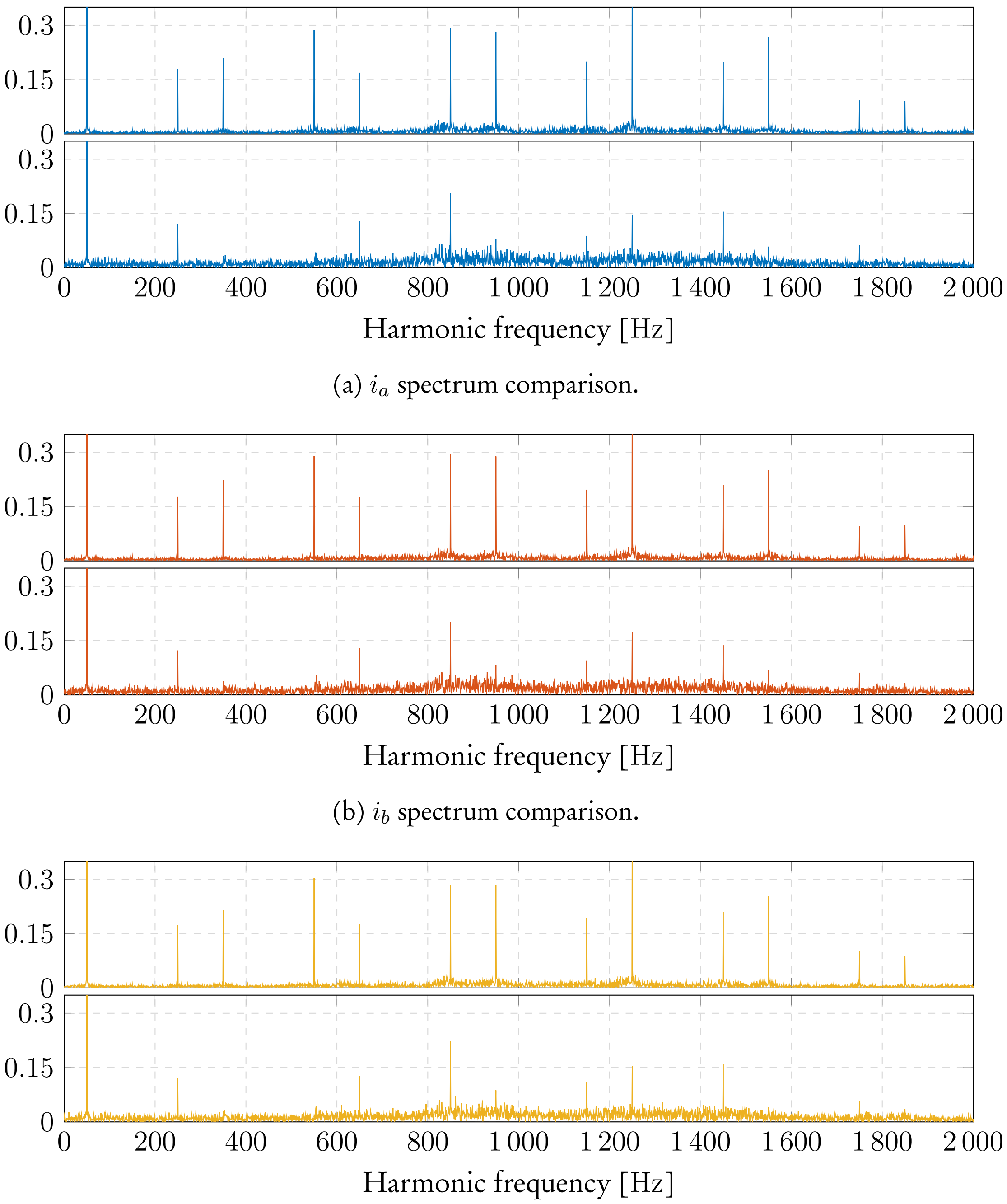
SHS terms	$w_{n,1}$ [rad/s]	σ_1 [rad/s]	$w_{n,2}$ [rad/s]	σ_2 [rad/s]	P_1	P_2	I_1 [A]	THD [%]
0							11.6	23
1	$2\pi150$	2.5			20		10.9	24.7
2	$2\pi150$	5	$2\pi250$	5	30	7	10.3	25.8

5.4.3 Evaluation of SHS-MPC for three-phase inverter

For the three-phase simulations, uniformly distributed pseudorandom current values from -0.0075 A to 0.0075 A are added to the orthogonal reference current measurement of the controller, acting as quantization noise.[5] This is done in order to prevent the controller from entering a fixed switching pattern, as preliminary simulations resulted in peculiar harmonic spectra. This is known as *dither*, and is used in digital processing in order to prevent large-scale patterns [39].

[5] Note that noise is only added to the measurement and not to the state variable itself.

Harmonic suppression during steady-state operation



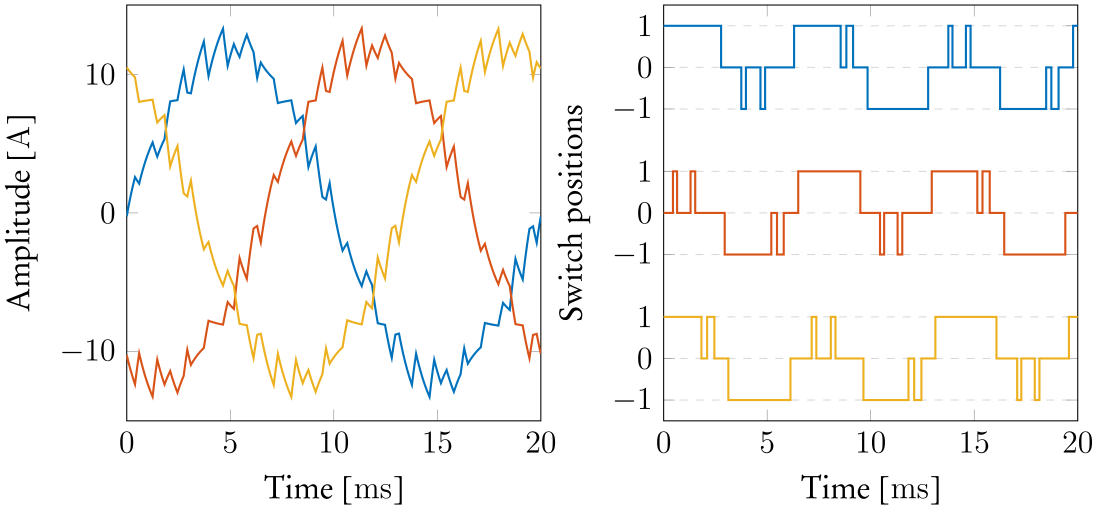
(a) i_a spectrum comparison.

(b) i_b spectrum comparison.

(c) i_c spectrum comparison.

Figure 5.8: Phase-current spectra comparison for three-phase simulations of MPC and SHS-MPC with $N_p = 1$. The top graphs show the current spectra for MPC and the bottom graphs show those of SHS-MPC. Magnitudes are given in ampere. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

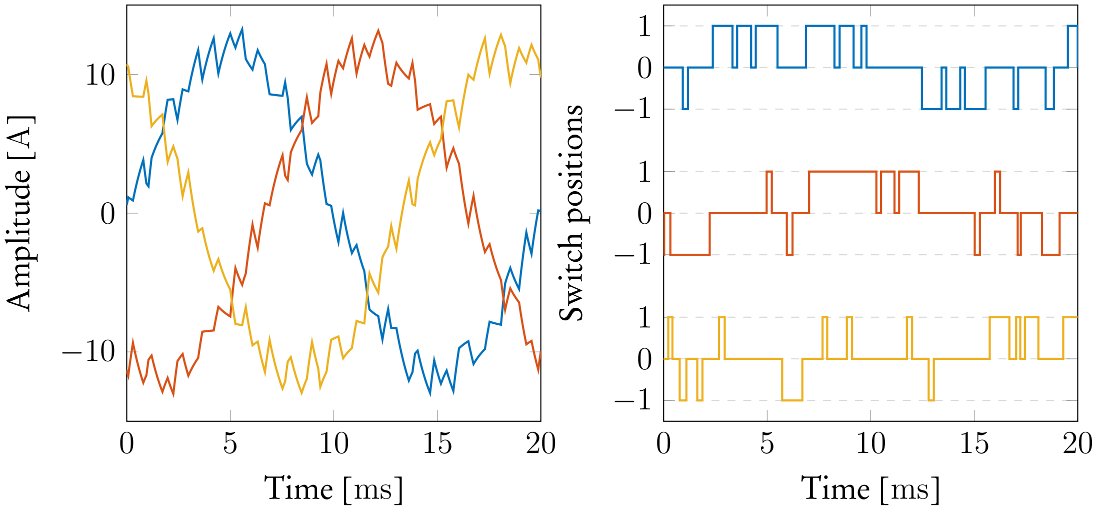
Figure 5.8 shows the comparison of the phase-current spectra between MPC and SHS-MPC with harmonic suppressions at 350 Hz and 550 Hz. The horizon is set to $N_p = 1$. The weighting factor is adjusted to $\lambda_u = 1$, which results in switching frequencies of 253 Hz and 275 Hz for MPC and SHS-MPC, respectively. Unlike the single-phase cases without dithering, the switching frequency changes when harmonics are suppressed. It is apparent that the harmonics at 350 Hz and 550 Hz are significantly attenuated. Also, as seen in the single-phase results, certain harmonics that are not included to be suppressed are significantly reduced, e.g, at 950 Hz and 1550 Hz. However, the noise floor does noticeably increase, since the energy of the suppressed harmonics is spread throughout the spectrum.



(a) Phase currents.

(b) Switch positions.

Figure 5.9: Current waveform and switch position of MPC with $N_p = 1$. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

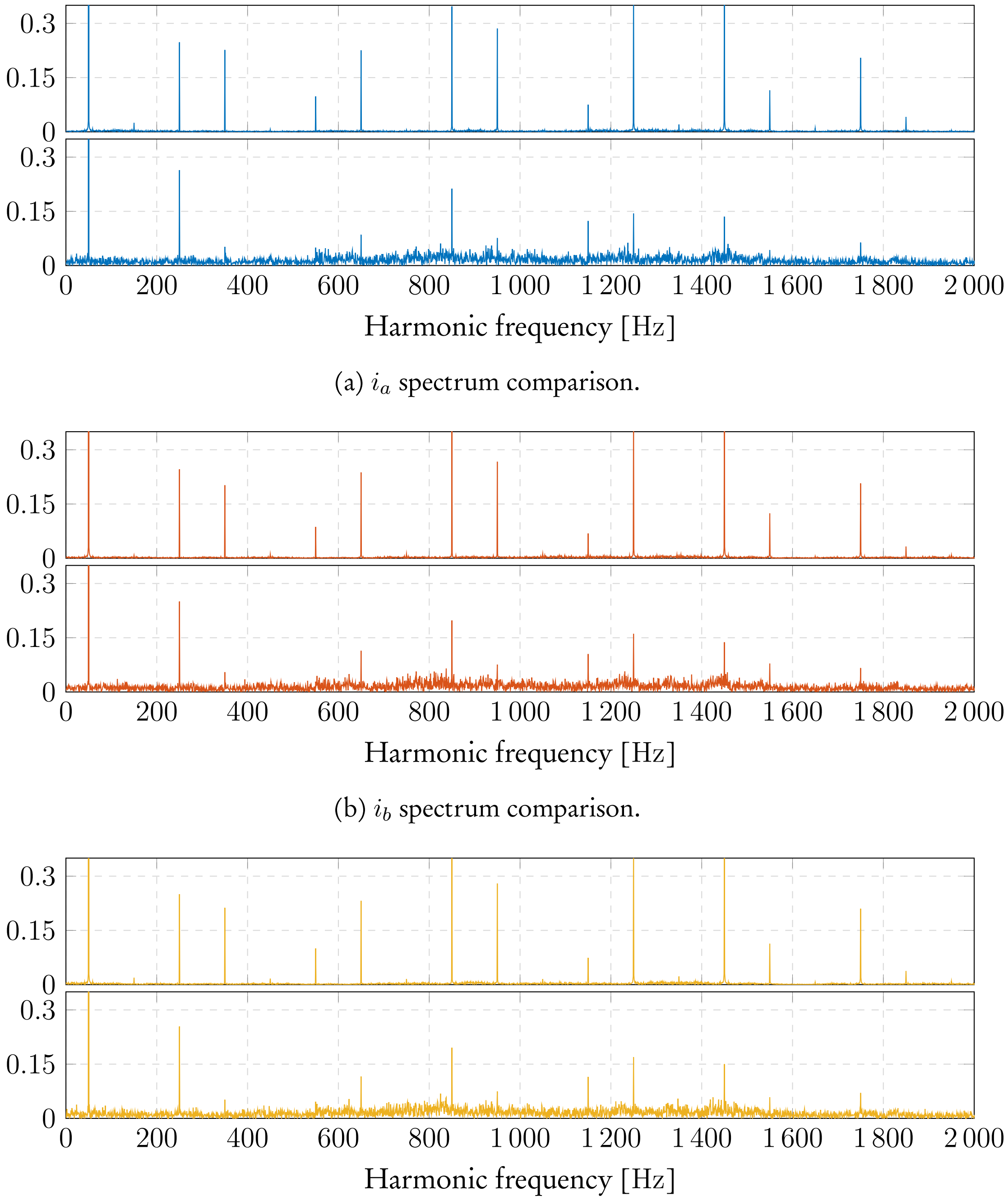


(a) Phase currents.

(b) Switch positions.

Figure 5.10: Current waveform and switch position of SHS-MPC with $N_p = 1$. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

The current waveforms and switch positions for MPC and SHS-MPC are shown in Figure 5.9 and Figure 5.10, respectively. Although neither MPC nor SHS-MPC had a fixed switching patterns, MPC does have noticeable half-wave symmetry and symmetry between the phases. The symmetry between the phases can also be seen in the phase currents in Figure 5.9a.



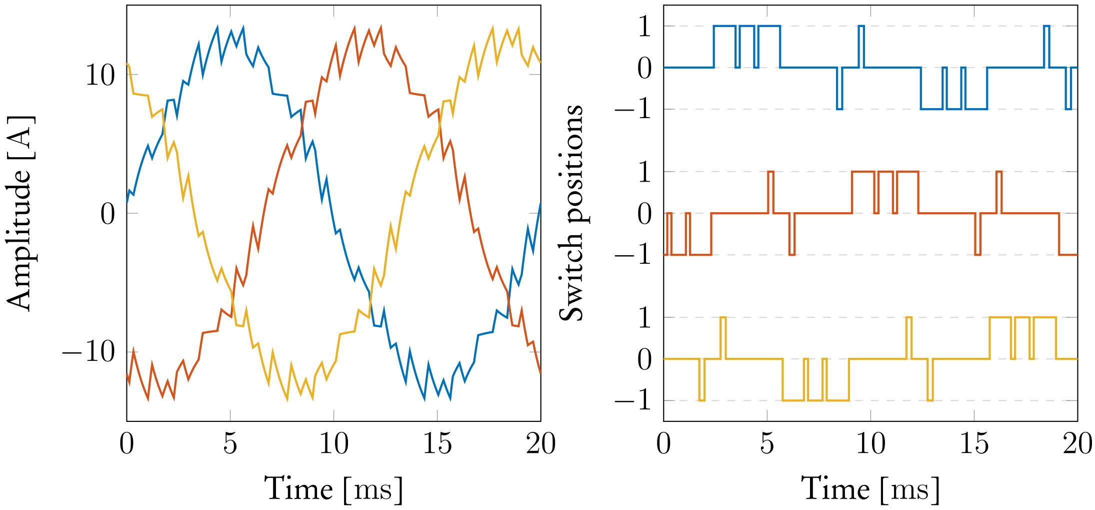
(a) i_a spectrum comparison.

(b) i_b spectrum comparison.

(c) i_c spectrum comparison.

Figure 5.11: Phase-current spectra comparison for three-phase simulations of MPC and SHS-MPC with $N_p = 5$. The top graphs show the current spectra for MPC and the bottom graphs show those of SHS-MPC. Magnitudes are given in ampere. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

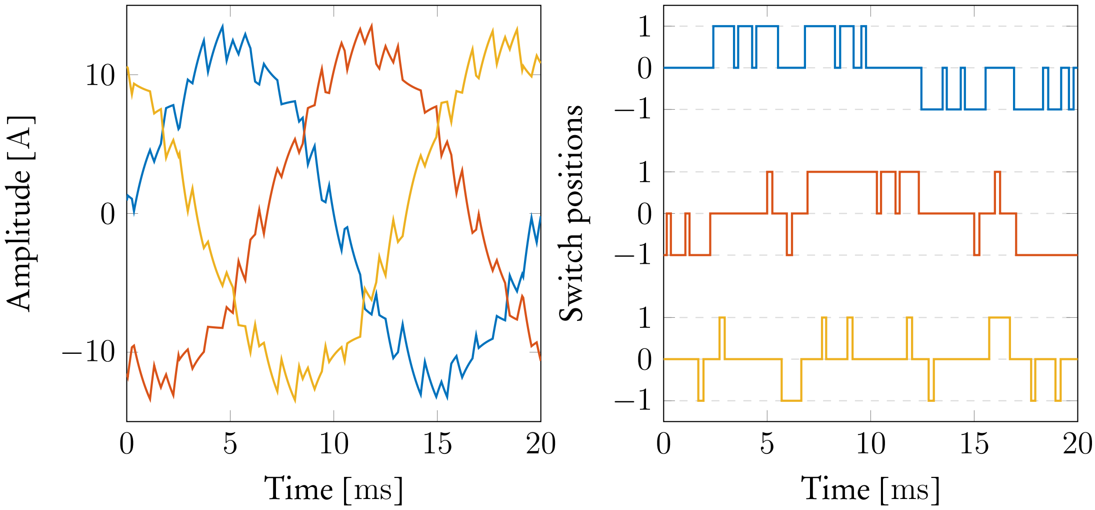
The phase-current spectra for MPC and SHS-MPC with harmonic suppressions at 350 Hz and 550 Hz are shown in Figure 5.11. The horizon is set to $N_p = 5$. Even with the addition of dithering, MPC entered a fixed switching pattern, giving rise to the discrete spectrum with barely noticeable inter-harmonics. The weighting factor is adjusted to $\lambda_u = 13$, resulting in switching frequencies of 250 Hz and 270 Hz for MPC and SHS-MPC, respectively. The addition of the harmonic suppression inhibited the fixed switching pattern of MPC, and a variable switching pattern is obtained with SHS-MPC. The current waveforms and switch positions for MPC and SHS-MPC are shown in Figure 5.12 and Figure 5.13, respectively. It is interesting to observe that SHS-MPC has a form of half-wave symmetry, as seen in Figure 5.13b. However, the half-wave symmetry is not manifested during all of the fundamental periods. The computational burden associated with sphere decoding, in terms of nodes visited, is near identical to that of Figure 3.12.



(a) Phase currents.

(b) Switch positions.

Figure 5.12: Current waveform and switch position of MPC with $N_p = 5$. Phase a, b, and c are indicated by blue, red, and yellow, respectively.

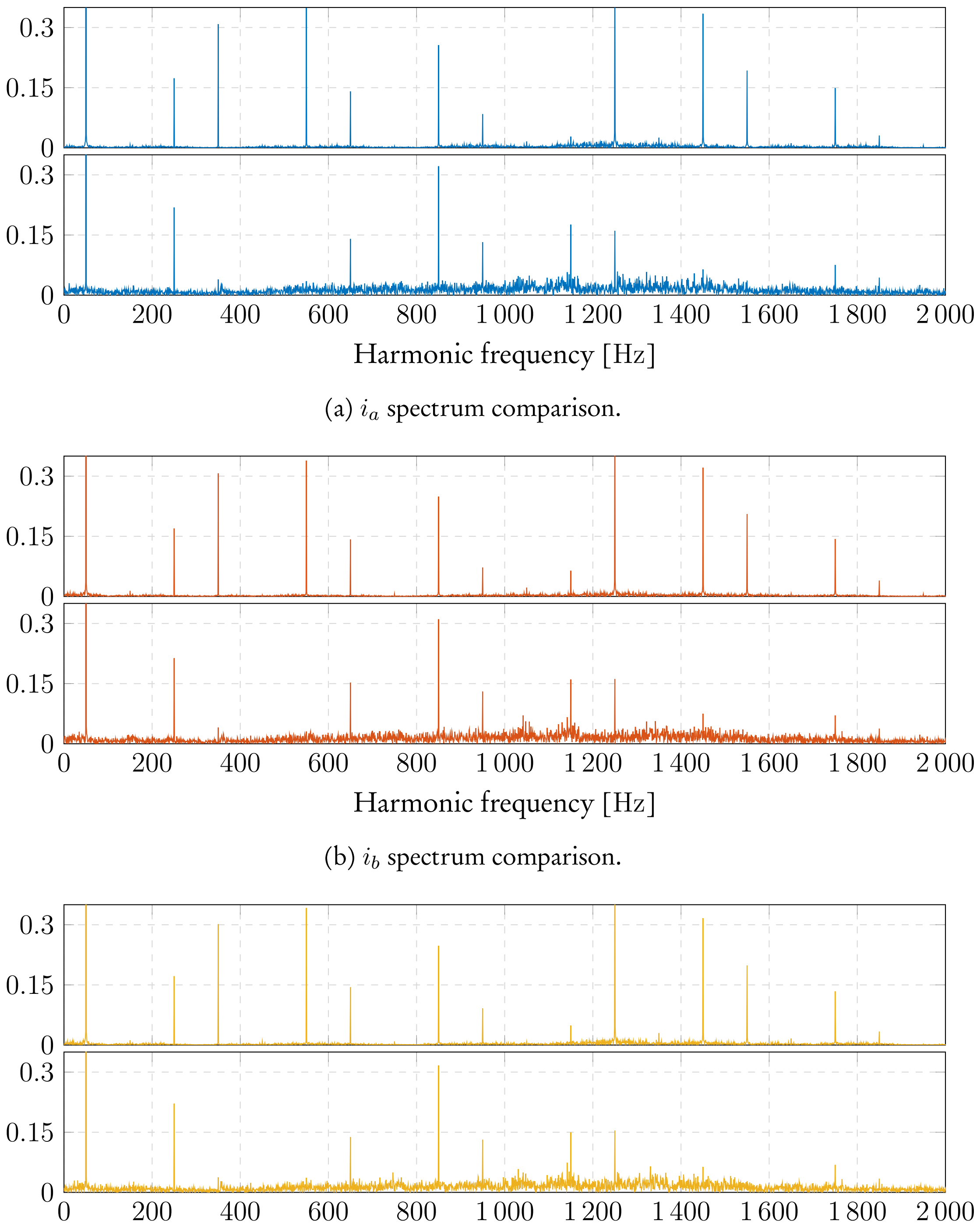


(a) Phase currents.

(b) Switch positions.

Figure 5.13: Current waveform and switch position of SHS-MPC with $N_p = 5$. Phase a, b, and c are indicated by blue, red, and yellow, respectively.

The horizon is increased to $N_p = 15$, and the phase-current spectra for MPC and SHS-MPC with harmonic suppressions at 350 Hz and 550 Hz are shown in Figure 5.14. The weighting factor for MPC is set to $\lambda_u = 19$, where a switching frequency of 250 Hz is obtained. The weighting factor for SHS-MPC is changed to $\lambda_u = 22$ so that a switching frequency of 273 Hz is obtained, similar to the previous SHS-MPC simulations.



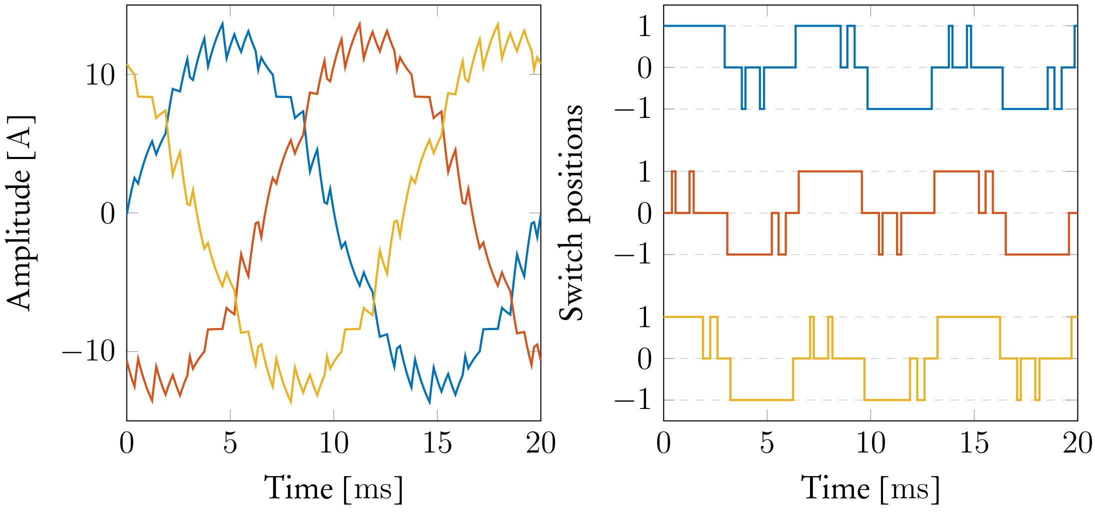
(a) i_a spectrum comparison.

(b) i_b spectrum comparison.

(c) i_c spectrum comparison.

Figure 5.14: Phase-current spectra comparison for three-phase simulations of MPC and SHS-MPC with $N_p = 15$. The top graphs show the current spectra for MPC and the bottom graphs show those of SHS-MPC. Magnitudes are given in ampere. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

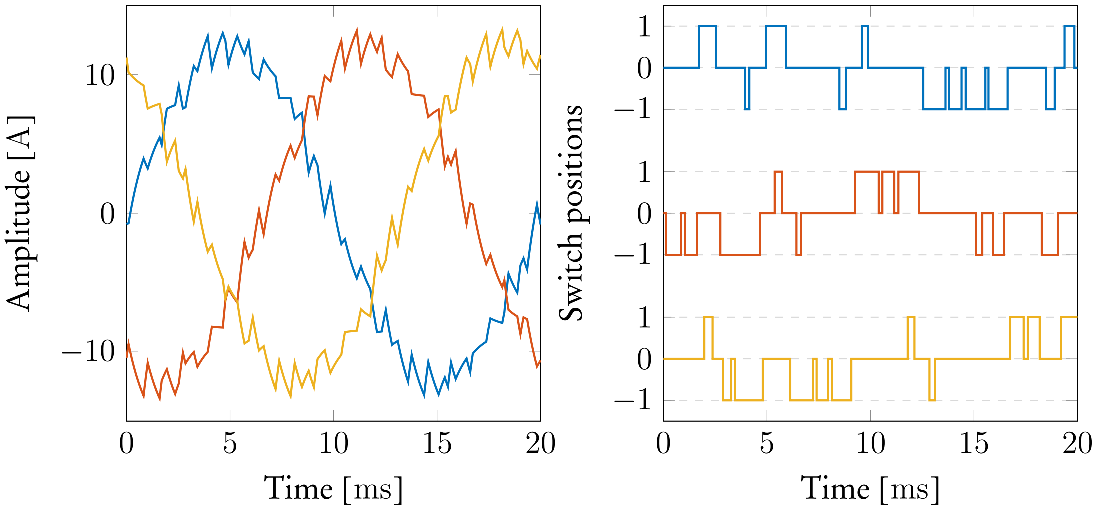
The harmonics at 350 Hz and 550 Hz are significantly suppressed when using SHS-MPC. Note that the harmonic at 550 Hz for MPC is noticeably larger than the horizon $N_p = 5$ case in Figure 5.11. The current waveforms and switch positions for MPC and SHS-MPC are shown in Figure 5.15 and Figure 5.16, respectively.



(a) Phase currents.

(b) Switch positions.

Figure 5.15: Current waveform and switch position of MPC with $N_p = 15$. Phases a, b, and c are indicated by blue, red, and yellow, respectively.



(a) Phase currents.

(b) Switch positions.

Figure 5.16: Current waveform and switch position of SHS-MPC with $N_p = 15$. Phases a, b, and c are indicated by blue, red, and yellow, respectively.

Table 5.3 shows the filter parameters, penalties, and harmonic distortion for the three-phase simulations. First and foremost, the advantage that MPC with a horizon of $N_p = 5$ has over $N_p = 1$, in terms of harmonic performance, is apparent. Increasing the horizon to $N_p = 15$ only yields a marginal improvement over a horizon of $N_p = 5$. For SHS-MPC with $N_p = 5$, the improvement is marginal over $N_p = 1$, even if the switching frequency decreased by 5 Hz. However, when adopting a horizon of $N_p = 15$, the benefit of long horizons is apparent.

Table 5.3: Comparison of three-phase SHS-MPC simulations. P_1 and P_2 denote the penalties on the suppressed frequencies. I_1 represents the amplitude for the fundamental component. The weighting factors for the horizon $N_p = 1$ cases are kept at $\lambda_u = 1$, and for the horizon $N_p = 5$ cases are kept at $\lambda_u = 13$. For $N_p = 15$, the weighting factors are $\lambda_u = 19$ and $\lambda_u = 22$ for MPC and SHS-MPC, respectively.

N_p	f_{sw}	$w_{n,1}$	σ_1	$w_{n,2}$	σ_2	P_1	P_2	I_1	THD
	[Hz]	[rad/s]	[rad/s]	[rad/s]	[rad/s]			[A]	[%]
1	253							12	8.3
1	275	$2\pi350$	5	$2\pi550$	4	0.2	0.1	11.8	8.7
5	250							12.3	7.6
5	270	$2\pi350$	5	$2\pi550$	4	0.01	0.003	12.1	8.5
15	250							12.5	7.5
15	273	$2\pi350$	5	$2\pi550$	4	0.0018	0.00014	12.1	8

Response time

One of the benefits of using MPC is the fast response time. Ideally, the SHS-MPC formulation should not impede the response time of the controller. In Figure 5.17, a reference step is given to 0 A and then back to 12 A for SHS-MPC with $N_p = 5$. As can be seen, the controller quickly regulates the phase currents to the new set-points; approximately 0.35 ms for the step down case and 0.5 ms for the step up case. The overshoot and settling time to 0 A, most noticeable in phase c, is due to the penalty on the switching transitions. The switch positions are shown in Figure 5.18. Figure 5.19 shows the same reference step applied to MPC. It can be seen that the response time with or without harmonic suppression is near identical.

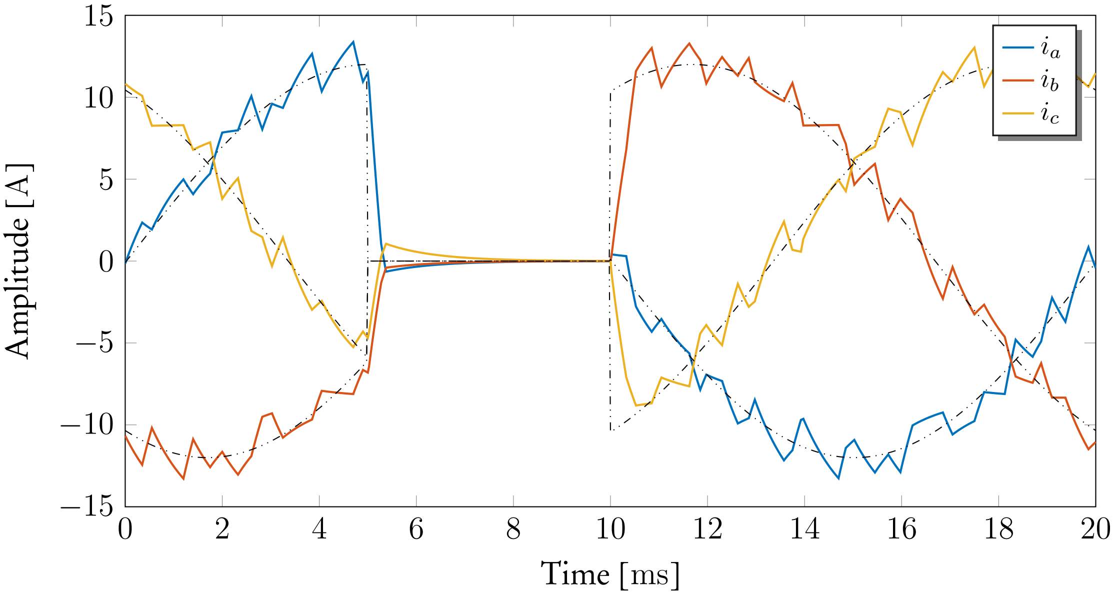


Figure 5.17: The current waveforms of SHS-MPC for reference step changes. The phase references are shown by the black dash dotted lines.

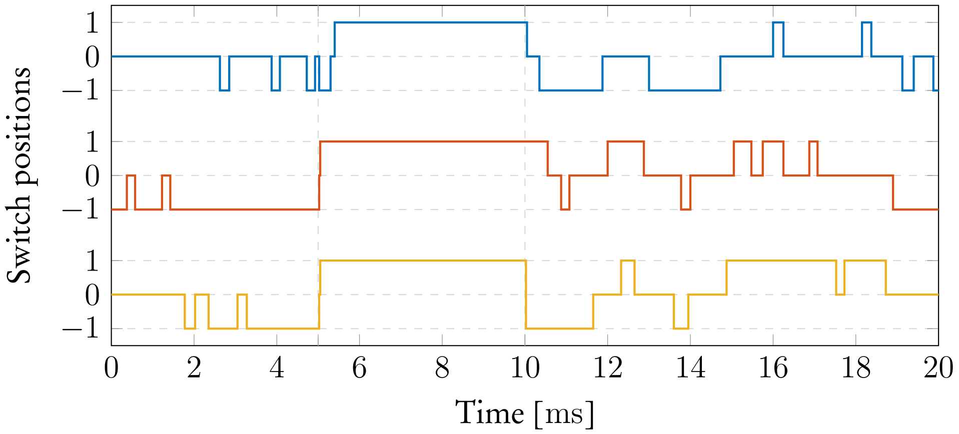


Figure 5.18: The switch positions of SHS-MPC during reference steps. The switch positions for phases a, b, and c are indicated by blue, red, and yellow, respectively. The grid lines on the x-axis represent the reference step points.

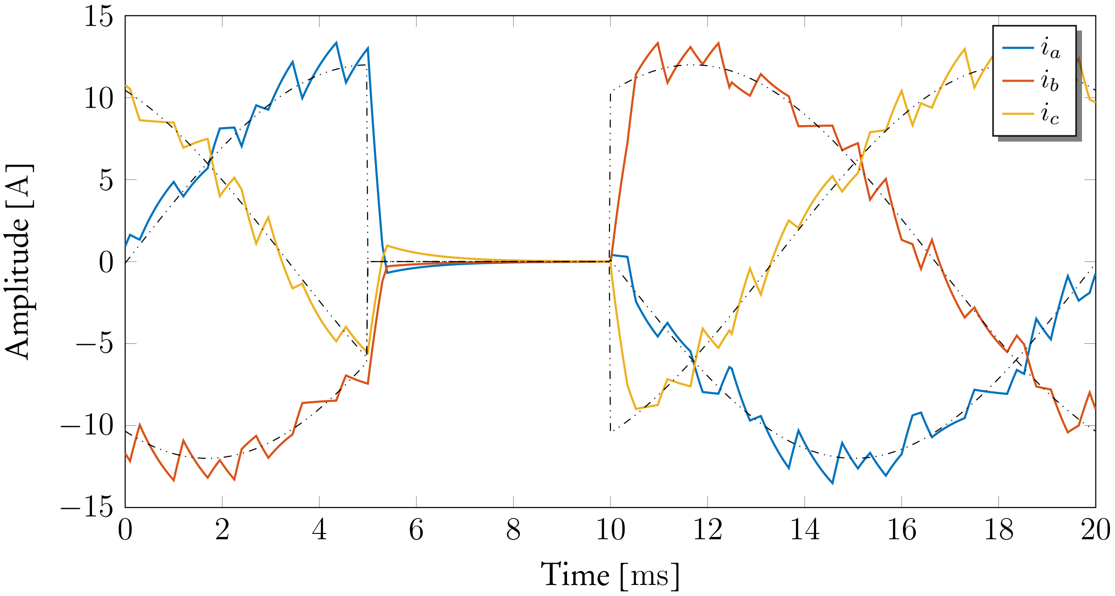


Figure 5.19: The current waveforms of MPC for reference step changes. The phase references are shown by the black dash dotted lines.

5.5 Summary

This chapter introduced a method to suppress harmonics at selected frequencies. The concept is straightforward and can immediately be implemented with the reformulated ILS problem, making it possible to be used with long horizons. The computational burden of the sphere decoder is minimally affected by the addition of harmonic suppressions. Long horizons are shown to improve harmonic performance of SHS-MPC. A decrease of 6.3 % in current THD is obtained when adopting a horizon of $N_p = 15$ instead of $N_p = 5$.

In [36], MPC was used to control an LC filter connected to an induction machine, also a second-order filter with a transfer function similar to that used in SHS-MPC. Very long horizons in time (i.e., N_pT_s) were required to achieve a performance benefit. The sampling interval was chosen as $T_s = 125\,\mu\text{s}$ due to the low switching frequency and slow dynamics of the LC filter. This is in contrast with the simulations conducted in this section, where a sampling interval of $T_s = 25\,\mu\text{s}$ was chosen. This results in a much shorter horizon in time when compared to [36].

It is worth mentioning that the proposed method does have its drawbacks. First of all, tuning is no trivial matter. Some effort is required to tune the damping of the filter and penalties associated with the suppressed harmonic terms. With these aforementioned inconveniences, incorrect tuning resulted in resonance at the selected suppressed frequency. Secondly, from a practical standpoint, the implementation of the proposed method will require more resources to implement than MPC. While the implementation of the sphere decoder will stay identical, the resources allocated for the calculation of the unconstrained solution will increase along with the number of suppressed terms.

Chapter 6

Conclusions and recommendations for future work

The following chapter concludes the thesis. Summaries of the main results for relevant chapters are given and recommendations are proposed for future work on the topics discussed in this thesis.

6.1 Overview of main results

6.1.1 FPGA implementation

By carefully considering the structure of the code and usage of available resources, direct MPC with a prediction horizon of $N_p = 5$ was successfully implemented within a Cyclone® V FPGA. A non-recursive sphere decoding algorithm was presented, which is ideal for implementation within an FPGA. A verification simulation executed on the FPGA was shown to be identical to a simulation conducted on MATLAB®. The optimization problem was solved in only $3\,\mu\text{s}$ in nearly $90\,\%$ of the cases. When the calculations required for the unconstrained solution and initial radius of the problem were also considered, the FPGA required $8.4\,\mu\text{s}$ to execute the control law. Considering that the sampling interval was $25\,\mu\text{s}$, and that a conservative clock frequency was used, the results are encouraging.

6.1.2 Performance evaluation of long horizons

The decrease in current harmonic distortions that is demonstrated in existing literature, based on simulations, is verified with practical experimental results. Note that this does not hold true for all switching frequencies, which was also observed in simulations of [15]. Even though a straightforward RL load was used, a decrease of $8.5\,\%$ in THD was achieved at a switching frequency of $250\,\text{Hz}$ when adopting a prediction horizon $N_p = 5$ instead of $N_p = 1$. When realising that the benefits of long horizons are more pronounced for higher-order and complex plants, the results proved to be promising.

The comparison between MPC, OPPs, and SVM demonstrated that MPC performs extremely well with the alternative modulation schemes, at least at low switching frequencies under $450\,\text{Hz}$. MPC did significantly outperform SVM at switching frequencies lower than $300\,\text{Hz}$, and also unexpectedly outperformed OPPs. However, this particular result (i.e., MPC outperforming OPPs) should not be taken at face value and expected to be the norm. The low output current did benefit MPC, since the output voltage can be lowered by using half of the available bus voltage by only applying the short voltage vectors. More importantly, the formulation of OPPs assumes that the load resistance can be neglected during the optimization stage. For the given system parameters, this assumption is invalid when considering low frequency harmonics.

The response time of the controller with a prediction horizon of $N_p = 5$ was demonstrated to react fast and regulate at new set-points. Although the optimal solution was found during transients for the given tests, this cannot always be guaranteed.

6.1.3 Selective harmonic suppression for long horizons

By including filters, which significantly amplify target harmonics, inside the state-space representation of the plant, selected harmonics can be significantly suppressed as demonstrated in this thesis. The proposed method is immediately compatible with the integer least-squares problem underlying long-horizon direct MPC. Although the desired harmonics are attenuated, the current THD and switching frequency do increase compared to when no harmonic suppressions are applied. A decrease of $6.3\,\%$ in current THD was achieved when using a horizon of $N_p = 15$ instead of $N_p = 5$.

The response time of the controller with the inclusion of harmonic suppression was shown to be minimally affected, at least for the given simulations.

6.2 Recommendations for future work

6.2.1 FPGA implementation

It is possible to better utilise the available resources of the FPGA. The die of the FPGA used included an ARM Cortex™ processor, which was not utilised in this thesis. Some of the arithmetic can be off-loaded to the ARM processor, which will free up FPGA resources. Also, instead of using MATLAB® to calculate the matrices when the weighting factor changes, the ARM processor can be used.

The calculations of the unconstrained solution, initial radius, and those calculated during sphere decoding can be exploited. A single process can be used that exclusively does most of the required arithmetic. By using logic checks and changing coefficients, this can be achieved. An example on how this can be done is given in Algorithm 6.

Algorithm 6 Universal multiplier

```
if Unconstrained solution should be calculated then
    $\boldsymbol{M} = -\boldsymbol{Q}^{-1}$
    $\boldsymbol{V} = \boldsymbol{\theta}$
else if Initial radius should be calculated, or sphere decoding is active then
    $\boldsymbol{M} = -\boldsymbol{H}$
    $\boldsymbol{V} = \boldsymbol{U}$
end if
$V_{temp,j} = M_{(j,1)}V_1 + M_{(j,2)}V_2 + \ldots + M_{(j,3N_p)}V_{3N_p}$
```

Naturally, the squaring of the term and accumulation of previous squared terms for the initial radius and sphere decoding is still required after the implementation of Algorithm 6.

6.2.2 Performance evaluation of long horizons

It is recommended that long-horizon direct MPC should be considered for more complex plants, such as induction machines, grid-connected inverters, and higher-order plants that include, for example, an LC filter. It is shown in [36] that long horizons significantly reduced the harmonic distortions for an LC filter connected to an induction machine.

During testing, only the weighting factor was adjusted to dictate the switching frequency. It is recommended that a wide range of sampling intervals, alongside different weighting factors, should be considered in order to investigate the effect of the prediction horizon on the harmonic distortions; similar to the Monte Carlo simulations in [15].

6.2.3 Selective harmonic suppression for long horizons

To conduct a thorough investigation on the benefits of long horizons for the proposed suppression method, it is recommended that a wide variety of damping factors, penalties on the suppressed terms, and sampling intervals should be considered. This can be achieved by using Monte Carlo simulations.

Different transfer functions can also be considered that will shape the current spectra differently. For example, at the time of this writing a new method proposed in [40] resulted in the switching energy to be distributed evenly throughout the harmonic spectrum, and resulted in very low harmonic distortions.

Appendices

Appendix A

Vector form of cost function

The derivation of the cost function in vector form follows that that was presented in [1].

The (quadratic) cost function for N_p prediction steps is given by

$$J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \sum_{l=k}^{k+N_p-1} \left\| \boldsymbol{y}_{ref}(l+1) - \boldsymbol{y}(l+1) \right\|_{\tilde{\boldsymbol{R}}}^2 + \lambda_u \left\| \boldsymbol{u}(l) - \boldsymbol{u}(l-1) \right\|_2^2, \tag{A.1}$$

with

$$\boldsymbol{y}(l+1) = \boldsymbol{C}\boldsymbol{x}(l+1) \tag{A.2a}$$

$$\boldsymbol{x}(l+1) = \boldsymbol{A}\boldsymbol{x}(l) + \boldsymbol{B}\boldsymbol{u}(l). \tag{A.2b}$$

For the general case, the output at time-step $l+1$ based on the state vector at k is given by

$$y(l+1) = \boldsymbol{C}\boldsymbol{A}^{l-k+1}\boldsymbol{x}(k) + \boldsymbol{C}\boldsymbol{B}\boldsymbol{A}^{l-k}\boldsymbol{u}(k) + \ldots + \boldsymbol{C}\boldsymbol{A}^{0}\boldsymbol{B}\boldsymbol{u}(l), \tag{A.3}$$

for $l = k, \ldots, k + N_p - 1$. By introducing the switching sequences $\boldsymbol{U}(k) \in \mathcal{U}^{n_u N_p}$ over the prediction horizon as

$$\boldsymbol{U}(k) = \begin{bmatrix} \boldsymbol{u}^T(k) & \boldsymbol{u}^T(k+1) & \boldsymbol{u}^T(k+2) & \cdots & \boldsymbol{u}^T(k+N_p-1) \end{bmatrix}^T, \tag{A.4}$$

the output trajectory $\boldsymbol{Y}(k)$ over the prediction horizon

$$\boldsymbol{Y}(k) = \begin{bmatrix} \boldsymbol{y}(k+1) & \boldsymbol{y}(k+2) & \boldsymbol{y}(k+3) & \cdots & \boldsymbol{y}(k+N_p) \end{bmatrix}^T, \tag{A.5}$$

is given by

$$\boldsymbol{Y}(k) = \boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{U}(k), \tag{A.6}$$

where the prediction matrices are given as

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{A} \\ \boldsymbol{C}\boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{N_p} \end{bmatrix} \text{ and } \boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{B} & \boldsymbol{0}_{n_y \times n_u} & \cdots & \boldsymbol{0}_{n_y \times n_u} \\ \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} & \cdots & \boldsymbol{0}_{n_y \times n_u} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}\boldsymbol{A}^{N_p-1}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{N_p-2}\boldsymbol{B} & \cdots & \boldsymbol{C}\boldsymbol{B} \end{bmatrix}.$$

The term of (A.1) that penalises the tracking error can be written as

$$\begin{aligned} J_{err} &= \sum_{l=k}^{k+N_p-1} \left\| \boldsymbol{y}_{ref}(l+1) - \boldsymbol{y}(l+1) \right\|_{\boldsymbol{R}}^2 \\ &= \begin{bmatrix} \boldsymbol{y}_e^T(k+1) & \cdots & \boldsymbol{y}_e^T(k+N_p) \end{bmatrix} \tilde{\boldsymbol{R}} \begin{bmatrix} \boldsymbol{y}_e^T(k+1) & \cdots & \boldsymbol{y}_e^T(k+N_p) \end{bmatrix}^T, \end{aligned} \tag{A.7}$$

where $\tilde{\boldsymbol{R}} \in \mathbb{R}^{n_y N_p \times n_y N_p}$ is introduced as the penalty matrix over the prediction horizon, and $\boldsymbol{y}_e(l)$ as the error between the predicted and reference current. With

$$\boldsymbol{Y}_{ref}(k) - \boldsymbol{Y}(k) = \begin{bmatrix} \boldsymbol{y}_e^T(k+1) & \boldsymbol{y}_e^T(k+2) & \cdots & \boldsymbol{y}_e^T(k+N_p) \end{bmatrix}^T, \tag{A.8}$$

and by inserting (A.6), (A.7) can be written as

$$J_{err} = \left\| \boldsymbol{Y}_{ref}(k) - \boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{U}(k) \right\|^2_{\tilde{\boldsymbol{R}}}, \tag{A.9}$$

where $\boldsymbol{Y}_{ref}(k)$ represents the reference over the prediction horizon

$$\boldsymbol{Y}_{ref}(k) = \begin{bmatrix} \boldsymbol{y}_{ref}^T(k+1) & \boldsymbol{y}_{ref}^T(k+2) & \boldsymbol{y}_{ref}^T(k+3) & \cdots & \boldsymbol{y}_{ref}^T(k+N_p) \end{bmatrix}^T. \tag{A.10}$$

Similarly, the term of (A.1) that penalises the switching transitions can be written as

$$\begin{aligned} J_{\lambda_u} &= \lambda_u \sum_{l=k}^{k+N_p-1} \left\| \boldsymbol{u}(l) - \boldsymbol{u}(l-1) \right\|_2^2 \\ &= \lambda_u \sum_{l=k}^{k+N_p-1} (\boldsymbol{u}(l) - \boldsymbol{u}(l-1))^T (\boldsymbol{u}(l) - \boldsymbol{u}(l-1)) \\ &= \lambda_u \left\| \boldsymbol{S}\boldsymbol{U}(k) - \boldsymbol{E}\boldsymbol{u}(k-1) \right\|_2^2, \end{aligned} \tag{A.11}$$

where

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{I}_{n_u} & \boldsymbol{0}_{n_u \times n_u} & \cdots & \boldsymbol{0}_{n_u \times n_u} \\ -\boldsymbol{I}_{n_u} & \boldsymbol{I}_{n_u} & \cdots & \boldsymbol{0}_{n_u \times n_u} \\ \boldsymbol{0}_{n_u \times n_u} & -\boldsymbol{I}_{n_u} & \cdots & \boldsymbol{0}_{n_u \times n_u} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_{n_u \times n_u} & \boldsymbol{0}_{n_u \times n_u} & \cdots & \boldsymbol{I}_{n_u} \end{bmatrix} \text{ and } \boldsymbol{E} = \begin{bmatrix} \boldsymbol{I}_{n_u} \\ \boldsymbol{0}_{n_u \times n_u} \\ \boldsymbol{0}_{n_u \times n_u} \\ \vdots \\ \boldsymbol{0}_{n_u \times n_u} \end{bmatrix}.$$

By combining (A.1) and (A.11), the cost function can be written in vector form as

$$J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \left\| \boldsymbol{Y}_{ref}(k) - \boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{\Upsilon}\boldsymbol{U}(k) \right\|^2_{\tilde{\boldsymbol{R}}} + \lambda_u \left\| \boldsymbol{S}\boldsymbol{U}(k) - \boldsymbol{E}\boldsymbol{u}(k-1) \right\|_2^2, \tag{A.12}$$

where the dimensions of the matrices are $\boldsymbol{\Gamma} \in \mathbb{R}^{n_y N_p \times n_x}$, $\boldsymbol{\Upsilon} \in \mathbb{R}^{n_y N_p \times n_u N_p}$, $\boldsymbol{S} \in \mathbb{R}^{n_u N_p \times n_u N_p}$, and $\boldsymbol{E} \in \mathbb{R}^{n_u N_p \times n_u}$.

Appendix B

Quadratic programming form of cost function

The derivation of the cost function in quadratic programming form follows that was presented in [1].

From Appendix A, the cost function is given in vector form as

$$J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \left\| \boldsymbol{Y}_{ref}(k) - \boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{\Upsilon}\boldsymbol{U}(k) \right\|_{\tilde{\boldsymbol{R}}}^2 + \lambda_u \left\| \boldsymbol{S}\boldsymbol{U}(k) - \boldsymbol{E}\boldsymbol{u}(k-1) \right\|_2^2. \tag{B.1}$$

Knowing that $\left\|\boldsymbol{\xi}\right\|_{\tilde{\boldsymbol{R}}}^2 = \boldsymbol{\xi}^T \tilde{\boldsymbol{R}} \boldsymbol{\xi}$, (B.1) can be written as

$$\begin{aligned} J = {} & (\boldsymbol{\Upsilon}\boldsymbol{U}(k))^T \tilde{\boldsymbol{R}} \boldsymbol{\Upsilon}\boldsymbol{U}(k) + \lambda_u (\boldsymbol{S}\boldsymbol{U}(k))^T \boldsymbol{S}\boldsymbol{U}(k) + (\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k)^T \tilde{\boldsymbol{R}} \boldsymbol{\Upsilon}\boldsymbol{U}(k) \\ & + (\boldsymbol{\Upsilon}\boldsymbol{U}(k))^T \tilde{\boldsymbol{R}} (\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k) - \lambda_u (\boldsymbol{E}\boldsymbol{u}(k-1))^T \boldsymbol{S}\boldsymbol{U}(k) - \lambda_u (\boldsymbol{S}\boldsymbol{U}(k))^T \boldsymbol{E}\boldsymbol{u}(k-1), \\ & + (\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k)^T \tilde{\boldsymbol{R}} ((\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k) + \lambda_u (\boldsymbol{E}\boldsymbol{u}(k-1))^T (\boldsymbol{E}\boldsymbol{u}(k-1)) \end{aligned} \tag{B.2}$$

where the argument of J is dropped for convenience. Since $\tilde{\boldsymbol{R}}$ is symmetric, (B.2) can be simplified to

$$J = \boldsymbol{U}^T(k) \boldsymbol{Q} \boldsymbol{U}(k) + 2\boldsymbol{\Theta}^T(k) \boldsymbol{U}(k) + \theta(k) \tag{B.3}$$

where the matrices and vectors defined as

$$\boldsymbol{Q} = \boldsymbol{\Upsilon}^T \tilde{\boldsymbol{R}} \boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^T \boldsymbol{S} \tag{B.4}$$

$$\boldsymbol{\Theta}(k) = ([\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k)]^T \tilde{\boldsymbol{R}} \boldsymbol{\Upsilon} - \lambda_u [\boldsymbol{E}\boldsymbol{u}(k-1)]^T \boldsymbol{S})^T \tag{B.5}$$

$$\theta(k) = \left\| \boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{ref}(k) \right\|_{\tilde{\boldsymbol{R}}}^2 + \lambda_u \left\| \boldsymbol{E}\boldsymbol{u}(k-1) \right\|_2^2. \tag{B.6}$$

Note that $\boldsymbol{Q} \in \mathbb{R}^{n_u N_p \times n_u N_p}$ is a symmetrical matrix.

Appendix C

State-space representation of NPC inverter with an RL load

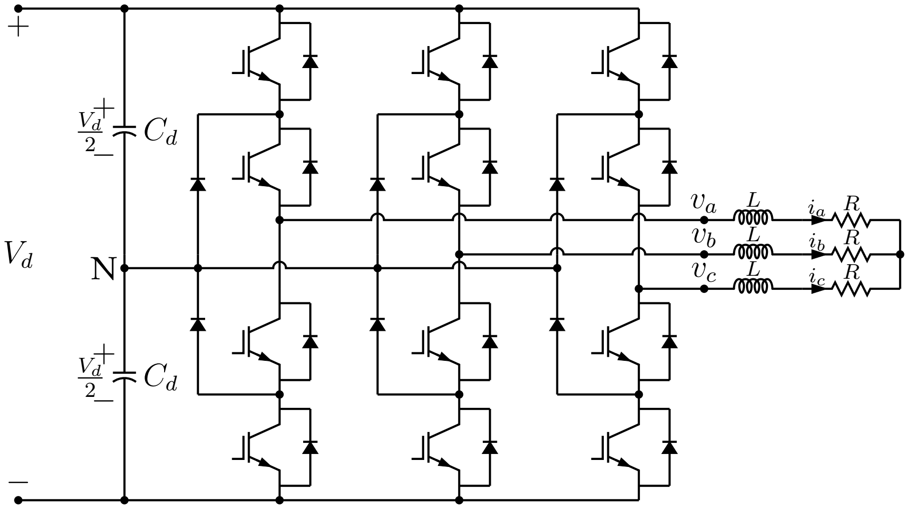


Figure C.1: NPC inverter with an RL load.

By analysing the circuit in Figure C.1, the following differential equations can be derived

$$v_a - v_b = L\frac{di_a}{dt} + Ri_a - L\frac{di_b}{dt} - Ri_b \tag{C.1a}$$

$$v_b - v_c = L\frac{di_b}{dt} + Ri_b - L\frac{di_c}{dt} - Ri_c, \tag{C.1b}$$

where the phase voltages $\boldsymbol{v}_{abc}$ are with respect to the neutral point N. The reduced Clarke transformation is defined as

$$\boldsymbol{\xi}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{\xi}_{abc}, \tag{C.2}$$

where

$$\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}. \tag{C.3}$$

From (C.2), v_α is given as

$$\begin{aligned} v_\alpha &= \frac{2}{3}(v_a - v_b - v_c) \\ &= \frac{2}{3}((v_a - v_b) + \frac{1}{2}(v_b - v_c)), \end{aligned} \tag{C.4}$$

and by inserting (C.1) and by applying (C.2), becomes

$$\begin{aligned} v_\alpha &= \frac{2}{3}(L(\frac{di_a}{dt} - \frac{1}{2}\frac{di_b}{dt} - \frac{1}{2}\frac{di_c}{dt}) + R(i_a - \frac{1}{2}i_b - \frac{1}{2}i_c)) \\ &= L\frac{di_\alpha}{dt} + Ri_\alpha. \end{aligned} \tag{C.5}$$

Similarly, from (C.2), v_β is given as

$$v_\beta = \frac{2}{3}(\frac{\sqrt{3}}{2}v_b - \frac{\sqrt{3}}{2}v_c), \tag{C.6}$$

and by inserting (C.1) and by applying (C.2), becomes

$$\begin{aligned} v_\beta &= \frac{2}{3}(\frac{\sqrt{3}}{2}(L(\frac{di_b}{dt} - \frac{di_c}{dt}) + R(i_b - i_c))) \\ &= L\frac{di_\beta}{dt} + Ri_\beta. \end{aligned} \tag{C.7}$$

With $\boldsymbol{v}_{\alpha\beta} = \frac{V_d}{2}\boldsymbol{K}\boldsymbol{u}_{abc}$, where $\boldsymbol{u}_{abc}$ denotes the three-phase switch positions, and from (C.5) and (C.7), the continuous-time state-space representation is given as

$$\dot{\boldsymbol{i}}_{\alpha\beta}(t) = \boldsymbol{F}\boldsymbol{i}_{\alpha\beta}(t) + \boldsymbol{G}\boldsymbol{u}_{abc}(t), \tag{C.8}$$

where

$$\boldsymbol{F} = -\frac{R}{L}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{C.9a}$$

$$\boldsymbol{G} = \frac{V_d}{2L}\boldsymbol{K}. \tag{C.9b}$$

Bibliography

[1] T. Geyer, *Model predictive control of high power converters and industrial drives*. Wiley, Nov 2016.

[2] A. K. Rathore, J. Holtz, and T. Boller, "Synchronous optimal pulsewidth modulation for low-switching-frequency control of medium-voltage multilevel inverters," *IEEE Transactions on Industrial Electronics*, vol. 57, pp. 2374–2381, July 2010.

[3] A. K. Rathore, J. Holtz, and T. Boller, "Generalized optimal pulsewidth modulation of multilevel inverters for low-switching-frequency control of medium-voltage high-power industrial ac drives," *IEEE Transactions on Industrial Electronics*, vol. 60, pp. 4215–4224, Oct 2013.

[4] W. Leonhard, *Control of Electrical Drives*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 3rd ed., 2001.

[5] D. W. Novotny and T. A. Lipo, *Vector control and dynamics of AC drives*. Oxford : Clarendon Press ; New York : Oxford University Press, repr. with corrections ed., 1998.

[6] M. Malinowski, M. P. Kazmierkowski, and A. M. Trzynadlowski, "A comparative study of control techniques for pwm rectifiers in ac adjustable speed drives," *IEEE Transactions on Power Electronics*, vol. 18, pp. 1390–1396, Nov 2003.

[7] N. Celanovic and D. Boroyevich, "A fast space-vector modulation algorithm for multilevel three-phase converters," *IEEE Transactions on Industry Applications*, vol. 37, pp. 637–641, Mar 2001.

[8] 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, pp. 820–827, Sept 1986.

[9] T. Noguchi, H. Tomiki, S. Kondo, and I. Takahashi, "Direct power control of pwm converter without power-source voltage sensors," *IEEE Transactions on Industry Applications*, vol. 34, pp. 473–479, May 1998.

[10] T. Geyer and D. E. Quevedo, "Multistep finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 29, pp. 6836–6846, Dec 2014.

[11] P. Cortes, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodriguez, "Predictive control in power electronics and drives," *IEEE Transactions on Industrial Electronics*, vol. 55, pp. 4312–4324, Dec 2008.

[12] J. Rodriguez, M. P. Kazmierkowski, J. R. Espinoza, P. Zanchetta, H. Abu-Rub, H. A. Young, and C. A. Rojas, "State of the art of finite control set model predictive control in power electronics," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 1003–1016, May 2013.

[13] T. Geyer, *Low Complexity Model Predictive Control in Power Electronics and Power Systems*. PhD thesis, Mar. 2005.

[14] P. Karamanakos, *Model Predictive Control Strategies for Power Electronics Converters and AC Drives*. PhD thesis, Mar. 2013.

[15] T. Geyer and D. E. Quevedo, "Performance of multistep finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 30, pp. 1633–1644, March 2015.

[16] B. Stellato, T. Geyer, and P. J. Goulart, "High-speed finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 32, pp. 4007–4020, May 2017.

[17] A. Nabae, I. Takahashi, and H. Akagi, "A new neutral-point-clamped pwm inverter," *IEEE Transactions on Industry Applications*, vol. IA-17, pp. 518–523, Sept 1981.

[18] J. B. Rawlings and D. Q. Mayne, *Model predictive control: theory and design*. Nob Hill Publishing, 2009.

[19] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733 – 764, 2003.

[20] F. Allgöwer and A. Zheng, eds., *Nonlinear model predictive control*. Springer Basel, 2000.

[21] R. D. Middlebrook and S. Cuk, "A general unified approach to modelling switching-converter power stages," in *1976 IEEE Power Electronics Specialists Conference*, pp. 18–34, June 1976.

[22] F. Borrelli, *Constrained Optimal Control for Hybrid Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[23] M. Senesky, G. Eirea, and T. J. Koo, "Hybrid modelling and control of power electronics," in *Proceedings of the 6th International Conference on Hybrid Systems: Computation and Control*, HSCC'03, (Berlin, Heidelberg), pp. 450–465, Springer-Verlag, 2003.

[24] E. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Transactions on Automatic Control*, vol. 26, pp. 346–358, Apr 1981.

[25] D. E. Quevedo, R. P. Aguilera, and T. Geyer, *Predictive Control in Power Electronics and Drives: Basic Concepts, Theory, and Methods*. Springer International Publishing, 2014.

[26] J. C. Nash, *Compact Numerical Methods for Computers Linear Algebra and Function Minimisation*. Taylor & Francis, 2 ed., 1990.

[27] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, pp. 2201–2214, Aug 2002.

[28] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. New York, NY, USA: Cambridge University Press, 1st ed., 2009.

[29] L. Babai, "On lovaśz' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, pp. 1–13, Jan 1986.

[30] D. G. Holmes and T. A. Lipo, *Pulse width modulation for power converters: principles and practice*. Hoboken, NJ John Wiley, 2003.

[31] H. W. van der Broeck, H. C. Skudelny, and G. V. Stanke, "Analysis and realization of a pulsewidth modulator based on voltage space vectors," *IEEE Transactions on Industry Applications*, vol. 24, pp. 142–150, Jan 1988.

[32] B. P. McGrath, D. G. Holmes, and T. Lipo, "Optimized space vector switching sequences for multilevel inverters," *IEEE Transactions on Power Electronics*, vol. 18, pp. 1293–1301, Nov 2003.

[33] "Intel FPGA Devices." `https://www.intel.com/content/www/us/en/fpga/devices.html`. Accessed: 12/07/2017.

[34] P. Karamanakos, T. Geyer, and R. Aguilera, "Computationally efficient long-horizon direct model predictive control for transient operation," in *Proceedings of the IEEE Energy Conversion Congress and Exposition*, Oct 2017.

[35] A. Ayad, P. Karamanakos, and R. Kennel, "Direct model predictive current control strategy of quasi-z-source inverters," *IEEE Transactions on Power Electronics*, vol. 32, pp. 5786–5801, July 2017.

[36] T. Geyer, P. Karamanakos, and R. Kennel, "On the benefit of long-horizon direct model predictive control for drives with lc filters," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 3520–3527, Sept 2014.

[37] P. Cortes, J. Rodriguez, D. E. Quevedo, and C. Silva, "Predictive current control strategy with imposed load current spectrum," *IEEE Transactions on Power Electronics*, vol. 23, pp. 612–618, March 2008.

[38] R. P. Aguilera, P. Acuña, P. Lezana, G. Konstantinou, B. Wu, S. Bernet, and V. G. Agelidis, "Selective harmonic elimination model predictive control for multilevel power converters," *IEEE Transactions on Power Electronics*, vol. 32, pp. 2416–2426, March 2017.

[39] K. C. Pohlmann, *Principles of Digital Audio*. McGraw-Hill Professional, 4th ed., 2000.

[40] C. D. Townsend, G. Mirzaeva, and G. C. Goodwin, "Short-horizon model predictive modulation of three-phase voltage source inverters," *IEEE Transactions on Industrial Electronics*, vol. PP, no. 99, pp. 1–1, 2017.